



Deep generative models : over-generalisation and mode-dropping

Thomas Lucas

► To cite this version:

Thomas Lucas. Deep generative models : over-generalisation and mode-dropping. Artificial Intelligence [cs.AI]. Université Grenoble Alpes [2020-..], 2020. English. NNT : 2020GRALM049 . tel-03102554

HAL Id: tel-03102554

<https://theses.hal.science/tel-03102554>

Submitted on 7 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Thomas LUCAS

Thèse dirigée par **Jakob VERBEEK**, Université Grenoble Alpes
et codirigée par **Kartee ALAHARI**, CRCN, Inria

préparée au sein du **Laboratoire Jean Kuntzmann** dans l'**École
Doctorale Mathématiques, Sciences et technologies de
l'information, Informatique**

Modèles génératifs profonds : sur- généralisation et abandon de mode

Deep generative models: over-generalisation and mode-dropping

Thèse soutenue publiquement le **25 septembre 2020**,
devant le jury composé de :

Monsieur JOOST VAN DE WEIJER

DOCTEUR EN SCIENCES, UNIV. AUTONOME DE BARCELONE -
ESPAGNE, Rapporteur

Monsieur MATTHIEU CORD

PROFESSEUR DES UNIVERSITÉS, SORBONNE UNIVERSITÉ -
PARIS, Président

Monsieur THIERRY ARTIERES

PROFESSEUR, ÉCOLE CENTRALE DE MARSEILLE, Rapporteur

Madame CAMILLE COUPRIE

DOCTEUR-INGÉNIEUR, CENTRE DE RECHERCHE FACEBOOK -
PARIS, Examinatrice



Abstract

This thesis explores the topic of generative modelling of natural images, which is the task of fitting a data generating distribution. Such models can be used to generate artificial data resembling true data, or to compress images. We focus on the class of latent variable models which seek to capture the main factors of variations of an image into a variable that can be manipulated. In particular we build on two successful latent variable generative models, the generative adversarial network (GAN) and the variational autoencoder (VAE).

Recently, GAN models significantly improved the quality of images generated by deep models, obtaining very compelling samples. Unfortunately these models struggle to capture all the modes of the original distribution, i.e., they do not cover the full variability of the dataset. Conversely, likelihood based models such as VAEs typically cover the full variety of the data well and provide an objective measure of coverage. However, these models produce samples of inferior visual quality that are more easily distinguished from real ones. The work presented in this thesis strives for the best of both worlds: to obtain compelling samples while modelling the full support of the distribution. To achieve that, we focus on i) the optimisation problems, and ii) practical model limitations that hinder performance.

The first contribution of this thesis is a deep generative model that encodes global image structure into latent variables, built on VAEs, and autoregressively models low level detail. We propose a training procedure relying on an auxiliary loss function to control what information is captured by the latent variables and what information is left to an autoregressive decoder. Unlike previous approaches to such hybrid models, ours does not restrict the capacity of the autoregressive decoder to prevent degenerate models that ignore the latent variables.

The second contribution builds on the standard GAN model, which trains a discriminator network to provide feedback to a generative network. The discriminator assesses the quality of individual samples, which makes it hard to evaluate the variability of the entire set of generated data. Instead, we propose to feed the discriminator with *batches* that mix both true and generated samples, and train it to predict the ratio of true samples in the batch. These batches work as approximations of the distribution of generated images and allow the discriminator to approximate statistics of the data distribution. We introduce an architecture that is well suited to solve this problem efficiently, and show experimentally that our approach reduces mode collapse in GANs on two synthetic datasets, and obtains good results on the CIFAR10 and CelebA datasets.

The mutual shortcomings of VAEs and GANs can in principle be addressed by training hybrid models that use both types of objectives. In our third contribution, we show that usual parametric assumptions made in VAEs induce a conflict between them, leading to lackluster performance of hybrid models. We propose a solution based on deep invertible transformations, that learns a feature space in which usual assumptions can be made without weakening performance. Our approach provides likelihood computations in the image space while being able to take advantage of adversarial training. It obtains GAN-like samples that are competitive with fully adversarial models while improving likelihood scores over recent hybrid models.

Keywords: Generative modelling, unsupervised learning, generative adversarial networks, convolutional neural networks, variational autoencoders, computer vision, machine learning.

