Modeling the latent space of variational autoencoders

Clément Chadebec

Université Paris Cité
École doctorale de Sciences Mathématiques de Paris Centre (ED 386)
Equipe INRIA - Health data- and model- driven Knowledge Acquisition

Modeling the Latent Space of Variational Autoencoders

Thèse de Doctorat en Mathématiques Appliquées

Présentée par Clément Chadebec

Dirigée par Stéphanie Allassonnière

Soutenue publiquement le 29 Juin 2023

Arnaud Doucet	Professeur, Oxford University	Rapporteur
Alexandre Gramfort	Research Scientist, Meta	Rapporteur
Jean-Philippe Vert	Professeur, Mines ParisTech	Examinateur
Antoine Chambaz	Professeur, Université Paris Cité	Examinateur
Ninon Burgos	Chargée de recherche, INRIA (ARAMIS)	Examinateur
Stéphanie Allassonnière	Professeure, Université Paris Cité	Directrice de thèse

Contents

List of Figures

List of Tables

Résumé

Cette thèse de doctorat s'intéresse à la modélisation et à la structuration de l'espace latent des modèles de type auto-encodeur variationnel. Ces derniers sont des modèles probabilistes dits génératifs car ils ont pour objectif d'être capables de générer de nouvelles données synthétiques à partir d'un ensemble d'entraînement. Une des particularités de ces modèles est l'existence d'un espace de plus faible dimension que celui dans lequel les données d'entrée sont considérées et que l'on appelle espace latent. La compréhension et l'étude de cet espace restent lacunaires et cette thèse a pour objective d'essayer d'apporter une meilleur compréhension de sa structure et de la façon de l'exploiter pour des tâches annexes. À travers l'utilisation d'outils de géométrie Riemannienne, nous proposerons différentes modélisations de l'espace latent des auto-encodeurs variationnels qui nous conduirons premièrement à construire un nouvel estimateur de la log-vraisemblance du modèle. L'étude de cette géométrie nous mènera également à considérer et à proposer de nouvelles méthodes d'échantillonnage de nouvelles données qui s'avéreront très adaptées à des contextes à faible nombre de données, connus pour être très limitant pour ces types de modèles. Une tâche connue sous le nom d'*augmentation de données* et consistant en la création de données synthétiques pour enrichir les bases de données existantes nous occupera tout particulièrement et sera l'une des applications majeures des modèles que nous développerons. Par exemple, nous appliquerons notre méthode pour améliorer la performance de modèles de classification sur des données réelles et complexes telles que des IRMs tri-dimensionnelles de cerveaux. Au cours de cette thèse, nous développerons également plusieurs outils informatiques permettant un usage facilité et ouvert au plus grand nombre de tels modèles. Enfin, nous nous intéresserons également aux données dites longitudinales, c'est à dire des données qui partagent une dépendance temporelle forte tel que le suivi de patients, mais dont le nombre d'observations par entité reste faible (typiquement moins de 10 observations par entité). En particulier, nous introduirons des modèles capables de générer des trajectoires synthétiques complètes.

Mots clés: Inférence Variationnelle, Modèles Génératifs, Géometrie Riemannanienne, Autoencodeurs Variationnels

Abstract

This PhD thesis focuses on the modeling and structuring of the latent space of Variational Autoencoder models. The latter are probabilistic models called generative models since their objective is to be able to generate new synthetic data from a set of training data. One of the particularities of these models is the existence of a space of lower dimension than the one in which the input data are considered called a latent space. The understanding and the study of this space remain incomplete, and this thesis aims at bringing a better understanding of its structure and how to exploit it for downstream tasks. Through the use of Riemannian geometry tools, I will propose different modelings of the latent space of the variational autoencoder, which will first lead us to build a new estimator of the log-likelihood of the model. The study of this geometry will also lead us to consider and propose new sampling methods to generate new data that will prove to be well suited to contexts with a small number of data which are known to be very challenging for these models. A task known as *data augmentation* and consisting in the creation of synthetic data to enrich existing databases will, in particular, be one of the major applications of the models I will develop. For example, I will apply our methods to improve the performance of classification models on real and complex data such as three-dimensional brain MRIs. During this thesis, I will also develop several software allowing easy and open use of such models. Finally, I will also be interested in the modeling of longitudinal data, *i.e.* data that share a temporal dependency, such as patient follow-up but whose number of observations per entity remains low (typically less than 10 observations per entity). In particular, I will introduce models capable of generating complete synthetic trajectories.

Key words: Variational Inference, Generative Models, Riemannian Geometry, Variational Autoencoders

Remerciements

J'aimerais remercier en premier lieu Stéphanie, ma directrice de thèse, sans qui ces travaux de thèse n'auraient jamais pu aboutir. Je te remercie sincèrement d'avoir accepté de me prendre comme doctorant. Ta bienveillance, ton enthousiasme et ton optimisme à chaque nouveau projet ont rendu ces trois années très stimulantes et agréables. Je te remercie également de m'avoir permis de découvrir l'univers de la recherche académique et d'avoir toujours été motrice lorsque j'étais à la recherche de personnes avec qui collaborer. Cela a vraiment été un grand plaisir de travailler à tes côtés sur des sujets toujours aussi passionants et ambitieux.

Je souhaiterais également remercier l'équipe HeKA de m'avoir accueilli ces trois ans durant au sein d'une équipe pluridisciplinaire aux profils aussi variés qu'enrichissants. Je remercie l'Université Paris Cité et le Centre de Recherche des Cordeliers pour m'avoir permis de débuter ma thèse dans les meilleures conditions. En particulier, je remercie Jean-Marc et Nassim pour leur disponibilité et la mise en place du serveur de calcul qui m'a été d'une aide plus que précieuse. Je remercie également GENCI pour la mise à disposition des chercheurs d'un environnement de travail aussi agréable.

Merci à la team Stéphanie: Vianney, Clément, Solange, Fleur, Pierre, Louis, Agathe, pour ces discussions sympathiques et ces bons moments que nous avons pu partager. Je vous souhaite à tous de vous épanouir pleinement pour la suite que ce soit en entreprise, en post-doc ou pour votre thèse! J'aimerais en particulier remercier Louis et Agathe avec qui j'ai eu la chance de collaborer directement au cours de ma thèse. Je vous souhaite à tous les deux le meilleur pour vos thèses qui s'annoncent, j'en suis sûr, excellentes.

Un grand merci à Elina et Ninon avec qui j'ai eu la chance et le grand plaisir de collaborer pour mon premier projet de thèse. J'ai vraiment adoré travailler en votre compagnie et vous souhaite également le meilleur pour la suite. J'aimerais également remercier Evi, Maureen et Josien de l'université de Tübingen pour ce projet que nous avons pu conduire ensemble sur les données longitudinales. Cela a été une expérience très enrichissante pour moi et j'espère sincèrement, Evi, que tu te plairas toujours autant dans ton projet de thèse.

Je souhaite également remercier sincèrement Arnaud Doucet et Alexandre Gramfort qui ont accepté de rapporter ma thèse ainsi que pour leurs précieux retours. Merci énormément à Ninon Burgos, Antoine Chambaz et Jean-Philippe Vert. Je suis ravi et honoré de pouvoir vous compter parmi les membres de mon jury de thèse.

À titre plus personnel, je souhaite remercier ma famille et mes amis qui m'ont toujours soutenu tout au long de ces trois ans de thèse et ont toujours été curieux de comprendre ce sur quoi je travaillais. Un grand merci en particulier à Galeje & Co avec qui j'ai pu partager mes doutes et réflexions et qui ont toujours eu une oreille attentive et objective. Enfin, j'aimerais te remercier, Lucile, pour ton soutien sans faille, ta patience à toute épreuve et de m'avoir toujours épaulé dans les bons comme dans les moments plus difficiles. Je n'en serais pas là sans toi.

Résumé Substantiel en Français

Cette thèse de doctorat s'intéresse à la modélisation et à la structuration de l'espace latent des modèles de type Autoencodeur Variationels (VAE). Ces derniers sont des modèles probabilistes dits génératifs car ils ont pour objectif d'être capables de générer de nouvelles données synthétiques à partir d'un ensemble d'entraînement. Introduit par (Kingma and Welling, 2014; Rezende et al., 2014), un VAE suppose que les données d'observation sont générées selon le schéma suivant :

$$\begin{cases} z \sim p(z)\,, \\ x \sim p_\theta(x|z)\,. \end{cases} \tag{1}$$

p est une distribution *a-priori* sur les variables latentes souvent prise comme une distribution simple (*e.g.* gaussienne standard) et $p_\theta(x|z)$ est une distribution conditionnelle des données étant donné les variables latentes. L'idée principale de ces modèles repose sur l'inférence variationelle amortie et consiste en l'introduction d'une distribution paramétrique q_ϕ visant à approximer la distribution *a-posteriori* $p_\theta(z|x)$ inconnue. Un VAE vise à maximiser l'ELBO introduite en Eq. (5) qui constitue une borne inférieure de la log-vraissemblance. Une fois le modèle entraîné, il est possible de générer de nouveaux échantillons en utilisant la distribution *a-priori* p et en fournissant les échantillons au *décodeur* $p_\theta(x|z)$ selon le modèle génératif donné en Eq. (1).

Le VAE tel que proposé dans (Kingma and Welling, 2014) suppose que les données x vivent dans un espace euclidien $\mathcal{X} = \mathbb{R}^d$ et la distribution variationnelle *a-posteriori* $q_\phi(z|x)$ est modélisée par une distribution gaussienne multivariée avec une covariance diagonale (*i.e.* $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$) dont les paramètres sont donnés par des réseaux de neurones. Par conséquent, l'espace latent est vu comme un espace euclidien *i.e.* $\mathcal{Z} = \mathbb{R}^d$. La distribution *a-priori* est choisie comme une simple distribution gaussienne standard $p(z) = \mathcal{N}(0, I_d)$. Pendant l'apprentissage, l'ELBO est estimée à l'aide d'un seul échantillon provenant de la distribution variationnelle $z \sim q_\phi(z|x)$ pour chaque donnée $x \in \mathcal{X}$.

Partant de la définition du VAE telle que présentée ci-dessus, nous avons proposé au cours de cette thèse différents axes d'amélioration du modèle en considérant entre-autre une modélisation de l'espace latent incluant des considérations géométriques.

Premièrement, nous avons introduit un modèle qui étend le cadre de l'autoencodeur variationnel hamiltonien proposé par (Salimans et al., 2015; Caterini et al., 2018) à des espaces latents vus comme des variétés riemanniennes. L'une des principales idées du modèle proposé est d'effectuer des étapes d'échantillonnage de type Monte Carlo Markov Chain supplémentaires à partir de variables échantillonnées de l'approximation variationnelle q_ϕ en utilisant des opérateurs de transition inspirés de la dynamique hamiltonienne mais bénéficiant également de la structure Riemannienne supposée de l'espace latent. En introduisant les variables de moment $v \in \mathbb{R}^d$ et en considérant une distribution variationnelle augmentée sur l'espace étendu $(z, v) \in \mathcal{Z} \times \mathbb{R}^d$, nous dérivons un estimateur de la vraisemblance marginale $p_\theta(x)$ comme suit :

$$\hat{p}_\theta(x) = \frac{p_\theta(x, z_K, v_K)}{q_\phi(z_K, v_K|x)} = \frac{p_\theta(x|z_K)p(v_K|z_K)p(z_K)}{q_\phi(z_0|x)p(v_0|z_0)\prod_{k=1}^K |\det \mathbf{J}_{g^k}|^{-1}}\,,$$

où $\mathbf{J}_{g^k}$ est la jacobienne de la k-ième étape de l'intégrateur *leapfrog* généralisé (Leimkuhler and Reich, 2004; Girolami et al., 2009; Girolami and Calderhead, 2011) et $p(v|z) = \mathcal{N}(0, \mathbf{G}(z))$ est une dis-

tribution de proposition pour les moments où $\mathbf{G}(z)$ est la valeur de la métrique Riemannienne en z. Puisque la métrique Riemannienne est inconnue, nous proposons de la paramétrer et de l'apprendre directement à partir des données à l'aide de réseaux de neurones. Ensuite, nous proposons un schéma d'échantillonnage exploitant la géométrie de l'espace latent appris par le modèle proposé et consistant à échantillonner près des chemins géodésiques. Ce mécanisme d'échantillonnage a effectivement démontré des performances de génération prometteuses, notamment dans un contexte de faible taille d'échantillon, où il surpasse la génération à partir de la distribution *a-priori* et d'autres méthodes d'estimation *post*-entraînement. Enfin, la méthode est utilisée pour effectuer de l'augmentation de données pour des tâches de classification et est validée par une vaste étude expérimentale où la robustesse aux données et aux classifieurs est testée. En particulier, la méthode a permis d'augmenter les performances obtenues par un classifieur *état-de-l'art* entraîné à détecter la maladie d'Alzheimer sur des IRM 3D. Cette première étude nous a conduit à nous interroger sur les façons les plus pertinentes d'échantillonner depuis l'espace latent d'un VAE entrainé. Par exemple, elle nous a conduit à présenter une nouvelle méthode d'échantillonnage à partir de variétés Riemanniennes inconnues *géodésiquement-complètes*. L'idée principale est de créer un algorithme de type marche aléatoire qui découvre la variété le long des chemins géodésiques. Partant d'un point z appartenant à la variété Riemannienne $\mathcal{M}$, un échantillon est obtenu en échantillonnant d'abord un vecteur $v \sim \mathcal{N}(0, \Sigma)$ dans l'espace tangent T_z à z et en appliquant la carte exponentielle Riemannienne pour ramener le vecteur sur la variété. Cela revient à échantillonner à partir de la distribution *wrapped* normale définie sur la variété Riemannienne $\mathcal{M}$. L'échantillon proposé est ensuite accepté selon un certain rapport de probabilité. Cette méthode est ensuite appliquée au VAE Riemannien introduit dans l'étude précédente.

L'introduction de considérations géometriques dans l'espace latent des VAEs nous a ensuite conduit à nous interroger sur la capacité du VAE pris dans sa forme la plus simple à rendre compte naturellement d'une certaine géométrie dans son espace latent. Ainsi, nous avons proposé une nouvelle perspective sur le cadre de l'autoencodeur variationnel en adoptant un point de vue entièrement géométrique. En bref, nous partons du constat qu'une distribution gaussienne multivariée $\mathcal{N}(\mu, \Sigma)$ n'est qu'un cas spécifique de la distribution gaussienne Riemannienne $\mathcal{N}^{\text{riem}}(z|\mu, \sigma) = \frac{1}{C} \exp\left(-\frac{\text{dist}_{\mathbf{G}}(z,\mu)^2}{2\sigma}\right)$ définie sur la variété Riemannienne $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ où $\sigma = 1$ et la métrique Riemannienne $\mathbf{G}$ est la métrique constante $\mathbf{G}(z) = \Sigma^{-1}$. Par conséquent, nous proposons d'interpréter la distribution variationnelle $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ non plus comme une gaussienne multivariée, mais comme une distribution gaussienne *Riemannienne*. Pour toute donnée d'apprentissage x_i, $\Sigma_\phi(x_i)^{-1}$ est vu comme une approximation de la métrique Riemannienne inconnue $\mathbf{G}$ évaluée au point $\mu_\phi(x_i)$ et définissant un espace latent Riemannien considéré comme étant la variété Riemannienne $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$. Pendant l'apprentissage, cette métrique est supposée localement constante à proximité du point $\mu_\phi(x_i)$ (*i.e.* $\mathbf{G}(z) \approx \Sigma(x_i)^{-1}$ pour z proche de $\mu_\phi(x_i)$). Par conséquent, dans un tel cas, l'échantillonnage de la distribution Riemannienne $q_\phi(z|x)$ peut être approximée par l'échantillonnage de la distribution gaussienne multivariée $\mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ comme lors de l'apprentissage du VAE classique. Enfin, à la fin de l'apprentissage, ces approximations $(\Sigma_i)_i^{-1}$ de la métrique Riemannienne $\mathbf{G}$ sont assemblées pour construire une métrique Riemannienne continue

sur l'ensemble de l'espace latent.

$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d \,,$$
$$\omega_i(z) = \exp\left(- \frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2} \right),$$

où $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i)))$ est la distance Riemannienne entre z et $\mu(x_i)$ par rapport à la métrique localement constante $\mathbf{G}(\mu(x_i)) = \mathbf{\Sigma}^{-1}(x_i)$. Nous avons également proposé un nouveau mécanisme d'échantillonnage qui consiste à échantillonner à partir de la distribution uniforme Riemannienne intrinsèquement définie sur $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ comme suit :

$$\mathcal{U}_{\text{Riem}}(z) = \frac{\sqrt{\det \mathbf{G}(z)}}{\int_{\mathbb{R}^d} \sqrt{\det \mathbf{G}(z) dz}} \,.$$

Nous montrons empiriquement qu'un tel schéma d'échantillonnage permet d'améliorer la capacité générative du VAE classique (dont le processus d'apprentissage reste inchangé) qui peut désormais surpasser des méthodes plus avancées et potentiellement plus coûteuses proposées par la suite.

Au cours de cette thèse nous avons également développé plusieurs outils informatiques permettant un usage facilité et ouvert au plus grand nombre de tels modèles. L'un d'eux est la librairie Pythae qui est une librairie *open-source* Python offrant à la fois une *implémentation unifiée* et un cadre dédié à l'utilisation des modèles génératifs de type autoencodeur. Au moment où j'écris cette thèse, cette bibliothèque implémente et unifie 25 implémentations de modèles de type autoencodeurs (variationnels) et a suscité un certain intérêt dans la communauté avec 1.2k+ étoiles sur github, plus de 10,000 téléchargements, 8 contributeurs externes et a également été présentée à la conférence PyTorch 2022 à la Nouvelle-Orléans. L'objectif principal de cette bibliothèque Python est d'abaisser la barrière d'entrée pour l'utilisation de ces modèles tout en assurant qu'ils sont suffisamment flexibles pour s'adapter à divers cas d'utilisation et types de données. La plupart des implémentations fournies ont été capables de reproduire les principaux résultats des articles originaux rendant les implémentations fiables. De plus, cette bibliothèque est unit-testée et répond aux critères des logiciels libres. Elle est disponible sous la licence Apache2.0 et peut être installée à partir de *pypi* ou *conda*. D'autre part, nous avons proposé un cas d'usage de Pythae sous la forme d'un benchmak de 19 modèles comparés dans les mêmes conditions et pour 5 tâches différentes telles que la reconstruction, la génération, la classification, le clustering et l'interpolation d'images.

Une partie substantielle de la thèse a également été consacrée à l'appplication et le développement de modèles adaptés à des données structurées telles que les données longitudinales. Ces dernières ne sont plus indépendantes et identiquement distribuées comme le suppose le cadre initial du VAE et nécessitent donc une nouvelle modélisation.

Nous avons premièrement proposé un modèle génératif pour les données d'images longitudinales agissant directement dans l'espace latent d'un VAE entraîné. Cette proposition est motivée par l'observation que, de part la structure intrinsèque de leur espace latent, les autoencodeurs variationnels rendent naturellement compte d'une trajectoire (qui n'est pas obligatoirement une ligne droite) dans l'espace latent lorsqu'ils sont entraînés avec de telles données. Nous proposons donc d'apprendre une fonction qui transforme ces trajectoires en une trajectoire linéaire définie par des

paramètres de vitesse, de retard et d'espace qui sont appris directement à partir des données dans un esprit similaire à celui de (Louis et al., 2019).

$$l_i(t) = \exp(\eta_i)(t - \tau_i) \cdot \mathbf{e_1} + \sum_{k=2}^{d} \lambda_i^k \cdot \mathbf{e_k}\,,$$

où η_i est un paramètre de vitesse, τ_i est un retard, et $\lambda_i = (\lambda_i^k)_{2\leq k\leq d}$ sont des paramètres spatiaux. Contrairement à (Louis et al., 2019), nous adoptons une approche entièrement variationnelle pour rendre le modèle génératif de manière similaire à (Kingma and Welling, 2014). Étant donné P séquences d'entrée $(x_i)_{i\in[1,P]}$ ayant chacune t_i observations telles que $x_i = (x_i^0, \ldots, x_i^{t_i})$, un premier autoencodeur variationnel est entraîné sur toutes les données sans tenir compte de leur structure temporelle. Cette étape permet d'apprendre une représentation en dimension réduite des données d'entrée. Ensuite, un second modèle est entraîné pour mettre en correspondance les trajectoires des données d'entrée $(x_i)_{i\in[1,P]}$ dans l'espace latent du VAE avec une trajectoire linéaire.

Adoptant un point de vue différent, nous avons également proposé un nouveau modèle génératif à variables latentes capable de gérer des données longitudinales de grande dimension. Étant donné une entité $i \in \{1, \ldots, P\}$ et une séquence d'observations $(x_0^i, \ldots, x_{t_i}^i)$, nous supposons que pour chaque x_j^i où $j \in \{0, \ldots, t_i\}$, il existe une variable latente associée $z_j^i \in \mathcal{Z} = \mathbb{R}^d$ impliquée dans le processus génératif de l'observation x_j^i telle que $x_j^i \sim p_\theta(x_j^i|z_j^i)$. Puisque les observations au sein d'une séquence ne sont plus indépendantes (cadre longitudinal), le cadre du VAE classique ne peut pas être appliqué car la vraisemblance ne se factorise plus entre les observations. Par conséquent, nous proposons de modéliser la dépendance temporelle entre les observations d'une séquence en utilisant des flux normalisant f_j sur les variables latentes.

$$z_0^i \sim p(z_0^i),\ z_1^i = f_1(z_0^i),\ \ldots\ , z_{t_i}^i = f_{t_i}(z_{t_i-1}^i)\,, \tag{2}$$

où p est une distribution *a-priori* simple sur z_0^i (*e.g.* gaussienne standard) et f_j sont des flux normalisant pour tout $j \in \{1, \ldots, t_i\}$. Puisque les f_j sont inversibles, on peut voir que nous avons également accès à une distribution *a-priori* connue pour z_j^i en utilisant le théorème de changement de variable. En supposant que les observations dans une séquence sont indépendantes lorsqu'elles sont conditionnées par rapport aux variables latentes associées et en notant qu'à z_j^i donné nous pouvons retrouver la séquence complète des variables latentes $(z_0^i, \ldots, z_{t_i}^i)$ en utilisant Eq. (2), nous pouvons écrire la vraisemblance jointe comme suit :

$$p_\theta(x_0^i, \cdots, x_{t_i}^i) = \int_{\mathcal{Z}} p_\theta(x_0^i, \cdots, x_{t_i}^i|z_j^i)p(z_j^i)\mathrm{d}z_j^i = \int_{\mathcal{Z}} \prod_{l=0}^{t_i} p_\theta(x_l^i|z_l^i)p(z_j^i)\mathrm{d}z_j^i\,.$$

Comme cette intégrale est la plupart du temps difficile à calculer, nous proposons de recourir à l'inférence variationnelle amortie et introduisons une distribution paramétrique $q_\phi(z_j^i|x_j^i)$ qui permet d'obtenir une estimation non biaisée de la vraisemblance jointe et ainsi d'obtenir une ELBO.

$$\log p_\theta(x_0^i, \cdots, x_{t_i}^i) \geq \mathbb{E}_{q_\phi}\left[\log \prod_{l=0}^{t_i} p_\theta(x_l^i|z_l^i)\right] - \mathrm{KL}(q_\phi(z_j^i|x_j^i)|p(z_j^i))\,.$$

L'apprentissage est effectué en choisissant aléatoirement un j dans $[0, t_i]$, en échantillonnant $z_j^i \sim q_\phi(z_j^i|x_j^i)$, en récupérant une séquence reconstruite $(\widehat{x}_0^i, \ldots, \widehat{x}_{t_i}^i)$ en utilisant Eq. (2) et en optimisant

l'ELBO. Une fois que le modèle est entraîné, il est possible de générer des séquences entièrement synthétiques en commençant par échantillonner $z_0 \sim p(z_0)$ et en appliquant les flux selon Eq. (2). Nous discutons également la manière de traiter les données manquantes au moment de l'entraînement et de l'inférence et comment générer des séquences conditionnées par une ou plusieurs observation(s) dans une séquence d'entrée.

Introduction

0.1 Context

The dimension of an input data is defined as the number of features that can vary independently. For instance, the dimension of an image is assumed to be given by the total number of pixels since, theoretically, each of the pixels can take a value comprised in $[0, 255]$ regardless of its neighbors. We refer to *high dimensional* data when the dimension of this input data is of the same order of magnitude or larger than that number of observations. High dimensional data are more than common in many application fields where one of the most representative is medicine. This type of data is pretty hard to apprehend and requires recourse to dimensionality reduction tools to extract meaningful and useful structural information needed to analyze them (Gillies et al., 2016; Lambin et al., 2017). In medical imaging, such dimensionality reduction may be needed to extract relevant features to better characterize and understand a given population or the evolution of a given disease from complex data (*e.g.* 3-dimensional Magnetic Resonance Imaging (MRI) images) having potentially thousands or millions of voxels. This task, known as radiomics, provides a lower dimensional representation of the data that can then be used for downstream tasks such as disease progression monitoring, treatment response prediction or diagnosis prediction. Another challenging aspect of high dimensional data relies in the number of available observations. It is indeed not rare to only have access to a (very) limited number of observations making the analysis even more challenging (*e.g.* rare disease) (Button et al., 2013; Turner et al., 2018). The recent rise in performance of Deep Latent Variable Generative Models (DLGM) such as generative adversarial networks (GAN) (Goodfellow et al., 2014) or variational autoencoders (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) has made them very attractive models to tackle both challenges at the same time. They indeed provide a flexible framework allowing for a lower dimensional representation of the input data through the latent variables while being able to generate new synthetic samples. VAEs are a specific type of DLGM model relying on an autoencoding structure that gives rise to the consideration of a lower dimensional structured space called the latent space. This PhD thesis aims at providing a better understanding of the underlying structure of such a space through different modeling.

0.2 Deep Latent Variable Models

Let $x \in \mathcal{X}$ be a set of data. A Latent Generative Model is a generative model that assumes that there exist latent variables $z \in \mathcal{Z}$ (*i.e.* that are not directly observed from our dataset) involved in the generation process of the observations. $\mathcal{Z}$, the space in which the latent variables live is referred to as the *latent space*. These models typically assume that the observation data are generated according to the following scheme:

$$\begin{cases} z \sim p(z)\,, \\ x \sim p_\theta(x|z)\,. \end{cases} \tag{3}$$

p is a prior distribution over the latent variables often taken as a simple distribution (*e.g.* standard Gaussian), and $p_\theta(x|z)$ is a conditional distribution of the data given the latent variables. A *Deep Latent Generative Model* (DLGM) further assumes that $p_\theta(x|z)$ is taken as a parametric distribution, the

parameters of which are estimated using deep neural networks. In practice, the distribution $p_\theta(x|z)$ is chosen depending on the modeling of the input data but is often taken as a simple distribution (*e.g* fixed variance Gaussian for RGB images, factorized Bernoulli for binary data...). Let us consider the following example extracted from (Kingma and Welling, 2014; Kingma et al., 2019)

Example 1 *Let assume that $\mathcal{X} = \mathbb{R}^D$ is a set of binary images such as the MNIST (LeCun, 1998) dataset representing binary images of handwritten digits. A DLGM can be defined for these data as follows*

$$p(z) = \mathcal{N}(0, I_d)\,,$$
$$p_\theta(x|z) = \prod_{i=1}^{D} p_\theta(x_i|z) = \prod_{i=1}^{D} \mathcal{B}(x_i; p_i)\,,$$

where $p_i = D_\theta(z)$ with D_θ a parametric function parameterized with neural networks that outputs the parameters p_i of the distribution $p_\theta(x|z)$ modeled by a factorized Bernoulli distribution.

The modeling proposed in Eq. (3) then allows to derive the marginal distribution of the data as follows

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p(z)dz\,, \tag{4}$$

A DLGM typically learns the set of parameters $\theta \in \Theta$ by maximizing the likelihood of the data (*i.e.* the above equation). However, since the integral in Eq. (4) is taken over the entire latent space, it is most of the time intractable. As a consequence, the posterior distribution $p_\theta(z|x)$ neither admits a closed-form expression:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p(z)}{\int_{\mathcal{Z}} p_\theta(x|z)p(z)dz}\,.$$

In particular, this makes the application of Bayesian inference impossible. In such a case, one can rely on samples from the posterior distribution using Markov Chain Monte Carlo methods that aim at sampling from it but can reveal quite costly or Variational Inference that aims at approximating the posterior distribution using a family of parametrized distributions $q_\phi(z|x)$.

0.3 Variational Inference

Given observations $x \in \mathcal{X}$ and associated latent variables $z \in \mathcal{Z}$ with joint distribution $p(x, z)$, variational inference is a method that aims at approximating an untractable conditional distribution $p(z|x)$ of the latent variables given the observations using a family of parametrized distributions $\mathcal{Q} = \{q(z)\}$ (Blei et al., 2017). The distributions q are referred to as *variational distributions*. The idea is to find the *optimal* distribution $q^* \in \mathcal{Q}$ that minimizes the Kullback-Leibler (KL) divergence between the approximate posterior and the true one *i.e.*

$$\min_{q\in\mathcal{Q}} \mathrm{KL}(q(z)||p(z|x)) = \min_{q\in\mathcal{Q}} \int \log\left(\frac{q(z)}{p(x|z)}\right) q(z)dz\,.$$

However, this objective is most of the time untractable since $p(z|x)$ is unknown and so a surrogate objective obtained using Jensen's inequality is introduced and optimized instead (Jordan et al., 1999):

$$\begin{aligned}
\log p(x) &= \log \int_{\mathcal{Z}} p(x,z)dz \\
&= \log \int_{\mathcal{Z}} \left[\frac{p(x,z)}{q(z)}\right] q(z)dz \\
&= \log \mathbb{E}_q \left[\frac{p(x,z)}{q(z)}\right], \\
&\geq \mathbb{E}_q \log \left[\frac{p(x,z)}{q(z)}\right], \\
&\geq \log p(x) - \mathrm{KL}(q(z)||p(z|x)) \coloneqq \mathcal{L}(\theta,\phi,x)\,.
\end{aligned} \tag{5}$$

The right-hand side of the equation $\mathcal{L}(\theta,\phi,x)$ is called the Evidence Lower BOund (ELBO) and one may notice that the difference between the left-hand side of the equation and the ELBO gives $\mathrm{KL}(q(z)||p(z|x))$. Hence, maximizing the ELBO amounts to minimizing the KL, and so the ELBO is used as the objective for the variational approximation.

0.4 Amortized Variational Inference

A drawback of Variational Inference in its current shape is that given a set of data $(x_1,\ldots,x_n)$ an *optimal* variational posterior q^* needs to be computed for each posterior $p(z|x_i)$. This requires a *per-datapoint* optimization loop that can quickly become prohibitively expensive for large datasets. Such limitation can be addressed using the idea of Amortization. The main idea behind *Amortized Variational Inference* (AVI) is to consider variational distributions q that are parametrized with a common set of parameters ϕ that are shared across the data. The optimization is now no longer performed on the distributions but on the parameters ϕ using the same objective

$$\min_{\phi} \mathrm{KL}(q_\phi(z)||p(z|x))\,.$$

Example 2 *Let us assume that we are given a set of data $(x_1,\ldots,x_n)$ and associated latent variables $(z_1,\ldots,z_n)$. We place ourselves in the context where the posterior distributions $p(z_i|x_i)$ with $i \in \{1,\ldots,n\}$ are also unknown. A simple idea would consist in approximating these posterior distributions using Variational Inference. Hence, one can for instance consider Gaussian distributions $q(z_i|x_i) = \mathcal{N}(z_i;\mu_i,\Sigma_i)$ as variational distributions. In such a case, VI would consist in finding the optimal $\mu_1,\ldots,\mu_n,\Sigma_1,\ldots,\Sigma_n$ for each x_i and $i \in \{1,\ldots,n\}$ whereas Amortized VI would model both the means and covariances as parametric functions of x_i with parameters ϕ. In such a case, for any $i \in \{1,\ldots,n\}$ we have a common definition for the variational distributions given by $q_\phi(z_i|x_i) = \mathcal{N}(z_i;\mu_\phi(x_i),\Sigma_\phi(x_i))$.*

0.5 The Variational Autoencoder

A Variational Autoencoder (VAE) introduced in (Kingma and Welling, 2014; Rezende et al., 2014) is a specific type of DLGM that rely on Amortized Variation Inference. The main idea is to introduce a parametric distribution q_ϕ that aims at approximating the true posterior distribution $p_\theta(z|x)$. In the context of VAE models, we typically assume that the latent variable $z \in \mathcal{Z}$ lives in a lower dimensional space than the input data $x \in \mathcal{X}$ such that $q_\phi(z|x)$ is often referred to as the *encoder* (or *recognition model*). Similarly, the conditional distribution $p_\theta(x|z)$ of the data given the latent variable is called the *decoder*. The encoder acts as variational distribution, and a VAE aims at maximizing the ELBO introduced in Eq. (5) with respect to both the variational parameters ϕ and the parameters of the *decoder*, θ.

$$\mathcal{L}(\theta, \phi, x) = \log p_\theta(x) - \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x))\,, \tag{6}$$

When looking as Eq. (6), one may see that by maximizing the ELBO with respect to both ϕ and θ, the VAE will at the same time 1) maximize the marginal likelihood $p_\theta(x)$ which is the main objective of a DLGM and 2) minimize $\mathrm{KL}(q_\phi(z|x)||p_\theta(z|x))$ which drives $q_\phi(z|x)$ to approximate the unknown posterior $p_\theta(z|x)$ as targeted in the Amortized Variational Inference framework. The ELBO can also be written as follows, allowing to perform Stochastic Gradient Descent (SGD).

$$\begin{aligned}
\mathcal{L}(\theta, \phi, x) &= \log p_\theta(x) - \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x))\,, \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \mathrm{KL}(q_\phi(z|x)||p(z))\,, \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z|x)\right]\,.
\end{aligned} \tag{7}$$

We see now in Eq. (7) that the ELBO can be written as an expectation of tractable terms since both $q_\phi(z|x)$ and $p_\theta(x, z) = p_\theta(x|z)p(z)$ are known and efficient to compute.

Remark 1 *The ELBO as written line 2 of Eq.* (7) *can also be seen as a two terms objective (Ghosh et al., 2020). The first one is a reconstruction term given by $p_\theta(x|z)$ while the second one is a regularizer given by the KL between the variational posterior q_ϕ and the prior p. For instance, in the case of a fixed variance Gaussian for $p_\theta(x|z) = \mathcal{N}(x; \mu_\theta(z), \sigma \cdot I_d)$ we have*

$$\mathcal{L}_{REC} = \|x - \mu_\theta(z)\|_2^2, \quad \mathcal{L}_{REG} = \mathrm{KL}(q_\phi(z|x)\|p(z))\,. \tag{8}$$

Since the VAE aims at optimizing the ELBO with respect to both the variational parameters ϕ and the parameters θ, using an SGD-based procedure requires being able to compute the gradients of the ELBO with respect to ϕ and θ *i.e.* $\nabla_{\theta,\phi}\mathcal{L}(\theta, \phi, x)$. The gradient with respect to the parameters θ can be derived as follows

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta, \phi, x) &= \nabla_\theta \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z|x)\right]\,, \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\theta \log p_\theta(x, z)\right]\,, \\
&= \int_{\mathcal{Z}} \nabla_\theta \log p_\theta(x, z) q_\phi(z|x) dz\,.
\end{aligned} \tag{9}$$

Hence, we can easily obtain an unbiased estimate of $\nabla_\theta \mathcal{L}(\theta, \phi, x)$ using Monte Carlo and samples coming from the variational distribution $z \sim q_\phi(z|x)$. While optimizing with respect to θ is pretty straightforward, as explained above, computing the gradient of the ELBO with respect to ϕ is a bit

trickier since the expectation depends on the parameters ϕ. Therefore, the gradient of the expectation no longer equals the expectation of the gradient. To alleviate this issue, one may recourse to what is called the *reparametrization trick*. The main idea behind this *trick* is to express the random variable $z \sim q_\phi(z|x)$, for any $x \in \mathcal{X}$, as a diffeomorphic transformation $g_{\phi,x}$ of another random variable $\varepsilon \sim w(\varepsilon)$ following a distribution easy to sample from and where w is independent of x and ϕ.

Example 3 *For instance, let us assume that $\mathcal{Z} = \mathbb{R}^d$ and the variational posterior is chosen as a multivariate Gaussian distribution $q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \Sigma_\phi(x))$ with diagonal covariance matrix $\Sigma_\phi(x)$. In such a case, one may introduce an auxiliary random variable ε following the non-parametric distribution $w(\varepsilon) = \mathcal{N}(0, I_d)$ and write:*

$$z = g_{\phi,x}(\varepsilon) \qquad \textit{with} \qquad \varepsilon \sim \mathcal{N}(0, I_d) \quad \textit{and} \quad g_{\phi,x} : \varepsilon \to \mu_\phi(x) + \Sigma_\phi(x)^{\frac{1}{2}} \cdot \varepsilon \, .$$

The above transformation is actually strictly equivalent to sampling z from the parametric variational distribution $q_\phi(z|x)$.

Assuming now that for any $x \in \mathcal{X}$ there exists such random variable $\varepsilon \sim w(\varepsilon)$ and diffeomorphism $g_{\phi,x}$ we can now rewrite the ELBO using the *reparametrization trick*:

$$\begin{aligned} \mathcal{L}(\theta, \phi, x) &= \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z|x) \right] \, , \\ &= \mathbb{E}_{\varepsilon \sim w(\varepsilon)} \left[\log p_\theta(x, g_{\phi,x}(\varepsilon)) - \log q_\phi(g_{\phi,x}(\varepsilon)|x) \right] \, . \end{aligned}$$

Now, since the expectation is taken with respect to $\varepsilon \sim w(\varepsilon)$, we can swap gradient with respect to ϕ and the expectation in a similar fashion as Eq. (9) making optimization with respect to the parameters ϕ possible. Finally, once the model is trained, one can generate new samples using the prior distribution and feeding the samples to the *decoder* $p_\theta(x|z)$ according to the generative model given in Eq. (3).

The vanilla VAE as proposed in (Kingma and Welling, 2014) (namely the Gaussian *mean-field* VAE) assumes that the data x live in $\mathcal{X} = \mathbb{R}^d$ as presented in Example 3 and the variational posterior $q_\phi(z|x)$ is modeled by a multivariate Gaussian distribution with diagonal covariance $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ the parameters of which are given by neural networks. Hence, the latent space is seen as a Euclidean space *i.e.* $\mathcal{Z} = \mathbb{R}^d$. The prior distribution is chosen as a simple standard Gaussian distribution $p(z) = \mathcal{N}(0, I_d)$. During training, the ELBO is estimated using only one sample coming from the variational distribution $z \sim q_\phi(z|x)$ for each data point $x \in \mathcal{X}$. Starting from the definition of the vanilla VAE, a wide literature has emerged trying to improve the model. In the following sections, we will elaborate on some of the major lines of development in this area that are relevant to this thesis.

0.6 Towards a Tighter Bound

Despite its apparent simplicity, the VAE framework nonetheless demonstrates a major weakness which is that we are always only optimizing a lower bound (the ELBO) on the actual objective (*i.e.* the marginal log-likelihood $\log p_\theta(x)$). Therefore, many works have been proposed in the literature to achieve tighter bounds that would hopefully lead to a better estimate of $p_\theta(x)$ and so a better model.

0.6.1 Using Better Estimators

First, it can be noted that Evidence Lower BOund (ELBO) objectives can also be obtained starting from an unbiased estimate of the marginal likelihood p_θ using Jensen's inequality. For instance, in the VAE framework, one may consider the variational distribution $q_\phi(z|x)$ as proposal density and apply importance sampling to derive an unbiased estimate of $p_\theta(x)$

$$\widehat{p}_\theta(x) = \frac{p_\theta(x,z)}{q_\phi(z|x)} \text{ and } \mathbb{E}_{z\sim q_\phi}\left[\hat{p}_\theta\right] = p_\theta(x)\,. \tag{10}$$

Using Jensen's inequality allows finding a lower bound on the objective function of Eq. (4)

$$\begin{aligned}
\log p_\theta(x) &= \log \mathbb{E}_{z\sim q_\phi}\left[\hat{p}_\theta\right]\,, \\
&\geq \mathbb{E}_{z\sim q_\phi}\left[\log \hat{p}_\theta\right]\,, \\
&\geq \mathbb{E}_{z\sim q_\phi}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right]\,, \\
&\geq \mathbb{E}_{z\sim q_\phi}\left[\log p_\theta(x|z)\right] - \mathrm{KL}(q_\phi(z|x)\|p(z))\,.
\end{aligned} \tag{11}$$

We see now that we obtain an equivalent expression of the ELBO as shown in Eq. (7). Hence, several works have been developing new estimators of the marginal likelihood p_θ and proposed to derive new ELBO objectives (Burda et al., 2016; Maddison et al., 2017; Domke and Sheldon, 2018; Thin et al., 2021).

Example 4 *In particular, since the vanilla VAE only uses one sample for the importance estimate, (Burda et al., 2016) introduced the Importance Weighted Autoencoder that optimizes an ELBO derived from the K-sample importance weighted estimator of the marginal log-likelihood.*

$$\hat{p}_\theta(x) = \frac{1}{K}\sum_{i=1}^{K}\frac{p_\theta(x,z_i)}{q_\phi(z_i|x)} \text{ and } \mathbb{E}_{z_1,\dots,z_K\sim q_\phi(z|x)}\left[\hat{p}_\theta\right] = p_\theta(x)\,.$$

Using the same reasoning as before allows obtaining an ELBO derived from the aforementioned estimator

$$\mathcal{L}_{\text{IWAE}}(\theta,\phi,x) = \mathbb{E}_{z_1,\dots,z_K\sim q_\phi(z|x)}\left[\log\frac{1}{K}\sum_{i=1}^{K}\frac{p_\theta(x,z_i)}{q_\phi(z_i|x)}\right]\,.$$

Note that in the case $K = 1$, we indeed find back the VAE ELBO. Using the reparametrization trick allows for optimization for both θ and ϕ parameters using stochastic gradient-based algorithms.

0.6.2 Enriching the Variational Distribution

When looking at the expression of the ELBO in Eq. (6), one can see that it is composed of two terms. The first one is the marginal log-likelihood of the data $\log p_\theta(x)$, which is the objective we actually want to maximize, while the second term is the Kullback–Leibler (KL) divergence between the variational posterior distribution $q_\phi(z|x)$ and the true posterior $p_\theta(z|x)$. The latter is always non-negative and vanishes if and only if $q_\phi = p_\theta$ almost everywhere. In other words, the

term $\mathrm{KL}(q_\phi(z|x)||p_\theta(z|x))$ quantifies the gap that exists between the ELBO and the marginal log-likelihood. Hence, a natural way to improve the tightness of the ELBO would consist in considering a richer family of variational distributions $q_\phi(z|x)$ that would hopefully better approximate the *true* posterior $p_\theta(z|x)$. Recall that in the initial version of the VAE, the variational distribution is set as a multivariate Gaussian with diagonal covariance, which is a very restrictive class of distributions, while the actual posterior $p_\theta(z|x)$ may be a complex distribution.

Normalizing Flows

Normalizing flows are flexible models that can be used to transform simple probability densities into much more complex ones by recoursing to sequences of invertible smooth mappings. These models rely on the rule of change of variables such that if $z \in \mathbb{R}^d$ is a random variable that follows the distribution $q(z)$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible smooth function, then the random variable $z' = f(z)$ has a distribution given by

$$q(z') = q(z) \left| \det \frac{\partial f^{-1}}{\partial z'} \right| = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} . \tag{12}$$

In this setting, f is called a *normalizing flow*, and so several flows can be composed to form a new flow $g = f_K \circ f_{K-1} \circ \cdots \circ f_1$ allowing to model richer distributions. The ability of these flows to transform simple distributions into much more complex ones was, in particular, proposed in the context of amortized variational inference in (Rezende and Mohamed, 2015). They were indeed combined with the VAE framework to enhance the expressiveness of the variational posterior $q_\phi(z|x)$. In such a context, the flows are parametrized and learned during training at the same time as the *encoder* and *decoder* networks. The complexity of the transformations in the flows considered is nonetheless constrained since the flows need to be invertible and smooth, and the computation of the det-Jacobian of the flow must remain tractable and should not constitute a strong additional computation burden.

Example 5 *The family of invertible linear transformations is, for instance, one of the simplest sets of transformations one may think of to build normalizing flows. In (Rezende and Mohamed, 2015), the authors proposed to consider the following transformation*

$$f(z) = z + u \cdot h(\omega^\top z + b) ,$$

*where $z \in \mathbb{R}^d$ and $\{\omega \in \mathbb{R}^d, u \in \mathbb{R}^d, b \in \mathbb{R}\}$ are parameters learned during training and h is a smooth non-linear function (*e.g. $\tanh$*). These types of flows are called Planar flows.*

We show in Fig. 1, another example on the two moons dataset (Pedregosa et al., 2011) using Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017) and a standard Gaussian prior. On the left, we show the resulting sampling using 2048 samples from the prior and transforming them using the learned flows. On the right, we show the learned inverse transformation applied to the input data. In addition to the flows presented in the above examples, amongst the most widely known flows, we can also mention NICE (Dinh et al., 2014), radial flows (Rezende and Mohamed, 2015), RealNVP (Dinh et al., 2016), Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017)

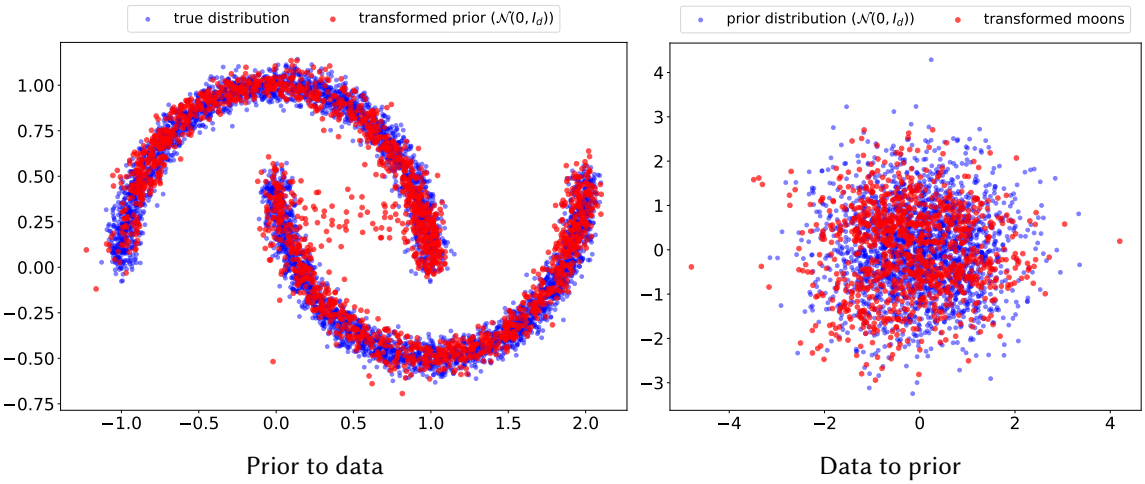


Figure 1: *Left*: Sampling using Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017) using a standard prior ($\mathcal{N}(0, I_d)$) on the two moons dataset (Pedregosa et al., 2011). *Right*: Mapping of the data to the prior using the flows. Plots are made using (Chadebec et al., 2022c).

or Inverse Autoregressive Flows (IAF) (Kingma et al., 2016). The latter flows were in particular proposed in (Kingma et al., 2016) to improve upon the works of (Rezende and Mohamed, 2015) with a new type of normalizing flow that better scales to high-dimensional latent spaces. The main idea is again to apply several transformations to a sample drawn from a simple distribution in order to model richer distributions. Starting from $z_0 \sim q_\phi(z|x)$, the proposed IAF flow consists in applying consecutively the following transformation

$$z_k = \mu_k + \sigma_k \odot z_{k-1} \, ,$$

where μ_k and σ_k are the outputs of an autoregressive neural network taking z_{k-1} (and potentially a context vector h) as input. The choice of autoregressive neural networks imposes by construction that the Jacobian of μ_k and σ_k with respect to z_{k-1} are triangular with zeros on their diagonal. This makes the computation of the det-Jacobian of each flows easy and fast.

Auxiliary Variables

Another way to increase the expressiveness of the variational distribution consists in adding auxiliary random variables (Salimans et al., 2015; Maaløe et al., 2016; Ranganath et al., 2016). The main idea is to work with an extended space by adding an *auxiliary* continuous random variable $u \in \mathcal{U}$ and consider an augmented inference model that writes

$$q_\phi(u, z|x) = q_\phi(u|x) q_\phi(z|u, x) \, .$$

This additional random variable u allows accessing to a potentially richer class of marginal distribution $q_\phi(z|x)$ since we have

$$q_\phi(z|x) = \int_{\mathcal{U}} q_\phi(u, z|x) du \, .$$

The extended generative model using the auxiliary random variable follows

$$p_\theta(x, z, u) = p_\theta(u|x, z)p_\theta(x, z)\,.$$

In a similar fashion as Eq. (10), one can build an unbiased estimator of the marginal likelihood $p_\theta(x)$

$$\widehat{p}_\theta(x) = \frac{p_\theta(x, z, u)}{q_\phi(u, z|x)} \text{ and } \mathbb{E}_{(u,z)\sim q_\phi}\left[\hat{p}_\theta\right] = p_\theta(x)\,.$$

This allows to derive an ELBO

$$\begin{aligned}\log p_\theta(x) &= \mathbb{E}_{(u,z)\sim q_\phi}\left[\widehat{p}_\theta(x)\right]\,,\\ &\geq \mathbb{E}_{(u,z)\sim q_\phi}\left[\log\left(\frac{p_\theta(x, z, u)}{q_\phi(u, z|x)}\right)\right] = \mathcal{L}_{\text{aux}}(\theta, \phi, x)\,.\end{aligned}$$

Moreover, one may note that we have

$$\begin{aligned}\mathcal{L}_{\text{aux}}(\theta, \phi, x) &= \mathbb{E}_{(u,z)\sim q_\phi}\left[\log\left(\frac{p_\theta(x, z)p_\theta(u|x, z)}{q_\phi(u|z, x)q_\phi(z|x)}\right)\right]\,,\\ &= \int_{\mathcal{Z}}\int_{\mathcal{U}} \log\left(\frac{p_\theta(x, z)}{q_\phi(z|x)}\right) q_\phi(u|z, x)q_\phi(z|x)dudz\\ &+ \int_{\mathcal{Z}}\left[\int_{\mathcal{U}} \log\left(\frac{p_\theta(u|x, z)}{q_\phi(u|z, x)}\right) q_\phi(u|z, x)du\right] q_\phi(z|x)dz\,,\\ &= \int_{\mathcal{Z}} \log\left(\frac{p_\theta(x, z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz - \mathbb{E}_{z\sim q_\phi}\left[\mathrm{KL}(q_\phi(u|z, x)||p_\theta(u|x, z))\right]\,,\\ &= \mathbb{E}_{z\sim q_\phi}\left[\frac{p_\theta(x, z)}{q_\phi(z|x)}\right] - \mathbb{E}_{z\sim q_\phi}\left[\mathrm{KL}(q_\phi(u|z, x)||p_\theta(u|x, z))\right]\,,\\ &= \mathcal{L}(\theta, \phi, x) - \mathbb{E}_{z\sim q_\phi}\left[\mathrm{KL}(q_\phi(u|z, x)||p_\theta(u|x, z))\right]\,,\\ &\leq \mathcal{L}(\theta, \phi, x) \quad (= \text{vanilla VAE ELBO})\,.\end{aligned}$$

This equation means that using an auxiliary random variable can be detrimental to the tightness of the ELBO since it becomes looser than the original VAE objective. Nonetheless, this approach allows to access to a wider class of variational distributions that will hopefully outweigh the term $\mathbb{E}_{z\sim q_\phi}\left[\mathrm{KL}(q_\phi(u|z, x)||p_\theta(u|x, z))\right]$. Several works actually show that considering auxiliary random variables to increase the expressiveness of the variational distribution can indeed lead to enhanced models (Salimans et al., 2015; Ranganath et al., 2016; Maaløe et al., 2016; Caterini et al., 2018; Thin et al., 2021; Chadebec et al., 2022b).

Example 6 *An approach proposed by Salimans et al. (2015) consists in benefitting from the ability of MCMC methods to sample from the exact unknown posterior distribution $p_\theta(z|x)$ and the authors applied it to the VAE framework. They indeed proposed to add a fixed number T of MCMC steps on the top of the variational approximation and targeting the true posterior distribution as follows*

$$z_0 \sim q_\phi(z_0|x) \text{ and } z_t \sim q(z_t|z_{t-1}, x),\ \forall t \in \{1, \dots, T\}\,,$$

where $q_\phi(z_0|x)$ is a simple initial distribution (typically $q_\phi(z_0|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ in the context of VAEs) and $q(z_t|z_{t-1}, x)$ is the transition operator used in the Markov Chain. The main idea of the

authors is to interpret the created Markov Chain $q_\phi(z|x) = q_\phi(z_0)\prod_{t=1}^{T} q(z_t|z_{t-1}, x)$ *as a variational approximation in an extended space by seeing the* $u = z_0, \dots, z_{t-1}$ *as* auxiliary *random variables. As explained in this section, this allows obtaining an unbiased estimate of the marginal likelihood (and so derive an ELBO) as follows:*

$$\hat{p}_\theta(x) = \frac{p_\theta(x, z_T)\prod_{t=1}^{T} r(z_{t-1}|z_t, x)}{q_\phi(z_0|x)\prod_{t=1}^{T} q(z_t|z_{t-1}, x)},$$

where $r(z_{t-1}|z_t, x)$ *are* artificial *reverse transition kernels being chosen in this case with a Markov structure aiming at approximating* $q(u|x, z_T)$*. As an example, the authors introduced what they called* Hamiltonian Variational Inference *where they chose to rely on transition operators inspired by the Hamiltonian Monte Carlo (HMC) sampler (Neal and others, 2011). This approach was further extended in several works (Wolf et al., 2016; Hoffman, 2017; Caterini et al., 2018; Chadebec et al., 2022b). One may also link this approach to Normalizing Flows described in Sec. 0.6.2 since Hamiltonian Monte Carlo can be interpreted as a normalizing flow on the extended space* $(z, v) \in \mathcal{Z} \times \mathbb{R}^d$ *where* v *are auxiliary variables called the* momentum *(Rezende and Mohamed, 2015). A very nice property of those flows relies in the fact that they are guided by the gradient of the true posterior, which is a well-known property of the Hamiltonian dynamics, but they are however quite computationally demanding.*

0.6.3 Rethinking our Priors

While much effort has been focused on improving the ELBO using more flexible variational distributions, (Hoffman and Johnson, 2016) argued that particular attention should be paid to the choice of the prior distribution as well. First, we recall that one of the expressions of the ELBO in the vanilla VAE setting writes as follows:

$$\mathcal{L}(\theta, \phi, x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \mathrm{KL}(q_\phi(z|x)||p(z))\,.$$

Assuming that we are given N *i.i.d* observations x with empirical distribution p_{data}, (Hoffman and Johnson, 2016) showed that the ELBO averaged across the observations can be written as follows

$$\begin{aligned}
\mathcal{L}(\theta, \phi) &= \mathbb{E}_{p_{\text{data}}}\left[\mathcal{L}(\theta, \phi, x)\right] \\
&= \frac{1}{N}\sum_{i=1}^{N}\left[\mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right] - \mathrm{KL}(q_\phi(z|x_i)||p(z))\right]\,, \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{q_\phi(z|x_i)}\left[\log p_\theta(x_i|z)\right] - \mathbb{I}_{q(x,z)}[x, z] - \mathrm{KL}(q^{\text{avg}}(z)||p(z))\,,
\end{aligned} \tag{13}$$

where $\mathbb{I}_{q(x,z)}[x, z] = \mathbb{E}_{q(x,z)}\left[\log\left(\frac{q(x_i,z)}{p(x_i)q^{\text{avg}}(z)}\right)\right]$ is the mutual information of the observation x and the latent code z under $q(x, z) = p_{\text{data}}(x)q_\phi(z|x)$, $q^{\text{avg}}(z) = \frac{1}{N}\sum_{i=1}^{N} q_\phi(z|x_i)$ is the *aggregated* posterior and $p(z)$ is the prior distribution (standard Gaussian in the vanilla VAE setting). In Eq. (13), the first term corresponds to the classic reconstruction term of the ELBO, while the second one drives the latent codes to overlap for different observations x_i. Interestingly, with such writing of the ELBO

the prior $p(z)$ only appears in the last term, indicating that our choice in the prior distribution will only impact this term without being detrimental to terms 1 and 2. This expression nicely highlights how the over-regularization coming from the prior arises when one chooses too simplistic priors. It is easy to see that the prior maximizing the ELBO is given by the *aggregated* posterior. Nonetheless, such a choice may lead to overfitting and would add a substantial computation burden to the model due to the sum over all the observations N (Hoffman and Johnson, 2016; Makhzani et al., 2015; Tomczak and Welling, 2018). To conclude, this discussion justifies the need for more expressive prior distributions but also shows that a trade-off between expressiveness and tractability of the prior in the ELBO needs to be taken into account.

As the standard VAE model uses a standard Gaussian distribution as prior, a natural improvement that was proposed in (Nalisnick et al., 2016; Dilokthanakul et al., 2017) consists in considering a mixture of Gaussian distributions instead. Keeping in mind that the *optimal* prior is the *aggregated* posterior, (Tomczak and Welling, 2018) introduced a VAriational Mixture of Posterior (VAMP) prior that aims at approximating the *aggregated* posterior while trying to address the undesirable effects such as overfitting and computational burden. The main idea is to introduce K *pseudo-inputs* variables $u_k \in \mathcal{X}$ (*i.e.* living in the observations space) and learn them at the same time as the model parameters θ and ϕ. The VAMP prior then writes:

$$p_\lambda^{\text{VAMP}}(z) = \frac{1}{K}\sum_{i=1}^{K} q_\phi(z|u_k)\,,$$

where λ corresponds to the prior's parameters $\lambda = \{\phi, u_1, \dots, u_K\}$.

0.7 Improving the Learned Latent Representations

Even though the VAE framework aims at maximizing the likelihood of the data by nature, its autoencoding structure constrains the observations to be encoded into a lower dimensional space called the latent space, the structure of which may reveal very interesting properties. Therefore, better understanding the inherent structure of this latent space or improving the quality of latent representations learned by the model are two questions that have also greatly intrigued the community over the past few years.

0.7.1 Learning Disentangled Representations

Although there is no clear consensus on the definition of disentanglement, it is commonly referred to as the independence between features in a representation (Bengio et al., 2013; Eastwood and Williams, 2018; Mathieu et al., 2019b).

In particular, it has been argued that learning disentangled representations of the input data can be beneficial to a model and for downstream tasks (Bengio et al., 2013; Lake et al., 2017; Higgins et al., 2017). Learning disentangled representations in the context of Variational Autoencoders consists in obtaining a latent space in which each direction corresponds to an independent generative factor of the data. In other words, trying to achieve disentanglement amounts to constraining the empirical distribution of the latent codes *i.e.* the *aggregated* posterior to be factorial *i.e.* $q^{\text{avg}}(z) = \prod_j^d q(z_j)$

where j is the j-th component of the latent code z. It was first shown in (Higgins et al., 2017) that the VAE can offer a simple yet efficient unsupervised way to learn disentangled factors in its latent space. The authors indeed proposed to amend the training objective of the vanilla VAE framework by adding a weight factor balancing the reconstruction term of the ELBO and the regularization. According to the authors, the rationale behind this is to put stronger constraints over the latent representations to follow a simple factorial prior ($\mathcal{N}(0, I_d)$) while still being able to reconstruct the input data.

$$\mathcal{L}_{\beta\text{-VAE}}(\theta, \phi, x) = \mathbb{E}_{q_\phi}\left[\log p_\theta(x|z) - \beta \cdot \mathrm{KL}(q_\phi(z|x)||p(z))\right] ,$$

where $\beta > 1$. Nonetheless, this extra pressure put on the latent variables can lead to over-regularization and, as a consequence, become detrimental to the reconstruction capability of the model. To address this limitation (Burgess, 2018) proposed to gradually increase a target value for $\mathrm{KL}(q_\phi(z|x)||p(z))$ during training up to a limit value. However, both approaches penalize the whole KL term, while this may not be needed as far as disentanglement is targeted. In particular, one may indeed note that from Eq. (13) we have

$$\mathrm{KL}(q_\phi(z|x)||p(z)) = \mathbb{I}_{q(x,z)}[x, z] - \mathrm{KL}(q^{\text{avg}}(z)||p(z)) ,$$

Hence, penalizing the term $\mathrm{KL}(q_\phi(z|x)||p(z))$ in the ELBO with $\beta > 1$ indeed pushes the model to learn a factorial *aggregated* posterior $q^{\text{avg}}(z)$ that is driven to resemble the prior but also adds a higher weight on the mutual information term $\mathbb{I}_{q(x,z)}[x, z]$ as well. The latter induces a reduction of the amount of information about the observations contained in the corresponding latent codes (Kim and Mnih, 2018), which leads to poorer reconstructions (Makhzani and Frey, 2017). Instead, more recent approaches proposed to explicitly penalize a term that forces independence over the *aggregated* posterior distribution (Kim and Mnih, 2018; Chen et al., 2018b). This term is known as the Total Correlation and measures the dependence between random variables (Watanabe, 1960). Other approaches also considered a semi-supervised setting where the generative factor are known (Kingma et al., 2014; Paige et al., 2017; Bouchacourt et al., 2018).

0.7.2 Exploring Latent Space Modeling

A particularly powerful aspect of the VAE framework relies in the flexibility the latent space $\mathcal{Z}$ offers. Modeled as a simple Euclidean space in its original version, several works have explored and proposed diverse modeling of the latent space.

Beyond Euclidean Latent Spaces

In particular, the geometry of the latent space learned by the model is something that has driven some interest in the community in recent years. For instance, (Arvanitidis et al., 2018) argued that the latent space learned by a VAE is naturally endowed with a Riemannian geometry where the Riemannian metric is a function of the Jacobian of the decoder. Indeed, assuming that $D_\theta : \mathcal{Z} \to \mathcal{X}$ is a parametric function that outputs the parameters of the decoding distribution $p_\theta(x|z)$, one may consider a latent code $z \in \mathcal{Z}$ and an infinitesimal δz and write the following first order Taylor expansion:

$$D_\theta(z + \delta z) - D_\theta(z) \approx \mathbf{J}_{D_\theta}(z)\delta z \;\; \textit{i.e.} \;\; \|D_\theta(z + \delta z) - D_\theta(z)\|^2 \approx (\delta z)^\top \mathbf{J}_{D_\theta}(z)^\top \mathbf{J}_{D_\theta}(z)\delta z ,$$

where $\mathbf{J}_{D_\theta}(z)$ is the Jacobian matrix of D_θ evaluated at z. Using this equation, the authors argued that if we assume that the observations live in a Euclidean space, typically $\mathcal{X} = \mathbb{R}^D$ endowed with the Euclidean distance, the decoder mapping induces a natural local distance in the latent space that is scaled by its Jacobian. In other words, the latent space is no longer seen as a Euclidean space but as the Riemannian manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ where $\mathbf{G}(z) = \mathbf{J}_{D_\theta}(z)^\top \mathbf{J}_{D_\theta}(z)$ is the Riemannian metric. This approach paved the way for considering a more flexible geometry of the latent space modeled as a Riemannian manifold where the Riemannian metric is given by the Jacobian of the generator function (Arvanitidis et al., 2018; Chen et al., 2018a; Shao et al., 2018). This metric was, for instance, used directly within the prior modeled as a Riemannian Brownian motion (Kalatzis et al., 2020). Other approaches also proposed to learn the metric directly from the data during training using *geometry-aware* normalizing flows (Chadebec et al., 2020) or learn the latent structure of the data using transport operators (Connor et al., 2021). With another viewpoint, some works also proposed to impose a specific structure on the latent space defined *a-priori*. For instance, modeling of the latent space as a torus (Falorsi et al., 2018), a hypersphere (Davidson et al., 2018) or a Poincaré disk (Mathieu et al., 2019a) were proposed. These very nice ideas allow learning a latent representation of the input data that is quite different from the one learned by the classic VAE. As an illustration, we show in Fig. 2, three different 2-dimensional latent spaces leaned either by a vanilla VAE ($\mathcal{N}$-VAE), a hyper-spherical VAE ($\mathcal{S}$-VAE) or a Poincaré Disk VAE ($\mathcal{P}$-VAE). In Example 7 is described more precisely the framework of the hyper-spherical VAE as an example. Another very interesting approach that proved very well suited for images consists in considering a discretized latent space (Van Den Oord et al., 2017). Therefore, in such a case, the latent space is defined as a $\mathbb{R}^{K\times D}$ vector space of K different D dimensional embedding vectors $\mathcal{E} = \{e_1, \dots, e_K\}$, referred to as the codebook, which is learned and updated at each iteration. Given an embedding size d and an input x, the output of the encoder $z_e(x)$ is of size $\mathbb{R}^{d\times D}$. Each of its d elements is then assigned to the closest embedding vector resulting in an embedded encoding $z_q(x) \in \mathcal{E}^d$ such that $\big(z_q(x)\big)_j = e_l$ where $l = \text{argmin}_{1\leq l\leq d} ||(z_e(x))_j - e_l||_2$ for $j \in [1, d]$. Since the argmin operation is not differentiable, the learning of the embeddings and regularization of the latent space is done by introducing the stop-gradient operator sg in the training objective:

$$\mathcal{L}_{\text{VQ-VAE}}(x) \coloneqq \log p(x|z_q(x)) + \alpha||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2\,,$$

where α and β are hyper-parameters scaling each term.

Example 7 *A nice example of a different modeling of the latent space with a geometry different from the Euclidean one is the hyper-spherical VAE proposed in (Davidson et al., 2018). The authors indeed noted that the conventional Gaussian prior pulls latent representation of the data towards the origin, which in some cases is not desirable. They propose to replace the conventional Euclidean latent space with a hyper-spherical latent space to allow data or clusters of data in the latent space to spread evenly.*

The uniform distribution over the hypersphere is used as a prior p whereas the Von-Mises-Fisher distribution is used as an approximate prior $q_\phi(z|x)$ leading to the modified ELBO

$$\mathcal{L}_{\mathcal{S}\text{-VAE}}(x) \coloneqq \mathbb{E}_{z\sim q_\phi(z|x)}\big[\log p_\theta(x|z)\big] - \text{KL}\left[\textit{vMF}(z|x)||\mathcal{U}(\mathcal{S})\right]\,,$$

where $\textit{vMF}(z|x)$ is the parametrized posterior Von-Mises-Fisher distribution generated by the encoder and $\mathcal{U}(\mathcal{S})$ the prior uniform distribution on the latent hyper-sphere.

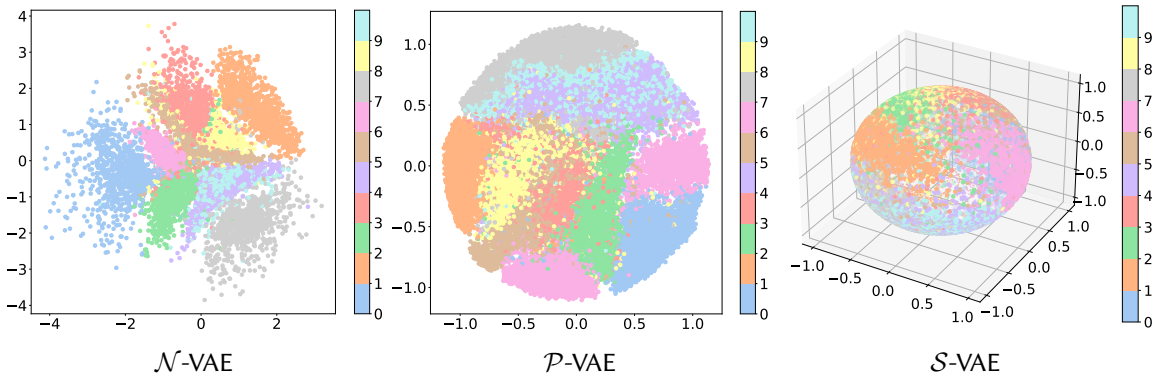


Figure 2: 2-dimensional latent spaces learned by a vanilla VAE ($\mathcal{N}$-VAE), Poincaré VAE ($\mathcal{P}$-VAE) and hyper-spherical VAE ($\mathcal{S}$-VAE) on MNIST. The colors represent the digits. Plots are made using (Chadebec et al., 2022c).

Some Elements of Riemannian Geometry

A d-dimensional manifold $\mathcal{M}$ is a manifold which is locally homeomorphic to a d-dimensional Euclidean space. If the manifold $\mathcal{M}$ is further differentiable, it possesses a tangent space T_z at any $z \in \mathcal{M}$ composed of the tangent vectors of the curves passing by z. If $\mathcal{M}$ is equipped with a smooth inner product $g : z \to \langle \cdot | \cdot \rangle_z$ defined on its tangent space T_z for any $z \in \mathcal{M}$ then $\mathcal{M}$ is called a Riemannian manifold and g is the associated Riemannian metric. A chart (or coordinate system) (U, ϕ) is a homeomorphic mapping from an open set U of the manifold to an open set V of a Euclidean space. Given $z \in U$, a chart $\phi : (z^1, \dots, z^d)$ induces a basis $\left(\frac{\partial}{\partial z^1}, \dots, \frac{\partial}{\partial z^d}\right)_z$ on the tangent space T_z. Hence, the metric of a Riemannian manifold can be locally represented in the chart ϕ as a positive definite matrix.

$$\mathbf{G}(z) = (g_{i,j})_{z, 0 \leq i,j \leq d} = \left(\left\langle \frac{\partial}{\partial z^i} | \frac{\partial}{\partial z^j} \right\rangle_z \right)_{0 \leq i,j \leq d},$$

for each point z of the manifold. That is for $v, w \in T_z\mathcal{M}$ and $z \in \mathcal{M}$, the inner product writes $\langle u | w \rangle_z = u^\top \mathbf{G}(z) w$. Assuming that the manifold is also connected, for any $z_1, z_2 \in \mathcal{M}$, two points of the manifold, we can consider a curve γ traveling in $\mathcal{M}$ and parametrized by $t \in [a, b]$ such that $\gamma(a) = z_1$ and $\gamma(b) = z_2$. Then, the length of γ is given by

$$L(\gamma) = \int_a^b \|\dot{\gamma}(t)\|_{\gamma(t)} dt = \int_a^b \sqrt{\langle \dot{\gamma}(t) | \dot{\gamma}(t) \rangle_{\gamma(t)}} dt$$

Curves γ that minimize L and are parameterized proportionally to the arc length are called *geodesic* curves. A distance $\text{dist}_{\mathbf{G}}$ on the manifold $\mathcal{M}$ can then be derived and writes

$$\text{dist}_{\mathbf{G}}(z_1, z_2) = \inf_\gamma L(\gamma) \quad \text{s.t.} \quad \gamma(a) = z_1, \ \gamma(b) = z_2$$

The manifold $\mathcal{M}$ is said to be *geodesically complete* if all geodesic curves can be extended to $\mathbb{R}$. For any $p \in \mathcal{M}$, the exponential map at p, Exp_p, maps a vector v of the tangent space $T_p\mathcal{M}$ to a point of the manifold $\widetilde{p} \in \mathcal{M}$ such that the geodesic starting at p with initial velocity v reaches $\widetilde{p}$ at time 1. In particular, if the manifold is *geodesically complete*, then Exp_p is defined on the entire tangent space $T_p\mathcal{M}$.

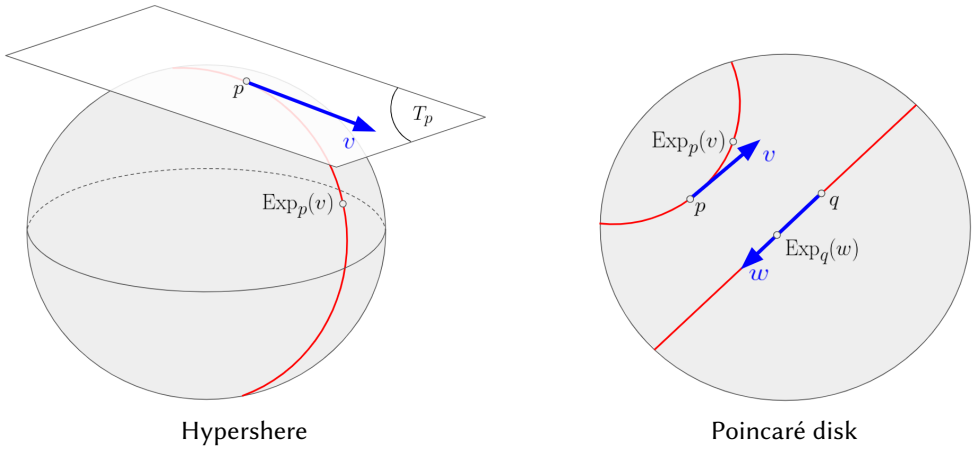


Figure 3: Geodesic and exponential map on the Hypershere and Poincaré disk.

Remark 2 *A d-dimensional Euclidean space is a specific d-dimensional Riemannian manifold. In such a case, the Riemannian metric $\mathbf{G}$ indeed reduces to I_d, and the distance becomes the classic Euclidean one. Geodesic curves are straight lines, and we have for any $z \in \mathbb{R}^d$, $T_z = \mathbb{R}^d$*

Example 8 *A simple extension of the Euclidean framework consists in assuming that the metric is given by a constant positive definite matrix $\mathbf{\Sigma}$ different from I_d. In such a case, the induced Riemannian distance is the well-known Mahalanobis distance* $\mathrm{dist}_{\mathbf{\Sigma}}(z_1, z_2) = \sqrt{(z_2 - z_1)^\top \mathbf{\Sigma}(z_2 - z_1)}$.

Example 9 *The Poincaré disk is a Riemannian manifold endowed with the Riemannian metric given by $\mathbf{G}(z) = \frac{2}{1-\|z\|^2} I_d$. Fig. 3 shows the geodesic curves and Exponential mapping resulting from such a geometry.*

Given the Riemannian manifold $\mathcal{M}$ endowed with the Riemannian metric $\mathbf{G}$ and a chart z, an infinitesimal volume element may also be defined on each tangent space T_z of the manifold $\mathcal{M}$ as follows

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)} dz \,,$$

where dz is the Lebesgue measure. This defines a canonical measure on the manifold and allows extending the notion of probability distributions to Riemannian manifolds. In particular, such a property allows referring to random variables with a density defined with respect to the measure on the manifold. We recall such definition from (Pennec, 2006) below

Definition 1 *Let $\mathcal{B}(\mathcal{M})$ be the Borel σ-algebra of $\mathcal{M}$. The random point $\mathbf{z}$ has a probability density function $\rho_{\mathbf{z}}$ if:*

$$\forall \mathcal{Z} \in \mathcal{B}(\mathcal{M}), \;\; \mathbb{P}(\mathbf{z} \in \mathcal{Z}) = \int_{\mathcal{Z}} \rho(z) d\mathcal{M}(z) \;\; \textit{and} \;\; \int_{\mathcal{M}} \rho(z) d\mathcal{M}(z) = 1$$

Finally, given a chart ϕ defined on the whole manifold $\mathcal{M}$ and a random point $\mathbf{z}$ on $\mathcal{M}$, the point $\mathbf{p} = \phi(\mathbf{z})$ is a random point whose density $\rho'_{\mathbf{p}}$ may be written with respect to the Lebesgue measure as such (Pennec, 2006):

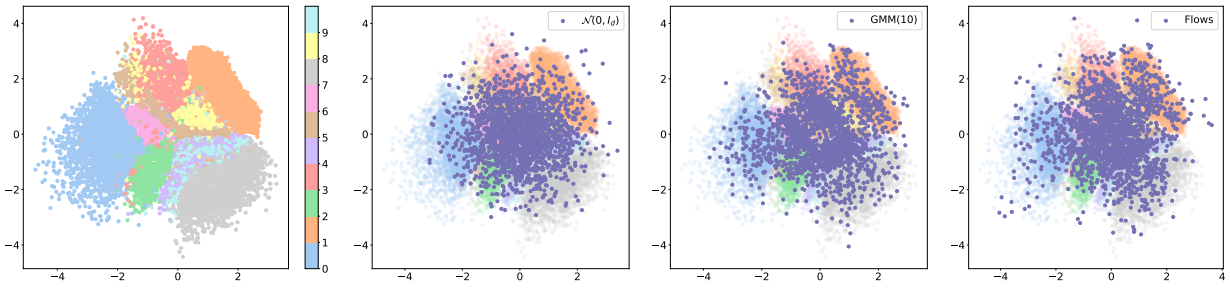
$$\rho'_{\mathbf{p}}(p) = \rho_{\mathbf{z}}(\phi^{-1}(p))\sqrt{\det g(\phi^{-1}(p))}$$

0.7.3 Improving the Generative Capability of the Model

Although the aforementioned enhancements proposed in the literature sometimes resulted in a model able to generate more realistic samples, this was not the main targeted goal. Instead, another branch of the literature on Variational Autoencoders specifically focused on improving the generative capabilities of the model. It was indeed often noted that VAE models tend to produce blurry and fuzzy samples when compared to other generative models such as Generative Adversarial Networks (Goodfellow et al., 2014). A natural way to tackle this issue consists in amending the distribution used for sampling *i.e.* the prior $p(z)$. We recall that assuming that a VAE model is trained properly, one may easily generate new samples by simply sampling latent codes using the prior distribution and feeding those samples directly to the decoder. Nonetheless, as explained in Sec. 0.6.3, there may exist a mismatch between the prior distribution set *a-priori* and the actual distribution of the latent codes given by the *aggregated* posterior inducing distribution mismatch (Connor et al., 2021). Moreover, the chosen prior may be too simplistic and cause over-regularization of the latent space leading to a degraded data generation (Dai and Wipf, 2018). Hence, some works seeking to find more expressive priors proposed to rely on hierarchical latent variables models (Sønderby et al., 2016; Burda et al., 2016; Klushyn et al., 2019; Maaløe et al., 2019; Vahdat and Kautz, 2020; Child, 2021). Another idea that has recently been proposed and also resulted in major improvements in image generation consists in learning the prior using for instance flows (Chen et al., 2016b), autoregressive models (Razavi et al., 2020), energy-based models (Pang et al., 2020; Aneja et al., 2020) or diffusion models (Vahdat et al., 2021). Depending on the approach, the prior distribution can be learned either during training or post-training.

Example 10 *For instance, (Ghosh et al., 2020) proposed to use* ex-post *density estimation consisting in fitting more expressive distributions such as mixtures of Gaussians or Normalizing flows in the latent space post-training. The density estimator is directly fitted on the latent codes of the training dataset and then used to produce samples in the latent space. Fig. 4 shows the resulting latent space sampling when using the prior distribution, fitting a 10-component mixture of Gaussians or using flows for a 2-dimensional latent space learned by a VAE trained on MNIST (LeCun, 1998). This example, in particular, shows that using a more expressive distribution after training can allow for a better prospecting of the latent space and potentially avoid un-informative locations. This paves the way for considering other generation schemes.*

From another perspective, papers proposed to amend the first term of the ELBO in Eq. (7) (*i.e.* the reconstruction term). On the ground that the L2 distance deriving from the decoding distribution (see Eq. (8)) often set as a Gaussian to model continuous input data (*e.g.* images) may be unadapted for structured data, some papers proposed to amend the distance between the input and the reconstructed samples coming from the decoder. For instance, (Snell et al., 2017) proposed to use a



$\mathcal{N}$-VAE latent space $\mathcal{N}(0, I_d)$ 10-component GMM MAF

Figure 4: *From left to right:* 2-dimensional latent spaces learned by a vanilla VAE ($\mathcal{N}$-VAE), a latent space sampling using the prior $\mathcal{N}(0, I_d)$, using a 10-component mixture of Gaussian distributions or using Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017). The colors represent the digits. Plots are made using (Chadebec et al., 2022c)

.

data-dependent deterministic reconstruction cost $\Delta(x, D_\theta(z))$ where D_θ : maps the latent codes back to the data space. The modified training objective is thus defined as

$$\mathcal{L}_{\text{EL-VAE}}(x) = \Delta(x, \hat{x}) - \beta \cdot \text{KL}\left(q_\phi(z|x)||p(z)\right) \ ,$$

with $\beta \leq 1$. In their paper, they proposed to use a multi-scale variant of the single-scale SSIM (Wang et al., 2004): the Multi-Scale Structural Similarity Metric (MS-SSIM) (Wang et al., 2003) for images. With the same objective, (Larsen et al., 2016) introduced the VAE-GAN model that combines the VAE framework while trying to benefit from the ability of GAN models to generate sharper images. They indeed used a GAN-based approach by training a discriminator at the same time as the VAE. The role of the discriminator is three-fold. Similarly to the GAN framework (Goodfellow et al., 2014), it first aims at distinguishing real data from the data generated using the prior distribution in the latent space and the decoder of the VAE. Second, it is also used to distinguish the real samples from the reconstructed ones. Finally, noting that intermediate layers of a discriminative network can act as data-specific features, the authors proposed to replace the reconstruction loss in the ELBO with a Gaussian log-likelihood between outputs of intermediate layers of the discriminator network. With another approach, (Tolstikhin et al., 2018) introduced a variant of the VAE framework where the problem is formulated using an Optimal Transport (OT) perspective. They indeed introduced the Wasserstein Autoencoder (WAE) models aiming at minimizing any optimal transport cost c between the distribution of the true data and the data generated using the prior distribution. Other approaches also proposed to enrich the decoder network with autoregressive models (Chen et al., 2016b; Gulrajani et al.) such as PixelCNN (Van den Oord et al., 2016; Van Den Oord et al., 2016).

0.8 VAE in Practice

Variational Autoencoders belong to a very versatile class of models and have been used in many different application fields and for various tasks. In this section, we detail some of these applications of the VAE framework (or variants). This section does not aim to be an exhaustive list of all the possible application cases of the VAE but rather aims to provide a flavour of the versatility of the

model.

Due to the availability of large and clean databases, image synthesis has been one of the most widely popular application fields of generative modeling. In particular, VAEs have demonstrated to be particularly well suited for image generation (Huang et al., 2018; Van Den Oord et al., 2017; Razavi et al., 2020; Vahdat and Kautz, 2020; Child, 2021). Moreover, the intrinsic structure of their latent space has proved very powerful to perform image edition. This can, for instance, be done by encoding an image and, starting from the resulting latent code, moving along a specific dimension of the latent space and finally decoding the resulting latent embedding (White, 2016; Berthelot* et al., 2019). Super resolution of images is also a downstream task that has been explored with Variational Autoencoders (Snell et al., 2017). As an example, (Liu et al., 2021b) proposed a VAE framework where the model can be conditioned on any given image that serves as *reference* to perform super-resolution on another image. The autoencoding structure of the VAEs also makes them a very good candidate to perform information compression or feature extraction. They revealed particularly useful and powerful when combined with diffusion models in recent works (Vahdat et al., 2021; Rombach et al., 2022). For instance, the Stable Diffusion model (Rombach et al., 2022) that demonstrated very impressive results for *text-to-image* data generation relies on a Vector-Quantized VAE (VQ-VAE) model to learn a lower dimensional representation of the images.

Beyond images, VAE models have been applied to different data structures such as Graphs (Kipf and Welling, 2016; Zhang et al., 2019), where they can be used to model chemical structures. For example, one application is to be able to generate molecules with given properties (Liu et al., 2018a; Lim et al., 2018a; Griffiths and Hernández-Lobato, 2020). The generative capabilities of the VAE model also revealed useful to perform data augmentation (Shorten and Khoshgoftaar, 2019). The main idea is to train a generative model on the training data available and use it to increase the size of the training set by generating new *labelled* synthetic data. In particular, (Hsu et al., 2017) proposed to rely on a Variational Autoencoder to augment the number of labeled training data in the context of speech recognition while (Chadebec et al., 2022b) proposed a VAE that allowed increasing the performances of a *state-of-the-art* classifier trained for the detection of Alzheimer's disease on 3D MRIs. A huge branch of the literature has also focused on applying VAEs to sequential data (Johnson et al., 2016; Fraccaro et al., 2016; Karl et al., 2017). Several approaches were indeed proposed in the context of Natural Language Processing (NLP) (Bowman et al., 2016; Miao et al., 2016; Yang et al., 2017; Serban et al., 2017; Zhao et al., 2017), video generation (Gur et al., 2020; Zhu et al., 2020) or longitudinal data (Sauty and Durrleman, 2022) and demonstrated promising results. Some models were also specifically designed for time-series (Casale et al., 2018; Fortuin et al., 2019) to perform either missing data imputation (Fortuin et al., 2020), anomaly detection (Niu et al., 2020; Lin et al., 2020) or prediction (Jin et al., 2022).

VAE models can also be used for tasks that are different from generation. For instance, (Dilokthanakul et al., 2017) and (Jiang et al., 2017) proposed to replace the standard Gaussian prior with a Gaussian mixture to perform unsupervised clustering directly in the latent space of the VAE. Semi supervised learning is also a task for which the VAE framework seemed well designed. First proposed in (Kingma et al., 2014), VAEs were then used for semi-supervised classification tasks and showed remarkable performances (Rezende et al., 2016; Pu et al., 2016; Xu et al., 2017; Ehsan Abbasnejad et al., 2017). Some works have also proposed and designed models inspired by the VAE to perform collaborative filtering (Liang et al., 2018; Sachdeva et al., 2019; Yu et al., 2019).