Résumé

Cette thèse explore le sujet des modèles génératifs, appliqués aux images naturelles. Cette tâche consiste à modéliser la distribution des données observées, et peut permettre de générer des données artificielles semblables aux données d'origine, ou de compresser des images. Nous nous focalisons sur les modèles à variables latentes, qui cherchent à résumer les principaux facteurs de variation d'une image en une variable qui peut être manipulée. En particulier, les contributions proposées sont basées sur deux modèles génératifs à variable latente: le modèle génératif adversarial (GAN) et l'encodeur variationnel (VAE).

Récemment, les GAN ont significativement amélioré la qualité des images générées par des modèles dits 'profonds', obtenant des images très convaincantes. Malheureusement ces modèles ont du mal à capturer tous les modes de la distribution d'origine, i.e. ils ne couvrent pas les données dans toute leur diversité. À l'inverse, les modèles basés sur le maximum de vraisemblance tels que les VAEs couvrent typiquement toute la variabilité des données; en outre, ils offrent un moyen objectif de mesurer cela. Malheureusement ces modèles produisent des échantillons de qualité visuelle inférieure, qui sont plus faciles à distinguer de vraies images. Le travail présenté dans cette thèse a pour but d'obtenir le meilleur des deux mondes: des échantillons de bonne qualité tout en modélisant tout le support de la distribution. Pour arriver à cela, nous nous focalisons sur i) les problèmes d'optimisation et ii) les limitations pratiques des modèles qui heurtent leur performance.

La première contribution de ce manuscrit est un modèle génératif profond qui encode la structure globale des images dans une variable latente. Ce modèle est basé sur le VAE, et utilise un modèle autoregressif pour capturer les détails de bas niveau. Nous proposons une procédure d'entraînement qui utilise une fonction de coût auxiliaire pour contrôler quelle information est capturée par la variable latente et quelle information est laissée à un décodeur autoregressif. Au contraire des précédentes approches pour construire des modèles hybrides de ce genre, notre modèle ne nécessite pas de contraindre la capacité du décodeur autoregressif pour éviter d'apprendre des modèles dégénérés qui ignorent la variable latente.

La deuxième contribution est bâtie sur le modèle du GAN standard, qui s'appuie sur un discriminateur pour guider le modèle génératif. Le discriminateur évalue généralement la qualité d'échantillons individuels, ce qui rend la tâche d'évaluer la variabilité des données difficile. À la place, nous proposons de fournir au discriminateur des ensembles de données, par 'batches', qui mélangent des vraies images et des images générées. Nous l'entraînons à prédire le ratio de vrais et de faux éléments dans l'ensemble. Ces batches servent d'approximation de la vraie distribution des images générées et permettent au discriminateur d'approximer des statistiques sur leur distribution. Nous proposons une architecture qui est bien adaptée pour résoudre ce problème efficacement, et montrons expérimentalement que notre approche réduit l'oubli de mode des GAN sur deux jeux de données synthétiques et obtient de bons résultats sur les datasets CIFAR10 et CelebA.

Les lacunes mutuelles des VAEs et des GANs peuvent, en principe, être réglées en entraînant des modèles hybrides qui utilisent les deux types d'objectif. Dans notre troisième contribution, nous montrons que les hypothèses paramétriques habituelles faites par les VAE produisent un conflit entre les deux, menant à des performances décevantes pour les modèles hybrides. Nous

proposons une solution basée sur des modèles profonds inversibles, qui entraîne un espace de features dans lequel les hypothèses habituelles peuvent être faites sans poser problème. Notre approche fournit des évaluations de vraisemblance dans l'espace des images tout en étant capable de tirer profit de l'entraînement adversaire. Elle obtient des échantillons de qualité équivalente au modèle pleinement adversaires tout en obtenant une meilleure vraisemblance, comparé aux modèles hybrides récents.

Mots-clefs : Modèles génératifs, apprentissage non supervisé, réseaux génératifs adversaires, réseaux neuronaux convolutionnels, vision par ordinateur, autoencoders variationnels, apprentissage automatique.

Acknowledgements

A vast amount of thanks go to Jakob, for being very supportive with all my projects through these last years. You were always generous with advice, scientific insight, and in general time. This created a positive, enjoyable work environment for which I am deeply grateful.

I had the chance to work with many collaborators who taught me much and made my day-to-day work interesting. I want to express my gratitude to Karteek, for his supervision during the end of my PhD. Marco, for showing me how to make experiments, and how to stay positive when they fail. Corentin for endless (sometimes) scientific discussions, valuable insight, and a very enjoyable and productive collaboration. Konstantin, for always being willing to solve my bugs, teaching me the proper use of a computer, and being an inexhaustible source of unexpected information. Piotr and Camille for their supervision during an internship, advice and friendly support.

Beyond my collaborators, I am grateful to many people, and for many reason. I have to cherry pick a few; in no particular order, I want to thank: Nathalie for always being quick to help with my many administrative errors and oversights. Philippe, Vicky, Xavier, Vlad and Daan for their warm welcome and help settling in. Vasilis, Ekaterina, Alex and Adria for being pleasant office mates. Nikita for many late night discussions about life, and many laughs. Dexiong for teaching me nice Chinese recipes, Alberto for his very resilient good mood, always finishing the guacamole and advice on skin care. Mathilde for being very supportive, coming up with many fun activities (and distractions) and generally boosting morale. I am thankful to all my other lab-mates at Inria, for entertaining coffee breaks and lunch discussions, and sharing their knowledge.

Aside from the work environment, it is hard to overstate how grateful I am for all the support and love I received from my family during all these years. Last but not least, I am very grateful to all jury members, who agreed to take some of their precious time to participate in my defense and review the manuscript.

Contents

1	Introduction	1
1.1	Unsupervised learning	2
1.2	Generative modelling	3
1.3	Challenges in generative modelling	5
1.4	Contributions to generative modelling	8
1.5	Publications	9
2	A primer on deep generative modelling	11
2.1	Introduction to generative modelling	12
2.2	Generative latent variable models	13
2.3	Linear latent variable models	14
2.4	Deep generative modelling	16
2.5	Variational auto-encoders	17
2.5.1	Deterministic auto-encoders	17
2.5.2	Variational auto-encoders	18
2.6	Deep invertible transformations	21
2.7	Generative adversarial networks	23
2.8	Autoregressive density estimation	25
2.9	Challenges in generative modelling	25
2.9.1	Understanding the conditional independence assumption	27
2.9.2	Understanding mode-dropping in adversarial networks	30
2.10	Details and refinements on Variational auto-encoders	32
2.10.1	Practical training algorithm	32
2.10.2	Top down sampling	33
2.10.3	Normalizing Flows for flexible posteriors	35
2.10.4	Importance weighted VAE	36
2.10.5	Quantized variational auto-encoders	37
2.11	Lipschitz continuity for generative adversarial networks	38
2.12	Evaluation metrics for generative models	42
2.12.1	Data dequantization	42
2.12.2	Discrete parametric densities	44
2.12.3	GAN evaluations	44

3	Auxiliary guided autoregressive VAE	47
3.1	Introduction	48
3.2	Related work	48
3.3	Auxiliary guided autoregressive variational autoencoders	50
3.3.1	Variational autoencoders with autoregressive decoders	50
3.3.2	Autoregressive decoders in variational autoencoders	51
3.3.3	Auxiliary Guided Autoregressive Variational Autoencoder	53
3.3.4	It is optimal for the autoregressive decoder to use the latent variable	55
3.4	Experimental evaluation	56
3.4.1	Dataset and implementation	56
3.4.2	Quantitative performance evaluation.	57
3.4.3	Effect of KL regularization strength.	58
3.4.4	Role of the auxilliary representation	59
3.4.5	Effect of different auxiliary images.	61
3.5	Conclusion	63
4	Mixed batches and symmetric discriminators for GANs	65
4.1	Outline of this chapter	66
4.2	Related work	68
4.3	Adversarial learning with permutation-invariant batch features	69
4.3.1	Batch smoothing as a regularizer	70
4.3.2	The optimal discriminator for batch smoothing	71
4.4	Permutation invariant networks	72
4.4.1	Building a permutation invariant architecture	73
4.4.2	A universal approximation theorem for permutation invariant functions	74
4.4.3	Practical architecture	76
4.5	Experiments	76
4.5.1	Synthetic 2D distributions	77
4.5.2	Experimental results on CIFAR10	77
4.5.3	Results on celebA and STL10 datasets	78
4.5.4	Interpretation of M-BGAN as an ensembling method	79
4.6	Conclusion	80
4.7	Optimal discriminator for general beta prior (*)	81
4.7.1	p_{balanced} and $p_{\text{unbalanced}}$ are well normalized:	84
4.8	Universal approximation theorem for symmetric functions (*)	85
4.8.1	Definitions	85
4.8.2	Algebraic structure of approximable functions	86
4.8.3	The closure contains a generative family of all symmetric functions (**)	90
5	Adaptive density estimation for generative models	95
5.1	Outline of this chapter	96
5.2	Related work	97
5.3	Preliminaries on MLE and adversarial training	99
5.4	Adaptive Density Estimation and hybrid adversarial-likelihood training	101