Some extensions of the framework presented in this introduction were also proposed in the liter-

ature to include, for instance, conditioning variables (Sohn et al., 2015). This proposal also conducted to the development of models able to handle more than one modality called *multi-modal* VAEs (Shi et al., 2019; Wu and Goodman, 2018; Sutter et al., 2021; Suzuki et al., 2016; Vedantam et al., 2018)

0.9 Contributions

0.9.1 List of Publications

Below are listed the contributions of the PhD

- **Variational Inference for Longitudinal Data Using Normalizing Flows**
  Chadebec, C. and Allassonnière, S., Under review

- **Improving Multimodal Joint Variational Autoencoders through Normalizing Flows and Correlation Analysis**
  Senellart, A., Chadebec, C., Allassonnière, S., Under review

- **A Geometric Perspective on Variational Autoencoders**
  Chadebec, C. and Allassonnière, S., Proceedings of the Neural Information Processing Systems, 2022

- **Pythae: Unifying Generative Autoencoders in Python, A Benchmarking Use Case**
  Chadebec, C., Vincent, L. J. and Allassonnière, S., Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmark. 2022.

- **Data Augmentation in High Dimensional Low Sample Size Setting Using a Geometry-Based Variational Autoencoder**
  Chadebec, C., Thibeau-Sutre, E., Burgos, N. and Allassonnière, S., IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022

- **An Image Feature Mapping Model for Continuous Longitudinal Data Completion and Generation of Synthetic Patient Trajectories**
  Chadebec, C., Huijben, E. M., Pluim, J. P., Allassonnière, S. and J.M van Eijnatten, M. A., Deep Generative Models: MICCAI 2022 Workshop

- **Data Augmentation with Variational Autoencoders and Manifold Sampling**
  Chadebec, C. and Allassonnière, S., Deep Generative Models, and Data Augmentation, Labelling, and Imperfections: MICCAI 2021 Workshop

0.9.2 Summary of the Main Contributions

In this section, we summarize the main contributions of the PhD thesis.

Data Augmentation in High Dimensional Low Sample Size Setting Using a Geometry-Based Variational Autoencoder This paper extends the Hamiltonian Variational AutoEncoder framework proposed in (Salimans et al., 2015; Caterini et al., 2018) to latent spaces seen as Riemannian manifolds. It also proposes a new way to generate synthetic data from the trained model using the geometry of the latent space. Finally, the model is validated and used for data augmentation in the challenging context of High Dimensional Low Sample Size data. One of the main ideas of the proposed model is to perform MCMC sampling steps on top of the variational approximation as described in Example 6 using transition operators inspired by Hamiltonian dynamics but also benefiting from the assumed Riemannian structure of the latent space. Introducing the momentum variables $v \in \mathbb{R}^d$ and considering an augmented variational distribution on the extended space $(z, v) \in \mathcal{Z} \times \mathbb{R}^d$, we derive an estimator of the marginal likelihood $p_\theta(x)$ as follows:

$$\hat{p}_\theta(x) = \frac{p_\theta(x, z_K, v_K)}{q_\phi(z_K, v_K|x)} = \frac{p_\theta(x|z_K)p(v_K|z_K)p(z_K)}{q_\phi(z_0|x)p(v_0|z_0)\prod_{k=1}^K |\det \mathbf{J}_{g^k}|^{-1}},$$

where $\mathbf{J}_{g^k}$ is the Jacobian of the k-th step g_k of the generalized *leapfrog* integrator (Leimkuhler and Reich, 2004; Girolami et al., 2009; Girolami and Calderhead, 2011), $p(v|z) = \mathcal{N}(0, \mathbf{G}(z))$ is a *position-specific* proposal distribution for the momentums where $\mathbf{G}(z)$ is the value of the Riemannian metric at z. Since the Riemannian metric is unknown, we propose to parametrize it and learn it directly from the data using neural networks. Then, we propose a *geometry-aware* sampling scheme exploiting the geometry of the latent space learned by the proposed model and consisting in sampling close to the geodesic paths. This sampling mechanism demonstrated promising generative performances, in particular in the context of small sample size where it outperforms the *prior-based* generation and other *post-training* density estimation methods. Finally, the method is used to perform data augmentation for classification tasks and is validated across a wide experimental study where robustness to data and classifiers is tested. In particular, the method was able to increase the performances achieved by a *state-of-the-art* classifier trained to detect Alzheimer's disease on 3D MRIs.

Data Augmentation with Variational Autoencoders and Manifold Sampling This paper introduces a new way to sample from unknown *geodesically-complete* Riemannian manifolds. The main idea is to create a random walk like algorithm that discovers the manifold along geodesic paths. Starting from a point z belonging to the Riemannian manifold $\mathcal{M}$, a proposal is obtained by first sampling a vector $v \sim \mathcal{N}(0, \Sigma)$ in the tangent space T_z at z and then applying the Riemannian exponential map to shoot the vector back onto the manifold. This amounts to sampling from the *wrapped* normal distribution defined on the Riemannian manifold $\mathcal{M}$. The proposal is finally accepted using some probability ratio. The method is then applied to the Riemannian VAE introduced in the previous paper to perform data augmentation. Fig. 5 illustrates how the sampling algorithm works.

A Geometric Perspective on Variational Autoencoders This paper introduces a new perspective on the Variational Autoencoder framework by taking a fully geometric viewpoint. In a nutshell, we start from the observation that a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ is only a specific case of the Riemannian Gaussian distribution $\mathcal{N}^{\text{riem}}(z|\mu, \sigma) = \frac{1}{C}\exp\left(-\frac{\text{dist}_{\mathbf{G}}(z,\mu)^2}{2\sigma}\right)$ defined on the Riemannian manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ where $\sigma = 1$ and the Riemannian metric $\mathbf{G}$ is the constant metric $\mathbf{G}(z) = \Sigma^{-1}$. Hence, we propose to interpret the variational distribution

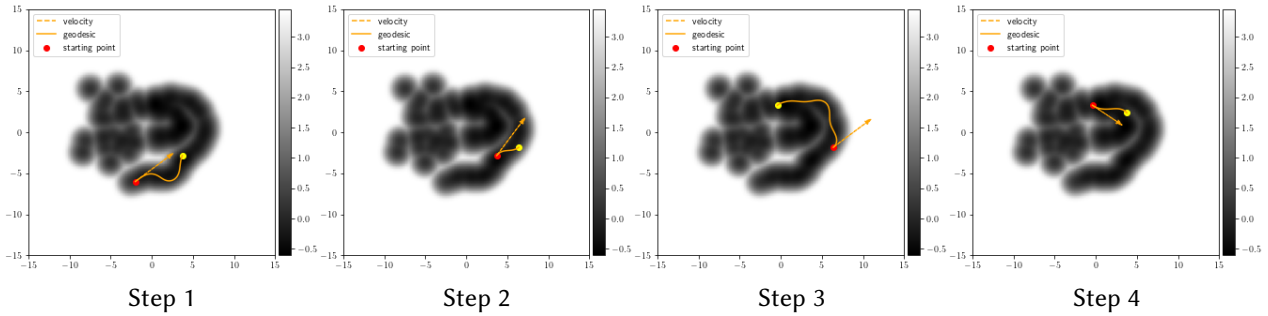


Figure 5: 4 steps of the proposed Riemannian Random Walk to discover a 2-dimensional latent space learned by a RHVAE.

$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ no longer as a multivariate Gaussian but as a *Riemannian* Gaussian distribution. For any training data point x_i, $\Sigma_\phi(x_i)$ is understood as an approximation of the unknown Riemannian metric $\mathbf{G}$ evaluated at the embedding point $\mu_\phi(x_i)$ and defining a Riemannian latent space seen as the Riemannian manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$. During training, this metric is assumed locally constant close to the embedding point $\mu_\phi(x_i)$ (*i.e.* $\mathbf{G}(z) \approx \Sigma(x_i)^{-1}$ for z close to $\mu_\phi(x_i)$). Hence, in such a case, sampling from the Riemannian distribution $q_\phi(z|x)$ can be approximated by sampling from the multivariate Gaussian distribution $\mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ as done for the training of the vanilla VAE. Finally, at the end of the training, these approximations $(\Sigma_i^{-1})_i$ of the Riemannian metric $\mathbf{G}$ are put together to build a smooth continuous Riemannian metric on the whole latent space of the trained VAE.

$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d \,,$$

$$\omega_i(z) = \exp\left(- \frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2} \right),$$

where $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z - \mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z - \mu(x_i))$ is the Riemannian distance between z and $\mu(x_i)$ with respect to the locally constant metric $\mathbf{G}(\mu(x_i)) = \mathbf{\Sigma}^{-1}(x_i)$. We also propose a new sampling mechanism that consists in sampling from the intrinsic uniform distribution defined on the learned latent space $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ as follows:

$$\mathcal{U}_{\text{Riem}}(z) = \frac{\sqrt{\det \mathbf{G}(z)}}{\int_{\mathbb{R}^d} \sqrt{\det \mathbf{G}(z) dz}} \,.$$

We empirically show that such a sampling scheme allows to enhance the generative capability of the vanilla VAE (the training process of which remains unchanged) that can outperform more advanced and potentially more costly methods proposed afterwards.

Pythae: Unifying Generative Autoencoders in Python, A Benchmarking Use Case This paper introduces Pythae, a versatile *open-source* Python library providing both a *unified implementation* and a dedicated framework allowing *straightforward*, *reproducible* and *reliable* use of generative autoencoder models. At the time I am writing this thesis, this library implements and unifies 25

implementations of (variational) autoencoder models corresponding to variants of the vanilla VAE framework. The main purpose of this Python library is to lower the entry barrier to using these models while ensuring they are flexible enough to adapt to various use cases and data types (*i.e.* not only images). Most of the provided implementations were able to reproduce the main results in the original papers making the library reliable. Moreover, this library is unit-tested and complies to open-source standards. It is available under the Apache2.0 license and can be installed from *pypi* or *conda*. At the time this thesis is made available, the library has raised some interest in the community with 1.3k+ stars on Github, 10k+ downloads, 8 external contributors and was also presented at the PyTorch 2022 conference in New Orleans. In addition to introducing Pythae, this paper also proposes a benchmark use-case where 19 implementations available in Pythae at the time are compared in the same settings and for 5 different tasks such as image reconstruction, generation, classification, clustering and interpolation. This benchmark aimed at providing potentially interesting insights on the model that are part of the comparison while showing the usefulness of the library for benchmarking applications. The code can be found at the following link.

An Image Feature Mapping Model for Continuous Longitudinal Data Completion and Generation of Synthetic Patient Trajectories This paper proposes a generative model for longitudinal image data acting directly in the latent space of a trained VAE. It indeed starts with the observation that thanks to the intrinsic structure of their latent space, Variational Autoencoder models naturally account for a trajectory (which is not a straight line) in the latent space when trained with such data. Hence, we propose to learn a function mapping these trajectories into a linear trajectory in a Euclidean space defined with velocity, delay, and spatial parameters that are learned directly from the data in a similar spirit as (Louis et al., 2019).

$$l_i(t) = \exp(\eta_i)(t - \tau_i) \cdot \mathbf{e_1} + \sum_{k=2}^{d} \lambda_i^k \cdot \mathbf{e_k}\,,$$

where η_i is a velocity parameter, τ_i is a delay, and $\lambda_i = (\lambda_i^k)_{2 \leq k \leq d}$ are spatial parameters. Contrary to (Louis et al., 2019), we adopt a fully variational approach to make the model generative in a similar fashion as (Kingma and Welling, 2014). Given P input sequences $(x_i)_{i \in [1,P]}$ having each t_i observations such that $x_i = (x_i^0, \dots, x_i^{t_i})$, a first Variational Autoencoder is trained on all the data without taking into account their temporal structure. This step allows learning a lower dimensional representation of the input data. Then, a second model is trained to map the embeddings of the input trajectories $(x_i)_{i \in [1,P]}$ onto a linear trajectory amounting to estimate the parameters $(\eta_i, \tau_i, \lambda_i)$.

Variational Inference for Longitudinal Data Using Normalizing Flows This paper proposes a new latent variable generative model and framework able to handle high-dimensional longitudinal data. Given an entity $i \in \{1, \dots, P\}$ and a sequence of observations $(x_0^i, \dots, x_{t_i}^i)$, we assume that for each x_j^i where $j \in \{0, \dots, t_i\}$, there exists an associated latent variable $z_j^i \in \mathcal{Z} = \mathbb{R}^d$ involved in the generative process of the observation x_j^i such that $x_j^i \sim p_\theta(x_j^i | z_j^i)$. Since the observations within a sequence are no longer independent (longitudinal setting), the VAE framework cannot be applied in such a case since the likelihood of Eq. (4) does not factorize across observations anymore. Hence, in this paper, we propose to model the time dependency between the observations within a sequence using normalizing flows f_j over the latent variables.

$$z_0^i \sim p(z_0^i),\ z_1^i = f_1(z_0^i),\ \dots\ , z_{t_i}^i = f_{t_i}(z_{t_i - 1}^i)\,, \tag{14}$$

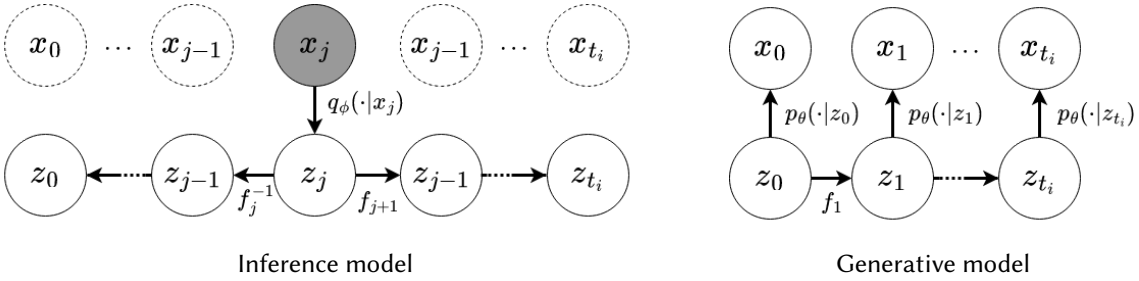


Figure 6: Proposed inference and generative models.

where p is a simple prior distribution over z_0^i (*e.g.* standard Gaussian), and f_j are normalizing flows for any $j \in \{1, \dots, t_i\}$. Since the f_j are chosen as invertible mappings, one may see that we also have a tractable *prior* for z_j^i using Eq. (12). Assuming that the observations within a sequence are independent when conditioned with respect to the associated latent variables and noting that, given z_j^i, we can retrieve the complete sequence of latent variables $(z_0^i, \dots, z_{t_i}^i)$ using Eq. (14), one may write the joint likelihood as follows:

$$p_\theta(x_0^i, \cdots, x_{t_i}^i) = \int_{\mathcal{Z}} p_\theta(x_0^i, \cdots, x_{t_i}^i | z_j^i) p(z_j^i) dz_j^i = \int_{\mathcal{Z}} \prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i) p(z_j^i) dz_j^i \,.$$

Since this integral is most of the time intractable, we propose to rely on amortized variational inference as in (Kingma and Welling, 2014) and introduce a parametric distribution $q_\phi(z_j^i | x_j^i)$ that allows obtaining an unbiased estimate of the joint likelihood and so derive an ELBO as shown in Eq. (11).

$$\log p_\theta(x_0^i, \cdots, x_{t_i}^i) \geq \mathbb{E}_{q_\phi} \left[\log \prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i) \right] - \mathrm{KL}(q_\phi(z_j^i | x_j^i) | p(z_j^i)) \,.$$

The training is performed by randomly picking a j in $[0, t_i]$, sampling $z_j^i \sim q_\phi(z_j^i | x_j^i)$, recovering a reconstructed sequence $(\widehat{x}_0^i, \dots, \widehat{x}_{t_i}^i)$ using Eq. (14) and optimizing the ELBO. Once the model is trained, one may generate fully synthetic sequences by first $z_0 \sim p(z_0)$ and apply the flows according to Eq. (14). Fig. 6 shows the graphical inference and generative models for the proposed approach. We also discuss in the paper how to handle missing data at training and inference time and how to generate sequences conditioned on 1 or several observation(s) in an input sequence.

Chapter 1

Toward a Geometry-Aware VAE

In this chapter, we propose a new method to perform data augmentation in a reliable way in the High Dimensional Low Sample Size (HDLSS) setting using a geometry-based variational autoencoder (VAE). Our approach combines the proposal of 1) a new VAE model, the latent space of which is modeled as a Riemannian manifold and which combines both Riemannian metric learning and normalizing flows and 2) a new generation scheme which produces more meaningful samples especially in the context of small data sets. The method is tested through a wide experimental study where its robustness to data sets, classifiers and training samples size is stressed. It is also validated on a medical imaging classification task on the challenging ADNI database where a small number of 3D brain magnetic resonance images (MRIs) are considered and augmented using the proposed VAE framework. In each case, the proposed method allows for a significant and reliable gain in the classification metrics. For instance, balanced accuracy jumps from 66.3% to 74.3% for a state-of-the-art *convolutional neural network classifier trained with 50 MRIs of cognitively normal (CN) and 50 Alzheimer disease (AD) patients and from 77.7% to 86.3% when trained with 243 CN and 210 AD while improving greatly sensitivity and specificity metrics.*

This chapter was published in IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). See (Chadebec et al., 2022b).

1.1 Introduction

Even though always larger data sets are now available, the lack of labelled data remains a tremendous issue in many fields of application. Among others, a good example is healthcare where practitioners have to deal most of the time with (very) low sample sizes (think of small patient cohorts) along with very high dimensional data (think of neuroimaging data that are 3D volumes with millions of voxels). Unfortunately, this leads to a very poor representation of a given population and makes classical statistical analyses unreliable (Button et al., 2013; Turner et al., 2018). Meanwhile, the remarkable performance of algorithms heavily relying on the deep learning framework (Goodfellow et al., 2016) has made them extremely attractive and very popular. However, such results are strongly conditioned by the number of training samples since such models usually need to be trained on huge data sets to prevent over-fitting or to give statistically meaningful results (Shorten and Khoshgoftaar, 2019).

A way to address such issues is to perform data augmentation (DA) (Tanner and Wong, 1987). In a nutshell, DA is the art of increasing the size of a given data set by creating synthetic labeled data. For instance, the easiest way to do this on images is to apply simple transformations such as the addition of Gaussian noise, cropping or padding, and assign the label of the initial image to the created ones. While such augmentation techniques have revealed very useful, they remain strongly data dependent and limited. Some transformations may indeed be uninformative or even induce bias. For instance, think of a digit representing a *6* which gives a *9* when rotated. While assessing the relevance of augmented data may be quite straightforward for simple data sets, it reveals very challenging for complex data and may require the intervention of an *expert* assessing the degree of relevance of the proposed transformations. In addition to the lack of data, imbalanced data sets also severely limit generalizability since they tend to bias the algorithm toward the most represented classes. Oversampling is a method that aims at balancing the number of samples per class by upsampling the minority classes. The Synthetic Minority Over-sampling TEchnique (SMOTE) was first introduced in (Chawla et al., 2002) and consists in interpolating data points belonging to the minority classes in their feature space. This approach was further extended in other works where the authors proposed to over-sample close to the decision boundary using either the k-Nearest Neighbor (k-NN) algorithm (Han et al., 2005) or a support vector machine (SVM) (Nguyen et al., 2011) and so insist on samples that are potentially misclassified. Other over-sampling methods aiming at increasing the number of samples from the minority classes and taking into account their difficulty to be learned were also proposed (Haibo He et al., 2008; Barua et al., 2012). However, these methods hardly scale to high-dimensional data (Blagus and Lusa, 2013; Fernández et al., 2018).

The recent rise in performance of generative models such as generative adversarial networks (GAN) (Goodfellow et al., 2014) or variational autoencoders (VAE) (Kingma and Welling, 2014) has made them very attractive models to perform DA. GANs have already seen a wide use in many fields of application (Zhu et al., 2018a; Mariani et al., 2018; Antoniou et al., 2018-03-21; Lim et al., 2018b; Zhu et al., 2018b), including medicine (Yi et al., 2019). For instance, GANs were used on magnetic resonance images (MRIs) (Shin et al., 2018; Calimeri et al., 2017), computed tomography (CT) (Frid-Adar et al., 2018; Sandfort et al., 2019), X-ray (Madani et al., 2018; Salehinejad et al., 2018; Waheed et al., 2020), positron emission tomography (PET) (Bi et al., 2017), mass spectroscopy data (Liu et al., 2019), dermoscopy (Baur et al., 2018) or mammography (Korkinof et al., 2018; Wu et al., 2018) and demonstrated promising results. Nonetheless, most of these studies involved either a quite large

training set (above 1000 training samples) or quite small dimensional data, whereas in everyday medical applications it remains very challenging to gather such large cohorts of labeled patients. As a consequence, as of today, the case of high dimensional data combined with a very low sample size remains poorly explored. When compared to GANs, VAEs have only seen a very marginal interest to perform DA and were mostly used for speech applications (Hsu et al., 2017; Nishizaki, 2017; Wu et al., 2019). Some attempts to use such generative models on medical data either for classification (Zhuang et al., 2019; Liu et al., 2018b) or segmentation tasks (Painchaud et al., 2019; Selvan et al., 2020; Myronenko, 2018) can nonetheless be noted. The main limitation to a wider use of these models is that they most of the time produce blurry and fuzzy samples. This undesirable effect is even more emphasized when they are trained with a small number of samples which makes them very hard to use in practice to perform DA in the high dimensional (very) low sample size (HDLSS) setting.

In this chapter, we argue that VAEs can actually be used for data augmentation in a reliable way even in the context of HDLSS data, provided that we bring some modeling of the latent space and amend the way we generate the data. Hence, in this chapter we propose the following contributions:

- We propose a new *geometry-aware* VAE model, the latent space of which is seen as a Riemannian manifold and combining Riemannian metric learning and normalizing flows.
- We introduce a new *non-prior* based generation procedure consisting in sampling from the inverse of the Riemannian metric volume element learned by the model. The choice of this framework is discussed, motivated and compared to other VAE models.[1]
- We propose to use such a framework to perform data augmentation in the challenging context of HDLSS data. The robustness of the augmentation method to data sets and classifiers changes along with its reliance to the number of training samples and the complexity of the classifier is then tested through a series of experiments.[2]
- We validate the proposed method on several *real-life* classification tasks on complex 3D MRI from ADNI and AIBL databases where the augmentation method allows for a significant gain in classification metrics even when only 50 samples per class are considered.

1.2 Variational Autoencoder

In this section, we quickly recall the idea behind VAEs along with some proposed improvements relevant to this chapter.

[1] An implementation of the models may be found at https://github.com/clementchadebec/benchmark_VAE

[2] A software implementing the method was developed and is available at https://github.com/clementchadebec/pyraug

1.2.1 Model Setting

Let $x \in \mathcal{X}$ be a set of data. A VAE aims at maximizing the likelihood of a given parametric model $\{\mathbb{P}_\theta, \theta \in \Theta\}$. It is assumed that there exist latent variables z living in a lower dimensional space $\mathcal{Z}$, referred to as the *latent space*, such that the marginal distribution of the data can be written as:

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p(z)dz\,, \tag{1.1}$$

where p is a prior distribution over the latent variables acting as a regulation factor and $p_\theta(x|z)$ is most of the time taken as a simple parametrized distribution (*e.g.* Gaussian, Bernoulli, etc.). Such a distribution is referred to as the *decoder*, the parameters of which are usually given by neural networks. Since the integral of Eq. (1.1) is most of the time intractable, so is the posterior distribution:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p(z)}{\int_{\mathcal{Z}} p_\theta(x|z)p(z)dz}\,.$$

This makes direct application of Bayesian inference impossible and so recourse to approximation techniques such as variational inference (Jordan et al., 1999) is needed. Hence, a variational distribution $q_\phi(z|x)$ is introduced and aims at approximating the true posterior distribution $p_\theta(z|x)$ (Kingma and Welling, 2014). This variational distribution is often referred to as the *encoder*. In the initial version of the VAE, q_ϕ is taken as a multivariate Gaussian whose parameters μ_ϕ and Σ_ϕ are again given by neural networks. Importance sampling is then applied to get an unbiased estimate of $p_\theta(x)$ we want to maximize in Eq. (1.1)

$$\hat{p}_\theta(x) = \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \quad \text{and} \quad \mathbb{E}_{z\sim q_\phi}\big[\hat{p}_\theta\big] = p_\theta(x)\,. \tag{1.2}$$

Using Jensen's inequality allows finding a lower bound on the objective function of Eq. (1.1)

$$\begin{aligned} \log p_\theta(x) &= \log \mathbb{E}_{z\sim q_\phi}\big[\hat{p}_\theta\big] \\ &\geq \mathbb{E}_{z\sim q_\phi}\big[\log \hat{p}_\theta\big] \\ &\geq \mathbb{E}_{z\sim q_\phi}\big[\log p_\theta(x|z)\big] - \mathrm{KL}(q_\phi(z|x)\|p(z))\,. \end{aligned} \tag{1.3}$$

The Evidence Lower BOund (ELBO) is now tractable since all distributions are known and so can be optimized with respect to the *encoder* and *decoder* parameters.

1.2.2 Improving the Model: Literature Review

In recent years, many attempts to improve the VAE model have been made and we briefly discuss three main areas of improvement that are relevant to this chapter in this section.

Enhancing the Variational Approximate Distribution

When looking at Eq. (1.3), it can be noticed that we are nonetheless trying to optimize only a lower bound on the true objective function. Therefore, much efforts have been focused on making this

lower bound tighter and tighter (Burda et al., 2016; Alemi et al., 2016; Higgins et al., 2017; Cremer et al., 2018; Zhang et al., 2018a; Ruiz and Titsias, 2019). One way to do this is to enhance the expressiveness of the approximate posterior distribution q_ϕ. This is indeed due to the ELBO expression which can be also written as follows:

$$ELBO = \log p_\theta(x) - \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)) \,.$$

This expression makes two terms appear. The first one is the function we want to maximize while the second one is the Kullback–Leibler (KL) divergence between the approximate posterior distribution $q_\phi(z|x)$ and the true posterior $p_\theta(z|x)$. This very term is always non-negative and equals 0 if and only if $q_\phi = p_\theta$ almost everywhere. Hence, trying to tweak the approximate posterior distribution so that it becomes *closer* to the true posterior should make the ELBO tighter and enhance the model. To do so, a method proposed in (Salimans et al., 2015) consisted in adding K Markov chain Monte Carlo (MCMC) sampling steps on the top of the approximate posterior distribution and targeting the true posterior. More precisely, the idea was to start from $z_0 \sim q_\phi(z|x)$ and use parametrized *forward* (resp. *reverse*) kernels $r(z_{k+1}|z_k, x)$ (resp. $r(z_k|z_{k+1}, x)$) to create a new estimate of the true marginal distribution $p_\theta(x)$. With the same objective, parametrized invertible mappings f_x called *normalizing flows* were instead proposed in (Rezende and Mohamed, 2015) to *sample* z. A starting random variable z_0 is drawn from an initial distribution $q_\phi(z|x)$ and then K normalizing flows are applied to z_0 resulting in a random variable $z_K = f_x^K \circ \cdots \circ f_x^1(z_0)$ whose density writes:

$$q_\phi(z_K|x) = q_\phi(z_0|x) \prod_{k=1}^{K} |\det \mathbf{J}_{f_x^k}|^{-1} \,,$$

where $\mathbf{J}_{f_x^k}$ is the Jacobian of the k^{th} normalizing flow. Ideally, we would like to have access to normalizing flows targeting the true posterior and allowing enriching the above distribution and so improve the lower bound. In that particular respect, a model inspired by the Hamiltonian Monte Carlo sampler (Neal and others, 2011) and relying on Hamiltonian dynamics was proposed in (Salimans et al., 2015) and (Caterini et al., 2018). The strength of such a model relies in the choice of the normalizing flows which are guided by the gradient of the true posterior distribution.

Improving the Prior Distribution

While enhancing the approximate posterior distribution resulted in major improvements of the model, it was also argued that the prior distribution over the latent variables plays a crucial role as well (Hoffman and Johnson, 2016). Since the vanilla VAE uses a standard Gaussian distribution as prior, a natural improvement consisted in using a mixture of Gaussian instead (Nalisnick et al., 2016; Dilokthanakul et al., 2017) which was further enhanced with the proposal of the variational mixture of posterior (VAMP) (Tomczak and Welling, 2018). In addition, other models trying to amend the prior and relying on hierarchical latent variables have been proposed (Sønderby et al., 2016; Burda et al., 2016; Klushyn et al., 2019). Prior learning is also a promising idea that has emerged (*e.g.* (Chen et al., 2016b)) or more recently (Razavi et al., 2020; Pang et al., 2020; Aneja et al., 2020) and allows accessing complex prior distributions. In the same vein, *ex-post* density estimation was also proposed and consists in fitting a simple distribution such as a mixture of Gaussian in the latent space post training (Ghosh et al., 2020). This approach aimed at alleviating the poor expressiveness of the prior. Another approach relying on accept/reject sampling to improve the prior distribution (Bauer

and Mnih, 2019a) can also be cited. While these proposals improved the model, the choice of the prior distribution remains tricky and strongly conditioned by the training data and the tractability of the ELBO.

Adding Geometrical Consideration to the Model

In the mean time, several papers have been arguing that geometrical aspects should also be taken into account. For instance, on the ground that the vanilla VAE fails to apprehend data having a latent space with a specific geometry, several latent space modelings were proposed as a hypershere (Davidson et al., 2018) where Von-Mises distributions are considered instead of Gaussian or as a Poincare disk (Mathieu et al., 2019a; Ovinnikov, 2020). Other works trying to introduce Riemannian geometry within the VAE framework proposed to model either the input data space (Falorsi et al., 2018; Miolane and Holmes, 2020) or the latent space (or both) (Arvanitidis et al., 2016; Chen et al., 2018a; Shao et al., 2018; Kalatzis et al., 2020) as Riemannian manifolds.

1.3 The Proposed Method

In this section, we first present a new *geometry-aware* VAE model bridging the gap between Sec. 1.2.2 and Sec. 1.2.2. It combines MCMC sampling and Riemannian metric learning to improve the expressiveness of the posterior distribution and learn meaningful latent representations of the data. Secondly, we propose a new *non-prior* based generation scheme taking into account the learned geometry of the data. We indeed argue that while the vast majority of works dealing with VAE generate new data using the prior distribution, which is standard procedure, this is often sub-optimal, in particular in the context of small data sets. We believe that the choice of the prior distribution is strongly data set dependent and is also constrained to be simple so that the ELBO in Eq. (1.3) remains tractable. Hence, the view adopted here is to consider the VAE only as a dimensionality reduction tool which is able to extract the latent structure of the data, *i.e.* the latent space modeled as the Riemannian manifold $(\mathbb{R}^d, g)$ where d is the dimension of the manifold and g is the associated Riemannian metric. Before going further we first recall some elements on Riemannian geometry.

1.3.1 Some Elements on Riemannian Geometry

In the framework of differential geometry, one may define a (connected) Riemannian manifold $\mathcal{M}$ as a smooth manifold endowed with a Riemannian metric g that is a smooth inner product $g : p \to \langle \cdot | \cdot \rangle_p$ on the tangent space $T_p\mathcal{M}$ defined at each point of the manifold $p \in \mathcal{M}$. We call a chart (or coordinate chart) (U, φ) a homeomorphism mapping an open set U of the manifold to an open set V of an Euclidean space. The manifold is called a $d-$dimension manifold if for each chart of an atlas we further have $V \subset \mathbb{R}^d$. That is there exists a neighborhood U of each point p of the manifold such that U is homeomorphic to $\mathbb{R}^d$. Given $p \in U$, the chart $\varphi : (x^1, \ldots, x^d)$ induces a basis $\left(\frac{\partial}{\partial x^1}, \ldots, \frac{\partial}{\partial x^d}\right)_p$ on the tangent space $T_p\mathcal{M}$. Hence, a local representation of the metric of a Riemannian manifold in the chart (U, φ) can be written as a positive definite matrix $\mathbf{G}(p) = (g_{i,j})_{p, 0 \leq i,j \leq d} = (\langle \frac{\partial}{\partial x^i} | \frac{\partial}{\partial x^j} \rangle_p)_{0 \leq i,j \leq d}$ at each point $p \in U$. That is for $v, w \in T_p\mathcal{M}$ and $p \in U$, we have $\langle u | w \rangle_p = u^\top \mathbf{G}(p) w$. Since we

propose to work in the ambient-like manifold $(\mathbb{R}^d, g)$, there exists a global chart given by $\varphi = id$. Hence, for the following, we assume that we work in this coordinate system and so $\mathbf{G}$ will refer to the metric's matrix representation in this chart. The length of a curve $\gamma : [0, 1] \to \mathcal{M}$ traveling from $z_1 \in \mathcal{M}$ to $z_2 \in \mathcal{M}$ such that $\gamma(0) = z_1$ and $\gamma(1) = z_2$ is then given by

$$\mathcal{L}(\gamma) = \int_0^1 \|\dot{\gamma}(t)\|_{\gamma(t)} \mathrm{d}t = \int_0^1 \sqrt{\langle \dot{\gamma}(t) | \dot{\gamma}(t) \rangle_{\gamma(t)}} \mathrm{d}t \,.$$

Curves minimizing $\mathcal{L}$ are called *geodesics* and a distance dist between any $z_1, z_2 \in \mathcal{M}$ can be introduced as follows:

$$\mathrm{dist}(z_1, z_2) = \inf_{\gamma} \mathcal{L}(\gamma) \quad \text{s.t.} \quad \gamma(0) = z_1, \;\; \gamma(1) = z_2$$

The manifold $\mathcal{M}$ is said to be *geodesically complete* if all geodesic curves can be extended to $\mathbb{R}$.

1.3.2 A Geometry-Aware VAE

We now assume that the latent space is the Riemannian manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ with $\mathbf{G}$ being the Riemannian metric. Building upon the Hamiltonian VAE (HVAE) (Caterini et al., 2018), we propose to exploit the assumed Riemannian structure of the latent space by using Riemannian Hamiltonian dynamics (Girolami and Calderhead, 2011) instead. The main goal remains the same and consists in using the Riemannian Hamiltonian Monte Carlo (RHMC) sampler to be able to enrich the variational posterior $q_\phi(z|x)$ such that it targets the true (unknown) posterior $p_\theta(z|x)$ while exploiting the properties of Riemannian manifolds.

Riemannian Hamiltonian Monte Carlo Sampler

The Riemannian Hamiltonian Monte Carlo (RHMC) sampler aims at sampling from complex target probability distributions $p_{\text{target}}(z)$ where z is assumed to live in a Riemannian manifold $\mathcal{M}$. The main idea is to introduce a random variable $v \sim \mathcal{N}(0, \mathbf{G}(z))$ where $\mathbf{G}$ is the Riemannian metric associated to $\mathcal{M}$ and rely on Riemannian Hamiltonian dynamics. Analogous to physical systems, $z \in \mathcal{M}$ is seen as the *position* and v as the *velocity* of a particle traveling in $\mathcal{M}$ whose potential energy $U(z)$ and kinetic energy $K(z, v)$ are given by

$$\begin{aligned} U(z) &= -\log p_{\text{target}}(z) \,, \\ K(v, z) &= \frac{1}{2}\Big[\log\left((2\pi)^d |\mathbf{G}(z)|\right) + v^\top \mathbf{G}^{-1}(z) v \Big] \,. \end{aligned}$$

The sum of these energies give together the Hamiltonian $H(z, v)$ (Duane et al., 1987; Leimkuhler and Reich, 2004).

$$H(z, v) = U(z) + \frac{1}{2}\log((2\pi)^D \det \mathbf{G}(z)) + \frac{1}{2} v^\top \mathbf{G}(z)^{-1} v \,.$$

The evolution in time of such a particle is governed by Hamilton's equations which write:

$$\begin{cases} \dfrac{\partial H}{\partial v_i} = \left(\mathbf{G}^{-1}(z)v\right)_i , \\ -\dfrac{\partial H}{\partial z_i} = \dfrac{\partial \log p_{\text{target}}(z)}{\partial z_i} \quad -\dfrac{1}{2}\text{tr}\left(\mathbf{G}^{-1}\dfrac{\partial \mathbf{G}(z)}{\partial z_i}\right) + \dfrac{1}{2}v^\top \mathbf{G}^{-1}(z)\dfrac{\partial \mathbf{G}(z)}{\partial z_i}\mathbf{G}^{-1}(z)v \, . \end{cases} \tag{1.4}$$

This system of equations can be integrated using a discretization scheme known as the generalized *leapfrog* integrator.

$$\begin{aligned} v(t+\varepsilon/2) &= v(t) - \frac{\varepsilon}{2}\nabla_z H\Big(z(t), v(t+\varepsilon/2)\Big) , \\ z(t+\varepsilon) &= z(t) + \frac{\varepsilon}{2}\Big[\nabla_v H\Big(z(t), v(t+\varepsilon/2)\Big) + \nabla_v H\Big(z(t+\varepsilon), v(t+\varepsilon/2)\Big)\Big] , \\ v(t+\varepsilon) &= v(t+\varepsilon/2) - \frac{\varepsilon}{2}\nabla_z H\Big(z(t+\varepsilon), v(t+\varepsilon/2)\Big) , \end{aligned} \tag{1.5}$$

where ε is the integrator step size. Running K times this integrator allows to simulate the behavior of the particle. This integrator ensures that the target distribution is preserved by Hamiltonian dynamics and it was shown that it is also volume preserving and time reversible (Hairer et al., 2006; Leimkuhler and Reich, 2004). Inspired by this idea, the RHMC sampler aims at creating a Markov Chain (z^n) using this integrator and converging to the target distribution p_{target}. More precisely, given z_0^n, the current state of the chain, an initial *velocity* is sampled $v_0 \sim \mathcal{N}(0, \mathbf{G}(z_0^n))$ and Eq. (1.5) are run K times to move from (z_0^n, v_0) to (z_K^n, v_K). The proposal z_K^n is then accepted with probability $\alpha = \min\left(1, \frac{\exp(-H(z_K^n, v_K))}{\exp(-H(z_0^n, v_0))}\right)$ and we iterate. It was shown that the chain (z^n) converges to its stationary distribution p_{target} (Duane et al., 1987; Liu, 2008; Neal and others, 2011).

RHMC within the VAE

Likewise the HVAE, we set p_{target} to the joint distribution $p_\theta(x, z) = p_\theta(x|z)p(z)$ since given an input data point $x \in \mathcal{X}$ we have $p_\theta(x, z) \propto p_\theta(z|x)$ the true posterior and so the RHMC sampler is guided by the gradient of the true posterior distribution through the leapfrog steps in Eq. (1.5). Note that the target distribution is now tractable since both the prior and the conditional distribution $p_\theta(x|z)$ are known thanks to the assumed generation process:

$$\begin{cases} z \sim p(z) = \mathcal{N}(0, I_d), \\ x \sim p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma \cdot I_d) \, . \end{cases}$$

Hence, we can compute every terms of Eq. (1.4) and so use the generalized leapfrog integrator as proposed in the manuscript. A typical choice for ε and K is $\varepsilon \in [0.0001, 0.01]$ and $K \in [1, 15]$. As in (Caterini et al., 2018), we also use a tempering scheme consisting in starting from an initial temperature β_0 (which can be learned) and decreasing the *velocity* v by a factor $\alpha_k = \sqrt{\beta_{k-1}/\beta_k}$ after each leapfrog step k ($\beta_K = 1$). The temperature is then updated:

$$\sqrt{\beta_k} = \left(\left(1 - \frac{1}{\sqrt{\beta_0}}\right)\frac{k^2}{K^2} + \frac{1}{\sqrt{\beta_0}}\right)^{-1} .$$

As discussed in (Salimans et al., 2015), the acceptation/rejection step is omitted throughout training so that the flow is differentiable with respect to the encoder's parameters allowing optimization. Hence, the RHMC steps can be seen as a specific kind of normalizing flow informed both by the target distribution through Eq. (1.5) and by the latent space geometry thanks to the metric $\mathbf{G}$. Our intuition is that using the underlying geometry of the manifold in which the latent variables live would better guide the approximate posterior distribution leading to better variational posterior estimates. It must be nonetheless noted that the generalized *leapfrog* integrator in Eq. (1.5) is no longer explicit and so requires the use of fixed point iterations to be solved. Fortunately, only few iterations are needed to stabilize the scheme (we use 3 iterations). To compute the gradient involved in the integrator we rely on automatic differentiation (Paszke et al., 2017). Finally, the volume preservation property of the flow leads to a closed form derivation of the extended approximate posterior:

$$q_\phi(z_K, v_K|x) = q_\phi(z_0|x)p(v_0|z_0)\prod_{k=1}^{K}|\det \mathbf{J}_{g^k}|^{-1} = q_\phi(z_0|x)p(v_0|z_0)\prod_{k=1}^{K}\left(\frac{\beta_{k-1}}{\beta_k}\right)^{-d/2},$$

where $\mathbf{J}_{g^k}$ is the Jacobian of k^{th} leapfrog step. Now, an unbiased estimate of the marginal $p_\theta(x)$ is given by:

$$\hat{p}_\theta(x) = \frac{p_\theta(x, z_K, v_K)}{q_\phi(z_K, v_K|x)} = \frac{p_\theta(x|z_K)p(v_K|z_K)q(z_K)}{q_\phi(z_0|x)p(v_0|z_0)\beta_0^{-d/2}}.$$

Note that the expression of the variational posterior remains computable so that the ELBO remains tractable.

$$\text{ELBO}_{\text{Riemannian}} = \mathbb{E}_{(z_0,v_0)\sim q_\phi(\cdot,\cdot)}[\log \widehat{p}_\theta(x)] \tag{1.6}$$

The Metric

Since the latent space is now seen as the Riemannian manifold $(\mathbb{R}^d, \mathbf{G})$, it is in particular characterized by the Riemannian metric $\mathbf{G}$ whose choice is crucial. While several attempts have been made to try to put a Riemannian structure over the latent space of VAEs (Arvanitidis et al., 2018; Chen et al., 2018a; Shao et al., 2018; Frenzel et al., 2019; Kalatzis et al., 2020; Arvanitidis et al., 2020-08-02), the proposed metrics involved the Jacobian of the generator function which is hard to use in practice and is constrained by the generator network architecture. As a consequence, we instead decide to rely on the idea of Riemannian metric learning (Lebanon, 2006). Hence, we propose to use a parametric metric inspired from (Louis, 2019) as follows:

$$\mathbf{G}^{-1}(z) = \sum_{i=1}^{N} L_{\psi_i} L_{\psi_i}^\top \exp\left(-\frac{\|z - c_i\|_2^2}{T^2}\right) + \lambda I_d\,, \tag{1.7}$$

where N is the number of observations, L_{ψ_i} are lower triangular matrices with positive diagonal coefficients learned from the data and parametrized with neural networks, c_i are referred to as the *centroids* and correspond to the mean $\mu_\phi(x_i)$ of the encoded distributions of the latent variables z_i ($z_i \sim q_\phi(z_i|x_i) = \mathcal{N}(\mu_\phi(x_i), \Sigma_\phi(x_i))$, T is a temperature scaling the metric close to the *centroids* and λ is a regularization factor that also scales the metric tensor far from the latent codes. The shape of this metric is very powerful since we have access to a closed-form expression of the inverse metric tensor which is usually useful to compute shortest paths (through the exponential map). Moreover, this metric is very smooth, differentiable everywhere and allows scaling the Riemannian volume element $\sqrt{\det \mathbf{G}(z)}$ far from the data very easily through the regularization factor λ.

Training Process

The model's architecture is displayed in Fig. 1.1. The idea is to encode the input data points x_i and so get the means $\mu_\phi(x_i)$ of the posterior distributions associated with the encoded latent variables $z_{i,0} \sim \mathcal{N}(\mu_\phi(x_i), \Sigma_\phi(x_i))$. These means are then used to update the metric centroids c_i. In the mean time, the input data points x_i are fed to another neural network which outputs the matrices L_{ψ_i} used to update the metric. The updated metric is then used to *sample* $z_{i,K}$ from $z_{i,0}$ using Eq. (1.5) as explained in Sec. 1.3.2. The $z_{i,K}$ are then fed to the decoder network which outputs the parameters π_θ of the conditional distribution $p_\theta(x|z)$. The reparametrization trick is used to sample $z_{i,0}$ as is common and since the Riemannian Hamiltonian equations are *deterministic* with respect to z, back-propagation can be performed. A scheme of the *geometry-aware* VAE model framework can be found in Fig. 1.1. In the following, we will refer to the proposed model either as *geometry-aware VAE* or *RHVAE* for short. Finally, we also provide the full pseudo-code training algorithm of the method in Alg. 1.

Algorithm 1 RHVAE with metric learning

Initialize $\mathbf{G}$ ▷ We put $c_i = 0$ and $L_{\psi_i} = I_d$
while not converged **do**
  $\mathcal{L} \leftarrow 0$
  for $n = 1 \rightarrow N_B$ **do**
    Collect a batch of data $X_n = (x_1, \cdots, x_{\text{bs}})$
    $c_i \leftarrow \text{encode}(x_i)$
    $L_{\psi_i} \leftarrow m_\psi(x_i)$ ▷ Use the metric network to get L_{ψ_i}
    Update the metric $\mathbf{G}$ according to Eq. (1.7)
    $z_0 \sim \mathcal{N}(\mu(x), \Sigma(x))$, $v_0 \sim \mathcal{N}(0, \mathbf{G}(z_0))$
    $v_0 \leftarrow v_0/\sqrt{\beta_0}$
    for $k = 1 \rightarrow K$ **do**
      $\bar{v} \leftarrow v_{k-1} - \frac{\varepsilon}{2}\nabla_z H(z_{k-1}, \bar{v})$ ▷ fixed point it.
      $z_k \leftarrow z_{k-1} + \frac{\varepsilon}{2}\Big(\nabla_v H(z_{k-1}, \bar{v}) + \nabla_v H(z_k, \bar{v})\Big)$ ▷ fixed point it.
      $v' \leftarrow \bar{v} - \frac{\varepsilon}{2}\nabla_z H(z_k, \bar{v})$
      $\sqrt{\beta_k} \leftarrow \left(\left(1 - \frac{1}{\sqrt{\beta_0}}\right)\frac{k^2}{K^2} + \frac{1}{\sqrt{\beta_0}}\right)^{-1}$
      $v_k \leftarrow \frac{\sqrt{\beta_{k-1}}}{\sqrt{\beta_k}} v'$
    end for
    $p \leftarrow p_\theta(x, z_K, v_K)$
    $q \leftarrow q_\phi(z_0, v_0|x)\beta_0^{-d/2}$
    $\mathcal{L}_{\text{batch}} \leftarrow \log p - \log q$
    $\mathcal{L} = \mathcal{L} + \mathcal{L}_{\text{batch}}/N_B$
  end for
  Update θ, ϕ and ψ using gradient descent
end while

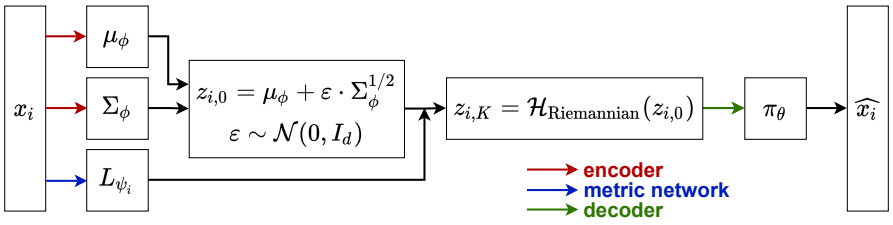


Figure 1.1: Geometry-aware VAE framework. Neural networks are highlighted with the colored arrows and $\mathcal{H}_{\text{Riemannian}}$ are the normalizing flows using Riemannian Hamiltonian equations.

Discussion on the Posterior Expressiveness

Theoretically, using *geometry-aware* Hamiltonian normalizing flows should conduct to a better estimate of the true posterior $p_\theta(z|x)$ and so a better ELBO leading to a potentially higher likelihood $p_\theta(x)$. To validate this empirically, we report the estimated log-likelihood computed using Importance Sampling with the approximate posterior and Eq. (1.2) and Eq. (1.6). We use 100 importance samples and compute it three times on MNIST test set. Hamiltonian based models use 3 leapfrog steps. We also report the value of the ELBO and compute $\mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x))$. As shown in Table 1.1, using *geometry-aware* normalizing flows leads to a higher estimated p_θ and a smaller gap between the estimated true posterior $p_\theta(z|x)$ and the variational approximation $q_\phi(z|x)$ measured by the KL divergence between both distributions. Note that all models are trained with the same architectures and training settings.

Table 1.1: Effect of geometrical considerations on the estimated log-likelihood and ELBO on MNIST test set.

Model	$\log p_\theta(x) \uparrow$	ELBO	$\mathrm{KL}(q_\phi(z\|x)\|p_\theta(z\|x)) \downarrow$
VAE	$-92.94(0.01)$	$-100.06(0.09)$	$7.12(0.09)$
HVAE	$-85.33(0.01)$	$-88.93(0.02)$	$3.61(0.02)$
RHVAE	$-82.64(0.01)$	$-86.21(0.04)$	$3.57(0.03)$

Sampling from the Latent Space

In this chapter, we propose to amend the standard sampling procedure of classic VAEs after training to better exploit the Riemannian structure of the latent space. The *geometry-aware* VAE is indeed here seen as a tool able to capture the intrinsic latent structure of the data and so we propose to exploit this property directly within the generation procedure. This differs greatly from the standard fully probabilistic view where the prior distribution is used to generate new data. We believe that such an approach remains far from being optimal when one considers small data sets since, depending on its choice, the prior may either poorly prospect the latent space or sample in locations without any usable information. In that respect, our approach can be seen as part of the recently proposed prior learning based methods or methods relying on *ex-post* density estimation discussed

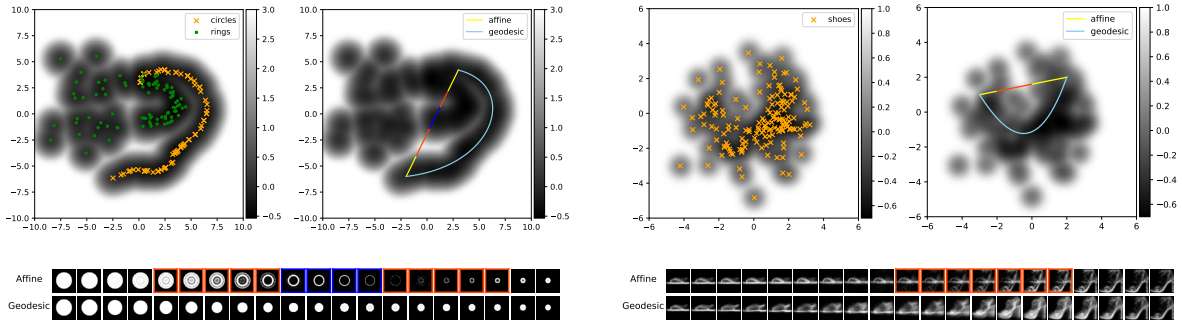


Figure 1.2: Geodesic interpolations under the learned metric in two different latent spaces. Top: Latent spaces with the log metric volume element presented in gray scale and the resulting interpolations under the Euclidean metric or the Riemannian metric. Bottom: Decoded samples all along the interpolation curves.

earlier. Some of these methods were indeed proposed on the ground that there may exist a mismatch between the chosen prior distribution $p(z)$ and the optimal one given by the the aggregated posterior distribution $q(z) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[q_\phi(z|x)]$ (Hoffman and Johnson, 2016; Dai and Wipf, 2018; Bauer and Mnih, 2019a; Ghosh et al., 2020), where $p_{\text{data}}(x)$ is the empirical distribution of the data (Tomczak and Welling, 2018). Moreover, since our method is mainly about increasing the expressiveness of the variational posterior q_ϕ there exists no apparent reason that the latent codes are distributed according to the prior either. However, instead of *learning* a prior, we propose to directly use the metric that provides information on the geometry of the latent space as discussed and illustrated in Sec. 1.3.2 and Sec. 1.3.3. We indeed propose to sample from the following distribution:

$$p(z) = \frac{\mathbf{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}}{\int_{\mathbb{R}^d} \mathbf{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)dz}}\,, \tag{1.8}$$

where S is a compact set[3] so that the integral is well defined. Fortunately, since we use a parametrized metric given by Eq. (1.7) and whose inverse has a closed form, it is pretty straightforward to evaluate the numerator of Eq. (1.8). Then, classic MCMC sampling methods can be employed to sample from p on $\mathbb{R}^d$. In this chapter, we propose to use the Hamiltonian Monte Carlo (HMC) sampler (Neal, 2005) since the gradient of the log-density is computable. We provide some additional details in Appendix 1.8.2.

Discussion on the Sampling Distribution

One may wonder what is the rationale behind the use of the distribution p formerly defined in Eq. (1.8). By design, the metric is such that the metric volume element $\sqrt{\det \mathbf{G}(z)}$ is scaled by the factor λ far from the encoded data points. Hence, choosing a relatively small λ imposes that shortest paths travel through the most populated area of the latent space, *i.e.* next to the latent codes. As such, the metric volume element can be seen as a way to quantify the amount of information contained at a specific location of the latent space. The smaller the volume element the more information we have access to. Fig. 1.2 illustrates well these aspects. On the first row are presented two learned latent

[3]Take for instance $\{z \in \mathcal{Z}, \|z\| \leq 2 \cdot \max_i \|c_i\|\}$

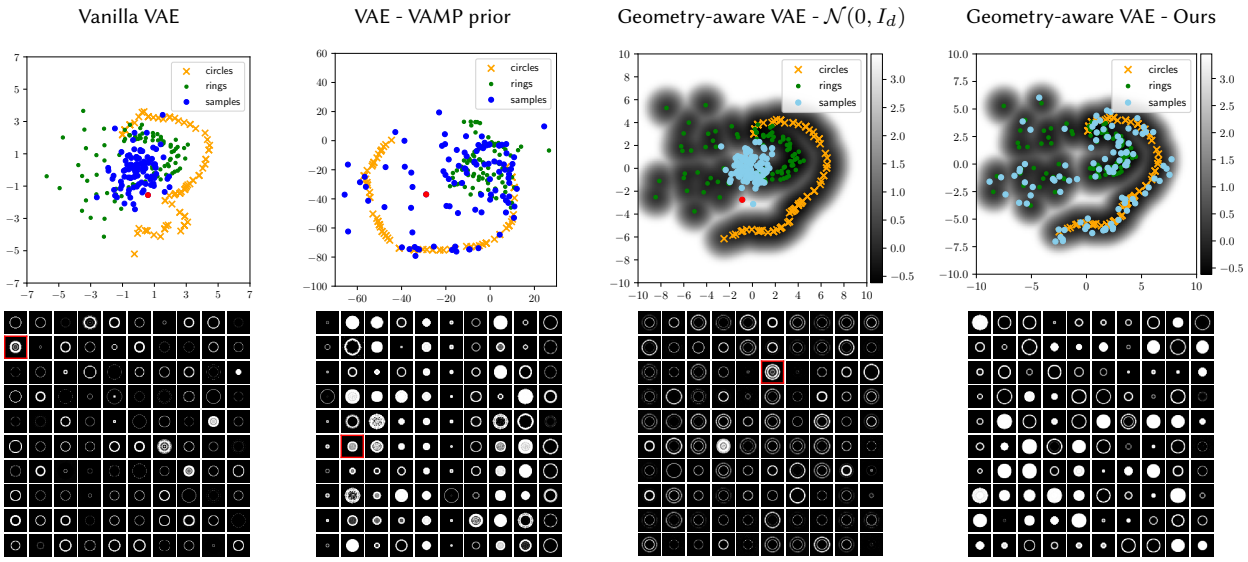


Figure 1.3: VAE sampling comparison. Top: The learned latent space along with the means $\mu_\phi(x_i)$ of the latent code distributions (colored dots and crosses) and 100 latent space samples (blue dots) using either the prior distribution or the proposed scheme. For the *geometry-aware* VAEs, the log metric volume element is presented in gray scale in the background. Bottom: The 100 corresponding decoded samples in the data space.

spaces along with the log of the metric volume element displayed in gray scale for two different data sets. The first one is composed of 180 binary disks and rings of different diameters and thicknesses while the second one is composed of 160 samples extracted from the FashionMNIST data set (Xiao et al., 2017). The means $\mu_\phi(x_i)$ of the distributions associated with the latent variables are presented with the crosses and dots for each class. As expected, the metric volume element is smaller close to the latent variables since small λ's were considered (10^{-3} resp. 10^{-1}). A common way to study the learned Riemannian manifold consists in finding geodesic curves, *i.e.* the shortest paths with respect to the learned Riemannian metric. Hence, in Fig. 1.2, we compare two types of interpolation in each latent space. For each experiment, we pick two points in the latent space and perform either a linear or a geodesic interpolation (*i.e.* using the Riemannian metric). The bottom row illustrates the decoded samples all along each interpolation curve. The first outcome of such an experiment is that, as expected, geodesic curves travel next to the codes and so do not explore areas of the latent space with no information whereas linear interpolations do. Therefore, decoding along geodesic curves produces far better and more meaningful interpolations in the input data space since in both cases we clearly see the starting sample being progressively distorted until the path reaches the ending point. This allows for instance interpolating between two shoes and keep the intrinsic topology of the data all along the path since each decoded sample on the interpolation curve looks like a shoe. This is made impossible under the Euclidean metric where shortest paths are straight lines and so may travel through areas of least interest. For instance, the affine interpolation travels through areas with no latent data and so produces decoded samples that are mainly a superposition of samples (see the red lines and corresponding decoded samples framed in red) or crosses areas with codes belonging to the other class (see the blue line and the corresponding blue frames). This study demonstrates that most of the information in the latent space is contained next to the codes and so, if we want

to generate new samples that look-like the input data, we need to sample around them and that is why we elected the distribution in Eq. (1.8). Noteworthy is the fact that likewise (Ghosh et al., 2020), the prior $\mathcal{N}(0, I_d)$ is now only reduced to a latent code regularizer during training ensuring that the covariances do not collapse to 0_d and the codes remains close to the origin and is never used to generate samples.

1.3.3 Generation Comparison

In this section, we propose to compare the new generation procedure with other generation methods in the context of low sample size data sets.

Qualitative Comparison

First, we validate the proposed generation method on a hand-made synthetic data set composed of 180 binary disks and rings of different diameters and thicknesses (see Appendix 1.8.4). We then train 1) a vanilla VAE, 2) a VAE with VAMP prior (Tomczak and Welling, 2018), 3) a *geometry-aware* VAE but using the prior to generate and 4) a *geometry-aware* VAE with the proposed generation scheme, and compare the generated samples. Each model is trained until the ELBO does not improve for 20 epochs and any relevant parameter setting is made available in Appendix 1.8.3. In Fig. 1.3, we compare the sampling obtained with each model. The first row shows the learned latent spaces along with the means of the encoded training data points for each class (crosses and dots) and 100 samples issued by the generation methods (blue dots). For the RHVAE models, the log metric volume element $\sqrt{\det \mathbf{G}}$ is also displayed in gray scale in the background. The bottom row shows the resulting 100 decoded samples in the data space.

The first outcome of this experiment is that sampling from the prior distribution leads to a quite poor latent space prospecting. This drawback is very well illustrated when a standard Gaussian distribution is used to sample from the latent space (see 1^{st} and 3^{rd} column of the 1^{st} row). The prior distribution having a higher mass close to zero will insist on latent samples close to the origin. Unfortunately, in such a case, latent codes close to the origin only belong to a single class (rings). Therefore, even though the number of training samples was roughly the same for disks and rings, we end up with a model over-generating samples belonging to a certain class (rings) and even to a specific type of data within this very class. This undesirable effect seems even ten-folded when considering the *geometry-based* VAE model since adding MCMC steps in the training process, as explained in Fig. 1.1, tends to stretch the latent space. It can be nonetheless noted that using a multi-modal prior such as the VAMP prior mitigates this and allows for a better prospecting. However, such a model remains hard to fit when trained with small data sets as it may overfit (resp. underfit) the training samples if the number of pseudo-inputs is too high (resp. low). Another limitation of prior-based generation methods lies in their inability to assess a given sample quality. They may indeed sample in areas of the latent space containing very few information and so conduct to generated samples that are meaningless. This appears even more striking when small data sets are considered. An interesting observation that was noted among others in (Arvanitidis et al., 2018) is that neural networks tend to interpolate very poorly in *unseen* locations (i.e. far from the training data points). When looking at the *decoded* latent samples (bottom row of Fig. 1.3) we eventually end up with the same conclusion. Actually, it appears that the networks interpolate quite linearly be-

tween the training data points in our case. This may be illustrated for instance by the red dots in the latent spaces in Fig. 1.3 whose corresponding decoded sample is framed in red. The sample is located *between* two classes and when decoded it produces an image mainly corresponding to a superposition of samples belonging to different classes. This aspect is also supported by the observations made when discussing the relevance of geodesic interpolations on Fig. 1.2 of Sec. 1.3.2. Therefore, these drawbacks may conduct to a (very) poor representation of the actual data set diversity while presenting quite a few *irrelevant* samples. Obviously the notion of *irrelevance* is here disputable but if the objective is to represent a given set of data we expect the generated samples to be close to the training data while having some specificities to enrich it. Impressively, sampling against the inverse of the metric volume element as proposed in Sec. 1.3.2 allows for a far more meaningful sample generation. Furthermore, the new sampling scheme avoids regions with no latent code, which thus contain poor information, and focuses on areas of interest so that almost every decoded sample is visually satisfying. Similar effects are observed on reduced versions of EMNIST (Cohen et al., 2017), MNIST (LeCun, 1998) and FashionMNIST data sets and higher dimensional latent spaces (dimension 10) where samples are most of the time degraded when the classic generation is employed while the new one allows the generation of more diverse and sharper samples (see Appendix 1.8.4). Finally, the proposed method does not overfit the train data since the samples are not located on the centroids. The quantitative metrics of the next section also support this point.

Table 1.2: *GAN-train* (the higher the better) and *GAN-test* (the closer to the baseline the better) scores. A benchmark DenseNet model is trained with five independent runs on the generated data $\mathcal{S}_g$ (resp. the *real* train set $\mathcal{S}_{\text{train}}$) and tested on the *real* test set $\mathcal{S}_{\text{test}}$ (resp. $\mathcal{S}_g$) to compute the *GAN-train* (resp. *GAN-test*) score. 1000 synthetic samples per class are considered for $\mathcal{S}_g$ so that it matches the size of $\mathcal{S}_{\text{test}}$.

	reduced MNIST (balanced)		*reduced* MNIST (unbalanced)		*reduced* EMNIST	
Metric	GAN-train	GAN-test	GAN-train	GAN-test	GAN-train	GAN-test
Baseline	90.6 ± 1.2	-	82.8 ± 0.7	-	84.5 ± 1.3	-
VAE - $\mathcal{N}(0, I_d)$	83.4 ± 2.4	67.1 ± 4.9	74.7 ± 3.2	52.8 ± 10.6	75.3 ± 1.4	54.5 ± 6.5
VAMP	72.8 ± 6.7	77.6 ± 4.8	68.2 ± 6.6	76.7 ± 11.0	70.7 ± 8.0	69.0 ± 6.4
VAE - GMM	82.9 ± 2.4	76.5 ± 8.9	74.4 ± 3.8	68.4 ± 12.3	74.0 ± 2.6	57.6 ± 4.6
RAE - GMM(2)	90.8 ± 3.0	91.7 ± 1.9	85.5 ± 1.3	83.8 ± 6.2	80.3 ± 1.5	69.8 ± 7.2
RAE - GMM(10)	90.3 ± 2.3	$\underline{95.3} \pm 1.6$	81.0 ± 4.4	$\underline{93.3} \pm 3.2$	80.6 ± 1.6	83.4 ± 4.8
RAE - GMM(20)	$\mathbf{91.1 \pm 1.6}$	$\underline{96.6} \pm 1.5$	84.3 ± 1.7	$\underline{95.4} \pm 3.1$	79.5 ± 1.1	$\mathbf{85.0 \pm 4.8}$
2-stage VAE	84.8 ± 2.3	71.4 ± 8.3	80.8 ± 2.7	60.2 ± 9.2	79.6 ± 2.3	55.9 ± 3.9
RHVAE - $\mathcal{N}(0, I_d)$	82.0 ± 2.9	63.1 ± 4.1	69.3 ± 1.8	46.9 ± 8.4	73.6 ± 4.1	55.6 ± 5.0
Ours	90.1 ± 1.4	$\mathbf{88.1 \pm 2.7}$	$\mathbf{86.2 \pm 1.8}$	$\mathbf{83.8 \pm 4.0}$	$\mathbf{82.6 \pm 1.3}$	76.0 ± 4.0

Quantitative Comparison

In order to compare quantitatively the diversity and relevance of the samples generated by a generative model, several metrics were proposed (Salimans et al., 2016; Heusel et al., 2017; Karras et al., 2017; Lucic et al., 2018). Since they suffer from some drawbacks (Shmelkov et al., 2018; Borji, 2019), we decide to use the *GAN-train / GAN-test* measure discussed in (Shmelkov et al., 2018) as it appears to us well suited to measure the ability of a generative model to perform data augmentation. These two metrics consist in comparing the accuracy of a benchmark classifier trained on a set of generated data $\mathcal{S}_g$ and tested on a set of *real* images $\mathcal{S}_{\text{test}}$ (*GAN-train*) or trained on the original train set $\mathcal{S}_{\text{train}}$ (*real* images used to train the generative model) and tested on $\mathcal{S}_g$ (*GAN-test*). Those accuracies are then compared to the baseline accuracy given by the same classifier trained on $\mathcal{S}_{\text{train}}$ and tested on $\mathcal{S}_{\text{test}}$. These two metrics are quite interesting for our application since the first one (*GAN-train*) measures the quality and diversity of the generated samples (the higher the better) while the second one (*GAN-test*) accounts for the generative model's tendency to overfit (a score significantly higher than the baseline accuracy means overfitting). Ideally, the closer to the baseline the *GAN-test* score the better. To stick to our low sample size setting, we compute these scores on three data sets created by down-sampling well-known databases. The first data set is created by extracting 500 samples from MNIST ensuring balanced classes (*reduced* MNIST). For the second one, 500 samples of the MNIST database are again considered but a random split is applied such that some classes are under-represented (*reduced* unbalanced MNIST). The last one consists in selecting 500 samples from 10 classes of the EMNIST data set having both lowercase and uppercase letters (*reduced* EMNIST) so that we end up with a small database with strong variability within classes. The balance matches the one in the initial data set (*by merge*). These three data sets are then divided into a baseline train set $\mathcal{S}_{\text{train}}$ (80%) and a validation set $\mathcal{S}_{\text{val}}$ (20%) used for the classifier training. Since the initial databases are huge, we use the original test set for $\mathcal{S}_{\text{test}}$ so that it provides statistically meaningful results. For this comparison, we add a regularized autoencoder (RAE) (Ghosh et al., 2020), a 2-stage VAE (Dai and Wipf, 2018) and a VAE where we use a 10-components mixture of Gaussian (GMM) instead of the prior to generate (Ghosh et al., 2020), to the models presented in Sec. 1.3.3. Each model is then trained on each class of $\mathcal{S}_{\text{train}}$ to generate 1000 samples per class and $\mathcal{S}_g$ is created for each VAE by gathering all generated samples. A benchmark classifier chosen as a DenseNet[4] (Huang et al., 2017) is then 1) trained on $\mathcal{S}_{\text{train}}$ and tested on $\mathcal{S}_{\text{test}}$ (*baseline*); 2) trained on $\mathcal{S}_g$ and tested on $\mathcal{S}_{\text{test}}$ (*GAN-train*) and 3) trained on $\mathcal{S}_{\text{train}}$ and tested on $\mathcal{S}_g$ (*GAN-test*) until the loss does not improve for 50 epochs on $\mathcal{S}_{\text{val}}$. For each experiment, the model is trained five times and we report the mean score and the associated standard deviation in Table 1.2. For the RAE we use a GMM and indicate the number of components between parentheses. As expected, the proposed method allows producing samples that are far more meaningful and relevant, in particular to perform DA. This is first illustrated by the *GAN-train* scores that are either very close to the accuracy obtained with the *baseline* or higher (see MNIST (unbalanced) in Table 1.2). The fact that we are able to enhance the classifier's accuracy even when trained only with synthetic data is very encouraging. Firstly, it proves that the created samples are close to the *real* ones and so we were able to capture the true distribution of the data. Secondly, it shows that we do not overfit the initial training data since we are able to add some relevant information through the synthetic samples. This last observation is also supported by the *GAN-test* scores for the proposed method which are quite close to the accuracies achieved on the *baseline*. In case of overfitting, the *GAN-test* score would be significantly higher than the *baseline* since the classifier is tested on the generated samples while trained on the *real* data that

[4]We used the PyTorch implementation provided in (Amos, 2020).

were also used to train the generative model. This is for instance the case for the RAE (underlined scores) where the number of components in the GMM impacts greatly the *GAN-test* metric. Having a score close to the *baseline* illustrates that the generative model is able to capture the distribution of the data and does not only *memorize* it (Shmelkov et al., 2018). Finally, this study again shows the relevance of considering new ways to generate data from VAEs, such as fitting a mixture of Gaussian in the latent space, using a 2-stage VAE or using the proposed method, as they all improve in almost all cases the metrics when compared to prior-based methods (lines 2 and 9 of Table 1.2).

1.4 Data Augmentation: Evaluation and Robustness

In this section we show the relevance of the proposed improvements to perform data augmentation in a HDLSS setting through a series of experiments.

1.4.1 Setting

The setting we employ for DA consists in selecting a data set and splitting it into a train set (the *baseline*), a validation set and a test set. The *baseline* is then augmented using the proposed VAE framework and generation procedure. The generated samples are finally added to the original train set (*i.e.* the *baseline*) and fed to a classifier. The whole data augmentation procedure is illustrated in Fig. 1.4.

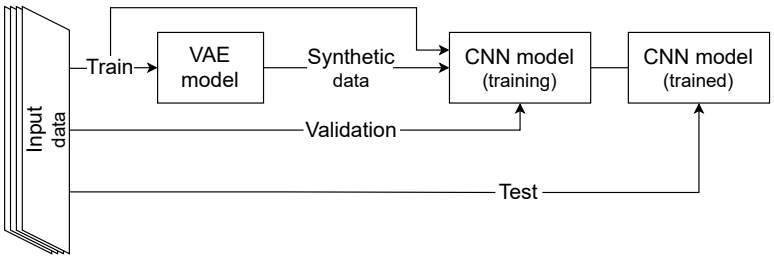


Figure 1.4: Overview of the data augmentation procedure. The input data set is divided into a train set (the *baseline*), a validation set and a test set. The train set is augmented using the VAE framework and generated data are then added to the *baseline* to train a benchmark classifier.

1.4.2 Toy Data Sets

The proposed VAE framework is here used to perform DA on several down-sampled well-known databases such that only tens of *real* training samples per class are considered so that we stick to the low sample size setting. First, the robustness of the method across these data sets is tested with a standard benchmark classifier. Then, its reliability across other common classifiers is stressed. Finally, its scalability to larger data sets and more complex models is discussed.

Materials

In this section, we use the same three data sets described in Sec. 1.3.3 and add one using the Fashion-MNIST data set and three classes we find hard to distinguish (*i.e. T-shirt*, *dress* and *shirt*). The data set is composed of 300 samples ensuring balanced classes (*reduced* Fashion). Finally, we also select 150 samples from three balanced classes of CIFAR10 (Krizhevsky et al., 2009) hard to classify (*cat*, *dog* and *horse*). In summary, we built five data sets having different class numbers, class splits and sample sizes. These data sets are again pre-processed such that 80% is allocated for training (referred to as the *Baseline*) and 20% for validation. Since the original data sets are huge, we use the test set provided in the original databases (*e.g.* $\approx$1000 samples per class for MNIST) so that it provides statistically meaningful results while allowing for a reliable assessment of the model's generalization power on unseen data.

Robustness Across Data Sets

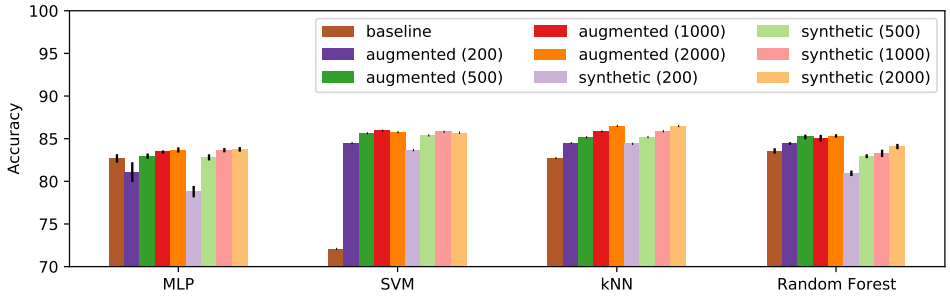
The first experiment we conduct consists in assessing the method's robustness across the five aforementioned data sets. For this study, we propose to consider a DenseNet model as benchmark classifier. On the one hand, the training data (the *baseline*) is augmented by a factor 5, 10 and 15 using classic data augmentation methods (random noise, random crop, rotation, etc.) so that the proposed method can be compared with classic and simple augmentation techniques. On the other hand, the protocol described in Fig. 1.4 is employed with the same VAEs as before. The generative models are trained individually on each class of the *baseline* until the ELBO does not improve for 20 epochs. The VAEs are then used to produce 200 or 1000 new synthetic samples per class using the same generation protocols as described in Sec. 1.3.3. Finally, the benchmark DenseNet model is trained with five independent runs on either 1) the *baseline*, 2) the augmented data using classic augmentation methods, 3) the augmented data using the VAEs or 4) only the synthetic data created by the generative models. For each experiment, the mean accuracy and the associated standard deviation across those five runs are reported in Table 1.3. An early stopping strategy is employed and CNN training is stopped if the loss does not improve on the validation set for 50 epochs.

The first outcome of such a study is that, as expected, generating synthetic samples with the proposed method seems to enhance their relevance in particular for data augmentation tasks. This is for instance illustrated by the first section of Table 1.3 where synthetic samples are added to the *baseline*. While adding samples generated either by a VAE or RHVAE and using the prior distribution seems to improve the classifier accuracy when compared with the *baseline*, the gain remains limited since it struggles to exceed the gain reached with classic augmentation methods. On the contrary, methods using either more complex priors (VAMP), a second VAE or a GMM allow improving classification results on MNIST and Fashion but still under-perform on EMNIST and CIFAR. Finally, the proposed generation method is able to produce very useful samples for the CNN model since in all cases it allows the classifier to either achieve the best result (highlighted in bold) or comparable performance than peers while keeping a relatively low standard deviation. Secondly, the relevance of the samples produced by the proposed scheme is even more supported by the second section of Table 1.3 where the classifier is trained only using the synthetic samples generated by the VAEs. First, even with a quite small number of samples generated with our method (200 per class), the classifier is almost able to reach the accuracy achieved with the *baseline*. For instance, when the CNN is trained on *reduced* MNIST with 200 synthetic samples per class generated with our method, it is

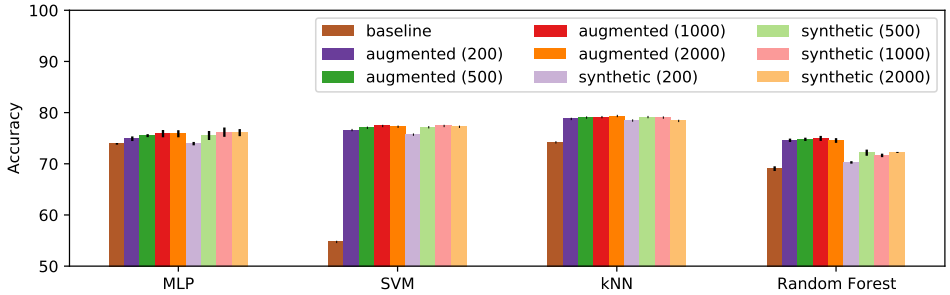
Table 1.3: Data augmentation with a DenseNet model as benchmark. Mean accuracy and standard deviation across five independent runs are reported. The first three rows (Aug.) correspond to basic transformations (noise, crop, etc.). In gray are the cells where the accuracy is higher on synthetic data than on the *baseline* (*i.e.* the raw data). The test set is the one proposed in the entire original data set (*e.g.* ≈1000 samples per class for MNIST) so that it provides statistically meaningful results and allows for a good assessment of the model's generalization power.

	MNIST	MNIST (unbal.)	EMNIST (unbal.)	FASHION	CIFAR
			Baseline + Synthetic		
Baseline	89.9/0.6	81.5/0.7	82.6/1.4	76.0/1.5	42.6/7.6
Aug. (×5)	92.8/0.4	86.5/0.9	85.6/1.3	77.5/2.0	47.7/2.3
Aug. (×10)	88.2/2.2	82.0/2.4	85.7/0.3	79.2/0.6	48.2/1.7
Aug. (×15)	92.8/0.7	85.8/3.4	86.6/0.8	80.0/0.5	48.0/2.2
VAE - 200	88.5/0.9	84.0/2.0	81.7/3.0	78.6/0.4	46.9/1.3
VAE - 1k	91.2/1.0	86.0/2.5	84.3/1.6	77.6/2.1	47.7/1.4
VAMP -200	91.4/1.9	81.1/2.7	84.2/0.8	79.8/0.8	45.6/6.9
VAMP - 1k	**93.6/0.9**	88.0/1.1	86.2/1.1	79.6/0.4	45.2/6.1
RHVAE - 200*	89.9/0.5	82.3/0.9	83.0/1.3	77.6/1.3	45.2/1.9
RHVAE - 1k*	91.7/0.8	84.7/1.8	84.7/2.4	79.3/1.6	42.1/2.9
VAE GMM - 200	90.5/1.1	82.9/2.2	84.8/1.0	79.6/0.7	44.9/1.9
VAE GMM - 1k	92.0/1.8	86.7/1.0	86.1/1.1	79.5/0.7	38.9/2.4
2-stage VAE - 200	91.2/1.2	83.5/1.5	85.3/1.9	**80.5/0.6**	44.4/2.3
2-stage VAE - 1k	93.3/0.7	87.7/2.4	86.7/1.1	79.5/0.9	38.8/3.0
RAE - 200	91.6/1.1	81.3/1.3	85.2/0.9	80.1/0.8	46.2/2.9
RAE - 1k	93.3/0.8	88.3/1.1	85.8/0.9	79.8/1.3	44.1/2.6
Ours - 200	91.0/1.0	84.1/2.0	85.1/1.1	77.0/0.8	46.8/2.2
Ours - 1k	93.2/0.8	**89.7/0.8**	**87.0/1.0**	80.2/0.8	**49.2/2.3**
			Synthetic Only		
VAE - 200	69.9/1.5	64.6/1.8	65.7/2.6	73.9/3.0	40.5/4.1
VAE - 1k	83.4/2.4	74.7/3.2	75.3/1.4	71.4/6.1	41.3/2.4
VAMP - 200	61.3/3.2	52.4/3.0	67.4/1.4	70.4/3.2	40.6/6.6
VAMP - 1k	72.8/6.7	68.2/6.6	70.7/8.0	69.2/5.4	39.7/7.7
RHVAE - 200*	76.0/1.8	61.5/2.9	59.8/2.6	72.8/3.6	42.4/1.2
RHVAE - 1k*	82.0/2.9	69.3/1.8	73.6/4.1	76.0/4.1	40.7/3.2
VAE GMM - 200	76.5/1.5	64.0/2.6	70.5/1.5	71.9/2.2	38.7/4.2
VAE GMM - 1k	82.9/2.4	74.4/3.8	74.0/2.6	73.9/2.5	41.6/2.7
2-stage VAE - 200	82.3/1.1	74.9/2.3	76.7/1.3	76.2/2.0	38.1/2.6
2-stage VAE - 1k	84.8/2.3	80.8/2.7	79.6/2.3	75.8/1.8	37.9/3.6
RAE - 200	83.6/2.8	74.5/1.6	76.9/1.6	66.5/4.4	33.7/1.7
RAE - 1k	**90.3/2.3**	81.0/4.4	80.6/1.6	62.0/5.1	33.6/0.4
Ours - 200	87.2/1.1	79.5/1.6	77.0/1.6	77.0/0.8	**47.3/1.7**
Ours - 1k	90.1/1.4	**86.2/1.8**	**82.6/1.3**	**79.3/0.6**	46.7/3.1

able to achieve an accuracy of 87.2% vs. 89.9% with the *baseline*. In comparison, any other method fails to produce meaningful samples since a quite significant loss in accuracy is observed. The fact that the classifier almost performs as well on the synthetic data as on the *baseline* is good news since it shows that the proposed framework is able to produce samples accounting for the original data set diversity even with a small number of generated samples. Even more interesting, as the number of synthetic data increases, the classifier is able to perform much better on the synthetic data than on the *baseline* since a gain of 3 to 6 points in accuracy is observed. Again, this strengthens the



(a) *reduced* MNIST balanced



(b) *reduced* MNIST unbalanced

Figure 1.5: Evolution of the accuracy of four benchmark classifiers on *reduced* balanced MNIST (left) and *reduced* unbalanced MNIST data sets (right). Stochastic classifiers are trained with five independent runs and we report the mean accuracy and standard deviation on the test set.

observations made in Sec. 1.3.3 and 1.3.3 where we noted that **the proposed method is able to enrich the initial data set** with relevant and realistic samples.

Finally, it can be seen in this experiment why geometric data augmentation methods are still questionable and remain data set dependent. For example, augmenting the *baseline* by a factor 10 (where we add flips and rotations on the original data) seems to have no significant effect on the *reduced* MNIST data sets while it still improves results on *reduced* EMNIST, Fashion and CIFAR. We see here how the *expert* knowledge comes into play to assess the relevance of the transformations applied to the data. Fortunately, the method we propose does not require such knowledge and **appears to be quite robust to data set changes.**

Robustness Across Classifiers

In addition to assessing the robustness of the method to data sets changes, we also propose to evaluate its reliability across classifiers. To do so, we consider very different common supervised classifiers: a multi layer perceptron (MLP) (Goodfellow et al., 2016), a random forest (Breiman, 2001), the k-NN algorithm and a SVM (Kotsiantis et al., 2007). Each of the aforementioned classifiers is again trained either on 1) the original training data set (the *baseline*); 2) the augmented data using the proposed method and 3) only the synthetic data generated by our method with five independent runs and using the same data sets as presented in Sec. 1.4.2. Finally, we report the mean accuracy and standard deviation across these runs for each classifier and data set. The results for the balanced

(resp. unbalanced) *reduced* MNIST data set can be found in Fig. 1.5a (resp. Fig. 1.5b). Metrics obtained on *reduced* EMNIST and Fashion are available in Appendix 1.8.5 but reflect the same tendency.

As illustrated in Fig. 1.5, the method appears quite robust to classifier changes as well since it allows improving the model's accuracy significantly for almost all classifiers (the accuracy achieved on the *baseline* is represented by the leftmost bar in Fig. 1.5 for each classifier). The method's strength is even more striking when unbalanced data sets are considered since the method is able to produce meaningful samples even with a very small number of training data and so it is able to over-sample the minority classes in a reliable way. Moreover, as observed earlier, synthetic samples are again helpful to enhance classifiers' generalization power since they perform better when trained only on synthetic data than on the *baseline* in almost all cases.

A Note on the Method Scalability

Finally, we also discuss the method scalability to larger data sets, bigger models and higher dimensional latent spaces. To do so, we consider the MNIST data set and a benchmark classifier taken as a DenseNet which performs well on such data. First, we down-sample the original MNIST database in order to progressively decrease the number of samples per class. We start by creating a data set having 1000 samples per class to finally reach 20 samples per class. For each created data set, we allocate 80% for training (the *baseline*) and reserve 20% for the validation set. A *geometry-aware* VAE is then trained on each class of the *baseline* until the ELBO does not improve for 50 epochs and is used to generate synthetic samples (12.5$\times$ the *baseline*). The benchmark CNN is trained with five independent runs on either 1) the *baseline*, 2) the augmented data or 3) only the synthetic data generated with our model. The evolution of the mean accuracy on the original test set ($\approx$1000 samples per class) according to the number of samples per class is presented in Fig. 1.6 (left). Second, we only consider 50 samples per class and train the VAE on each class to generate 1000 samples per class. The number of the classifier's parameters is also progressively changed and we report the mean accuracy of the CNN according to the number of parameters in Fig. 1.6 (middle). Finally, we consider several latent space dimensions for the VAE ranging from 2 to 50 and plot the evolution of the CNN accuracy according the latent space dimension (right).

First, this experiment shows that the fewer samples in the training set, the more useful the method appears. Using the proposed augmentation framework indeed allows for a gain of more than 9.0 points in the CNN accuracy when only 20 samples per class are considered. In other words, as the number of samples increases, the marginal gain seems to decrease. Nevertheless, this reduction must be put into perspective since it is commonly acknowledged that, as the results on the *baseline* increase (and thus get closer to the perfect score), it is even more challenging to improve the score with the augmented data. In this experiment, we are nonetheless still able to improve the model accuracy even when it already achieves a very high score. For instance, with 500 samples per class, the augmentation method still allows increasing the model accuracy from 97.7% to 98.8%. Finally, for data sets with fewer than 500 samples per class, the classifier is able to outperform the *baseline* even when trained only with the synthetic data. This shows again the strong generalization power of the proposed method which allows creating new relevant data for the classifier. Another interesting take from these experiments is that the augmentation method seems to benefit both simple and more complex models since the gain in the model accuracy remains quite steady ($\approx$ 3 pts) regardless of the number of parameters in the classifier (Fig. 1.6 (middle)). Finally, the impact of the dimension of

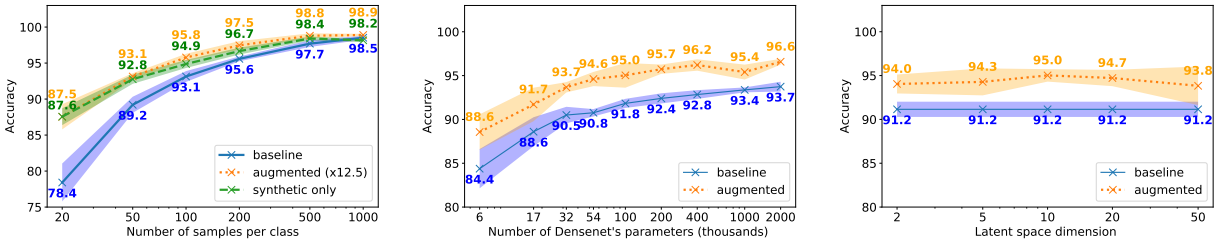


Figure 1.6: Evolution of the accuracy of a benchmark DenseNet classifier according to the number of samples in the train set (*i.e.* the *baseline*) (*left*), the number of parameters of the Densenet (*middle*) and the latent space dimension of the VAE (*right*) on MNIST. Curves show the mean accuracy and standard deviation across 5 runs on the original test set for the *baseline* (blue), the augmented data (orange) and the synthetic ones (green).

the latent space remains limited for such a framework as the classification accuracy remains stable. Nonetheless, this may be due to the simplicity of the database and more complex data might need higher dimensional latent spaces.

1.5 Validation on Medical Imaging

With this last series of experiments, we assess the validity of our data augmentation framework on a binary classification task consisting in differentiating Alzheimer's disease (AD) patients from cognitively normal (CN) subjects based on T1-weighted (T1w) MR images of human brains. Such a task is performed using a CNN trained, as before, either on 1) *real* images, 2) synthetic samples or 3) both. In this section, label definition, preprocessing, quality check, data split and CNN training and evaluation is done using Clinica[5] (Routier et al., 2021) and ClinicaDL[6] (Thibeau-Sutre et al., 2022), two open-source software packages for neuroimaging processing.

1.5.1 Data Augmentation Literature for AD vs CN Task

Even though many studies use CNNs to detect AD from CN subjects with anatomical MRI (Wen et al., 2020), we did not find any meta-analysis on the use of data augmentation for this task. Some results involving DA can nonetheless be cited and are presented in Table 1.4. However, assessing the real impact of data augmentation on the performance of the models remains challenging. For instance, this is illustrated by the works of (Aderghal et al., 2017) and (Aderghal et al., 2018), which are two examples in which DA was used and led to two significantly different results, although a similar framework was used in both studies. Interestingly, as shown in Table 1.4, studies using DA for this task only relied on simple affine and pixel-level transformations, which may reveal data dependent. Note that complex DA was actually performed for AD vs CN classification tasks on PET images, but PET is less frequent than MRI in neuroimaging data sets (Islam and Zhang, 2020). As noted in the previous sections, our method would apply pretty straightforwardly to this modality as

[5] https://github.com/aramis-lab/clinica
[6] https://github.com/aramis-lab/clinicadl

Table 1.4: Accuracy obtained by studies performing AD vs CN classification with CNNs applied on T1w MRI and using data augmentation

Study	Methods	Subj.	Images	Accuracy	
				Baseline	Augmented
(Valliani and Soni, 2017)	rotate, flip, shift	417	417	78.8	81.3
(Backstrom et al., 2018)	flip	340	1198	–	90.1
(Cheng and Liu, 2017)	shift, sample, rotate	193	193	–	85.5
(Aderghal et al., 2017)	shift, blur, flip	720	720	82.8	83.7
(Aderghal et al., 2018)	shift, blur	720	720	–	90.0

well. For MRI, other techniques such as transfer learning (Oh et al., 2019) and weak supervision (Liu et al., 2020) were preferred to handle the small amount of samples in data sets and may be coupled with DA to further improve the classifier performance.

1.5.2 Materials

Data used in this section were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu) and the Australian Imaging, Biomarkers and Lifestyle (AIBL) study (aibl.csiro.au).

The ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial MRI, PET, other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment and early AD. For up-to-date information, see www.adni-info.org. The ADNI data set is composed of four cohorts: ADNI-1, ADNI-GO, ADNI-2 and ADNI-3. The data collection of ADNI-3 has not ended yet, hence our data set contains all images and metadata that were already available on May 6, 2019. Similarly to ADNI, the AIBL data set seeks to discover which biomarkers, cognitive characteristics, and health and lifestyle factors determine the development of AD. This cohort is also longitudinal and the diagnosis is given according to a series of clinical tests (Ellis et al., 2009). Data collection for this cohort is over.

Two diagnoses are considered for the classification task:

- CN: baseline session of participants who were diagnosed as cognitively normal at baseline and stayed stable during the follow-up;
- AD: baseline session of participants who were diagnosed as demented at baseline and stayed stable during the follow-up.

Table 1.5 summarizes the demographics, the mini-mental state examination (MMSE) and global clinical dementia rating (CDR) scores at baseline of the participants included in our data set. The MMSE

Table 1.5: Summary of participant demographics, mini-mental state examination (MMSE) and global clinical dementia rating (CDR) scores at baseline.

Data set	Label	Subj.	Age	Sex M/F	MMSE	CDR
ADNI	CN	403	73.3 ± 6.0	185/218	29.1 ± 1.1	0: 403
	AD	362	74.9 ± 7.9	202/160	23.1 ± 2.1	0.5: 169, 1: 192, 2: 1
AIBL	CN	429	73.0 ± 6.2	183/246	28.8 ± 1.2	0: 406, 0.5: 22, 1: 1
	AD	76	74.4 ± 8.0	33/43	20.6 ± 5.5	0.5: 31, 1: 36, 2: 7, 3: 2

and the CDR scores are classical clinical scores used to assess dementia. The MMSE score has a maximal value of 30 for cognitively normal persons and decreases if symptoms are detected. The CDR score has a minimal value of 0 for cognitively normal persons and increases if symptoms are detected.

1.5.3 Preprocessing of T1-Weighted MRI

The steps performed in this section correspond to the procedure followed in (Wen et al., 2020) and are listed below:

1. Raw data are converted to the BIDS standard (Gorgolewski et al., 2016),
2. Bias field correction is applied using N4ITK (Tustison et al., 2010a),
3. T1w images are linearly registered to the MNI standard space (Fonov et al., 2009; 2011) with ANTS (Avants et al., 2014) and cropped. This produced images of size $169 \times 208 \times 179$ with 1 mm^3 isotropic voxels.
4. An automatic quality check is performed using an open-source pretrained network (Fonov et al., 2018). All images passed the quality check.
5. NIfTI files are converted to tensor format.
6. (Optional) Images are down-sampled with a trilinear interpolation to a size of $84 \times 104 \times 89$.
7. Intensity rescaling between the minimum and maximum values of each image is performed.

These steps lead to either 1) down-sampled images ($84 \times 104 \times 89$) or 2) high resolution images of size $169 \times 208 \times 179$.

1.5.4 Evaluation Procedure

The ADNI data set is split into three sets: training, validation and test. First, the test set is created using 100 randomly chosen participants for each diagnostic label (i.e. 100 CN, 100 AD). The rest of the data set is split between the training (80%) and the validation (20%) sets. We ensure that age, sex and site distributions between the three sets are not significantly different.

A smaller training set (denoted as *train-50*) is extracted from the obtained training set (denoted as *train-full*). This set comprises only 50 images per diagnostic label, instead of 243 CN and 210 AD for *train-full*. We ensure that age and sex distributions between *train-50* and *train-full* are not significantly different. This is not done for the site distribution as there are more than 50 sites in the ADNI data set (so they could not all be represented in this smaller training set). AIBL data are never used for training or hyperparameter tuning and are only used as an independent test set.

1.5.5 CNN Classifiers

A CNN takes as input an image and outputs a vector of size C corresponding to the number of labels existing in the data set. Then, a CNN predicts the label of a given image by selecting the highest probability in the output vector.

Hyperparameter Choices

As for the VAE, the architecture of the CNN depends on the size of the input. Then, there is one architecture per input size: down-sampled images and high-resolution images (see Fig 1.7). Moreover, two different paradigms are used to choose the architecture. First, we reuse the same architecture as in (Wen et al., 2020). This architecture was obtained by optimizing manually the networks on the ADNI data set for the same task (AD vs CN). A slight adaption is done for the down-sampled images, which consists in resizing the number of nodes in the fully-connected layers to keep the same ratio between the input and output feature maps in all layers. We denote these architectures as **baseline**. Secondly, we launch a random search (Bergstra and Bengio, 2012) that allows exploring different hyperperameter values. The hyperparameters explored for the architecture are the number of convolutional blocks, of filters in the first layer and of convolutional layers in a block, the number of fully-connected layers and the dropout rate. Other hyperparameters such as the learning rate and the weight decay are also part of the search. 100 different random architectures are trained on the 5-fold cross-validation done on *train-full*. For each input, we choose the architecture that obtained the best mean balanced accuracy across the validation sets of the cross-validation. We denote these architectures as **optimized**.

Network Training

The weights of the convolutional and fully-connected layers are initialized as described in (He et al., 2015), which corresponds to the default initialization method in PyTorch. Networks are trained for 100 epochs for **baseline** and 50 epochs for **optimized**. The training and validation losses are

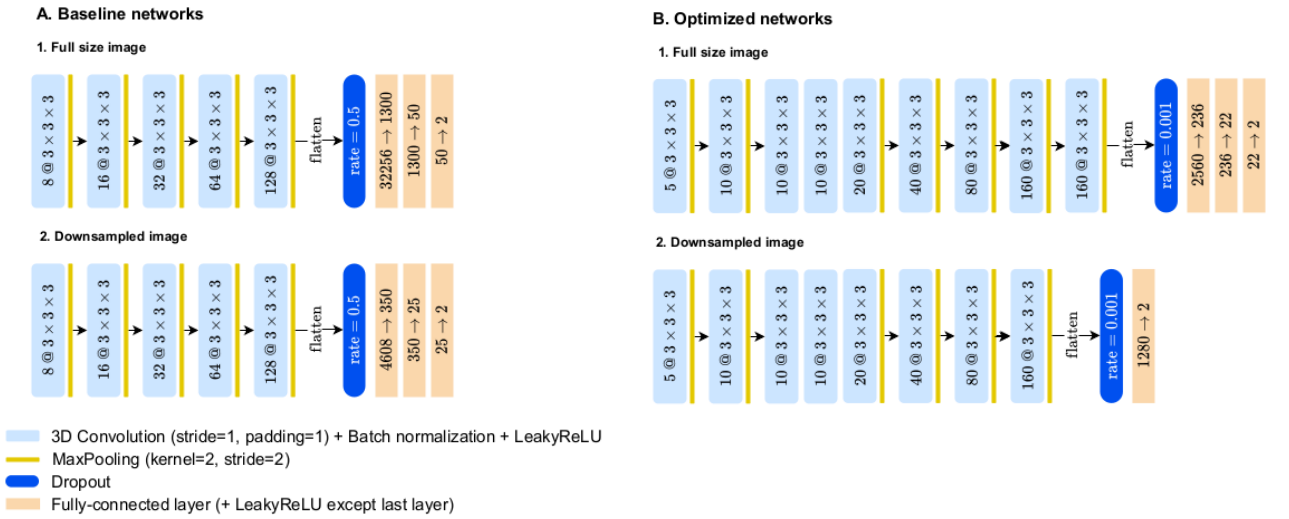


Figure 1.7: Diagrams of the network architectures used for classification. The first **baseline** architecture (A1) is the one used in (Wen et al., 2020), the second one (A2) is a very similar one adapted to process smaller inputs. The **optimized** architectures (B1) and (B2) are obtained independently with two different random searches. For convolution layers we specify the number of channels @ the kernel size and for the fully-connected layers we specify the number of input nodes → the number of output nodes. Each fully-connected layer is followed by a LeakyReLU activation except for the last one. For the dropout layer, the dropout rate is specified.

computed with the cross-entropy loss. For each experiment, the final model is the one that obtained the highest validation balanced accuracy during training. The balanced accuracy of the model is evaluated at the end of each epoch.

1.5.6 Experimental Protocol

As done in the previous sections, we perform three types of experiments and train the model on 1) only the *real* images, 2) only on synthetic data and 3) on synthetic and real images. Due to the current implementation, augmentation on high-resolution images is not possible due to computational time and so these images are only used to assess the baseline performance of the CNN with the maximum information available. Each series of experiments is done once for each training set (*train-50* and *train-full*). The CNN and the VAE share the same training set, and the VAE does not use the validation set during its training. For each training set, two VAEs are trained, one on the AD label only and the other on the CN label only. Examples of real and generated AD images are shown in Fig. 1.8. For each experiment 20 runs of the CNN training are launched. The use of a smaller training set *train-50* allows mimicking the behavior of the framework on smaller data sets, which are frequent in the medical domain.

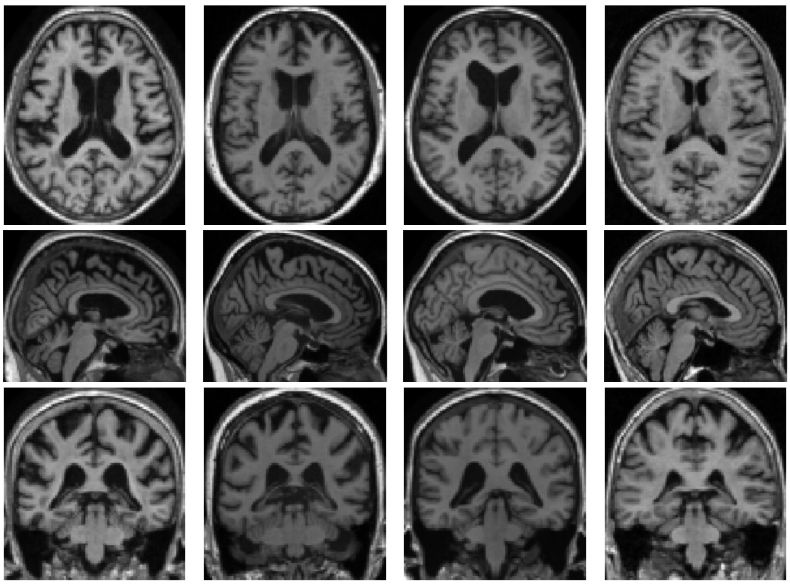


Figure 1.8: Example of two *true* patients compared to two generated by our method. Can you find the intruders ? Answers in Appendix 1.8.7.

1.5.7 Results

Results presented in Tables 1.6 and 1.7 (resp. Tables 1.8 and 1.9) are obtained with **baseline** (resp. **optimized**) hyperparameters and using either the *train-full* or *train-50* data set and augmented data (real + synthetic) or only synthetic data. In each table, the first two rows display the baseline performance obtained on real images only. Experiments are done on down-sampled images unless *high-resolution* is specified.

As observed on the *toy* examples, the proposed model is again able to produce meaningful synthetic samples since each CNN outperforms greatly the *baseline* (*i.e.* the real training data) either on *train-50* or *train-full*. The fact that classification performances on AIBL (which is never used for training) are better for a classifier trained on synthetic data than on the *baseline* shows again that the generative model does not overfit the training data (coming from ADNI) but rather produces samples that are also relevant for another database. Moreover, we again see that the classifier is able to outperform the *baseline* with only synthetic samples proof of good generalization power. Nonetheless, it can be noted that the performance increase thanks to DA is higher when using the **baseline** hyperparameters than the **optimized** ones. A possible explanation could be that the **optimized** network is already close to the maximum performance that can be reached with this setup and cannot be much improved with DA. Moreover, the VAE has not been subject to a similar search, which places it at a disadvantage. For both hyperparameters, the performance gain is higher on *train-50* than on *train-full*, which supports the results obtained in the previous section (see Fig. 1.6). The baseline balanced accuracy with the **baseline** hyperparameters on *train-full*, 80.6% on ADNI and 80.4% on AIBL, are similar to the results of (Wen et al., 2020). With DA, we improve our balanced accuracy to 86.3% on ADNI and 85.1% on AIBL: this performance is similar to their result using autoencoder pretraining (which can be very long to compute) and longitudinal data (1830 CN and 1106 AD images) instead of baseline data (243 CN and 210 AD images) as we did. Finally, the

main results from these experiments can be summarized as follows:

- *train-50* and **baseline** model: balanced accuracy increases by 6.2 pts on ADNI and 8.9 pts on AIBL,
- *train-full* and **baseline** model: balanced accuracy increases by 5.7 pts on ADNI and 4.7 pts on AIBL,
- *train-50* and **optimized** model: balanced accuracy increases by 2.5 pts on ADNI and 6.3 pts on AIBL,
- *train-full* and **optimized** model balanced accuracy increases by 1.5 pts on ADNI and -0.1 pts on AIBL.

1.6 Discussion

Contrary to techniques that are specific to a field of application, our method produced relevant data for diverse data sets including 2D natural images (MNIST, EMNIST, Fashion and CIFAR) or 3D medical images (ADNI and AIBL). Moreover, we noted that the networks trained on ADNI gave similar balanced accuracies on the ADNI test subset and AIBL showing that our synthetic data learned on ADNI benefit in the same way AIBL, and that it did not overfit the characteristics of ADNI. In addition to the robustness across data sets, the relevance of synthetic data for diverse classifiers was assessed. For toy data, these classifiers were a MLP, a random forest, a k-NN algorithm and a SVM. On medical image data, two different CNNs were studied: a **baseline** one that has been only slightly optimized in a previous study and an **optimized** one found with a more extensive search (random search). All these classifiers performed best on augmented data than real data only. However, for medical image data, we noted that the data augmentation was more beneficial to the **baseline** network, than to the **optimized** one but both networks obtained a similar performance with data augmentation on the largest training set. This means that data augmentation could avoid spending time and/or resources optimizing a classifier. The ability of the model to generate relevant data and enrich the original training data was also supported by the fact that almost all classifiers could achieve a better classification performance when trained only on synthetic data than on the *real* train. The method scalability to larger data sets and more complex models was also discussed.

Our generation framework appears also very well suited to perform data augmentation in a HDLSS setting (the binary classification of AD and CN subjects using T1w MRI). In all cases, the classification performance was at least as good as the maximum performance obtained with real data and could even be much better. For instance, the method allowed the balanced accuracy of the **baseline** CNN to jump from 66.3% to 74.3% when trained with only 50 images per class and from 77.7% to 86.3% when trained with 243 CN and 210 AD while still improving greatly sensitivity and specificity metrics. We witnessed a greater performance improvement than the other studies using a CNN on T1w MRI to differentiate AD and CN subjects (Valliani and Soni, 2017; Backstrom et al., 2018; Cheng and Liu, 2017; Aderghal et al., 2017; 2018). Indeed, these studies used simple transforms (affine and pixel-wise) that may not bring enough variability to improve the CNN performance. Though many complex methods now exist to perform data augmentation, they are still not widely adopted in the field of medical imaging. We suspect that this is mainly due to the lack of reproducibility

Table 1.6: Mean test performance of the 20 runs trained on *train-50* with the baseline hyperparameters

		ADNI			AIBL		
image type	synthetic images	sensitivity	specificity	balanced accuracy	sensitivity	specificity	balanced accuracy
real	-	70.3 ± 12.2	62.4 ± 11.5	66.3 ± 2.4	60.7 ± 13.7	73.8 ± 7.2	67.2 ± 4.1
real (high-res.)	-	78.5 ± 9.4	57.4 ± 8.8	67.9 ± 2.3	57.2 ± 11.2	75.8 ± 7.0	66.5 ± 3.0
synthetic	500	72.4 ± 6.4	65.6 ± 8.1	69.0 ± 1.9	56.6 ± 9.9	80.0 ± 5.3	68.3 ± 3.0
synthetic	1000	75.0 ± 6.2	65.6 ± 7.4	70.3 ± 2.0	62.7 ± 9.7	78.8 ± 5.3	70.8 ± 3.5
synthetic	2000	71.4 ± 6.6	70.4 ± 6.6	70.9 ± 3.0	62.1 ± 8.8	80.5 ± 4.7	71.3 ± 3.6
synthetic	3000	70.6 ± 5.2	$\mathbf{73.8 \pm 4.2}$	72.2 ± 1.4	65.7 ± 6.9	80.5 ± 4.6	73.1 ± 1.8
synthetic	5000	$\mathbf{78.1 \pm 6.1}$	69.0 ± 6.9	73.5 ± 2.0	$\mathbf{74.5 \pm 7.8}$	77.3 ± 5.4	$\mathbf{76.5 \pm 2.9}$
synthetic	10000	75.2 ± 6.8	73.4 ± 4.8	$\mathbf{74.3 \pm 1.9}$	73.6 ± 10.8	$\mathbf{79.4 \pm 6.0}$	75.9 ± 2.5
synthetic + real	500	71.9 ± 5.3	67.0 ± 4.5	69.4 ± 1.6	55.9 ± 6.8	81.1 ± 3.1	68.5 ± 2.5
synthetic + real	1000	69.8 ± 6.6	71.2 ± 3.7	70.5 ± 2.1	59.1 ± 9.0	82.1 ± 3.7	70.6 ± 3.1
synthetic + real	2000	72.2 ± 4.4	70.3 ± 4.3	71.2 ± 1.6	66.6 ± 7.1	79.0 ± 4.1	72.8 ± 2.2
synthetic + real	3000	71.8 ± 4.9	73.4 ± 5.5	72.6 ± 1.6	66.1 ± 9.3	81.1 ± 5.0	73.6 ± 3.0
synthetic + real	5000	$\mathbf{74.7 \pm 5.3}$	$\mathbf{73.5 \pm 4.8}$	$\mathbf{74.1 \pm 2.2}$	$\mathbf{71.7 \pm 10.0}$	80.5 ± 4.4	$\mathbf{76.1 \pm 3.6}$
synthetic + real	10000	74.7 ± 7.0	73.4 ± 6.1	74.0 ± 2.7	69.1 ± 9.9	$\mathbf{80.7 \pm 5.1}$	74.9 ± 3.2

Table 1.7: Mean test performance of the 20 runs trained on *train-full* with the baseline hyperparameters

		ADNI			AIBL		
image type	synthetic images	sensitivity	specificity	balanced accuracy	sensitivity	specificity	balanced accuracy
real	-	79.1 ± 6.2	76.3 ± 4.2	77.7 ± 2.5	70.6 ± 6.7	86.3 ± 3.6	78.4 ± 2.4
real (high-res.)	-	84.5 ± 3.8	76.7 ± 4.0	80.6 ± 1.1	71.6 ± 6.4	89.2 ± 2.7	80.4 ± 2.6
synthetic	500	81.6 ± 6.8	79.5 ± 5.8	80.5 ± 2.4	74.7 ± 9.3	87.3 ± 4.8	81.0 ± 3.2
synthetic	1000	82.9 ± 4.5	82.0 ± 5.8	82.4 ± 1.9	77.2 ± 7.4	88.8 ± 5.2	83.0 ± 2.0
synthetic	2000	81.9 ± 4.5	87.7 ± 3.4	84.8 ± 2.0	74.7 ± 6.3	92.1 ± 1.9	83.4 ± 2.7
synthetic	3000	$\mathbf{84.9 \pm 3.5}$	87.4 ± 3.5	86.1 ± 1.5	77.4 ± 5.8	90.9 ± 3.0	84.2 ± 1.8
synthetic	5000	84.0 ± 3.5	88.4 ± 3.3	86.2 ± 1.7	76.8 ± 4.2	$\mathbf{92.2 \pm 1.8}$	$\mathbf{84.5 \pm 1.8}$
synthetic	10000	84.2 ± 5.4	$\mathbf{88.6 \pm 3.9}$	$\mathbf{86.4 \pm 1.8}$	$\mathbf{77.5 \pm 7.4}$	91.0 ± 3.2	84.2 ± 2.4
synthetic + real	500	82.5 ± 3.4	81.9 ± 5.4	82.2 ± 2.4	76.0 ± 6.3	89.7 ± 3.3	82.9 ± 2.5
synthetic + real	1000	84.6 ± 4.4	84.3 ± 5.1	84.4 ± 1.8	77.0 ± 7.0	90.4 ± 3.4	83.7 ± 2.3
synthetic + real	2000	$\mathbf{85.4 \pm 4.0}$	86.4 ± 5.9	85.9 ± 1.6	77.2 ± 6.9	90.4 ± 3.8	83.8 ± 2.2
synthetic + real	3000	84.7 ± 3.6	86.8 ± 4.5	85.8 ± 1.7	77.2 ± 4.8	$\mathbf{91.7 \pm 2.9}$	84.4 ± 1.8
synthetic + real	5000	84.6 ± 4.2	86.9 ± 3.6	85.7 ± 2.1	76.9 ± 5.2	91.4 ± 3.0	84.2 ± 2.2
synthetic + real	10000	84.2 ± 2.8	$\mathbf{88.5 \pm 2.9}$	$\mathbf{86.3 \pm 1.8}$	$\mathbf{79.1 \pm 4.7}$	91.0 ± 2.6	$\mathbf{85.1 \pm 1.9}$

Table 1.8: Mean test performance of the 20 runs trained on *train-50* with the optimized hyperparameters

		ADNI			AIBL		
image type	synthetic images	sensitivity	specificity	balanced accuracy	sensitivity	specificity	balanced accuracy
real	-	75.4 ± 5.0	75.5 ± 5.3	75.5 ± 2.7	68.6 ± 8.5	82.6 ± 4.2	75.6 ± 4.1
real (high-res.)	-	73.6 ± 6.2	70.6 ± 5.9	72.1 ± 3.1	57.8 ± 12.3	84.6 ± 4.2	71.2 ± 5.1
synthetic	500	75.8 ± 3.0	77.6 ± 5.3	76.7 ± 2.8	73.2 ± 9.0	$\mathbf{83.6 \pm 4.0}$	78.4 ± 4.0
synthetic	1000	76.7 ± 4.6	78.5 ± 4.9	$\mathbf{77.6 \pm 3.7}$	78.7 ± 7.5	83.2 ± 4.8	80.9 ± 4.3
synthetic	2000	73.9 ± 3.6	$\mathbf{79.8 \pm 4.0}$	76.8 ± 3.0	78.2 ± 6.9	82.4 ± 3.7	80.3 ± 3.5
synthetic	3000	74.4 ± 6.1	79.8 ± 4.9	77.1 ± 4.0	76.4 ± 10.1	82.4 ± 4.3	79.4 ± 4.7
synthetic	5000	77.1 ± 4.5	77.4 ± 5.2	77.2 ± 2.1	81.1 ± 5.9	82.0 ± 3.9	$\mathbf{81.5 \pm 2.6}$
synthetic	10000	$\mathbf{77.5 \pm 5.3}$	77.3 ± 4.7	77.4 ± 3.1	$\mathbf{81.7 \pm 5.4}$	79.7 ± 4.1	80.7 ± 2.9
synthetic + real	500	73.2 ± 4.2	78.0 ± 3.3	75.6 ± 2.5	69.2 ± 9.4	$\mathbf{82.7 \pm 4.1}$	76.0 ± 4.2
synthetic + real	1000	76.1 ± 5.3	$\mathbf{79.5 \pm 2.9}$	77.8 ± 2.3	79.3 ± 5.8	82.5 ± 4.2	80.9 ± 3.2
synthetic + real	2000	75.2 ± 3.8	78.6 ± 4.4	76.9 ± 2.4	77.8 ± 8.8	82.2 ± 4.5	80.0 ± 3.6
synthetic + real	3000	76.5 ± 3.8	79.2 ± 4.2	77.8 ± 1.9	80.9 ± 7.9	81.4 ± 4.2	81.2 ± 3.7
synthetic + real	5000	77.1 ± 3.7	76.7 ± 4.1	76.9 ± 2.5	80.7 ± 6.1	81.2 ± 3.7	80.9 ± 2.7
synthetic + real	10000	$\mathbf{77.8 \pm 4.6}$	78.2 ± 4.9	$\mathbf{78.0 \pm 2.1}$	$\mathbf{81.7 \pm 4.9}$	81.9 ± 4.6	$\mathbf{81.9 \pm 2.2}$

Table 1.9: Mean test performance of the 20 runs trained on *train-full* with the optimized hyperparameters

		ADNI			AIBL		
image type	synthetic images	sensitivity	specificity	balanced accuracy	sensitivity	specificity	balanced accuracy
real	-	82.5 ± 4.2	88.5 ± 6.6	85.5 ± 2.4	75.1 ± 8.4	88.7 ± 9.0	81.9 ± 3.2
real (high-res.)	-	82.6 ± 4.5	88.9 ± 6.3	85.7 ± 2.5	78.9 ± 5.4	89.9 ± 4.0	84.4 ± 1.7
synthetic	500	81.7 ± 3.6	90.5 ± 3.9	86.1 ± 1.4	75.5 ± 7.1	89.8 ± 4.3	82.6 ± 2.9
synthetic	1000	82.8 ± 3.4	90.0 ± 4.0	86.4 ± 2.1	76.8 ± 4.5	91.5 ± 2.5	84.2 ± 1.7
synthetic	2000	81.3 ± 2.8	91.2 ± 2.8	86.2 ± 1.7	76.2 ± 6.7	$\mathbf{92.2 \pm 3.6}$	84.2 ± 2.6
synthetic	3000	82.2 ± 4.9	90.6 ± 4.5	86.4 ± 2.0	77.7 ± 6.3	90.8 ± 4.4	84.3 ± 2.0
synthetic	5000	80.6 ± 3.4	$\mathbf{91.6 \pm 2.5}$	86.1 ± 1.9	75.3 ± 5.4	92.4 ± 2.5	83.8 ± 2.0
synthetic	10000	$\mathbf{84.0 \pm 3.8}$	89.1 ± 3.1	$\mathbf{86.5 \pm 1.7}$	$\mathbf{79.2 \pm 5.2}$	90.1 ± 3.7	$\mathbf{84.7 \pm 2.3}$
synthetic + real	500	82.3 ± 2.3	89.8 ± 2.7	86.0 ± 1.8	74.9 ± 5.0	91.4 ± 2.6	83.2 ± 2.4
synthetic + real	1000	82.5 ± 3.3	90.5 ± 4.1	86.5 ± 1.9	76.4 ± 5.6	91.0 ± 3.4	83.7 ± 2.0
synthetic + real	2000	$\mathbf{83.1 \pm 4.2}$	$\mathbf{91.3 \pm 3.2}$	$\mathbf{87.2 \pm 1.7}$	76.0 ± 4.7	92.0 ± 2.4	84.0 ± 2.0
synthetic + real	3000	81.3 ± 3.7	90.4 ± 3.4	85.8 ± 2.6	74.9 ± 7.3	92.3 ± 2.6	83.6 ± 3.2
synthetic + real	5000	81.9 ± 3.5	90.9 ± 2.5	86.4 ± 1.3	74.1 ± 4.9	$\mathbf{92.9 \pm 1.9}$	83.5 ± 2.2
synthetic + real	10000	82.2 ± 3.4	91.2 ± 3.6	86.7 ± 1.8	$\mathbf{76.4 \pm 4.2}$	92.1 ± 2.1	$\mathbf{84.3 \pm 1.8}$

of such frameworks. Hence we provide the source code, as well as scripts to easily reproduce the experiments of this chapter from the ADNI and AIBL data set download to the final evaluation of the CNN performance. We also developed a software [7] implementing the method and making it easily accessible to the community.

However, our classification performance on synthetic data could be improved in many ways. First, we chose in this study not to spend much time optimizing the VAE's hyperparameters and so in Sec. 1.5 we chose to work with down-sampled images to deal with memory issues. We could look for another architecture to train the VAE directly on high-resolution images leading potentially to a better performance as witnessed in experiments on real images only. Moreover, we could couple the advantages of other techniques such as autoencoder pretraining or weak supervision to our data augmentation framework. However, the advantages may not stack as observed when using DA on optimized hyperparameters. Finally, we chose to train our networks with only one image per participant, but our framework could also benefit from the use of the whole follow-up of all patients to further improve performance. However, a long follow-up is rather an exception in the context of medical imaging. This is why we assessed the relevance of our DA framework in the context of small data sets which is a main issue in this field. Nonetheless, a training set of 50 images per class can still be seen as large in the case of rare diseases and so it may be interesting to evaluate the reliability of our method on even smaller training sets.

1.7 Conclusion

In this chapter, we proposed a new VAE-based data augmentation framework whose performance and robustness were validated on classification tasks on *toy* and *real-life* data sets. This method relies on a model combining a proper latent space modeling of the VAE seen as a Riemannian manifold and a new generation procedure exploiting such geometrical aspects. In particular, the generation method does not use the prior as is standard since we showed that, depending on its choice and the data set considered, it may lead to a very poor latent space prospecting and a degraded sampling while the proposed method does not suffer from such drawbacks. The proposed amendments were motivated, discussed and compared to other VAE models and demonstrated promising results. The model indeed appeared to be able to generate new data faithfully and demonstrated a strong generalization power which makes it very well suited to perform data augmentation even in the challenging context of HDLSS data. For each augmentation experiment, it was able to enrich the initial data set so that a classifier performs better on augmented data than only on the *real* ones. Future work would consist in building a framework able to handle longitudinal data and so able to generate not only one image but a whole patient trajectory.

Acknowledgment

The research leading to these results has received funding from the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and reference ANR-10-IAIHU-06 (Agence Na-

[7] https://github.com/clementchadebec/pyraug

tionale de la Recherche-10-IA Institut Hospitalo-Universitaire-6). This work was granted access to the HPC resources of IDRIS under the allocation 101637 made by GENCI (Grand Équipement National de Calcul Intensif).

Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical Imaging and Bioengineering, and through generous contributions from the following: AbbVie, Alzheimer's Association; Alzheimer's Drug Discovery Foundation; Araclon Biotech; BioClinica, Inc.; Biogen; Bristol-Myers Squibb Company; CereSpir, Inc.; Cogstate; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; EuroImmun; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; Fujirebio; GE Healthcare; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC.; Johnson & Johnson Pharmaceutical Research & Development LLC.; Lumosity; Lundbeck; Merck & Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Neurotrack Technologies; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Takeda Pharmaceutical Company; and Transition Therapeutics. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimer's Therapeutic Research Institute at the University of Southern California. ADNI data are disseminated by the Laboratory for Neuro Imaging at the University of Southern California.

1.8 Appendices

1.8.1 Riemannian Geometry

In the framework of differential geometry, one may define a Riemannian manifold $\mathcal{M}$ as a smooth manifold endowed with a Riemannian metric g that is a smooth inner product $g : p \to \langle \cdot | \cdot \rangle_p$ on the tangent space $T_p\mathcal{M}$ defined at each point of the manifold $p \in \mathcal{M}$. We call a chart (or coordinate chart) (U, φ) a homeomorphism mapping an open set U of the manifold to an open set V of an Euclidean space. The manifold is called a $d-$dimension manifold if for each chart of an atlas we further have $V \subset \mathbb{R}^d$. That is there exists a neighborhood U of each point p of the manifold such that U is homeomorphic to $\mathbb{R}^d$. Given $p \in U$, the chart $\varphi : (x^1, \ldots, x^d)$ induces a basis $\left(\frac{\partial}{\partial x^1}, \ldots, \frac{\partial}{\partial x^d}\right)_p$ on the tangent space $T_p\mathcal{M}$. Hence, a local representation of the metric of a Riemannian manifold in the chart (U, φ) can be written as a positive definite matrix $\mathbf{G}(p) = (g_{i,j})_{p, 0 \leq i,j \leq d} = (\langle \frac{\partial}{\partial x^i} | \frac{\partial}{\partial x^j} \rangle_p)_{0 \leq i,j \leq d}$ at each point $p \in U$. That is for $v, w \in T_p\mathcal{M}$ and $p \in U$, we have $\langle u | w \rangle_p = u^\top \mathbf{G}(p) w$. Since we propose to work in the ambient-like manifold $(\mathbb{R}^d, g)$, there exists a global chart given by $\varphi = id$. Hence, for the following, we assume that we work in this coordinate system and so $\mathbf{G}$ will refer to the metric's matrix representation in this chart.

There are two ways to apprehend manifolds. The extrinsic view assumes that the manifold is embedded within a higher dimensional Euclidean space (think of the 2-dimensional sphere $\mathcal{S}^2$ embedded within $\mathbb{R}^3$). The intrinsic view, which is adopted in this chapter, does not make such an assumption since the manifold is studied using its underlying structure. For example, a curve's length cannot be interpreted using the distance defined on an Euclidean space but requires the use of the metric defined onto the manifold itself. The length of a curve γ between two points of the manifold $z_1, z_2 \in \mathcal{M}$ and parametrized by $t \in [0, 1]$ such that $\gamma(0) = z_1$ and $\gamma(1) = z_2$ is then given by

$$\mathcal{L}(\gamma) = \int_0^1 \|\dot{\gamma}(t)\|_{\gamma(t)} \mathrm{d}t = \int_0^1 \sqrt{\langle \dot{\gamma}(t) | \dot{\gamma}(t) \rangle_{\gamma(t)}} \mathrm{d}t \,.$$

Curves minimizing such a length are called *geodesics* and a distance dist between elements of a (connected) manifold can be introduced as follows:

$$\mathrm{dist}(z_1, z_2) = \inf_\gamma \mathcal{L}(\gamma) \quad \text{s.t.} \quad \gamma(0) = z_1, \;\; \gamma(1) = z_2$$

The manifold $\mathcal{M}$ is said to be *geodesically complete* if all geodesic curves can be extended to $\mathbb{R}$. In other words, at each point p of the manifold one may draw a *straight* line (with respect to the formerly defined distance) indefinitely and in any direction.

1.8.2 On the Generation Process

We recall that to sample from the defined target distribution given by the inverse of the volume element of the Riemannian manifold we recourse to the Hamiltonian Monte Carlo (HMC) sampler since the normalizing constant is hard to compute. Hence, we recall in this section some elements on the HMC sampler and how it applies in our specific framework.

Likewise the RHMC presented in the previous section, given a target density p_{target} we want to sample from, the idea behind the HMC sampler is to introduce a random variable $v \sim \mathcal{N}(0, I_d)$ independent from z and rely on Hamiltonian dynamics. Analogous to physical systems, z can again be seen as the *position* and v as the *velocity* of a particle whose potential energy $U(z)$ and kinetic energy $K(v)$ are given by

$$U(z) = -\log p_{\text{target}}(z), \qquad K(v) = \frac{1}{2} v^\top v \,.$$

These two energies give together the Hamiltonian (Duane et al., 1987; Leimkuhler and Reich, 2004)

$$H(z, v) = U(z) + K(v) \,.$$

The evolution in time of such a particle is governed by Hamilton's equations as follows

$$\frac{\partial z_i}{\partial t} = \frac{\partial H}{\partial v_i}, \qquad \frac{\partial v_i}{\partial t} = -\frac{\partial H}{\partial z_i} \,.$$

Such equations can be integrated using a discretization scheme known as the *Stormer-Verlet* or *leapfrog* integrator which is run l times

$$\begin{aligned} v(t + \gamma/2) &= v(t) - \frac{\gamma}{2} \cdot \nabla_z U(z(t)) \,, \\ z(t + \gamma) &= z(t) + \gamma \cdot v(t + \gamma/2) \,, \\ v(t + \gamma) &= v(t + \gamma/2) - \frac{\gamma}{2} \nabla_z U(z(t + \gamma)) \,, \end{aligned} \tag{1.9}$$

where γ is the integrator step size. The HMC sampler produces a Markov chain (z^n) with the aforementioned integrator. More precisely, given z_0^n, the current state of the chain, an initial *velocity* is sampled $v_0 \sim \mathcal{N}(0, I_d)$ and then Eq. (1.9) are run l times to move from (z_0^n, v_0) to (z_l^n, v_l). The proposal z_l^n is then accepted with probability $\alpha = \min\left(1, \frac{\exp(-H(z_l^n, v_l))}{\exp(-H(z_0^n, v_0))}\right)$. It was shown that the chain (z^n) is time-reversible and converges to its stationary distribution p_{target} (Duane et al., 1987; Liu, 2008; Neal and others, 2011).

In our method p_{target} is given by Eq. (1.7) and

$$p(z) = \frac{\mathbf{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}}{\int_{\mathbb{R}^d} \mathbf{1}_S(z)\sqrt{\det \mathbf{G}^{-1}(z)dz}} \,,$$

where S is a compact set[8] so that the integral is well defined. Fortunately, since the HMC sampler allows sampling from densities known up to a normalizing constant (thanks to the acceptance ratio), the computation of the denominator of p_{target} is not needed and the Hamiltonian follows

$$H(z, v) = U(z) + K(v) \propto -\frac{1}{2} \log\det \mathbf{G}^{-1}(z) + \frac{1}{2} v^\top v$$

and is easy to compute. Hence, the only *difficulty* left is the computation of the gradient $\nabla_z U(z)$ needed in the *leapfrog* integrator which is actually pretty straightforward using the chain rule. In this chapter, a typical choice for γ and l, the sampler's parameters, is $\gamma \in [0.01, 0.05]$ and $l \in [10, 15]$. We would also like to mention the recent work of (Arvanitidis et al., 2020-08-02) where the authors used the distribution $q(z) \propto (1 + \sqrt{\det \mathbf{G}(z)})^{-1}$ to sample from a Wasserstein GAN (Arjovsky et al., 2017-12-06). Nonetheless, both the framework and the metric remain quite different.

[8] Take for instance $\{z \in \mathcal{Z}, \|z\| \leq 2 \cdot \max_i \|c_i\|\}$

1.8.3 Detailed Experimental Setting

Parameters of Sec. 1.3.3

For this experiment and for a fair comparison, each model is trained with the same neural network architecture for the encoder and decoder presented in Table 1.10 along with the same latent space dimension set to 2. The main parameters for the *geometry-aware* VAE are presented in Table 1.11. Each model is trained until the ELBO does not improve for 20 epochs with an Adam optimizer (Kingma and Ba, 2014) and a learning rate of 10^{-3}. Since the data sets sizes are small, the training is performed in a single batch.

Table 1.10: Neural Net Architectures for MNIST, EMNIST and fashion. The same architectures are used for the VAEs, VAMP, RAE and *geometry-aware* VAEs.

μ_ϕ Σ_ϕ	(D, 400, ReLU)	(400, d, LINEAR) (400, d, LINEAR)
π_θ	(d,400, ReLU)	(400, D, SIGMOID)
$L_\psi^{\text{DIAG.}}$ $L_\psi^{\text{LOW.}}$	(D, 400, ReLU)	(400, d, LINEAR) (400, $\frac{d(d-1)}{2}$, LINEAR)

D: INPUT SPACE DIMENSION, d: LATENT SPACE DIMENSION

Table 1.11: *Geometry-aware* VAE parameters.

DATA SETS	PARAMETERS					
	d^*	K	ε	T	λ	$\sqrt{\beta_0}$
SYNTHETIC SHAPES	2	3	10^{-2}	0.8	10^{-3}	0.3
REDUCED MNIST (BAL.)	2	3	10^{-2}	0.8	10^{-3}	0.3
REDUCED MNIST (UNBAL.)	2	3	10^{-2}	0.8	10^{-3}	0.3
REDUCED EMNIST	2	3	10^{-2}	0.8	10^{-3}	0.3

* LATENT SPACE DIMENSION (SAME FOR THE OTHER MODELS)

Parameters of Sec. 1.4

For this experiment, we consider a vanilla VAE, a VAE with VAMP prior, a *geometry-aware* VAE using the prior to generate, a *geometry-aware* VAE using the proposed method, a regularized autoencoder with a penalty on the gradient of the decoder as proposed in (Ghosh et al., 2020) and consider two other approaches proposed in the literature to improve the generation from a VAE. The first one is a two stage VAE as proposed in (Dai and Wipf, 2018) and the second one consists in fitting a mixture of Gaussian in the latent space of the VAE post-training (Ghosh et al., 2020).

MNIST, EMNIST and Fashion For these data sets, we use the same parameters and neural network architectures as presented in the former section and Table 1.10 except for *reduced* Fashion where the dimension of the latent space is set to 5. As to training parameters for the VAEs, for each model we use an Adam optimizer with a learning rate set to 10^{-3}. Since the data sets sizes are small the training is performed in a single batch. An implementation of all the models can be found at https://github.com/clementchadebec/benchmark_VAE.

CIFAR For CIFAR, each model is trained for 500 epochs and we keep the model achieving the best ELBO. The latent space dimension is set to 5 for all models. The training is performed with an Adam optimizer (Kingma and Ba, 2014) and a learning rate of 10^{-4}. Since the data sets sizes are small the training is performed in a single batch. All the models share again the same neural network architectures for both the encoder and decoder which is described in Table 1.12.

Table 1.12: Neural Net Architectures for CIFAR. The same architectures are used for the VAEs, VAMP, RAE and *geometry-aware* VAEs.

	CIFAR10
Encoder	$(3, 32, 32)$
Layer 1	Conv$(128, (4, 4)$, stride=2) Batch normalization ReLU
Layer 2	Conv$(256, (4, 4)$, stride=2) Batch normalization ReLU
Layer 3	Conv$(512, (4, 4)$, stride=2) Batch normalization ReLU
Layer 4	Conv$(1024, (4, 4)$, stride=2) Batch normalization ReLU
Layer 5	Linear$(4096, 10)$
Decoder	(10)
Layer 1	Linear(65536) Reshape$(1024, 8, 8)$
Layer 2	ConvT$(512, (4, 4)$, stride=2) Batch normalization ReLU
Layer 3	ConvT$(256, (4, 4)$, stride=2) Batch normalization ReLU
Layer 4	ConvT$(3, (4, 4)$, stride=1) Batch normalization Sigmoid

Classifiers Settings As to the classifiers, for Sec. 1.4.2, we use a DenseNet (Huang et al., 2017) as benchmark for data augmentation. The implementation we use is the one proposed in (Amos, 2020) with a *growth rate* equals to 10, *depth* of 20 and 0.5 *reduction* and the model is trained with a learning rate of 10^{-3}, weight decay of 10^{-4} and a batch size of 200. The classifier is trained until the loss does not improve on the validation set for 50 epochs and tested on the original test sets (*e.g.* ≈ 1000 samples per class for MNIST). For Sec. 1.4.2, the MLP has 400 hidden units with ReLU activation function. It is trained with Adam optimizer and a learning rate of 10^{-3}. Training is stopped if the loss does not improve on the validation set for 20 epochs. In Sec. 1.4.2, we consider a DenseNet again and increase (resp. decrease) its depth to increase (resp. decrease) the number of parameters of the classifier. Any other parameter is set to the value mentioned earlier.

Parameters of Sec. 1.5

To generate new data on the ADNI database we amend the neural network architectures and use the one described in Table 1.13. The parameters used in the *geometry-aware* VAE are provided in Table 1.14. An Adam optimizer with a learning rate of 10^{-5} and batch size of 25 are used. The VAE model is trained until the ELBO does not improve for 50 epochs. Generating 50 ADNI images takes approx. 30 s.[9] with the proposed method on Intel Core i7 CPU (6x1.1GHz) and 16 GB RAM.

Table 1.13: Neural Net Architecture

μ_ϕ	(D, H1, RELU)	(H1, H2, RELU)	(H2, H3, RELU)	(H3, d, LIN)
Σ_ϕ		(H1, H2, RELU)	(H2, H3, RELU)	(H3, d, LIN)
π_θ	(d, H3, RELU)	(H3, H2, RELU)	(H2, H1, RELU)	(H1, D, SIG)
$L_\psi^{\text{DIAG.}}$	(D, H3, RELU)	(H3, d, LIN)	-	-
$L_\psi^{\text{LOW.}}$		(H3, $\frac{d(d-1)}{2}$, LIN)	-	-

D	H1	H2	H3	D
777504	500	500	400	10

Table 1.14: *Geometry-aware* parameters settings for ADNI database

DATA SET	PARAMETERS					
	d	K	ε	T	λ	$\sqrt{\beta_0}$
ADNI	10	3	10^{-3}	1.5	10^{-2}	0.3

1.8.4 A Few More Sampling Comparisons (Sec. 1.3.3)

In addition to the comparison performed in Sec. 1.3.3, we also compare qualitatively a Vanilla VAE, a VAE with VAMP prior and a *geometry-aware* VAE on four reduced data sets and in higher dimensional latent spaces of dimension 10. The first one is created with 180 binary rings and disks with different diameters and thicknesses ensuring balanced classes. The second one is composed of 120 samples of EMNIST (letter *M*) and referred to as *reduced* EMNIST. Another one is created with 120 samples from the classes *0*, *1* and *2* of MNIST database ensuring balanced classes and is called *reduced* MNIST. The last one, *reduced* Fashion, is again composed of 120 samples from three classes (*shoes*, *trouser* and *bag*) from FashionMNIST and ensuring balanced classes. The models have the same architectures as described in Table 1.10 and are trained with the parameters stated in Table 1.15. Each model is trained until the ELBO does not improve for 20 epochs with Adam optimizer, a learning rate of 10^{-3} and in a single batch. In Fig. 1.10 are presented from top to bottom: 1) an extract of the training samples for each data set; 2) samples obtained with a vanilla VAE with a Gaussian prior; 2) data generated from a VAE with VAMP prior; 3) samples created by a *geometry-aware* VAE and using the prior or 4) samples from our method. As discussed in the paper, the proposed method is again able to visually outperform peers since for all data sets it is able to create sharper and more meaningful samples even if the number of training samples is quite small.

[9]Depends on the length of the MCMC chain and HMC hyper-parameter, l. We used 300 steps with $l = 15$.

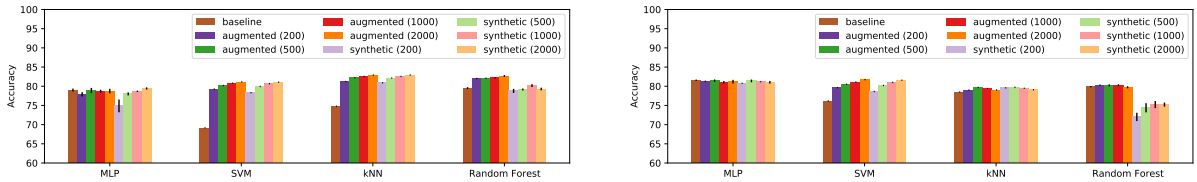
Table 1.15: *Geometry-aware* VAE parameters.

Data sets	Parameters					
	d^*	K	ε	T	λ	$\sqrt{\beta_0}$
Synthetic shapes	10	3	10^{-2}	1.5	10^{-3}	0.3
reduced MNIST	10	3	10^{-2}	1.5	10^{-3}	0.3
reduced EMNIST	10	3	10^{-2}	1.5	10^{-3}	0.3
reduced Fashion	10	3	10^{-2}	1.5	10^{-3}	0.3

* Latent space dimension (same for VAE and VAMP-VAE)

1.8.5 Additional Results (Sec. 1.4.2)

Further to the experiments presented in Sec. 1.4.2, we also provide the results of the four classifiers on *reduced* EMNIST and *reduced* Fashion in Fig. 1.9. Again, for most classifiers the proposed method either equals or greatly outperform the *baseline*.



(a) *reduced* EMNIST (b) *reduced* FashionMNIST

Figure 1.9: Evolution of the accuracy of four benchmark classifiers on the *reduced* EMNIST data set (top) and the *reduced* Fashion data set (bottom). Stochastic classifiers are trained with five independent runs and we report the mean accuracy and standard deviation on the test set.

1.8.6 A few More Sample Generation on ADNI

In this section, we first provide several slices of a 3D image generated by our model. The model is trained on the class AD of *train-50* (*i.e.* on 50 MRI of patient having been diagnosed with Alzheimer's disease). The generated image is presented in Fig. 1.11. We also present in Fig. 1.12, four generated patients for a model trained on *train-50*. The two left images show *cognitively normal* generated patients while the rightmost images represent AD generated patients.

1.8.7 The Intruders: Answers to Fig. 1.8

In Fig. 1.8 of the chapter, the synthetic samples are the leftmost and rightmost images while the *real* patients are in the middle. The model is trained on the class AD of *train-full i.e.* 210 images.

é

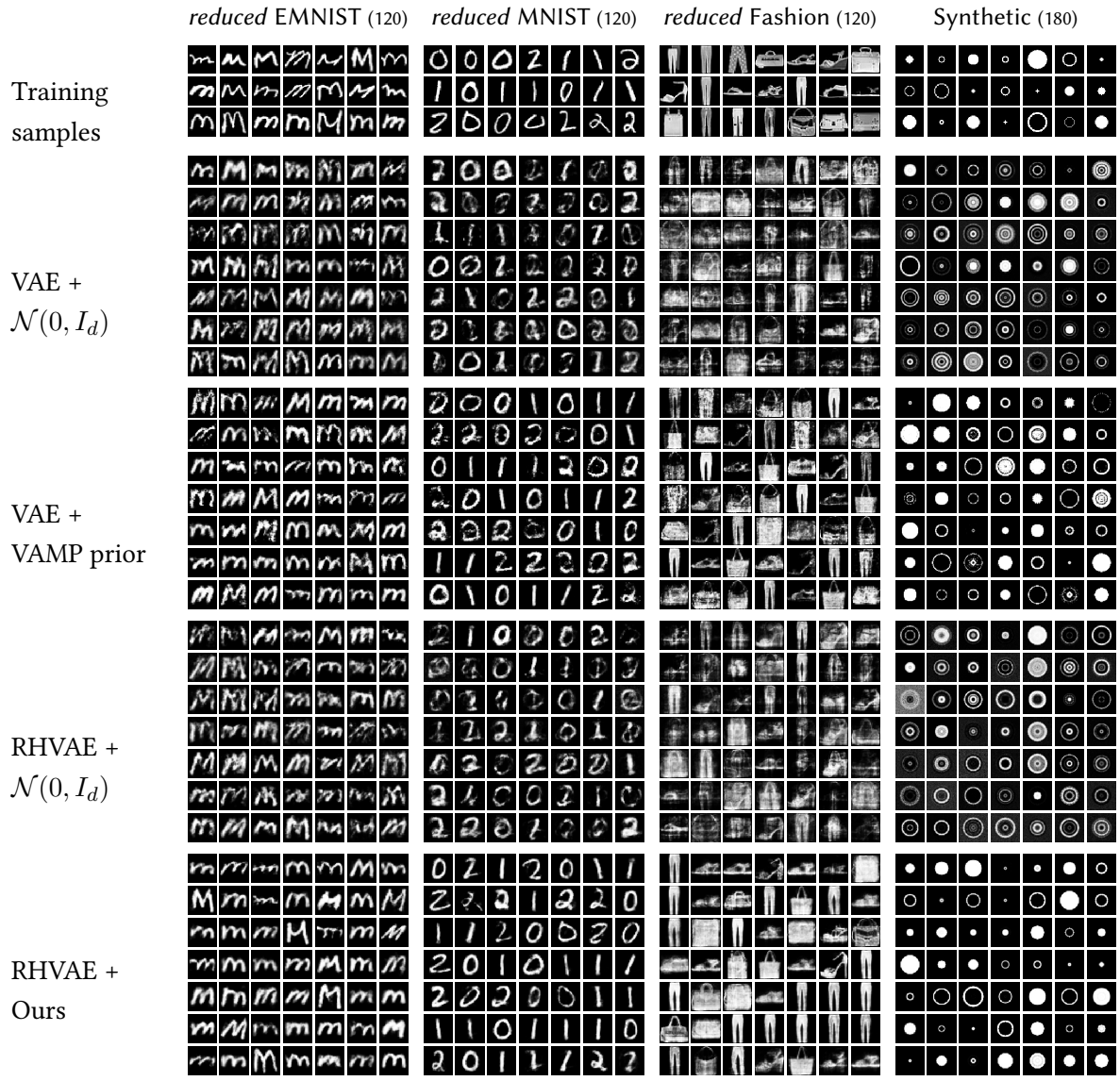


Figure 1.10: Comparison of four sampling methods on *reduced* EMNIST (120 letters *M*), *reduced* MNIST, *reduced* FashionMNIST and the synthetic data sets in higher dimensional latent spaces (dimension 10). From top to bottom: 1) samples extracted from the training set; 2) samples generated with a Vanilla VAE and using the prior ($\mathcal{N}(0, I_d)$); 3) from the VAMP prior VAE ; 4) from a RHVAE and the *prior-based* generation scheme and 5) from a RHVAE and using the proposed method. All the models are trained with the same encoder and decoder networks and identical latent space dimension. An early stopping strategy is adopted and consists in stopping training if the ELBO does not improve for 20 epochs. The number of training samples is noted between parenthesis.

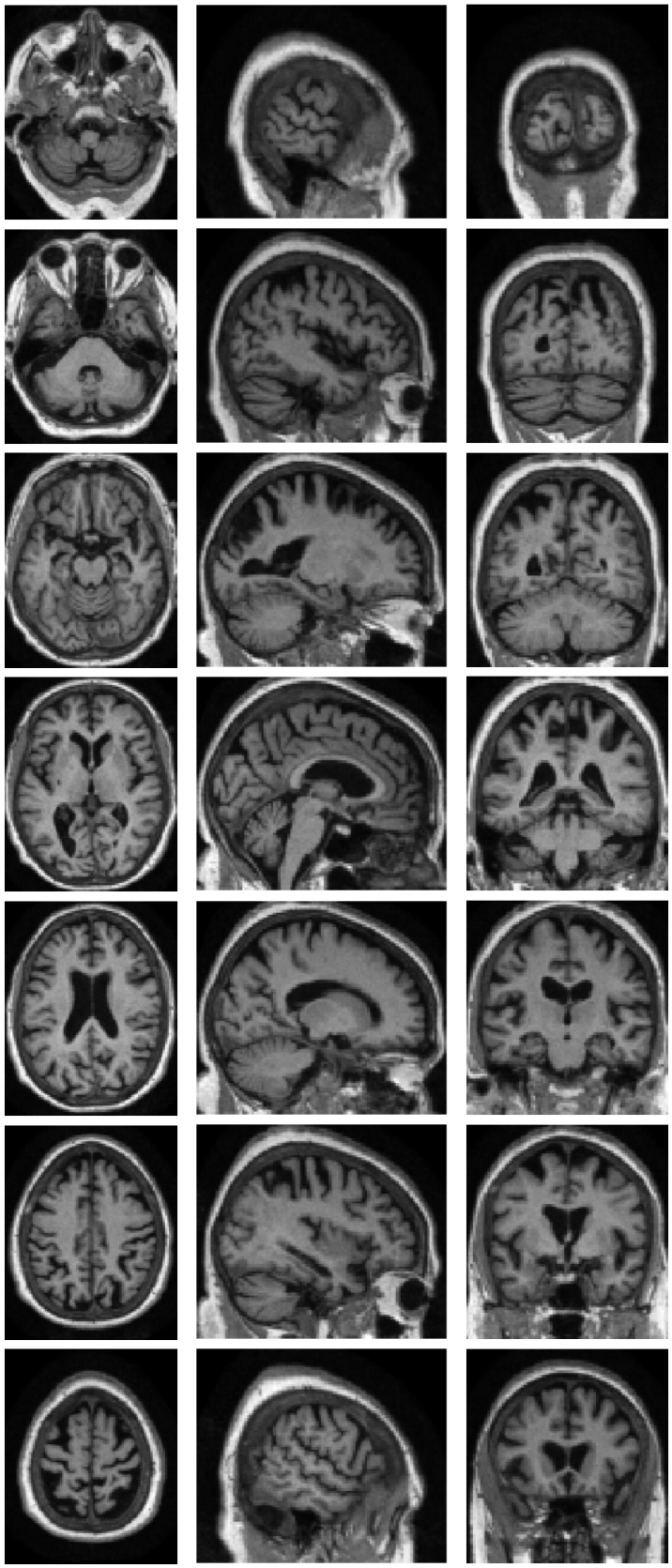


Figure 1.11: Several slices of a generated image. The model is trained on the AD class of *train-50* (*i.e.* 50 images of AD patients).

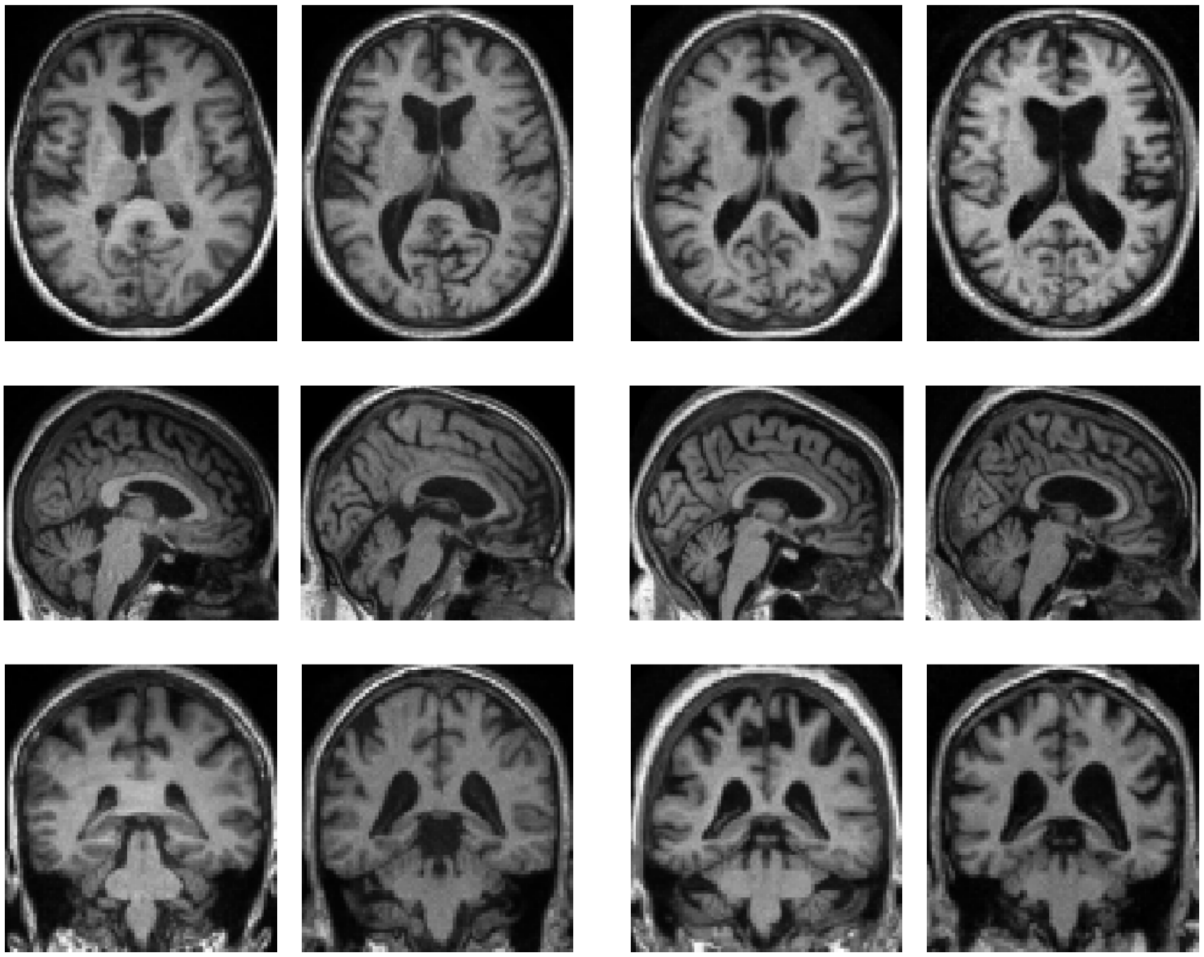


Figure 1.12: Images generated by our method when trained on *train-50*. *Left*: CN generated patients. *Right*: AD generated patients.

Chapter 2

Sampling from Riemannian Manifolds - Application to the RHVAE

In this chapter, we propose a new way to sample from Riemannian manifolds. The method consists in sampling from a Riemannian random walk that follows geodesic paths which is a natural way of exploring Riemannian manifolds. We apply this method to the geometry-based Variational Autoencoder proposed in the previous chapter and evaluate its usefulness for data augmentation in the low data regime. The proposed method is validated across various standard and real-life *data sets. In particular, this scheme allows to greatly improve classification results on the OASIS database where balanced accuracy jumps from 80.7% for a classifier trained with the raw data to 88.6% when trained only with the synthetic data generated by our method. Such results were also observed on 3 standard data sets and with other classifiers.*[1].

This chapter led to a publication in a MICCAI 2021 workshop (DALI). See (Chadebec and Allassonnière, 2021).

[1] A code is available at https://github.com/clementchadebec/Data_Augmentation_with_VAE-DALI

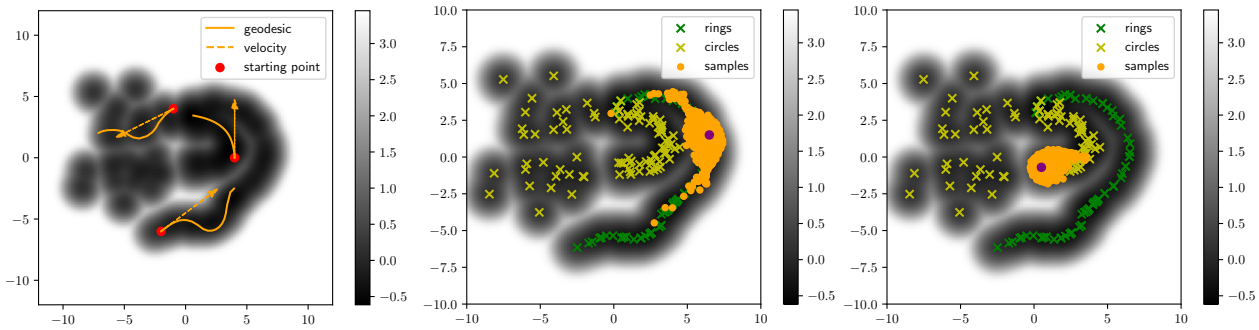


Figure 2.1: *Left*: Geodesic *shooting* in a latent space learned by a RHVAE with different starting points (red dots) and initial velocities (orange arrows). *Middle and right*: Samples from the wrapped normal $\mathcal{N}^W(p, I_d)$. The log metric volume element $\log \sqrt{\det \mathbf{G}(z)}$ is presented in gray scale.

2.1 The Wrapped Normal Distribution

We assume that $\mathcal{M}$ is a *geodesically-complete* Riemannian manifold endowed with a known Riemannian metric $\mathbf{G}$. We recall that for any $p \in \mathcal{M}$, the exponential map at p, Exp_p, maps a vector v of the tangent space $T_p\mathcal{M}$ to a point of the manifold $\widetilde{p} \in \mathcal{M}$ such that the geodesic starting at p with initial velocity v reaches $\widetilde{p}$ at time 1. In particular, since the manifold is *geodesically complete*, then Exp_p is defined on the entire tangent space $T_p\mathcal{M}$. The notion of normal distribution may be extended to Riemannian manifolds in several ways. One of them is the *wrapped* normal distribution. The main idea is to define a classic normal distribution $\mathcal{N}(0, \Sigma)$ on the tangent space $T_p\mathcal{M}$ for any $p \in \mathcal{M}$ and pushing it forward to the manifold using the exponential map. This defines a probability distribution on the manifold $\mathcal{N}^W(p, \Sigma)$ called the *wrapped* normal distribution. Sampling from this distribution is straightforward and consists in drawing a velocity in the tangent space from $\mathcal{N}(0, \Sigma)$ and mapping it onto the manifold using the exponential map (Mallasto and Feragen, 2018). Hence, the *wrapped* normal allows for a manifold prospecting along geodesic paths. Nonetheless, this requires to compute Exp_p which can be performed with a numerical scheme (see. Sec. 2.2). On the left of Fig. 2.1 are displayed some geodesic paths in the latent space and with respect to the metric learned by a RHVAE presented in Chapter 1 with different starting points (red dots) and initial velocities (orange arrows). Samples from $\mathcal{N}^W(p, I_d)$ are also presented in the middle and the right along with the encoded input data. As expected this distribution takes into account the local geometry of the manifold thanks to the geodesic shooting steps. This is a very interesting property since it encourages the samples to remain close to the data as geodesics tend to travel through locations with the lowest volume element $\sqrt{\det \mathbf{G}(z)}$ and so avoid areas with very poor information.

2.2 Computing the Exponential Map

Sampling from the *wrapped* normal distribution nonetheless requires to compute the exponential map at any given point $p \in \mathcal{M}$ and for any tangent vector $v \in T_p\mathcal{M}$. To do so, we can rely on the

Hamiltonian definition of geodesic curves. First, for any given $v \in T_p\mathcal{M}$, the linear form:

$$q_v : \begin{cases} T_p\mathcal{M} & \to \mathbb{R} \\ u & \to g_p(u, v) \end{cases},$$

is called a moment and is a representation of v in the dual space. In short, we may write $q_v(u) = u^\top \mathbf{G} v$. Then, the definition of the Hamiltonian follows

$$H(p, q) = \frac{1}{2} g_p^*(q, q)\,,$$

where g_p^* is the dual metric whose local representation is given by $\mathbf{G}^{-1}(p)$, the inverse of the metric tensor. Finally, all along geodesic curves the following equations hold

$$\frac{\partial p}{\partial t} = \frac{\partial H}{\partial q}, \qquad \frac{\partial q}{\partial t} = -\frac{\partial H}{\partial p}\,. \tag{2.1}$$

Such a system of differential equations may be integrated pretty straightforwardly using simple numerical schemes such as the second order Runge Kutta integration method and Alg. 2 as in (Louis, 2019).

Algorithm 2 Computing the Exponential map

Input: $z_0 \in \mathcal{M}$, $v \in T_{z_0}\mathcal{M}$ and T
$q \leftarrow \mathbf{G} \cdot v$
$dt \leftarrow \frac{1}{T}$
for $t = 1 \to T$ **do**
  $p_{t+\frac{1}{2}} \leftarrow p_t + \frac{1}{2} \cdot dt \cdot \nabla_q H(p_t, q_t)$;
  $q_{t+\frac{1}{2}} \leftarrow q_t - \frac{1}{2} \cdot dt \cdot \nabla_p H(p_t, q_t)$;
  $p_{t+1} \leftarrow p_t + dt \cdot \nabla_q H(p_{t+\frac{1}{2}}, q_{t+\frac{1}{2}})$;
  $q_{t+1} \leftarrow q_t - dt \cdot \nabla_p H(p_{t+\frac{1}{2}}, q_{t+\frac{1}{2}})$;
end for
Return p_T

2.3 Riemannian Random Walk

A natural way to explore *geodesically-complete* Riemannian manifolds consists in using a random walk like algorithm which moves from one location to another with a certain probability. The idea here is to create a *geometry-aware* Markov Chain $(z^t)_{t\in\mathbb{N}}$ where z^{t+1} is sampled using the *wrapped* normal $z^{t+1} \sim \mathcal{N}^W(z^t, \Sigma)$. However, a drawback of such a method is that every sample of the chain is accepted regardless of its relevance. Hence, we propose to adopt the same setting as (Lebanon, 2006) where the author proposed to see the inverse metric volume element as a maximum likelihood objective to perform metric learning. For instance, if we consider the metric learned by a RHVAE, the likelihood definition writes

$$\mathcal{L}(z) = \frac{\rho_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}}{\int_{\mathbb{R}^d} \rho_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}dz}\,, \tag{2.2}$$

where $\rho_S(z) = 1$ if $z \in S$, 0 otherwise, and S is taken as a compact set so that the integral is well defined. Noteworthy is the fact that in this case, by design, the learned metric is such that it has a high volume element far from the data (Chadebec et al., 2020). This implies that it encodes in a way the amount of information contained at a specific location of the latent space. The higher the volume element, the less information we have. Hence, we propose to use this measure to assess the samples quality as an *acceptance-rejection* rate α in the chain where $\alpha(\widetilde{z}, z) = \min\left(1, \frac{\sqrt{\det \mathbf{G}^{-1}(\widetilde{z})}}{\sqrt{\det \mathbf{G}^{-1}(z)}}\right)$, z is the current state of the chain and $\widetilde{z}$ is the proposal obtained by sampling from the *wrapped* Gaussian $\mathcal{N}^W(z, \Sigma)$. The idea is to compare the relevance of the proposed sample to the current one. The ratio is such that any new sample improving the likelihood metric $\mathcal{L}$ is automatically accepted while a sample degrading the measure is more likely to be rejected in the spirit of Hasting-Metropolis sampler. A pseudo-code is provided in Alg. 3.

Algorithm 3 Riemannian random walk

Input: z_0, Σ
for $t = 1 \to T$ **do**
  Draw $v_t \sim \mathcal{N}(0, \Sigma)$
  $\widetilde{z}_t \leftarrow \mathrm{Exp}_{z_{t-1}}(v_t)$
  Accept the proposal $\widetilde{z}_t$ with probability α
end for

2.4 Experiments

We have propose a new sampling method to discover *geodesically-complete* Riemannian manifolds. We now propose to apply such method to the model proposed in Chapter 1 *i.e.* the RHVAE. Hence, we will adopt the same setting as (Chadebec et al., 2022b) and so use a RHVAE since the metric is easily computable, constraints geodesic paths to travel through most populated areas of the latent space and the learned Riemannian manifold is geodesically complete. The view we adopt is to consider the VAE as a tool to perform dimensionality reduction by extracting the latent structure of the data within a lower dimensional space. Having learned such a structure, we propose to exploit it to enhance the data generation process. This differs from the fully probabilistic view which uses the prior to generate. We believe that this is far from being optimal since the prior appears quite strongly data dependent. We now assume that we are given a latent space with a Riemannian structure where the metric has been estimated from the input data. Noteworthy is the fact that in such a case, Alg. 2 involves closed form operations since the inverse metric tensor $\mathbf{G}^{-1}$ is known and so the gradients in Eq. (2.1) can be easily computed.

2.4.1 Qualitative Comparison with Prior-Based Methods

In this section, we compare the samples quality between *prior-based* methods and ours on various standard and *real-life* data sets.

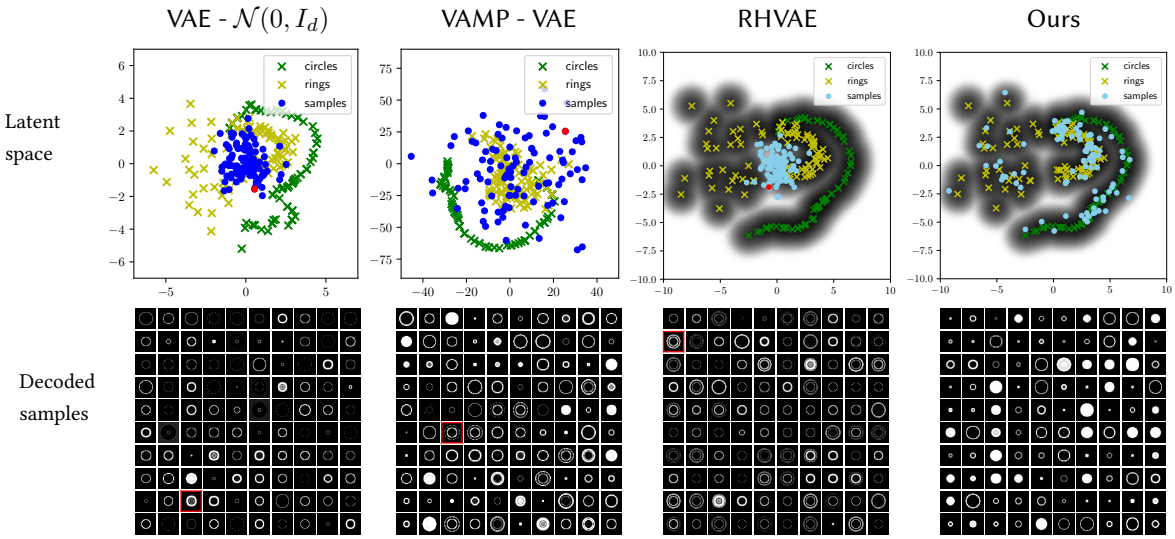


Figure 2.2: Comparison between prior-based generation methods and the proposed Riemannian random walk (ours). Top: the learned latent space with the encoded training data (crosses) and 100 samples for each method (blue dots). Bottom: the resulting decoded images. The models are trained on 180 binary circles and rings with the same neural network architectures.

Standard Data Sets

The method is first validated on a hand-made synthetic data set composed of 180 binary images of circles and rings of different diameters and thicknesses (see training samples in Fig. 2.3). We then train a VAE with a normal prior, a VAE with a VAMP prior (Tomczak and Welling, 2018) and a RHVAE until the ELBO does not improve for 50 epochs. Any relevant parameters setting is stated in Appendix 2.7.1.

Fig. 2.2 highlights the obtained samplings with each model using either the *prior-based* generation procedure or the one proposed in this chapter. The first row presents the learned latent space along with the means of the posteriors associated to the training data (crosses) and 100 latent space samples for each generation method (blue dots). The second row displays the corresponding decoded images. The first outcome of such a study is that sampling from the prior distribution $\mathcal{N}(0, I_d)$ leads to a poor latent space prospecting. Therefore, even with balanced training classes, we end up with a model over-representing certain elements of a given class (rings). This is even more striking with the RHVAE since it tends to stretch the resulting latent space. This effect seems nonetheless mitigated by the use of a multimodal prior such as the VAMP. However, another limitation of *prior-based* methods is that they may sample in locations of the latent space potentially containing very few information (*i.e.* where no data is available). Since the decoder appears to interpolate quite linearly, the classic scheme will generate images which mainly correspond to a superposition of samples (see an example with the red dots in Fig. 2.2 and the corresponding samples framed in red). Moreover, there is no way to assess a sample quality before decoding it and assessing visually its relevance. These limitations may lead to a (very) poor representation of the actual data set diversity while presenting quite a few *irrelevant* samples. Impressively, sampling along geodesic paths leads to far more diverse and sharper samples. The new sampling scheme avoids regions that have been poorly prospected

so that almost every decoded sample is visually satisfying and accounts for the data set diversity. In Fig. 2.3, we also compare the models on a *reduced* MNIST (LeCun, 1998) data set composed of 120 samples of 3 different classes and a *reduced* FashionMNIST (Xiao et al., 2017) data set composed again of 120 samples from 3 distinct classes. The models are trained with the same neural network architectures, batch size and learning rate. An early stopping strategy is adopted and consists in stopping training if the ELBO does not improve for 50 epochs. As discussed earlier, changing the prior may indeed improve the model generation capacity. For instance samples from the VAE with the VAMP prior (3^{rd} row of Fig. 2.3) are closer to the training data (1^{st} row of Fig. 2.3) than with the Gaussian prior (2^{nd} and 4^{th} row). The model is for instance able to generate circles when trained with the synthetic data while models using a standard normal prior are not. Nonetheless, a non negligible part of the generated samples are degraded (see saturated images for the *reduced* MNIST data for instance). This aspect is mitigated with the proposed generation method which generates more diverse and sharper samples.

OASIS Database

The new generation scheme is then assessed on the publicly available OASIS database composed of 416 patients aged 18 to 96, 100 of whom have been diagnosed with very mild to moderate Alzheimer disease (AD). A VAE and a RHVAE are then trained to generate either cognitively normal (CN) or AD patients with the same early stopping criteria as before. Fig. 2.4 shows samples extracted from the training set (top), MRI generated by a vanilla VAE (2^{nd} row) and images from the Riemannian random walk we propose (3^{rd} row). For each method, the upper row shows images of patients diagnosed CN while the bottom row presents an AD diagnosis. Again the proposed sampling seems able to generate a wider range of sharp samples while the VAE appears to produce non-realistic degraded images which are very similar (see red frames). For example, the proposed scheme allows us to generate realistic *old*[2] patients with no AD (blue frames) or younger patients with AD (orange frames) even though they are under-represented in the training set. Generating 100 images of OASIS database takes 1 min. with the proposed method and 40 sec.[3] with Intel Core i7 CPU (6x1.1GHz) and 16 GB RAM.

2.4.2 Discussion

It may be easily understood that the choice of the covariance matrix Σ in Alg. 3 has quite an influence on the resulting sampling. On the one hand, a Σ with strong eigenvalues will imply drawing velocities of potentially high magnitude allowing for a better prospecting but proposals are more likely to be rejected. On the other hand, small eigenvalues involve a high acceptance rate but it will take longer to prospect the manifold. An adaptive method where Σ depends on $\mathbf{G}$ may be considered and will be part of future work.

[2] An older person is characterized by larger ventricles.

[3] Depends on the chains' length (here 200 steps per image).

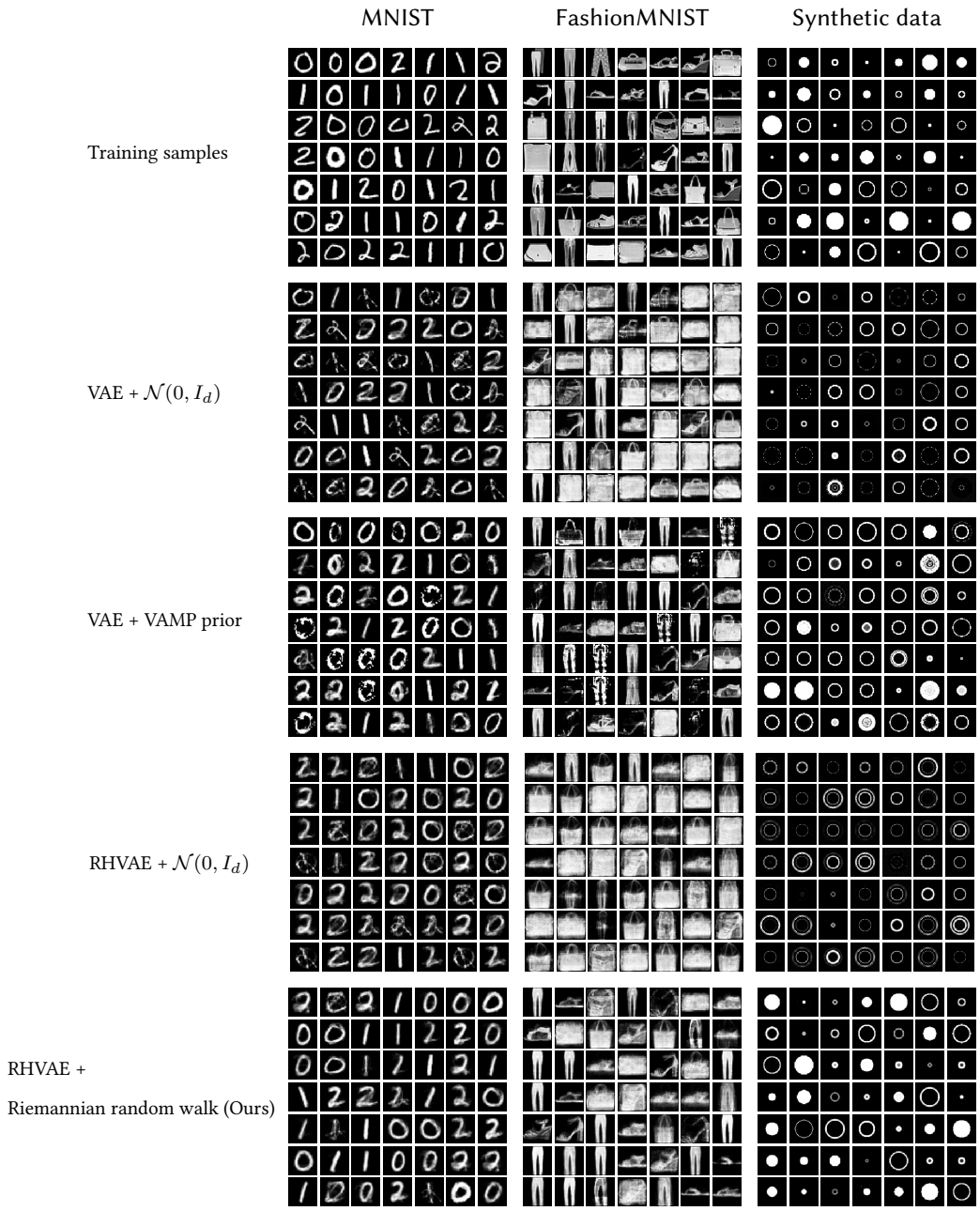


Figure 2.3: Comparison of 4 sampling methods on the *reduced* MNIST, *reduced* Fashion and the synthetic data sets. From top to bottom: 1) samples extracted from the training set; 2) samples generated with a Vanilla VAE and using the prior; 3) from the VAMP prior VAE; 4) from a RHVAE and the *prior-based* generation scheme; 5) from a RHVAE and using the proposed Riemannian random walk. All the models are trained with the same encoder and decoder networks and identical latent space dimension. An early stopping strategy is adopted and consists in stopping training if the ELBO does not improve for 50 epochs.