5.5	Experimental evaluation	103
5.5.1	Evaluation metrics	103
5.5.2	Ablation study and comparison to VAE and GAN baselines	104
5.5.3	Refinements and comparison to the state of the art	105
5.5.4	Results on additional datasets	107
5.6	Complementary evaluations	111
5.6.1	Evaluation using samples as training data for a discriminator	111
5.6.2	Model evaluation using precision and recall	112
5.7	Qualitative influence of the feature space flexibility	112
5.8	Conclusion	113
6	Conclusion	115
6.1	Summary of contributions	115
6.1.1	Going beyond conditional independence	115
6.1.2	Distributional statistics in adversarial training	116
6.1.3	Hybrid adversarial and maximum-likelihood based training	116
6.2	Future work	116
6.2.1	Video compression - Entropy coding and lossless compression	117
6.2.2	Representation learning	118
A	Adaptive density estimation: samples and other results	119
A.1	Additional Samples	120
A.1.1	Additional samples on CIFAR10 and STL10	120
A.1.2	Samples on all Lsun datasets	122
A.2	Implementation details	126
A.2.1	Visualisations of reconstructions	127
A.3	Coverage and Quality driven training algorithm	128
B	Areas of attention for image captioning	129
B.1	Introduction	129
B.2	Related work	131
B.3	Attention in encoder-decoder captioning	133
B.3.1	Baseline CNN-RNN encoder-decoder model	133
B.3.2	Attention for prediction and feedback	134
B.3.3	Areas of attention	135
B.4	Experimental evaluation	137
B.4.1	Experimental setup and implementation details	137
B.4.2	Experimental results	138
B.5	Conclusion	143

Chapter 1

Introduction

Several landmark areas of machine learning have seen remarkable improvement in the last several years, notably in computer vision with image classification [Krizhevsky et al., 2012, Simonyan and Zisserman, 2015], and segmentation [Long et al., 2015], in natural language processing [Bahdanau et al., 2015, Sutskever et al., 2014, Vaswani et al., 2017b] audio processing [van den Oord et al., 2016a], or game agents [Silver et al., 2016, Vinyals et al., 2019]. A well recognized catalyst for this progress has been the simultaneous availability of highly expressive models and huge datasets. The former is largely due to advancements in dedicated hardware and software and the latter to the internet, which produces vast amounts of data everyday and provides the means to crowd-source its processing. For instance, more than 300 hours of video are uploaded every minute, on average, to YouTube alone¹.

Similarly to classical computer programs, learning based models should accept instances from some input domain and (approximately) solve a chosen task relating to those inputs. A significant difference is that they do not require an alorist to explicitly specify rules to map the input to the solution. In particular, *supervised learning* works by observing numerous pairs of possible inputs and desired outputs, called labels. Mappings between the two are built by iteratively modifying some internal machinery, e.g. using gradient descent based methods [LeCun et al., 1998], until the model performs well. This paradigm is a potent tool to solve tasks where: i) it is impractical for an alorist to explicitly enunciate rules to solve it, such as object recognition and ii) large amounts of data and labels can be collected.

While the input data is often easy to collect - think images on the internet - training labels are almost always expensive. For instance, training an image classifier involves asking humans to manually tag images; see Figure 1.1 for examples of labels required by different supervised learning tasks. This expensive work limits the scale of the datasets that can be used and constitutes a prominent performance bottleneck on many computer vision tasks. A natural question arises in this context: can something useful be learned by observing raw data alone?

¹ See Youtube press statistics at <https://www.youtube.com/about/press/>, accessed 2020-07-08

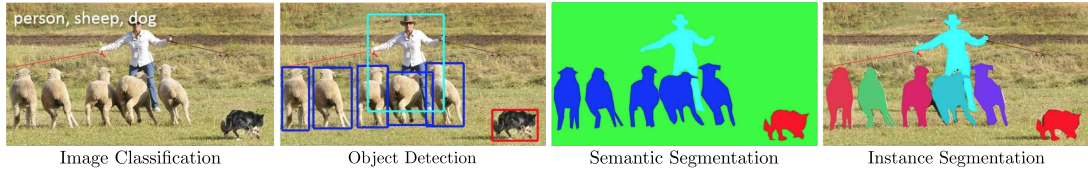


Figure 1.1: Classical supervised-learning tasks on natural images include image classification, object detection, semantic segmentation and instance segmentation. In that order, the supervision is ranked from weakest to strongest as each signal can be deduced from the next. These annotations provide high level summaries of the data, but their necessity limits the scale of the dataset considered. Figure adapted from [Lin et al. \[2014b\]](#)

1.1 Unsupervised learning

Raw data often contains a lot of information. Images, videos, text or audio clips all live in high dimensional spaces, are highly structured, and therefore constitute rich and complex signal. From the point of view of Shannon information theory [[Shannon, 1948](#)], natural images contain thousands of bits of information. In contrast labels often contain only a few bits of information, for instance in image classification. This motivates the pursuit of models that are able to exploit the information in unlabelled data. In the words of Yann LeCun:

”Most of human and animal learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, [...]. We know how to make the icing [...], but we don’t know how to make the cake. We need to solve the unsupervised learning problem before we can even think of getting to true AI.”

– Yann LeCun, invited talk at NeurIPS 2016.

Intuition suggests it should be possible to learn something about how the data is structured without labels. For instance observing photographs of an unknown object will allow the reader to recognize it in future photographs, though not to name it. Because a virtually unlimited amount of raw data is available for the first time in human history, this is an exciting research area with many yet unexplored corners. The goal of learning from unlabelled data is, however, an ill-posed endeavour and turning it into a problem that can be tackled requires specifications. The first is how to select or improve a model in the absence of objective targets, and the second is in what sense a model obtained this way can then be useful. The work presented in this manuscript is part of the rich topic of *unsupervised learning*, which aims at answering these questions.

While labels contain few bits of information, these bits are critical to defining the supervised approach. They can be seen as a compressed summary of parts of the input data as perceived by humans. This high level semantic information about the input provides a target to guide iterative improvement of the models. This is missing in the context of unsupervised learning, and it is apriori unclear how to define a useful objective for optimization. A popular approach is to extract some structure from the raw data, and use it as a target label as in a supervised approach. This approach is referred to as *self-supervised* training. For instance, an image can be cut into pieces, and the right arrangement used as target [[Doersch et al., 2015](#), [Noroozi and Favaro, 2016](#)].

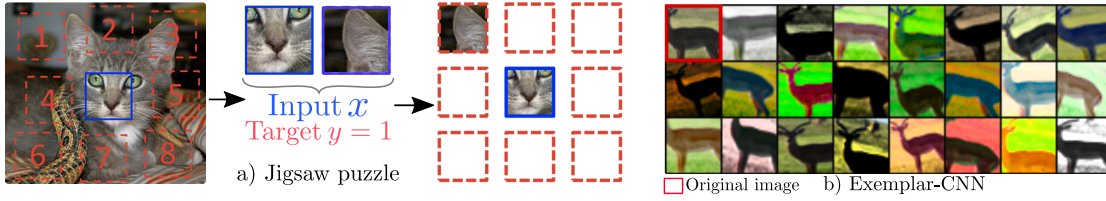


Figure 1.2: The Jigsaw-puzzle task involves cutting images to pieces and asking a model to reorder them correctly. In the exemplar-CNN task, an original image is transformed into many variants using data augmentations and a classifier must map all variants to the same target. In both cases, some target is extracted from the raw data without requiring human annotators. Figures adapted from Doersch et al. [2015] and Dosovitskiy et al. [2014] respectively.

Another possibility is to obtain many variants of each image using data augmentation and asking that a model classify all of them together. Both approaches are illustrated in Figure 1.2. They make seemingly arbitrary choices about the target signal being extracted. It is possible to not make any such choice by defining the task as "being able to predict the data". This is the high level idea of *generative modelling* and is the setting of the work presented in this manuscript.

1.2 Generative modelling

In generative modelling each data point is seen as the realisation of some random experiment. A simple example is to toss a dice many times, the data being the collection of values obtained. Similarly, natural images collected over the internet by a web crawler can be seen as realisations of some more complex random phenomenon. In general, the goal is to learn a probability density model as similar as possible to the data generating distribution. This kind of model can have direct applications such as data compression [Huffman, 1952] or data generation. It is also reasonable to assume that to perform well, such a model needs to build rich, abstract representations of the data.