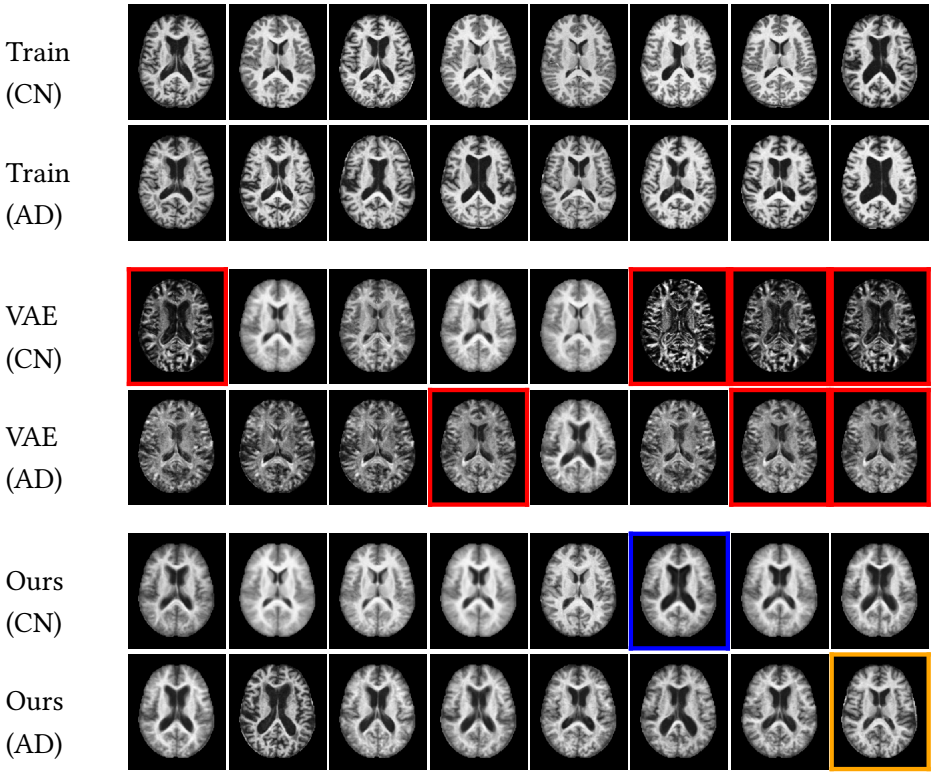


Figure 2.4: Generation of CN or AD patients from the OASIS database. Training samples (top), generation with a VAE and normal prior (2^{nd} row) and with the Riemannian random walk (bottom). Generating using the prior leads to either unrealistic images or similar samples (red frames) while the proposed method generates sharper and more diverse samples. For instance, it is able to generate CN *older* patients (blue frames) or younger AD (orange frames) even though they are under-represented within the training set.

Remark 3 *If Σ has small enough eigenvalues then Alg. 3 samples from*

$$\mathcal{L}(z) = \frac{\rho_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}}{\int_{\mathbb{R}^d} \rho_S(z)\sqrt{\det \mathbf{G}^{-1}(z)}dz}\,, \tag{2.3}$$

where $\rho_S(z) = 1$ if $z \in S$, 0 otherwise, and S is taken as a compact set so that the integral is well defined.

If Σ has small enough eigenvalues, it means that the initial velocity $v \sim \mathcal{N}(0, \Sigma)$ will have a low magnitude with high probability. In such a case, we can show with some approximation that the ratio α in the Riemannian random walk is a Hasting-Metropolis ratio with target density given by Eq. (2.3). We recall that the classic Hasting-Metropolis ratio writes

$$\alpha(x, y) = \frac{\pi(y)}{\pi(x)} \cdot \frac{q(x, y)}{q(y, x)}\,,$$

where π is the target distribution and q a proposal distribution. In the case of small magnitude

velocities, one may show that q is symmetric that is

$$q(x, y) = q(y, x)\,.$$

In our setting, a proposal $\widetilde{z}$ is made by computing the geodesic γ starting at $\gamma(0) = z$ with initial velocity $\dot{\gamma}(0) = v$ where $v \sim \mathcal{N}(0, \Sigma)$ and evaluating it at time 1. First, we remark that γ is well defined since the Riemannian manifold $\mathcal{M} = (\mathbb{R}, g)$ is *geodesically complete* and γ is unique. Moreover, we have that $\overleftarrow{\gamma}$ is the unique geodesic with initial position $\overleftarrow{\gamma}(0) = \widetilde{z} = \gamma(1)$ and initial velocity $\dot{\overleftarrow{\gamma}}(0) = -\dot{\gamma}(1)$ and we have

$$\overleftarrow{\gamma}(t) = \gamma(1 - t), \quad \forall\, t \in [0, 1]\,.$$

In the case of small enough initial velocity, a taylor expansion of the exponential may be performed next to $0 \in T_z\mathcal{M}$ and consists in approximating geodesic curves with straight lines. That is, for $t \in [0, 1]$

$$\mathrm{Exp}_z(vt) \approx z + vt\,,$$

where $v = \widetilde{z} - z$. In such a case we have

$$\dot{\gamma}(t) = v = \dot{\gamma}(0) = \dot{\gamma}(1) = -\dot{\overleftarrow{\gamma}}(0)\,.$$

Moreover, we have on the one hand

$$q(\widetilde{z}, z) = p(\widetilde{z}|z) = p(\mathrm{Exp}_z(v)|z) \simeq \mathcal{N}(z, \Sigma)\,.$$

On the other hand

$$q(z, \widetilde{z}) = p(z|\widetilde{z}) = p(\mathrm{Exp}_{\widetilde{z}}(-v)|\widetilde{z}) \simeq \mathcal{N}(\widetilde{z}, \Sigma)$$

Therefore,

$$q(\widetilde{z}, z) = q(z, \widetilde{z})\,.$$

Finally, the ratio α in the Riemannian random walk may be seen as a Hasting-Metropolis ratio where the target density is given by Eq. (2.2) and so the algorithm samples from such a distribution.

For the following DA experiments we will assume that Σ has small eigenvalues and so will sample directly using this distribution. See Sec. 2.4.1 for sampling results using the aforementioned method.

2.5 Data Augmentation Experiments For Classification

In this section, we explore the ability of the method to enrich data sets to improve classification results.

2.5.1 Augmentation Setting

We first test the augmentation method on three reduced data sets extracted from *well-known* databases MNIST and EMNIST. For MNIST, we select 500 samples applying either a balanced split or a random split ensuring that some classes are far more represented. For EMNIST, we select 500 samples

Table 2.1: Summary of OASIS database demographics, mini-mental state examination (MMSE) and global clinical dementia rating (CDR) scores.

Data set	Label	Obs.	Age	Sex M/F	MMSE	CDR
OASIS	CN	316	45.1 ± 23.9	119/197	29.1 ± 1.1	$0:316$
	AD	100	76.8 ± 7.1	41/59	24.3 ± 4.1	$0.5:70\,, 1:28, 2:2$
Train	CN	220	45.6 ± 23.6	86/134	29.1 ± 1.2	$0:220$
	AD	70	77.4 ± 6.8	29/41	23.7 ± 4.3	$0.5:47\,, 1:21, 2:2$
Val	CN	30	48.9 ± 24.1	11/19	29.2 ± 0.8	$0:30$
	AD	12	75.4 ± 7.2	4/8	25.8 ± 4.2	$0.5:7, 1:5, 2:0$
Test	CN	66	41.7 ± 24.3	22/44	29.0 ± 1.0	$0:66$
	AD	18	75.1 ± 7.5	8/10	25.8 ± 2.7	$0.5:16\,, 1:2, 2:0$

from 10 classes such that they are composed of both lowercase and uppercase characters so that we end up with a small database with strong variability within classes. These data sets are then split such that 80% is allocated for training (referred to as the *raw data*) and 20% for validation. For a fair comparison, we use the original test set (*e.g.* $\sim$1000 samples per class for MNIST) to test the classifiers. This ensures statistically meaningful results while assessing the generalization power on unseen data. We also validate the proposed DA method on the OASIS database which represents a nice example of day-to-day challenges practitioners have to face and is a benchmark database. We use 2D gray scale MR Images (208x176) with a mask notifying brain tissues and are referred to as the *masked T88 images* in (Marcus et al., 2007). We refer the reader to their paper for further image preprocessing details. We consider the binary classification problem consisting in trying to detect MRI of patients having been diagnosed with Alzheimer Disease (AD). We split the 416 images into a training set (70%) (*raw data*), a validation set (10%) and a test set (20%). A summary of demographics, mini-mental state examination (MMSE) and global clinical dementia rating (CDR) is made available in Table 2.1. On the one hand, for each data set, the train set (*raw data*) is augmented by a factor 5, 10 and 15 using classic DA methods (random noise, cropping etc.). On the other hand, VAE models are trained individually on each class of the *raw data*. The generative models are then used to produce 200, 500, 1k or 2k synthetic samples per class with either the classic generation scheme (*i.e.* the prior) or the proposed method. We then train classifiers with 5 independent runs on 1) the *raw data*; 2) the augmented data using basic transformations; 3) the augmented data using the VAE models; 4) only the synthetic data generated by the VAEs. A DenseNet model[4] (Huang et al., 2017) is used for the toy data while we also train hand made MLP and CNN models on OASIS (See Appendix 2.7.2). The main metrics obtained on the test set are reported in Tables 2.2 and 2.3.

2.5.2 Results

Toy Data

As expected generating new samples using the proposed method improves their relevance. The method indeed allows for a quite impressive gain in the model accuracy when synthetic samples

[4]We use the code in (Amos, 2020) (See Appendix 2.7.2).

Table 2.2: DA on *toy* data sets. Mean accuracy and standard deviation across 5 independent runs are reported. In gray are the cells where the accuracy is higher on synthetic data than on the *raw data*.

Data sets	MNIST	MNIST**	EMNIST**	MNIST	MNIST**	EMNIST**
Raw data	89.9(0.6)	81.6(0.7)	82.6(1.4)	-	-	-
	Raw + Synthetic			Synthetic only		
Aug. (X5)	92.8(0.4)	86.5(0.9)	85.6(1.3)	-	-	-
Aug. (X10)	88.3(2.2)	82.0(2.4)	85.8(0.3)	-	-	-
Aug. (X15)	92.8(0.7)	85.9(3.4)	86.6(0.8)	-	-	-
VAE-200*	88.5(0.9)	84.1(2.0)	81.7(3.0)	69.9(1.5)	64.6(1.8)	65.7(2.6)
VAE-500*	90.4(1.4)	87.3(1.2)	83.4(1.6)	72.3(4.2)	69.4(4.1)	67.3(2.4)
VAE-1k*	91.2(1.0)	86.0(2.5)	84.4(1.6)	83.4(2.4)	74.7(3.2)	75.3(1.4)
VAE-2k*	92.2(1.6)	88.0(2.2)	86.0(0.2)	86.6(2.2)	79.6(3.8)	78.9(3.0)
RHVAE-200*	89.9(0.5)	82.3(0.9)	83.0(1.3)	76.0(1.8)	61.5(2.9)	59.8(2.6)
RHVAE-500*	90.9(1.1)	84.0(3.2)	84.4(1.2)	80.0(2.2)	66.8(3.3)	67.0(4.0)
RHVAE-1k*	91.7(0.8)	84.7(1.8)	84.7(2.4)	82.0(2.9)	69.3(1.8)	73.7(4.1)
RHVAE-2k*	92.7(1.4)	86.8(1.0)	84.9(2.1)	85.2(3.9)	77.3(3.2)	68.6(2.3)
Ours-200*	91.0(1.1)	84.1(2.0)	85.1(1.1)	87.2(1.1)	79.5(1.6)	77.1(1.6)
Ours-500*	92.3(1.1)	87.7(0.9)	85.1(1.1)	89.1(1.3)	80.4(2.1)	80.2(2.0)
Ours-1k*	93.3(0.8)	**89.7(0.8)**	87.0(1.0)	90.2(1.4)	86.2(1.8)	82.6(1.3)
Ours-2k*	**94.3(0.8)**	89.1(1.9)	**87.6(0.8)**	**92.6(1.1)**	**87.6(1.3)**	**86.0(1.0)**

* Number of generated samples ** Unbalanced data sets

are added to the real ones (leftmost column of Table 2.2). This is even more striking when looking at the rightmost column where only synthetic samples are used to train the classifier. For instance, when only 200 synthetic samples per class for MNIST are generated with a VAE and used to train the classifier, the classic method fails to produce meaningful samples since a loss of 20 pts in accuracy is observed when compared to the *raw data*. Interestingly, our method seems to avoid such an effect. Even more impressive is the fact that we are able to produce synthetic data sets on which the classifier outperforms greatly the results observed on the *raw data* (3 to 6 pts gain in accuracy) while keeping a relatively low standard deviation (see gray cells). Secondly, this example also shows why geometric DA is still questionable and remains data dependent. For instance, augmenting the *raw data* by a factor 10 (including flips and rotations) does not seem to have a notable effect on the MNIST data sets but still improves results on EMNIST. On the contrary, our method seems quite **robust to data set changes**.

OASIS

Balanced accuracy obtained on OASIS with 3 classifiers is made available in Table 2.3. In this experiment, using the new generation scheme again improves overall the metric for each classifier when compared to the *raw data* and other augmentation methods. Moreover, the strong relevance of the created samples is again supported by the fact that the classifiers are again able to strongly outperform the results on the *raw data* even when trained only with synthetic ones. Finally, the method appears **robust to classifiers** and can be used with high-dimensional complex data such as MRI.

Table 2.3: DA on OASIS data base. Mean balanced accuracy on independent 5 runs with several classifiers.

Networks	MLP		CNN		Densenet	
Raw data	80.7(4.1)	-	72.5(3.5)	-	77.4(3.3)	-
	Raw + Synthetic	Synthetic Only	Raw + Synthetic	Synthetic Only	Raw + Synthetic	Synthetic Only
Aug. (X5)	84.3(1.3)	-	80.0(3.5)	-	73.9(5.1)	-
Aug. (X10)	76.0(2.8)	-	82.8(3.7)	-	78.3(4.1)	-
Aug. (X15)	78.7(5.3)	-	80.3(3.7)	-	76.6(1.1)	-
VAE-200*	80.7(1.5)	77.8(1.3)	79.4(3.6)	65.0(12.3)	76.5(3.2)	74.0(3.0)
VAE-500*	79.7(1.4)	77.4(1.5)	72.6(7.0)	70.2(5.0)	74.9(4.3)	72.8(1.8)
VAE-1000*	81.3(0.0)	76.5(0.6)	74.4(9.4)	73.0(3.3)	73.5(1.3)	74.9(2.6)
VAE-2000*	80.7(0.3)	78.1(1.6)	71.1(4.9)	76.9(2.6)	74.0(4.9)	73.3(3.4)
Ours-200*	84.3(0.0)	86.7(0.4)	76.4(5.0)	75.4(6.6)	78.2(3.0)	74.3(4.8)
Ours-500*	**87.2(1.2)**	**88.6(1.1)**	81.8(4.6)	81.8(3.7)	80.2(2.8)	**84.2(2.8)**
Ours-1000*	84.2(0.3)	84.4(1.8)	83.5(3.2)	79.8(2.8)	82.2(4.7)	76.7(3.8)
Ours-2000*	85.3(1.9)	84.2(3.3)	**84.5(1.9)**	**83.9(1.9)**	**82.9(1.8)**	73.6(5.8)

* Number of generated samples

2.6 Conclusion

In this chapter, we proposed a new way to explore *geodesically-complete* Riemannian manifolds using a random walk like algorithm. We then applied the method to a Variational Autoencoder which has learned the latent geometry of the input data. This method was then used to perform DA to improve classification tasks in the low sample size setting on both toy and real data and with different kind of classifiers. In each case, the method allows for an impressive gain in the classification metrics (*e.g.* balanced accuracy jumps from 80.7 to 88.6 on OASIS). Moreover, the relevance of the generated data was supported by the fact that classifiers were able to perform better when trained with only synthetic data than on the *raw data* in all cases. Future work would consist in using the method on even more challenging data such as 3D volumes and using smaller data sets.

Acknowledgment

The research leading to these results has received funding from the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and reference ANR-10-IAIHU-06 (Agence Nationale de la Recherche-10-IA Institut Hospitalo-Universitaire-6). Data were provided in part by OASIS: Cross-Sectional: Principal Investigators: D. Marcus, R, Buckner, J, Csernansky J. Morris; P50 AG05681, P01 AG03991, P01 AG026276, R01 AG021910, P20 MH071616, U24 RR021382

2.7 Appendices

2.7.1 VAEs Parameters Setting

Table 2.4 summarizes the main hyper-parameters we use to perform the experiments presented in the chapter while Table 2.5 shows the neural networks architectures employed. As to training parameters, we use a Adam optimizer (Kingma and Ba, 2014) with a learning of 10^{-3}. For the augmentation experiments, we stop training if the ELBO does not improve for 20 epochs for all data sets except for OASIS where the learning rate is decreased to 10^{-4} and training is stopped if the ELBO does not improve for 50 epochs.

Table 2.4: RHVAE parameters for each data set.

Data sets	Parameters					
	d^*	n_{lf}	ε_{lf}	T	λ	$\sqrt{\beta_0}$
Synthetic	2	3	10^{-2}	0.8	10^{-3}	0.3
reduced Fashion	2	3	10^{-2}	0.8	10^{-3}	0.3
MNIST (bal.)	2	3	10^{-2}	0.8	10^{-3}	0.3
MNIST (unbal.)	2	3	10^{-2}	0.8	10^{-3}	0.3
EMNIST	2	3	10^{-2}	0.8	10^{-3}	0.3
OASIS	2	3	10^{-3}	0.8	10^{-2}	0.3

* Latent space dimension (same for VAE and VAMP-VAE)

Table 2.5: Neural networks architectures of the VAE, VAMP-VAE and RHVAE for each data set. The *encoder* and *decoder* are the same for all models.

Net	Layer 1	Layer 2	Layer 3
Synthetic, MNIST & Fashion			
μ^*_ϕ	(D, 400, ReLU)	(400, d, lin.)	-
Σ^*_ϕ		(400, d, lin.)	-
π^*_θ	(d, 400, ReLU)	(400, D, sig.)	-
L_ψ (diag.)	(D, 400, ReLU)	(400, d, lin.)	-
L_ψ (low.)		(400, $\frac{d(d-1)}{2}$, lin.)	-
OASIS			
μ^*_ϕ	(D, 1k, ReLU)	(1k, 400, relu)	(400, d, lin.)
Σ^*_ϕ			(400, d, lin.)
π^*_θ	(d, 400, ReLU)	(400, 1k, ReLU)	(1k, D, sig.)
L_ψ (diag.)	(D, 400, ReLU)	(400, d, lin.)	-
L_ψ (low.)		(400, $\frac{d(d-1)}{2}$, lin.)	-

* Same for all VAE models

2.7.2 Classifier Parameter Setting

As to the models used as benchmark for data augmentation, the DenseNet implementation we use is the one in (Amos, 2020) with a *growth rate* equals to 10, *depth* of 20 and 0.5 *reduction* and is trained with a learning rate of 10^{-3}. For OASIS, the MLP has 400 hidden units and ReLU activation function and the CNN is as follows

Table 2.6: CNN classifier architecture used. Each convolutional block has a padding of 1.

Layer	Architectures
input	$(1, 208, 176)$
Layer 1	Conv2D(1, 8, kernel=(3, 3), stride=1) Batch normalization LeakyReLU MaxPool (2, 2, stride=2)
Layer 2	Conv2D(8, 16, kernel=(3, 3), stride=1) Batch normalization LeakyReLU MaxPool (2, 2, stride=2)
Layer 3	Conv2D(16, 32, kernel=(3, 3), stride=2) Batch normalization LeakyReLU MaxPool (2, 2, stride=2)
Layer 4	Conv2D(32, 64, kernel=(3, 3), stride=2) Batch normalization LeakyReLU MaxPool (2, 2, stride=2)
Layer 5	MLP(256, 100) ReLU
Layer 6	MLP(100, 2) Log Softmax

For the toy data, the DenseNet is trained until the loss does not improve on the validation set for 50 epochs. On OASIS, we make a random search on the learning rate for each model chosen in the range $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$. The model is trained on 5 independent runs with the same learning rate and keep the model achieving the best mean balanced accuracy on the validation set. For the CNN and MLP, we stop training if the validation loss does not improve for 20 epochs and for the densenet training is stopped if no improvement is observed on the validation loss for 10 epochs. Each model is trained with an Adam optimizer.

Chapter 3

A Geometric Perspective on Variational Autoencoders

This chapter introduces a new interpretation of the Variational Autoencoder framework by taking a fully geometric point of view. We argue that vanilla VAE models unveil naturally a Riemannian structure in their latent space and that taking into consideration those geometrical aspects can lead to better interpolations and an improved generation procedure. This new proposed sampling method consists in sampling from the uniform distribution deriving intrinsically from the learned Riemannian latent space, and we show that using this scheme can make a vanilla VAE competitive and even better than more advanced versions on several benchmark datasets. Since generative models are known to be sensitive to the number of training samples we also stress the method's robustness in the low data regime.

This chapter was published at the NeurIPS Conference 2022. See (Chadebec and Allassonnière, 2022).

3.1 Introduction

Variational Autoencoders (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) are powerful generative models that map complex input data in a much lower dimensional space referred to as the latent space while driving the latent variables to follow a given prior distribution. Their simplicity to use in practice has made them very attractive models to perform various tasks such as high-fidelity image generation (Razavi et al., 2020), speech modeling (Blaauw and Bonada, 2016), clustering (Yang et al., 2019) or data augmentation (Chadebec et al., 2022b).

Nonetheless, when taken in their simplest version, it was noted that these models produce blurry samples on image generation tasks most of the time. This undesired behavior may be due to several limitations of the VAE framework. First, the training of a VAE aims at maximizing the Evidence Lower BOund (ELBO) which is only a lower bound on the true likelihood and so does not ensure that we are always actually improving the true objective (Burda et al., 2016; Alemi et al., 2016; Higgins et al., 2017; Cremer et al., 2018; Zhang et al., 2018a). Second, the prior distribution may be too simplistic (Dai and Wipf, 2018) leading to poor data generation and there exists no guarantee that the actual distribution of the latent code will match a given prior distribution inducing distribution mismatch (Connor et al., 2021). Hence, trying to tackle those limitations through richer posterior distributions (Salimans et al., 2015; Rezende and Mohamed, 2015) or better priors (Tomczak and Welling, 2018) represents a major part of the proposed improvements over the past few years. However, the tractability of the ELBO constrains the choice in distributions and so finding a trade-off between model expressiveness and tractability remains crucial. In this chapter, we take a rather different approach and focus on the geometrical aspects a vanilla VAE is able to capture in its latent space. In particular, we propose the following contributions:

- We show that VAEs unveil naturally a latent space with a structure that can be modeled as a Riemannian manifold through the learned covariance matrices in the variational posterior distributions and that such modeling can lead to better interpolations.
- We propose a natural sampling scheme consisting in sampling from a uniform distribution defined on the learned manifold and given by the Riemannian metric. We show that this procedure improves the generation process from a *vanilla* VAE significantly without complexifying the model nor the training. The proposed sampling method outperforms more advanced VAE models in terms of Frechet Inception Distance (Heusel et al., 2017) and Precision and Recall (Sajjadi et al., 2019) scores on four benchmark datasets. We also discuss and show that it can benefit more recent VAEs as well. An implementation is available on github.
- We show that the method appears robust to dataset size changes and outperforms even more strongly peers when only *smaller* sample sizes are considered.
- We discuss the link of the proposed metric to the *pull-back* metric.

3.2 Variational Autoencoders

Considering that we are given $x \in \mathbb{R}^D$ a set of data points deriving from an unknown distribution $p(x)$, a VAE aims at inferring p with a parametric model $\{p_\theta, \theta \in \Theta\}$ using a maximum likelihood

estimator. A key assumption behind the VAE is to assume that the generation process involves latent variables z living in a lower dimensional space such that the generative model writes

$$z \sim p(z) \quad ; \quad x \sim p_\theta(x|z)\,,$$

where p is a prior distribution over the latent variables often taken as a standard Gaussian and $p_\theta(x|z)$ is referred to as the decoder and is most of the time taken as a parametric distribution the parameters of which are estimated using neural networks. Hence, the likelihood p_θ writes:

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p(z)dz\,.$$

As this integral is most of the time intractable so is $p_\theta(z|x)$, the posterior distribution. Hence, Variational Inference (Jordan et al., 1999) is used and a simple parametrized variational distribution $q_\phi(z|x)$ is introduced to approximate the posterior $p_\theta(z|x)$. $q_\phi(z|x)$ is referred to as the *encoder* and, in the vanilla VAE, q_ϕ is chosen as a multivariate Gaussian whose parameters μ_ϕ and $\mathbf{\Sigma}_\phi$ are again given by neural networks. An unbiased estimate $\hat{p}_\theta$ of the likelihood $p_\theta(x)$ can then be derived using importance sampling with $q_\phi(z|x)$ and the ELBO objective follows using Jensen's inequality:

$$\begin{aligned} \log\ p_\theta(x) = \log \mathbb{E}_{z\sim q_\phi}\left[\hat{p}_\theta\right] &\geq \mathbb{E}_{z\sim q_\phi}\left[\log \hat{p}_\theta\right] \\ &\geq\ \mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z) - \mathrm{KL}(q_\phi(z|x)\|p(z)) = \mathcal{L} \end{aligned} \tag{3.1}$$

The ELBO is now tractable since both $q_\phi(z|x)$ and $p_\theta(x|z)$ are known and so can be optimized with respect to the *encoder* and *decoder* parameters.

Remark 4 *In practice, $p_\theta(x|z)$ is chosen depending on the modeling of the input data but is often taken as a simple distribution (*e.g *fixed variance Gaussian, Bernoulli ...) and a weight β can be applied to balance the weight of the* KL *term (Higgins et al., 2017). Hence, the ELBO can also be seen as a two terms objective (Ghosh et al., 2020). The first one is a reconstruction term given by $p_\theta(x|z)$ while the second one is a regularizer given by the KL between the variational posterior q_ϕ and the prior p. For instance, in the case of a fixed variance Gaussian for $p_\theta(x|z)$ we have*

$$\mathcal{L}_{REC} = \|x - \mu_\theta(z)\|_2^2, \quad \mathcal{L}_{REG} = \beta \cdot \mathrm{KL}(q_\phi(z|x)\|p(z))\,. \tag{3.2}$$

3.3 Related Work

A natural way to improve the generation from VAEs consists in trying to use more complex priors (Hoffman and Johnson, 2016) than the standard Gaussian distribution used in the initial version such that they better match the true distribution of the latent codes. For instance, using a Mixture of Gaussian (Nalisnick et al., 2016; Dilokthanakul et al., 2017) or a Variational Mixture of Posterior (VAMP) (Tomczak and Welling, 2018) as priors was proposed. In the same vein, hierarchical latent variable models (Sønderby et al., 2016; Klushyn et al., 2019) or prior learning (Chen et al., 2016b; Aneja et al., 2020) have recently emerged and aimed at finding the best suited prior distribution for a given dataset. Acceptance/rejection sampling method was also proposed to try to improve the expressiveness of the prior distribution (Bauer and Mnih, 2019a). Some recent works linking

energy-based models (EBM) and VAEs (Xiao et al., 2020) or modeling the prior as an EBM (Pang et al., 2020) have demonstrated promising results and are also worth citing.

On the ground that the latent space must adapt to the data as well, *geometry-aware* latent space modelings as hypersphere (Davidson et al., 2018), torus (Falorsi et al., 2018) or Poincaré disk (Mathieu et al., 2019a) or discrete latent representations (Razavi et al., 2020) were proposed. Other recent contributions proposed to see the latent space as a Riemannian manifold where the Riemannian metric is given by the Jacobian of the generator function (Arvanitidis et al., 2018; Chen et al., 2018a; Shao et al., 2018). This metric was then used directly within the prior modeled by Brownian motions (Kalatzis et al., 2020). Others proposed to learn the metric directly from the data throughout training thanks to *geometry-aware* normalizing flows (Chadebec et al., 2020) or learn the latent structure of the data using transport operators (Connor et al., 2021). While these geometry-based methods show interesting properties of the learned latent space they either require the computation of a time consuming model-dependent function, the Jacobian, or add further parameters to the model to learn the metric or transport operators adding some computational burden.

Arguing that VAEs are essentially Autoencoders regularized with a Gaussian noise, Ghosh et al. (2020) proposed another interesting interpretation of the VAE framework and showed that other types of regularization may be of interest as well. Since the generation process from these Autoencoders is no longer relying on the prior distribution, the authors proposed to use ex-post density estimation by fitting simple distributions such as a Gaussian mixture in the latent space. While this paves the way for consideration of other ways to generate data, it mainly reduces the VAE framework to an Autoencoder while we believe that it can also unveil interesting geometrical aspects.

Another widely discussed improvement of the model consists in trying to tweak the approximate posterior in the ELBO so that it better matches the true posterior using MCMC methods (Salimans et al., 2015) or normalizing flows (Rezende and Mohamed, 2015). For instance, methods using Hamiltonian equations in the flows to target the true posterior (Caterini et al., 2018) were proposed.

Finally, while discussing the potential link between PCA and Autoencoders some intuitions arose on the impact of both the intrinsic structure of the variance of the data (Rakowski and Lippert, 2021) and the shape of the covariance matrices in the posterior distributions (Rolinek et al., 2019) on disentanglement in the latent space. We also believe that these covariance matrices indeed play a crucial role in the modeling of the latent space but in this chapter, we instead propose to see their inverse as the value of a Riemannian metric.

3.4 Proposed Method

In this section, we show that a vanilla VAE unveils naturally a Riemannian structure in its latent space through the learned covariance matrices in the variational posterior distribution. We then propose a new natural generation scheme guided by this estimated geometry and consisting in sampling from a uniform distribution deriving intrinsically from the learned Riemannian manifold.

3.4.1 A Word on Riemannian Geometry

First, we briefly recall some basic elements of Riemannian geometry needed in the rest of the chapter. A more detailed discussion on Riemannian manifolds may be found in Appendix 3.7.1. A d-dimensional manifold $\mathcal{M}$ is a manifold which is locally homeomorphic to a d-dimensional Euclidean space. If the manifold $\mathcal{M}$ is further differentiable it possesses a tangent space T_z at any $z \in \mathcal{M}$ composed of the tangent vectors of the curves passing by z. If $\mathcal{M}$ is equipped with a smooth inner product $g : z \to \langle \cdot | \cdot \rangle_z$ defined on its tangent space T_z for any $z \in \mathcal{M}$ then $\mathcal{M}$ is called a Riemannian manifold and g is the associated Riemannian metric. Then, a local representation of g at any $z \in \mathcal{M}$ is given by the positive definite matrix $\mathbf{G}(z)$ (See Appendix 3.7.1). If $\mathcal{M}$ is connected, a Riemannian distance between two points z_1, z_2 of $\mathcal{M}$ can be defined

$$\mathrm{dist}_{\mathbf{G}}(z_1, z_2) = \inf_{\gamma} \int_a^b \sqrt{\dot{\gamma}(t)^\top \mathbf{G}(\gamma(t))\dot{\gamma}(t)} dt = \inf_{\gamma} L(\gamma) \quad \text{s.t.} \quad z_1 = \gamma(a), z_2 = \gamma(b)\,, \tag{3.3}$$

where L is the length of curves $\gamma : \mathbb{R} \to \mathcal{M}$ traveling from z_1 to z_2. Curves minimizing L and parametrized proportionally to the arc length are *geodesic*. The manifold $\mathcal{M}$ is said to be *geodesically complete* if all geodesic curves can be extended to $\mathbb{R}$. In an Euclidean space, $\mathbf{G}$ reduces to I_d and the distance becomes the classic Euclidean one. A simple extension of this Euclidean framework consists in assuming that the metric is given by a constant positive definite matrix $\mathbf{\Sigma}$ different from I_d. In such a case the induced Riemannian distance is the well-known Mahalanobis distance $\mathrm{dist}_{\mathbf{\Sigma}}(z_1, z_2) = \sqrt{(z_2 - z_1)^\top \mathbf{\Sigma} (z_2 - z_1)}$.

3.4.2 The Riemannian Gaussian Distribution

Given the Riemannian manifold $\mathcal{M}$ endowed with the Riemannian metric $\mathbf{G}$ and a chart z, an infinitesimal volume element may be defined on each tangent space T_z of the manifold $\mathcal{M}$ as follows

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)} dz\,, \tag{3.4}$$

with dz being the Lebesgue measure. This defines a canonical measure on the manifold and allows to extend the notion of random variables to Riemannian manifolds whose density can be defined with respect to that Riemannian measure (see Appendix 3.7.1). Hence, a Riemannian Gaussian distribution on $\mathcal{M}$ can be defined using the Riemannian distance of Eq. (3.3) instead of the Euclidean one.

$$\mathcal{N}_{\mathrm{riem}}(z|\sigma, \mu) = \frac{1}{C} \exp\Big(- \frac{\mathrm{dist}_{\mathbf{G}}(z, \mu)^2}{2\sigma} \Big)\,, \qquad C = \int_{\mathcal{M}} \exp\Big(- \frac{\mathrm{dist}_{\mathbf{G}}(z, \mu)^2}{2\sigma} \Big) d\mathcal{M}_z\,, \tag{3.5}$$

where $d\mathcal{M}_z$ is the volume element defined in Eq. (3.4). Thus, a multivariate normal distribution with covariance matrix $\mathbf{\Sigma}$ is only a specific case of the Riemannian distribution with $\sigma = 1$ and defined on the manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ where $\mathbf{G}$ is the constant Riemannian metric $\mathbf{G}(z) = \mathbf{\Sigma}^{-1},\ \forall z \in \mathcal{M}$.

3.4.3 Geometrical Interpretation of the VAE Framework

Within the VAE framework, the variational distribution $q_\phi(z|x)$ is often chosen as a simple multivariate Gaussian distribution defined on $\mathbb{R}^d$ with d being the latent space dimension. Hence, as explained in the previous section, given an input data point x_i, the posterior $q_\phi(z|x_i) = \mathcal{N}(\mu(x_i), \mathbf{\Sigma}(x_i))$ can also be seen as a Riemannian Gaussian distribution where the Riemannian distance is simply the distance with respect to the metric tensor $\mathbf{\Sigma}^{-1}(x_i)$. Hence, the VAE framework can be seen as follow: As with an Autoencoder, the VAE provides a code $\mu(x_i)$ which is a lower dimensional representation of an input data point x_i. However, it also gives a tensor $\mathbf{\Sigma}^{-1}(x_i)$ depending on x_i which can be seen as the value of a Riemannian metric $\mathbf{G}$ at $\mu(x_i)$ *i.e.*

$$\mathbf{G}(\mu(x_i)) = \mathbf{\Sigma}^{-1}(x_i)\,.$$

This metric is crucial since it impacts the notion of distance in the latent space now seen as the Riemannian manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ and so changes the directions that are favored in the sampling from the posterior distribution $q_\phi(z|x_i)$. Then, a sample z is drawn from a standard (*i.e.* $\sigma = 1$ in Eq. (3.5)) Riemannian Gaussian distribution and fed to the decoder. Since we only have access to a finite number of metric tensors $\mathbf{\Sigma}^{-1}(x_i)$, as a first approximation the VAE model assumes that the metric is locally constant close to $\mu(x_i)$ and so the Riemannian distance reduces to the Mahalanobis distance in the posterior distribution. This drastically simplifies the training process since now Riemannian distances have closed form and so are easily computable. Interestingly, the VAE framework will impose through the ELBO expression given in Eq. (3.2), that z gives a sample $x \sim p_\theta(x|z)$ close to x_i when decoded. Since z has a probability density function imposing higher probability for samples having the smallest Riemannian distance to μ, the VAE imposes in a way that latent variables that are close in the latent space with respect to the metric $\mathbf{G}$ will also provide samples that are close in the data space $\mathcal{X}$ in terms of L2 distance as noticed in Remark. 4. Noteworthy is that the latter distance can be amended through the choice of the decoder $p_\theta(x|z)$. This is an interesting property since it allows the VAE to directly link the learned Riemannian distance in the latent space to the distance in the data space. The regularization term in Eq. (3.2) ensures that the covariance matrices do not collapse to $\mathbf{0}_d$ and constraints the latent codes to remain close to the origin easing optimization. Finally, at the end of training, we have a lower dimensional representation of the training data given by the means of the posteriors $\mu(x_i)$ and a family of metric tensors ($\mathbf{G}_i = \mathbf{\Sigma}^{-1}(x_i)$) corresponding to the value of a Riemannian metric defined locally on the latent space. Inspired from (Hauberg et al., 2012), we propose to build a smooth continuous Riemannian metric defined on the entire latent space as follows:

$$\begin{aligned}
\mathbf{G}(z) &= \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau\|z\|_2^2} \cdot I_d\,, \\
\omega_i(z) &= \exp\left(-\frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2}\right),
\end{aligned} \tag{3.6}$$

where $\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2 = (z-\mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z-\mu(x_i))$ is the Riemannian distance between z and $\mu(x_i)$ with respect to the locally constant metric $\mathbf{G}(\mu(x_i)) = \mathbf{\Sigma}^{-1}(x_i)$. Since the sum in Eq. (3.6) is made on the total number of training samples N, the number of centroids ($\mu(x_i)$) and so of reference metric tensors can be decreased for huge datasets by selecting only $k < N$ elements[1]

[1]This may be performed with k-medoids algorithm.

and increasing ρ to reduce memory usage. We provide an ablation study on the impact of λ, the number of centroids and their choice along with a discussion on the choice for ρ in Appendix 3.7.6. The parameter τ is only there to ensure that the volume of $(\mathbb{R}^d, \mathbf{G})$ is finite, a property that is needed in Sec. 3.4.5, and its value can be set as close as desired to zero so the norm of z does not influence the metric close to the centroids. In practice, it is set below computer precision (*i.e.* $\tau \approx 0$). Rigorously, the metric defined in Eq. (3.6) should have been used during the training process. Nonetheless, this would have made the training longer and trickier since it would involve i) the computation of Riemannian distances that have no longer closed form and so make the resolution of the optimization problem in Eq. (3.3) needed, ii) the sampling from Eq. (3.5) which is not trivial and iii) the computation of the regularization term. Moreover, for small values of β in Eq. (3.2), the samples generated from the variational distribution $z \sim \mathcal{N}(\mu(x_i), \mathbf{\Sigma}(x_i))$ can be assumed to be concentrated around $\mu(x_i)$ and so we have the following first-order Taylor expansion around $\mu(x_i)$

$$\mathbf{G}(z) \approx \mathbf{\Sigma}^{-1}(x_i) + \sum_{j=1, j\neq i}^{N} \mathbf{\Sigma}^{-1}(x_j) \cdot \underbrace{\omega_j(\mu(x_i))}_{\approx 0} + \mathbf{\Sigma}^{-1}(x_i) \cdot \underbrace{\mathbf{J}_{\omega_i}(\mu(x_i))}_{=0}(z - \mu(x_i)) \,,$$

where $\mathbf{J}_{\omega_i}(\mu(x_i))$ is the Jacobian of the interpolant ω_i evaluated at $\mu(x_i)$. Note that we have further assumed small enough ρ and λ to neglect the influence of the other $\mathbf{\Sigma}(x_j)$ in Eq. (3.6). Hence by approximating the value of the metric during training by its value at $\mu(x_i)$ (*i.e.* $\mathbf{\Sigma}^{-1}(x_i)$), the VAE training remains unchanged, stable and computationally reasonable since Riemannian Gaussians become multivariate Gaussians in $q_\phi(z|x)$ as explained before. Noteworthy is the fact that following the discussion on the role of the KL loss in the VAE framework and the experiments conducted in (Ghosh et al., 2020), in our vision of the VAE, the prior distribution is only seen as a regularizer though the KL term and other latent space regularization schemes may have been also envisioned. In the following, we keep the proposed vision and do not amend the training.

3.4.4 Link with the *pull-back* Metric

It has been shown that a natural Riemannian metric on the latent space of generative models can be the *pull-back* metric given by $\mathbf{G}(z) = \mathbf{J}_g(z)^\top \mathbf{J}_g(z)$ (Arvanitidis et al., 2018) and induced by the decoder mapping $g : \mathbb{R}^d \to \mathbb{R}^D$ outputting the parameters of the conditional distribution $p_\theta(x|z)$. Actually, there exists a strong relation linking the metric proposed in this chapter to the *pull-back* metric. Indeed, assuming that samples from the variational posterior $z \sim q_\phi(z|x) = \mathcal{N}(\mu(x), \mathbf{\Sigma}(x))$ remain close to $\mu(x)$ (*e.g.* by setting a small β in Eq. (3.2)) allows to consider an approximation of the log density $h(z) \coloneqq \log p_\theta(x|z)$ next to $\mu(x)$ for a given x (Kumar and Poole, 2020).

$$h(z) \approx\ h(\mu(x)) + \mathbf{J}_h(\mu(x))(z - \mu(x))\ +\ \frac{1}{2}(z - \mu(x))^\top \mathbf{H}_h(\mu(x))(z - \mu(x)) \,,$$

where $\mathbf{J}_h(\mu(x))$ is the Jacobian and $\mathbf{H}_h(\mu(x))$ is the Hessian of h. Using this and remarking that

$$\mathbb{E}_{z\sim q_\phi}\Big[\mathbf{J}_h(\mu)(z - \mu)\Big] = 0 \quad \text{and} \quad \mathbb{E}_{z\sim q_\phi}\Big[(z - \mu)^\top \mathbf{H}_h(\mu)(z - \mu)\Big] = \mathrm{Tr}(\mathbf{H}_h(\mu)\mathbf{\Sigma}) \,,$$

makes the ELBO in Eq. (3.2) write:

$$\mathcal{L} \approx h(\mu(x)) + \frac{1}{2}\mathrm{Tr}(\mathbf{H}_h(\mu(x))\mathbf{\Sigma}(x)) - \beta\ \mathrm{KL}(q_\phi(z|x)\|p(z)) \,.$$

Assuming a standard Gaussian prior, Kumar and Poole (2020) showed that $\widetilde{\mathbf{\Sigma}}$ maximizing the ELBO is

$$\widetilde{\mathbf{\Sigma}}(x) = \left(I_d - \frac{1}{\beta}\mathbf{H}_h(\mu(x))\right)^{-1},$$

and if we further assume some regularity on the neural networks used for the decoder mapping g (*e.g.* piece-wise linear activation functions) we have

$$\widetilde{\mathbf{\Sigma}}(x) = \left(I_d - \frac{1}{\beta}\mathbf{J}_g(\mu(x))^\top \mathbf{H}_p(g(\mu(x)))\mathbf{J}_g(\mu(x))\right)^{-1}, \tag{3.7}$$

where $\mathbf{H}_p(g(\mu(x)))$ is the Hessian of $\log p_\theta(x; g(z))$. A standard case for the VAE is to assume that $p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma \cdot I_D)$ and so gives $\mathbf{H}_p(g(\mu(x))) = -\frac{1}{\sigma} \cdot I_D$. If we further set $\sigma = \frac{1}{\beta}$, Eq. (3.7) gives a relation between the *pull-back* and the metric we propose

$$\widetilde{\mathbf{\Sigma}}^{-1}(x) = \mathbf{J}_g(\mu(x))^\top \mathbf{J}_g(\mu(x)) + I_d\,.$$

Hence, the proposed metric is closely linked to the *pull-back* metric and may be useful to approximate it (at least close to the $\mu(x)$) and so avoid the computation of a potentially costly function.

3.4.5 Geometry-Aware Sampling

Assuming that the VAE has learned a latent representation of the data in a space seen as a Riemannian manifold, we propose to exploit this strong property to enhance the generation procedure. A natural way to sample from such a latent space would consist in sampling from the uniform distribution intrinsically defined on the learned manifold. Similar to the Gaussian distribution presented in Sec. 3.4.2, the notion of uniform distribution can indeed be extended to Riemannian manifolds. Given a set $\mathcal{A} \subset \mathcal{M}$ having a finite volume, a Riemannian uniform distribution on $\mathcal{A}$ writes (Pennec, 2006)

$$p_{\mathcal{A}}(z) = \frac{\mathbf{1}_{\mathcal{A}}(z)}{\mathrm{Vol}(\mathcal{A})} = \frac{\mathbf{1}_{\mathcal{A}}(z)}{\int_{\mathcal{M}} \mathbf{1}_{\mathcal{A}}(z) d\mathcal{M}_z}\,.$$

This density is taken with respect to $d\mathcal{M}_z$, the Riemannian measure but using Eq. (3.4) and a coordinate system z allows to obtain a pdf defined with respect to the Lebesgue one. Moreover, since the volume of the whole manifold $\mathcal{M} = (\mathbb{R}^d, \mathbf{G})$ is finite, we can now define a *uniform distribution* on $\mathcal{M}$

$$\mathcal{U}_{\text{Riem}}(z) = \frac{\sqrt{\det \mathbf{G}(z)}}{\int_{\mathbb{R}^d} \sqrt{\det \mathbf{G}(z) dz}}\,.$$

Since the Riemannian metric has a closed form expression given by Eq. (3.6) sampling from this distribution is quite easy and may be performed using the HMC sampler (Neal, 2005) for instance. Now we are able to sample from the intrinsic uniform distribution which is a natural way of exploring the estimated manifold and the sampling is guided by the geometry of the latent space. A discussion on practical outcomes can be found in Appendix 3.7.2. Noteworthy is the fact that this approach can also be easily applied to more recent VAE models having a Gaussian posterior (*e.g.* (Burda et al., 2016; Larsen et al., 2016; Tomczak and Welling, 2018; Makhzani et al., 2015)). We detail this and show that the proposed method can also benefit these models in Appendix 3.7.7.

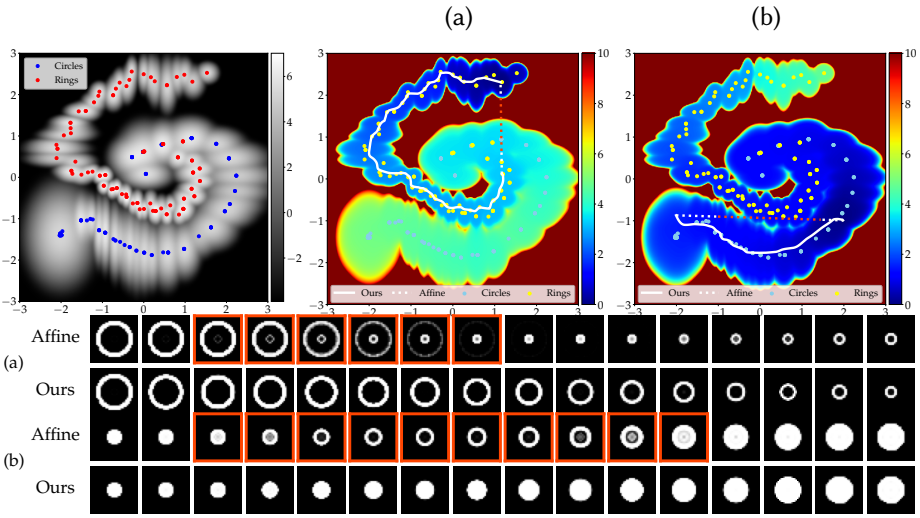


Figure 3.1: *Top left:* Visualization and interpolation in a 2D latent space learned by a VAE trained with binary images of rings and disks. The log of the metric volume element $\sqrt{\det \mathbf{G}(z)}$ (proportional to the log of the density we propose to sample from) is shown in gray scale. *Top middle and right*: Riemannian distance from a starting point (color maps). The dashed lines are affine interpolations between two points in the latent space and the solid ones are obtained by solving Eq. (3.8). *Bottom:* Decoded samples along the interpolation curves.

3.4.6 Illustration on a Toy Dataset

The usefulness of such sampling procedure can be observed in Figure 3.1 where a vanilla VAE was trained with a toy dataset composed of binary images of disks and rings of different size and thickness (example inspired by (Chadebec et al., 2022b)). On the left is presented the learned latent space along with the embedded training points given by the colored dots. The log of the metric volume element is given in gray scale. In this example, we clearly see a geometrical structure appearing since the disks and rings seem to wrap around each other. Obviously, sampling using the prior (taken as a $\mathcal{N}(0, I_d)$) in such a case is far from being optimal since the sampling will be performed regardless of the underlying distribution of the latent variables and so will create irrelevant samples. To further illustrate this, we propose to interpolate between points in the latent space using different cost functions. Dashed lines represent affine interpolations while the solid ones show interpolations aiming at minimizing the potential $V(z) = (\sqrt{\det \mathbf{G}(z)})^{-1}$ all along the curve *i.e.* solving the minimization problem

$$\inf_{\gamma} \int_0^1 V(\gamma(t)) \|\dot{\gamma}(t)\| dt \quad \text{s.t.} \quad \gamma(0) = z_1, \ \gamma(1) = z_2 \,. \tag{3.8}$$

In Figure 3.1 are presented the decoded samples all along the interpolation curves. Thanks to those interpolations we can see that i) the latent space seems to really have a specific geometrical structure since decoding all along the interpolation curves obtained by solving Eq. (3.8) leads to qualitatively satisfying results, ii) certain locations of the latent space must be avoided since sampling there will produce irrelevant samples (see red frames and corresponding red dashes). Using the proposed sampling scheme will allow to sample in the light-colored areas and so ensure that the sampling

remains close to the data *i.e.* where information is available and so does not produce irrelevant images when decoded while still proposing relevant variations from the input data.

3.5 Experiments

In this section, we conduct a comparison with other VAE models using other regularization schemes, more complex priors, richer posteriors, ex-post density estimation or trying to take into account geometrical aspects. In the following, all the models share the same auto-encoding neural network architectures and we used the code and hyper-parameters provided by the authors if available[2]. See Appendix 3.7.4 for models descriptions and the comprehensive experimental set-up.

3.5.1 Generation with Benchmark Datasets

First, we compare the proposed sampling method to several VAE variants such as a Wasserstein Autoencoder (WAE) (Tolstikhin et al., 2018), Regularized Autoencoders (RAEs) (Ghosh et al., 2020), a vamp-prior VAE (VAMP) (Tomczak and Welling, 2018), a Hamiltonian VAE (HVAE) (Caterini et al., 2018), a geometry-aware VAE (RHVAE) (Chadebec et al., 2020) and an Autoencoder (AE). We elect these models since they use different ways to generate the data using either the prior or ex-post density estimation. For the latter, we fit a 10-component mixture of Gaussian in the latent space after training like (Ghosh et al., 2020) .

Figure 3.2 shows a qualitative comparison between the resulting generated samples for MNIST (LeCun, 1998) and CELEBA (Liu et al., 2015), see Appendix 3.7.3 for SVHN (Netzer et al., 2011) and CIFAR 10 (Krizhevsky et al., 2009). Interestingly, using the non-prior based methods seems to produce qualitatively better samples (rows 7 to end). Nonetheless, the resulting samples seem even sharper when the sampling takes into account geometrical aspects of the latent space as we propose (last row). Additionally, even though the exact same model is used, we clearly see that using the proposed method represents a strong improvement of the generation process from a vanilla VAE when compared to the samples coming from a normal prior (second row). This confirms that even the simplest VAE model actually contains a lot of information in its latent space but the limited expressiveness of the prior impedes to access to it. Hence, using more complex priors such as the VAMP may be a tempting idea. However, one must keep in mind that the ELBO objective in Eq. (3.1) must remain tractable and so using more expressive priors may be impossible.

These observations are even more supported by Table 3.1 where we report the Frechet Inception Distance (FID) and the precision and recall (PRD) score against the test set to assess the sampling quality and diversity. Again, fitting a mixture of Gaussian (GMM) in the latent space appears to be an interesting idea since it allows for a better expressiveness and latent space prospecting. For instance, on MNIST the FID falls from 40.7 with the prior to 13.1 when using a GMM. Nonetheless, with the proposed method we are able to make it even smaller (8.5) and PRD scores higher without changing the model and performing post processing. This can also be observed on the 3 other datasets. Impressively, in almost all cases, the proposed generation method can either compete or outperform peers both in terms of FID and PRD scores.

[2]We also perform a wider hyper-parameter search on MNIST and CELEBA for each model in Appendix 3.7.3

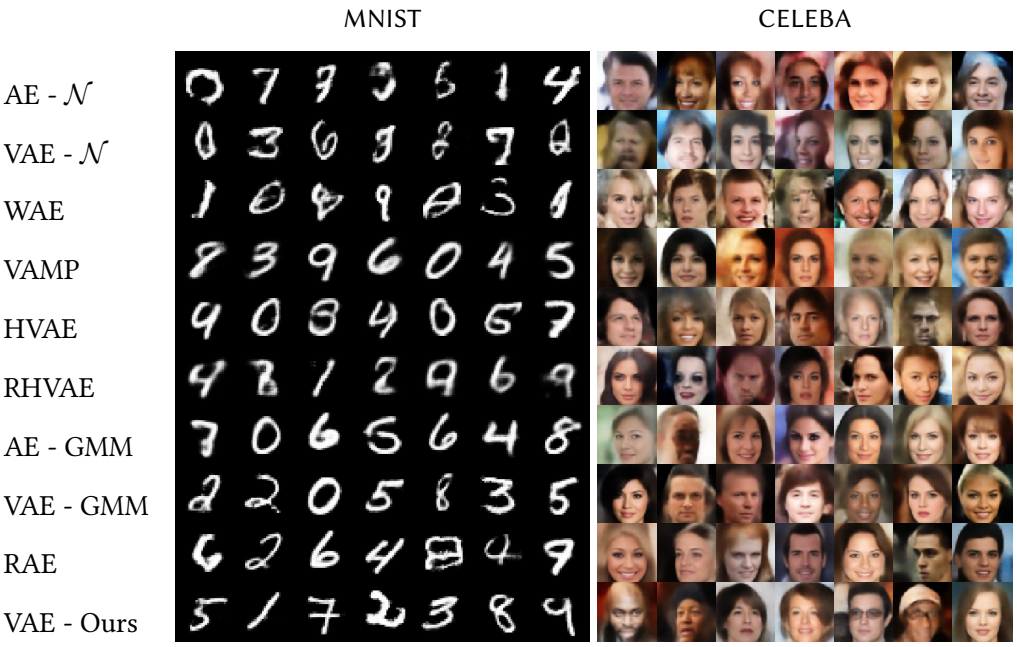


Figure 3.2: Generated samples with different models and generation methods. Results with RAE variants are provided in Appendix 3.7.3.

Finally, we check if the proposed method does not overfit the training data and is able to produce diverse samples by showing the nearest neighbor in the train set and the nearest image in all the reconstructions of the train images to a generated image in Figure 3.3 (left). We also provide the FID score between 10k generated samples and 10k train reconstructions in Figure 3.3 (right). These experiments show that the generated samples are not only resampled train images and that the sampling prospects quite well the manifold. To support even more this claim we provide in Appendix 3.7.6 an analysis in a case where only two centroids are selected in the metric. This also shows that the generated samples are not only an interpolation between the k selected centroids since some generated images contain attributes that are not present in the images of the decoded centroids.

The outcome of such an experiment is that using post training latent space processing such as ex-post density estimation or adding some geometrical consideration to the model allows to strongly improve the sampling without adding more complexity to the model. Generating 1k samples on CELEBA takes approx. 5.5 min for our method vs. 4 min for a 10-component GMM on a GPU V100-16GB.

3.5.2 Investigating Robustness in Low Data Regime

We perform a comparison using the same models and datasets as before but we progressively decrease the size of the training set to see the robustness of the methods according to the number of samples. Despite rarely performed in most generative models related papers, this setup appeared to us relevant since 1) it is well known that it may be challenging for these models, 2) in day-to-day applications collecting large databases may reveal costly if not impossible (*e.g.* medicine). Hence,

reconstruction vs. generation		
	MNIST	CELEBA
FID	11.27	30.12

Figure 3.3: *Left:* Nearest train image (near. train) and nearest image in all reconstructions of train images (near. rec.) to the generated one (Gen.) with the proposed method. Note: the nearest reconstruction may be different from the reconstruction of the nearest train image. *Right*: The FID score between 10k generated images and 10k reconstructed train samples.

Table 3.1: FID (lower is better) and PRD score (higher is better) for different models and datasets. For the mixture of Gaussian (GMM), we fit a 10-component mixture of Gaussian in the latent space.

Model	MNIST (16)		SVHN (16)		CIFAR 10 (32)		CELEBA (64)	
	FID ↓	PRD ↑	FID ↓	PRD ↑	FID ↓	PRD ↑	FID ↓	PRD ↑
AE - $\mathcal{N}(0,1)$	46.41	0.86/0.77	119.65	0.54/0.37	196.50	0.05/0.17	64.64	0.29/0.42
WAE	20.71	0.93/0.88	49.07	0.80/**0.85**	132.99	0.24/0.52	54.56	**0.57**/0.55
VAE - $\mathcal{N}(0,1)$	40.70	0.83/0.75	83.55	0.69/0.55	162.58	0.10/0.32	64.13	0.27/0.39
VAMP	34.02	0.83/0.88	91.98	0.55/0.63	198.14	0.05/0.11	73.87	0.09/0.10
HVAE	15.54	0.97/0.95	98.05	0.64/0.68	201.70	0.13/0.21	52.00	0.38/0.58
RHVAE	36.51	0.73/0.28	121.69	0.55/0.41	167.41	0.12/0.22	55.12	0.45/0.56
AE - GMM	9.60	0.95/0.90	54.21	0.82/0.83	130.28	0.35/0.58	56.07	0.32/0.48
RAE (GP)	9.44	0.97/**0.98**	61.43	0.79/0.78	120.32	0.34/0.58	59.41	0.28/0.49
RAE (L2)	9.89	0.97/**0.98**	58.32	0.82/0.79	123.25	0.33/0.54	54.45	0.35/0.55
RAE (SN)	11.22	0.97/**0.98**	95.64	0.53/0.63	114.59	0.32/0.53	55.04	0.36/0.56
RAE	11.23	**0.98**/**0.98**	66.20	0.76/0.80	118.25	0.35/0.57	53.29	0.36/0.58
VAE - GMM	13.13	0.95/0.92	52.32	0.82/**0.85**	138.25	0.29/0.53	55.50	0.37/0.49
VAE - Ours	**8.53**	**0.98**/0.97	**46.99**	**0.84**/**0.85**	**93.53**	**0.71**/**0.68**	**48.71**	0.44/**0.62**

we consider MNIST, CIFAR10 and SVHN and use either the full dataset size, 10k, 5k or 1k training samples. For each experiment, the best retained model is again the one achieving the best ELBO on the validation set the size of which is set as 20% of the train size. See Appendix 3.7.4 for further details about experiments set-up. Then, we report the evolution of the FID against the test set in Figure 3.4. Results obtained on SVHN are presented in Appendix 3.7.5. Again, the proposed sampling method appears quite robust to the dataset size since it outperforms the other models' FID even when the number of training samples is smaller. This is made possible thanks to the proposed metric that allows to avoid regions of the latent space having poor information. Finally, our study shows that although using more complex generation procedures such as ex-post density estimation seems to still enhance the generation capability of the model when the number of training samples remains quite high ($\geq$5k), this gain seems to worsen when the dataset size reduces as illustrated on CIFAR. In addition, we also evaluate the model on a data augmentation task with neuroimaging data from OASIS (Marcus et al., 2007) mimicking a day-to-day scenario where the limited data regime is very common in Appendix 3.7.3.

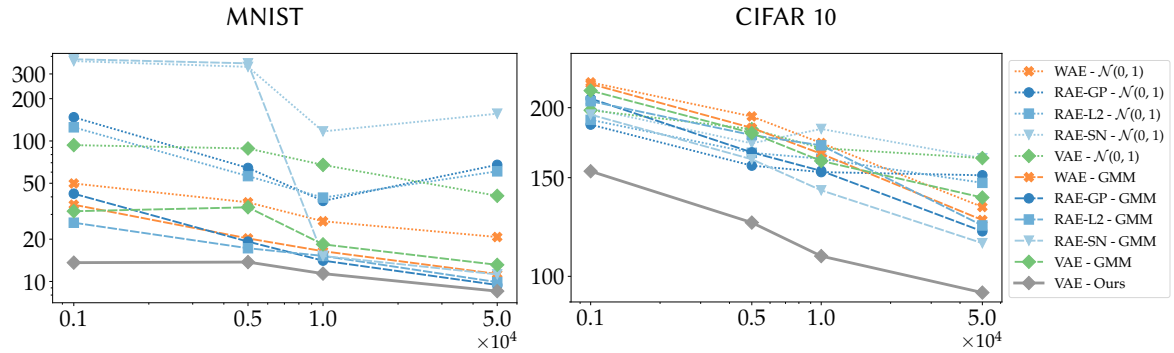


Figure 3.4: Evolution of the FID score according to the number of training samples.

3.6 Conclusion

In this chapter, we provided a geometric understanding of the latent space learned by a VAE and showed that it can actually be seen as a Riemannian manifold. We proposed a new natural generation process consisting in sampling from the intrinsic uniform distribution defined on this learned manifold. The proposed method was empirically shown to be competitive with more advanced versions of the VAEs using either more complex priors, ex-post density estimation, normalizing flows or other regularization schemes. Interestingly, the proposed method revealed good robustness properties in complex settings such as high dimensional data or low sample sizes and appeared to benefit more recent VAE models as well. Future work would consist in trying to use this method to perform data augmentation in those challenging contexts and compare its reliability for such a task with state of the art methods or trying to use this metric to perform clustering in the latent space.

Acknowledgment

The research leading to these results has received funding from the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and reference ANR-10-IAIHU-06 (Agence Nationale de la Recherche-10-IA Institut Hospitalo-Universitaire-6). This work was granted access to the HPC resources of IDRIS under the allocation AD011013517 made by GENCI (Grand Equipement National de Calcul Intensif).

3.7 Appendices

3.7.1 Further Elements on Riemannian Geometry

A d-dimensional Riemannian manifold $\mathcal{M}$ can be defined as a d-dimensional differentiable manifold equipped with is a smooth inner product $g : z \to \langle \cdot | \cdot \rangle_z$ defined on each tangent space $T_z\mathcal{M}$ of the manifold with $z \in \mathcal{M}$. A chart (or coordinate system) (U, ϕ) is a homeomorphism mapping an open set U of the manifold to an open set V of an Euclidean space. Given $z \in U$, a chart $\phi : (z^1, \dots, z^d)$ induces a basis $\left(\frac{\partial}{\partial z^1}, \dots, \frac{\partial}{\partial z^d}\right)_z$ on the tangent space $T_z\mathcal{M}$. Hence, the metric of a Riemannian manifold can be locally represented in the chart ϕ as a positive definite matrix as mentioned in Sec. 3.4.1.

$$\mathbf{G}(z) = (g_{i,j})_{z, 0 \leq i,j \leq d} = \left(\left\langle \frac{\partial}{\partial z^i} | \frac{\partial}{\partial z^j} \right\rangle_z \right)_{0 \leq i,j \leq d},$$

for each point z of the manifold. That is for $v, w \in T_z\mathcal{M}$ and $z \in \mathcal{M}$, the inner product writes $\langle u | w \rangle_z = u^\top \mathbf{G}(z) w$. Assuming that the manifold is also connected, for any $z_1, z_2 \in \mathcal{M}$, two points of the manifold, we can consider a curve γ traveling in $\mathcal{M}$ and parametrized by $t \in [a, b]$ such that $\gamma(a) = z_1$ and $\gamma(b) = z_2$. Then, the length of γ is given by

$$L(\gamma) = \int_a^b \|\dot{\gamma}(t)\|_{\gamma(t)} dt = \int_a^b \sqrt{\langle \dot{\gamma}(t) | \dot{\gamma}(t) \rangle_{\gamma(t)}} dt$$

Curves γ that minimize L and are parameterized proportionally to the arc length are called *geodesic* curves. A distance $\mathrm{dist}_{\mathbf{G}}$ on the manifold $\mathcal{M}$ can then be derived and writes

$$\mathrm{dist}_{\mathbf{G}}(z_1, z_2) = \inf_\gamma L(\gamma) \quad \text{s.t.} \quad \gamma(a) = z_1, \gamma(b) = z_2$$

The manifold $\mathcal{M}$ is said to be *geodesically complete* if all geodesic curves can be extended to $\mathbb{R}$. Given the Riemannian manifold $\mathcal{M}$ endowed with the Riemannian metric $\mathbf{G}$ and a chart z, an infinitesimal volume element may also be defined on each tangent space T_z of the manifold $\mathcal{M}$ as follows

$$d\mathcal{M}_z = \sqrt{\det \mathbf{G}(z)} dz \,,$$

with dz being the Lebesgue measure. This defines a canonical measure on the manifold and allows to extend the notion of probability distributions to Riemannian manifolds. In particular, such a property allows to refer to random variables with a density defined with respect to the measure on the manifold. We recall such definition from (Pennec, 2006) below

Definition 2 *Let $\mathcal{B}(\mathcal{M})$ be the Borel σ-algebra of $\mathcal{M}$. The random point $\mathbf{z}$ has a probability density function $\rho_{\mathbf{z}}$ if:*

$$\forall \mathcal{Z} \in \mathcal{B}(\mathcal{M}),\ \ \mathbb{P}(\mathbf{z} \in \mathcal{Z}) = \int_{\mathcal{Z}} \rho(z) d\mathcal{M}(z) \ \ \textit{and} \ \int_{\mathcal{M}} \rho(z) d\mathcal{M}(z) = 1$$

Finally, given a chart ϕ defined on the whole manifold $\mathcal{M}$ and a random point $\mathbf{z}$ on $\mathcal{M}$, the point $\mathbf{p} = \phi(\mathbf{z})$ is a random point whose density $\rho'_{\mathbf{p}}$ may be written with respect to the Lebesgue measure as such (Pennec, 2006):

$$\rho'_{\mathbf{p}}(p) = \rho_{\mathbf{z}}(\phi^{-1}(p)) \sqrt{\det g(\phi^{-1}(p))}$$

3.7.2 The Generation Process Algorithm - Implementation Details

In this appendix, we provide pseudo-code algorithms explaining how to build the metric from a trained VAE and how to use the proposed sampling process. Noteworthy is the fact that we do not amend the training process of the vanilla VAE which remains pretty simple and stable.

Building the metric

In this section, we explain how to build the proposed Riemannian metric. For the sake of clarity, we recall the expression of the metric below

$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}^{-1}(x_i) \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d \,, \tag{3.9}$$

where

$$\omega_i(z) = \exp\left(- \frac{\mathrm{dist}_{\mathbf{\Sigma}^{-1}(x_i)}(z, \mu(x_i))^2}{\rho^2} \right) = \exp\left(- \frac{(z-\mu(x_i))^\top \mathbf{\Sigma}^{-1}(x_i)(z-\mu(x_i))}{\rho^2} \right).$$

Algorithm 4 Building the metric from a trained model

Input: A trained VAE model m, the training dataset $\mathcal{X}$, λ, τ ▷ In practice $\tau \approx 0$

for $x_i \in \mathcal{X}$ **do**

$\mu_i, \mathbf{\Sigma}_i = m(x_i)$ ▷ Retrieve training embeddings and covariance matrices

end for

Select k centroids c_i in the μ_i ▷ e.g. with k-medoids

Get corresponding covariance matrices $\mathbf{\Sigma}_i$

$\rho \leftarrow \max_i \min_{j \neq i} \|c_i - c_j\|_2$ ▷ Set ρ to the max distance between two closest neighbors

Build the metric using Eq. (3.9)

$$\mathbf{G}(z) = \sum_{i=1}^{N} \mathbf{\Sigma}_i^{-1} \cdot \omega_i(z) + \lambda \cdot e^{-\tau \|z\|_2^2} \cdot I_d$$

Return G ▷ Return $\mathbf{G}$ as a function

As is standard in VAE implementations, we assume that the covariance matrices $\mathbf{\Sigma}_i$ given by the VAE are diagonal and that the encoder outputs a mean vector and the log of the diagonal coefficients. In the implementation, the exponential is then applied to recover the $\mathbf{\Sigma}_i$ so that no singular matrix arises.

The HMC sampler

In the sampling process we propose to rely on the Hamiltonian Monte Carlo sampler to sample from the Riemannian uniform distribution. In a nutshell, the HMC sampler aims at sampling from a

target distribution $p_{\text{target}}(z)$ with $z \in \mathbb{R}^d$ using Hamiltonian dynamics. The main idea behind such a sampler is to introduce an auxiliary random variable $v \sim \mathcal{N}(0, I_d)$ independent from z and mimic the behavior of a particle having z (resp. v) as location (resp. velocity). The Hamiltonian of the particle then writes

$$H(z, v) = U(z) + K(v)\,,$$

where $U(z)$ is the potential energy and $K(v)$ is its kinetic energy both given by

$$U(z) = -\log p_{\text{target}}(z), \qquad K(v) = \frac{1}{2} v^\top v\,.$$

The following Hamilton's equations govern the evolution in time of the particle.

$$\begin{cases} \frac{\partial H(z,v)}{\partial v} = & v\,, \\ \frac{\partial H(z,v)}{\partial z} = & -\nabla_z \log p_{\text{target}}(z)\,. \end{cases}$$

In order to integrate these equations, recourse to the leapfrog integrator is needed and consists in applying n_{lf} times the following equations.

$$\begin{cases} v(t + \frac{\varepsilon_{\text{lf}}}{2}) &= v(t) + \frac{\varepsilon_{\text{lf}}}{2} \cdot \nabla_z \log p_{\text{target}}(z(t))\,, \\ z(t + \varepsilon_{\text{lf}}) &= z(t) + \varepsilon_{\text{lf}} \cdot v(t + \frac{\varepsilon_{\text{lf}}}{2})\,, \\ v(t + \varepsilon_{\text{lf}}) &= v(t + \frac{\varepsilon_{\text{lf}}}{2}) + \frac{\varepsilon_{\text{lf}}}{2} \cdot \nabla_z \log p_{\text{target}}(z(t + \varepsilon_{\text{lf}}))\,, \end{cases} \tag{3.10}$$

where ε_{lf} is called the leapfrog step size. This algorithm produces a proposal $(\widetilde{z}, \widetilde{v})$ that is accepted with probability α where

$$\alpha = \min\left(1, \exp\left(H(z, v) - H(\widetilde{z}, \widetilde{v})\right)\right).$$

This procedure is then repeated to create an ergodic Markov chain (z^n) converging to the distribution p_{target} (Duane et al., 1987; Liu, 2008; Neal and others, 2011; Girolami and Calderhead, 2011).

The proposed algorithm

In our setting the target density is given by the density of the Riemannian uniform distribution which writes with respect to Lebesgue measure as follows

$$p(z) = \mathcal{U}_{\text{Riem}}(z) = \frac{1}{C}\sqrt{\det \mathbf{G}(z)} \qquad C = \int_{\mathbb{R}^d} \sqrt{\det \mathbf{G}(z)} dz\,.$$

Note that thanks to the shape of the metric, this distribution is well defined since $C < +\infty$. The log density follows

$$\log p(z) = \frac{1}{2} \log\det \mathbf{G}(z) - \log C\,,$$

Hence, the Hamiltonian writes

$$H(z, v) = -\log p(z) + \frac{1}{2} v^\top v\,,$$

and Hamilton's equations become

$$\begin{cases} \frac{\partial H(z,v)}{\partial v} = & v\,, \\ \frac{\partial H(z,v)}{\partial z_i} = & -\frac{\partial \log p(z)}{\partial z_i} = -\frac{1}{2}\mathrm{tr}\Big(\mathbf{G}^{-1}(z)\frac{\partial \mathbf{G}(z)}{\partial z_i}\Big) \end{cases}$$

Since the covariance matrices are supposed to be diagonal as is standard in VAE implementations, the computation of the inverse metric is straightforward. Moreover, since $\mathbf{G}(z)$ is smooth and has a closed form, it can be differentiated with respect to z pretty easily. Now, the leapfrog integrator given in Eq. (3.10) can be used and the acceptance ratio α is easy to compute. Noteworthy is the fact that the normalizing constant C is never needed since it vanishes in the gradient computation and simplifies in the acceptance ratio α. We provide a pseudo-code of the proposed sampling procedure in Alg. 5. A typical choice in the sampler's hyper-parameters used in the chapter is $N = 100$, $n_{\text{lf}} = 10$ and $\varepsilon_{\text{lf}} = 0.01$. The initialization of the chain can be done either randomly or on points that belong to the manifold (i.e. the centroids c_i or $\mu(x_i)$).

Algorithm 5 Proposed sampling process

Input: The metric function $\mathbf{G}$, hyper-parameters of the HMC sampler (chain length N, number of leapfrog steps n_{lf}, leapfrog step size ε_{lf})
Initialization: z ▷ Initialize the chain
for $i = 1 \to N$ **do**
  $v \sim \mathcal{N}(0, I_d)$ ▷ Draw a velocity
  $H_0 \leftarrow H(z, v)$ ▷ Compute the starting Hamiltonian
  $z_0 \leftarrow z$
  for $k = 1 \leftarrow n_{\text{lf}}$ **do**
    $\bar{v} \leftarrow v - \frac{\varepsilon_{\text{lf}}}{2} \cdot \nabla_z H(z, v)$
    $\widetilde{z} \leftarrow z + \varepsilon_{\text{lf}} \cdot \bar{v}$ ▷ Leapfrog step Eq. (3.10)
    $\widetilde{v} \leftarrow \bar{v} - \frac{\varepsilon_{\text{lf}}}{2} \cdot \nabla_z H(\widetilde{z}, \bar{v})$
    $v \leftarrow \widetilde{v}$
    $z \leftarrow \widetilde{z}$
  end for
  $H \leftarrow H(\widetilde{z}, \widetilde{v})$ ▷ Compute the ending Hamiltonian
  Accept $\widetilde{z}$ with probability $\alpha = \min\Big(1, \exp(H_0 - H)\Big)$
  if Accepted **then**
    $z \leftarrow \widetilde{z}$
  else
    $z \leftarrow z_0$
  end if
end for
Return z

3.7.3 Other Generation Results

Some further samples on CELEBA and MNIST

In this section, we provide some further generated samples using the proposed method. Figure 3.5 and Figure 3.6 again support the fact that the method is able to generate sharp and diverse samples. We also add the other variants of the RAE model in Figure 3.7.

Figure 3.5: 100 samples with the proposed method on MNIST dataset.

Figure 3.6: 100 samples with the proposed method on CELEBA dataset.

Figure 3.7: Generated samples with different models and generation processes.

CIFAR and SVHN

In this appendix, we gather the resulting samplings from the different models considered for SVHN and CIFAR 10.

Figure 3.8: Generated samples with different models and generation processes.

Generation with complex data

Finally, we also propose to stress the proposed generation procedure in a day-to-day scenario where the limited data regime is more than common. To stress the model in such condition, we consider the publicly available OASIS database (Marcus et al., 2007) composed of 416 MRI of patients, 100 of whom were diagnosed with Alzheimer disease (AD). Since both FID and PRD scores are not reliable when no large test set is available, we propose to assess quantitatively the generation quality with a data augmentation task. Hence, we split the dataset into a train (70%), a validation (10%) and a test set (20%). Each model is trained on each label of the train set and used to generate 2k samples per class. Then a CNN classifier is trained on i) the original train set and ii) the 4k generated samples from the generative models and tested on the test set. Table 3.2 shows classification results averaged across 20 runs for each considered model. The line *raw (resampled)* corresponds to a case where the train set is obtained by balancing the classes with simple repetitions of the samples from the under-represented class. These metrics provide a way to assess i) if the model can generate data adding

Figure 3.9: Closest element in the training set (Near.) to the generated one (Gen.) with the proposed method.

relevant information for classification and ii) allows to quantify the amount of overfitting. The proposed method is the only one allowing to achieve higher balanced accuracy and F1 scores for both labels than on the original (unbalanced) data meaning that the samples are relevant to the classifier and this is also sign of a good generalization. Moreover, we provide generated samples using each generation procedure in Figure 3.10. Again, the proposed method appears to produce visually the sharpest samples. However, such augmentation method for medical data requires caution and needs further assessment on the possibly induced biases before being used on a *real-life* application case.

Table 3.2: Classification results averaged on 20 independent runs. For the VAEs, the classifier is trained on 2K generated samples per class.

Generation method	Balanced Accuracy	F1		Precision		Recall	
		AD	CN	AD	CN	AD	CN
Original*	66.2 ± 7.6	47.6 ± 15.8	87.3 ± 2.0	74.7 ± 8.4	80.3 ± 4.0	35.7 ± 16.3	95.7 ± 1.5
Original (resampled)	81.8 ± 2.6	72.1 ± 3.6	$\mathbf{88.0 \pm 2.3}$	67.0 ± 5.3	91.4 ± 1.8	78.5 ± 5.2	85.1 ± 4.2
AE - $\mathcal{N}$	50.0 ± 0.0	0.0 ± 0.0	84.1 ± 0.0	0.0 ± 0.0	72.6 ± 0.0	0.0 ± 0.0	100.0 ± 0.0
WAE	57.4 ± 9.7	21.0 ± 24.5	84.4 ± 2.3	48.5 ± 42.8	76.7 ± 6.1	19.3 ± 27.5	95.4 ± 9.3
VAE - $\mathcal{N}$	51.8 ± 3.8	6.1 ± 11.8	84.6 ± 1.1	38.0 ± 47.3	73.4 ± 1.7	3.7 ± 7.8	99.8 ± 0.7
VAMP	83.1 ± 2.6	70.4 ± 3.6	82.2 ± 4.7	56.3 ± 5.2	97.5 ± 2.1	94.8 ± 4.7	71.5 ± 7.4
HVAE	56.3 ± 7.9	19.6 ± 21.7	85.4 ± 1.7	48.7 ± 41.7	75.5 ± 3.8	13.9 ± 17.6	98.6 ± 2.2
RHVAE	68.0 ± 10.9	47.0 ± 24.2	85.1 ± 3.3	56.1 ± 25.3	83.0 ± 7.5	46.7 ± 30.2	89.2 ± 10.6
AE - GMM	82.4 ± 2.3	69.5 ± 3.1	82.0 ± 3.6	55.8 ± 4.9	96.8 ± 2.4	93.3 ± 5.6	71.5 ± 6.2
RAE (GP)	63.9 ± 9.8	46.5 ± 15.9	70.6 ± 19.6	45.3 ± 18.5	84.2 ± 8.6	60.9 ± 28.6	67.0 ± 24.9
RAE (L2)	74.1 ± 6.0	60.6 ± 9.5	82.1 ± 5.9	57.8 ± 10.1	88.3 ± 5.2	70.0 ± 18.7	78.3 ± 11.7
RAE (SN)	62.3 ± 8.9	37.8 ± 22.6	80.1 ± 7.9	43.1 ± 24.9	80.6 ± 6.6	41.7 ± 30.1	82.9 ± 16.4
RAE	69.3 ± 8.1	53.8 ± 12.9	80.0 ± 10.7	56.2 ± 13.5	85.2 ± 6.2	60.0 ± 24.0	78.5 ± 17.5
VAE - GMM	83.0 ± 3.6	71.4 ± 4.3	85.3 ± 3.0	60.7 ± 5.4	94.9 ± 3.7	88.0 ± 9.5	77.9 ± 5.9
VAE - Ours	$\mathbf{85.4 \pm 2.5}$	$\mathbf{74.7 \pm 3.5}$	87.3 ± 2.7	64.0 ± 5.3	95.8 ± 2.2	90.4 ± 5.6	80.3 ± 5.1

*unbalanced

Wider hyper-parameter search

As stated in the main paper, for the experiments, we used the official implementation and hyper-parameters provided by the authors when available. However, we also propose to perform a hyper-parameter search for the models considered in the benchmark *i.e.* WAE, VAMP-VAE, RAE-GP and RAE-L2 [3] on MNIST and CELEBA. Since both HVAE and RHVAE models have a very time consuming training, we propose to replace these approaches with models having the same objective

(i.e. enriching the posterior distribution). Do to so we consider a VAE with inverse autoregressive flows (Kingma et al., 2016) (VAE-IAF) and a VAE with normalizing flows with radial/planar invertible transformations (Rezende and Mohamed, 2015) (VAE-NF).

We train these models with 10 different hyper-parameter configurations on MNIST and CELEBA. For the WAE, we vary the kernel bandwidth in {0.01, 0.1, 0.5, 1, 2, 5} and change the regularization factor weighting the reconstruction and regularization in {0.01, 0.1, 1, 10, 100}. For the RAEs, we vary the L2 latent code regularization factor and the factor before the explicit regularization in $\{1e^{-6}, 1e^{-4}, 1e^{-3}, 0.01, 0.1, 1\}$. For the VAMP we vary the number of pseudo-inputs in {10, 20, 30, 50, 100, 150, 200, 250, 300,189 500}. Finally, for the flow-based VAEs we vary the complexity of the flows with different number of IAF blocks (VAE-IAF) or different flow lengths (VAE-NF). To assess the influence of the neural architecture, the experiment is performed twice each time with a different neural network architecture (CNN in Table 3.4 or a simpler ResNet). In Table 3.3, we show the generation vs. test FID of the model achieving the lowest FID on the validation set.

Table 3.3: FID (lower is better) for different models and datasets. For the mixture of Gaussian (GMM), we fit a 10-component mixture of Gaussian in the latent space.

Models	MNIST		CELEBA	
Nets	CNN	ResNet	CNN	ResNet
AE - N(0,1)	46.4	221.8	64.6	275.0
WAE	18.9	20.3	54.6	67.1
VAE - N(0,1)	40.7	47.8	64.1	69.5
VAMP	34.0	34.5	56.0	67.2
VAE-NF	29.3	32.5	55.4	67.1
VAE-IAF	27.5	30.6	56.5	66.2
AE - GMM	9.6	11.0	56.1	57.4
RAE-GP	9.4	11.4	52.5	59.0
RAE-L2	9.1	11.5	54.5	58.3
VAE - GMM	13.1	12.4	55.5	59.9
Ours	**8.5**	**10.7**	**48.7**	**53.2**

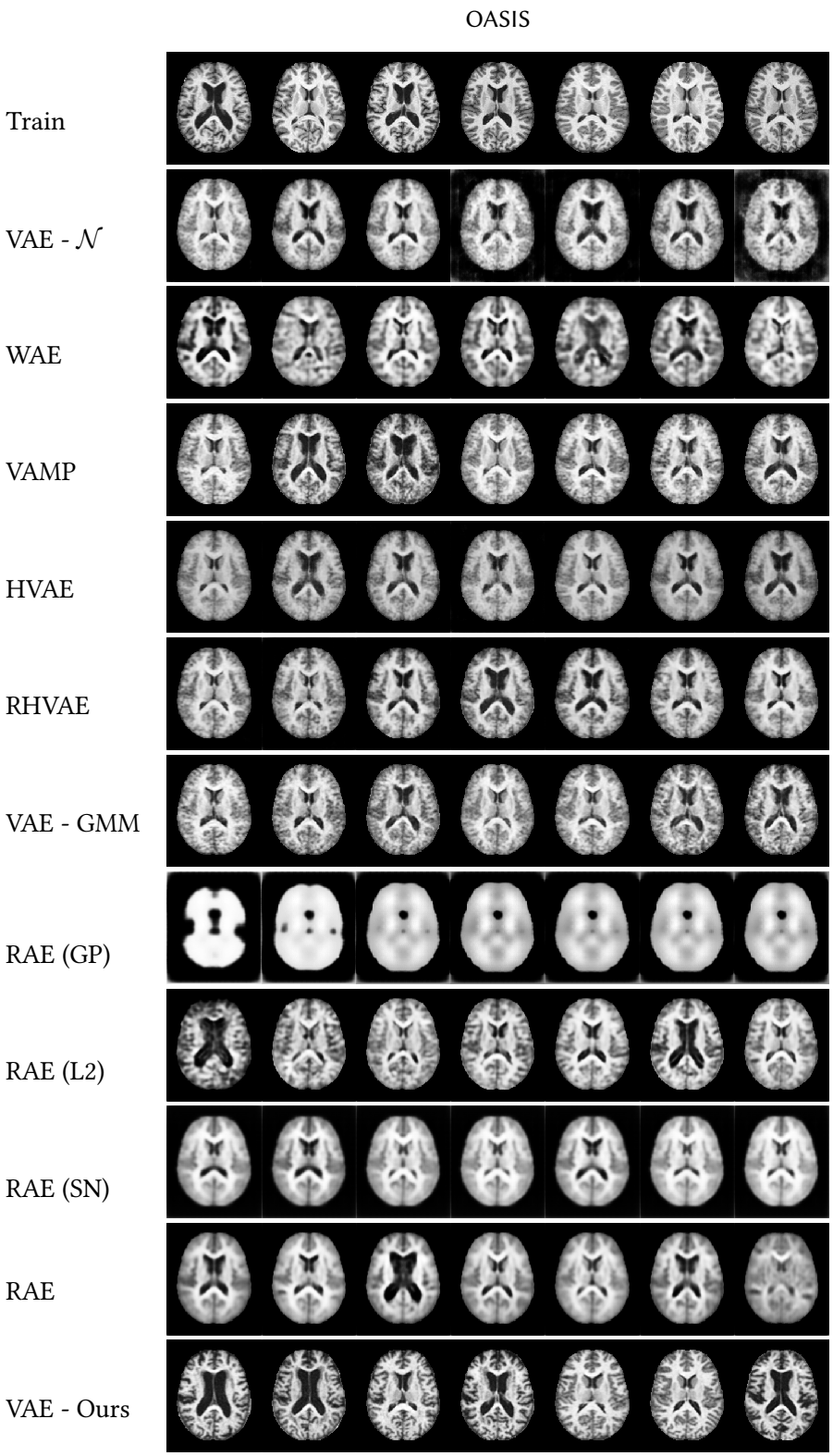


Figure 3.10: Generated samples with different models and generation processes.