The ability to predict the outcome of a random experiment with low uncertainty requires a form of *understanding* of it. For instance, a human that has good knowledge of the English language will be able to predict missing letters from a text with much better accuracy than random [Shannon, 1951]. This is because high level understanding of English implicitly provides the reader with a low entropy model. A similar experiment can be to remove a few patches from an image, and to ask a human to predict their content. An accuracy much better than random will be achieved, owing to good understanding of how natural images are likely to be structured. Fitting the data generating distribution requires the model to understand the data, and build usefull representations of it. These representations can then be used to solve other tasks, for instance by first training them with huge amounts of unlabelled data, then refining them using smaller amounts of labelled data. In this case the philosophy is that the abstractions learned by the generative model should be better representations of the data than its raw representation in the input space, and it can thus be called *representation learning*, see e.g. Bengio et al. [2012] for an overview.

The work presented in this manuscript is focused on modelling natural images. Though most

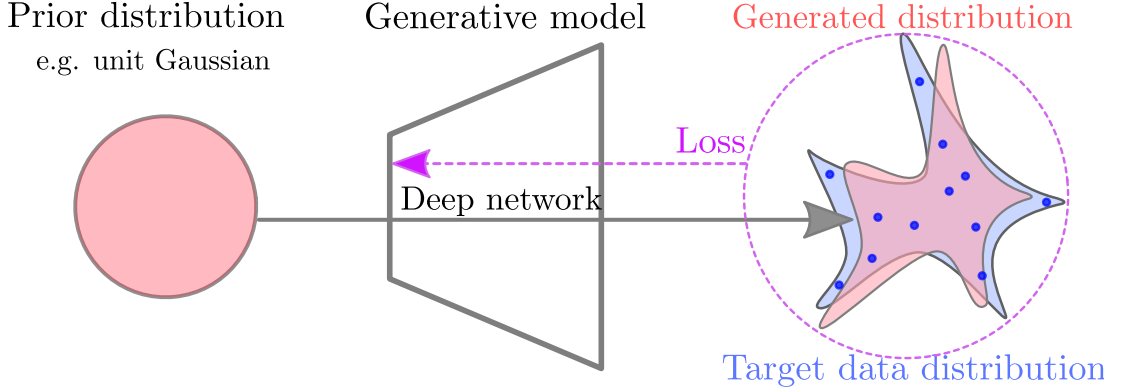


Figure 1.3: A simple distribution (depicted on the left by a level set of a unit Gaussian) is reshaped using a deep network into a more complex density. The generated and target distributions are then compared to each other, and this loss is used to improve the model.

of the discussion applies to other types of data, examples will be focused on images from now on. A natural image is composed of the scalar RGB values of many pixels, and the phenomena that control the relations between them are complex: for instance perspective, lighting or mechanics. These phenomena are too complex to be specified by hand and the goal is to use machine learning to model them. The probabilistic approach to this problem is to view the model as a distribution in image space. That distribution has to specify which combination of scalar values are likely to happen together in an image and which are not. This works by starting from a much simpler distribution, called a prior, over some space and using the observed data to learn a mapping that reshapes the prior into a more complex distribution as illustrated by Figure 1.3.

A model of the raw data can be leveraged in other problems. Many supervised machine learning problems can be formulated probabilistically as trying to maximize the probability of observing some target outcome y , given some input data x . Denoting x as the input data and y the target label, one typically seeks to maximize the probability y given x under some model, $p(y|x)$. Bayes' law [Bayes, 1763] can be invoked to see how the predictions $p(y|x)$ relate to generative modelling:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}. \quad (1.1)$$

This yields the conditional generative modelling task of fitting $p(x|y)$, using maximum likelihood estimation and labelled data, and the prior $p(y)$ can also be fitted using the labels. Unlabelled data can be leveraged by optimising the marginal $p(x)$ e.g. by gradient descent, to train $p(x|y)$. In terms of implementation, adding some conditioning to an unconditional generative modelling is typically easy; it can be done, for instance, by giving some extra inputs. To train the conditional model, labels are required. Thus this Bayesian view shows how to train simultaneously on labelled and unlabelled data, a problem called *semi-supervised learning*. Interestingly, modern deep-learning techniques enabled training of very complex priors over types of data that could not previously be considered.