3.7.4 Experimental Set-Up

We compare the proposed sampling method to several VAE variants such as a Wasserstein Autoencoder (WAE) (Tolstikhin et al., 2018), Regularized Autoencoders (Ghosh et al., 2020) with either L2 decoder's parameters regularization (RAE-L2), gradient penalty (RAE-GP), spectral normalization (RAE-SN) or simple L2 latent code regularization (RAE), a vamp-prior VAE (VAMP) (Tomczak and Welling, 2018), a Hamiltonian VAE (HVAE) (Caterini et al., 2018), a geometry-aware VAE (RHVAE) (Chadebec et al., 2020) and an Autoencoder (AE). The RAEs, VAEs and AEs are trained for 100 epochs for SVHN, MNIST[3] and CELEBA and 200 on CIFAR10. Each time we use the official train and test split of the data. For MNIST and SVHN, 10k samples out of the train set are reserved for validation and 40k for CIFAR10. As to CELEBA, we use the official validation set for validation. The model that is kept at the end of training is the one achieving the best validation loss. All the models are trained with a batch size of 100 and starting learning rate of $1e{-}3$ (but CIFAR where the learning rate is set to $5e{-}4$) with an Adam optimizer (Kingma and Ba, 2014). We also use a scheduler decreasing the learning rate by half if the validation loss stops increasing for 5 epochs. For the experiments on the sensitivity to the training set size, we keep the same set-up. For each dataset we ensure that the validation set is $1/5^{\text{th}}$ the size of the train set but for CIFAR where we select the best model on the train set. The neural networks architectures can be found in Table 3.4 and are inspired by (Ghosh et al., 2020). The metrics (FID and PRD scores) are computed with 10000 samples against the test set (for CELEBA we selected only the 10000 first samples of the official test set). The factor ρ is set to $\rho = \max_i \min_{j \neq i} \|c_i - c_j\|_2$ to ensure some *smoothness* of the manifold. For models coming from peers, we use the parameters and code provided by the authors when available and allowed by licenses.

For the data augmentation task, the generative models are trained on each class for 1000 epochs with a batch size of 100 and a starting learning rate of $1e{-}4$. Again a scheduler is used and the learning rate is cut by half if the loss does not improve for 20 epochs. All the models have the autoencoding architecture described in Table 3.4. As to the classifier, it is trained with a batch size of 200 for 50 epochs with a starting learning rate of $1e{-}4$ and Adam optimizer. A scheduler reducing the learning rate by half every 5 epochs if the validation loss does not improve is again used. The best kept model is the one achieving the best balanced accuracy on the validation set. Its neural network architecture may be found in Table 3.5. MRIs are only pre-processed such that the maximum value of a voxel is 1 and the minimum 0 for each data point.

[3] MNIST images are re-scaled to 32x32 images with a 0 padding.

Table 3.4: Neural networks used for the encoder and decoders of VAEs in the benchmarks

	MNIST [CIFAR10]	SVHN	CELEBA	OASIS
Encoder	(1[3], 32, 32)	(3, 32, 32)	(3, 64, 64)	(1, 208, 176)
Layer 1	Conv(128, (4, 4), stride=2) Batch normalization ReLU	Linear(1000) ReLU	Conv(128, (5, 5), stride=2) Batch normalization ReLU	Conv(64, (5, 5), stride=2) ReLU
Layer 2	Conv(256, (4, 4), stride=2) Batch normalization ReLU	Linear(500) ReLU	Conv(256, (5, 5), stride=2) Batch normalization ReLU	Conv(128, (5, 5), stride=2) ReLU
Layer 3	Conv(512, (4, 4), stride=2) Batch normalization ReLU	Linear(500, 16)	Conv(512, (5, 5), stride=2) Batch normalization ReLU	Conv(256, (5, 5), stride=2) ReLU
Layer 4	Conv(1024, (4, 4), stride=2) Batch normalization ReLU	-	Conv(1024, (5, 5), stride=2) Batch normalization ReLU	Conv(512, (5, 5), stride=2) ReLU
Layer 5	Linear(4096, 16)	-	Linear(16384, 64)	Conv(1024, (5, 5), stride=2) ReLU
Layer 6	-	-	-	Linear(4096, 16)
Decoder	(16 [32])	(16)	(64)	(16)
Layer 1	Linear(65536) Reshape(1024, 8, 8)	Linear(500) ReLU	Linear(65536) Reshape(1024, 8, 8)	Linear(65536) Reshape(1024, 8, 8)
Layer 2	ConvT(512, (4, 4), stride=2) Batch normalization ReLU	Linear (1000) ReLU	ConvT(512, (5, 5), stride=2) Batch normalization ReLU	ConvT(512, (5, 5), stride=(3, 2)) ReLU
Layer 3	ConvT(256, (4, 4), stride=2) Batch normalization ReLU	Linear(3072) Reshape(3, 32, 32) Sigmoid	ConvT(256, (5, 5), stride=2) Batch normalization ReLU	ConvT(256, (5, 5), stride=2) ReLU
Layer 4	ConvT(3, (4, 4), stride=1) Batch normalization Sigmoid	-	ConvT(128, (5, 5), stride=2) Batch normalization ReLU	ConvT(128, (5, 5), stride=2) ReLU
Layer 5	-	-	ConvT(3, (5, 5), stride=1) Batch normalization Sigmoid	ConvT(64, (5, 5), stride=2) ReLU
Layer 6	-	-	-	ConvT(1, (5, 5), stride=1) ReLU

Table 3.5: Neural Network used for the classifier in Sec. 3.7.3

	OASIS Classifier
Input Shape	$(1, 208, 176)$
Layer 1	Conv(8, (3, 3), stride=1) Batch normalization LeakyReLU MaxPool(2, stride=2)
Layer 2	Conv(16, (3, 3), stride=1) Batch normalization LeakyReLU MaxPool(2, stride=2)
Layer 3	Conv(32, (3, 3), stride=2) Batch normalization LeakyReLU MaxPool(2, stride=2)
Layer 4	Conv(64, (3, 3), stride=2) Batch normalization LeakyReLU MaxPool(2, stride=2)
Layer 5	Linear$(256, 100)$ ReLU
Layer 6	Linear$(100, 2)$ Softmax

3.7.5 Dataset Size Sensibility on SVHN

In Figure 3.11, we show the same plot for SVHN as in Sec. 3.5.2. Again the proposed method appears to be part of the most robust generation procedures to dataset size changes.

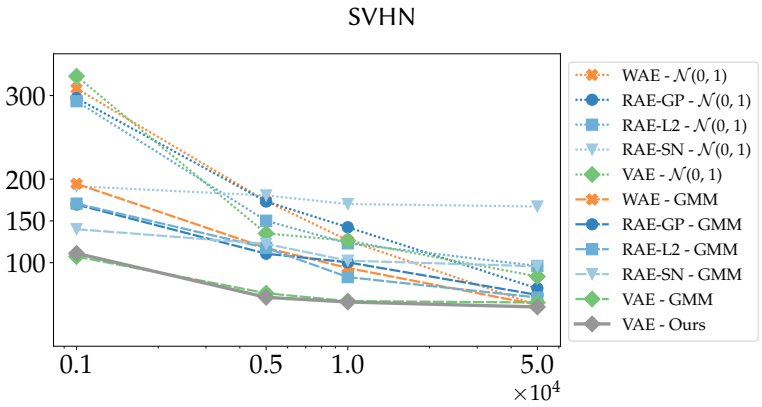


Figure 3.11: FID score evolution according to the number of training samples.

3.7.6 Ablation Study

Influence of the number of centroids in the metric

In order to assess the influence of the number of centroids and their choice in the metric in Eq. (3.6), we show in Figure 3.12 the evolution of the FID according to the number of centroids in the metric (left) and the variation of FID according to the choice in the centroids (right). As expected, choosing a small number of centroids will increase the value of the FID since it reduces the variability of the generated samples that will remain *close* to the centroids. Nonetheless, as soon as the number of centroids is higher than 1000 the FID score is either competitive or better than peers and continues decreasing as the number of centroids increases.

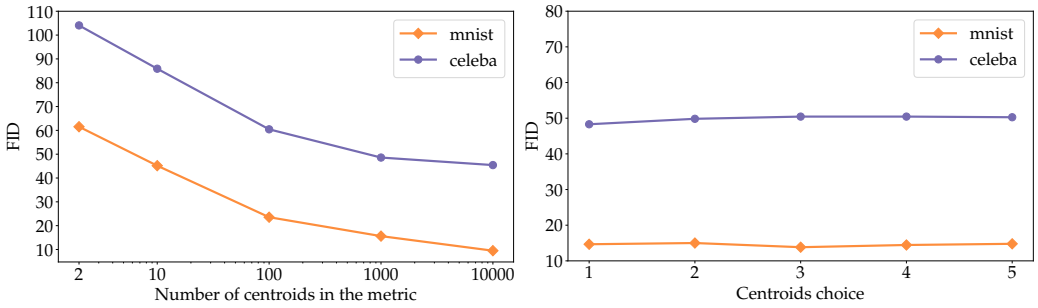


Figure 3.12: *Left:* FID score evolution according to the number of centroids in the metric (Eq. (3.6)). *Right*: The FID variation with respect to the choice in centroids. We generate 10000 samples by selecting each time different centroids ($k = 1000$).

To assess the variability of the generated samples, we propose to analyze some generated samples when only 2 centroids are considered. In Figure 3.13, we display on the left the decoded centroids along with the closest image to these decoded centroids in the train set. On the right are presented some generated samples. We place these samples in the top row if they are closer to the first decoded centroid and in the bottom row otherwise. Interestingly, even with a small number of centroids the proposed sampling scheme is able to access to a relatively good diversity of samples. These samples are not simply resampled train images or a simple interpolation between selected centroids as some of the generated samples have attributes such as glasses that are not present in the images of the decoded centroids.

Influence of λ in the metric

In this section, we also assess the influence of the regularization factor λ in Eq. (3.6) on the resulting sampling. To do so, we generate 10k samples using the proposed method on both MNIST and CELEBA datasets for values of $\lambda \in [1e^{-6}, 1e^{-4}, 1e^{-2}, 1e^{-1}, 1]$. Then, we compute the FID against the test set. Each time, we consider $k = 1000$ centroids in the metric. As shown in Figure 3.14, the influence of λ remains limited. In the implementation, a typical choice for λ is $1e^{-2}$.

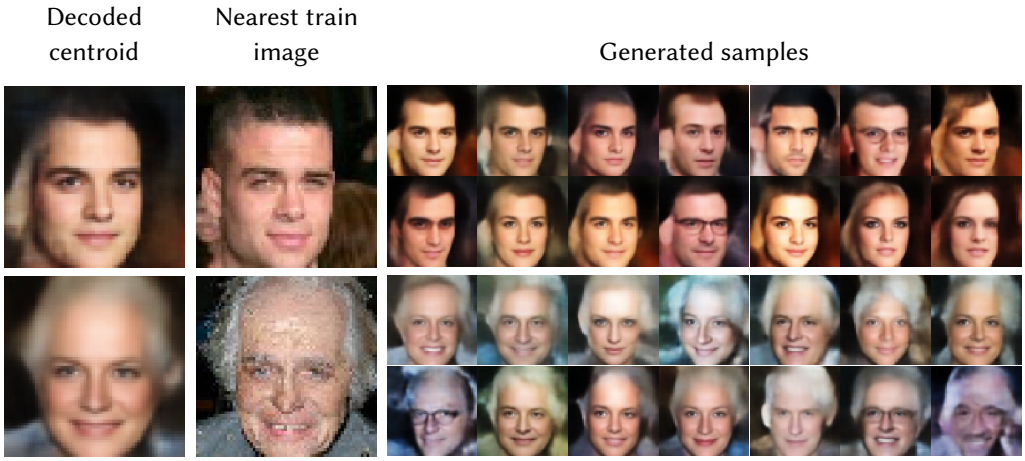


Figure 3.13: Variability of the generated samples when only two centroids are considered in the metric. *Left:* The image obtained by decoding the centroids. *Middle*: The nearest image in the train set to the decoded centroids. *Right:* Some generated samples. Each generated sample is assigned to the closest decoded centroid (top row for the first centroid and bottom row for the second one).

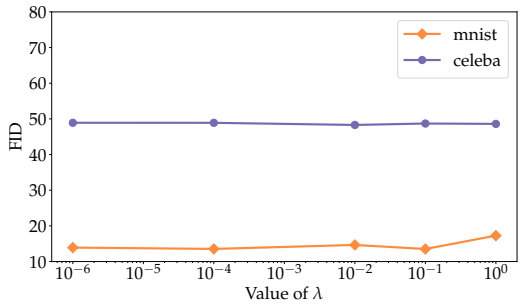


Figure 3.14: FID score evolution according to the value of λ in the metric (Eq. (3.6)).

The choice of ρ

In the experiments presented, the smoothing factor ρ in Eq. (3.6) is set to the value of the maximum distance between two closest centroids $\rho = \max_i \min_{j \neq i} \|c_j - c_i\|_2$. This choice is motivated by the fact that we wanted to build a smooth metric and so ensure some *smoothness* of the manifold while trying to interpolate faithfully between the metric tensors $\mathbf{G}_i = \mathbf{\Sigma}(x_i)^{-1}$. In particular, a too small value of ρ would have allowed disconnected regions and the sampling may have not prospected well the learned manifold and would have only become a resampling of the centroids. On the other hand, setting a high value for ρ would have biased the interpolation and the value of the metric at a $\mu(x_i)$. As a result, $\mathbf{G}(\mu(x_i))$ might have been very different from the one observed $\mathbf{\Sigma}(x_i)^{-1}$ since the other $\mu(x_j)$ would have had a strong influence on its value. The proposed value for ρ appeared to work well in practice.

3.7.7 Can the Method Benefit More Recent Models ?

Our method proposes to build a Riemannian metric using the covariances in the posterior distributions. Thus, it can be easily plugged into more recent models provided that they have a Gaussian posterior distribution. In order to assess how it would benefit to more recent VAE models, we train a VAMP-VAE (Tomczak and Welling, 2018), a VAEGAN (Larsen et al., 2016), an Adversarial AE (Makhzani et al., 2015) and an IWAE (Burda et al., 2016) and compare the generation FID obtained 1) with the prior or 2) when plugging our method. For this experiment, we conduct a hyper-parameter search consisting in training each model with 10 different configurations. For the VAMP we vary the number of pseudo-inputs in {10, 20, 30, 50, 100, 150, 200, 250, 300, 500}. For the VAEGAN, we use a discriminator similar to the encoder described in Table 3.4 and vary the layer depth considered for the reconstruction loss in {2, 3, 4} and the factor balancing reconstruction/generation for the decoder's loss in {0.3, 0.5, 0.7, 0.8, 0.9, 0.99, 0.999}. For the AAE, we change the factor balancing the reconstruction loss and the regularization in {0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99, 0.999}. Finally, for the IWAE, we vary the number of importance samples in {2, 3, 4, 5, 6, 7, 8, 9, 10, 12}. For each model and generation scheme, we report the results of the model achieving the lowest FID on the validation set. According to Table 3.6, the proposed generation method seems to benefit these models in almost all cases since the FID decreases when compared to the prior-based generation.

Table 3.6: FID (lower is better) vs. the test set using either the prior (classic approach) or by plugging our generation method.

Model	Generation	MNIST	CELEBA
VAMP	prior	34.5	67.2
	ours	**32.7**	**60.9**
IWAE	prior	**32.4**	67.6
	ours	33.8	**60.3**
AAE	prior	19.1	64.8
	ours	**11.7**	**51.4**
VAEGAN	prior	8.7	39.7
	ours	**6.1**	**31.4**

Another approach that is interesting to compare to is the 2-stage VAE model proposed in (Dai and Wipf, 2018). Our method can indeed be seen as part of the methods trying to counterbalance the poor expressiveness of the prior distribution. In (Dai and Wipf, 2018), the authors argue that the actual distribution of the latent codes (i.e. the aggregated posterior) is "likely not close to a standard Gaussian distribution" (Dai and Wipf, 2018) leading to a distribution mismatch degrading the generation capability of the model. To address this issue, they propose to use a second VAE to estimate the learned distribution of the latent variables. Our approach starts with the same observation that the latent codes have no reason to follow the prior. However, it differs since we propose to adopt a fully geometric perspective and propose instead a sampling scheme using the intrinsic uniform distribution defined on the learned Riemannian manifold.

We nonetheless compare our method with models obtained with the official implementation provided by the authors of (Dai and Wipf, 2018) on MNIST and CELEBA. To allow a fair comparison, we simply plug our method to the obtained trained models and build the metric using the posteriors coming from the 1^{st} stage VAE. In Table 3.7, we compare the FID obtained 1) with the first stage

VAE (*i.e.* prior), 2) with the second stage VAE (Dai and Wipf, 2018) and 3) with our method. Again, our proposed generation method allows to achieve lower FID results.

Table 3.7: FID (lower is better) vs. the test set using the 2-stage VAE implementation (Dai and Wipf, 2018) for either the reconstructed samples (recon.), using the prior (1st stage), using the 2-stage approach (2nd stage) or by plugging our generation method.

Dataset	Nets	Recon.	1st stage	2nd stage	Ours
MNIST	similar to (Chen et al., 2016a)	14.8	20.0	12.9	**9.9**
CELEBA	similar to (Chen et al., 2016a)	44.9	67.8	53.3	**49.6**
CELEBA	similar to (Tolstikhin et al., 2018)	34.3	70.8	40.7	**37.9**

Chapter 4

Pythae: Unifying Generative Autoencoders in Python

In recent years, deep generative models have attracted increasing interest due to their capacity to model complex distributions. Among those models, variational autoencoders have gained popularity as they have proven both to be computationally efficient and yield impressive results in multiple fields. Following this breakthrough, extensive research has been done in order to improve the original publication, resulting in a variety of different VAE models in response to different tasks. In this chapter we present ***Pythae****, a versatile open-source Python library providing both a unified implementation and a dedicated framework allowing straightforward,* reproducible *and reliable use of generative autoencoder models. As an example of application, we propose to use this library to perform a case study benchmark where we present and compare 19 generative autoencoder models representative of some of the main improvements on downstream tasks such as image reconstruction, generation, classification, clustering and interpolation. The open-source library can be found at* `https://github.com/clementchadebec/benchmark_VAE`

This chapter was published at the NeurIPS Conference 2022 (track on Benchmark and Datasets). See (Chadebec et al., 2022c).

4.1 Introduction

Over the past few years, generative models have proven to be a promising approach for modeling datasets with complex inherent distributions such a natural images. Among those, Variational AutoEncoders (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) have gained popularity due to their computational efficiency and scalability, leading to many applications such as speech modeling (Blaauw and Bonada, 2016), clustering (Dilokthanakul et al., 2017; Yang et al., 2019), data augmentation (Chadebec et al., 2022b) or image generation (Razavi et al., 2020). Similarly to autoencoders, these models encourage good reconstruction of an observed input data from a latent representation, but they further assume latent vectors to be random variables involved in the generation process of the observed data. This imposes a latent structure wherein latent variables are driven to follow a prior distribution that can then be used to generate new data. Since this breakthrough, various contributions have been made to enrich the original VAE scheme through new generating strategies (Dilokthanakul et al., 2017; Tomczak and Welling, 2018; Ghosh et al., 2020; Bauer and Mnih, 2019a; Pang et al., 2020), reconstruction objectives (Larsen et al., 2016; Snell et al., 2017) and more adapted latent representations (Higgins et al., 2017; Kim and Mnih, 2018; Van Den Oord et al., 2017; Arvanitidis et al., 2018; Chadebec et al., 2022b) to cite a few. A drawback of VAEs is that due to the intractability of the log-likelihood objective function, VAEs have to resort to optimizing a lower bound on the true objective as a proxy, which has been mentioned as a major limitation of the model (Burda et al., 2016; Alemi et al., 2016; Higgins et al., 2017; Cremer et al., 2018; Zhang et al., 2018a). Hence, extensive research has been proposed to improve this bound through richer distributions (Salimans et al., 2015; Rezende and Mohamed, 2015; Kingma et al., 2016; Caterini et al., 2018). More recently, it has been shown that autoencoders can be turned into generative approaches through latent density estimation (Ghosh et al., 2020), extending the concept of *Generative AutoEncoders* (GAE) to a more general class of autoencoder models.

Nonetheless, most of this research has been done in parallel across disjoint sub-fields of research and to the best of our knowledge little to no work has been done on homogenizing and integrating these distinct methods in a common framework. Moreover, for many of the aforementioned publications, implementations may not be available or maintained, therefore requiring time-consuming re-implementation. This induces a strong bottleneck for research to move forward in this field and makes reproducibility challenging, which calls for the need of a unified generative autoencoder framework. To address this issue we introduce **Pythae** (**Pyth**on **A**uto**E**ncoder), a versatile open source Python library for generative autoencoders providing unified implementations of common methods, along with a reproducible framework allowing for easy model training, data generation and experiment tracking. We then propose to illustrate the usefulness of the proposed library on a benchmark case study of 19 generative autoencoder methods on classical image datasets. We consider five different downstream tasks: image reconstruction and generation, latent vector classification and clustering, and image interpolation on three well known imaging datasets.

4.2 Variational Autoencoders

In this section, we recall the original VAE setting and present some of the main improvements that were proposed to enhance the model.

4.2.1 Background

Given $x \in \mathbb{R}^D$, a set of observed variables deriving from an unknown distribution $p(x)$, a VAE assumes that there exists $z \in \mathbb{R}^d$ such that z is a latent representation of x. The generation process of x thus decomposes as

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p_z(z)dz\,, \tag{4.1}$$

where p_z is the *prior distribution* on the latent space $\mathbb{R}^d$. The distribution $p_\theta(x|z)$ is referred to as the *decoder* and is modeled with a simple parametric distribution whose parameters are given by a neural network. Since the true posterior $p_\theta(z|x)$ is most of the time intractable due to the integral in Eq. (4.1) recourse to Variational Inference (Jordan et al., 1999) is needed and a variational distribution $q_\phi(z|x)$ which we refer to as the *encoder* is introduced. The approximate posterior q_ϕ is again taken as a simple parametric distribution whose parameters are also modeled by a neural network. This allows to define an unbiased estimate $\widehat{p}_\theta$ of the marginal distribution $p_\theta(x)$ using importance sampling with $q_\phi(z|x)$ *i.e.* $\widehat{p}_\theta(x) = \frac{p_\theta(x|z)p_z(z)}{q_\phi(z|x)}$ and $\mathbb{E}_{z\sim q_\phi}\left[\widehat{p}_\theta\right] = p_\theta$. Applying Jensen's inequality leads to a lower bound on the likelihood given in Eq. (4.1):

$$\log \underbrace{\mathbb{E}_{z\sim q_\phi}\left[\widehat{p}_\theta(x)\right]}_{p_\theta(x)} \geq \mathbb{E}_{z\sim q_\phi}\left[\log \widehat{p}_\theta(x)\right] = \underbrace{\mathbb{E}_{z\sim q_\phi}[\log p_\theta(x|z)]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{KL}\left[q_\phi(z|x)||p_z(z)\right]}_{\text{regularization}}\,, \tag{4.2}$$

where $\mathcal{D}_{KL}(p||q)$ is the Kullback-Leibler divergence between distributions p and q. This bound is referred to as the Evidence Lower Bound (ELBO) (Kingma and Welling, 2014) and is used as the training objective to maximize in the traditional VAE scheme. It can be interpreted as a two terms objective (Ghosh et al., 2020) where the *reconstruction* loss forces the output of the decoder to be close to the original input x, while the *regularization* loss forces the posterior distribution $q_\phi(z|x)$ outputted by the encoder to be close to the prior distribution $p_z(z)$. Under standard VAE assumption, the prior distribution is a multivariate standard Gaussian $p_z = \mathcal{N}(0, I_d)$, the approximate posterior is set to $q_\phi(z|x) = \mathcal{N}\big(z\big|\mu(x), \Sigma(x)\big)$ where $\big(\mu(x), \Sigma(x)\big)$ are outputs of the encoder network.

4.2.2 Improvements Upon the Classical VAE Method

Building on the breakthrough of VAEs, several papers have proposed improvements to the model. In this section we present 4 axes which we consider to be representative of the major advancements made on VAEs, as well as classical models characterizing the main improvements within each of these axes.

Improving the prior It has been shown that the role of the prior distribution p_z is crucial in the good performance of the VAE (Hoffman and Johnson, 2016) and choosing a family of overly simplistic priors can lead to over-regularization (Connor et al., 2021) and poor reconstruction performance (Dai and Wipf, 2018). In particular, it was shown that the prior maximizing the ELBO objective is the *aggregated posterior* $q(z) = \frac{1}{N}\sum_{i=1}^{N} q_\phi(z|x_i)$ (Tomczak and Welling, 2018). However, it should be noted that a perfect fit between the prior and the aggregated posterior is not necessarily desired since it has been shown in (Bauer and Mnih, 2019b; Tomczak and Welling, 2018) that it may lead to over-fitting as it essentially amounts to the model memorizing the training set. Hence, multi-modal

priors (Nalisnick et al., 2016; Dilokthanakul et al., 2017; Tomczak and Welling, 2018) were proposed, followed by hierarchical latent variable models (Sønderby et al., 2016; Klushyn et al., 2019) and prior learning based approaches (Chen et al., 2016b; Aneja et al., 2020) to address the poor expressiveness of the prior distribution and model richer generative distributions. Considering a specific geometry of the latent space also led to alternative priors taking into account geometrical aspects of the latent space (Davidson et al., 2018; Falorsi et al., 2018; Mathieu et al., 2019a; Arvanitidis et al., 2018; Chen et al., 2018a; Shao et al., 2018; Kalatzis et al., 2020; Chadebec et al., 2022b). Another interesting approach proposed for instance in (Van Den Oord et al., 2017; Ghosh et al., 2020) consists in using density estimation post training with another distribution or normalizing flows (Rezende and Mohamed, 2015) on the learned latent codes.

Towards a better lower bound Another major axis of improvement of the VAE model has been to tighten the gap between the ELBO objective and the true log probability (Burda et al., 2016; Alemi et al., 2016; Higgins et al., 2017; Cremer et al., 2018; Zhang et al., 2018a). The ELBO objective can indeed be written as the difference between the true log probability and a KL divergence between the approximate posterior and the true posterior

$$\mathcal{L}_{\text{ELBO}}(x) = \log p_\theta(x) - \mathcal{D}_{KL}\left[q_\phi(z|x)||p(z|x)\right] .$$

Hence, if one wants to make the ELBO gap tighter, particular attention should be paid to the choice in the approximate posterior $q_\phi(z|x)$. In the original model, $q_\phi(z|x)$ is chosen as a simple distribution for tractability of the ELBO in Eq. (4.2). However, several approaches have been proposed to extend the choice of q_ϕ to a wider class of distributions using MCMC sampling (Salimans et al., 2015) or normalizing flows (Rezende and Mohamed, 2015). For instance, (Kingma et al., 2016) improve upon the works of (Rezende and Mohamed, 2015) with an inverse auto-regressive normalizing flow (IAF), a new type of normalizing flow that better scales to high-dimensional latent spaces. With this objective in mind a Hamiltonian VAE aimed at targeting the true posterior during training with a Hamiltonian Monte Carlo (Neal, 2005) inspired scheme was proposed (Caterini et al., 2018) and extended to Riemannian latent spaces in (Chadebec et al., 2022b).

Encouraging disentanglement Although there is no clear consensus upon the definition of disentanglement, it is commonly referred to as the independence between features in a representation (Mathieu et al., 2019b). This is a desirable behavior for VAEs, as it is argued that disentangled features may be more representative and interpretable (Higgins et al., 2017). In that regard, several approaches have been proposed encouraging a better disentanglement of the features in the latent space. (Higgins et al., 2017) first argue that increasing the weight of the KL divergence term in the ELBO loss enforces a higher disentanglement of the latent features as the posterior probability is forced to match a multivariate normal standard Gaussian. Following this idea, (Burgess, 2018) propose to achieve disentanglement by gradually increasing the proximity between the posterior and the prior (Burgess, 2018). Other methods challenge the view that disentanglement can be achieved by simply forcing the posterior to match the prior, or raise the point that in this case disentanglement is achieved at the cost of a bad reconstruction. From these observations, new approaches arise such as (Kim and Mnih, 2018) who augment the VAE objective with a penalty that encourages factorial representation of the marginal distributions, or (Chen et al., 2018b) that enforce a penalty on the total correlation favouring disentanglement.

Amending the distance between distributions It can be stressed that the reconstruction term $\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$ in eq. (4.2) has a crucial role in the reconstruction and that its choice should be dependent of the application. For instance, methods using a discriminator (Larsen et al., 2016) or using a deterministic differentiable loss function (Snell et al., 2017) acting as a distance between the input data and its reconstruction were also proposed. The second term in the ELBO measures the distance between the approximate posterior and the prior distribution through the KL divergence and it has however been argued that other distances between probability distributions could be used instead. Hence, approaches using a GAN to distinguish samples from the posterior from samples from the prior distribution (Makhzani et al., 2015) or methods based on optimal transport have also been proposed (Tolstikhin et al., 2018; Zhao et al., 2019).

4.3 The Pythae Library

Why Pythae ? To the best of our knowledge, although some well referenced libraries grouping different Variational Auto-Encoder methods exist (*e.g.* (Subramanian, 2020)), there exists no framework providing both adaptable and easy-to-use unified implementations of state-of-the-art Generative AutoEncoder (GAE) methods. This induces both a strong brake for reproducible research and democratization of the models since implementations might be difficult to adapt to other use-cases, no longer maintained, or completely unavailable.

Project vision Starting from this observation, we created **Pythae**, an open-source python library inspired from (Pedregosa et al., 2011; Wolf et al., 2020) providing unified implementations of generative autoencoding methods, allowing for easy use and training of GAE models. Pythae is designed with the following points in mind:

- **Usable by all** Pythae makes GAE models accessible to all - beginners to experts. This means beginners can run *ready-to-use* models with a few lines of code, while more advanced users can easily access and adapt different methods to their specific use-cases, with custom encoder/decoder definition. Indeed, the library was designed to be flexible enough to allow users to use existing implementations on their own data, with custom model hyper-parameters, training configurations and network architectures.

  The library has an online documentation[1] and is also explained and illustrated through tutorials available either on a local machine or on the *Google Colab* platform (Bisong, 2019).

- **Unified implementation** The brick-like structure of Pythae allows for seamless but efficient interchange between models, sampling techniques, network architectures, model hyper-parameters and training schemes. Pythae is unit-tested ensuring code quality and continuous development with a code coverage of 98% as of release 0.6. The library is made available on *pip* and *conda* allowing an easy integration. Its development is performed through releases that ensure stable and robust implementations.

- **A reproducible research environment** Pythae is open to all and as such encourages transparent and reproducible research, as illustrated in the next section. With a variety of different

[1]The full documentation can be found at https://pythae.readthedocs.io/en/latest/.

interchangeable models gathered in a common library, it can be used as a sandbox for research and applications. Moreover, the library also integrates an easy-to-use experiment tracking tool (*wandb*) (Biewald, 2020) allowing to monitor runs launched with Pythae and compare them through a graphic interface, and an online model sharing tool, the HuggingFace Hub, allowing to share models with peers.

- **Evolving and driven by the community** Pythae's design is intended to evolve with the addition of new models to enrich the existing model base. Furthermore, peers can contribute by reviewing and submitting models to enrich the library, a few of which have already been added at the time of this publication.

Code structure Pythae was thought for easy model training and data generation, while striving for simplicity with a quick and user-friendly model selection and configuration. The backbone of the library is the module *pythae.models* in which all the autoencoder models are implemented. Each model implementation is accompanied with a configuration *dataclass* containing any hyper-parameters relative to the model and allowing easing configuration loading and saving from *json* files.

All the models are implemented using a common API allowing for a seamless integration with *pythae.trainers* (for training) and *pythae.samplers* (for generation) along with a simplified usage as illustrated in Fig. 4.1. In particular, Pythae provides pipelines allowing to train an autoencoder model or to generate new data with only a few lines of code, as shown in Appendix 4.6.1.

It mainly relies on the Pytorch (Paszke et al., 2017) framework and in its basic usage only essential hyper-parameter configurations and data (arrays or tensors) are needed to launch a model training or generation. More advanced options allowing further flexibility such as defining custom encoder and decoder neural-networks are also available and can be found in the documentation and tutorials. It can adapt to various types of data through the use of different already-implemented or user provided encoder and decoder neural-network architectures. In addition, Pythae also provides several ways to generate new data through different popular sampling methods in the *pythae.samplers* module. We detail some aspects of the library in Appendix 4.6.1.

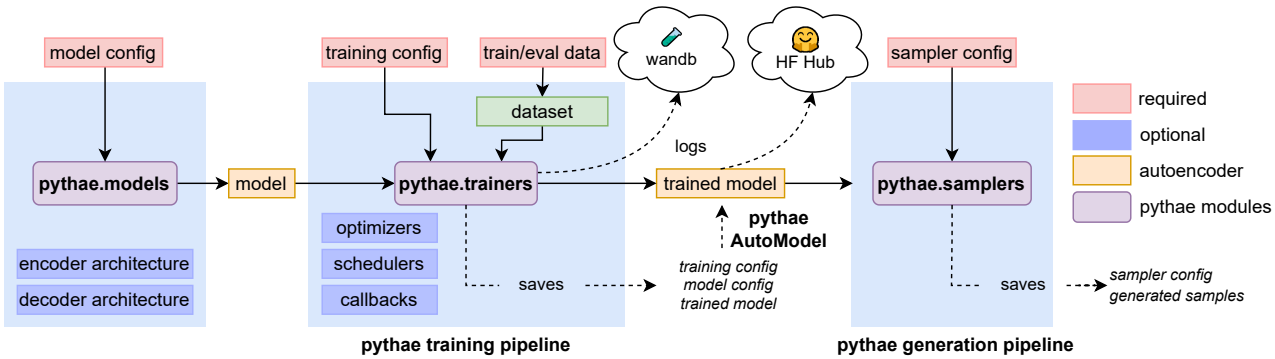


Figure 4.1: Pythae library diagram

4.4 Case Study Benchmark

By nature of its structured framework, Pythae allows for easy comparison between models on any chosen task. As an illustrative purpose, we propose a case study where we use Pythae to perform a straightforward benchmark comparison of models implemented in Pythae on a selection of well-known elementary tasks. The aim of these tasks is to underline general trends within groups of GAEs, based on common behaviors, as well as judge the versatility of the models. However, this benchmark should not be considered as a means to rank models on these tasks, as performances depend on sometimes complex hyper-parameter tuning and training, which we consider to be outside of the scope of this benchmarking use case. The scripts used for the benchmark are provided in supplementary materials.

4.4.1 Benchmark Setting

In this section, we present the setting of the benchmark. Comprehensive results for all the experiments are available through the monitoring tool (Biewald, 2020) used in Pythae to allow complete transparency.

The Data To perform the different tasks presented in this chapter, 3 classical and widely used image datasets are considered: MNIST (LeCun, 1998), CIFAR10 (Krizhevsky et al., 2009) and CELEBA (Liu et al., 2015). These datasets are publicly available, widely used for generative model related papers and have well known associated metrics in the literature. Each dataset is split into a train set, a validation set and a test set. For MNIST and CIFAR10 the validation set is composed of the last 10k images extracted from the official train set and the test set corresponds to the official one. For CELEBA, we use the official train/val/test split.

The Models We propose to compare 19 generative autoencoder models representative of the improvements proposed in the literature and presented in Sec. 4.2.2. Descriptions and explanations of each implemented model can be found in Appendix 4.6.4. We use as baseline an Autoencoder (**AE**) and a Variational Autoencoder (**VAE**). To assess the influence of a *more expressive prior*, we propose using a VAE with VAMP prior (**VAMP**) (Tomczak and Welling, 2018) and regularized autoencoders with either a gradient penalty (**RAE-GP**) or a L2 penalty on the weights of the decoder (**RAE-L2**) that use *ex-post* density estimation (Ghosh et al., 2020). To represent models trying to reach a *better lower bound*, we choose a Importance Weighted Autoencoder (**IWAE**) (Burda et al., 2016) and VAEs adding either simple linear normalizing flows (**VAE-lin-NF**) (Rezende and Mohamed, 2015) or using IAF (**VAE-IAF**) (Kingma et al., 2016). For *disentanglement-based models*, we select a β-**VAE** (Higgins et al., 2017), a **FactorVAE** (Kim and Mnih, 2018) and a β-**TC VAE** (Chen et al., 2018b). To stress the *influence of the distance* used between distributions we add a Wasserstein Autoencoder (**WAE**)(Tolstikhin et al., 2018) and an **InfoVAE** (Zhao et al., 2019) with either Inverse Multi-Quadratic (IMQ) or a Radial Basis Function kernel (RBF) together with an Adversarial Autoencoder (**AAE**)(Makhzani et al., 2015), a **VAEGAN**(Larsen et al., 2016) and a VAE using structural similarity metric for reconstruction (**MSSSIM-VAE**) (Snell et al., 2017). Finally, we add a **VQVAE** (Van Den Oord et al., 2017) since having a discrete latent space has shown to yield promising results.

Models implemented in Pythae requiring too much training time or more intricate hyper-parameter tuning were excluded from the benchmarks.

In the following, we will distinguish *AE-based* (autoencoder-based) methods (AE, RAE, WAE and VQVAE) from the other *variational-based* methods.

Training paradigm Each of the aforementioned models is equipped with the same neural network architecture for both the encoder and decoder leading to a comparable number of parameters [2]. For each task, 10 different configurations are considered for each model, allowing a simple exploration of the models' hyper-parameters, leading to 10 trained models for each dataset and each neural network type (ConvNet or ResNet) leading to a total of 1140 models[3]. It is important to note that the hyper-parameter exploration is not exhaustive and models sensitive to hyper-parameter tuning may have better performances with a more extensive parameter search. The sets of hyper-parameters explored are detailed in Appendix 4.6.4 for each model.

4.4.2 Experiments

In this section, we present the main results observed on 5 downstream tasks.

Fixed Latent Dimension

In this first part, latent dimensions are set to 16, 256 and 64 for the MNIST, CIFAR10 and CELEBA datasets respectively, as we observed those latent dimensions to lead to good performances. See results per model and across the 10 configurations specified in Appendix 4.6.3 to assess the influence of the parameters on the tasks.

Task 1: Image reconstruction For each model, reconstruction error is evaluated by selecting the configuration minimizing the Mean Square Error (MSE) between the input and the output of the model on the validation set, while results are shown on the test set[4]. We show in Table 4.1 the MSE and Frechet Inception Distance[5] (FID) (Heusel et al., 2017) of the reconstructions from this model on the test set. It is important to note that using the MSE as a metric places models using different reconstruction losses (VAEGAN and MSSSIM-VAE) at a disadvantage.

As expected, the autoencoder-based models seem to perform best for the reconstruction task. Nonetheless, this experiment also shows the interest of adding regularization to the autoencoder since improvements over the AE (RAE-GP, RAE-L2) achieve better performance than the regular

[2]Some models may actually have additional parameters in their intrinsic structure *e.g.* a VQVAE learns a dictionary of embeddings, a VAMP learns the pseudo-inputs, a VAE-IAF learns auto-regressive flows. Nonetheless, since we work on images, the number of parameters remains in the same order of magnitude.

[3]The training setting (curves, configs ...) can be found at https://wandb.ai/benchmark_team/trainings while detailed experimental set-up is available in Appendix 4.6.3.

[4]See the whole results at https://wandb.ai/benchmark_team/reconstructions

[5]We used the implementation of https://github.com/bioinf-jku/TTUR

AE. Moreover, β-VAE type models demonstrate their versatility as small enough β values can lead to less regularization, therefore favouring a better reconstruction.

Task 2: Image generation We consider an image generation task with the trained models. In this experiment, we also explore different ways of sampling new data, either 1) using a simple distribution chosen as $\mathcal{N}(0, I_d)$ and corresponding to the standard prior for variational approaches ($\mathcal{N}$); 2) fitting a 10 components mixture of Gaussian in the latent space post training as proposed in (Ghosh et al., 2020) (GMM), 3) fitting a normalizing flow taken as a Masked Autoregressive Flow (MAF) (Papamakarios et al., 2017) or 4) fitting a VAE in a similar fashion as (Dai and Wipf, 2018). For the MAF, two-layer MADE (Germain et al., 2015) are used. For each sampler, we select the models achieving the lowest FID on the validation set and compute the Inception Score[6] (IS) (Salimans et al., 2016) and the FID on the test set[7]. It should be noted that although the use of IS and FID has been criticized (Barratt and Sharma, 2018; Shmelkov et al., 2018; Chong and Forsyth, 2020; Morozov et al., 2020; Jung and Keuper, 2021), we still choose to use those metrics for clarity's sake as they are within the most commonly used metrics for image generation on generative models. The main results are shown in Table 4.3 for the normal and GMM sampler (see Appendix 4.6.4 for the other sampling schemes).

One of the key findings of this experiment is that performing *ex-post* density (therefore not using the standard Gaussian prior) for the variational approach tends to almost always lead to better generation metrics even when a simple 10-components mixture of Gaussian is used. Interestingly, we note that when a more advanced density estimation model such as a MAF is used, results appear equivalent to those of the GMM (see Appendix 4.6.4). This may be due to the simplicity of the database we used and in consequence of the distribution of the latent codes that can be approximated well enough with a GMM. It should nonetheless be noted that the number of components in the GMM remains a key parameter which was set to the number of classes for MNIST and CIFAR10 since it is known, however too high a value may lead to overfitting while a low one may lead to worse results. VAEGAN clearly surpasses peers on MNIST and CELEBA with impressive results when compared to other models. Nonetheless, fitting such a model may be challenging due to the adversarial approach and results can be considerably affected if parameters are not chosen correctly (see Appendix 4.6.4).

Task 3: Classification To measure the meaningfulness of the learned latent representations we perform a simple classification task with a single layer classifier as proposed in (Coates et al., 2011). The rationale behind this is that if a GAE succeeds in learning a disentangled latent representation a simple linear classifier should perform well (Berthelot* et al., 2019). A single layer classifier is trained in a supervised manner on the latent embeddings of MNIST and CIFAR10. The train/val/test split used is the same as for the autoencoder training. For each model configuration, we perform 20 runs of the classifier on the latent embeddings and define the best hyper-parameter configuration as the one achieving the highest mean accuracy on these 20 runs on the validation set. We report the mean accuracy on the test set across the 20 runs for the selected configuration in Table 4.2 (left)[8].

[6]We used the implementation of https://github.com/openai/improved-gan
[7]See the whole results at https://wandb.ai/benchmark_team/generations
[8]See the whole results at https://wandb.ai/benchmark_team/classifications

As expected, models explicitly encouraging disentanglement in the latent space such as the β-VAE and β-TC VAE achieve better classification when compared to a standard VAE. Nonetheless, AE-based models seem again the best suited for such a task since variational approaches tend to enforce a continuous space, consequently bringing latent representations of different classes closer to each other. As a general observation, we can state that models with a more flexible prior achieve better results on this task.

Task 4: Clustering As a complement to the previous task, performing clustering directly in the latent space of the trained autoencoders can give insights on the quality of the latent representation. Indeed, a well defined latent space will maintain the separation of the classes inherent to the datasets, leading to easy and stable k-means performances. To do so, we propose to fit 100 separate runs of the k-means algorithm and we show the mean accuracy obtained on the train embeddings in Table 4.2 (right)[9]. This experiment allows us to explore and measure the *clusterability* of the generated latent spaces (Berthelot* et al., 2019). To measure accuracy we assign the label of the most prevalent class to each cluster.

The conclusions of this experiment are slightly different from the previous one since models targeting disentanglement seem to be equalled by the original VAE. Interestingly, adversarial approaches and other alternatives to the standard VAE KL regularization method seem to achieve the best results.

Task 5: Interpolation Finally, we propose to assess the ability of the model to perform meaningful interpolations. For this task, we consider a starting and ending image in the test set of MNIST and CIFAR10 and perform a linear interpolation in the generated latent spaces between the two encoded images. We show in Appendix 4.6.2 the decoding along the interpolation curves. For this task, no metric was found relevant since the notion of "good" interpolation can be disputable. Nonetheless, the obtained interpolated images can be reconstructed and qualitatively evaluated.

For this task, variational approaches were found to obtain better results as the inherent structure the posterior distribution imposes in the latent space results in a "smoother" transition from one image to another when compared to autoencoders that mainly superpose images, especially in higher dimensional latent spaces.

Varying Latent Dimension

An important parameter of autoencoder models which is too often neglected in the literature is the dimension of the latent space. We now propose to keep the same configurations as previously but re-evaluate Tasks 1 to 5 with the latent space varying in the range $[16, 32, 64, 128, 256, 512]$. Results are shown in Fig. 4.2 for MNIST and a ConvNet[10] (see Appendix 4.6.4 for CIFAR, ResNet and interpolations). For the generation task, we select the sampler with lowest FID on the validation set.

[9] See the whole results at https://wandb.ai/benchmark_team/clustering

[10] MSSSIM-VAE was removed from this plot for visualization purposes.

Assessing the influence of the latent dimension A clear difference in behavior is exhibited between variational-based and AE-based methods. For each given task, AEs share a common trend with respect to the evolution of the latent dimension: a common optimal latent dimension within the range $[16, 32, 64, 128, 256, 512]$ is found for each task, but differs drastically among different tasks (*e.g.* 512 for reconstruction, 16 for generation, either 16 or 512 for classification and 512 for clustering with the MNIST dataset). This suggests the existence of a common intra-group optimal latent space dimension for a given task. In addition, we observe that β-VAE type methods (with the right hyper-parameter choice) can exhibit similar behaviors to AE models. The same observation can be made for variational-based methods, where it is interesting to note that although lower performances are achieved, the apparent optimal latent dimension varies less with respect to the choice of the task. Therefore, a latent dimension of 16 to 32 appears to be the optimal choice for all 4 Tasks on the MNIST dataset, and 32 to 128 on the CIFAR10 dataset. For AE based methods, the choice of the latent dimension seems to have a measurable impact on its performance, and the optimal choice varies drastically from one dataset to another. It should be noted that unsupervised tasks such as clustering of the latent representation of the CIFAR10 dataset are hard and models are expected to perform poorly, leading to less interpretable results.

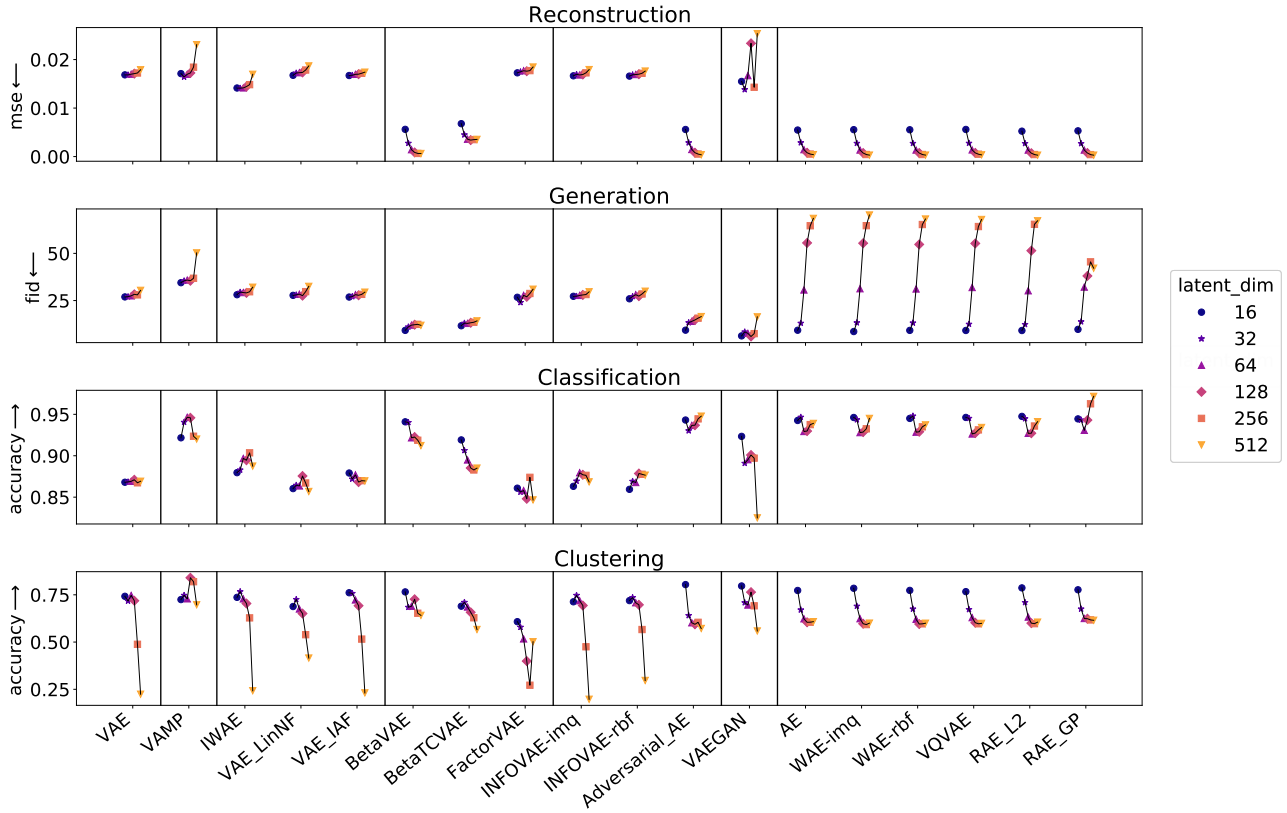


Figure 4.2: *From top to bottom:* Evolution of the reconstruction MSE, generation FID, classification accuracy and clustering accuracy with respect to the latent space dimension on the MNIST dataset.

Table 4.1: Mean Squared Error (10^{-3}) and FID (lower is better) computed with 10k samples on the test set. For each model, the best configuration is the one achieving the lowest MSE on the validation set.

Model	ConvNet						ResNet					
	MNIST (16)		CIFAR10 (256)		CELEBA (64)		MNIST (16)		CIFAR10 (256)		CELEBA (64)	
	MSE ↓	FID ↓	MSE ↓	FID ↓	MSE ↓	FID ↓	MSE ↓	FID ↓	MSE ↓	FID ↓	MSE ↓	FID ↓
VAE	16.85	30.71	16.24	218.66	9.83	49.22	17.24	36.06	16.33	176.63	10.59	58.75
VAMP	24.17	44.95	17.45	221.40	10.81	51.64	17.11	37.58	16.87	177.03	11.50	60.89
IWAE	14.14	34.28	16.19	237.14	9.47	50.00	15.79	38.74	16.02	183.37	10.14	60.18
VAE-lin-NF	16.75	31.14	16.57	221.39	9.90	49.84	17.23	36.74	16.59	177.08	10.68	58.73
VAE-IAF	16.71	30.64	16.33	223.65	9.87	50.05	17.05	35.98	16.39	177.05	10.63	58.41
β-VAE	5.61	10.55	3.60	50.55	7.28	**46.96**	5.87	15.81	2.40	55.67	7.78	51.59
β-TC VAE	6.78	14.11	5.06	53.49	7.65	50.82	7.12	18.44	4.05	66.89	8.08	52.70
Factor VAE	17.27	30.39	16.41	224.3	10.16	53.61	18.13	37.97	16.55	176.8	10.93	59.46
InfoVAE-IMQ	16.65	30.62	16.19	216.44	9.81	50.51	17.17	37.33	16.32	173.79	10.63	58.04
InfoVAE-RBF	16.59	30.63	16.23	217.52	9.85	50.14	17.01	37.04	16.32	175.37	10.64	58.68
AAE	5.59	10.87	2.60	40.66	7.25	50.22	5.98	17.01	**2.33**	55.93	7.76	50.97
MSSSIM-VAE	32.60	37.91	39.42	276.70	35.60	124.52	33.67	40.25	39.61	254.34	35.43	119.92
VAEGAN	15.49	**5.54**	31.40	289.35	8.91	86.58	23.25	**11.35**	30.22	300.07	9.32	86.32
AE	5.47	11.61	2.82	41.98	7.03	51.08	6.13	13.74	2.34	55.43	7.74	50.54
WAE-IMQ	5.55	11.29	2.81	41.79	7.04	52.11	5.78	16.21	2.34	56.55	7.74	50.50
WAE-RBF	5.53	11.34	2.82	42.21	7.03	51.43	5.80	16.14	2.34	56.00	7.74	51.38
VQVAE	5.59	11.02	2.84	44.60	7.06	52.27	6.00	15.27	2.34	**55.84**	**7.73**	**50.29**
RAE-L2	**5.24**	15.37	**2.25**	49.28	**6.90**	53.98	**5.76**	17.27	2.35	57.85	7.74	51.07
RAE-GP	5.31	12.08	2.81	**41.15**	7.06	51.85	5.83	15.69	2.34	56.71	7.76	51.36

Table 4.2: *Left:* Mean test accuracy of a single layer classifier on the embedding obtained in the latent spaces of each model average on 20 runs. *Right:* Mean accuracy of 100 k-means fitted on the training embeddings coming from the autoencoders.

Model	Classification				Clustering			
	ConvNet		ResNet		ConvNet		ResNet	
	MNIST	CIFAR10	MNIST	CIFAR10	MNIST	CIFAR10	MNIST	CIFAR10
VAE	86.75(0.05)	32.61(0.03)	86.80(0.03)	32.37(0.03)	69.71(2.01)	17.18(0.68)	74.21(0.97)	18.12(0.74)
VAMP	92.17(0.02)	33.46(0.17)	92.58(0.04)	33.03(0.22)	67.26(1.25)	24.03(0.24)	72.48(0.96)	23.35(0.11)
IWAE	87.96(0.04)	31.86(0.04)	88.18(0.03)	32.26(0.04)	63.93(1.73)	19.55(0.67)	73.66(2.30)	18.44(0.86)
VAE-lin-NF	86.04(0.04)	31.57(0.02)	85.85(0.05)	31.74(0.03)	65.48(2.76)	17.09(0.64)	68.80(3.65)	18.74(0.68)
VAE-IAF	88.32(0.02)	33.52(0.02)	87.91(0.02)	32.41(0.02)	75.31(1.69)	17.81(0.73)	76.11(2.15)	18.42(0.66)
β-TC VAE	90.96(0.02)	45.40(0.05)	91.91(0.02)	**42.17(0.07)**	65.68(0.91)	24.14(0.65)	68.98(2.67)	**25.57(0.61)**
Factor VAE	86.08(0.06)	31.38(0.04)	83.44(0.05)	31.76(0.04)	51.02(1.73)	15.77(0.60)	60.79(2.06)	17.56(0.68)
InfoVAE-IMQ	86.33(0.04)	32.48(0.02)	86.31(0.06)	32.10(0.05)	68.17(2.34)	16.65(0.80)	71.31(2.62)	18.10(0.79)
InfoVAE-RBF	85.94(0.03)	32.50(0.03)	86.12(0.04)	31.67(0.03)	66.02(1.14)	16.22(0.69)	71.93(1.91)	18.61(0.67)
AAE	93.28(0.03)	43.93(0.07)	94.31(0.03)	40.62(0.11)	74.19(3.22)	24.72(0.75)	**80.41(2.09)**	24.76(0.53)
MSSSIM-VAE	78.30(0.03)	20.26(0.06)	76.54(0.03)	20.24(0.04)	49.33(1.32)	11.70(0.19)	48.58(1.37)	11.70(0.17)
VAEGAN	92.34(0.02)	26.56(0.04)	90.31(0.03)	29.90(0.03)	**77.29(1.19)**	17.20(0.45)	79.67(0.90)	22.23(0.44)
AE	93.81(0.02)	42.15(0.07)	94.26(0.03)	40.47(0.13)	73.55(0.60)	23.19(0.52)	77.30(0.84)	23.18(0.37)
WAE-IMQ	93.60(0.02)	**45.89(0.07)**	94.62(0.03)	41.35(0.03)	72.33(2.92)	23.81(0.61)	78.46(3.48)	25.09(0.82)
WAE-RBF	93.72(0.02)	43.38(0.08)	94.51(0.02)	40.63(0.08)	74.20(1.94)	23.70(0.71)	77.33(1.92)	24.66(0.63)
VQVAE	93.45(0.02)	42.89(0.07)	**94.63(0.04)**	40.40(0.09)	72.61(0.40)	23.85(0.48)	76.68(2.36)	23.68(0.37)
RAE-L2	94.75(0.01)	42.76(0.08)	94.43(0.03)	40.22(0.05)	74.07(0.36)	23.77(0.54)	78.66(0.29)	24.84(0.73)
RAE-GP	**94.10(0.02)**	43.66(0.07)	94.45(0.02)	40.93(0.14)	72.88(0.52)	**24.84(0.53)**	77.66(1.29)	23.86(0.32)

Table 4.3: Inception Score (higher is better) and FID (lower is better) computed with 10k samples on the test set. For each model and sampler we report the results obtained by the model achieving the lowest FID score on the validation set.

Model	Sampler	ConvNet						ResNet					
		MNIST		CIFAR10		CELEBA		MNIST		CIFAR10		CELEBA	
		FID ↓	IS ↑	FID	IS	FID	IS ↑	FID ↓	IS ↑	FID	IS	FID	IS
VAE	$\mathcal{N}$	28.5	2.1	241.0	2.2	54.8	1.9	31.3	2.0	181.7	2.5	66.6	1.6
	GMM	26.9	2.1	235.9	2.3	52.4	1.9	32.3	2.1	179.7	2.5	63.0	1.7
VAMP	VAMP	64.2	2.0	329.0	1.5	56.0	1.9	34.5	2.1	181.9	2.5	67.2	1.6
IWAE	$\mathcal{N}$	29.0	2.1	245.3	2.1	55.7	1.9	32.4	2.0	191.2	2.4	67.6	1.6
	GMM	28.4	2.1	241.2	2.1	52.7	1.9	34.4	2.1	188.8	2.4	64.1	1.7
VAE-lin NF	$\mathcal{N}$	29.3	2.1	240.3	2.1	56.5	1.9	32.5	2.0	185.5	2.4	67.1	1.6
	GMM	28.4	2.1	237.0	2.2	53.3	1.9	33.1	2.1	183.1	2.5	62.8	1.7
VAE-IAF	$\mathcal{N}$	27.5	2.1	236.0	2.2	55.4	1.9	30.6	2.0	183.6	2.5	66.2	1.6
	GMM	27.0	2.1	235.4	2.2	53.6	1.9	32.2	2.1	180.8	2.5	62.7	1.7
β-VAE	$\mathcal{N}$	21.4	2.1	115.4	3.6	56.1	1.9	19.1	2.0	124.9	3.4	65.9	1.6
	GMM	9.2	2.2	92.2	3.9	51.7	1.9	11.4	2.1	112.6	3.6	59.3	1.7
β-TC VAE	$\mathcal{N}$	21.3	2.1	116.6	2.8	55.7	1.8	20.7	2.0	125.8	3.4	65.9	1.6
	GMM	11.6	2.2	89.3	4.1	51.8	1.9	13.3	2.1	**106.5**	**3.7**	59.3	1.7
FactorVAE	$\mathcal{N}$	27.0	2.1	236.5	2.2	53.8	1.9	31.0	2.0	185.4	2.5	66.4	1.7
	GMM	26.9	2.1	234.0	2.2	52.4	2.0	32.7	2.1	184.4	2.5	63.3	1.7
InfoVAE - RBF	$\mathcal{N}$	27.5	2.1	235.2	2.1	55.5	1.9	31.1	2.0	182.8	2.5	66.5	1.6
	GMM	26.7	2.1	230.4	2.2	52.7	1.9	32.3	2.1	179.5	2.5	62.8	1.7
InfoVAE - IMQ	$\mathcal{N}$	28.3	2.1	233.8	2.2	56.7	1.9	31.0	2.0	182.4	2.5	66.4	1.6
	GMM	27.7	2.1	231.9	2.2	53.7	1.9	32.8	2.1	180.7	2.6	62.3	1.7
AAE	$\mathcal{N}$	16.8	2.2	139.9	2.6	59.9	1.8	19.1	2.1	164.9	2.4	64.8	1.7
	GMM	9.3	2.2	92.1	3.8	53.9	2.0	11.1	2.1	118.5	3.5	58.7	1.8
MSSSIM-VAE	$\mathcal{N}$	26.7	2.2	279.9	1.7	124.3	1.3	28.0	2.1	254.2	1.7	119.0	1.3
	GMM	27.2	2.2	279.7	1.7	124.3	1.3	28.8	2.1	253.1	1.7	119.2	1.3
VAEGAN	$\mathcal{N}$	8.7	2.2	199.5	2.2	39.7	1.9	12.8	2.2	198.7	2.2	122.8	2.0
	GMM	**6.3**	**2.2**	197.5	2.1	**35.6**	1.8	**6.5**	2.2	188.2	2.6	84.3	1.7
AE	$\mathcal{N}$	26.7	2.1	201.3	2.1	327.7	1.0	221.8	1.3	210.1	2.1	275.0	2.9
	GMM	9.3	2.2	97.3	3.6	55.4	2.0	11.0	2.1	120.7	3.4	**57.4**	1.8
WAE - RBF	$\mathcal{N}$	21.2	2.2	175.1	2.0	332.6	1.0	21.2	2.1	170.2	2.3	69.4	1.6
	GMM	9.2	2.2	97.1	3.6	55.0	2.0	11.2	2.1	120.3	3.4	58.3	1.7
WAE - IMQ	$\mathcal{N}$	18.9	2.2	164.4	2.2	64.6	1.7	20.3	2.1	150.7	2.5	67.1	1.6
	GMM	8.6	2.2	96.5	3.6	51.7	2.0	11.2	2.1	119.0	3.5	57.7	1.8
VQVAE	$\mathcal{N}$	28.2	2.0	152.2	2.0	306.9	1.0	170.7	1.6	195.7	1.9	140.3	**2.2**
	GMM	9.1	2.2	95.2	3.7	51.6	2.0	10.7	2.1	120.1	3.4	57.9	1.8
RAE-L2	$\mathcal{N}$	25.0	2.0	156.1	2.6	86.1	2.8	63.3	2.2	170.9	2.2	168.7	3.1
	GMM	9.1	2.2	**85.3**	**3.9**	55.2	1.9	11.5	2.1	122.5	3.4	58.3	1.8
RAE - GP	$\mathcal{N}$	27.1	2.1	196.8	2.1	86.1	**2.4**	61.5	2.2	229.1	2.0	201.9	3.1
	GMM	9.7	2.2	96.3	3.7	52.5	1.9	11.4	2.1	123.3	3.4	59.0	1.8

4.5 Conclusion

In this chapter, we introduce **Pythae**, a new open-source Python library unifying common and state-of-the-art Generative AutoEncoder (GAE) implementations, allowing reliable and reproducible model training, data generation and experiment tracking. This library was designed as an open model testing environment driven by the community, wherein peers are encouraged to contribute by adding their own models, and by doing so favour reproducible research and accessibility to ready-to-use GAE models. As an illustration of the capabilities of Pythae, we perform a benchmarking of 19 generative autoencoder models on 5 downstream tasks (image reconstruction, generation, classification, clustering and interpolation) leading to some interesting findings on the general behaviours of generative autoencoder models. We hope that the library will continue to be adopted by the community and expand thanks to the increasing number of contributions.

Acknowledgment

The research leading to these results has received funding from the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and reference ANR-10-IAIHU-06 (Agence Nationale de la Recherche-10-IA Institut Hospitalo-Universitaire-6). This work was granted access to the HPC resources of IDRIS under the allocation AD011013517 made by GENCI (Grand Equipement National de Calcul Intensif).

4.6 Appendices

4.6.1 Usage of Pythae

In this section we illustrate through simple examples how to use **Pythae** pipelines. The library is documented[11] and also available on pypi[12] allowing a wider use and easier integration in other codes. All of the implementations proposed in the library are adaptations of the official code when available and allowed by the licence. If not, the method is re-implemented. Table 4.4 lists all the implemented models as of June 2022.

1. **Training configuration** Before launching a model training, one must specify the training configuration that should be used. This can be done easily by instantiating a **BaseTrainerConfig** instance taking as input all the hyper-parameters related to the training (number of training epochs, learning rate to apply...). See the full documentation for additional arguments that can be passed to the **BaseTrainerConfig**.

```
from pythae.trainers import BaseTrainerConfig
# Set up training configuration
my_training_config = BaseTrainerConfig(
  output_dir='my_model',
  num_epochs=50,
  learning_rate=1e-3,
  per_device_train_batch_size=200,
  per_device_eval_batch_size=200,
  train_dataloader_num_workers=2,
  eval_dataloader_num_workers=2,
  steps_saving=20,
  optimizer_cls="AdamW",
  optimizer_params={"weight_decay": 0.05, "betas": (0.91, 0.995)},
  scheduler_cls="ReduceLROnPlateau",
  scheduler_params={"patience": 5, "factor": 0.5} )
```

[11] https://pythae.readthedocs.io/en/latest/?badge=latest
[12] https://pypi.org/project/pythae/

2. **Model configuration** Similarly to the TrainerConfig, the model can then be instantiated with the model configuration specifying any hyper-parameters relevant to the model. Note that each model has its own configuration with specific hyper-parameters. See the online documentation for more details.

```
from pythae.models import VAE, VAEConfig
my_vae_config = model_config = VAEConfig(
  input_dim=(1, 28, 28),
  latent_dim=10
)
# Build the model
my_vae_model = VAE(
  model_config=my_vae_config
)
```

3. **Training** A model training can then be launched by simply using the built-in training pipeline in which only the training/evaluation data (torch.Tensor, np.array or torch datasets) need to be specified.

```
from pythae.pipelines import TrainingPipeline
# Build the Pipeline
pipeline = TrainingPipeline(
  training_config=my_training_config,
  model=my_vae_model
)
# Launch the Pipeline
pipeline(
    train_data=your_train_data,
    eval_data=your_eval_data
)
```

4. **Model reloading** The weights and configuration of the trained model can be reloaded using the **AutoModel** instance proposed in Pythae.

```
from pythae.models import AutoModel
my_trained_vae = AutoModel.load_from_folder(
  'path/to/trained_model'
)
```

5. **Data generation** A data generation pipeline can be instantiated similarly to a model training. The pipeline can then be called with any relevant arguments such as the number of samples to generate or the training and evaluation data that may be needed to fit the sampler.

```python
from pythae.samplers import GaussianMixtureSamplerConfig
from pythae.pipelines import GenerationPipeline
# Define your sampler configuration
gmm_sampler_config = GaussianMixtureSamplerConfig(
    n_components=10
)
# Build the pipeline
pipeline = GenerationPipeline(
    model=my_trained_vae,
    sampler_config=gmm_sampler_config
)
# Launch generation
generated_samples = pipeline(
    num_samples=100,
    return_gen=True,
    train_data=train_data,
    eval_data=None
)
```

Table 4.4: List of implemented VAEs

Name	Reference
Variational Autoencoder (VAE)	(Kingma and Welling, 2014)
Beta Variational Autoencoder (BetaVAE)	(Higgins et al., 2017)
VAE with Linear Normalizing Flows (VAE_LinNF)	(Rezende and Mohamed, 2015)
VAE with Inverse Autoregressive Flows (VAE_IAF)	(Kingma et al., 2016)
Disentangled β-VAE (DisentangledBetaVAE)	(Higgins et al., 2017)
Disentangling by Factorising (FactorVAE)	(Kim and Mnih, 2018)
Beta-TC-VAE (BetaTCVAE)	(Chen et al., 2018b)
Importance Weighted Autoencoder (IWAE)	(Burda et al., 2016)
VAE with perceptual metric similarity (MSSSIM_VAE)	(Snell et al., 2017)
Wasserstein Autoencoder (WAE)	(Tolstikhin et al., 2018)
Info Variational Autoencoder (INFOVAE_MMD)	(Zhao et al., 2019)
VAMP Autoencoder (VAMP)	(Tomczak and Welling, 2018)
Hyperspherical VAE (SVAE)	(Davidson et al., 2018)
Adversarial Autoencoder (Adversarial_AE)	(Makhzani et al., 2015)
Variational Autoencoder GAN (VAEGAN)	(Larsen et al., 2016)
Vector Quantized VAE (VQVAE)	(Van Den Oord et al., 2017)
Hamiltonian VAE (HVAE)	(Caterini et al., 2018)
Regularized AE with L2 decoder param (RAE_L2)	(Ghosh et al., 2020)
Regularized AE with gradient penalty (RAE_GP)	(Ghosh et al., 2020)
Riemannian Hamiltonian VAE (RHVAE)	(Chadebec et al., 2022b)

Maintenance plan: We intend for this library to be maintained in the long term. In that view, the main author's contact details will remain available and up-to-date on the github repository, which will remain the main discussion channel. Additionally, we are currently considering adding back-up contributors that will also support this effort in the long-term. Since this library has already started to be a community effort with external contributors, we further hope that the community will also continue to help reviewing and updating the current implementations.

Original papers reproducibility We validate the implementations by reproducing some results presented in the original publications when the official code has been released or when enough details about the experimental section of the papers were available (we indeed noted that in many papers key elements for reproducibility were missing such as the data split considered, which criteria is used to select the model on which the metrics are computed, the hyper-parameters are not fully disclosed or the network architectures is unclear making reproduction very hard if not impossible in certain cases). This insists on the fact that the framework is flexible enough to reproduce results from publications. Finally, we have open-sourced the scripts, configurations and results on the repository at this link and made the trained models available on the HuggingFace Hub (e.g. trained iwae).

4.6.2 Interpolations

In this section, we show the interpolations obtained on the three considered datasets. For each model, we select both a starting image and an ending image from the test set and perform a linear interpolation between the corresponding embeddings in the learned latent space. We then show the decoded trajectory all along the interpolation line. For this task, we use the model configuration that obtained the lowest FID on the validation set with a GMM sampler from the generation task. We show the resulting interpolations for latent spaces of dimension 16 and 256 for MNIST, 32 and 256 for CIFAR10 and 64 for CELEBA. As mentioned in the chapter, for this complex task, variational approaches tend to outperform the AE-based methods. This is well illustrated on MNIST with a latent space of dimension 256 since all the AE-based approaches eventually superpose the starting and ending image, making the interpolation visually irrelevant. Impressively, the regularization imposed by the variational approaches prevents such undesirable behaviours from occurring. This adds to the observation made in Sec. 4.4.2 of the chapter where we note some robustness to the latent dimension for the variational methods. Nonetheless, as stated in the chapter this regularization can also degrade image reconstruction, leading to very blurry interpolations, as illustrated on Fig. 4.5.

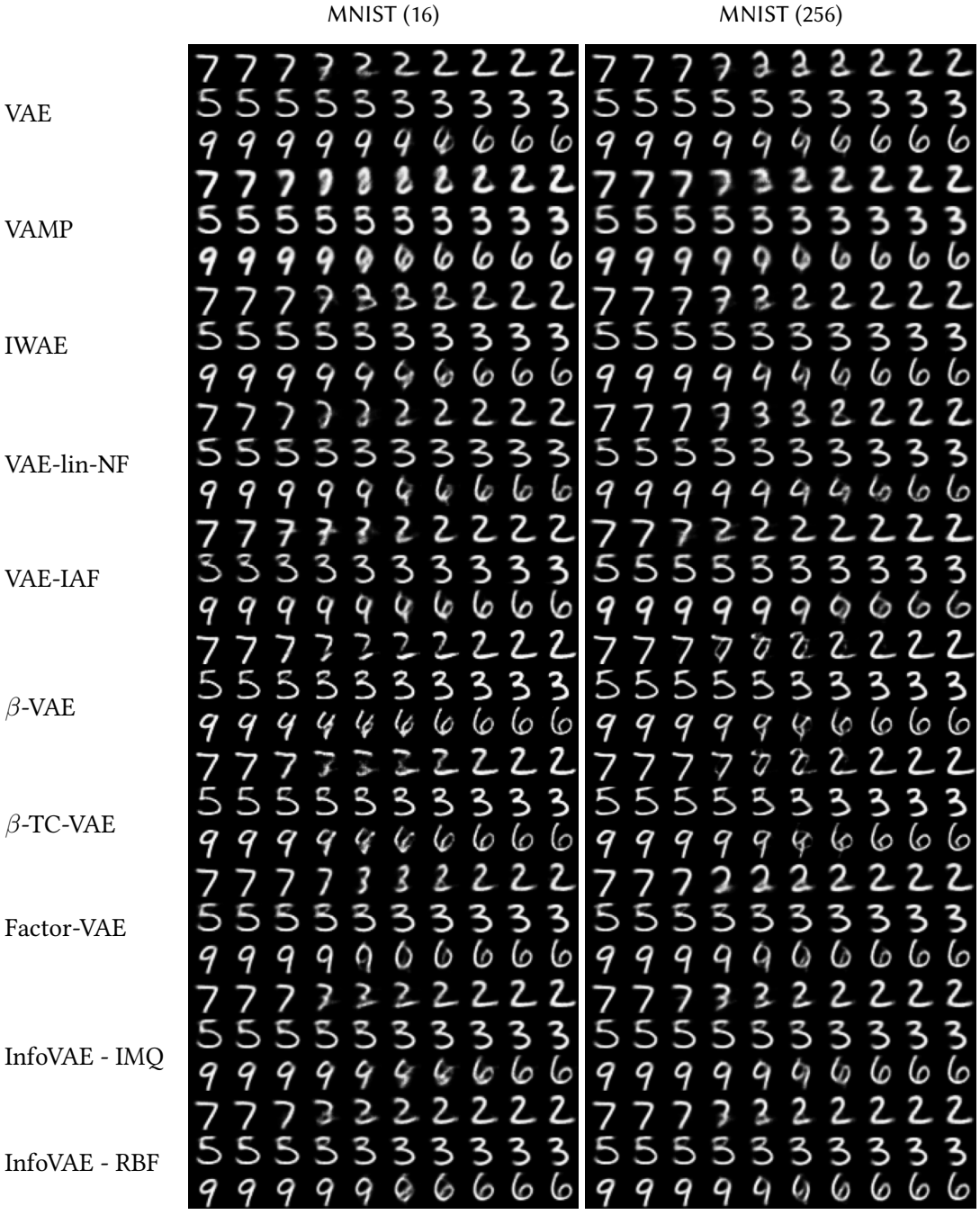


Figure 4.3: Interpolations on MNIST with the same starting and ending images for latent spaces of dimension 16 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

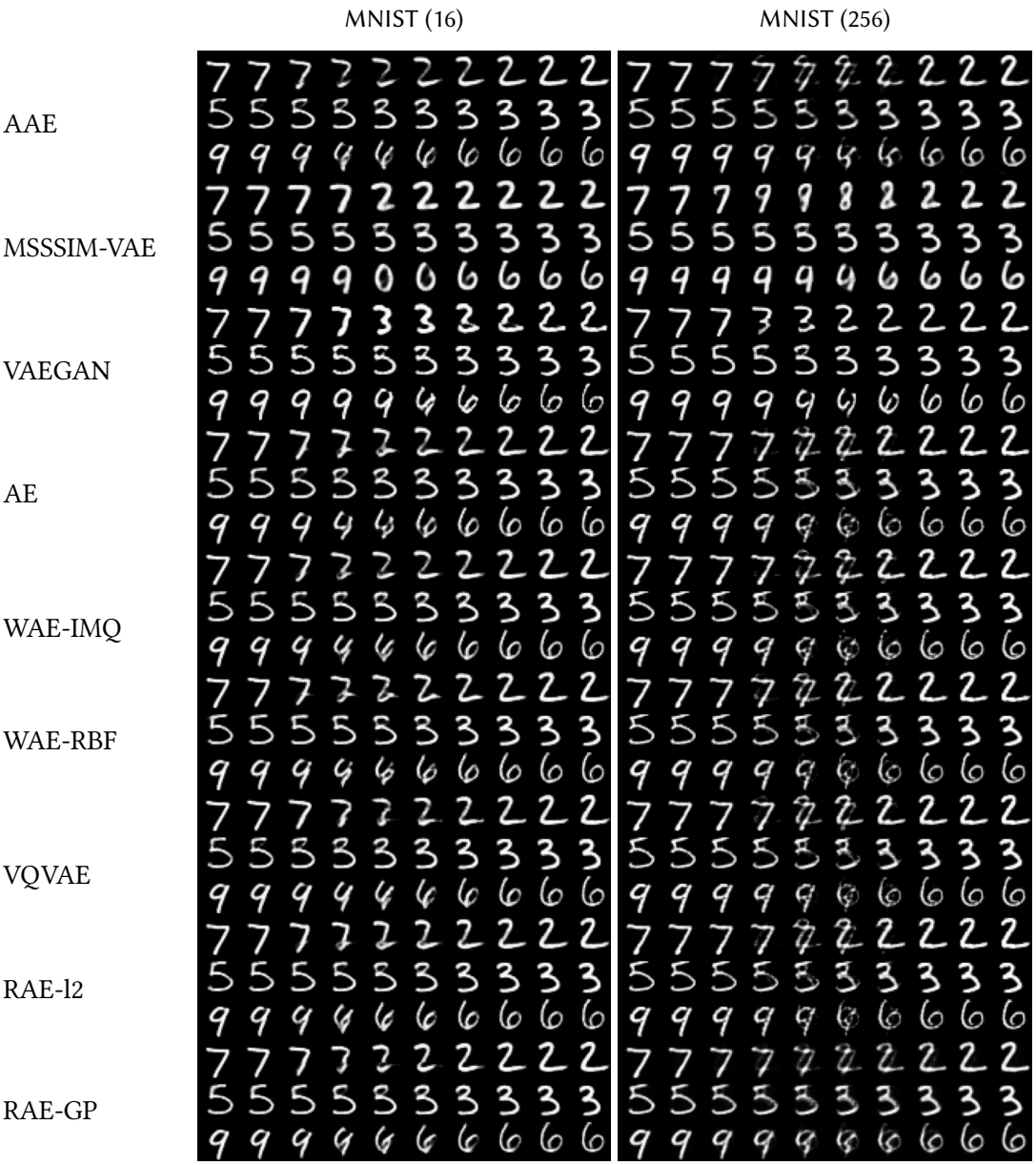


Figure 4.4: Interpolations on MNIST with the same starting and ending images for latent spaces of dimension 16 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

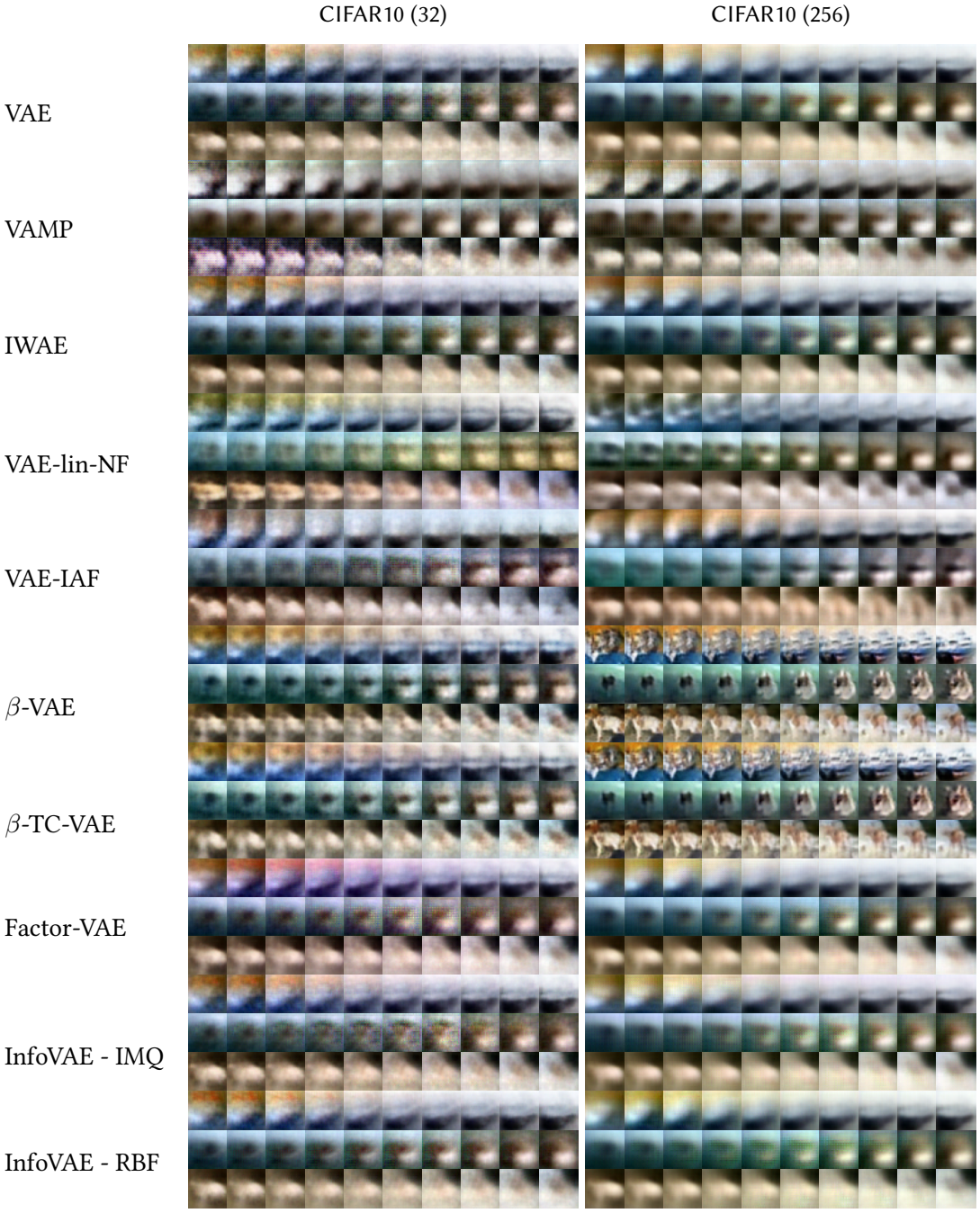


Figure 4.5: Interpolations on CIFAR10 with the same starting and ending images for latent spaces of dimension 32 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

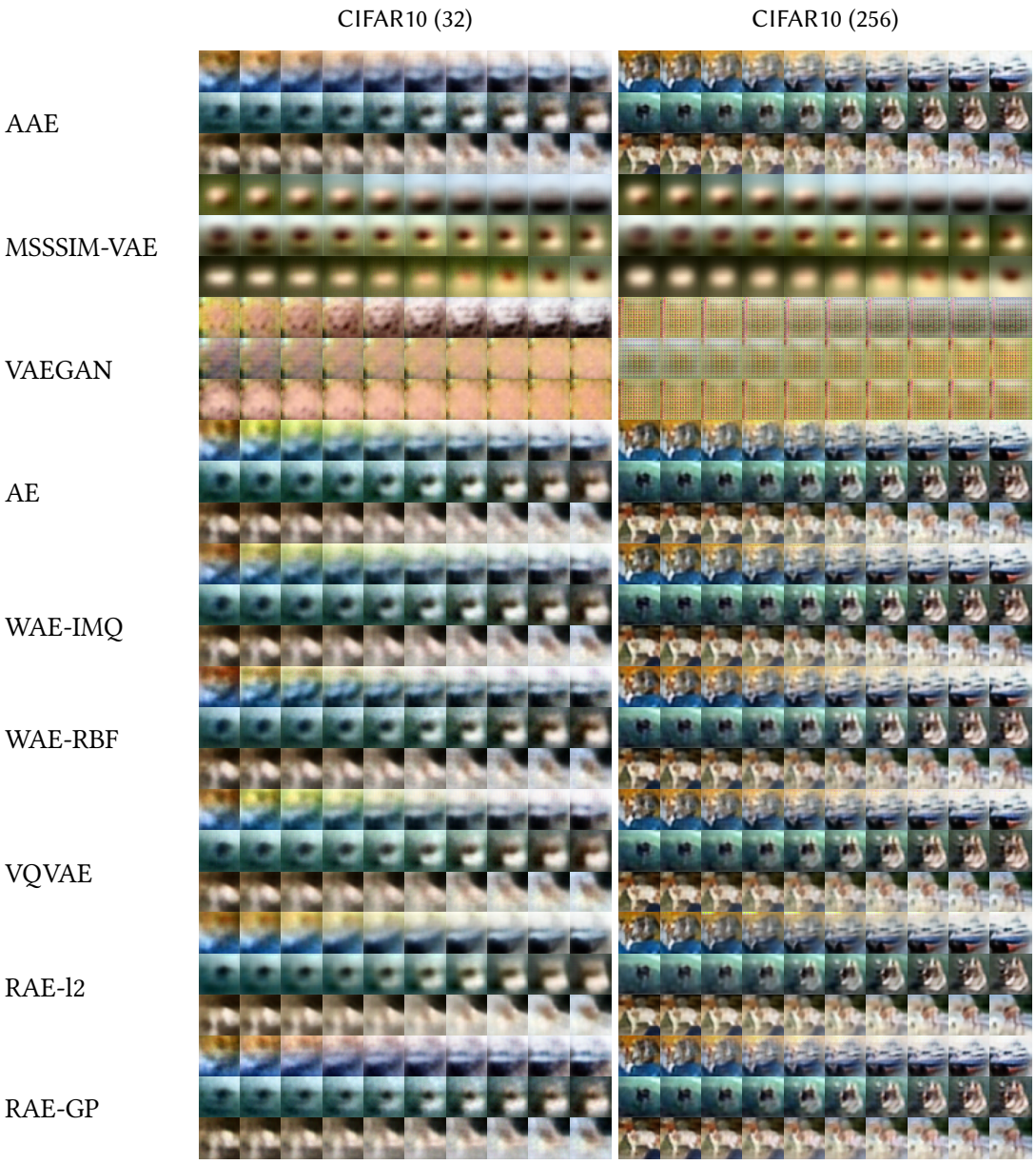


Figure 4.6: Interpolations on CIFAR10 with the same starting and ending images for latent spaces of dimension 32 and 256. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

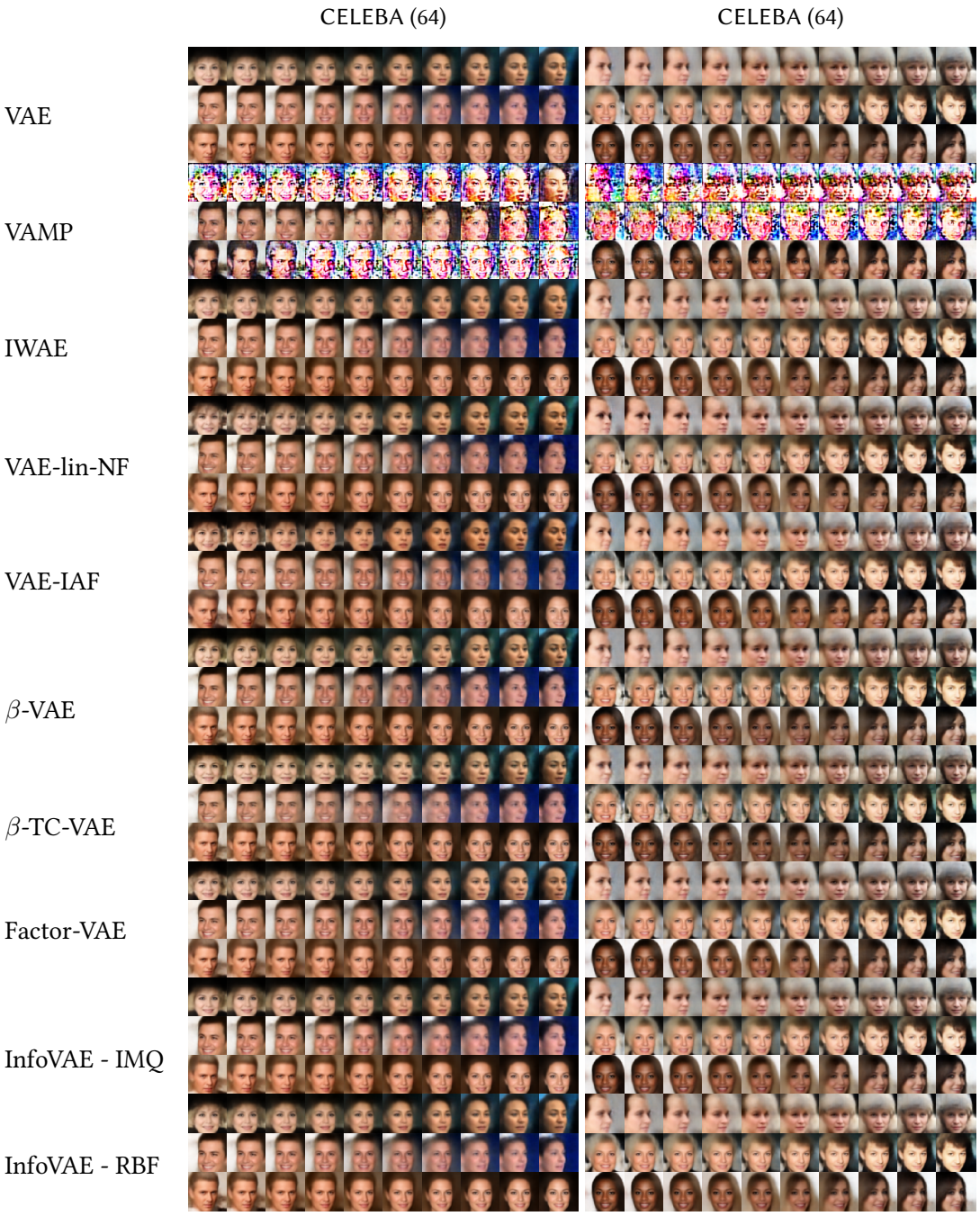


Figure 4.7: Interpolations on CELEBA with the same starting and ending images for a latent space of dimension 64. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

Figure 4.8: Interpolations on CELEBA with the same starting and ending images for a latent space of dimension 64. For each model we select the configuration achieving the lowest FID on the generation task on the validation set with a GMM sampler.