Applications of generative models. Direct practical applications include information compression, which is one of the backbones of telecommunications. The intuition behind compression algorithms is that very probable events should be associated to short messages, while rare events should be transmitted with longer messages. Implementing this requires a density model of the data being transmitted and is thus a prime application of density estimation. Recent developments allow training of such model on very complex data, and thus hold the promise of compression gains, which can be very meaningful for telecommunications. Another direct and useful application is to generate real looking artificial data, which is possible if the learned generative model provides a way to sample new data from it. For instance, the goal can be to generate real looking image data to flesh out a virtual environment, after training on real scenes. This is useful in cinema and video games, and may become a crucial aspect of virtual worlds. More down to earth, generative modelling is also a good sandbox research problem for machine learning. Indeed, an almost infinite amount of low-cost data is available, and because the signal being fitted is rich and complex, over-fitting is unlikely to be an issue in the foreseeable future. This means that in this context the bottleneck will lie with model flexibility rather than data acquisition, and this is ideal for research.

Representation learning. Many self-supervised objectives can be seen as restoring the input given parts of it, so it is conceptually similar to generative modelling. With self-supervised pretext tasks, the targets can have far fewer bits of information than in generative modelling. In that sense, it appears more similar to the final regression tasks, as part of the information in the input can be discarded and more salient information kept to solve the task. However, because the pretext task in self-supervised learning is different from the final one, the information being discarded as irrelevant for the pretext task may be important for the target one. In contrast, the target of generative modelling is the data itself, and it is thus a very rich target with all the bits of information available. The model has to fit *everything* in the data to be optimal, and cannot discard less salient information. The representations learned may be more generic, and this forms a motivation for exploring generative modelling as a tool for representation learning. Figure 1.4 provides images samples from an unsupervised generative model that demonstrate the ability of the model to capture complex dependencies in the data.

1.3 Challenges in generative modelling

Training a model to fit a data distribution requires defining in what sense the model should match the target. A first guiding principle is that the model should be likely to generate the data that was actually observed when collecting the dataset. This is our best guess of what the model should produce and thus constitutes a reasonable target. If the model is likely to generate *any* point in the dataset then in some sense the model has captured the phenomenon in its entirety. A second principle is that the model should be able to generate artificial data that is realistic, but not a copy of a point in the training set, which shows that the model has learned how the data is structured. This approach requires defining a measure of quality for a sample, dependent on the observed

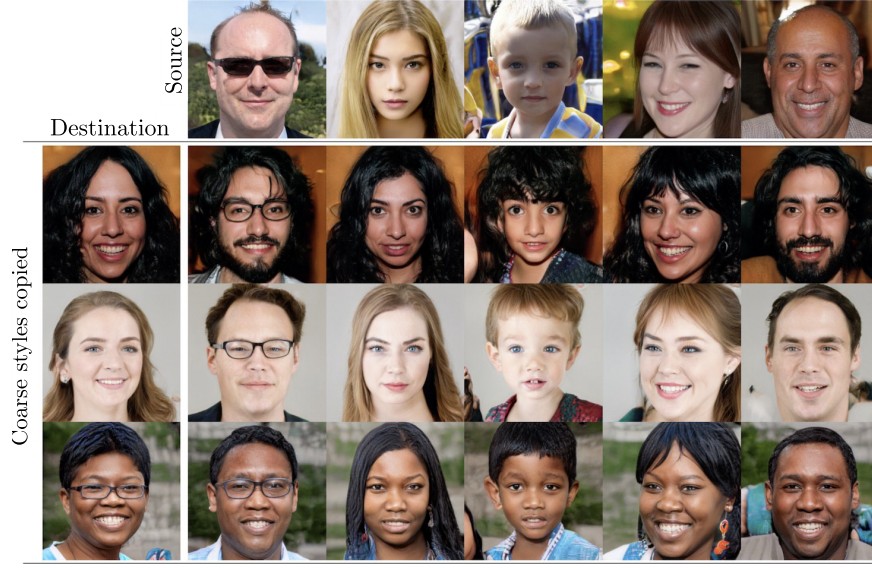


Figure 1.4: The images in the bottom grid are all generated by a generative model trained in an unsupervised manner. Samples can be conditioned on some desired style (skin colour, hair colour, hair texture) and on high level semantic information (apparent gender, head shape and pose, presence of glasses). The model is able to learn these complex notions by observing vast amounts of unlabelled data. Figure slightly adapted from [Karras et al. \[2019\]](#)

data. Both principles have their advantages and drawbacks, and are in some sense complementary. We now discuss some challenges in generative modelling that will be of core interest to the work presented in the rest of the manuscript.