4.6.3 Detailed Experiments Set-Up

We detail here the main experimental set-up and implementation choices made in the benchmark. We let the reader refer to the code available online for specific implementation aspects

The data To perform the benchmaks presented in the chapter, we select 3 classical *free-to-use* image datasets: MNIST (LeCun, 1998), CIFAR10 (Krizhevsky et al., 2009) and CELEBA (Liu et al., 2015). These datasets are publicly available, widely used for generative model related papers and have well known associated metrics in the literature. Each dataset is split into a train set, a validation set and a test set. For MNIST and CIFAR10 the validation set is composed of the last 10k images extracted from the official train set and the test set corresponds to the official one. For CELEBA, we use the official train/val/test split.

Training paradigm We equip each model used in the benchmark with the same neural network architecture for both the encoder and decoder, taken as a ConvNet and ResNet (architectures given in Tables 4.5 and 4.6) leading to a comparable number of parameters [13]. For the 19 considered models, due to computational limitations, 10 different configurations are considered, allowing a simple exploration of the models' hyper-parameters. The sets of hyper-parameters explored are detailed in Appendix 4.6.4 for each model. The models are then trained on MNIST and CIFAR10 for 100 epochs, a starting learning rate of $1e^{-4}$ and batch size of 100 with Adam optimizer (Kingma and Ba, 2014). A scheduler reducing the learning rate by half if the validation loss does not improve for 10 epochs is also used. For CELEBA, we use the same setting but we train the models for 50 epochs with a starting learning rate of $1e^{-3}$. Models with unstable training (NaN, huge training spikes...) are iteratively retrained with a starting learning rate divided by 10 until training stabilises. All 19 models are trained on a single 32GB V100 GPU. This leads to 10 trained models for each method, each dataset (MNIST, CIFAR10 or CELEBA) and each neural network (ConvNet or ResNet) leading to a total of 1140 models. The training setting (curves, configs ...) can be found at https://wandb.ai/benchmark_team/trainings.

Sampling paradigm for the MAF and VAE samplers For the Masked Autoregressive Flow sampler used for sampling we use a 3-layer MADE (Germain et al., 2015) with 128 hidden units and ReLU activation for each layer and stack 2 blocks of MAF to create the flow. For the masked layers, the mask is made sequentially and the ordering is reversed between each MADE. For this normalizing flow we consider a starting distribution given by a standard Gaussian. For the auxiliary VAE sampling method proposed in (Dai and Wipf, 2018), we consider a simple VAE with a Multi Layer Perceptron (MLP) encoder and decoder, with 2 hidden layers composed of 1024 units and ReLU activation. Both samplers are fitted with 200 epochs using the train and evaluation embeddings coming from the trained autoencoder models. A learning rate of $1e^{-4}$, a scheduler decreasing the learning rate by half if the validation loss does not improve for 10 epochs and a batch size of 100 are used for these samplers.

[13] Some models may actually have additional parameters in their intrinsic structure *e.g.* a VQVAE learns a dictionary of embeddings, a VAMP learns the pseudo-inputs, a VAE-IAF learns the auto-regressive flows. Nonetheless, since we work on images, the number of parameters remains in the same order of magnitude.

Table 4.5: Neural network architecture used for the convolutional networks.

	MNIST	CIFAR10	CELEBA
Encoder	$(1, 28, 28)$	$(3, 32, 32)$	$(3, 64, 64)$
Layer 1	Conv$(128, 4, 2)$, BN, ReLU	Conv$(128, 4, 2)$, BN, ReLU	Conv$(128, 4, 2)$, BN, ReLU
Layer 2	Conv$(256, 4, 2)$, BN, ReLU	Conv$(256, 4, 2)$, BN, ReLU	Conv$(256, 4, 2)$, BN, ReLU
Layer 3	Conv$(512, 4, 2)$, BN, ReLU	Conv$(512, 4, 2)$, BN, ReLU	Conv$(512, 4, 2)$, BN, ReLU
Layer 4	Conv$(1024, 4, 2)$, BN, ReLU	Conv$(1024, 4, 2)$, BN, ReLU	Conv$(1024, 4, 2)$, BN, ReLU
Layer 5	Linear$(1024,$ latent_dim$)^*$	Linear$(4096,$ latent_dim$)^*$	Linear$(16384,$ latent_dim$)^*$
Decoder			
Layer 1	Linear(latent_dim, 16384)	Linear(latent_dim, 65536)	Linear(latent_dim, 65536)
Layer 2	ConvT$(512, 3, 2)$, BN, ReLU	ConvT$(512, 4, 2)$, BN, ReLU	ConvT$(512, 5, 2)$, BN, ReLU
Layer 3	ConvT$(256, 3, 2)$, BN, ReLU	ConvT$(256, 4, 2)$, BN, ReLU	ConvT$(256, 5, 2)$, BN, ReLU
Layer 4	Conv$(1, 3, 2)$, Sigmoid	Conv$(3, 4, 1)$, Sigmoid	ConvT$(128, 5, 2)$, BN, ReLU
Layer 5	-	-	ConvT$(3, 5, 1)$, Sigmoid

*Doubled for VAE-based models

Table 4.6: Neural network architecture used for the residual networks.

	MNIST	CIFAR10	CELEBA
Encoder	$(1, 28, 28)$	$(3, 32, 32)$	$(3, 64, 64)$
Layer 1	Conv$(64, 4, 2)$	Conv$(64, 4, 2)$	Conv$(64, 4, 2)$
Layer 2	Conv$(128, 4, 2)$	Conv$(128, 4, 2)$	Conv$(128, 4, 2)$
Layer 3	Conv$(128, 3, 2)$	Conv$(128, 3, 1)$	Conv$(128, 3, 2)$
Layer 4	ResBlock**	ResBlock**	Conv$(128, 3, 2)$
Layer 5	ResBlock**	ResBlock**	ResBlock**
Layer 6	Linear$(2048,$ latent_dim$)^*$	Linear$(8192,$ latent_dim$)^*$	ResBlock**
Layer 7	-	-	Linear$(2048,$ latent_dim$)^*$
Decoder			
Layer 1	Linear(latent_dim, 2048)	Linear(latent_dim, 8192)	Linear(latent_dim, 2048)
Layer 2	ConvT$(128, 3, 2)$	ResBlock**	ConvT$(128, 3, 2)$
Layer 3	ResBlock**	ResBlock**	ResBlock**
Layer 4	ResBlock**, ReLU	ConvT$(64, 4, 2)$	ResBlock**
Layer 5	ConvT$(64, 3, 2)$, ReLU	ConvT$(3, 4, 2)$, Sigmoid	ConvT$(128, 5, 2)$, Sigmoid
Layer 6	ConvT$(1, 3, 2)$, Sigmoid	-	ConvT$(64, 5, 2)$, Sigmoid
Layer 6	-	-	ConvT$(3, 4, 2)$, Sigmoid

*Doubled for VAE-based models

**The ResBlocks are composed of one Conv$(32, 3, 1)$ followed by Conv$(128, 1, 1)$ with ReLU.

4.6.4 Additional Results

Effect of the Latent Dimension on the 4 Tasks with the CIFAR10 Database

Analogously to the results shown in the chapter on the MNIST dataset for the 4 chosen tasks (reconstruction, generation, classification and clustering), Fig. 4.9 shows the impact the choice of the latent space dimension has on the performances of the models on the CIFAR10 dataset, whose image arguably have a greater intrinsic latent dimension than images of the MNIST dataset. Similarly to MNIST, two distinct groups appear: the AE-based methods and variational methods. Again, for all tasks but clustering, variational based methods demonstrate good robustness properties with respect to the dimension of the latent space when compared to AE approaches.

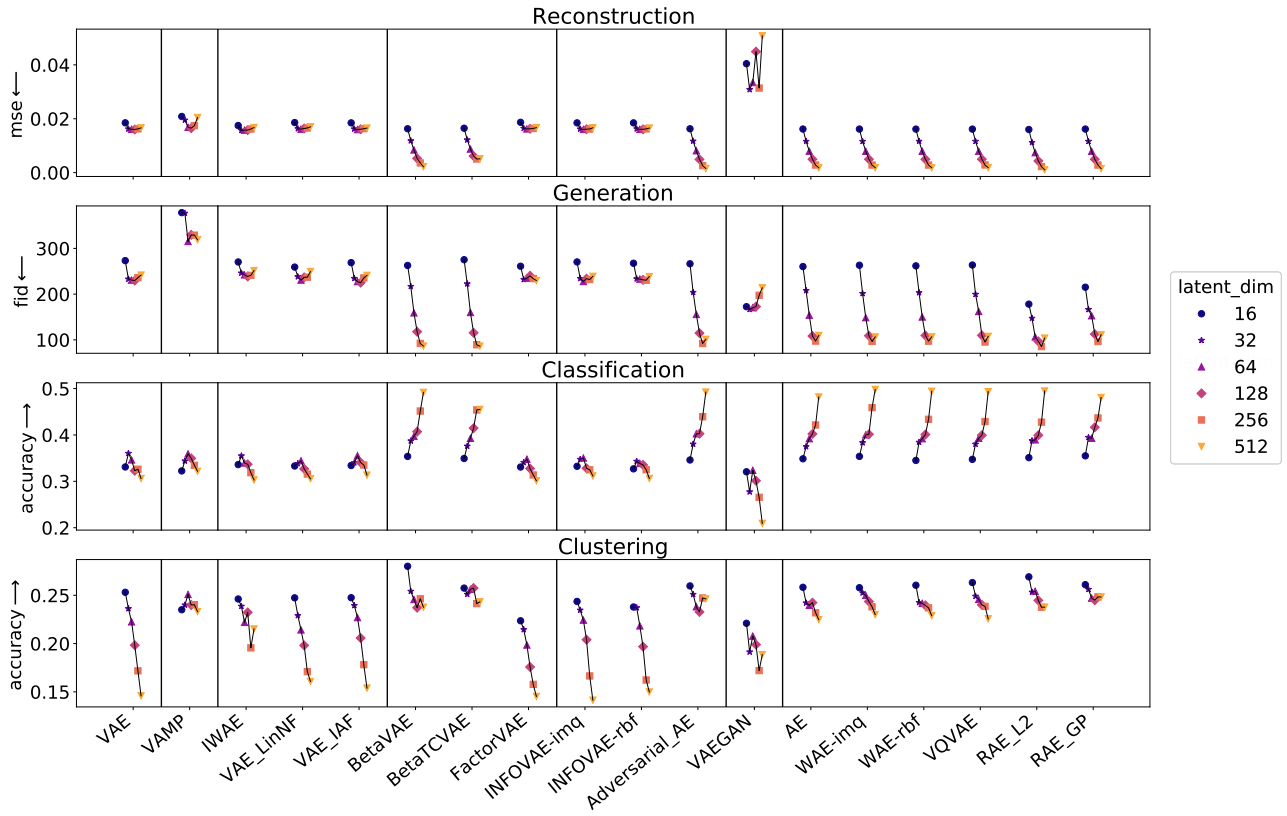


Figure 4.9: *From top to bottom:* Evolution of the reconstruction MSE, generation FID, classification accuracy and clustering accuracy with respect to the latent space dimension on the CIFAR dataset.

Complete Generation Table

In Table 4.7 are presented the full results obtained for generation *i.e.* including the MAF and 2-stage VAE sampler (Dai and Wipf, 2018). As mentioned in the chapter, it is interesting to note that fitting a GMM instead of using the prior for the variational-based approaches seems to often allow a better image generation since it allows a better prospecting of the learned latent space of each model. Interestingly, it seems that fitting more complex density estimators such as a normalizing

flow (MAF sampler) or another VAE (2-stage sampler) does not improve the generation results when compared to the GMM for those datasets.

Table 4.7: Inception Score (higher is better) and FID (lower is better) computed with 10k samples on the test set. For each model and sampler we report the results obtained by the model achieving the lowest FID score on the validation set.

Model	Sampler	ConvNet						ResNet					
		MNIST		CIFAR10		CELEBA		MNIST		CIFAR10		CELEBA	
		FID ↓	IS ↑	FID	IS	FID	IS ↑	FID ↓	IS ↑	FID	IS	FID	IS
VAE	$\mathcal{N}$	28.5	2.1	241.0	2.2	54.8	1.9	31.3	2.0	181.7	2.5	66.6	1.6
	GMM	26.9	2.1	235.9	2.3	52.4	1.9	32.3	2.1	179.7	2.5	63.0	1.7
	VAE	40.3	2.0	337.5	1.7	70.9	1.6	48.7	1.8	358.0	1.3	76.4	1.4
	MAF	26.8	2.1	239.5	2.2	52.5	2.0	31.0	2.1	181.5	2.5	62.9	1.7
VAMP	VAMP	64.2	2.0	329.0	1.5	56.0	1.9	34.5	2.1	181.9	2.5	67.2	1.6
IWAE	$\mathcal{N}$	29.0	2.1	245.3	2.1	55.7	1.9	32.4	2.0	191.2	2.4	67.6	1.6
	GMM	28.4	2.1	241.2	2.1	52.7	1.9	34.4	2.1	188.8	2.4	64.1	1.7
	VAE	42.4	2.0	346.6	1.5	74.3	1.5	50.1	1.9	364.8	1.2	76.4	1.4
	MAF	28.1	2.1	243.4	2.1	52.7	1.9	32.5	2.1	190.4	2.4	64.3	1.7
VAE-lin NF	$\mathcal{N}$	29.3	2.1	240.3	2.1	56.5	1.9	32.5	2.0	185.5	2.4	67.1	1.6
	GMM	28.4	2.1	237.0	2.2	53.3	1.9	33.1	2.1	183.1	2.5	62.8	1.7
	VAE	40.1	2.0	311.0	1.6	71.1	1.6	49.7	1.9	296.2	1.7	75.6	1.4
	MAF	27.7	2.1	239.1	2.1	53.4	2.0	32.4	2.0	184.2	2.5	62.7	1.7
VAE-IAF	$\mathcal{N}$	27.5	2.1	236.0	2.2	55.4	1.9	30.6	2.0	183.6	2.5	66.2	1.6
	GMM	27.0	2.1	235.4	2.2	53.6	1.9	32.2	2.1	180.8	2.5	62.7	1.7
	VAE	39.4	2.0	330.5	1.1	73.0	1.5	44.8	1.9	322.7	1.5	76.7	1.4
	MAF	26.9	2.1	236.8	2.2	53.6	1.9	30.6	2.1	182.5	2.5	63.0	1.7
β-VAE	$\mathcal{N}$	21.4	2.1	115.4	3.6	56.1	1.9	19.1	2.0	124.9	3.4	65.9	1.6
	GMM	9.2	2.2	92.2	3.9	51.7	1.9	11.4	2.1	112.6	3.6	59.3	1.7
	VAE	14.0	2.2	139.6	3.6	55.0	1.9	20.3	2.1	152.5	3.5	61.5	1.7
	MAF	9.5	2.2	100.9	3.5	51.5	2.0	12.0	2.1	120.0	3.6	59.7	1.8
β-TC VAE	$\mathcal{N}$	21.3	2.1	116.6	2.8	55.7	1.8	20.7	2.0	125.8	3.4	65.9	1.6
	GMM	11.6	2.2	89.3	4.1	51.8	1.9	13.3	2.1	**106.5**	**3.7**	59.3	1.7
	VAE	18.4	2.2	127.9	4.2	59.7	1.8	28.3	2.0	164.0	3.3	66.4	1.5
	MAF	12.0	2.2	95.6	3.6	52.2	1.9	13.7	2.1	116.6	3.4	60.1	1.7
FactorVAE	$\mathcal{N}$	27.0	2.1	236.5	2.2	53.8	1.9	31.0	2.0	185.4	2.5	66.4	1.7
	GMM	26.9	2.1	234.0	2.2	52.4	2.0	32.7	2.1	184.4	2.5	63.3	1.7
	VAE	41.2	1.9	338.3	1.5	75.0	1.5	54.7	1.8	316.2	1.3	77.7	1.4
	MAF	26.7	2.2	236.7	2.2	52.7	1.9	32.8	2.1	185.8	2.5	63.4	1.7
InfoVAE - RBF	$\mathcal{N}$	27.5	2.1	235.2	2.1	55.5	1.9	31.1	2.0	182.8	2.5	66.5	1.6
	GMM	26.7	2.1	230.4	2.2	52.7	1.9	32.3	2.1	179.5	2.5	62.8	1.7
	VAE	39.7	2.0	327.2	1.5	73.7	1.5	50.6	1.9	363.4	1.2	75.8	1.4
	MAF	25.9	2.1	233.5	2.2	52.2	2.0	30.5	2.1	181.3	2.5	62.7	1.7
InfoVAE - IMQ	$\mathcal{N}$	28.3	2.1	233.8	2.2	56.7	1.9	31.0	2.0	182.4	2.5	66.4	1.6
	GMM	27.7	2.1	231.9	2.2	53.7	1.9	32.8	2.1	180.7	2.6	62.3	1.7
	VAE	40.4	1.9	323.8	1.6	73.7	1.5	49.9	1.9	341.8	1.8	75.7	1.4
	MAF	27.2	2.1	232.3	2.1	53.8	2.0	30.6	2.1	182.5	2.5	62.6	1.7
AAE	$\mathcal{N}$	16.8	2.2	139.9	2.6	59.9	1.8	19.1	2.1	164.9	2.4	64.8	1.7
	GMM	9.3	2.2	92.1	3.8	53.9	2.0	11.1	2.1	118.5	3.5	58.7	1.8
	VAE	13.4	2.2	144.0	3.4	58.2	1.8	15.1	2.1	145.2	3.6	59.0	1.7
	MAF	9.3	2.2	101.1	3.2	53.8	2.0	11.9	2.1	133.6	3.1	59.2	1.8
MSSSIM-VAE	$\mathcal{N}$	26.7	2.2	279.9	1.7	124.3	1.3	28.0	2.1	254.2	1.7	119.0	1.3
	GMM	27.2	2.2	279.7	1.7	124.3	1.3	28.8	2.1	253.1	1.7	119.2	1.3
	VAE	51.2	1.9	355.5	1.1	137.9	1.2	51.6	1.9	372.1	1.1	136.5	1.2
	MAF	26.9	2.2	279.8	1.7	124.0	1.3	27.5	2.1	254.1	1.7	119.5	1.3
VAEGAN	$\mathcal{N}$	8.7	2.2	199.5	2.2	39.7	1.9	12.8	2.2	198.7	2.2	122.8	2.0
	GMM	**6.3**	**2.2**	197.5	2.1	**35.6**	1.8	**6.5**	2.2	188.2	2.6	84.3	1.7
	VAE	11.2	2.1	310.9	2.0	54.5	1.6	9.2	2.1	272.7	2.0	88.8	1.6
	MAF	6.9	2.3	199.0	2.1	36.7	1.8	6.6	**2.2**	191.9	2.5	84.8	1.7
AE	$\mathcal{N}$	26.7	2.1	201.3	2.1	327.7	1.0	221.8	1.3	210.1	2.1	275.0	2.9
	GMM	9.3	2.2	97.3	3.6	55.4	2.0	11.0	2.1	120.7	3.4	**57.4**	1.8
	MAF	9.9	2.2	108.3	3.1	55.7	2.0	12.0	2.1	136.5	3.0	58.3	1.8
WAE - RBF	$\mathcal{N}$	21.2	2.2	175.1	2.0	332.6	1.0	21.2	2.1	170.2	2.3	69.4	1.6
	GMM	9.2	2.2	97.1	3.6	55.0	2.0	11.2	2.1	120.3	3.4	58.3	1.7
	MAF	9.8	2.2	108.2	3.1	56.0	2.0	11.8	2.2	135.3	3.0	58.3	1.8
WAE - IMQ	$\mathcal{N}$	18.9	2.2	164.4	2.2	64.6	1.7	20.3	2.1	150.7	2.5	67.1	1.6
	GMM	8.6	2.2	96.5	3.6	51.7	2.0	11.2	2.1	119.0	3.5	57.7	1.8
	MAF	9.5	2.2	107.8	3.1	51.6	2.0	11.8	2.1	130.2	3.0	58.7	1.7
VQVAE	$\mathcal{N}(0,1)$	28.2	2.0	152.2	2.0	306.9	1.0	170.7	1.6	195.7	1.9	140.3	**2.2**
	GMM	9.1	2.2	95.2	3.7	51.6	2.0	10.7	2.1	120.1	3.4	57.9	1.8
	MAF	9.6	2.2	104.7	3.2	52.3	1.9	11.7	2.2	136.8	3.0	57.9	1.8
RAE-L2	$\mathcal{N}$	25.0	2.0	156.1	2.6	86.1	**2.8**	63.3	2.2	170.9	2.2	168.7	3.1
	GMM	9.1	2.2	**85.3**	**3.9**	55.2	1.9	11.5	2.1	122.5	3.4	58.3	1.8
	MAF	9.5	2.2	93.4	3.5	55.2	2.0	12.3	2.2	136.6	3.0	59.1	1.7
RAE - GP	$\mathcal{N}$	27.1	2.1	196.8	2.1	86.1	2.4	61.5	2.2	229.1	2.0	201.9	3.1
	GMM	9.7	2.2	96.3	3.7	52.5	1.9	11.4	2.1	123.3	3.4	59.0	1.8
	MAF	9.7	2.2	106.3	3.2	52.5	1.9	12.2	2.2	139.4	3.0	59.5	1.8

Further Interesting Results

Generated samples In addition to quantitative metrics, we also provide in Fig. 4.10 and Fig. 4.11 some samples coming from the different models using either a $\mathcal{N}(0, I_d)$ or fitting a GMM with 10 components on MNIST and CELEBA. This allows to visually differentiate the quality of the different sampling methods.

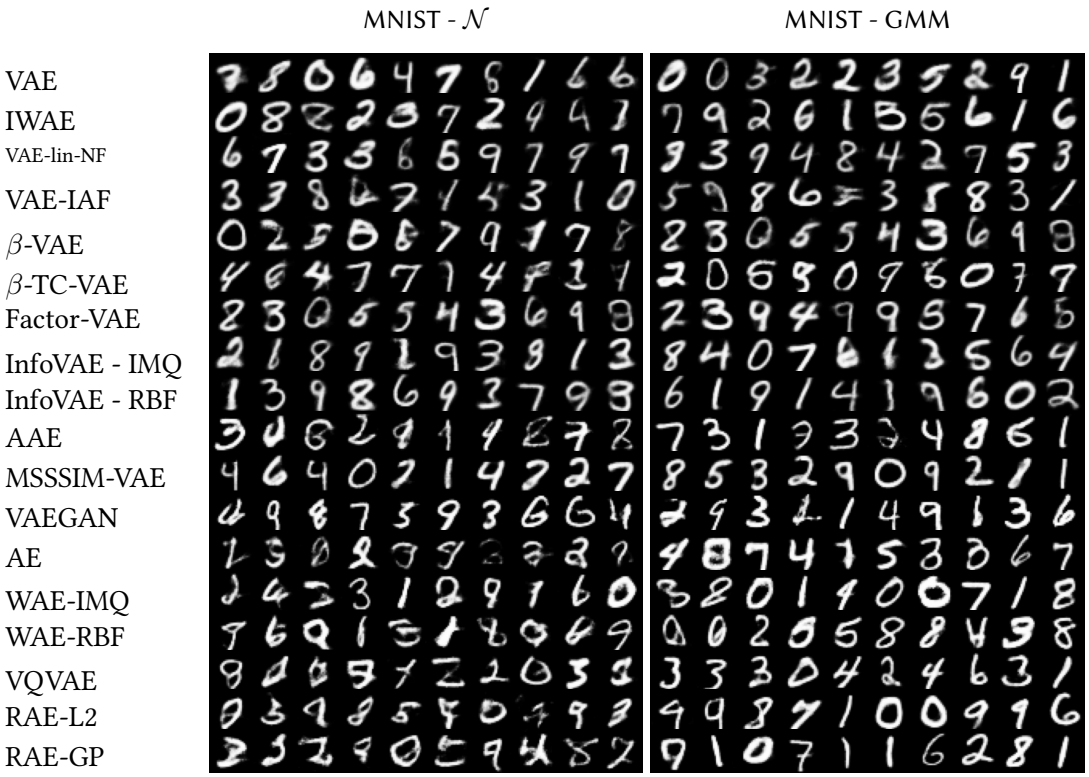


Figure 4.10: Generated samples on MNIST for a latent space of dimension 16 and ConvNet architecture. For each model, we select the configuration achieving the lowest FID on the validation set.

Figure 4.11: Generated samples on CELEBA for a latent space of dimension 64 and ConvNet architecture. For each model, we select the configuration achieving the lowest FID on the validation set.

Sampler ablation study Fig. 4.12 shows the same results as Table 4.7 but under a different prism. In this plot, we show the influence each sampler has on the generation quality for all the models considered in this study. Note that sampling using a $\mathcal{N}(0, I_d)$ for an AE, RAE or VQVAE is far from being optimal since those models do not enforce explicitly the latent variables to follow this distribution. As mentioned in the chapter, this experiment shows that using more complex density estimators such as a GMM or a normalizing flow almost always improves the generation metric.

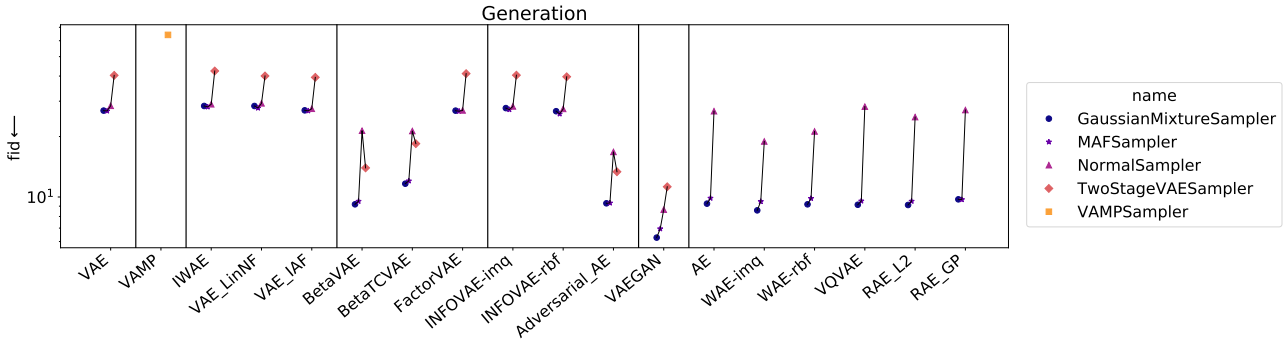


Figure 4.12: Evolution of the FID for the generation task depending on the sampler, for a ConvNet, the MNIST dataset and a latent dimension of 16. For each sampler and model, we select the configuration achieving the lowest FID on the validation set.

Neural network architecture ablation study As explained in the chapter and in Appendix 4.6.3, we consider two different neural architectures for the encoder and decoder of each model: a ConvNet (convolutional neural network) and a ResNet (residual neural network). Fig. 4.13 shows the influence the choice of the neural architecture has on the ability of the model to perform the 4 tasks presented in the chapter. The results are computed for each model on MNIST and a latent dimension of 16. The ConvNet architecture has approximately 20 times more parameters than the ResNet in such conditions. We select the best configuration for each model and each task on the validation set and report the results on the test set. Unsuprisingly, we see in Fig. 4.13 that the ConvNet architecture, more adapted to capture features intrinsic to images, leads to the best performances for reconstruction and generation. Interestingly, the ResNet outperforms the ConvNet for the classification and clustering tasks, meaning that in addition to the network complexity, its structure can play a major role in the representation learned by the models.

Training time Fig. 4.14 shows the training times required for each model for both network architectures on the MNIST dataset. For each model, we show the results obtained with the configuration giving the best performances on the generation task with fixed latent dimension 16. It is interesting to note that although VAEGAN outperforms other models on the generation task, it is at the price of a higher computational time. This is due to the discriminator network (a convolutional neural net) that is called several times during training and takes images as inputs. It should be noted that methods applying normalizing flows to the posterior (VAE-lin-NF and VAE-IAF) maintain a reasonable training time, as the flows were chosen for their scalability.

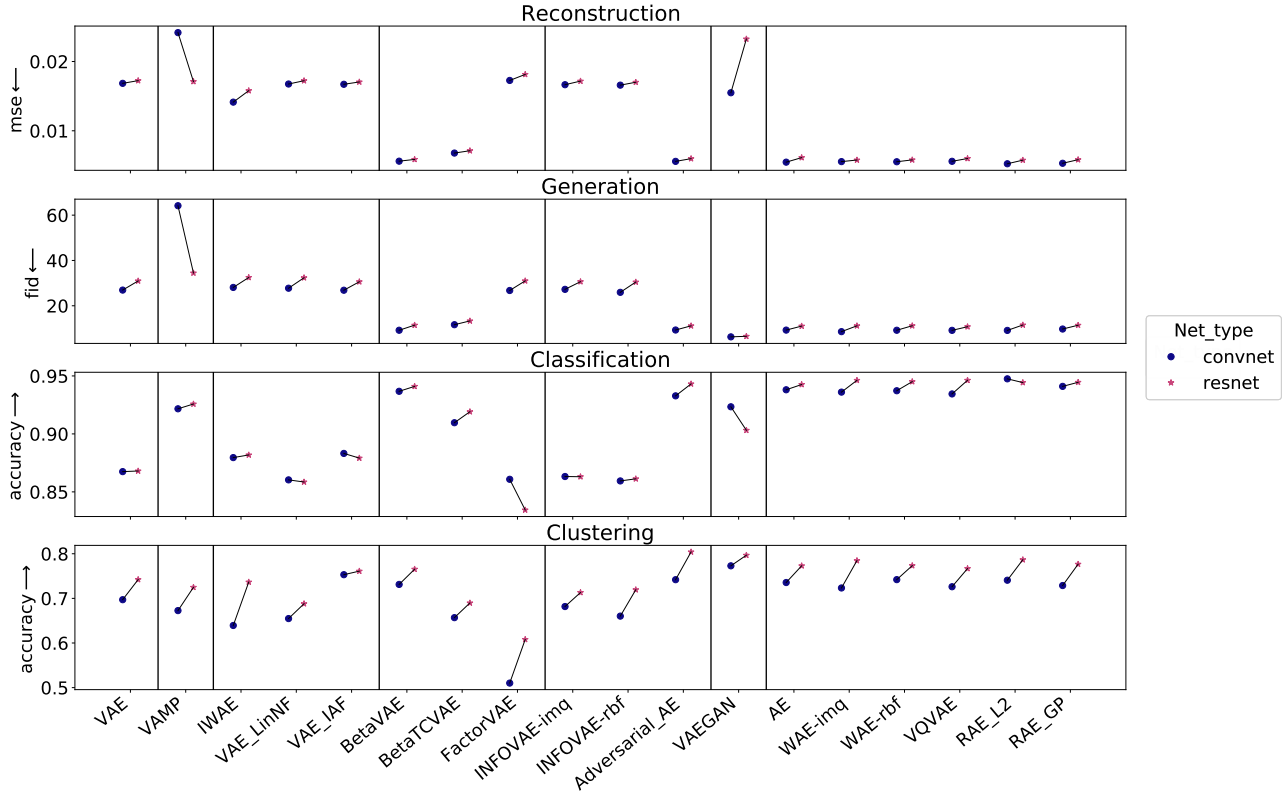


Figure 4.13: Evolution of the metrics for the 4 tasks depending on the network type on the MNIST dataset and a latent dimension of 16.

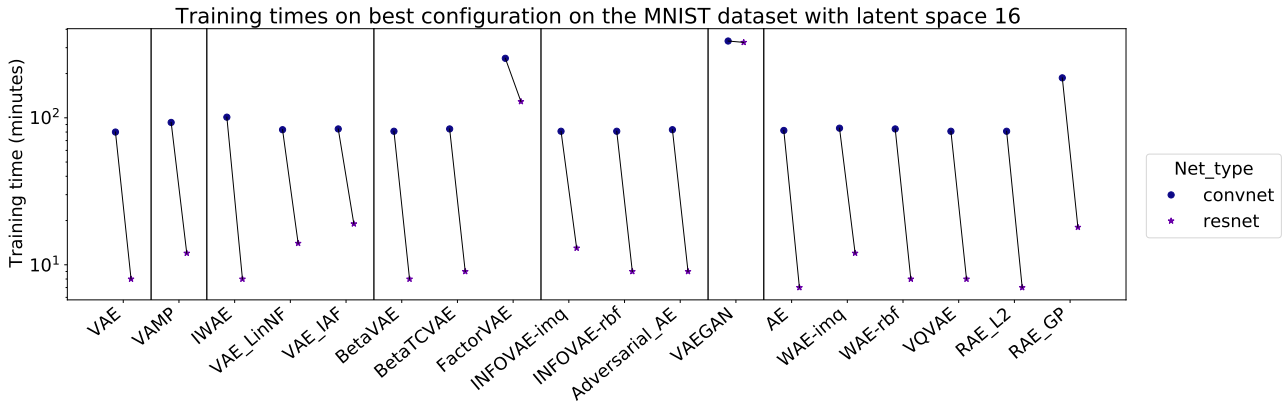


Figure 4.14: Total training time for the models trained on the MNIST dataset with latent dimension 16 with the best performance on the generation task.

Configurations and Results by Models

In this section we briefly explain each model considered in the benchmark, and show the evolution of performances on the 4 tasks and the training speed with respect to the choice of the hyper-parameters. For all 4 tasks we consider the MNIST dataset and a fixed latent space of dimension 16, as well as the Normal Gaussian sampler (if applicable) and the convolutional network architecture. For each model, 10 configuration runs with different hyper-parameters were tested. It should be noted that this configuration search was done empirically and is not exhaustive, therefore models with multiple hyper-parameters or that are sensitive to the choice of hyper-parameters will tend to have sub-optimal configuration choices. Although hyper-parameter choices are dependant on both the auto-encoder architecture and the dataset, it is interesting to note the relative evolution of the performances on the different tasks and the training time induced by different hyper-parameter choices.

Notations In order to better underline the differences between different models and for clarity purposes, we set the following unified notations:

- $X = \{x_1, \dots, x_N\} \in \mathcal{X}^N$ the input dataset
- $x \in \mathcal{X}$ an observation from the dataset, and $z \in \mathcal{Z} = \mathbb{R}^d$ its corresponding latent vector
- $\hat{x}$ the reconstruction of x by the auto-encoder model
- $p_z(z)$ the prior distribution, with $p_z \equiv \mathcal{N}(0, I_d)$ under standard VAE assumption
- $q_\phi(z|x)$ the approximate posterior distribution, modeled by the encoder. (Kingma and Welling, 2014) set
$$q_\phi(z|x) \equiv \mathcal{N}\big(\mu_\phi(x), \Sigma_\phi(x)\big)\big)$$
where $\Sigma_\phi(x) = diag[\sigma_\phi(x)]$ and $\big(\mu_\phi(x), \sigma_\phi(x)\big) \in \mathbb{R}^{2\times d}$ are outputs of the encoder network. The sampling process $z \sim q_\phi(z|x)$ is therefore performed by sampling $\varepsilon \sim \mathcal{N}(0, I_d)$ and setting $z = \mu_\phi(x) + \Sigma_\phi(x)^{1/2} \cdot \varepsilon$ (re-parametrization trick).
- $p_\theta(x|z)$ the distribution of x given z
- $p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p_z(z)dz$ the marginal distribution of x
- $q_\phi(z) = \dfrac{1}{N}\sum_{i=1}^{N} q_\phi(z|x_i)$ the aggregated posterior integrated over the training set.
- $\mathcal{D}_{KL}$ the Kullback-Leibler divergence

We further recall that (Kingma and Welling, 2014) use the unbiased estimate $\widehat{p}_\theta(x)$ of $p_\theta(x)$ defined as
$$\widehat{p}_\theta(x) = \frac{p_\theta(x|z)p_z(z)}{q_\phi(z|x)}$$
to derive the standard Evidence Lower Bound (ELBO) of the log probability $\log p_\theta(x)$ which we wish to maximize:
$$\mathcal{L}_{ELBO} = \mathbb{E}_{x\sim p_\theta(x)}\big[\mathcal{L}_{ELBO}(x)\big]$$

with

$$\mathcal{L}_{ELBO}(x) = \underbrace{\mathbb{E}_{z\sim q_\phi}[\log p_\theta(x|z)]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{KL}\big[q_\phi(z|x)||p_z(z)\big]}_{\text{regularization}}$$

The *reconstruction* loss is maximized when $\hat{x}$ is close to x, thus encouraging a good reconstruction of the input x, while the *regularization* term is maximized when $q_\phi(z|x)$ is close to $p_z(z)$, encouraging the posterior distribution to follow the chosen prior distribution.

The integration over $p_\theta(x)$ is approximated by the empirical distribution of the training dataset, and the negative ELBO function acts as a loss function to minimize for the encoder and decoder networks.

VAE with a VampPrior (VAMP) Starting from the observation that a standard Gaussian prior may be too simplistic, (Tomczak and Welling, 2018) proposes a less restrictive prior: the Variational Mixture of Posteriors (VAMP). A VAE with a VAMP prior aims at relaxing the posterior constraint by replacing the conventional normal prior with a multimodal aggregated posterior given by:

$$p_z(z) = \frac{1}{K} \sum_{k=1}^{K} q_\phi(z|u_k) \,,$$

where u_k are pseudo-inputs living in the data space $\mathcal{X}$ learned through back-propagation and acting as anchor points for the prior distribution. For the VAMP VAE implementation, we use the same architecture as the authors' implementation for the network generating the pseudo-inputs: a MLP with a single layer and Tanh activation.

Results by configuration

Table 4.8: VAMP configurations

Config	1	2	3	4	5	6	7	8	9	10
Number of pseudo-inputs (K)	10	20	30	500	100	150	200	250	300	500

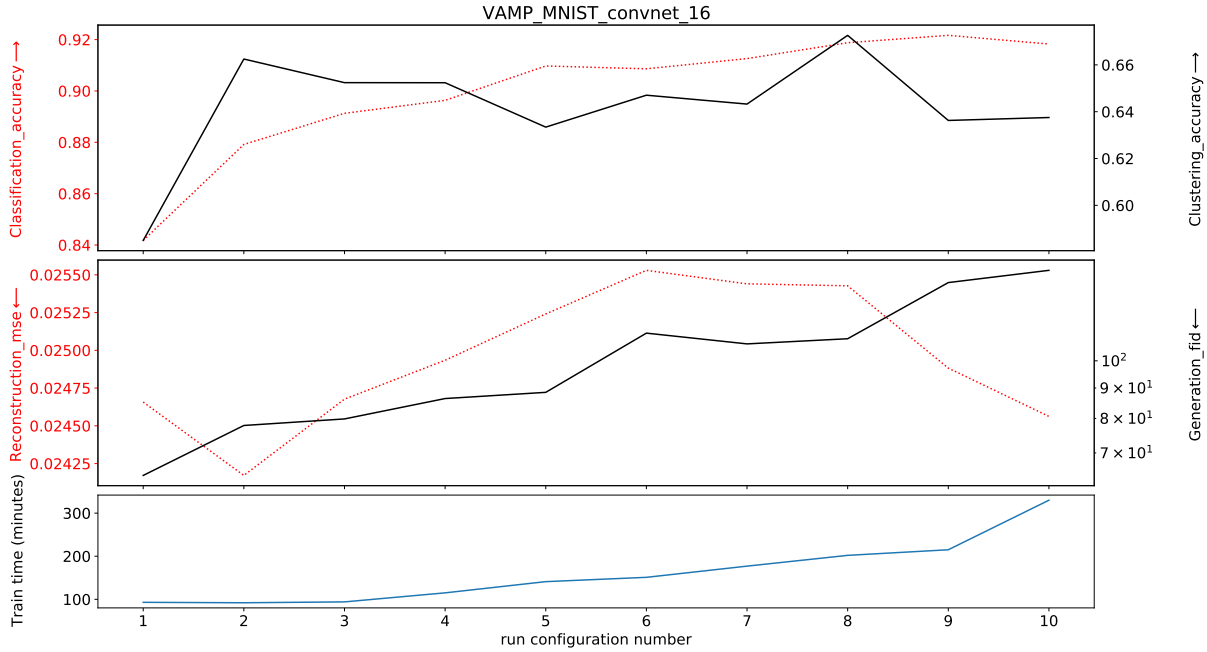


Figure 4.15: Results on VAMP

Importance Weighted Autoencoder (IWAE) (Burda et al., 2016) introduce an alternative lower bound to maximize, derived from importance weighting where the new unbiased estimate $\widehat{p}_\theta(x)$ of the marginal distribution $p_\theta(x)$ is computed with L samples $z_1, \dots, z_L \sim q_\phi(z|x)$:

$$\widehat{p}_\theta(x) = \frac{1}{L} \sum_{i=1}^{L} \frac{p_\theta(x|z_i)p_z(z_i)}{q_\phi(z_i|x)} \,.$$

This estimate induces a new lower bound of the true marginal distribution $p_\theta(x)$ using Jensen's inequality:

$$\mathcal{L}_{\text{IWAE}}(x) := \mathbb{E}_{z_1,\dots,z_L \sim q(z|x)} \left[\log \frac{1}{L} \sum_{i=1}^{L} \frac{p_\theta(x|z_i)p_z(z_i)}{q_\phi(z_i|x)} \right] \leq \log \underbrace{\mathbb{E}_{z_1,\dots,z_L \sim q(z|x)} \left[\widehat{p}_\theta(x) \right]}_{p_\theta(x)} .$$

As the number of samples L increases, $\mathcal{L}_{\text{IWAE}}(x)$ becomes closer to $\log p_\theta(x)$, therefore providing a tighter bound on the true objective. Note that when $L = 1$ we recover the original VAE framework.

As expected the reconstruction quality increases with the number of samples. Nonetheless, we note that increasing the number of samples has a significant impact on the computation of a single training step, therefore leading to a much slower training process.

Results by configuration

Table 4.9: IWAE configurations

Config	1	2	3	4	5	6	7	8	9	10
Number of samples (L)	2	3	4	5	6	7	8	9	10	12

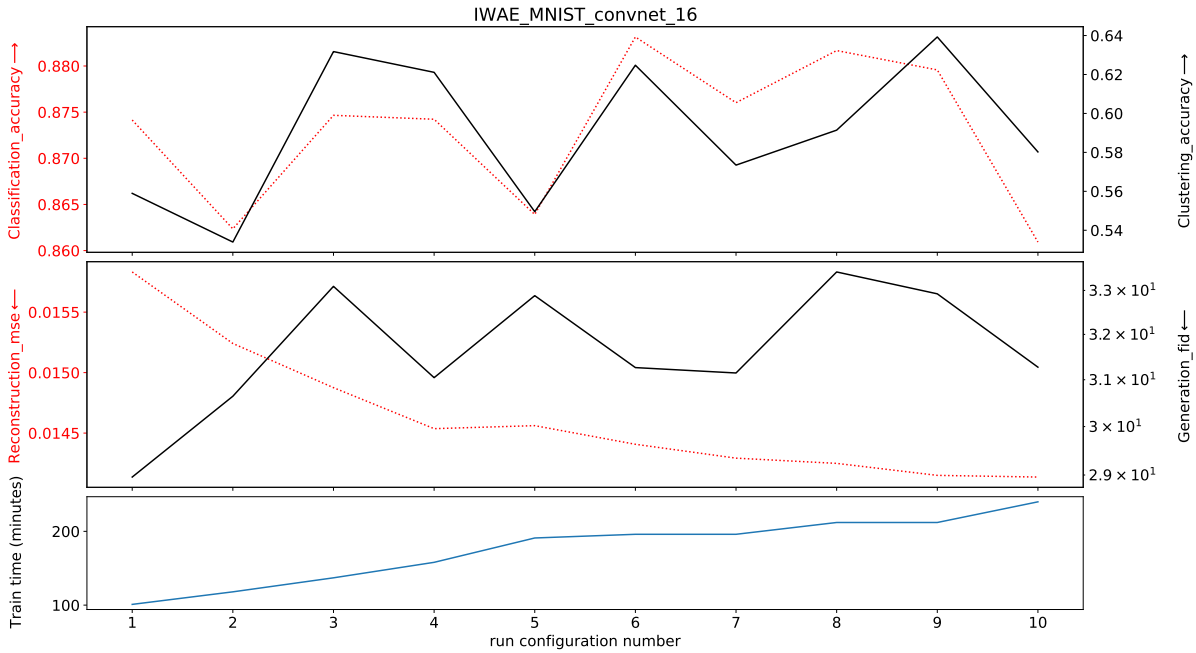


Figure 4.16: Results on IWAE

Variational Inference with Normalizing Flows (VAE-lin-NF) In order to model a more complex family of approximate posterior distributions, (Rezende and Mohamed, 2015) propose to use a succession of normalizing flows to transform the simple distribution $q_\phi(z|x)$, allowing it to model more complex behaviours. In practice, after having sampled $z_0 \sim q_\phi(z|x)$, z_0 is passed through a chain of K invertible smooth mappings from the latent space $\mathbb{R}^d$ to itself:

$$z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0)\,.$$

The modified latent vector z_K is then used as input z for the decoder network. In their paper, the authors propose to use two types of transformations: planar and radial flows.

$$f_{\text{planar}}(z) = z + uh(w^\top z + b) \quad ; \quad f_{\text{radial}}(z) = z + \beta g(\alpha, r)(z - z_0)\,,$$

where h is a smooth non-linearity with tractable derivatives and $g(\alpha, r) = \frac{1}{\alpha + r}$. The parameters are such that $r = \|z - z_0\|$, $u, w, z_0 \in \mathbb{R}^d$, $\alpha \in \mathbb{R}^+$ and $b, \beta \in \mathbb{R}$. We can easily compute the resulting density q given by

$$\log q(z_K) = \log q_\phi(z_0|x) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial z} \right| .$$

This makes the ELBO tractable and optimisation possible.

Results by configuration

Table 4.10: VAE-lin-NF configurations

Config	1	2	3	4	5	6	7	8	9	10
Flow sequence	PPPPP	RRRRR	PRPRP	10P	15P	20P	30P	PRPRPRPRPR	PPP	PRPPRPPPPR

'P' stands for planar flow - 'R' stands for radial flow

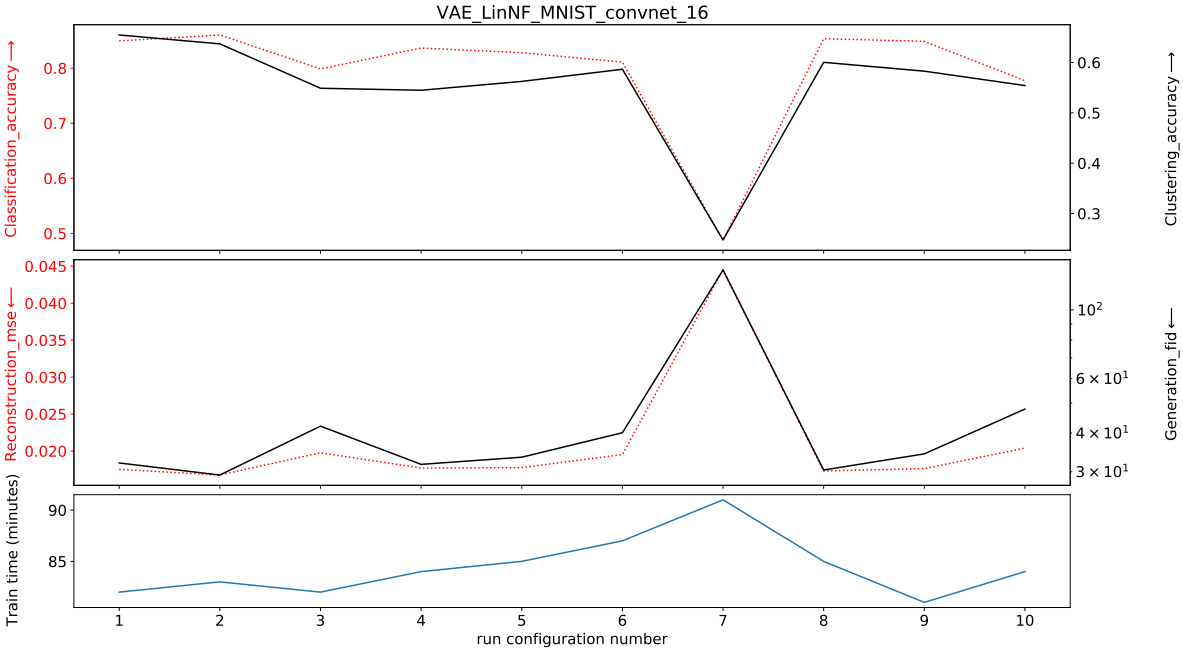


Figure 4.17: Results on VAE-lin-NF

Variational Inference with Inverse Autoregressive Flow (VAE-IAF) (Kingma et al., 2016) improve upon the works of (Rezende and Mohamed, 2015) with a new type of normalizing flow that better scales to high-dimensional latent spaces. The main idea is again to apply several transformations to a sample from a simple distribution in order to model richer distributions. Starting from $z_0 \sim q_\phi(z|x)$, the proposed IAF flow consists in applying consecutively the following transformation

$$z_k = \mu_k + \sigma_k \odot z_{k-1} \, ,$$

where μ_k and σ_k are the outputs of an autoregressive neural network taking z_{k-1} as input. Inspired from the original paper, to implement one Inverse Autoregressive Flow we use MADE (Germain et al., 2015) and stack multiple IAF together to create a richer flow. The MADE mask is made sequentially for the masked autoencoders and the ordering is reversed after each MADE.

Results by configuration

Table 4.11: VAE-IAF configurations

Config	1	2	3	4	5	6	7	8	9	10
hidden size in MADE	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	64.0	128.0
number hidden units in MADE	2.0	2.0	2.0	2.0	2.0	2.0	4.0	6.0	2.0	2.0
number of IAF blocks	1.0	2.0	5.0	10.0	20.0	4.0	4.0	4.0	4.0	4.0

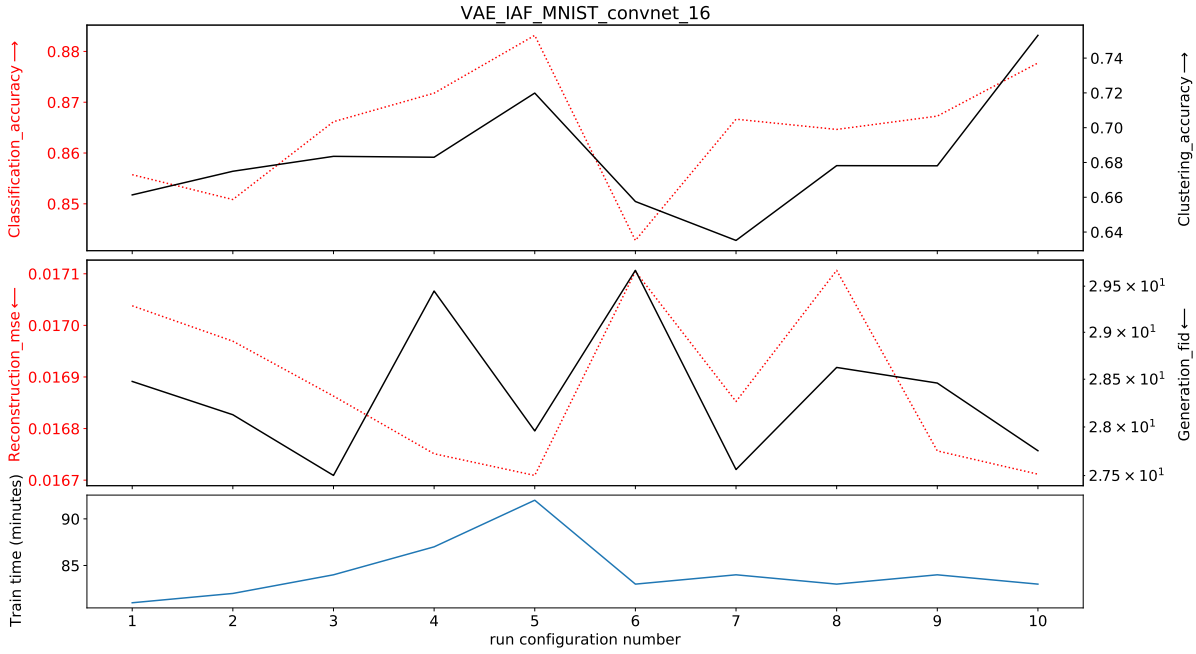


Figure 4.18: Results on VAE-IAF

β-VAE (Higgins et al., 2017) argue that increasing the weight of the KL divergence term in the ELBO loss enforces a stronger disentanglement of the latent features as the posterior probability is forced to match a multivariate standard Gaussian. They propose to add a hyper-parameter β in the ELBO leading to the following objective to maximise:

$$\mathcal{L}_{\beta\text{-VAE}}(x) = \mathbb{E}_{z\sim q_\phi}[\log p_\theta(x|z)] - \beta\mathcal{D}_{KL}\left[q_\phi(z|x)||p_z(z)\right] .$$

Although the original publication specifies $\beta > 1$ to encourage a better disentanglement, a smaller value of β can be used to relax the regularization constraint of the VAE. Therefore, for this model we consider a range of values for β from $1e^{-3}$ to $1e^{3}$.

As expected, we see in Fig. 4.19 a trade-off appearing between reconstruction and generation. Indeed, a very small β will tend to less regularize the model since the latent variables will no longer be driven to follow the prior, favouring a better reconstruction. On the other hand, a higher value for β will constrain the model, leading to a better generation quality. Moreover, as can be seen in Fig. 4.19, too high a value of β will lead to over-regularization, resulting in poor performances on all evaluated tasks.

Results by configuration

Table 4.12: β-VAE configurations

Config	1	2	3	4	5	6	7	8	9	10
β	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	0.5	2	5	10	20	$1e^{2}$	$1e^{3}$

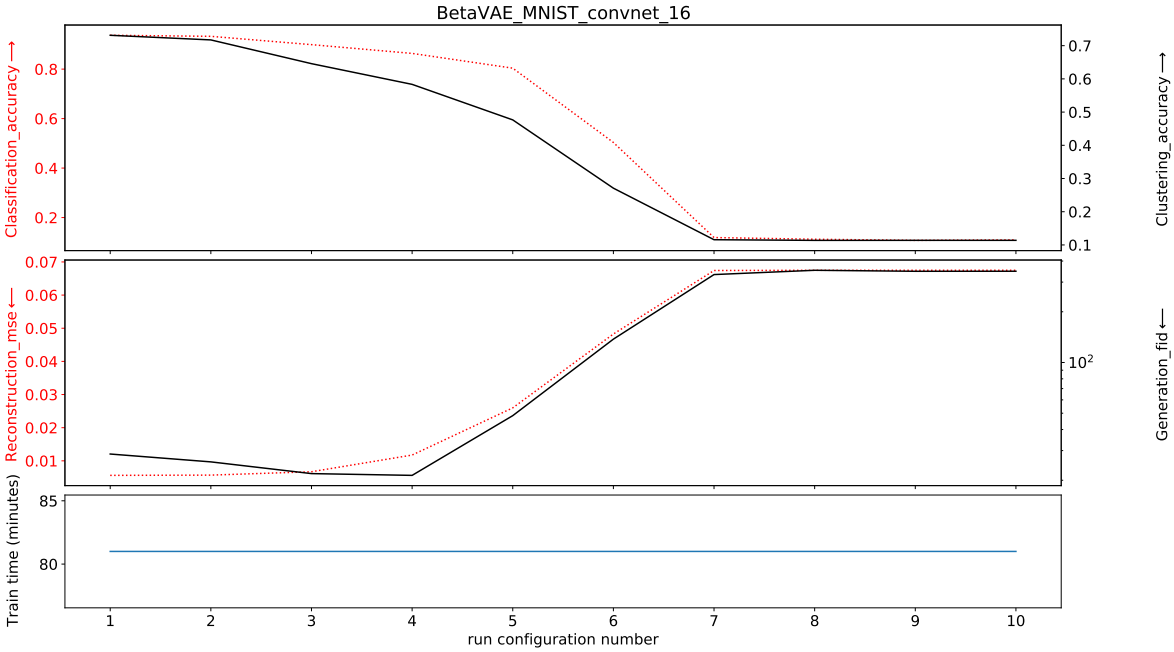


Figure 4.19: Results on β-VAE

β-TC-VAE (Chen et al., 2018b) extend on the ideas of (Higgins et al., 2017) by rewriting and reweighting specific terms in the ELBO loss with multiple hyperparameters. The authors note that the KL-divergence term of the ELBO loss can be rewritten as

$$\mathbb{E}_{x\sim p_\theta}\Big[\mathcal{D}_{KL}\big[q_\phi(z|x)||p_z(z)\big]\Big] = \underbrace{I(x,z)}_{\text{Mutual information}} + \underbrace{\mathcal{D}_{KL}\big[q_\phi(z)||\prod_{j=1}^{d} q_\phi(z_j)\big]}_{\text{TC-loss}} + \underbrace{\sum_{j=1}^{d}\mathcal{D}_{KL}\big[q_\phi(z_j)||p_z(z_j)\big]}_{\text{Dimension-wise KL}}$$

- The mutual information term corresponds to the amount of information shared by x and its latent representation z. It is claimed that maximising the mutual information encourages better disentanglement and a more compact representation of the data.
- The TC-loss corresponds to the total correlation between the latent distribution and its fully disentangled version, maximising it enforces the dimensions of the latent vector to be uncorrelated.
- Maximising the dimension-wise KL prevents the marginal distribution of each latent dimension from diverging too far from the prior Gaussian distribution

The authors therefore propose to replace the classical regularization term with the more general term

$$\mathcal{L}_{\text{reg}} := \alpha I(x,n) + \beta\mathcal{D}_{KL}\big[q_\phi(z)||\prod_j q_\phi(z_j)\big] + \gamma\sum_j \mathcal{D}_{KL}\big[q_\phi(z_j)||p_z(z_j)\big]\,.$$

Similarly to the authors, we set $\alpha = \gamma = 1$ and only perform a search on the parameter β. Fig. 4.20 shows a reconstruction-generation trade-off similar to the β-VAE model

Results by configuration

Table 4.13: β-TC-VAE configurations

CONFIG	1	2	3	4	5	6	7	8	9	10
β	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	0.5	1	2	5	10	50	$1e^{2}$

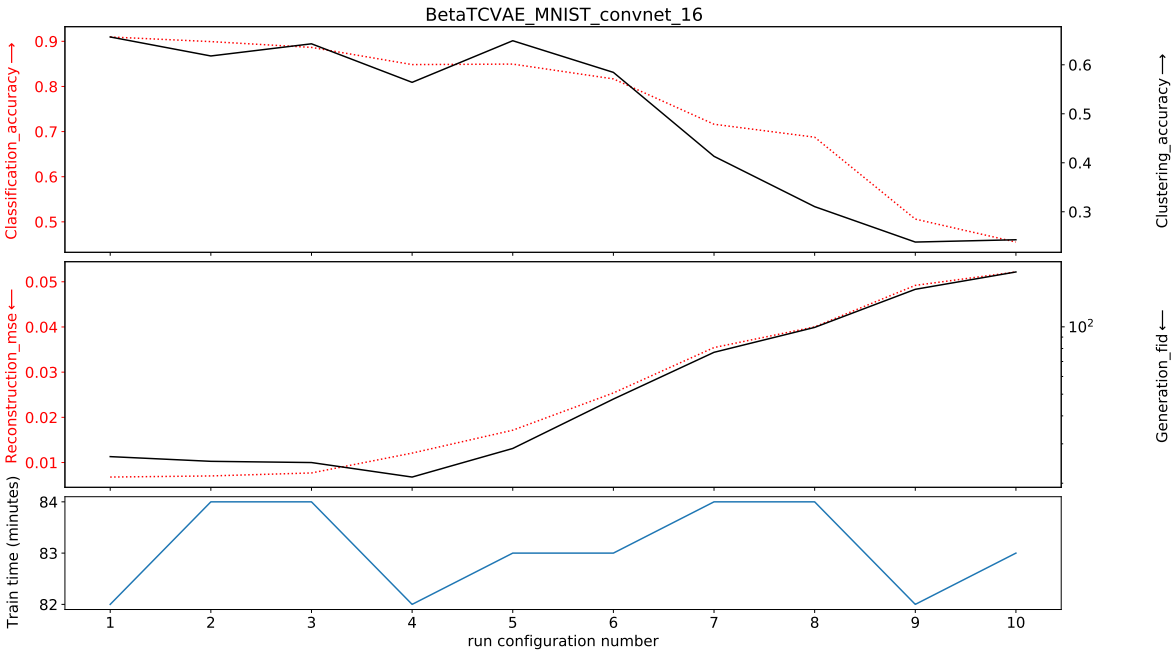


Figure 4.20: Results on β-TC-VAE

Factor VAE (Kim and Mnih, 2018) augment the VAE objective with a penalty that encourages factorial representation of the marginal distributions, enforcing a stronger disentangling of the latent space. Noting that a high β value in β-VAE ELBO loss encourages disentanglement at the expense of reconstruction quality, FactorVAE proposes a new lower bound of the log likelihood with an added disentanglement term:

$$\mathcal{L}_{\text{FactorVAE}}(x) := \mathcal{L}_{ELBO}(x) - \gamma \mathcal{D}_{KL}\big(q_\phi(z)||\bar{q}_\phi(z)\big) \text{ , with } \bar{q}_\phi(z) := \prod_{j=1}^{d} q_\phi(z_j)$$

The distribution of representations $q_\phi(z) = \frac{1}{N}\sum_{i=1}^{N} q_\phi(z|x_i)$ of the entire dataset is therefore forced to be close to its fully-disentangled equivalent $\bar{q}_\phi(z)$ while leaving the ELBO loss as it is. They further propose to approximate the KL divergence with a discriminator network D that is trained jointly to the VAE:

$$\mathcal{D}_{KL}\big(q(z)||\bar{q}(z)\big) \approx \mathbb{E}_{q_z(z)}\left[\log \frac{D(z)}{1 - D(z)}\right]$$

As suggested in the authors's paper, the discriminator is set as a MLP composed of 6 layers each with 1000 hidden units and LeakyReLU activation.

Results by configuration

Table 4.14: FactorVAE configurations

Config	1	2	3	4	5	6	7	8	9	10
γ	1	2	5	10	15	20	30	40	50	100

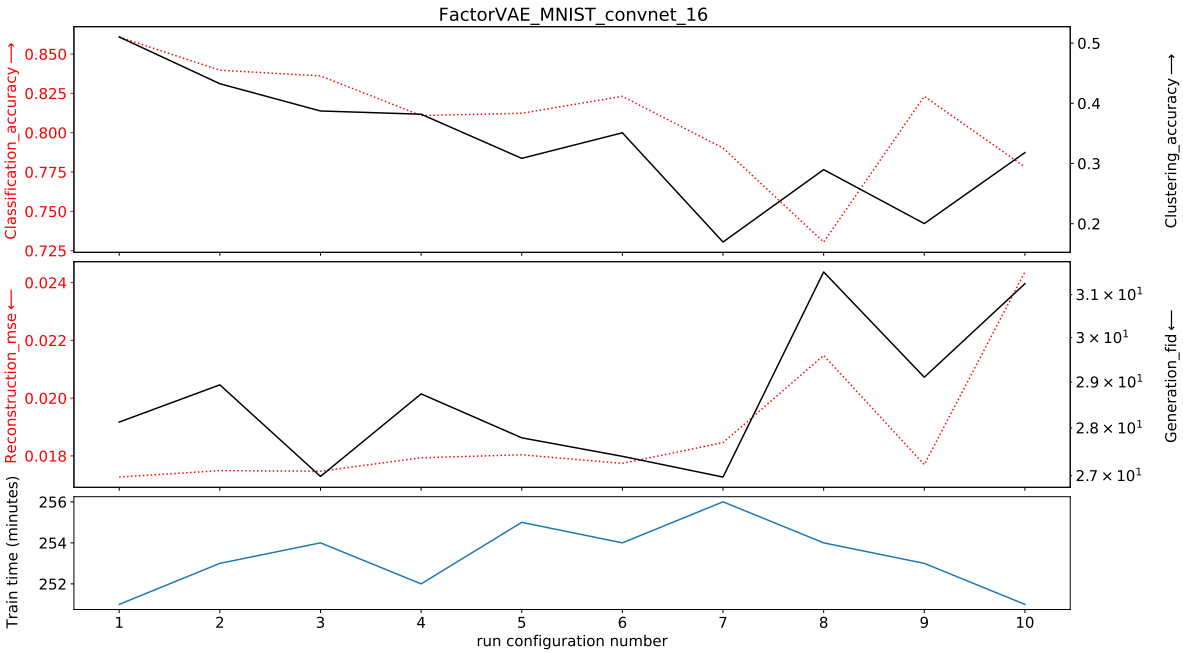


Figure 4.21: Results on FactorVAE

InfoVAE (Zhao et al., 2019) note that the traditional VAE ELBO objective can lead to both inaccurate amortized inference and VAE models that tend to ignore most of the latent variables, therefore not fully taking advantage of the modeling capacities of the VAE scheme and learning less meaningful latent representations. In order to counteract these two issues, they propose to rewrite and re-weight the ELBO objective in order to counterbalance the imbalance between the distribution in the data space and the latent space, and add a mutual information term between x and z to encourage a stronger dependency between the two variables, preventing the model from ignoring the latent encoding. One can re-write the ELBO loss in order to explicit the KL divergence between the marginalised posterior and the prior

$$\mathcal{L}_{\text{ELBO}}(x) := -\mathcal{D}_{KL}[q_\phi(z)||p_z(z)] - \mathbb{E}_{z\sim p_z}\left[\mathcal{D}_{KL}[q_\phi(x|z)||p_\theta(x|z)]\right].$$

Introducing an additional mutual information term $I_q(x;z)$ and extending the objective function to use any given divergence D between probability measures instead of the KL objective, the authors propose a new objective defined as

$$\mathcal{L}_{\text{InfoVAE}}(x) := -\lambda D[q_\phi(z)||p_z(z)] - \mathbb{E}_{z\sim p_z}\left[\mathcal{D}_{KL}[q_\phi(x|z)||p_\theta(x|z)]\right] + \alpha I_q(x;z)$$

where λ and α are hyperparameters. In our experiments, α is set to 0 as recommended in the paper in the case where $p_\theta(x|z)$ is a simple distribution. D is chosen as the Maximum Mean Discrepancy (MMD) (Gretton et al., 2012), defined as

$$\text{MMD}_k(p_\lambda(z), q_\phi(z)) = ||\int_{\mathcal{Z}} k(z,.)dp_\lambda(z) - \int_{\mathcal{Z}} k(z,.)dq_\phi(z)||_{\mathcal{H}_k}$$

with $k : \mathcal{Z}\times\mathcal{Z} \to \mathbb{R}$ a positive-definite kernel and its associated RKHS $\mathcal{H}_k$. We choose to differentiate 2 cases in the benchmark: one with a Radial Basis Function (RBF) kernel, the other with the Inverse MultiQuadratic (IMQ) kernel as proposed in (Tolstikhin et al., 2018) where the kernel is given by $k(x,y) = \sum_{s\in\mathcal{S}} \frac{s\cdot C}{s\cdot C + ||x-y||_2^2}$ with $s \in [0.1, 0.2, 0.5, 1, 2, 5, 10]$ and $C = 2\cdot d\cdot\sigma^2$, d being the dimension of the latent space and σ a parameter part of the hyper-parameter search.

The authors underline that choosing $\lambda > 0$, $\alpha = 1-\lambda$ and $D = \mathcal{D}_{KL}$, we recover the β-VAE model (Higgins et al., 2017), while choosing $\alpha = \lambda = 1$ and setting D as the Jensen Shannon divergence we recover the Adversarial AE model (Makhzani et al., 2015).

Results by configuration

Table 4.15: InfoVAE configurations

CONFIG	1	2	3	4	5	6	7	8	9	10
KERNEL BANDWIDTH - σ	$1e^{-2}$	$1e^{-1}$	0.5	1	1	1	1	1	2	5
λ	10	10	10	$1e^{-2}$	$1e^{-1}$	10	100	100	10	10

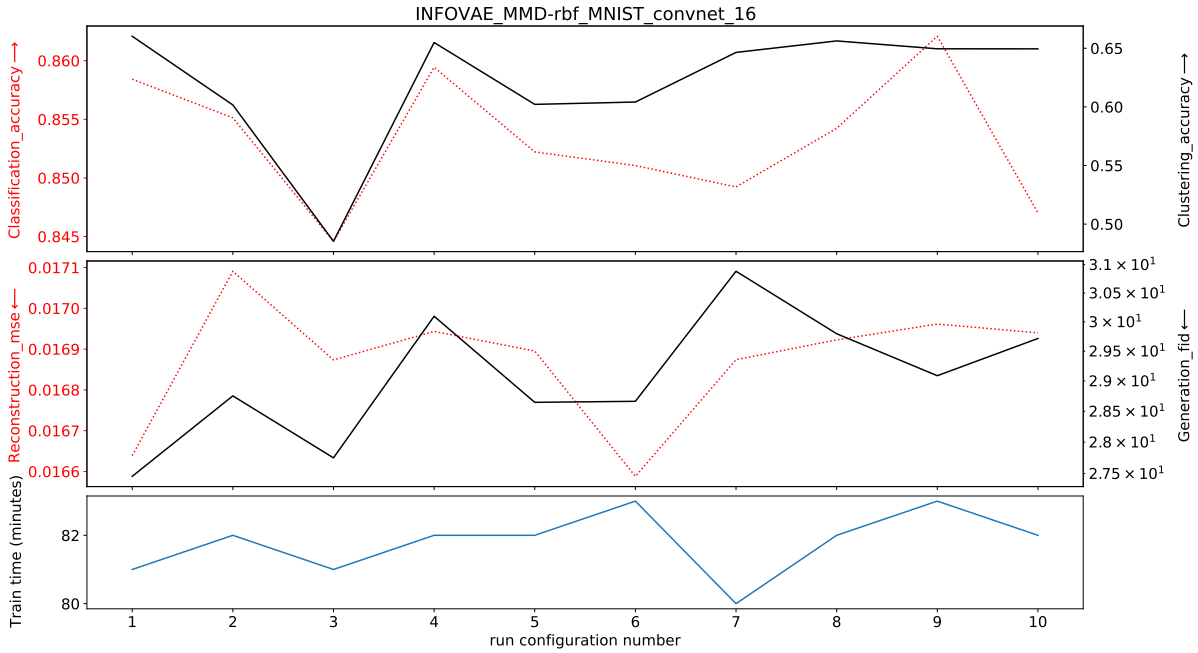


Figure 4.22: Results on InfoVAE-RBF

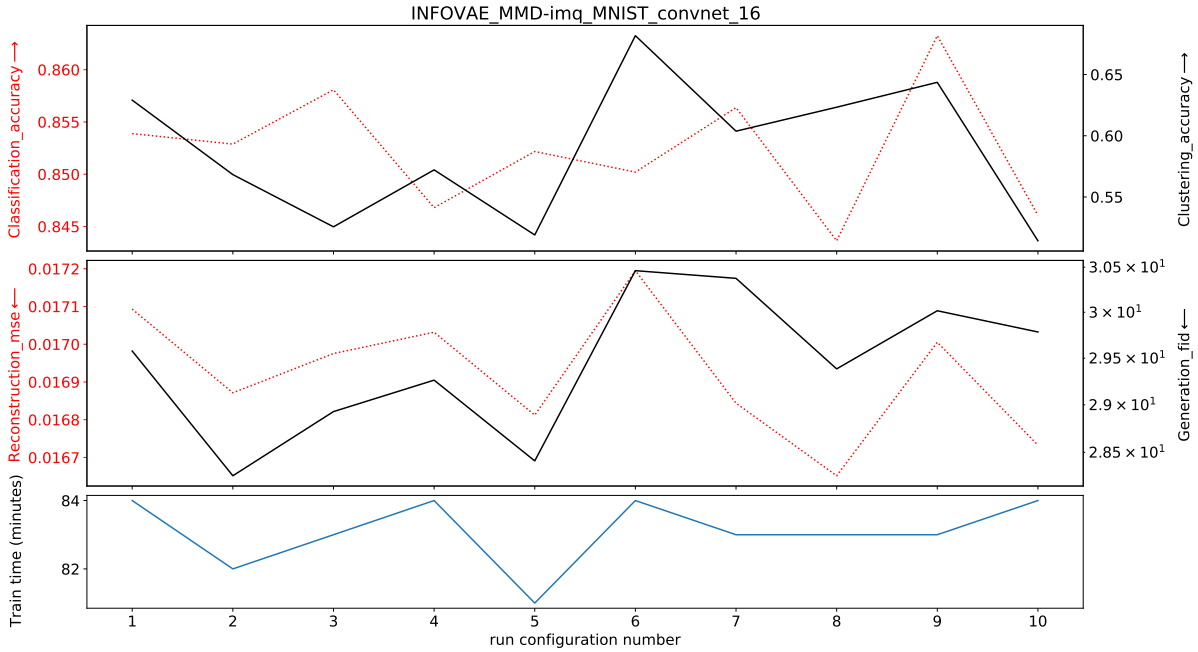


Figure 4.23: Results on InfoVAE-IMQ

Adversarial AE (AAE) (Makhzani et al., 2015) propose to use a GAN-like approach by replacing the regularization induced by the KL divergence with a discriminator network D trained to differentiate between samples from the prior and samples from the posterior distribution. The encoder network therefore acts as a generator network, leading to the following objective

$$\mathcal{L}_{\text{AAE}}(x) = \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] + \alpha\mathcal{L}_{\text{GAN}}\,,$$

with $\mathcal{L}_{\text{GAN}}$ the standard GAN loss defined by

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{\tilde{z}\sim p_z(z)}\Big[\log(1 - D(\tilde{z}))\Big] + \mathbb{E}_{x\sim p_\theta}\Big[\mathbb{E}_{z\sim q_\phi(z|x)}[\log D(z)]\Big]\,.$$

For the Adversarial Autoencoder implementation, we use a MLP neural network for the discriminator composed of a single hidden layer with 256 units and ReLU activation.

We observe a similar trade-off between reconstruction and generation quality as observed with β-VAE type models, as the α term acts like the β term, balancing between regularization and reconstruction.

Results by configuration

Table 4.16: AAE configurations

Config	1	2	3	4	5	6	7	8	9	10
α	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	0.25	0.5	0.75	0.9	0.95	0.99	0.999

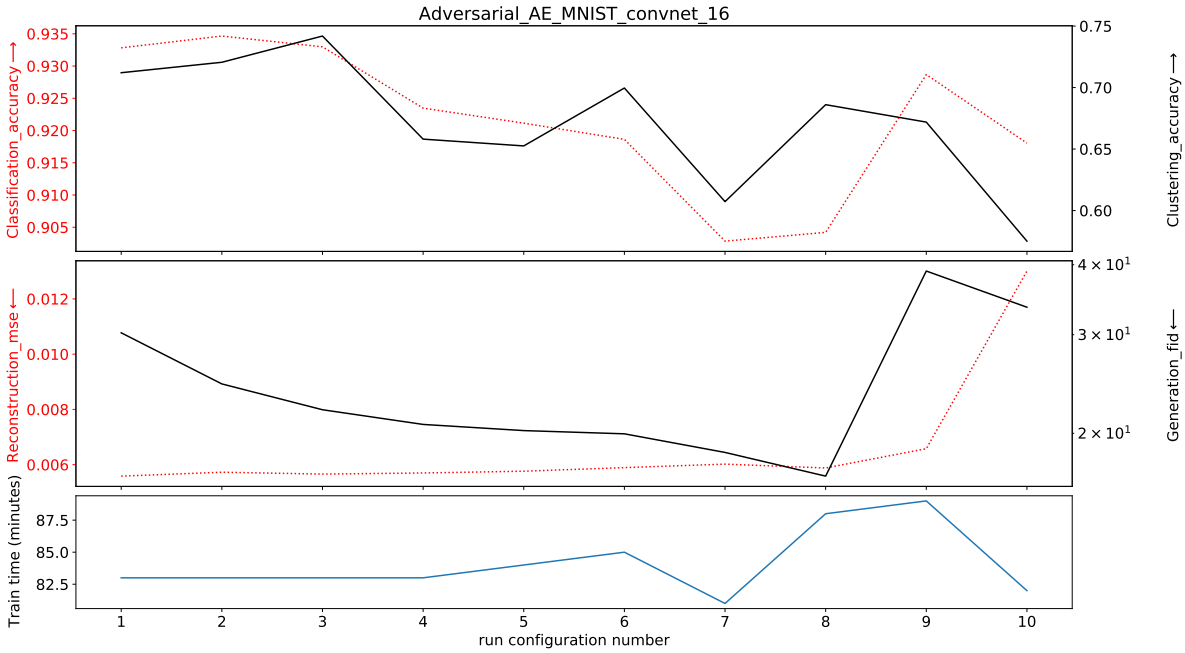


Figure 4.24: Results on Adversarial AE

EL-VAE (MSSSIM-VAE) (Snell et al., 2017) propose an extension of the ELBO loss to a more general case where any deterministic reconstruction loss $\Delta(x, \hat{x})$ can be used by replacing the probabilistic decoder p_θ with a deterministic equivalent f_θ such that the reconstruction $\hat{x}$ of x given $z \sim q_\phi(z|x)$ is defined as $\hat{x} = f_\theta(z)$. The modified ELBO objective is thus defined as

$$\mathcal{L}_{\text{EL}-\text{VAE}}(x) = \Delta(x, \hat{x}) - \beta \mathcal{D}_{KL}\big(q_\phi(z|x)||p(z)\big) \,,$$

with $\beta \leq 1$. As suggested in the original paper we use a multi scale variant of the single scale SSIM (Wang et al., 2004): the Multi Scale Structural Similarity Metric (MS-SSIM) (Wang et al., 2003).

Results by configuration

Table 4.17: MSSSIM-VAE configurations

Config	1	2	3	4	5	6	7	8	9	10
β	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$	$1e^{-1}$	$1e^{-1}$	$1e^{-1}$	1	1	1	1
window size in MSSSIM	3	5	11	5	3	11	11	5	3	15

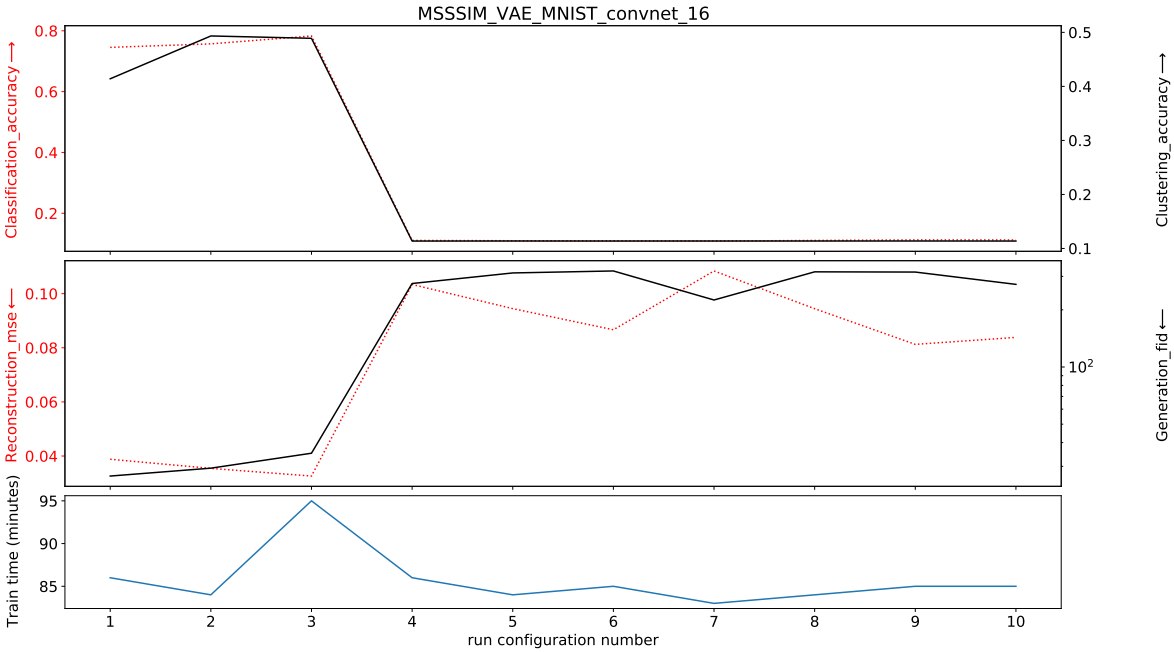


Figure 4.25: Results on MSSSIM-VAE

VAE-GAN (Larsen et al., 2016) use a GAN like approach by training a discriminator to distinguish real data from reconstructed data. In addition, the discriminator learns to distinguish between real data and data generated by sampling from the prior distribution in the latent space.

Noting that intermediate layers of a discriminative network trained to differentiate real from generated data can act as data-specific features, the authors propose to replace the reconstruction loss of the ELBO with a Gaussian log-likelihood between outputs of intermediate layers of a discriminative network D:

$$\mathcal{L}_{\text{VAE-GAN}} = \underbrace{\mathbb{E}_{z\sim q_\theta(z|x)}\Big[\log \mathcal{N}(D_l(x)|D_l(\hat{x}), I)\Big]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{KL}\big[q_\phi(z|x)||p_z(z)\big]}_{\text{regularization}} - \mathcal{L}_{GAN}\,,$$

where D_l is the output of the l^{th} layer of the discriminator D, chosen to be representative of abstract intermediate features learned by the discriminator, and $\mathcal{L}_{\text{GAN}}$ is the standard GAN objective defined as

$$\mathcal{L}_{\text{GAN}} = \log\left(\frac{D(x)}{1 - D(x_{\text{gen}})}\right),$$

where x_{gen} is generated using $z \sim p_z(z)$. As encouraged by the authors, we add a hyper-parameter α to the reconstruction loss for the decoder only, such that a higher value of α will encourage better reconstruction abilities with respect to the features extracted at the l^{th} layer of the discriminator network, whereas a smaller value will encourage fooling the discriminator, therefore favouring regularization toward the prior distribution. For the VAEGAN implementation, we use a discriminator whose architecture is similar to the model's encoder given in Table 4.5. For MNIST and CIFAR we remove the BatchNorm layer and change the activation of layer 2 to Tanh instead of ReLU. For CELEBA, the BatchNorm layer is kept and the activation of layer 2 is also changed to Tanh. For all datasets, the output size of the last linear layer is set to 1 instead of d and followed by a Sigmoid activation.

Results by configuration

Table 4.18: VAEGAN configurations

CONFIG	1	2	3	4	5	6	7	8	9	10
α	0.3	0.5	0.7	0.8	0.8	0.8	0.9	0.9	0.99	0.999
RECONSTRUCTION LAYER (L)	3	3	3	3	2	4	3	3	3	3

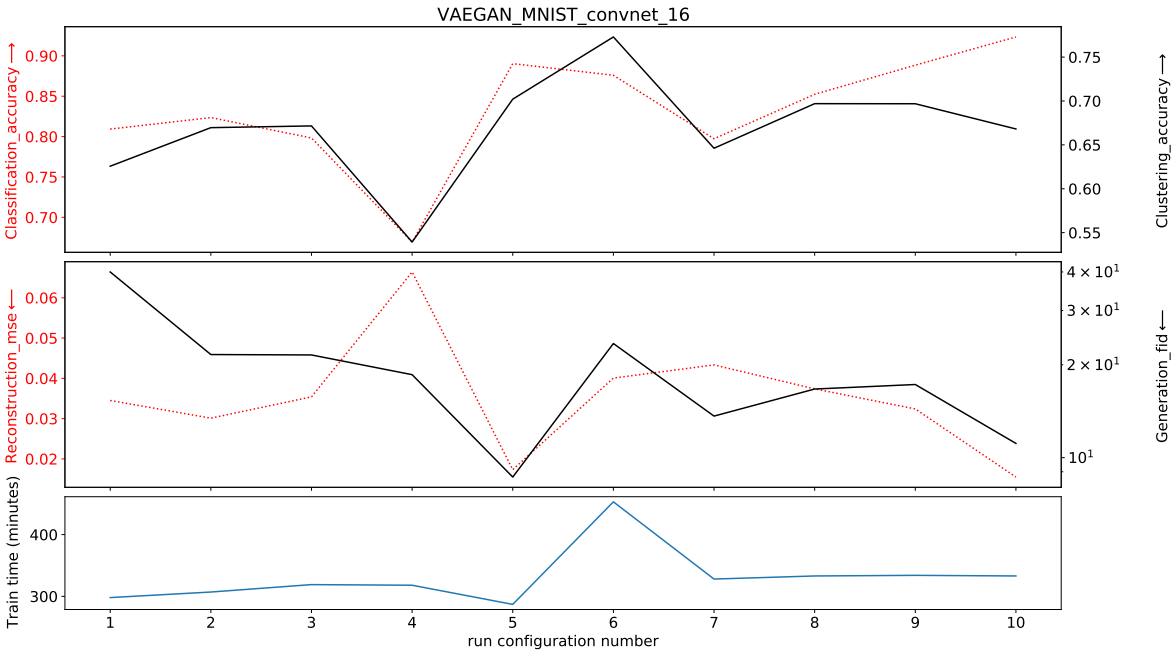


Figure 4.26: Results on VAEGAN

Wasserstein Autoencoder (WAE) (Tolstikhin et al., 2018) generalise the VAE objective by replacing both terms in the ELBO: similarly to (Snell et al., 2017) (EL-VAE), the reconstruction loss is replaced by any measurable cost function Δ, and the standard KL divergence is substituted with any arbitrary divergence $\mathcal{D}$ between two distributions, leading to the following objective function

$$E_{q_\phi(z|x)}[\Delta(x, \hat{x})] + \lambda \mathcal{D}_z(p_z(z), q_\phi(z)) \,,$$

with λ a hyper-parameter. The authors propose two different penalties for $\mathcal{D}_z$:

1. **GAN-based: WAE-GAN**

   An adversarial discriminatory network $D(z, z')$ is trained jointly to separate the "true" points sampled from the prior $p_z(z)$ from the "fake" ones sampled from $q_\phi(z|x)$, similarly to (Makhzani et al., 2015) (Adversarial AE).

2. **MMD-based**

   The Maximum Mean Discrepancy is used as a distance between the prior and the posterior distribution. This is the case considered in the benchmark. We choose to differentiate 2 cases in the benchmark: one with a Radial Basis Function (RBF) kernel, the other with the Inverse MultiQuadratic (IMQ) kernel as proposed in (Tolstikhin et al., 2018) where the kernel is given by $k(x, y) = \sum_{s \in \mathcal{S}} \frac{s \cdot C}{s \cdot C + \|x-y\|_2^2}$ with $s \in [0.1, 0.2, 0.5, 1, 2, 5, 10]$ and $C = 2 \cdot d \cdot \sigma^2$, d being the dimension of the latent space and σ a parameter part of the hyper-parameter search. As proposed by the authors, for this model we choose to use a deterministic encoder meaning that $q_\phi(z|x) = \delta_{\mu_\phi(x)}$.

Results by configuration

Table 4.19: WAE configurations

Config	1	2	3	4	5	6	7	8	9	10
kernel bandwidth - σ	$1e^{-2}$	$1e^{-1}$	0.5	1	1	1	1	1	2	5
λ	1	1	1	$1e^{-2}$	$1e^{-1}$	1	10	100	1	1

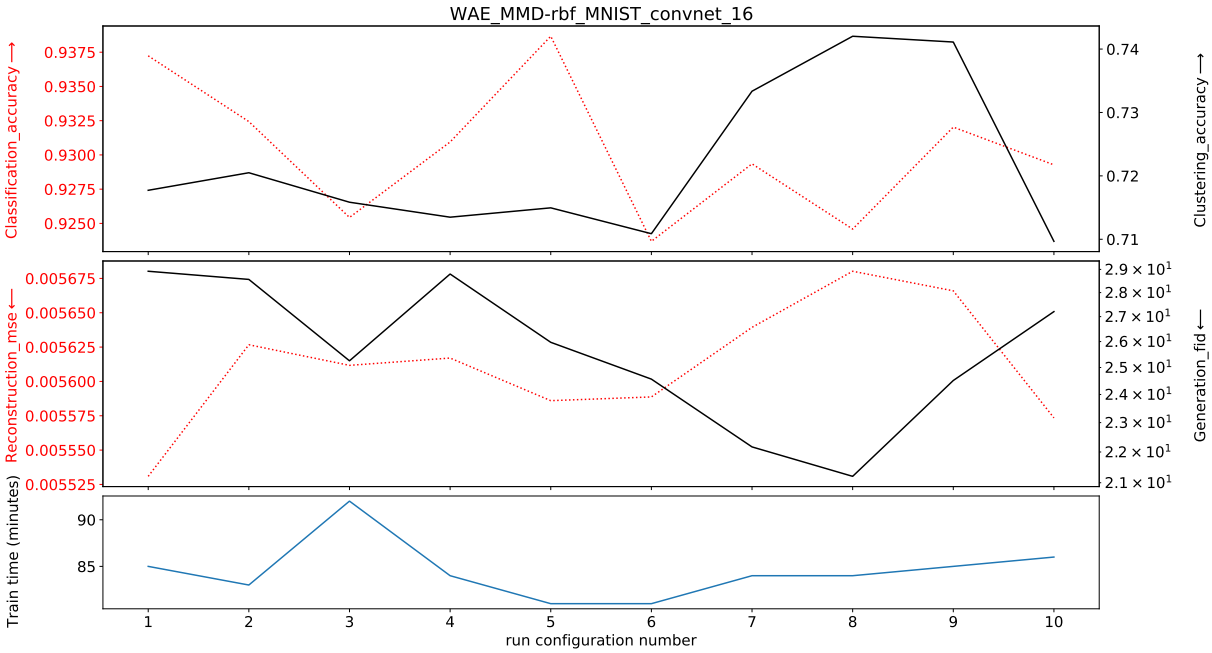


Figure 4.27: Results on WAE-RBF

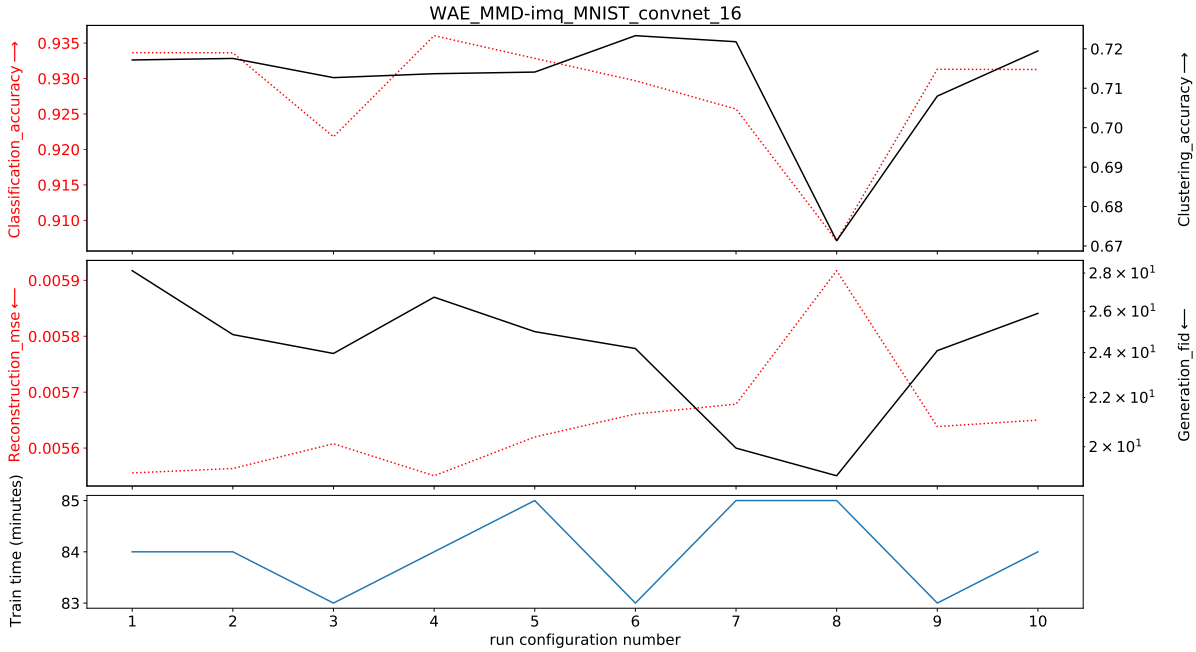


Figure 4.28: Results on WAE-IMQ

Vector Quantized VAE (VQ-VAE) (Van Den Oord et al., 2017) propose to use a discrete space. Therefore, the latent embedding space is defined as a $\mathbb{R}^{K\times D}$ vector space of K different D dimensional embedding vectors $\mathcal{E} = \{e_1, \ldots, e_K\}$ which are learned and updated at each iteration.

Given an embedding size d and an input x, the output of the encoder $z_e(x)$ is of size $\mathbb{R}^{d\times D}$. Each of its d elements is then assigned to the closest embedding vector resulting in an embedded encoding $z_q(x) \in \mathcal{E}^d$ such that $\big(z_q(x)\big)_j = e_l$ where $l = \text{argmin}_{1\leq l\leq d}||(z_e(x))_j - e_l||_2$ for $j \in [1, d]$. Since the argmin operation is not differentiable, learning of the embeddings and regularization of the latent space is done by introducing the stopgradient operator sg in the training objective:

$$\mathcal{L}_{\text{VQ-VAE}}(x) := \log p(x|z_q(x)) + \alpha||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2 .$$

For the VQVAE implementation we use the Exponential Moving Average update as proposed in (Van Den Oord et al., 2017) to replace the term $||sg[z_e(x)] - e||_2^2$ in the loss. Thus, we consider only two hyper-parameters in the search: the size of the dictionary of embeddings K and the regularization factor β.

Results by configuration

Table 4.20: VQVAE configurations

CONFIG	1	2	3	4	5	6	7	8	9	10
K	128	256	512	512	512	512	512	512	1024	2948
β	0.25	0.25	0.9	0.1	0.5	0.25	0.75	0.25	0.25	0.25

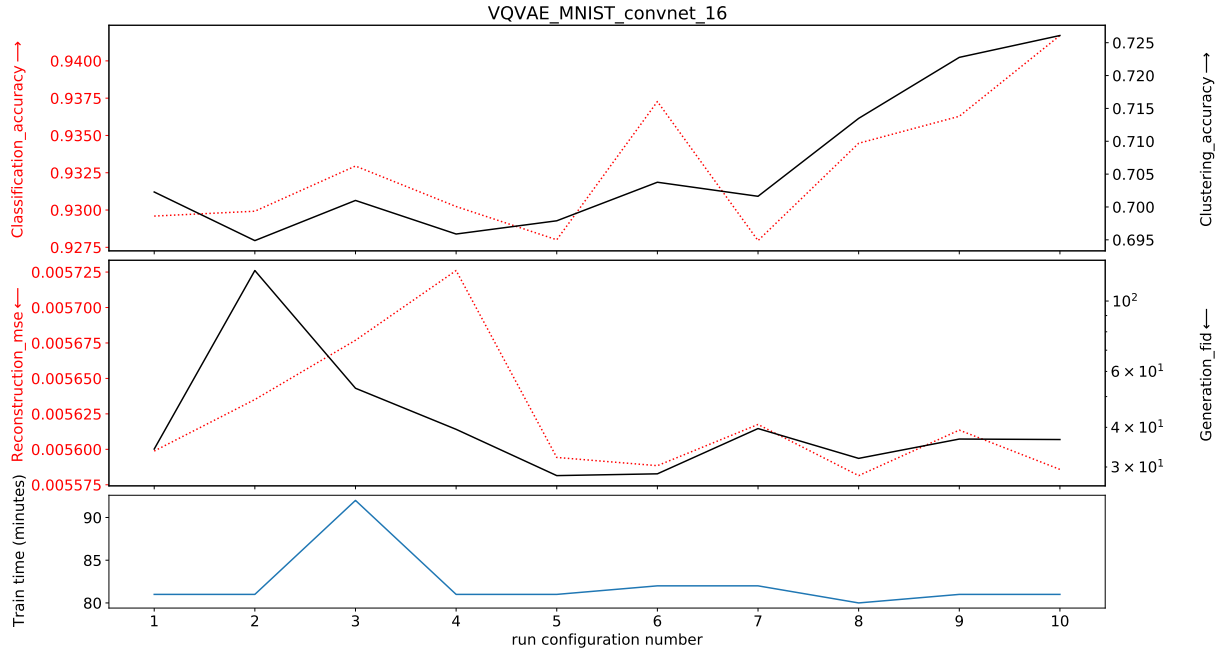


Figure 4.29: Results on VQVAE

RAE L2 and RAE GP (Ghosh et al., 2020) propose to replace the stochastic VAE with a deterministic autoencoder by adapting the ELBO objective to a deterministic case. Under standard VAE assumption with Gaussian decoder, both the reconstruction and the regularization terms in the ELBO loss can be written in closed form as

$$\begin{cases} \mathcal{L}_{\text{reconstruction}}(x) = ||x - \hat{x}||_2^2 \,, \\ \mathcal{L}_{\text{regularization}}(x) = \frac{1}{2}\Big[||z||_2^2 - d + \sum_{i=1}^{d}(\sigma_\phi(x)_i - \log \sigma_\phi(x)_i)\Big] \,. \end{cases}$$

Arguing that the regularization of the VAE model is done through a noise injection mechanism by sampling from the approximate posterior distribution $z \sim \mathcal{N}(\mu_\phi, diag(\sigma_\phi))$, the authors propose to replace this stochastic regularization with an explicit regularization term, leading to the following deterministic objective:

$$\mathcal{L}_{\text{RAE}} = ||x - \hat{x}||_2^2 + \frac{\beta}{2}||z||_2^2 + \lambda \mathcal{L}_{\text{REG}} \,,$$

where $\mathcal{L}_{\text{REG}}$ is an explicit regularization. They propose to use either

- a L2 loss on the weights of the decoder (RAE-L2), which amounts to applying weight decay on the parameters of the decoder.
- a gradient penalty on the output of the decoder (RAE-GP), which amounts to applying a L2 norm on the gradient of the output of the decoder.

Results by configuration

Table 4.21: RAE configurations

Config	1	2	3	4	5	6	7	8	9	10
β	$1e^{-6}$	$1e^{-4}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-2}$	$1e^{-1}$	1
λ	$1e^{-3}$	$1e^{-3}$	$1e^{-6}$	$1e^{-4}$	$1e^{-2}$	$1e^{-1}$	1	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$

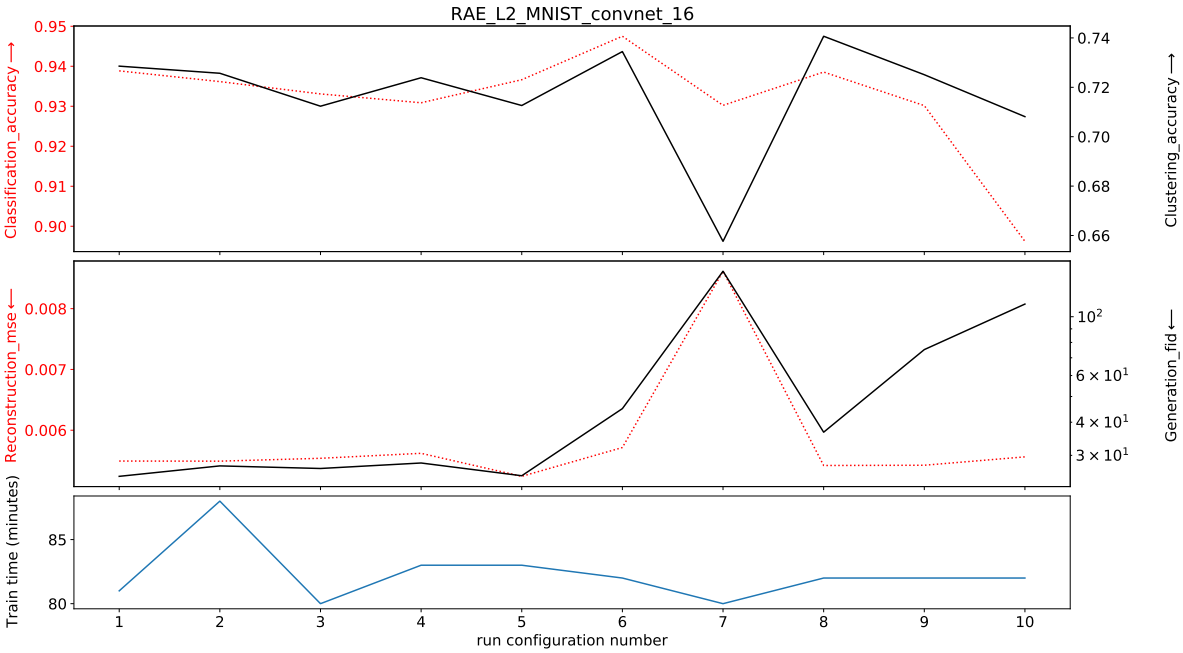


Figure 4.30: Results on RAE-L2

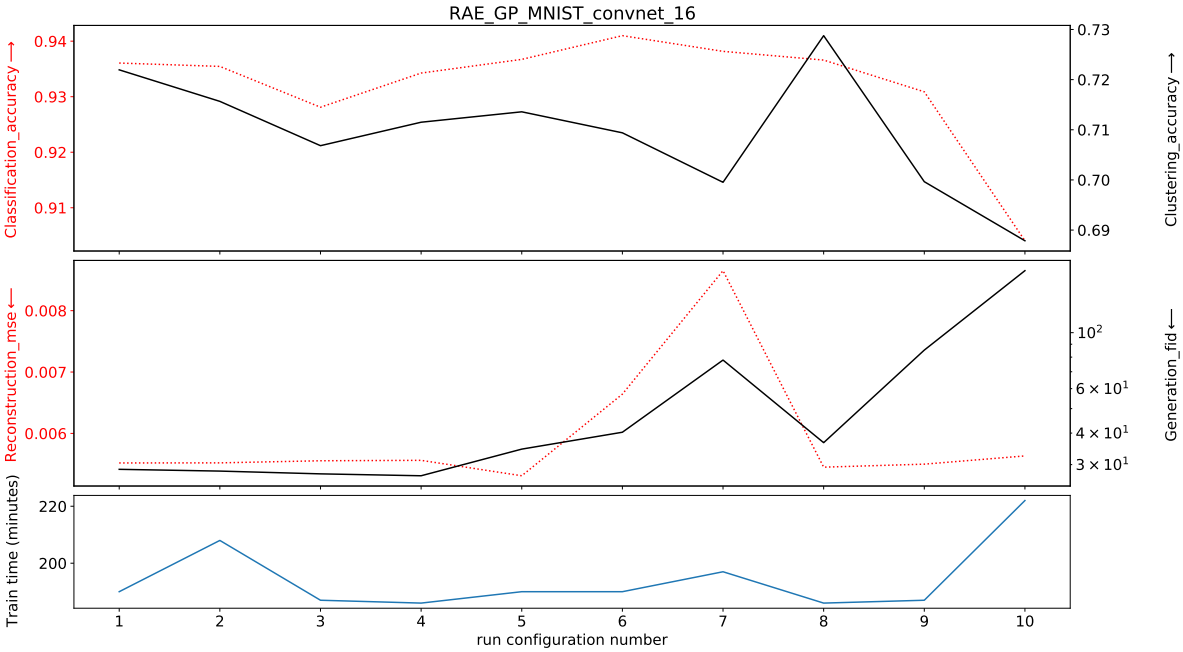


Figure 4.31: Results on RAE-GP

Chapter 5

An Image Feature Mapping Model for Continuous Longitudinal Data Completion and Generation of Synthetic Patient Trajectories

Longitudinal medical image data are becoming increasingly important for monitoring patient progression. However, such datasets are often small, incomplete, or have inconsistencies between observations. Thus, we propose a generative model that not only produces continuous trajectories of fully synthetic patient images, but also imputes missing data in existing trajectories, by estimating realistic progression over time. Our generative model is trained directly on features extracted from images and maps these into a linear trajectory in a Euclidean space defined with velocity, delay, and spatial parameters that are learned directly from the data. We evaluated our method on toy data and face images, both showing simulated trajectories mimicking progression in longitudinal data. Furthermore, we applied the proposed model on a complex neuroimaging database extracted from ADNI. All datasets show that the model is able to learn overall (disease) progression over time.

This chapter was published in a MICCAI 2022 workshop (DGM4MICCAI). See (Chadebec et al., 2022a).

5.1 Introduction

Longitudinal medical image data are important for *e.g.* modeling disease progression (Aghili et al., 2018; Zhao et al., 2021) or monitoring treatment response (Blackledge et al., 2014). However, such datasets often suffer from incomplete or inconsistent observations, and are often limited in terms of size, diversity, and balance. Generally, using inadequate data can lead to poor performances when being used to train machine learning (ML) models (Shin et al., 2016) for medical image analysis tasks such as classification (Wen et al., 2020) or segmentation (Liu et al., 2021a).

To increase the size and variability of (non-longitudinal) medical imaging datasets, conventional data augmentation techniques such as rotation, cropping, or more resourceful augmentations (Hussain et al., 2017) have been widely used (Shorten and Khoshgoftaar, 2019). However, the improved performances of deep generative models have given them the potential to perform image synthesis. Examples of such models are Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), which generate realistic images using a discriminator that distinguishes between real and synthetic images, and Variational Autoencoders (VAEs) (Kingma and Welling, 2014), which constrain image features to follow a given prior distribution in order to generate synthetic images. These models have shown potential for synthesizing medical images of various modalities such as magnetic resonance imaging (MRI) (Calimeri et al., 2017; Shin et al., 2018; Chadebec et al., 2022b), computed tomography (CT) (Frid-Adar et al., 2018; Sandfort et al., 2019), X-ray (Madani et al., 2018; Salehinejad et al., 2018), or positron emission tomography (PET) (Bi et al., 2017). In addition, several methods have been proposed to address data imputation or progression modeling in longitudinal imaging data of *e.g.* MRI (Louis et al., 2019; Kim et al., 2021) or simulated discrete progressions (Ramchandran et al., 2021).

Although the topics of inter- and extrapolating longitudinal (medical) imaging data are well studied, to the best of our knowledge there is no model that addresses both of these aspects at once and is able to continuously generate realistic trajectories. In this chapter, we propose a new deep generative model that is capable of: (1) generating realistic progression in images, (2) imputing missing data in existing patient trajectories, and (3) producing synthetic images with corresponding trajectories of non-existent patients[1].

5.2 Proposed Method

We propose a new generative model for longitudinal imaging data that consists of two steps. In the first step, relevant features are extracted from the input images using a VAE, and the second step maps these features into a linear trajectory to account for the progression over time. In the following, we refer to an observation, *e.g.* an image, as $y_{i,j} \in \mathcal{Y}$, with $i \in [1, N]$ the individual's identifier, $t_j \in \mathbb{R}_+^*$, where $j \in [0, P_i]$ the time of the observation. N is the number of individuals and P_i is the number of observations of i after the first time visit t_0.

[1]Code and dataset details are available at https://github.com/evihuijben/longVAE

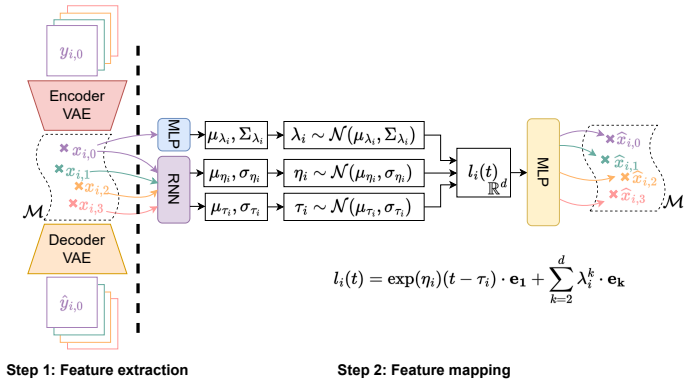


Figure 5.1: Model sketch. First, features are extracted from images using the VAE (step 1), then, the proposed generative model maps these features to a straight line in Euclidean space (step 2). Network details are provided in Appendix 5.6.2.

5.2.1 Feature Extraction

Medical images are often complex and high-dimensional data. Therefore, instead of proposing a model directly acting on images, we propose to first extract meaningful features using a VAE (referred to as *the VAE* in the following). We use an autoencoder because it constrains comparable images to be encoded into similar locations such that minor variations in the latent space lead to smooth transformations in the image space. Since we expect smooth progressions, the VAE is likely to directly unveil trajectories in the latent space, thereby facilitating the second step of our method (referred to as *the generative model* in the following). In the following $x_{i,j} \in \mathcal{M}$ refers to the features of observation $y_{i,j}$.

5.2.2 Trajectory Modeling

We propose to learn parametric functions that map the features onto a linear trajectory in a d-dimensional Euclidean space $\mathbb{R}^d$ with standard basis $\{\mathbf{e_1}, \dots, \mathbf{e_d}\}$, accounting for an individual's progression. We use the framework proposed in (Louis et al., 2019), in which an individual's progression trajectory at time t is modeled in $\mathbb{R}^d$ as

$$l_i(t) = \exp(\eta_i)(t - \tau_i) \cdot \mathbf{e_1} + \sum_{k=2}^{d} \lambda_i^k \cdot \mathbf{e_k}\,, \tag{5.1}$$

where η_i is a velocity parameter, τ_i is a delay, and $\lambda_i = (\lambda_i^k)_{2 \leq k \leq d}$ are spatial parameters. Contrary to (Louis et al., 2019), we adopt a fully variational approach to make the model generative in a similar fashion as (Kingma and Welling, 2014). Assuming a set of embeddings $x = \{(x_{i,j})_{1 \leq i \leq N, 0 \leq j \leq P_i}\} \in \mathcal{M}$, we first assume that given two individuals i and i', the features $x_{i,j}$ and $x_{i',j}$ are independent. Therefore, we propose to maximize the following likelihood objective $p(x) = \prod_{i=1}^{N} p(x_i)$, where $x_i = (x_{i,0}, \cdots, x_{i,P_i})$. We further assume that the latent variables $z_i = (\eta_i, \tau_i, \lambda_i^2, \cdots, \lambda_i^d) \in \mathbb{R}^{d+1}$ in Eq. (5.1) are such that the features of individual i at time t_j are generated by:

$$p_\theta(x_{i,j}|z_i) = \mathcal{N}\Big(\mu_\theta(l_i(t_j)), \sigma \cdot I_d\Big)\,,$$

where $l_i(t_j)$ is the linear trajectory evaluated at t_j, and $\mu_\theta : \mathbb{R}^d \to \mathcal{M}$ is parameterized using a multilayer perceptron (MLP) and maps $\mathbb{R}^d$ to the feature space. We further assume that η_i, τ_i, and λ_i are independent and that for a given individual i, the features $x_{i,j}$ taken conditionally to z_i are independent. Furthermore, we set the prior distributions over the latent variables to: $\eta_i \sim \mathcal{N}(0, \sigma_\eta)$, $\tau_i \sim \mathcal{N}(0, \sigma_\tau)$, $\lambda_i \sim \mathcal{N}(0, I_{d-1})$, with $\sigma_\eta > 0$ and $\sigma_\tau > 0$. Finally, the likelihood for an individual i writes:

$$p(x_i) = \int_{z_i \in \mathbb{R}^{d+1}} p_\theta(x_i|z_i)p(z_i)dz_i = \int_{z_i \in \mathbb{R}^{d+1}} \prod_{j=0}^{P_i} p_\theta(x_{i,j}|z_i) \prod_{\kappa_i \in \{\eta_i, \tau_i, \lambda_i\}} p(\kappa_i)d\kappa_i \,.$$

Since $p(z_i|x_i)$, the true posterior distribution, is unknown, we rely on variational inference (Jordan et al., 1999). Hence, we introduce a variational distribution $q_\varphi(z_i|x_i) = q_\varphi(\eta_i|x_i)q_\varphi(\tau_i|x_i)q_\varphi(\lambda_i|x_i)$ and derive a new estimate of the likelihood $p(x_i) = \mathbb{E}_{z_i \sim q_\varphi(z_i|x_i)}\left[\frac{p(x_i,z_i)}{q_\varphi(z_i|x_i)}\right]$. We then compute a lower bound on the true objective using Jensen inequality and importance sampling using the variational distribution.

$$\begin{aligned}
\log\ p(x_i) &= \log\ \mathbb{E}_{z_i \sim q_\varphi(z_i|x_i)}\left[\frac{p(x_i, z_i)}{q_\varphi(z_i|x_i)}\right] \\
&\geq \mathbb{E}_{z_i \sim q_\varphi(z_i|x_i)}\Big[\log\ p(x_i|z_i)\Big] - \sum_{\kappa_i \in \{\eta_i, \tau_i, \lambda_i\}} \mathrm{KL}(q_\varphi(\kappa_i|x_i)|p(\kappa_i))\,, \\
&\geq \mathbb{E}_{z_i \sim q_\varphi(z_i|x_i)}\Big[\log\ p(x_i|z_i)\Big] - \mathrm{KL}(q_\varphi(\eta_i|x_i)|p(\eta_i)) \\
&\quad - \mathrm{KL}(q_\varphi(\tau_i|x_i)|p(\tau_i)) - \mathrm{KL}(q_\varphi(\lambda_i|x_i)|p(\lambda_i))
\end{aligned}$$

with KL the Kullback–Leibler divergence. In practice, we use multivariate Gaussians as variational distributions: $\eta_i \sim \mathcal{N}(\mu_\varphi^{\eta_i}, \Sigma_\varphi^{\eta_i})$, $\tau_i \sim \mathcal{N}(\mu_\varphi^{\tau_i}, \Sigma_\varphi^{\tau_i})$ and $\lambda_i \sim \mathcal{N}(\mu_\varphi^{\lambda_i}, \Sigma_\varphi^{\lambda_i})$. The parameters for progression, η_i and τ_i, are estimated from an input sequence using a recurrent neural network (RNN), while the spatial parameters, $(\lambda_i^2, \ldots, \lambda_i^d)$, are computed from the features of the image acquired at time t_0 using a MLP. The implementation details of the RNN and MLP can be found in Appendix 5.6.2, and a sketch of the model is presented in Fig. 5.1. Taking only the first image's features for the spatial parameters allows to estimate their value even if only one observation is available and to generate possible future progressions. Finally, we obtain the following loss function for one individual (removing constant terms):

$$\mathcal{L}_i = \sum_{j=0}^{P_i} \|x_{i,j} - \mu_\theta(l_i(t_j))\|^2 + \sum_{\kappa_i \in \{\eta_i, \tau_i, \lambda_i\}} KL(q_\varphi(\kappa_i|x_i)|p(\kappa_i))\,.$$

After training, we can either 1) generate fully synthetic trajectories using the aforementioned prior distributions, 2) produce possible progressions for a given individual i by estimating its λ_i and varying η_i and τ_i, or 3) interpolate and extrapolate existing trajectories by estimating the latent variables. Image sequences are then generated by recovering the features corresponding to a linear trajectory evaluated at a given time using a second MLP and passing them to the decoder of the VAE. In practice, we sample λ_i with a mixture of Gaussians since (Ghosh et al., 2020) recently showed that this approach alleviates the low expressiveness of the prior and allows to generate more convincing samples.

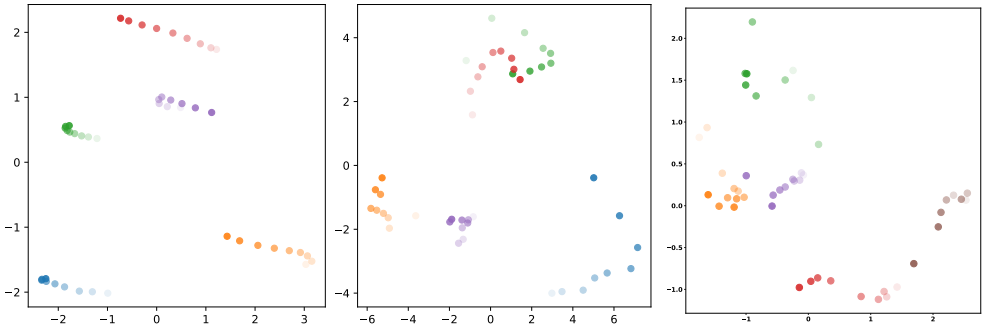


Figure 5.2: PCA projections for 'Starmen' (left), 'CelebA' (middle) and ADNI (right).

5.3 Data

We evaluate the proposed model using three longitudinal datasets. The first dataset is a toy dataset referred to as Starmen[2] (Couronné et al., 2021) consisting of 64×64 binary images of 1,000 individuals that portray synthetic transformations based on the longitudinal model of (Bône et al., 2018), captured in 10 observations per individual. The second dataset, CelebA (aligned and cropped version downloaded in 2021) (Liu et al., 2015), consists of 64×64 RGB images of celebrities' faces. To resemble longitudinal medical images, we converted these images to gray scale and applied a simulated progression model by applying a non-linear intensity transform, a growth factor, a rotation, and adding Gaussian noise. This dataset can be considered very challenging since the images undergo global and local geometric transformations and photometric variations. The last dataset was obtained from the Alzheimer's Disease Neuroimaging Initiative[3]. We used a total of 8,318 MRI scans, obtained from 1,799 subjects, with an average of 4.6 ± 2.3 scans per person. The average time between the first and the last scan was 2.9 ± 2.4 years. We selected the 100^{th} axial slice of every preprocessed scan and cropped it to 182×182. The subject's ages were used to define the observation times for the generative model, which were normalized between the overall oldest and youngest age. Details of the datasets (*e.g.* preprocessing steps, progression model, data splits, and example image trajectories) can be found in Appendix 5.6.1.

5.4 Experiments

Most experiments in this section are performed using Starmen and CelebA because these datasets are fully controlled and allow visual evaluation by non-medical experts. ADNI is used to show that results can be extended to medical data. In what follows, the models are selected on the validation set and tested on an hold-out test set. Experimental and implementation details are provided in Appendix 5.6.2.

[2]Downloaded from https://doi.org/10.5281/zenodo.5081988

[3]Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (http://adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf

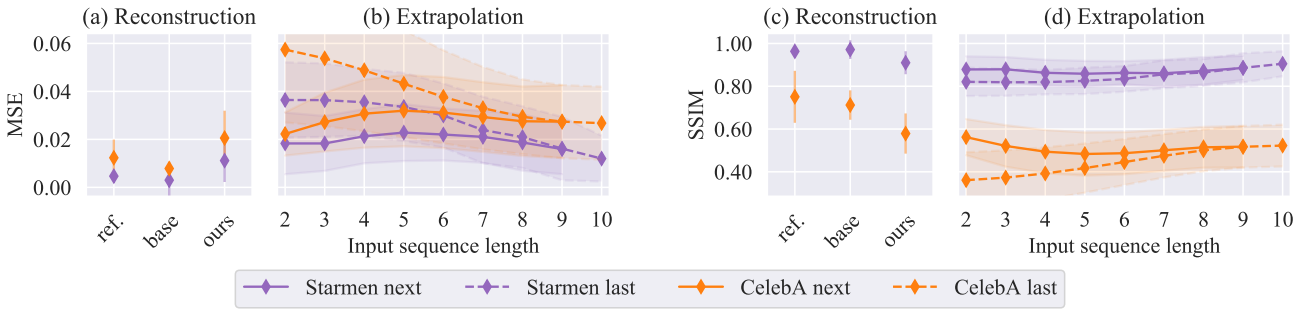


Figure 5.3: Mean and standard deviation of MSE/SSIM (a,b/c,d) for various evaluations. (a/c) Metric between consecutive images in the test sequences (*ref.*) and reconstruction metrics using only the VAE (*base*) or the generative model (*ours*). (b/d) Metrics for the next and last image extrapolated based on a varying input sequence length.

Feature Extraction and Reconstruction First, we train the VAE on each training set, disregarding the longitudinal component, and confirm the hypothesis that the features directly unveil clear trajectories over time, as can be seen in Fig. 5.2. To justify that mapping those trajectories to linear ones (step 2 in Fig. 5.1) is not too constraining, we analyze the reconstruction results obtained by 1) only encoding and decoding test images using the VAE (*base*), and 2) training the generative model to map the extracted feature trajectories to straight lines (Eq. (5.1)), evaluate $l(t)$ at observation times and pass the corresponding features to the decoder of the VAE (*ours*). Fig. 5.3a and 5.3c show the mean squared error (MSE) and structural similarity (SSIM), respectively, of the test set reconstructions. Note that the results obtained using the proposed model is not expected to be better than the one obtained using the VAE (*base*) because the generative model only acts on the features and we do not use any image-based reconstruction cost during its training. The metric values can be put into perspective by considering the mean value between two consecutive images in the test set (*ref.*). The visual reconstructions in the second row of Fig. 5.4 show that linear trajectory modeling does not considerably affect the image reconstruction ability of the model.

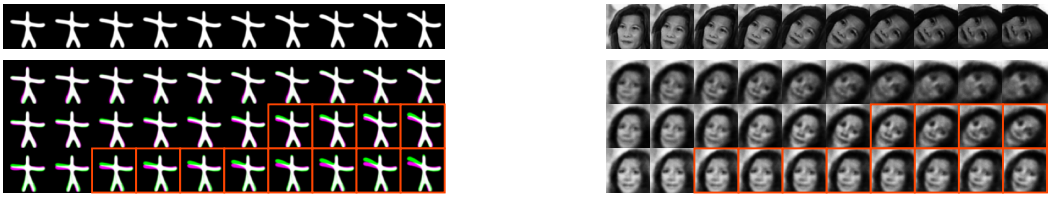


Figure 5.4: Extrapolation of different test input sequences for Starmen (left) and CelebA (right). The first two rows represent the ground truth and reconstructions (*ours*), respectively. Red squares highlight images that were not provided to the model. Deviation from the true test Starmen image is presented in color.

Trajectory Extrapolation In this section, we investigate whether the proposed model is able to extrapolate realistic trajectories from existing input data. To do so, we use the same model as before, but only provide the model with an image sequence of varying length and assess its ability to reconstruct either the next or the last image in the sequence. Fig. 5.3b and Fig. 5.3d show the MSE and SSIM, respectively, of the ground truth and the extrapolated images based on a varying input sequence length. It can be seen that extrapolations become more reliable when a longer input

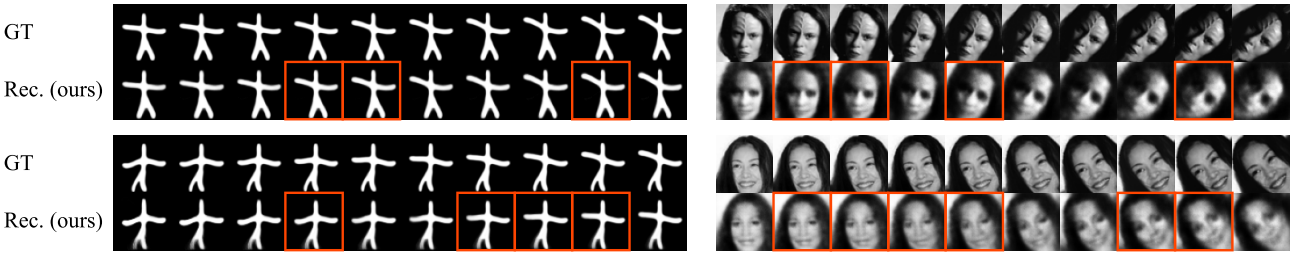


Figure 5.5: Data imputation in test sequences with 50% missing data after t_0. Top rows show ground truth trajectories, red squares represent imputed images.

sequence is given. This can also be observed from the visuals in Fig. 5.4, which show larger deviations from the ground truth when fewer images are presented. This experiment shows that in each case the model is able to estimate the progression: the left arm of the Starmen is raising and the CelebA head rotates, becomes bigger and contrast changes as expected. However, the model seems to underestimate the trajectory velocity as the input sequence becomes shorter. This aspect could potentially be mitigated by training using sequences of different lengths.

Data Imputation We validate the ability of the model to impute missing data using input sequences simulating partial patient follow-ups. We simulate this by removing 50% of the training, validation, and test data acquired after t_0 using the Starmen and CelebA datasets. The VAE is trained using the 50% available images, after which the generative model learns to map the features onto a linear trajectory. In Fig. 5.5 we show the reconstructed samples at observation times.

Trajectory Generation We also demonstrate that the proposed model can generate synthetic trajectories. We consider two cases: generating possible trajectories for a single image acquired at t_0 and generating a fully synthetic trajectory based on a synthetic image at t_0. In the first case, we first recover λ_i by encoding the real image using the VAE, estimate its value using the generative model and then sample η and τ from their priors as described in Section 5.2.2 and Appendix 5.6.2. In the second case, we first generate a synthetic λ and sample η and τ as aforementioned. To demonstrate the differences in these parameters, Fig. 5.6 shows trajectories obtained with varying delay τ (a) and velocity η (b), possible trajectories from an input image (c) and fully synthetic trajectories (d). Real images are extracted from the test set and highlighted with blue frames. The results show that the proposed model allows to decorrelate spatial (λ) and time parameters (η and τ) since all images in a trajectory represent the same individual that undergoes smooth progressive change.

Neuroimaging Data Finally, we validate the ability of the model to generate Alzheimer's disease progression trajectories. Fig. 5.6e and 5.6f show trajectories generated from an existing input image and a synthetic image, respectively. The generated trajectories appear realistic because the ventricles grow over time, which is a marker of ageing and Alzheimer's disease progression (Nestor et al., 2008). Moreover, the proposed model seems to preserve the morphology represented at the first time point for both real and fake subjects. However, the generated disease progression trajectories still need to be assessed in more detail, for example by means of visual analysis by a medical expert or by training a deep learning-based classifier.

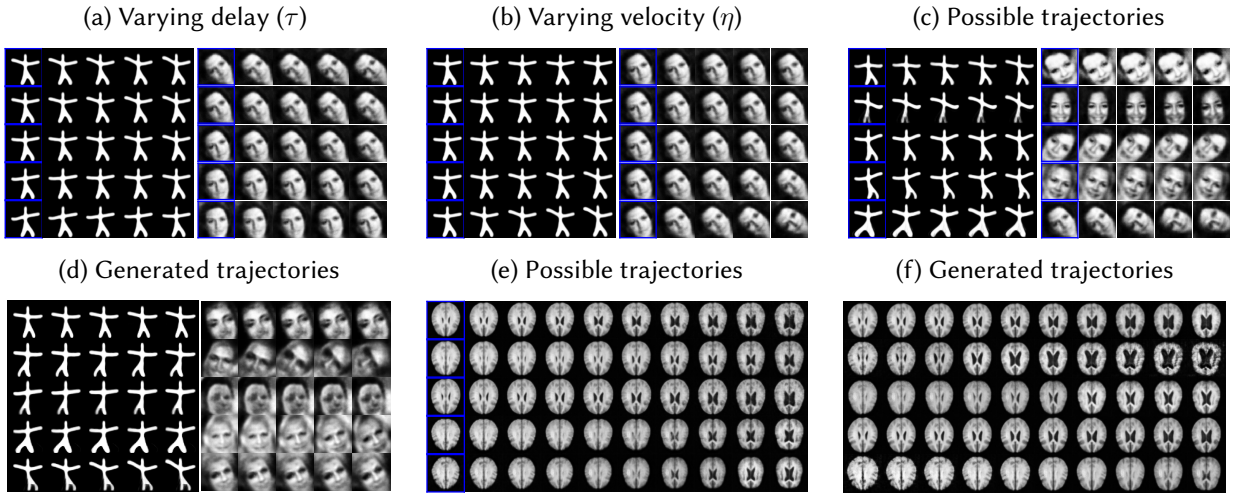


Figure 5.6: Synthetic trajectories derived from real images (indicated by blue frames): (a-c, e) or synthetic images (d, f).

5.5 Discussion and Conclusion

In this study we proposed a new continuous generative model capable of synthesizing longitudinal imaging data to perform trajectory extrapolation, data imputation and smooth and probable synthetic trajectory generation. A notable strength of our model lies in its *two-step* architecture, which allows substituting the VAE to make the model suitable for any data type, *e.g.* using clinical scores directly as features. We believe that this work is a step towards synthesis and augmentation of longitudinal medical datasets. Future work should focus on validating the ability of the model to perform reliable data augmentation for ML-based classification tasks or assess its relevance to perform treatment response analysis. Furthermore, the hypothesis of smooth trajectories should be put into perspective for diseases or patients showing abrupt changes in dynamics.

5.6 Appendices

5.6.1 Dataset details

Figure 5.7: Example trajectories for a training subject of Starmen (left), CelebA (middle), and ADNI (right). ADNI patients have a variable number of observations.

Table 5.1: Division of sets for 'Starmen' data.

SET	NUMBER OF SUBJECTS	NUMBER OF OBSERVATIONS
TRAINING SET	700	7,000
VALIDATION SET	200	2,000
TEST SET	100	1,000

Further details on the datasets are provided here.

Starmen It was only divided into a train (70%), a validation (20%) and a test set (10%) with no pre-processing.

CELEBA We explain below the proposed progression model applied to 'CelebA' dataset. First, we normalize the images between zero and one and apply a non-linear intensity transform, mimicking intensity differences in medical images for the same tissue between observations:

$$y_i(t_{10}) = \begin{cases} y_i(t_0)^{|\alpha|}, & \text{if } \alpha \geq 1 \\ y_i(t_0)^{|\frac{1}{\alpha}|}, & \text{if } \alpha < 1, \end{cases}$$

where $y_i(t_0)$ represents the original image, $y_i(t_{10})$ represents the last image in the longitudinal series and $\alpha = \alpha_{dir} \cdot \alpha_{max}$, with $\alpha_{dir} \sim \mathcal{U}(\{-1, 1\})$ and $\alpha_{max} \sim \mathcal{U}([1, 2])$. Secondly, we apply a linear growth factor mimicking e.g. weight gain, where the image size is multiplied with factor β, with $\beta \sim \mathcal{U}([1, 1.5])$. Thirdly, we apply a rotation of maximum $\gamma°$, with $\gamma \sim \mathcal{U}([-90, 90])$. Lastly, we crop the images to the original size, add Gaussian noise ($\sigma = 0.001$), and normalize the images between zero and one. The images from time points t_1 to t_{10} construct the full progression. We used a subset of 5,000 for training and 1,000 people for validation and an independent test set.

ADNI dataset The last dataset was obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). The ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial MRI, PET, other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment and early Alzheimer's disease. All scans were preprocessed by first applying N4 bias field correction (Tustison et al., 2010b) and registering the images to the MNI152 (1-mm isotropic resolution) standard space using FSL (Jenkinson and Smith, 2001; Jenkinson et al., 2002). Furthermore, we applied brain extraction using HD-BET (Isensee et al., 2019) and normalized the images between zero and one, where

Table 5.2: ADNI information per image (not per patient).

Set	Diagnosis	Gender (F/M/X)	Age
Training set	NC	1052/1037/0	76.2 ± 6.4
	MCI	1184/1839/0	75.1 ± 7.6
	AD	634/826/0	76.3 ± 7.4
Validation set	NC	128/150/0	76.9 ± 6.1
	MCI	152/251/1	73.3 ± 8.0
	AD	88/128/0	76.0 ± 8.3
Test set	NC	136/144/0	74.8 ± 6.2
	MCI	168/204/0	72.6 ± 7.9
	AD	84/112/0	76.2 ± 7.5

the 99.99^{th} percentile of voxel intensities within the brain mask were clipped to one. We used a total of 8318 MRI scans, obtained from 1799 patients, with an average of 4.6 ± 2.3 scans per patient. The average time between the first and the last scan was 2.9 ± 2.4 years. Lastly, we selected slice number 100 and cropped the slice to 182×182. The dataset was split into a training, validation and test set consisting of 1432 (6572), 184 (898), and 183 (848) subjects (images), respectively. All sets were balanced based on gender and Alzheimer's disease classification, information regarding the demographics can be found in Table 5.2.

5.6.2 Implementation details

ENCODER	$(1, 64, 64)$
LAYER 1	LINEAR$(4096, 512)$ RELU
LAYER 2*	LINEAR$(512, 8)$
DECODER	(8)
LAYER 1	LINEAR$(8, 512)$ RELU
LAYER 2*	LINEAR$(512, 4096)$ SIGMOID

*SAME FOR MEAN AND VARIANCE

ENCODER	$(1, 64, 64)$
LAYER 1	CONV$(128, 4, 2)$
LAYER 2	CONV$(256, 4, 2)$
LAYER 3	CONV$(512, 4, 2)$
LAYER 3	CONV$(1024, 4, 2)$
LAYER 4*	LINEAR$(16384, 64)$
DECODER	(64)
LAYER 1	LINEAR$(64, 16384)$
LAYER 2	CONVT$(512, 5, 2)$
LAYER 3	CONVT$(256, 5, 2)$
LAYER 4	CONV$(128, 5, 2)$
LAYER 5	CONVT$(1, 5, 1)$

*SAME FOR MEAN AND VARIANCE

ENCODER	$(1, 182, 182)$
LAYER 1	CONV$(64, 4, 2)$
LAYER 2	CONV$(128, 4, 2)$
LAYER 3	CONV$(256, 4, 2)$
LAYER 4	CONV$(512, 4, 2)$
LAYER 5	CONV$(1024, 4, 2)$
LAYER 6*	LINEAR$(36864, 8)$
DECODER	(8)
LAYER 1	LINEAR$(8, 36864)$
LAYER 2	CONVT$(512, 4, 2)$
LAYER 3	CONVT$(256, 4, 2)$
LAYER 4	CONVT$(128, 4, 2)$
LAYER 5	CONVT$(64, 4, 2)$
LAYER 6	CONVT$(1, 4, 2)$

*SAME FOR MEAN AND VARIANCE

Table 5.3: VAE neural networks for 'Starmen' (left), 'CelebA' (middle) and ADNI (right). Convolutional layers are followed by batch normalization and relu except the final layer of decoder where sigmoid is used.

MLP 1	(8)
LAYER 1	LINEAR$(8, 512)$ RELU
LAYER 2	LINEAR$(512, 128)$ RELU
LAYER 3*	LINEAR$(128, 7)$
RNN	(SEQUENCE LENGTH, 8)
LAYER 1	ELMAN NET
LAYER 2	LINEAR$(512, 128)$
LAYER 3*	LINEAR$(128, 1)$
MLP 2	(8)
LAYER 1	LINEAR$(8, 512)$ RELU
LAYER 2	LINEAR$(512, 8)$

*SAME FOR MEAN AND VARIANCE

MLP 1	(64)
LAYER 1	LINEAR$(64, 512)$ RELU
LAYER 2	LINEAR$(512, 128)$ RELU
LAYER 3*	LINEAR$(128, 63)$
RNN	(SEQUENCE LENGTH, 64)
LAYER 1	ELMAN NET
LAYER 2	LINEAR$(256, 128)$
LAYER 3*	LINEAR$(128, 1)$
MLP 2	(8)
LAYER 1	LINEAR$(64, 512)$ RELU
LAYER 2	LINEAR$(512, 64)$

*SAME FOR MEAN AND VARIANCE

MLP 1	$(8 + 1)$
LAYER 1	LINEAR$(8, 512)$ RELU
LAYER 2	LINEAR$(512, 128)$ RELU
LAYER 3*	LINEAR$(128, 7)$
RNN	(SEQUENCE LENGTH, 8)
LAYER 1	ELMAN NET
LAYER 2	LINEAR$(512, 128)$
LAYER 3*	LINEAR$(128, 1)$
MLP 2	(8)
LAYER 1	LINEAR$(8, 512)$ RELU
LAYER 2	LINEAR$(512, 8)$

*SAME FOR MEAN AND VARIANCE

Table 5.4: Generative model neural networks. Elman networks comprise 3 layers with tanh activation.

	VAE			Generative model		
Dataset	'Starmen'	'CelebA'	ADNI	'Starmen'	'CelebA'	ADNI
Epochs	50	100	200	1k	10k	50
Learning rate	10^{-3}	10^{-5}	10^{-5}	10^{-4}	10^{-4}	10^{-3}
Batch size	100	100	100	100	100	1
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam

Table 5.5: Training parameters. For each experiment the model that is selected in the one achieving the best loss on the validation set. In practice, a parameter $\beta = 0.1$ weighting the reconstruction and regularization in Eq. (5.2.2) applied for the generative model.

Chapter 6

Variational Inference for Longitudinal Data Using Normalizing Flows

This chapter introduces a new latent variable generative model able to handle high dimensional longitudinal data and relying on variational inference. The time dependency between the observations of an input sequence is modeled using normalizing flows over the associated latent variables. The proposed method can be used to generate either fully synthetic longitudinal sequences or trajectories that are conditioned on several data in a sequence and demonstrates good robustness properties to missing data. We test the model on 6 datasets of different complexity and show that it can achieve better likelihood estimates than some competitors as well as more reliable missing data imputation.

This chapter was submitted to the NeurIPS Conference 2023.

6.1 Introduction

Longitudinal data are more than common in many application fields such a medicine *e.g.* for disease progression modeling (Aghili et al., 2018; Zhao et al., 2021) or monitoring treatment response (Blackledge et al., 2014). They consist in the observation of a given entity's or individual's evolution though time but contrary to *time-series*, the number of observations of a single entity may be pretty small. Moreover, such data can be of high dimension (*e.g.* images) and we may only have access to a reduce number of different entities (*e.g.* rare diseases follow-ups) leading to small databases and missing values (*e.g.* a missing observation at a given time or loss in follow-up of a given entity). All of these aspects make these data challenging to model.

Generative models such as Variational Autoencoders (VAEs) introduced in (Kingma and Welling, 2014; Rezende et al., 2014) appeared to be powerful models to model distributions and would be an interesting choice to consider for longitudinal data. Unfortunately, while they appear to be able to perform some disentanglement of the input data in their latent space (Higgins et al., 2017; Burgess, 2018; Kim and Mnih, 2018; Chen et al., 2018b), they struggle to capture more complex correlations such as time evolution for longitudinal data (Ramchandran et al., 2021). To address this limitation and improve the latent representations of the input data, methods trying to account for the correlations of the data in the latent space of VAEs (Sohn et al., 2015), proposing new prior distributions (Nalisnick et al., 2016; Sønderby et al., 2016; Dilokthanakul et al., 2017; Tomczak and Welling, 2018; Razavi et al., 2020; Pang et al., 2020) or seeking to enhance the expressiveness of the approximate posterior distribution (Salimans et al., 2015; Rezende and Mohamed, 2015) were proposed. With a specific focus on temporal coherence, works introducing priors using Gaussian Processes were also introduced (Casale et al., 2018; Fortuin et al., 2020; Ramchandran et al., 2021). Nonetheless, those models were mainly designed to perform missing data imputation or for conditional settings and so are not well suited for unconditional sequence generation.

Focusing more specifically on medical applications, several works have analyzed longitudinal data through the prism of progression models using in particular mixed-effects models (Schiratti et al., 2015; Bône et al., 2018). In these approaches, patients are assumed to follow a given trajectory that deviates from a reference curve that may, for example, represent the average progression of a given disease. These approaches were then combined with dimensionality reduction using autoencoders (Louis et al., 2019) or VAEs (Sauty and Durrleman, 2022). However, these methods remain limited to the context of disease progression because they assume the existence of an intrinsic average trajectory from which each subject deviates, which may no longer be a valid assumption for heterogeneous datasets.

In this chapter, we take quite a different approach and propose the following contributions:

- We propose a new generative latent variable model imposing time dependency of the observations in an input sequence using normalizing flows on the associated latent variables. A training procedure relying on variational inference is also derived.
- We show that the model is capable of handling high dimensional longitudinal data and able to generate fully synthetic sequences or trajectories conditioned on several input data.
- We discuss the modularity of the proposed model and show that it can benefit pretty easily from improvements available in the variational inference literature.

- We show that the method achieves better likelihood estimates that competitors on benchmark datasets and can outperform them for missing data imputation.

6.2 Background

In this section, we first recall some elements on variational inference and normalizing flows needed in the proposed method.

6.2.1 Variational Inference

Given observations $x \in \mathbb{R}^D$ and associated latent variables $z \in \mathbb{R}^d$ with joint distribution $p(x, z)$, variational inference (Jordan et al., 1999) is a method that aims at approximating an untractable conditional distribution $p(z|x)$ of the latent variables given the observations using a family of parametrized distributions $q_\phi(z|x)$ (Blei et al., 2017). The idea is to find the set of parameters ϕ that minimizes the Kullback-Leibler (KL) divergence between the approximate posterior and the true one *i.e.*

$$\min_\phi \mathrm{KL}(q_\phi(z|x)||p(z|x)) \,.$$

However, this objective is most of the time untractable since $p(z|x)$ is unknown and so a surrogate objective is optimized instead and obtained using Jensen's inequality (Jordan et al., 1999):

$$\begin{aligned} \log p(x) = \log \int_{\mathbb{R}^d} p(x, z) dz = \log \mathbb{E}_{q_\phi} \left[\frac{p(x, z)}{q_\phi(z|x)} \right] , \\ \geq \mathbb{E}_{q_\phi} \log \left[\frac{p(x, z)}{q_\phi(z|x)} \right] . \end{aligned}$$

The right hand side of the equation is called the Evidence Lower BOund (ELBO) and one may notice that the difference between the left hand side of the equation and the ELBO gives $\mathrm{KL}(q_\phi(z|x)||p(z|x))$. Hence, maximizing the ELBO amounts to minimizing the KL and so the ELBO is used as objective for the variational approximation.

6.2.2 Normalizing Flows

Normalizing flows are flexible models that can be used to transform simple probability densities into much complex ones by re-coursing to sequences of invertible smooth mappings. They have, for instance, been proposed to enhance the expressiveness of the approximate posterior distribution used in the context of variational inference in (Rezende and Mohamed, 2015). These models rely on the rule of change of variables such that if $z \in \mathbb{R}^d$ is a random variable that follows the distribution $q(z)$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible smooth function, then the random variable $z' = f(z)$ has a distribution given by

$$q(z') = q(z) \left| \det \frac{\partial f^{-1}}{\partial z'} \right| = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} . \tag{6.1}$$

In this setting, f is called a *normalizing flow* and so several flows can be composed to form a new flow $g = f_K \circ f_{K-1} \circ \cdots \circ f_1$ allowing to model richer distributions. In the context of variational inference, these flows can be parameterized as well and so can be used to have access to enhanced approximate posterior distributions $q_\phi(z|x)$ provided that the computation of the det-Jacobian of the flows is tractable. Amongst the most widely known flows we can cite NICE (Dinh et al., 2014), linear and planar flows (Rezende and Mohamed, 2015), RealNVP (Dinh et al., 2016), Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017) or Inverse Autoregressive Flows (IAF) (Kingma et al., 2016).

6.3 The Proposed Model

In this section, we propose a new generative latent variable model suited for longitudinal data.

6.3.1 Problem Setting

Let us define P as the number of entities observed through time. For each entity $i \in \{1, \dots, P\}$, we are given a sequence of $t_i + 1$ observations $x^i = (x^i_0, \dots, x^i_{t_i})$ such that $x^i_j \in \mathcal{X} = \mathbb{R}^D$, $\forall j \in \{0, \dots, t_i\}$. Assuming that the sequence x^i is generated from an unknown distribution p, our goal is to infer p with a parametric model $\{p_\theta, \theta \in \Theta\}$.

6.3.2 The Probabilistic Model

Given an entity $i \in \{1, \dots, P\}$ and a sequence of observation $(x^i_0, \dots, x^i_{t_i})$, we assume that for each x^i_j where $j \in \{0, \dots, t_i\}$, there exists an associated latent variable $z^i_j \in \mathcal{Z} = \mathbb{R}^d$ involved in the generative process of the observation x^i_j such that $x^i_j \sim p_\theta(x^i_j|z^i_j)$. One may further write the joint distribution p_θ as follows:

$$p_\theta(x^i_0, \cdots, x^i_{t_i}) = \int_{\mathcal{Z}} p_\theta(x^i_0, \cdots, x^i_{t_i}|z^i_j) p(z^i_j) dz^i_j \,, \tag{6.2}$$

where $p(z^i_j)$ is a prior distribution over z^i_j. An important point in this setting is that the observations x^i_j are no longer independent and so the joint likelihood is no longer factorisable. In this chapter, we propose to model the time dependency of the observations in an input sequence using the latent variables and normalizing flows as follows

$$z^i_0 \sim p(z^i_0), \; z^i_1 = f_1(z^i_0), \; \dots \; , z^i_{t_i} = f_{t_i}(z^i_{t_i - 1}) \,, \tag{6.3}$$

where p is a simple prior distribution over z^i_0 (*e.g.* standard Gaussian) and f_j are normalizing flows for any $j \in \{1, \dots, t_i\}$. The main idea is to assume that it is the distribution of the latent variables that evolves through time and we propose to model this evolution using the flows. As such, the time dependency is imposed on the latent variables and not directly on the observations. Note that the initial distribution can be chosen as complex as desired and that for any $j \in \{1, \dots, t_i\}$ we have access to a tractable density for $p(z^i_j)$ using Eq. (6.1):

$$p(z^i_j) = p(z^i_0) \prod_{l=1}^{j} \left| \det \frac{\partial f_l}{\partial z^i_{l-1}} \right|^{-1} . \tag{6.4}$$

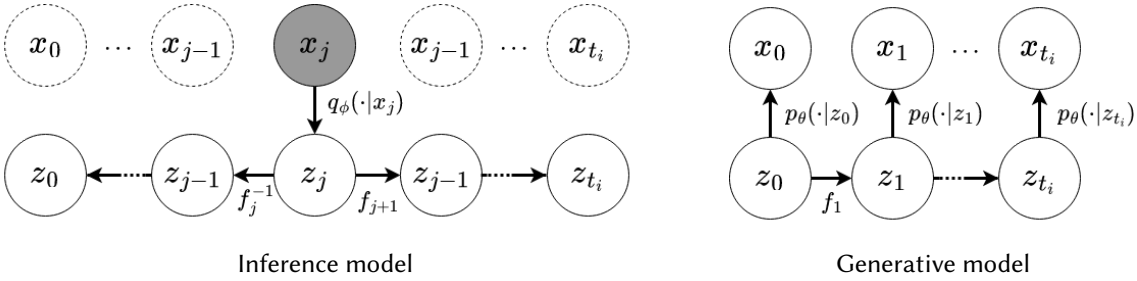


Figure 6.1: Proposed inference and generative models.

In addition, the relation between two latent variables z_j^i and z_k^i with $j, k \in \{0, \dots, t_i\}$ such that $j < k$ is explicit and completely deterministic since we have:

$$z_k^i = \bigcirc_{l=j+1}^{k} f_l(z_j^i) \text{ and } z_j^i = \bigcirc_{l=k}^{j+1} (f_l)^{-1}(z_k^i)\,. \tag{6.5}$$

Hence, we can see that given a latent vector z_j^i we can now retrieve the complete sequence $(z_0^i, \dots, z_{t_i}^i)$ using Eq. (6.5). Assuming $(x_j^i)_{j\in\{1,\dots,t_i\}}$ are independent knowing $(z_0^i, \dots, z_{t_i}^i)$, the conditional distribution in Eq. (6.2) writes

$$p_\theta(x_0^i, \cdots, x_{t_i}^i | z_j^i) = \prod_{l=0}^{t_i} p_\theta(x_l^i | z_j^i) = \prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i)\,. \tag{6.6}$$

Using Eq. (6.4) and Eq. (6.6) allows to derive another expression of the joint distribution of the observations:

$$p_\theta(x_0^i, \cdots, x_{t_i}^i) = \int_{\mathcal{Z}} \prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i) p(z_j^i) dz_j^i\,.$$

Since this integral is most of the time intractable, we propose to rely on variational inference (Jordan et al., 1999). We indeed introduce a parametrized variational distribution $q_\phi(z_j^i | x_j^i)$ such that we can obtain an unbiased estimate of the joint likelihood:

$$\mathbb{E}_{q_\phi}\left[\frac{\prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i) p(z_j^i)}{q_\phi(z_j^i | x_j^i)}\right] = p_\theta(x_0^i, \cdots, x_{t_i}^i)\,.$$

Using Jensen's inequality allows to derive a lower bound (ELBO) on the true objective *i.e.* the log joint likelihood :

$$\begin{aligned}
\log p_\theta(x_0^i, \cdots, x_{t_i}^i) &= \log \mathbb{E}_{q_\phi}\left[\frac{\prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i) p(z_j^i)}{q_\phi(z_j^i | x_j^i)}\right], \\
&\geq \mathbb{E}_{q_\phi} \log\left[\frac{\prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i) p(z_j^i)}{q_\phi(z_j^i | x_j^i)}\right], \\
&\geq \mathbb{E}_{q_\phi} \log \prod_{l=0}^{t_i} p_\theta(x_l^i | z_l^i) - \mathrm{KL}(q_\phi(z_j^i | x_j^i) | p(z_j^i))\,.
\end{aligned} \tag{6.7}$$

Algorithm 6 Training Procedure

Input: Observations $(x_0^i, \cdots, x_{t_i}^i)$
while not converged **do**
  Pick $j \in \{0, \dots, t_i\}$ randomly
  $z_j^i \sim q_\phi(\cdot|x_j^i)$
  for $l = j+1$ **to** t_i **do**
    $z_l^i = f_l(z_{l-1})$ ▷ propagate in future
  end for
  for $l = j-1$ **to** 0 **do**
    $z_l^i = (f_{l+1})^{-1}(z_{l+1})$ ▷ propagate in past
  end for
  $\mathcal{L} = -\frac{1}{t_i+1} \sum_{l=0}^{t_i} \log p_\theta(x_l^i|z_l^i) + \log q_\phi(z_j^i|x_j^i) - \log p(z_0^i) - \sum_{l=1}^{j} \log \left| \det \frac{\partial (f_l)^{-1}}{\partial z_l^i} \right|$
end while

The graphical models for the proposed method can be found in Fig. 6.1. In practice and inspired from the VAE framework, the variational distribution is chosen as a multivariate Gaussian distribution $q_\phi(z_j^i|x_j^i) = \mathcal{N}(z_j^i; \mu_\phi(x_j^i), \Sigma_\phi(x_j^i))$ for $j \in \{0, \dots, t_i\}$ and where μ_ϕ and Σ_ϕ are given by neural networks and Σ_ϕ is chosen as a diagonal matrix. The conditional distributions $p_\theta(x_j^i|z_j^i)$ are chosen depending on the input data (*e.g.* multivariate Gaussians for RGB images) and $p(z_j^i)$ is given by Eq. (6.4). To mitigate the impact of the sequence length on the ELBO in Eq. (6.7), we average the left hand side term over the sequence length. This impedes the reconstruction term to over-weight the KL for long sequences. As for the normalizing flows, in this chapter, we use Inverse Autoregressive Flows (IAF) (Kingma et al., 2016) with MADE (Germain et al., 2015) for the autoregressive networks since we need a tractable inverse. It should be noted that for such flows the computation of the inverse is however sequential and its time proportional to the dimensionality of the latent variables due to the autoregressive property of the flows. Nonetheless, in practice the dimension of the latent variables is often much smaller than the dimensionality of the input data making this choice reasonable. We choose IAF over MAF (Papamakarios et al., 2017) so that the generation of a synthetic sequence from the prior $z_0 \sim p(z_0)$ is fast since it does not require inverting the flows. Finally, a pseudo code of the training algorithm is provided in Alg. 6 and an implementation using PyTorch (Paszke et al., 2017) and based on (Chadebec et al., 2022c) is made available in the supplementary materials.

6.3.3 Dealing with Missing Data in the Sequence

In *real-life* applications, it is not rare to find sequences with missing observations (*e.g.* in medicine a loss of patient follow-up or a patient not coming to a specific visit induces missing observations for the patient's evolution). As explained above and shown in Alg. 6, during training we perform variational inference using only one element in the sequence. Thus, the training can be modified pretty easily to handle such missing data in the input sequences and consists in only using the observed data.

Nonetheless, this can be seen as a weakness of the method at inference time. Let us indeed

Algorithm 7 Inference Procedure for Missing Observations

Input: A sequence $(x^i_j)_{j\in\mathcal{O}_i}$ with missing observations
for $j \in \mathcal{O}_i$ **do**
  $z^i_{j,j} \sim q_\phi(\cdot|x^i_j)$
  $\hat{x}^i_{j,j} \sim p_\theta(\cdot|z^i_{j,j})$
  for $l = j+1$ **to** t_i **do**
    $z^i_{l,j} = f_l(z_{l-1,j})$
    $\hat{x}^i_{l,j} \sim p_\theta(\cdot|z^i_{l,j})$
  end for
  for $l = j-1$ **to** 0 **do**
    $z^i_{l,j} = (f_{l+1})^{-1}(z_{l+1,j})$
    $\hat{x}^i_{l,j} \sim p_\theta(\cdot|z^i_{l,j})$
  end for
end for
$j_{\text{opt}} = \arg\max_{j\in\mathcal{O}_i} \sum_{l\in\mathcal{O}_i} \log p_\theta(x^i_{l,j}|z^i_{l,j})$
return $(\hat{x}^i_{0,j_{\text{opt}}}, \cdots, \hat{x}^i_{t_i,j_{\text{opt}}})$ ▷ obtained with j_{opt}

imagine that we are given a sequence of 5 measure times $(x^i_0, x^i_1, x^i_2, x^i_3, x^i_4)$ where only 3 are actually observed, say x^i_1, x^i_2 and x^i_4. In its current shape, during inference, the method will choose an observation time $j \in \{1, 2, 4\}$, say $j = 2$, sample a latent variable associated to observation x^i_2 using the approximate posterior $q_\phi(\cdot|x^i_2)$ and then generate a sequence $(z^i_l)_{l\in\{0,\dots,4\}}$ using the learned flows. This sequence is then used to sample a reconstructed sequence in the observations space using $p_\theta(x|z)$. This is actually sub-optimal since this would be equivalent to only have access to observation x^i_2 without benefiting from the information provided by x^i_1 and x^i_4. In order to address this potential limitation of the model, we propose to generate a sequence (actually we can generate an arbitrary number of sequences) for each index corresponding to an observed input data in the sequence (*i.e.* $\{1, 2, 4\}$ in the example) and keep the generated sequence achieving the highest likelihood on the observed data. In other words, if we denote $\mathcal{O}_i$ the set of observed indices in the input sequence, we define the *optimal* index that should be used to complete the sequence as follows:

$$j_{\text{opt}} = \arg\max_{j\in\mathcal{O}_i} \sum_{l\in\mathcal{O}_i} \log p_\theta(x^i_{l,j}|z^i_{l,j}) \,, \tag{6.8}$$

where $z^i_{l,j}$ is the latent variable and $x^i_{l,j}$ the data generated at time l using index j. Alg. 7 shows the inference procedure.

6.3.4 Enhancing the Model

One interesting aspect of the model is that one may use improvements that have been proposed and proved useful in the literature related to variational inference and VAEs to enhance several part of the model independently.

Improving the prior Even-though, a simple distribution such as a standard Gaussian appeared to work well in practice, a smarter choice in the prior distribution may result in an enhanced data generation or better likelihood estimates (Hoffman and Johnson, 2016). As such, richer priors (Nalisnick et al., 2016; Dilokthanakul et al., 2017) or priors that are learned (Chen et al., 2016b; Razavi et al., 2020; Pang et al., 2020; Aneja et al., 2020) can be easily plugged into our model. We show in the experiments section how changing the prior from a standard Gaussian to a VAMP prior (Tomczak and Welling, 2018) can influence the results.

Improving the variational bound Following (Rezende and Mohamed, 2015) insights, another way to improve the expressiveness of the model and ideally achieve a tighter ELBO consists in enriching the potentially too simplistic parameterized variational distribution $q_\phi(z|x)$ using flows. This improvement can be easily integrated within our framework as well. In the experiments section, we also propose a variant model where the posterior distributions are enriched using IAF flows as proposed in (Kingma et al., 2016). Methods proposing to use MCMC sampling steps with learned Markov kernels (Salimans et al., 2015) or relying on Hamiltonian dynamics (Caterini et al., 2018; Chadebec et al., 2022b) could also be envisioned but are not tested in combination with our model due to the strong computation burden they imply.

6.4 Related Works

Variational Autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) share some aspects with our method. First, they try to maximize the likelihood of a set of data using a variational approach. Second, they try to take advantage of the flexibility a latent space provides by mapping potentially high dimensional input data into a lower dimensional space. However, they assume that the input data are independent and so are the latent variables. This impedes the model to capture the potentially complex time dependency that exists with longitudinal data.

There exist however some works on VAEs that are worth citing since they stress the flexibility offered by considering a latent space. In particular, they motivated our idea to impose the time dependency over the latent variables and not directly on the observations. First, several papers argued that the latent space of the VAE can reveal representative and interpretable features through its ability to perform disentanglement (Higgins et al., 2017; Burgess, 2018; Korkinof et al., 2018; Chen et al., 2018b). Studying the latent space with a geometric point of view, other works showed that this latent space can actually be modeled with a specific geometry (*e.g.* hyper-sphere, Poincaré Disk, Riemannian manifold) (Davidson et al., 2018; Falorsi et al., 2018; Mathieu et al., 2019a; Kalatzis et al., 2020; Chadebec et al., 2022b) or that a Riemannian geometry can naturally arise in the latent space (Arvanitidis et al., 2018; Chen et al., 2018a; Shao et al., 2018; Chadebec and Allassonnière, 2022). Finally, another way to enhance the representation capability of the model that was proposed in the literature consists in increasing the expressiveness of the variational posterior distribution using MCMC sampling (Salimans et al., 2015) or normalizing flows (Rezende and Mohamed, 2015).

Arguing that VAEs still fail to capture complex correlations, there were some proposals in the literature trying to constraint the model to account for these correlations in the latent space. For instance, the conditional VAE (Sohn et al., 2015) feeds auxiliary variables directly to the encoder and decoder networks but fails to model the evolution of a given subject through time. Gaussian

processes that are a powerful tool for time series (Seeger, 2004; Roberts et al., 2013) were also proposed as prior for the VAE (Casale et al., 2018; Fortuin et al., 2020) to account for the temporal structure across the samples. (Ramchandran et al., 2021) enriched these models with the inclusion of covariates different from time using a multi-output additive Gaussian process prior.

Modeling longitudinal data and trying to understand the underlying evolution dynamic is something that has also been quite studied under the prism of disease progression modeling (Jedynak et al., 2012; Fonteijn et al., 2012). In such literature, *mixed-effect* models (Laird and Ware, 1982) that parameterize a patient's evolution as a deviation from a reference trajectory have become more and more popular (Diggle et al., 2002; Singer et al., 2003). First applied on Euclidean data (Bernal-Rusiel et al., 2013), they were then extended with a Riemannian geometry viewpoint (Schiratti et al., 2015; Singh et al., 2016; Koval et al., 2017; Bône et al., 2018) or combined with dimensionality reduction (Louis et al., 2019; Sauty and Durrleman, 2022). Despite being adapted to model disease progression, it is unclear how these models would apply to datasets where there is no clear *average* evolution.

Finally, deep learning based methods relying on recurrent neural networks are also worth citing as they revealed useful for time varying data (Pearlmutter, 1989). To cite a few, GRUI-GAN (Luo et al., 2018) and BRITS (Cao et al., 2018) were proposed with the aim of handling missing data but with the drawback of relying on adversarial training for the first one and not being generative for the second. (Chung et al., 2015) proposed a combination of VAE and RNN for structured sequential data but there exists no clear way how the model would handle missing data.

6.5 Experiments

In this section, we validate the proposed method through series of experiments. We place ourselves in the context of high-dimensional data (images) and so set $d \ll D$ (*i.e.* the latent variables live in a much lower-dimensional latent space when compared to the input images size). In line with the VAE framework, the inference network providing the parameters of the variational distribution $q_\phi(z|x)$ can then be interpreted as an *encoder* and the generative model $p_\theta(x|z)$ as a *decoder*. Note that neither the *encoder* nor the *decoder* depend on time. First, we show that the proposed model is able to achieve better joint likelihood estimates than several models proposed in the literature on 5 datasets. Then, we show that the method is also able to impute missing data (and features) and compare its performances in term of reconstruction with benchmark models. Finally, we evaluate the ability of the proposed model to generate relevant conditioned and fully synthetic sequences. We also conduct in Appendix 6.7.4 an ablation study stressing the influence of the flows, latent space dimension and prior complexity and discuss the relevance of Eq. (6.8) for missing data imputation in Appendix 6.7.5.

6.5.1 Data

For these experiments, we consider 5 different databases that mimic longitudinal datasets. The first one shown in Fig. 6.2a is a synthetic longitudinal dataset composed of 1,000, 64x64 images of *starmen* raising their left arm and generated according to the diffeomorphic model of (Bône et al., 2018). The second one shown in Fig. 6.2b consists of 8 evenly separated rotations applied to the MNIST database (LeCun, 1998) from 0 to 360 degrees, we call it *rotMNIST*. In addition to these two *toy*

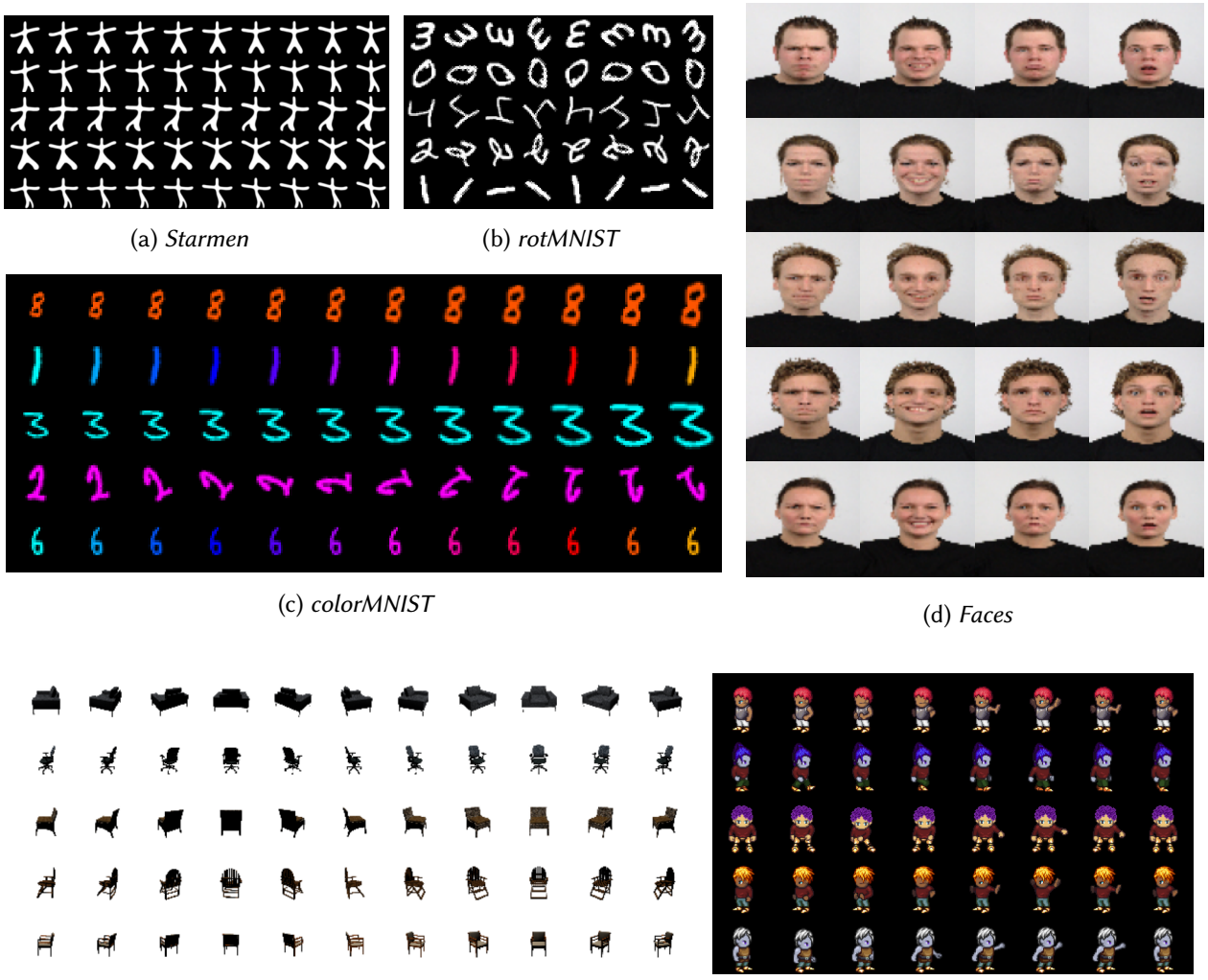
Table 6.1: Negative log joint likelihood divided by the sequence length computed on an independent test set with 5 independent runs and 100 importance samples.

Model	Starmen	RotMNIST	ColorMNIST	3d chairs	Sprites
VAE	3781.82 ± 0.01	741.03 ± 0.00	2179.88 ± 0.00	$\mathbf{11359.39 \pm 0.02}$	11313.38 ± 0.00
VAMP	3780.99 ± 0.01	740.82 ± 0.00	2179.60 ± 0.00	11361.01 ± 0.02	11313.43 ± 0.00
TVAE (0)	3806.26 ± 0.07	748.18 ± 0.00	2185.40 ± 0.00	11419.32 ± 0.11	11332.09 ± 0.01
TVAE (short)	3782.39 ± 0.05	744.07 ± 0.00	2175.54 ± 0.00	11373.76 ± 0.07	11318.96 ± 0.01
TVAE (part)	3780.75 ± 0.05	739.53 ± 0.00	2174.19 ± 0.00	11364.21 ± 0.18	11308.40 ± 0.01
TVAE (half)	3777.57 ± 0.07	745.55 ± 0.00	2173.58 ± 0.00	11363.27 ± 0.12	11305.62 ± 0.01
BubbleVAE	3780.46 ± 0.07	742.66 ± 0.00	2174.74 ± 0.00	11369.59 ± 0.19	11310.69 ± 0.01
GPVAE (Cauchy)	3780.36 ± 0.03	740.05 ± 0.00	2177.21 ± 0.00	11367.43 ± 0.02	11309.11 ± 0.01
GPVAE (rbf)	3787.80 ± 0.03	745.58 ± 0.00	2187.15 ± 0.00	11390.33 ± 0.04	11315.68 ± 0.01
GPVAE (diffusion)	3780.96 ± 0.01	740.26 ± 0.00	2178.63 ± 0.00	$\mathbf{11359.21 \pm 0.00}$	11312.50 ± 0.00
GPVAE (matern)	3779.29 ± 0.02	739.68 ± 0.00	2176.63 ± 0.00	11360.36 ± 0.03	11309.90 ± 0.00
Ours ($\mathcal{N}$)	3773.23 ± 0.17	735.71 ± 0.00	2173.16 ± 0.05	11362.00 ± 0.62	11301.51 ± 0.04
Ours (VAMP)	$\mathbf{3772.91 \pm 0.16}$	736.15 ± 0.00	2173.00 ± 0.05	11364.73 ± 0.51	$\mathbf{11301.30 \pm 0.02}$
Ours (IAF)	$\mathbf{3773.01 \pm 0.17}$	$\mathbf{735.27 \pm 0.01}$	$\mathbf{2172.85 \pm 0.05}$	$\mathbf{11359.48 \pm 0.67}$	11301.97 ± 0.02

datasets, we also consider more challenging ones. The third one called *colorMNIST* is created using the approach of (Keller and Welling, 2021). It consists of sequences of colored MNIST digits that can undergo three distinct types of transformations: color change (from turquoise to yellow), scale change or rotations and is presented in Fig. 6.2c. It is important to note that for this database, the time dynamic cannot be fully recovered from a single image since it can correspond to different transformations. For instance, a starting turquoise *6* can either change of color or undergo a change in scale as shown on line 2 and 3 of Fig. 6.2c. The fourth database is created using the *3d chairs* dataset (Aubry et al., 2014) consisting of 3D CAD chair models and considering as input sequences, 11 evenly separated rotations of a chair (from 0 to 360°). Some samples are displayed in Fig. 6.2e. We also use the *sprites* dataset (Li and Mandt, 2018) shown in Fig. 6.2f consisting of 64x64 RGB images of characters performing actions such as dancing or walking. Finally, we also consider the Radboud Faces Database consisting of 67 individuals expressing different emotions (Langner et al., 2010). For this dataset, we create sequences of 4 time steps corresponding to the emotions: *anger*, *happiness*, *sadness* and *surprise*; and down-sample the images so they are of size 64x64 as shown in Fig. 6.2d.

6.5.2 Likelihood Estimation

First, we compare the proposed model and two of its variants (using either a VAMP prior or IAF flows to enrich the posterior approximation) to a vanilla VAE (Kingma and Ba, 2014), a VAE with a VAMP prior (Tomczak and Welling, 2018) and models incorporating temporal coherence such as the BubbleVAE (Hyvärinen et al., 2004) or the Topographic VAE (TVAE) (Keller and Welling, 2021). For the latter, we consider several models with different temporal coherence length L: TVAE (0) *i.e.* no temporal coherence, TVAE (short) *i.e.* $L \approx \frac{1}{8}$ of the input sequence length S, TVAE (part) where $L \approx \frac{1}{4}S$ and TVAE (half) with $L = \frac{1}{2}S$, *i.e.* the model takes into account the full sequence. We also compare our model to a VAE using a Gaussian Process as prior (GPVAE) proposed in (Fortuin et al., 2020). For this model, we consider several GP kernels (RBF, Cauchy, diffusion and matern). We use the 5 first datasets presented in the previous section and train all the models on a train set, keep the



(a) *Starmen* (b) *rotMNIST*

(c) *colorMNIST* (d) *Faces*

(e) *3d chairs* (f) *Sprites*

Figure 6.2: 5 training sequences for each dataset considered in the chapter.

best model on a validation set and compute the negative log joint-likelihood on an independent test set using 100 importance samples in a similar fashion as (Burda et al., 2016). We train the models for 200 epochs for *sprites* and *rotMNIST*, 250 for *colorMNIST* and 400 for *starmen* and *chairs* with a latent dimension set to 16 for all datasets but for the *3d chairs* dataset where it is set to 32. Any other relevant piece of information about training configurations is provided in Appendix 6.7.3. We show in Table 6.1, the mean and standard deviation of the negative log joint likelihood obtained with 5 independent runs. For all datasets, the model is able to either compete or outperform the competitors. Moreover, as expected, using a richer prior (VAMP) or enriching the expressiveness of the variational posterior with flows (IAF) leads most of the time to a better likelihood estimation. This is an encouraging aspect since it shows that the model can be improved pretty easily using independent bricks available in the variational inference literature.

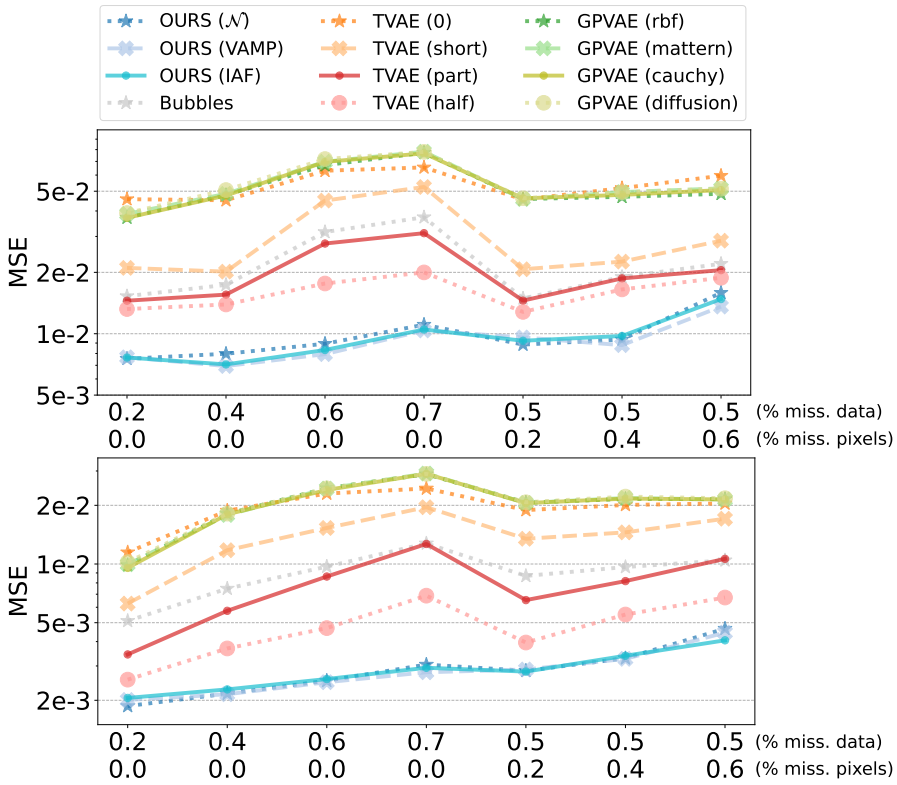


Figure 6.3: Mean Square Error (MSE) on the test data for different proportions of missing observations (0.2 to 0.7) and missing pixels (0.2 to 0.6) in the input train, validation and test sequences for the *starmen* (top) and *sprites* (bottom) datasets. The proposed model appears very robust to incomplete sequences thanks to the flows-based structure.

6.5.3 Missing Data Imputation

The second experiment that we conduct consists in assessing the robustness of the model when it faces missing data and test its ability to impute missing values. To do so we consider 2 databases: *starmen* and *sprites*; and randomly remove observations in input sequences with probability 0.2, 0.4, 0.6 and 0.7. To challenge the model in the context of missing features, we also create sequences with missing observations (randomly removed with probability 0.5) and missing pixels in the observed images (randomly removed with probability 0.2, 0.4 and 0.6). All the models are trained with the same masks and are optimized using an objective computed only on the seen pixels. The charts in Fig. 6.3 show the Mean Square Error (MSE) obtained on an independent test set. In all scenarios, the proposed model outperforms the TVAEs, BubbleVAEs and GPVAEs and appears as expected quite robust to missing observations in the input sequences. This is made possible thanks to the training structure that uses only one seen observation to perform variational inference.

In Fig. 6.4, we also show some conditional generations obtained with the proposed model on the *colorMNIST* dataset. At the top, we show 5 generated trajectories using 2 different images. In each case, we draw 5 random latent variables from the corresponding variational posterior $q_\phi(z|x)$. They are then passed through the flows according to Eq. (6.3) leading to 5 sequences and finally decoded using $p_\theta(x|z)$. In a), the model is able to produce a range of possible evolutions (changes of color

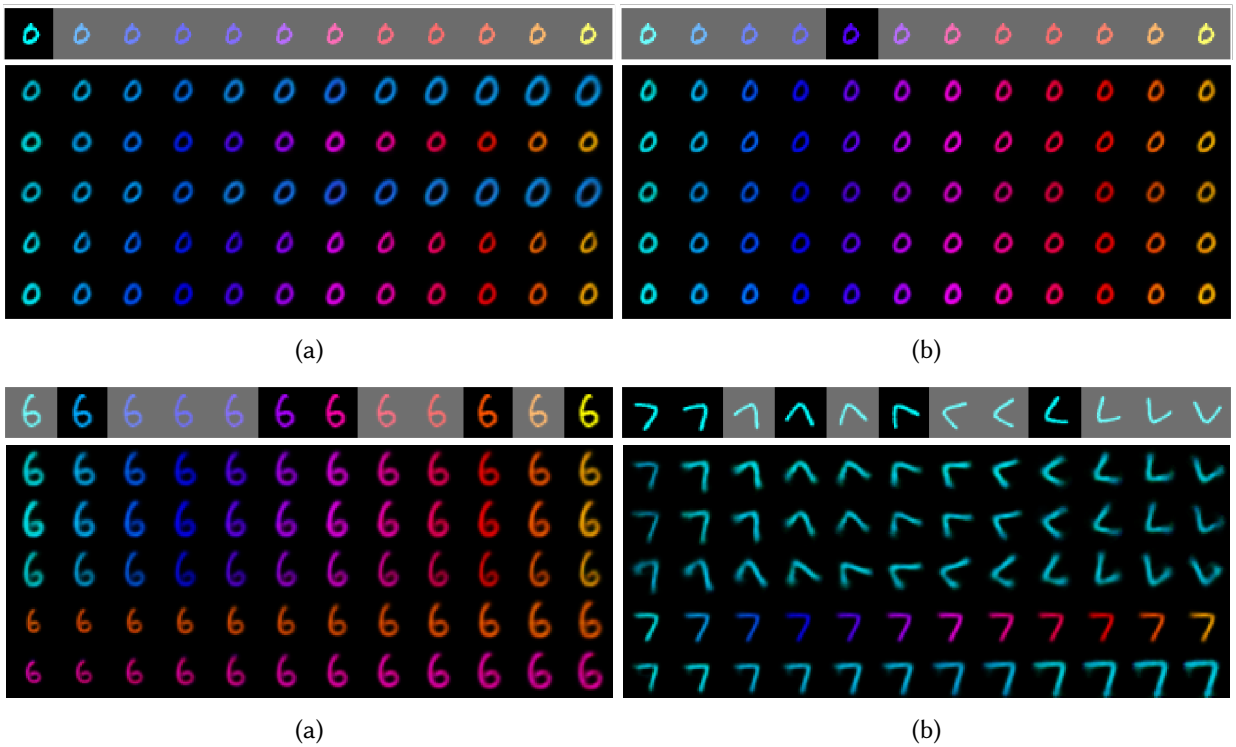


Figure 6.4: Conditionally generated trajectories (greyed are unseen data). *Top:* 5 generated sequences using the same input image. For each trajectory, 5 latent variables are drawn from the posterior distribution $q_\phi(z|x)$, passed trough the flows and decoded using $p_\theta(x|x)$. In a), the model is able to produce possible evolutions (changes of color or scale) for the dataset considered. *Bottom:* Generated sequences using each seen data in the input sequence. The generated sequences are ranked as they maximize the likelihood on the seen data according to Eq. (6.8) (best at the top).

or scale) that are plausible given the dataset considered. This is a very important property of the model since thanks to the variational posterior distribution it can generate an infinite number of possible trajectories from a single observation. Moreover, we see that the model is clearly able to keep the shape coherence all along the trajectory. At the bottom, we show the sequences obtained by using each image available in the sequence (not greyed). We rank the generated trajectories as they maximize the likelihood on the seen data (*i.e.* according to Eq. (6.8)). This experiment shows how the model can benefit from the information available in the sequence despite only using one image to generate. In practice, one may generate as many trajectories as desired for each image available in the sequence (and not just one as in this example) and choose the one that maximizes Eq. (6.8). As a conclusion, these experiments show that even-though the relation between the latent variables of a sequence is deterministic, the stochasticity in the conditionally generated sequences arises from the sampling from the variational posterior that is able to capture the modularity of the data.

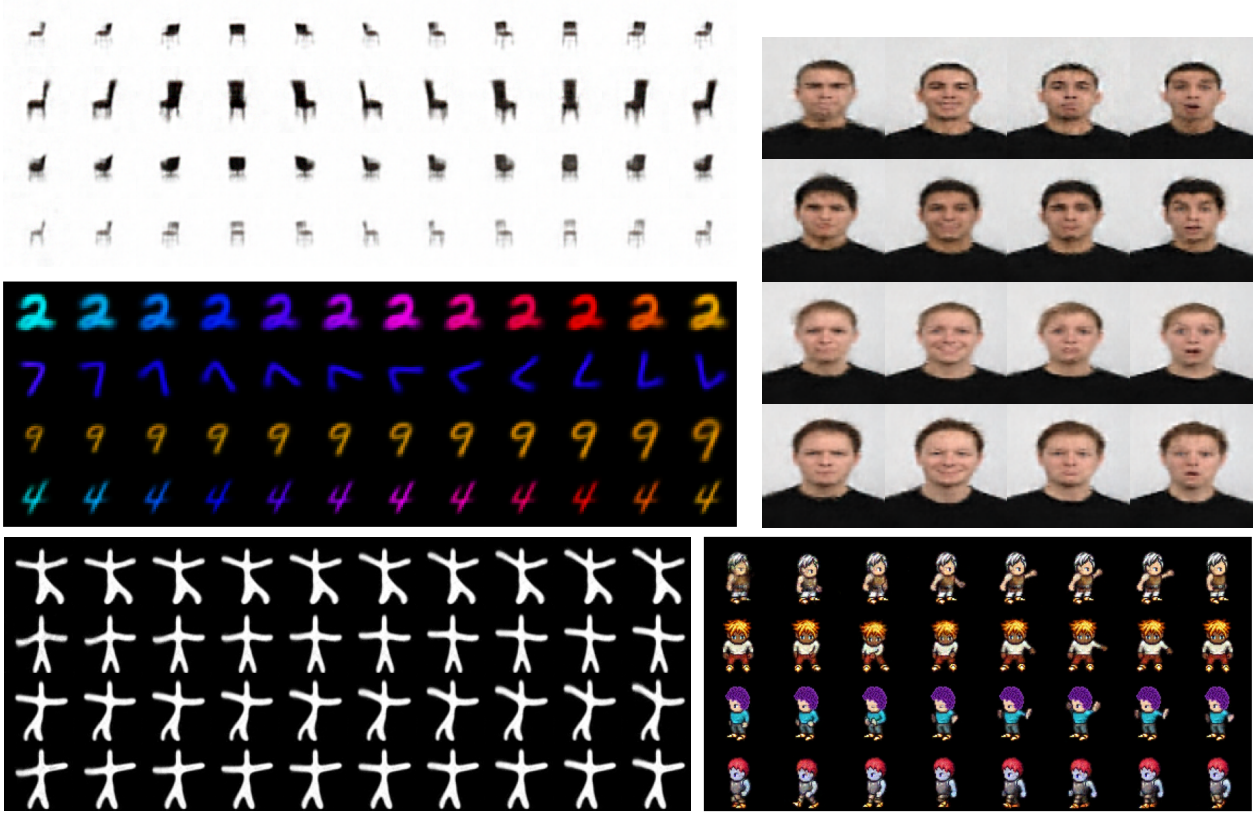


Figure 6.5: Generated sequences using the proposed model. Latent variables are sampled from the prior distribution (taken as a standard Gaussian in this example) and propagated through the flows according to Eq. (6.3). The obtained latent sequences are then decoded using the conditional distribution $p_\theta(x|z)$ to create the image sequences.

6.5.4 Unconditional Sequence Generation

In this section, we evaluate the ability of the proposed model to generate relevant fully synthetic trajectories. For this experiment, we first compute the Frechet Inception Distance (FID) (Heusel et al., 2017) on the *colorMNIST* and *sprites* datasets. The FID is computed by generating the same number of images as available in an independent test set: 21,312 for *sprites* (2,664 sequences of 8 time steps) and 120,000 for *colorMNIST*. Note that in this setting the FID does not account for the temporal coherence between the generated samples within a sequence. As shown in Table 6.2, the proposed model achieves the lowest FID (lower is better). The fact that it is able to outperform a VAE or a VAMP-VAE shows that the temporal coherence constraint imposed by the flows does not affect the quality of the generated images. Moreover, we see the influence of using a more complex prior or enriching the variational approximation on the generative capability of the model that can achieve better FIDs. Finally, we show generated samples for the *3d chairs*, *starmen*, *sprites*, *colorMNIST* and the Radboud Faces Database. We show 4 generated sequences for each dataset in Fig. 6.5. Thanks to the flow-based structure, the model is able to generate relevant sequences that clearly keep a temporal consistency. Additional samples can be found in Appendix 6.7.1. We also show that the proposed model does not simply *memorize* the training samples by showing the closest training sequence to the generated ones in Fig. 6.6 and Appendix 6.7.2.

Table 6.2: FID (lower is better) computed on an independent test set with the same number of generated samples as available in the test set.

Model	ColorMNIST	Sprites
VAE	29.79	53.37
VAMP	33.92	59.85
GPVAE	31.93	56.74
Ours ($\mathcal{N}$)	28.62	44.82
Ours (VAMP)	**25.07**	**40.23**
Ours (IAF)	28.14	41.81

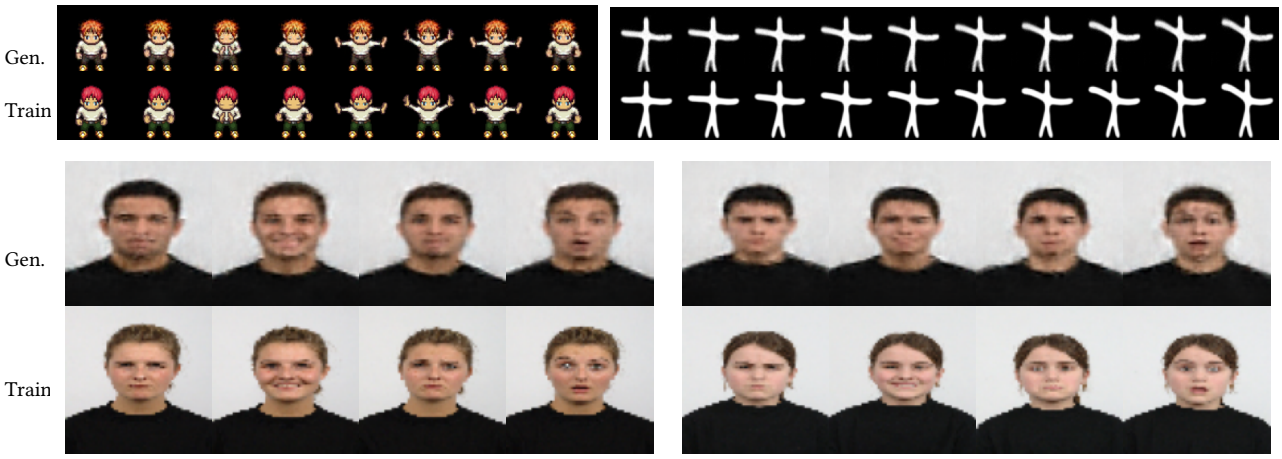


Figure 6.6: Closest train sequences (train) to the generated ones (gen.). See more examples in Appendix 6.7.2.

6.6 Conclusion

In this chapter, we introduced a new generative model for longitudinal data that relies on variational inference and normalizing flows. It proved able to generate relevant fully synthetic sequences and to propose plausible trajectories when conditioned on one or several seen samples in an input sequence. We also discussed and showed that our model can benefit from improvements proposed in the variational inference literature. In particular, we proposed two variants of our model using either a more complex prior or a more flexible variational posterior using flows. These independent enhancements revealed particularly useful for likelihood estimation and unconditional generations. Moreover, the proposed model demonstrated quite a good robustness to missing data and showed to be useful for missing data imputation. Nonetheless, a potential *weakness* of the model is that the flow-based structure makes it discrete. Future work would consist in adapting the model to make it continuous.

6.7 Appendices

6.7.1 Some More Generations

In this section, we show 20 additional generated sequences for the *starmen* dataset in Fig. 6.7, the *colorMNIST* dataset in Fig. 6.8, the *sprites* data in Fig. 6.9, the *faces* dataset in Fig. 6.10 and the *chairs* dataset in Fig. 6.11. This experiment shows the diversity of the generated trajectories as well as their relevance.

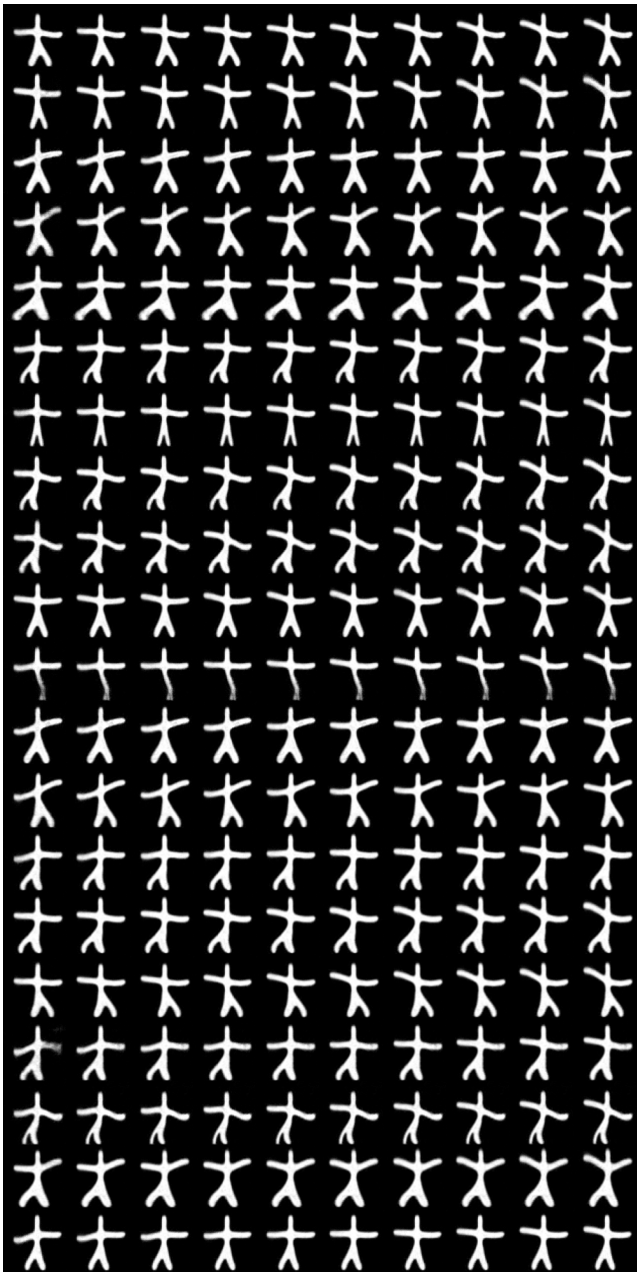


Figure 6.7: 20 sequences generated by our model trained on the *starmen* dataset.

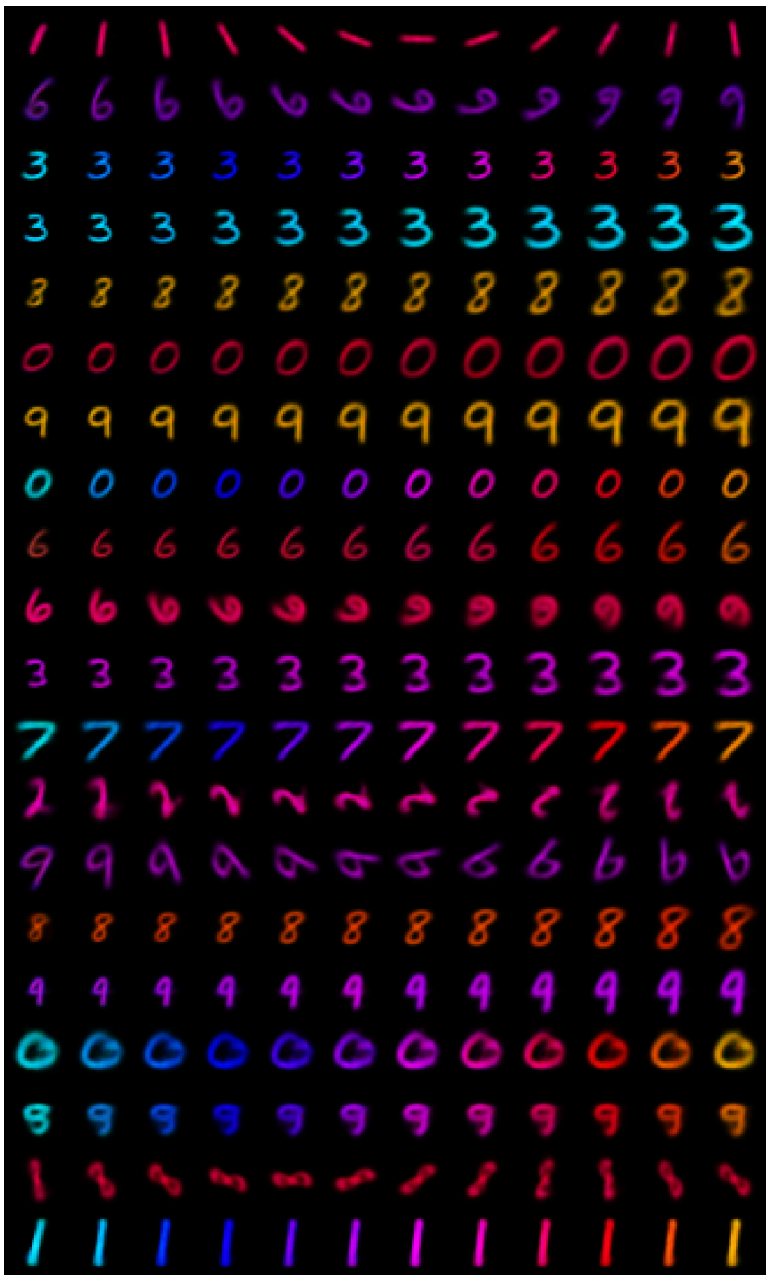


Figure 6.8: 20 sequences generated by our model trained on the *colorMNIST* dataset.

Figure 6.9: 20 sequences generated by our model trained on the *sprites* dataset.

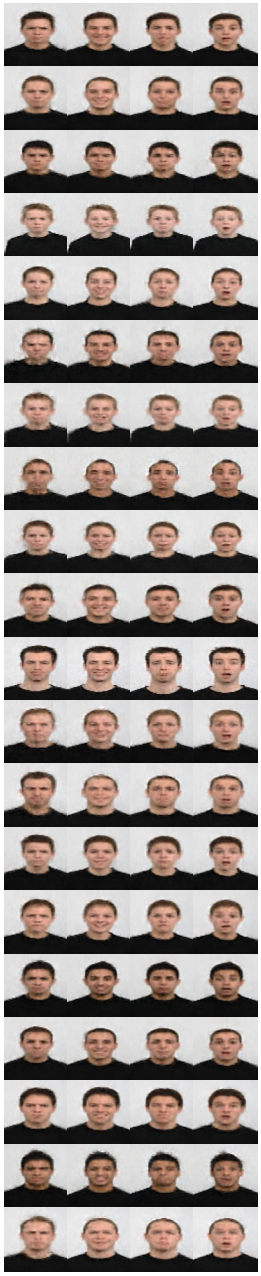


Figure 6.10: 20 sequences generated by our model trained on the *faces* dataset.

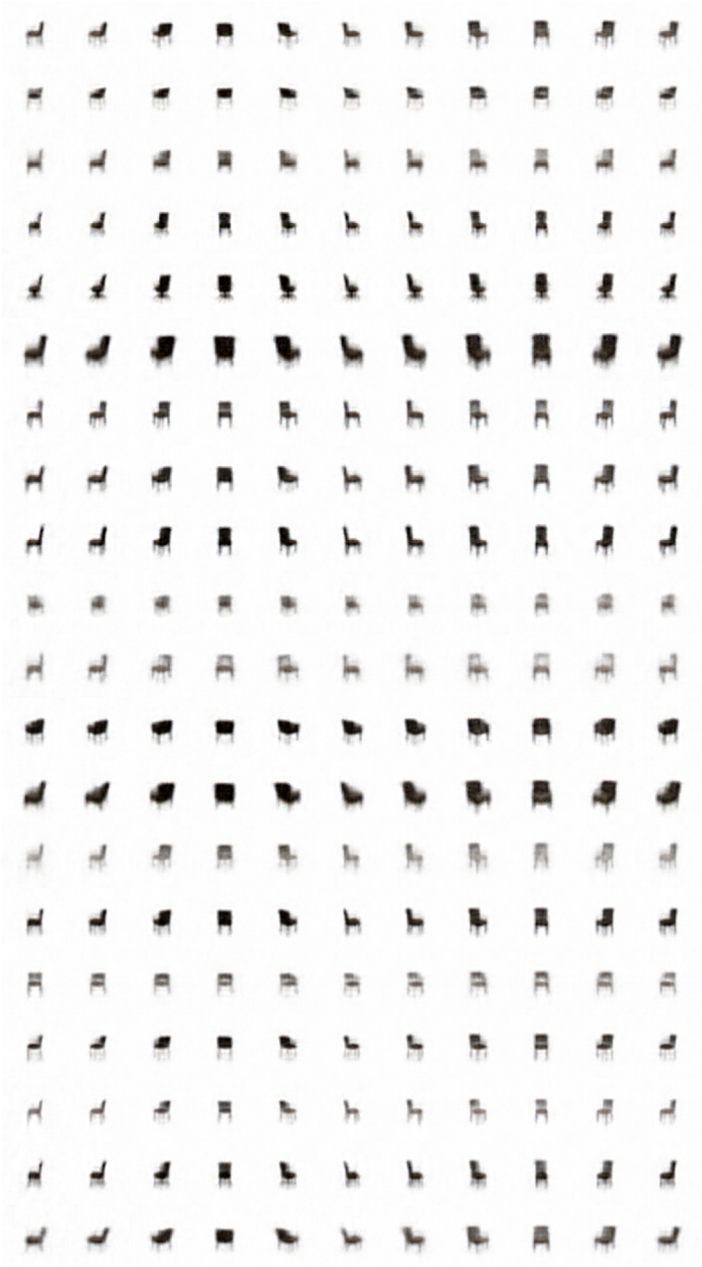


Figure 6.11: 20 sequences generated by our model trained on the *chairs* dataset.

6.7.2 Exploring Overfitting

In this section, we show that the proposed model generates unseen sequences by comparing 4 generated trajectories to the closest one in the train set (using L2 norm). For each dataset, we see that the generated sequence is different from the training data. For instance, for the *starmen*, the individual has a different shape while for the *sprites*, the individual has different pants, hair or top's color.

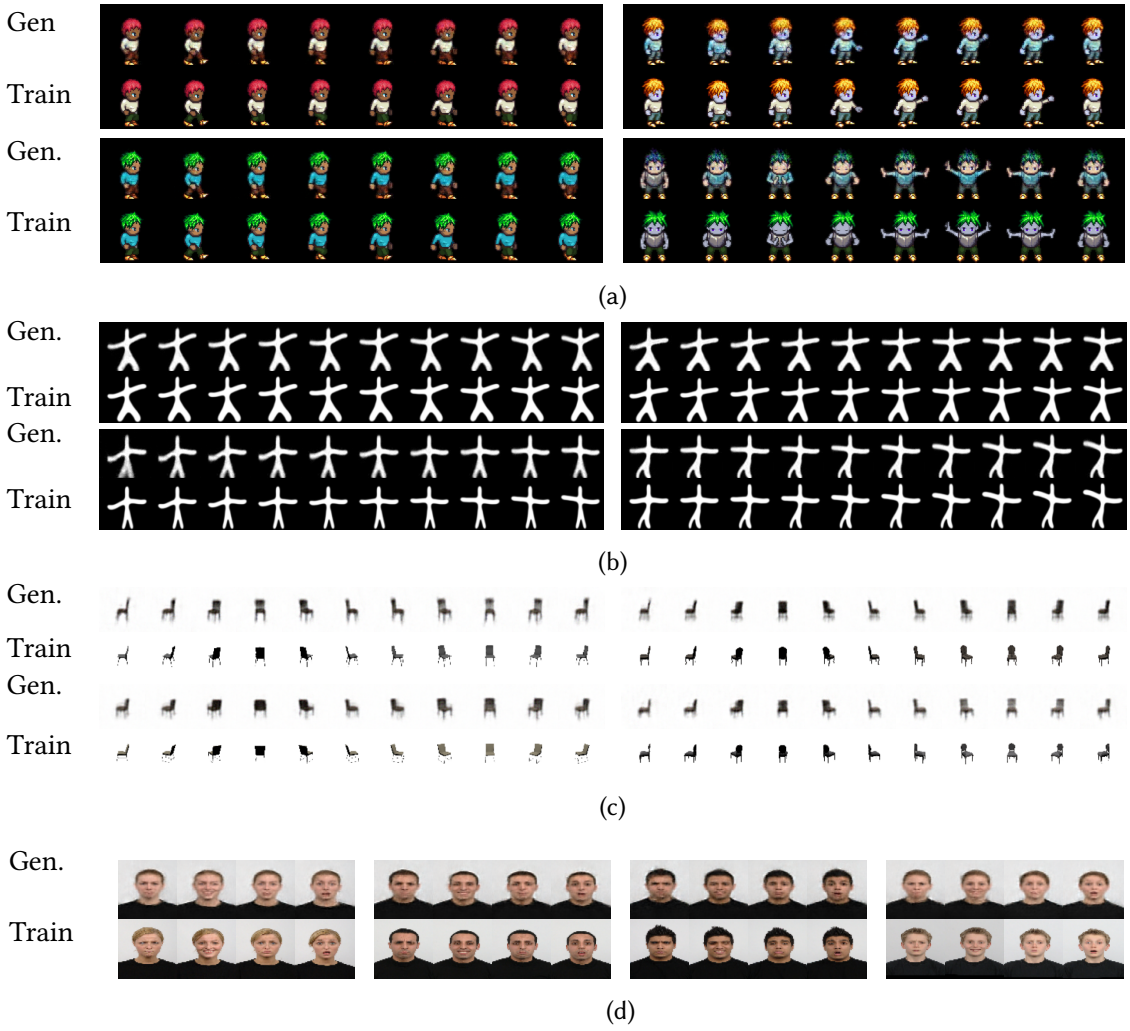


Figure 6.12: Closest train sequences (train) to the generated ones (gen.) using our model trained on (a) the *sprites*, (b) *starmen*, (c) *3d chairs* and (d) *faces* datasets.

6.7.3 Experimental Details

In this section, we detail all the relevant parameters we used for the experiments. The datasets presented in Sec. 6.5.1 are first split into a train set, a validation set and a test set as shown in Table 6.3. We train the models for 200 epochs for *sprites* and *rotMNIST*, 250 for *colorMNIST* and 400 for *starmen* and *chairs* with a latent dimension set to 16 for all datasets but for the *chairs* dataset where it is set to 32. We select the model achieving the lowest validation loss in each case. We use the Adam optimizer (Kingma and Ba, 2014) with a starting learning rate of 10^{-3} together with schedulers reducing the learning rate by a factor 0.5 at epoch 50, 100, 125 and 150 for *starmen*, by a factor 10^{-4} at epoch 50, 75, 100, 125 and 150 for *rotMNIST*, by a factor 10^{-4} at epoch 50, 100, 150 and 200 for *colorMNIST*, by a factor 0.5 at epoch 150, 200, 250, 300 and 350 for *3d chairs* and a factor 0.5 at epoch 50, 100, 125 and 150 for *sprites*. For the *faces* dataset we use a scheduler multiplying the learning rate by 10^{-6} every 2,000 epochs and train the model for 10,000 epochs. We use a batch of size 128 for *rotMNIST*, *colorMNIST* and *faces* and 64 otherwise. For the proposed model, we also use 10 *warm-up* epochs where we train it like a VAE to stabilize the encoder and decoder networks and ease the learning of the flows. This hyper-parameter does not influence much the performances as shown in Appendix 6.7.4. The flows are implemented using (Chadebec et al., 2022c) and are composed of 2 IAF blocks using 3-layer MADE (Germain et al., 2015) with 128 hidden units. For the variants of our model, we use 500 components in the VAMP prior and IAF flows are composed of 3 IAF transformations using 2-layer MADE with 128 hidden units. All models are trained on a single 32-GB V100 GPU and the FID metrics are computed using the implementation of https://github.com/mseitzer/pytorch-fid. Finally, we provide the neural networks we use in Table 6.4. For *faces* we use the same networks as for *sprites* dataset.

Datasets	Train	Validation	Test
Starmen	700	200	100
rotMNIST	9,000	1,000	10,000
colorMNIST	48,000	12,000	10,000
sprites	8,000	1,000	2,664
3d chairs	1,000	200	193

Table 6.3: Number of sequences considered in the Train/Val/Test splits used in the experiments.

Dataset	Starmen	rotMNIST	colorMNIST	3D chairs	Sprites
Input dimension	$(1, 64, 64)$	$(1, 28, 28)$,	$(3, 28, 28)$	$(3, 64, 64)$	$(3, 64, 64)$
Inference Network	Conv2D$(1, 16, 4, 2)$	Linear(1024)	Linear(1024)	Conv2D$(3, 16, 4, 2)$	Conv2D$(3, 16, 4, 2)$
	Conv2D$(16, 32, 4, 2)$	ReLU	ReLU	Conv2D$(16, 32, 4, 2)$	Conv2D$(16, 32, 4, 2)$
	LeakyReLU	Linear(256)	Linear(256)	LeakyReLU	LeakyReLU
	Conv2D$(32, 64, 3, 2)$	ReLU	ReLU	Conv2D$(32, 64, 3, 2)$	Conv2D$(32, 64, 3, 2)$
	LeakyReLU	Linear(2×16)*	Linear(2×16)*	LeakyReLU	LeakyReLU
	Conv2D$(64, 128, 3, 2)$	-	-	Conv2D$(64, 128, 3, 2)$	Conv2D$(64, 128, 3, 2)$
	LeakyReLU	-	-	LeakyReLU	LeakyReLU
	6 ResBlocks	-	-	6 ResBlocks	6 ResBlocks
	Linear $(2048, 2x16)$*	-	-	Linear $(2048, 2x32)$*	Linear $(2048, 2x16)$*
Input dimension	16	16	16	32	16
Generative Network	Linear(2048)	Linear(256)	Linear(256)	Linear(2048)	Linear(2048)
	ConvT$(128, 3, 2)$	ReLU	ReLU	ConvT$(128, 3, 2)$	ConvT$(128, 3, 2)$
	6 ResBlocks	Linear(1024)	Linear(1024)	6 ResBlocks	6 ResBlocks
	ConvT$(64, 5, 2)$	ReLU	ReLU	ConvT$(64, 5, 2)$	ConvT$(64, 5, 2)$
	LeakyReLU	Linear(784)	Linear(2352)	LeakyReLU	LeakyReLU
	ConvT$(32, 5, 2)$	Sigmoid	Sigmoid	ConvT$(32, 5, 2)$	ConvT$(32, 5, 2)$
	LeakyReLU	-	-	LeakyReLU	LeakyReLU
	ConvT$(16, 4, 2)$	-	-	ConvT$(16, 4, 2)$	ConvT$(16, 4, 2)$
	LeakyReLU	-	-	LeakyReLU	LeakyReLU
	ConvT$(1, 4, 2)$	-	-	ConvT$(3, 4, 2)$	ConvT$(3, 4, 2)$

*Layer outputting the mean and covariance of the variational posterior q_ϕ

Table 6.4: Neural networks architectures used in the experiments and keep the same for all the models in the benchmarks. The ResBlocks use 2 convolution layers with kernel of size 3 and 1, 32 channels and stride 1.

6.7.4 Ablation Study

In this section, we present an ablation study of the proposed model where we study the influence of the flow complexity, the latent space dimension, the number of *warm-up* steps (when the model is trained as a VAE) and the prior complexity. We see in Table 6.5 and Table 6.6 that neither the choice in the flows nor the number of *warm-up* steps influence much the resulting likelihoods. Table 6.7 shows that as expected choosing a too small latent space dimension is detrimental to the model performance. Finally, Table 6.8 shows the influence of the prior complexity (number of components used in the VAMP prior). As expected, increasing the complexity of the prior allows achieving better likelihood estimates.

IAF Blocks	MADE layers	Starmen	Sprites
1	3	3774.79 ± 0.19	11302.90 ± 0.02
2	1	3773.31 ± 0.15	11302.25 ± 0.03
2	2	3774.35 ± 0.17	11301.49 ± 0.04
2	3	3773.23 ± 0.18	11301.51 ± 0.04
2	4	3773.45 ± 0.12	11302.23 ± 0.03
2	5	3773.88 ± 0.17	11301.47 ± 0.02
3	3	3773.13 ± 0.10	11302.92 ± 0.03
4	3	3774.12 ± 0.15	11301.05 ± 0.05

Table 6.5: Influence of the flow complexity

Warmup	Starmen	Sprites
2	3773.73 ± 0.10	11301.59 ± 0.02
5	3773.49 ± 0.10	11301.15 ± 0.03
10	3773.23 ± 0.18	11301.51 ± 0.04
20	3774.03 ± 0.10	11302.28 ± 0.03
50	3773.42 ± 0.11	11301.32 ± 0.08
100	3772.44 ± 0.12	11302.40 ± 0.06

Table 6.6: Influence of the warmup steps

Warmup	Starmen	Sprites
2	3817.92 ± 0.20	11346.92 ± 0.09
8	3774.20 ± 0.19	11303.18 ± 0.02
16	3773.23 ± 0.18	11301.51 ± 0.04
32	3773.16 ± 0.16	11302.23 ± 0.02
64	3773.22 ± 0.13	11301.79 ± 0.02

Table 6.7: Influence of the latent dimension

VAMP Components	Starmen	Sprites
10	3773.55 ± 0.07	11302.82 ± 0.04
50	3772.71 ± 0.15	11301.26 ± 0.02
100	3772.89 ± 0.16	11302.07 ± 0.03
200	3772.66 ± 0.22	11302.03 ± 0.03
500	3772.91 ± 0.02	11301.30 ± 0.02

Table 6.8: Influence to the prior complexity

6.7.5 Influence of Eq. (6.8) on Missing Data Imputation

In this appendix, we demonstrate empirically the relevance of the method proposed in Section 6.3.3 to handle missing observations at inference time. We recall that it consists in drawing one (or several) latent variables from the posterior associated to each data point observed in an input sequence (See Alg. 7). The latent variables are then propagated though the flows and sequences are generated in the observation space using the conditional distribution $p_\theta(x|z)$. Using Eq. (6.8), we propose to keep the trajectory achieving the highest likelihood on the observed data. This allows to benefit from all the information observed in the sequence. In Fig. 6.13, we show the Mean Square Error (MSE) on the missing pixels only for the *starmen* dataset (top) and *sprites* dataset (bottom). We keep the same setting as presented in the chapter and remove some data in the input sequences with probability 0.2, 0.4, 0.6 and 0.7 or create sequences with missing observations (randomly removed with probability 0.5) and missing pixels in the observed images (randomly removed with probability 0.2, 0.4 and 0.6). Results obtained with the naive method that consists in using only one randomly chosen data point in the sequence to reconstruct the full sequence as done during training (see Alg. 6) are presented by the slightly transparent bars while results obtained using the method proposed in Section 6.3.3 are shown by the solid bars. In this example, we generate one trajectory per observed data point and keep the one achieving the highest likelihood according to Eq. (6.8). These graphs show the relevance of this method that allows achieving lower MSE in each scenario. Moreover, it allows to decrease significantly the standard deviation (represented by the black bar) leading to a more reliable missing data imputation.

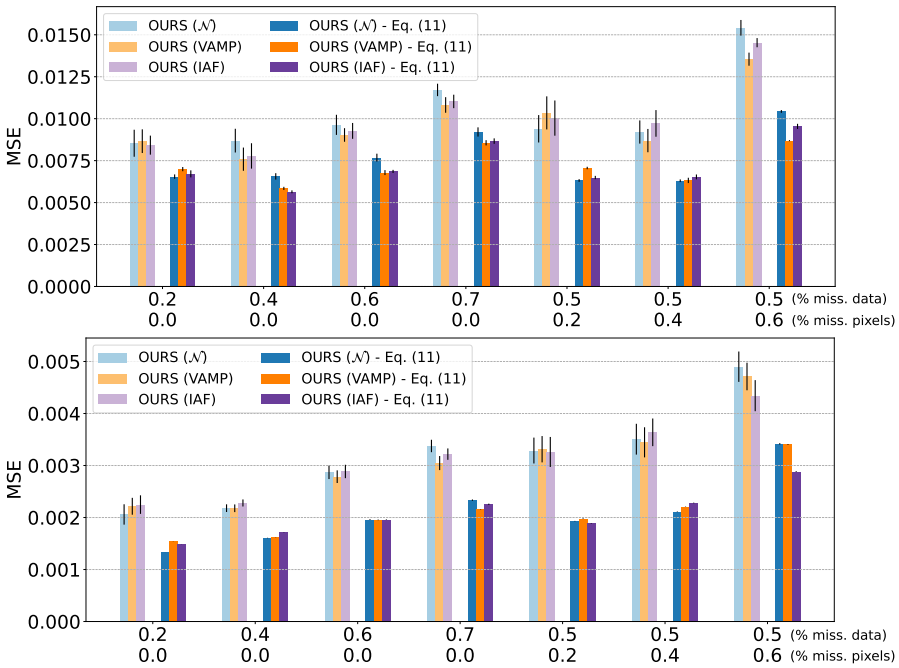


Figure 6.13: Mean Square Error (MSE) on missing pixels only of the test data for different proportions of missing observations (0.2 to 0.7) and missing pixels (0.2 to 0.6) in the input train, validation and test sequences for the *starmen* (top) and *sprites* (bottom) datasets. Slightly transparent bars represent the naive method (consisting in using only one randomly chosen data point in the sequence to reconstruct the full sequence as done during training) while solid bars show the results obtained using the method proposed in Section 6.3.3.

Conclusion and Perspectives

Throughout this PhD thesis, I have proposed several contributions to enhance the VAE framework. This section concludes the thesis and provides interesting research directions that may build upon these works.

In Chapters 1 and 2, I proposed a new model inspired by the VAE framework but including some geometric considerations in the latent space seen as a Riemannian manifold. I also introduced new ways to sample from the latent space of the newly proposed *geometry-aware* VAE considering the learned geometry. These sampling schemes proved useful to generate new data from the trained VAE and were particularly well suited in the context of low sample sizes. Nonetheless, this model and the proposed sampling methods were only applied to perform data generation and data augmentation on images while it can reveal useful for other tasks and structured data as well. It would indeed be interesting to extend the empirical studies performed in the context of data augmentation to other modalities different from the image data. In addition, one model was trained per label in each of the considered cases. Hence, an interesting extension of this work would consist in making it applicable to a conditional setting. This would potentially allow the training of the model to benefit from more data while reducing the number of models to use. Moreover, since the geometry of the latent space revealed able to achieve quite smooth interpolations, it would be interesting to derive clustering algorithms using the learned metric such as a Riemannian k-means and see how it performs when compared to the classic Euclidean VAE. Moreover, these Riemannian interpolations may also be used to perform data augmentation similarly to mix-up (Zhang et al., 2018b).

In Chapter 3, I proposed a novel interpretation of the vanilla VAE framework under a fully geometric viewpoint. This study showed that even taken in its simplest shape, the Variational Autoencoder allows for quite an informative and structured latent space. A deeper study of this learned latent space could also show how and if the proposed modeling could reveal useful and be adapted to the context of representation learning. In particular, one may wonder if it can help to learn disentangled representations of the data. The role that the imposed structure on the covariance matrices in the posterior distributions has on representation learning is besides something that has raised some interest in particular for disentanglement (Rolinek et al., 2019). I also believe that these covariance matrices indeed play a crucial role in modeling the latent space, and a deeper understanding of their impact on the latent space structure is an exciting research direction. It would also be interesting to plug the proposed metric (*i.e.* involving the inverse of the covariances in the approximate posterior distribution) in the model proposed in Chapter 1 since this would reduce the number of learned parameters (since the metric is no longer learned with an independent network). Moreover, this method was only applied to image data, while it would be interesting to see how the latent structures when trained with structured data such as molecular configurations.

Chapter 4 was mainly dedicated to the development of an *open-source* Python library providing both a *unified implementation* and a dedicated framework allowing *reproducible* and *reliable* use of generative autoencoder models. I aim to maintain this library in the future and hope it will remain active and useful for a long time. With another PhD student, I am currently considering extending this framework to the multi-modal VAE setting. I still deeply think that such tools are necessary for our research community to allow researchers to reproduce existing results, compare their new models to the ones already proposed in the literature or use them in any way in their research.

Beyond researchers, these libraries are also important since they make recent research advances accessible and applicable in other use cases.

Finally, Chapter 6 mainly focused on developing a new latent variable generative model capable of handling longitudinal data. I decided to model the time dependency between the observations within an input sequence using normalizing flows which constrained the model to remain discrete, meaning that we cannot interpolate continuously between two observations at two consecutive times. Hence, a possible extension of this framework would consist in trying to make it continuous. An aspect that I thought to be interesting in this modeling is the fact that this is the latent code distribution that evolves through time. Hence, an exciting research perspective may consist in using these flows to perform trajectory clustering (Debavelaere et al., 2020). Moreover, using the conditional setting explained in Chapter 6 may be useful in the medical context. Indeed, one may think of applications such as treatment response prediction where one observation before the treatment could be given to the model that should be able to generate *plausible* future trajectories (*i.e.* responses to a given treatment).

Bibliography

K. Aderghal, M. Boissenin, J. Benois-Pineau, G. Catheline, and K. Afdel. Classification of sMRI for AD diagnosis with convolutional neuronal networks: A pilot 2-D+ϵ study on ADNI. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10132 LNCS, pages 690–701, Jan. 2017. doi: 10.1007/978-3-319-51811-4_56. ZSCC: NoCitationData[s0].

K. Aderghal, A. Khvostikov, A. Krylov, J. Benois-Pineau, K. Afdel, and G. Catheline. Classification of Alzheimer Disease on Imaging Modalities with Deep CNNs Using Cross-Modal Transfer Learning. In *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*, pages 345–350, 2018. doi: 10.1109/CBMS.2018.00067. ISSN: 2372-9198.

M. Aghili, S. Tabarestani, M. Adjouadi, and E. Adeli. Predictive modeling of longitudinal data for alzheimer's disease diagnosis using rnns. In *PRedictive Intelligence in MEdicine*, pages 112–119, Cham, 2018. Springer International Publishing.

A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

B. Amos. bamos/densenet.pytorch, 2020. URL https://github.com/bamos/densenet.pytorch. original-date: 2017-02-09T15:33:23Z.

J. Aneja, A. Schwing, J. Kautz, and A. Vahdat. NCP-VAE: Variational autoencoders with noise contrastive priors. *arXiv:2010.02917 [cs, stat]*, 2020.

A. Antoniou, A. Storkey, and H. Edwards. Data augmentation generative adversarial networks. *arXiv:1711.04340 [cs, stat]*, 2018-03-21.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*, 2017-12-06.

G. Arvanitidis, L. K. Hansen, and S. Hauberg. A locally adaptive normal distribution. *Advances in Neural Information Processing Systems*, pages 4258–4266, 2016.

G. Arvanitidis, L. K. Hansen, and S. Hauberg. Latent space oddity: On the curvature of deep generative models. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

G. Arvanitidis, S. Hauberg, and B. Schölkopf. Geometrically enriched latent spaces. *arXiv:2008.00565 [cs, stat]*, 2020-08-02.

M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3762–3769, 2014.

B. B. Avants, N. J. Tustison, M. Stauffer, G. Song, B. Wu, and J. C. Gee. The Insight ToolKit image registration framework. *Frontiers in Neuroinformatics*, 8, 2014. ISSN 1662-5196. doi: 10.3389/fninf.2014.00044.

K. Backstrom, M. Nazari, I.-H. Gu, and A. Jakola. An efficient 3D deep convolutional network for Alzheimer's disease diagnosis using MR images. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, volume 2018-April, pages 149–153, 2018. doi: 10.1109/ISBI.2018.8363543.

S. Barratt and R. Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

S. Barua, M. M. Islam, X. Yao, and K. Murase. MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2012. ISSN 1041-4347.

M. Bauer and A. Mnih. Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 66–75. PMLR, 2019a.

M. Bauer and A. Mnih. Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 66–75. PMLR, 2019b. ISBN 2640-3498.

C. Baur, S. Albarqouni, and N. Navab. Generating highly realistic images of skin lesions with GANs. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 260–267. Springer, 2018.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012. ISSN ISSN 1533-7928.

J. L. Bernal-Rusiel, D. N. Greve, M. Reuter, B. Fischl, M. R. Sabuncu, A. D. N. Initiative, et al. Statistical analysis of longitudinal neuroimage data with linear mixed effects models. *Neuroimage*, 66:249–260, 2013.

D. Berthelot*, C. Raffel*, A. Roy, and I. Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1fQSiCcYm.

L. Bi, J. Kim, A. Kumar, D. Feng, and M. Fulham. Synthesis of Positron Emission Tomography (PET) Images via Multi-channel Generative Adversarial Networks (GANs). In *Molecular Imaging, Reconstruction and Analysis of Moving Body Organs, and Stroke Imaging and Treatment*, LNCS, pages 43–51. Springer, 2017. doi: 10.1007/978-3-319-67564-0_5.

L. Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

E. Bisong. *Google Colaboratory*, pages 59–64. Apress, Berkeley, CA, 2019. ISBN 978-1-4842-4470-8. doi: 10.1007/978-1-4842-4470-8_7. URL https://doi.org/10.1007/978-1-4842-4470-8_7.

M. Blaauw and J. Bonada. Modeling and transforming speech using variational autoencoders. *Morgan N, editor. Interspeech 2016; 2016 Sep 8-12; San Francisco, CA.[place unknown]: ISCA; 2016. p. 1770-4.*, 2016. Publisher: International Speech Communication Association (ISCA).

M. D. Blackledge, D. J. Collins, N. Tunariu, M. R. Orton, A. R. Padhani, M. O. Leach, and D.-M. Koh. Assessment of treatment response by total tumor volume and global apparent diffusion coefficient using diffusion-weighted mri in patients with metastatic bone disease: A feasibility study. *PLOS ONE*, 9(4):e91779, 2014.

R. Blagus and L. Lusa. SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14 (1):106, 2013. ISSN 1471-2105. doi: 10.1186/1471-2105-14-106.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

A. Bône, O. Colliot, and S. Durrleman. Learning distributions of shape trajectories from longitudinal datasets: a hierarchical model on a manifold of diffeomorphisms. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9271–9280, 2018.

A. Borji. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019. ISSN 1077-3142.

D. Bouchacourt, R. Tomioka, and S. Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv:1509.00519 [cs, stat]*, 2016.

C. P. e. a. Burgess. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

K. S. Button, J. P. Ioannidis, C. Mokrysz, B. A. Nosek, J. Flint, E. S. Robinson, and M. R. Munafò. Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, 14(5):365–376, 2013.

F. Calimeri, A. Marzullo, C. Stamile, and G. Terracina. Biomedical data augmentation using generative adversarial neural networks. In *International conference on artificial neural networks*, pages 626–634. Springer, 2017.

W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, 31, 2018.

F. P. Casale, A. Dalca, L. Saglietti, J. Listgarten, and N. Fusi. Gaussian process prior variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

A. L. Caterini, A. Doucet, and D. Sejdinovic. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, pages 8167–8177, 2018.

C. Chadebec and S. Allassonnière. Data augmentation with variational autoencoders and manifold sampling. In *Deep Generative Models, and Data Augmentation, Labelling, and Imperfections: First Workshop, DGM4MICCAI 2021, and First Workshop, DALI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings 1*, pages 184–192. Springer, 2021.

C. Chadebec and S. Allassonnière. A geometric perspective on variational autoencoders. *Advances in Neural Information Processing Systems*, 2022.

C. Chadebec, C. Mantoux, and S. Allassonnière. Geometry-aware hamiltonian variational auto-encoder. *arXiv:2010.11518*, 2020.

C. Chadebec, E. M. Huijben, J. P. Pluim, S. Allassonnière, and M. A. van Eijnatten. An image feature mapping model for continuous longitudinal data completion and generation of synthetic patient trajectories. In *Deep Generative Models: Second MICCAI Workshop, DGM4MICCAI 2022, Held in Conjunction with MICCAI 2022, Singapore, September 22, 2022, Proceedings*, pages 55–64. Springer, 2022a.

C. Chadebec, E. Thibeau-Sutre, N. Burgos, and S. Allassonnière. Data augmentation in high dimensional low sample size setting using a geometry-based variational autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022b.

C. Chadebec, L. J. Vincent, and S. Allassonnière. Pythae: Unifying generative autoencoders in python–a benchmarking use case. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2022c.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. ISSN 1076-9757.

N. Chen, A. Klushyn, R. Kurle, X. Jiang, J. Bayer, and P. Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550. PMLR, 2018a.

R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018b.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016a.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016b.

D. Cheng and M. Liu. CNNs based multi-modality classification for AD diagnosis. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–5, 2017. doi: 10.1109/CISP-BMEI.2017.8302281.

R. Child. Very deep vaes generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2021.

M. J. Chong and D. Forsyth. Effectively unbiased fid and inception score and where to find them. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6070–6079, 2020.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

M. Connor, G. Canal, and C. Rozell. Variational autoencoder with learned latent structure. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2367. PMLR, 2021.

R. Couronné, P. Vernhet, and S. Durrleman. Longitudinal self-supervision to disentangle inter-patient variability from disease progression. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 231–241, Cham, 2021. Springer International Publishing.

C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.

B. Dai and D. Wipf. Diagnosing and Enhancing VAE Models. In *International Conference on Learning Representations*, 2018.

T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak. Hyperspherical variational auto-encoders. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 856–865. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

V. Debavelaere, S. Durrleman, S. Allassonnière, and A. D. N. Initiative. Learning the clustering of longitudinal shape data sets into a mixture of independent or branching trajectories. *International Journal of Computer Vision*, 128:2794–2809, 2020.

P. Diggle, P. J. Diggle, P. Heagerty, K.-Y. Liang, S. Zeger, et al. *Analysis of longitudinal data*. Oxford university press, 2002.

N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv:1611.02648 [cs, stat]*, 2017.

L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

J. Domke and D. R. Sheldon. Importance weighting and variational inference. *Advances in neural information processing systems*, 31, 2018.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195 (2):216–222, 1987.

C. Eastwood and C. K. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

M. Ehsan Abbasnejad, A. Dick, and A. van den Hengel. Infinite variational autoencoder for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5888–5897, 2017.

K. A. Ellis, A. I. Bush, D. Darby, D. De Fazio, J. Foster, P. Hudson, N. T. Lautenschlager, N. Lenzo, R. N. Martins, P. Maruff, C. Masters, A. Milner, K. Pike, C. Rowe, G. Savage, C. Szoeke, K. Taddei, V. Villemagne, M. Woodward, D. Ames, and AIBL Research Group. The Australian Imaging, Biomarkers and Lifestyle (AIBL) study of aging: methodology and baseline characteristics of 1112 individuals recruited for a longitudinal study of Alzheimer's disease. *International Psychogeriatrics*, 21(4):672–687, 2009. ISSN 1041-6102. doi: 10.1017/S1041610209009405.

L. Falorsi, P. de Haan, T. R. Davidson, N. De Cao, M. Weiler, P. Forré, and T. S. Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv:1807.04689 [cs, stat]*, 2018.

A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018. ISSN 1076-9757.

V. Fonov, A. Evans, R. McKinstry, C. Almli, and D. Collins. Unbiased nonlinear average age-appropriate brain templates from birth to adulthood. *NeuroImage*, 47:S102, 2009. ISSN 1053-8119. doi: 10.1016/S1053-8119(09)70884-5.

V. Fonov, A. C. Evans, K. Botteron, C. R. Almli, R. C. McKinstry, and D. L. Collins. Unbiased average age-appropriate atlases for pediatric studies. *NeuroImage*, 54(1):313–327, 2011. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2010.07.033.

V. S. Fonov, M. Dadar, T. P.-A. R. Group, and D. L. Collins. Deep learning of quality control for stereotaxic registration of human brain MRI. *bioRxiv*, page 303487, 2018. doi: 10.1101/303487.

H. M. Fonteijn, M. Modat, M. J. Clarkson, J. Barnes, M. Lehmann, N. Z. Hobbs, R. I. Scahill, S. J. Tabrizi, S. Ourselin, N. C. Fox, et al. An event-based model for disease progression and its application in familial alzheimer's disease and huntington's disease. *NeuroImage*, 60(3):1880–1889, 2012.

V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch. Som-vae: Interpretable discrete representation learning on time series. In *International Conference on Learning Representations*, 2019.

V. Fortuin, D. Baranchuk, G. Rätsch, and S. Mandt. Gp-vae: Deep probabilistic time series imputation. In *International conference on artificial intelligence and statistics*, pages 1651–1661. PMLR, 2020.

M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. *Advances in neural information processing systems*, 29, 2016.

M. F. Frenzel, B. Teleaga, and A. Ushio. Latent space cartography: Generalised metric-inspired measures and measure-based transformations for generative models. *arXiv preprint arXiv:1902.02113*, 2019.

M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018. ISSN 0925-2312.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

P. Ghosh, M. S. Sajjadi, A. Vergari, M. Black, and B. Schölkopf. From variational to deterministic autoencoders. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

R. J. Gillies, P. E. Kinahan, and H. Hricak. Radiomics: images are more than pictures, they are data. *Radiology*, 278(2):563–577, 2016.

M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

M. Girolami, B. Calderhead, and S. A. Chin. Riemannian manifold hamiltonian monte carlo. *arXiv preprint arXiv:0907.1100*, 2009.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. Issue: 2.

K. J. Gorgolewski, T. Auer, V. D. Calhoun, R. C. Craddock, S. Das, E. P. Duff, G. Flandin, S. S. Ghosh, T. Glatard, Y. O. Halchenko, D. A. Handwerker, M. Hanke, D. Keator, X. Li, Z. Michael, C. Maumet, B. N. Nichols, T. E. Nichols, J. Pellman, J.-B. Poline, A. Rokem, G. Schaefer, V. Sochat, W. Triplett, J. A. Turner, G. Varoquaux, and R. A. Poldrack. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific Data*, 3(1):160044, 2016. ISSN 2052-4463. doi: 10.1038/sdata.2016.44.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

R.-R. Griffiths and J. M. Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586, 2020.

I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville. Pixelvae: A latent variable model for natural images. In *International Conference on Learning Representations*.

S. Gur, S. Benaim, and L. Wolf. Hierarchical patch vae-gan: Generating diverse videos from a single sample. *Advances in Neural Information Processing Systems*, 33:16761–16772, 2020.

Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008. ISBN 978-1-4244-1820-6. doi: 10.1109/IJCNN.2008.4633969.

E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In D.-S. Huang, X.-P. Zhang, and G.-B. Huang, editors, *Advances in Intelligent Computing*, volume 3644, pages 878–887. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28226-6 978-3-540-31902-3. doi: 10.1007/11538059_91. Series Title: LNCS.

S. Hauberg, O. Freifeld, and M. Black. A Geometric take on Metric Learning. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/ec5aa0b7846082a2415f0902f0da88f2-Paper.pdf.

K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Santiago, Chile, 2015. IEEE. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.123.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5): 6, 2017.

M. D. Hoffman. Learning deep latent gaussian models with markov chain monte carlo. In *International conference on machine learning*, pages 1510–1519. PMLR, 2017.

M. D. Hoffman and M. J. Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2, 2016.

W.-N. Hsu, Y. Zhang, and J. Glass. Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 16–23. IEEE, 2017.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269. IEEE, 2017. ISBN 978-1-5386-0457-1.

H. Huang, R. He, Z. Sun, T. Tan, et al. Introvae: Introspective variational autoencoders for photographic image synthesis. *Advances in neural information processing systems*, 31, 2018.

Z. Hussain, F. Gimenez, D. Yi, and D. Rubin. Differential data augmentation techniques for medical imaging classification tasks. In *AMIA annual symposium proceedings*, volume 2017, page 979. American Medical Informatics Association, 2017.

A. Hyvärinen, J. Hurri, and J. Väyrynen. A unifying framework for natural image statistics: spatiotemporal activity bubbles. *Neurocomputing*, 58:801–806, 2004.

F. Isensee, M. Schell, I. Pflueger, G. Brugnara, D. Bonekamp, U. Neuberger, A. Wick, H.-P. Schlemmer, S. Heiland, W. Wick, et al. Automated brain extraction of multisequence mri using artificial neural networks. *Human brain mapping*, 40(17):4952–4964, 2019.

J. Islam and Y. Zhang. GAN-based synthetic brain PET image generation. *Brain Informatics*, 7(1), 2020. doi: 10.1186/s40708-020-00104-2.

B. M. Jedynak, A. Lang, B. Liu, E. Katz, Y. Zhang, B. T. Wyman, D. Raunig, C. P. Jedynak, B. Caffo, J. L. Prince, et al. A computational neurodegenerative disease progression score: method and results with the alzheimer's disease neuroimaging initiative cohort. *Neuroimage*, 63(3):1478–1486, 2012.

M. Jenkinson and S. Smith. A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, 5(2):143–156, 2001.

M. Jenkinson, P. Bannister, M. Brady, and S. Smith. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *Neuroimage*, 17(2):825–841, 2002.

Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1965–1972, 2017.

X.-B. Jin, W.-T. Gong, J.-L. Kong, Y.-T. Bai, and T.-L. Su. Pfvae: a planar flow-based variational auto-encoder prediction model for time series data. *Mathematics*, 10(4):610, 2022.

M. J. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. *Advances in neural information processing systems*, 29, 2016.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

S. Jung and M. Keuper. Internalized biases in fréchet inception distance. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

D. Kalatzis, D. Eklund, G. Arvanitidis, and S. Hauberg. Variational autoencoders with riemannian brownian motion priors. In *International Conference on Machine Learning*, pages 5053–5066. PMLR, 2020.

M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2017.

T. A. Keller and M. Welling. Topographic vaes learn equivariant capsules. *Advances in Neural Information Processing Systems*, 34:28585–28597, 2021.

H. Kim and A. Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

S. T. Kim, U. Küçükaslan, and N. Navab. Longitudinal brain mr image modeling using personalized memory for alzheimer's disease. *IEEE Access*, 9:143212–143221, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv:1312.6114 [cs, stat]*, 2014.

D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

D. P. Kingma, M. Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

A. Klushyn, N. Chen, R. Kurle, and B. Cseke. Learning Hierarchical Priors in VAEs. *Advances in neural information processing systems*, page 10, 2019.

D. Korkinof, T. Rijken, M. O'Neill, J. Yearsley, H. Harvey, and B. Glocker. High-resolution mammogram synthesis using progressive generative adversarial networks. *arXiv preprint arXiv:1807.03401*, 2018.

S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

I. Koval, J.-B. Schiratti, A. Routier, M. Bacci, O. Colliot, S. Allassonnière, S. Durrleman, A. D. N. Initiative, et al. Statistical learning of spatiotemporal patterns from longitudinal manifold-valued networks. In *International conference on medical image computing and computer-assisted intervention*, pages 451–459. Springer, 2017.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

A. Kumar and B. Poole. On implicit regularization in β-vaes. In *International Conference on Machine Learning*, pages 5480–5490. PMLR, 2020.

N. M. Laird and J. H. Ware. Random-effects models for longitudinal data. *Biometrics*, pages 963–974, 1982.

B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

P. Lambin, R. T. Leijenaar, T. M. Deist, J. Peerlings, E. E. De Jong, J. Van Timmeren, S. Sanduleanu, R. T. Larue, A. J. Even, A. Jochems, et al. Radiomics: the bridge between medical imaging and personalized medicine. *Nature reviews Clinical oncology*, 14(12):749–762, 2017.

O. Langner, R. Dotsch, G. Bijlstra, D. H. Wigboldus, S. T. Hawk, and A. Van Knippenberg. Presentation and validation of the radboud faces database. *Cognition and emotion*, 24(8):1377–1388, 2010.

A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

G. Lebanon. Metric learning for text documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):497–508, 2006. ISSN 0162-8828.

Y. LeCun. The MNIST database of handwritten digits. 1998.

B. Leimkuhler and S. Reich. *Simulating hamiltonian dynamics*, volume 14. Cambridge university press, 2004.

Y. Li and S. Mandt. Disentangled sequential autoencoder. In *International Conference on Machine Learning*, 2018.

D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*, pages 689–698, 2018.

J. Lim, S. Ryu, J. W. Kim, and W. Y. Kim. Molecular generative model based on conditional variational autoencoder for de novo molecular design. *Journal of cheminformatics*, 10(1):1–9, 2018a.

S. K. Lim, Y. Loo, N.-T. Tran, N.-M. Cheung, G. Roig, and Y. Elovici. Doping: Generative data augmentation for unsupervised anomaly detection with gan. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1122–1127. IEEE, 2018b.

S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts. Anomaly detection for time series using vae-lstm hybrid model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4322–4326. Ieee, 2020.

J. S. Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

M. Liu, J. Zhang, C. Lian, and D. Shen. Weakly Supervised Deep Learning for Brain Disease Prognosis Using MRI and Incomplete Clinical Scores. *IEEE Transactions on Cybernetics*, 50(7):3381–3392, 2020. doi: 10.1109/TCYB.2019.2904186.

Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018a.

X. Liu, Y. Zou, L. Kong, Z. Diao, J. Yan, J. Wang, S. Li, P. Jia, and J. You. Data augmentation via latent space interpolation for image classification. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 728–733. IEEE, 2018b.

X. Liu, L. Song, S. Liu, and Y. Zhang. A review of deep-learning-based medical image segmentation methods. *Sustainability*, 13(3):1224, 2021a.

Y. Liu, Y. Zhou, X. Liu, F. Dong, C. Wang, and Z. Wang. Wasserstein gan-based small-sample augmentation for new-generation artificial intelligence: a case study of cancer-staging data in biology. *Engineering*, 5(1):156–163, 2019. ISSN 2095-8099.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Z.-S. Liu, W.-C. Siu, and L.-W. Wang. Variational autoencoder for reference based image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 516–525, 2021b.

M. Louis. *Computational and statistical methods for trajectory analysis in a Riemannian geometry setting*. PhD Thesis, Sorbonnes universités, 2019.

M. Louis, R. Couronné, I. Koval, B. Charlier, and S. Durrleman. Riemannian geometry learning for disease progression modelling. In *International Conference on Information Processing in Medical Imaging*, pages 542–553. Springer, 2019.

M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? a large-scale study. In *Advances in Neural Information Processing Systems*, page 10, 2018.

Y. Luo, X. Cai, Y. Zhang, J. Xu, et al. Multivariate time series imputation with generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. Biva: A very deep hierarchy of latent variables for generative modeling. *Advances in neural information processing systems*, 32, 2019.

A. Madani, M. Moradi, A. Karargyris, and T. Syeda-Mahmood. Chest x-ray generation and data augmentation for cardiovascular abnormality classification. In *Medical Imaging 2018: Image Processing*, volume 10574, page 105741M. International Society for Optics and Photonics, 2018.

C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. Filtering variational objectives. *Advances in Neural Information Processing Systems*, 30, 2017.

A. Makhzani and B. J. Frey. Pixelgan autoencoders. *Advances in Neural Information Processing Systems*, 30, 2017.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

A. Mallasto and A. Feragen. Wrapped gaussian process regression on riemannian manifolds. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5580–5588. IEEE, 2018. ISBN 978-1-5386-6420-9.

D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner. Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. *Journal of Cognitive Neuroscience*, 19(9):1498–1507, 2007.

G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi. BAGAN: Data Augmentation with Balancing GAN. *arXiv:1803.09655*, 2018.

E. Mathieu, C. Le Lan, C. J. Maddison, R. Tomioka, and Y. W. Teh. Continuous hierarchical representations with poincaré variational auto-encoders. In *Advances in neural information processing systems*, pages 12565–12576, 2019a.

E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh. Disentangling disentanglement in variational autoencoders. In *International Conference on Machine Learning*, pages 4402–4412. PMLR, 2019b.

Y. Miao, L. Yu, and P. Blunsom. Neural variational inference for text processing. In *International conference on machine learning*, pages 1727–1736. PMLR, 2016.

N. Miolane and S. Holmes. Learning weighted submanifolds with variational autoencoders and riemannian variational autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14503–14511, 2020.

S. Morozov, A. Voynov, and A. Babenko. On self-supervised image representations for gan evaluation. In *International Conference on Learning Representations*, 2020.

A. Myronenko. 3D MRI brain tumor segmentation using autoencoder regularization. In *International MICCAI Brainlesion Workshop*, pages 311–320. Springer, 2018.

E. Nalisnick, L. Hertel, and P. Smyth. Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, volume 2, page 131, 2016.

R. M. Neal. Hamiltonian importance sampling. In *talk presented at the Banff International Research Station (BIRS) workshop on Mathematical Issues in Molecular Dynamics*, 2005.

R. M. Neal and others. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

S. M. Nestor, R. Rupsingh, M. Borrie, M. Smith, V. Accomazzi, J. L. Wells, J. Fogarty, R. Bartha, and the Alzheimer's Disease Neuroimaging Initiative. Ventricular enlargement as a possible measure of Alzheimer's disease progression validated using the Alzheimer's disease neuroimaging initiative database. *Brain*, 131(9):2443–2454, 2008.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

H. M. Nguyen, E. W. Cooper, and K. Kamei. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1):4–21, 2011. ISSN 1755-3210.

H. Nishizaki. Data augmentation and feature extraction using variational autoencoder for acoustic modeling. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1222–1227. IEEE, 2017. ISBN 978-1-5386-1542-3. doi: 10.1109/APSIPA.2017.8282225.

Z. Niu, K. Yu, and X. Wu. Lstm-based vae-gan for time-series anomaly detection. *Sensors*, 20(13): 3738, 2020.

K. Oh, Y.-C. Chung, K. W. Kim, W.-S. Kim, and I.-S. Oh. Classification and Visualization of Alzheimer's Disease using Volumetric Convolutional Neural Network and Transfer Learning. *Scientific Reports*, 9(1):18150, 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-54548-6.

I. Ovinnikov. Poincaré wasserstein autoencoder. *arXiv:1901.01427 [cs, stat]*, 2020.

B. Paige, J.-W. van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, P. Torr, et al. Learning disentangled representations with semi-supervised deep generative models. *Advances in neural information processing systems*, 30, 2017.

N. Painchaud, Y. Skandarani, T. Judge, O. Bernard, A. Lalande, and P.-M. Jodoin. Cardiac MRI segmentation with strong anatomical guarantees. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 632–640. Springer, 2019.

B. Pang, T. Han, E. Nijkamp, S.-C. Zhu, and Y. N. Wu. Learning latent space energy-based prior model. *Advances in Neural Information Processing Systems*, 33, 2020.

G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269, 1989.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

X. Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, 2006. ISSN 0924-9907, 1573-7683. doi: 10.1007/s10851-006-6228-4.

Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29, 2016.

A. Rakowski and C. Lippert. Disentanglement and local directions of variance. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–34. Springer, 2021.

S. Ramchandran, G. Tikhonov, K. Kujanpää, M. Koskinen, and H. Lähdesmäki. Longitudinal variational autoencoder. In *International Conference on Artificial Intelligence and Statistics*, pages 3898–3906. PMLR, 2021.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR, 2016.

A. Razavi, A. v. d. Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in Neural Information Processing Systems*, 2020.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

D. Rezende, I. Danihelka, K. Gregor, D. Wierstra, et al. One-shot generalization in deep generative models. In *International conference on machine learning*, pages 1521–1529. PMLR, 2016.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.

M. Rolinek, D. Zietlow, and G. Martius. Variational autoencoders pursue pca directions (by accident). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

A. Routier, N. Burgos, M. Díaz, M. Bacci, S. Bottani, O. El-Rifai, S. Fontanella, P. Gori, J. Guillon, A. Guyot, et al. Clinica: An open-source software platform for reproducible clinical neuroscience studies. *Frontiers in Neuroinformatics*, 15:689675, 2021.

F. Ruiz and M. Titsias. A contrastive divergence for combining variational inference and mcmc. In *International Conference on Machine Learning*, pages 5537–5545. PMLR, 2019.

N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 600–608, 2019.

M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, pages 5228–5237, 2019.

H. Salehinejad, S. Valaee, T. Dowdell, E. Colak, and J. Barfett. Generalization of deep neural networks for chest pathology classification in x-rays using generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 990–994. IEEE, 2018.

T. Salimans, D. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. *Scientific reports*, 9(1):16884, 2019. ISSN 2045-2322.

B. Sauty and S. Durrleman. Progression models for imaging data with longitudinal variational auto encoders. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 3–13. Springer, 2022.

J.-B. Schiratti, S. Allassonniere, O. Colliot, and S. Durrleman. Learning spatiotemporal trajectories from manifold-valued longitudinal data. *Advances in neural information processing systems*, 28, 2015.

M. Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02): 69–106, 2004.

R. Selvan, E. B. Dam, N. S. Detlefsen, S. Rischel, K. Sheng, M. Nielsen, and A. Pai. Lung segmentation from chest x-rays using variational data imputation. *arXiv:2005.10052 [cs, eess, stat]*, 2020.

I. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

H. Shao, A. Kumar, and P. T. Fletcher. The riemannian geometry of deep generative models. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 428–4288. IEEE, 2018. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00071.

Y. Shi, B. Paige, P. Torr, et al. Variational mixture-of-experts autoencoders for multi-modal deep generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

H.-C. Shin, N. A. Tenenholtz, J. K. Rogers, C. G. Schwarz, M. L. Senjem, J. L. Gunter, K. P. Andriole, and M. Michalski. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *International Workshop on Simulation and Synthesis in Medical Imaging*, LNCS, pages 1–11. Springer, 2018.

K. Shmelkov, C. Schmid, and K. Alahari. How good is my gan? In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, 2018.

C. Shorten and T. M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, 2019. ISSN 2196-1115.

J. D. Singer, J. B. Willett, J. B. Willett, et al. *Applied longitudinal data analysis: Modeling change and event occurrence*. Oxford university press, 2003.

N. Singh, J. Hinkle, S. Joshi, and P. T. Fletcher. Hierarchical geodesic models in diffeomorphisms. *International Journal of Computer Vision*, 117(1):70–92, 2016.

J. Snell, K. Ridgeway, R. Liao, B. D. Roads, M. C. Mozer, and R. S. Zemel. Learning to generate images with perceptual similarity metrics. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4277–4281. IEEE, 2017.

K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoder. In *29th Annual Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.

A. Subramanian. Pytorch-vae. https://github.com/AntixK/PyTorch-VAE, 2020.

T. M. Sutter, I. Daunhawer, and J. E. Vogt. Generalized multimodal elbo. In *International Conference on Learning Representations*, 2021.

M. Suzuki, K. Nakayama, and Y. Matsuo. Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*, 2016.

M. A. Tanner and W. H. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540, 1987. ISSN 0162-1459.

E. Thibeau-Sutre, M. Diaz, R. Hassanaly, A. Routier, D. Dormont, O. Colliot, and N. Burgos. Clinicadl: An open-source deep learning software for reproducible neuroimaging processing. *Computer Methods and Programs in Biomedicine*, 220:106818, 2022.

A. Thin, N. Kotelevskii, A. Doucet, A. Durmus, E. Moulines, and M. Panov. Monte carlo variational auto-encoders. In *International Conference on Machine Learning*, pages 10247–10257. PMLR, 2021.

I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf. Wasserstein auto-encoders. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

J. Tomczak and M. Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

B. O. Turner, E. J. Paul, M. B. Miller, and A. K. Barbey. Small sample sizes reduce the replicability of task-based fMRI studies. *Communications Biology*, 1(1):1–10, 2018.

N. J. Tustison, B. B. Avants, P. A. Cook, Yuanjie Zheng, A. Egan, P. A. Yushkevich, and J. C. Gee. N4ITK: Improved N3 Bias Correction. *IEEE Transactions on Medical Imaging*, 29(6):1310–1320, 2010a. ISSN 0278-0062, 1558-254X. doi: 10.1109/TMI.2010.2046908.

N. J. Tustison, B. B. Avants, P. A. Cook, Y. Zheng, A. Egan, P. A. Yushkevich, and J. C. Gee. N4itk: improved n3 bias correction. *IEEE transactions on medical imaging*, 29(6):1310–1320, 2010b.

A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

A. Valliani and A. Soni. Deep Residual Nets for Improved Alzheimer's Diagnosis. In *8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics - ACM-BCB '17*, pages 615–615, Boston, Massachusetts, USA, 2017. ACM Press. ISBN 978-1-4503-4722-8. doi: 10.1145/3107411.3108224.

A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

R. Vedantam, I. Fischer, J. Huang, and K. Murphy. Generative models of visually grounded imagination. In *International Conference on Learning Representations*, 2018.

A. Waheed, M. Goyal, D. Gupta, A. Khanna, F. Al-Turjman, and P. R. Pinheiro. Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection. *Ieee Access*, 8: 91916–91923, 2020. ISSN 2169-3536.

Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

S. Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4(1):66–82, 1960.

J. Wen, E. Thibeau-Sutre, M. Diaz-Melo, J. Samper-González, A. Routier, S. Bottani, D. Dormont, S. Durrleman, N. Burgos, and O. Colliot. Convolutional neural networks for classification of Alzheimer's disease: Overview and reproducible evaluation. *Medical Image Analysis*, 63:101694, 2020. ISSN 1361-8415. doi: 10.1016/j.media.2020.101694.

T. White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

C. Wolf, M. Karl, and P. van der Smagt. Variational inference with hamiltonian monte carlo. *arXiv preprint arXiv:1609.08203*, 2016.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

E. Wu, K. Wu, D. Cox, and W. Lotter. Conditional infilling gans for data augmentation in mammogram classification. In *Image analysis for moving organ, breast, and thoracic images*, pages 98–106. Springer, 2018.

M. Wu and N. Goodman. Multimodal generative models for scalable weakly-supervised learning. *Advances in neural information processing systems*, 31, 2018.

Z. Wu, S. Wang, Y. Qian, and K. Yu. Data augmentation using variational autoencoder for embedding based speaker verification. In *Interspeech 2019*, pages 1163–1167. ISCA, 2019. doi: 10.21437/Interspeech.2019-2248.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Z. Xiao, K. Kreis, J. Kautz, and A. Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*, 2020.

W. Xu, H. Sun, C. Deng, and Y. Tan. Variational autoencoder for semi-supervised text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

L. Yang, N.-M. Cheung, J. Li, and J. Fang. Deep clustering by gaussian mixture variational autoencoders with graph embedding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6440–6449, 2019.

Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International conference on machine learning*, pages 3881–3890. PMLR, 2017.

X. Yi, E. Walia, and P. Babyn. Generative adversarial network in medical imaging: A review. *Medical image analysis*, 58:101552, 2019. ISSN 1361-8415.

X. Yu, X. Zhang, Y. Cao, and M. Xia. Vaegan: A collaborative filtering framework based on adversarial variational autoencoders. In *IJCAI*, pages 4206–4212, 2019.

C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2018a.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018b.

M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen. D-vae: A variational autoencoder for directed acyclic graphs. *Advances in Neural Information Processing Systems*, 32, 2019.

Q. Zhao, Z. Liu, E. Adeli, and K. M. Pohl. Longitudinal self-supervised learning. *Medical Image Analysis*, 71:102051, 2021.

S. Zhao, J. Song, and S. Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 5885–5892, 2019.

T. Zhao, R. Zhao, and M. Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, 2017.

X. Zhu, Y. Liu, J. Li, T. Wan, and Z. Qin. Emotion classification with data augmentation using generative adversarial networks. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 349–360. Springer, 2018a.

Y. Zhu, M. Aoun, M. Krijn, J. Vanschoren, and H. T. Campus. Data Augmentation using Conditional Generative Adversarial Networks for Leaf Counting in Arabidopsis Plants. In *BMVC*, page 324, 2018b.

Y. Zhu, M. R. Min, A. Kadav, and H. P. Graf. S3vae: Self-supervised sequential vae for representation disentanglement and data generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6538–6547, 2020.

P. Zhuang, A. G. Schwing, and O. Koyejo. fMRI data augmentation via synthesis. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 1783–1787. IEEE, 2019.