Over-generalisation. Natural images are too complex for artificial models to capture their structure in all its finesse. Therefore training a model to cover all points in the observed data pushes it to accept wider sets of input than it should, in order to miss nothing. This phenomenon is illustrated in Figure 1.5. The lack of flexibility can stem from problematic assumptions, in particular on the loss being used. Given a target image, the model prediction will be imperfect, so the evaluation requires a notion of distance between images. Intuitively, the most common distances define concentric spheres around training points, and consider that all points on the same sphere are equally good approximations – the closer to the center the better. Such a distance is called isotropic and leads the model to assign probability to unrealistic data points. Note that autoregressive models sequentially compare pixels in the target and prediction, and thus there is no need for isotropic distances. These models still over-generalise, because of the incentive from the training criterion to not miss any data point, but loss design is less problematic in that context. A core focus of the work presented in this manuscript is to mitigate over-generalisation. This is achieved by i) foregoing isotropic distances and ii) leveraging other training paradigms.

Mode-dropping. The second principle to train generative models is to optimize the quality of generated data. It is achieved by sampling data points from the model and comparing these

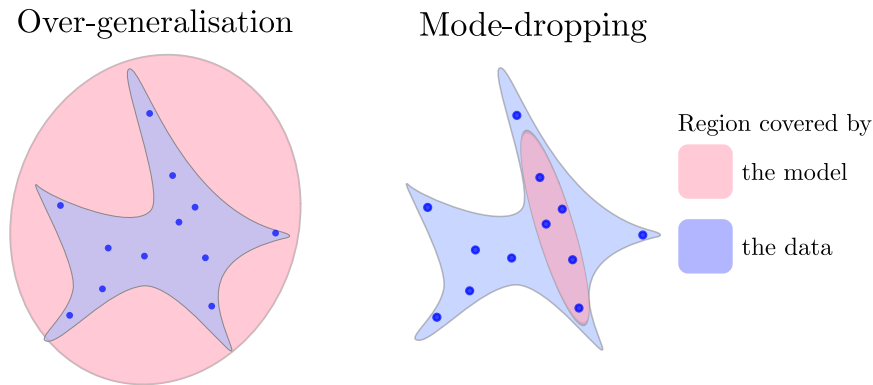


Figure 1.5: A model over-generalises if it covers the support of the true data distribution well, but also covers parts of the space that it should not, as depicted on the left. Conversely, a model that fails to cover the full support of the data distribution is said to mode-drop, as depicted on the right.

samples to real data. This approach is taken by the Generative Adversarial network model (GAN) [Goodfellow et al., 2014]. Here the evaluation metric is trickier to design, as there is not a single candidate to which the sample should be compared. An important aspect is that the variability in the generated data must also be considered. Indeed otherwise memorising a single image of high quality solves the problem, but this solution is not desirable. A much studied pitfall of GANs is that it is hard to obtain a model that fits the full variety of the data, a problem called mode-dropping.

Posterior collapse. One way to view the mapping from the prior distribution to the generated one (see Figure 1.3) is to see the variable as "causes" that determine the content of an image, and the generator as a mapping from causes to results. For tractability, it is beneficial to also have a model that performs the reverse operation, i.e. that maps an image to a possible explanation, which is referred to as an *encoder*. This is the core idea of the successful Variational auto-encoder model (VAE). A promising solution to tackle over-generalisation is to use an expressive type of decoder called an *autoregressive* model. These models do not require isotopic distances, and have a lot of complementarities with VAEs. However, in practice autoregressive decoders do not work well in conjunction with encoders. This is due to the fact that latent variables have a cost in terms of likelihood estimation. In the presence of sufficiently powerful autoregressive decoders it is easier and potentially even optimal not to use latent variables, a phenomenon known as *posterior collapse*.

Hybrid models. The two learning principles presented here, maximizing the probability of observed data and optimizing sample quality, intertwine and are complementary. Ideally, a model should simultaneously satisfy both types of evaluation: if a model fits the true distribution well, it both covers all the variety of the data, and produces high quality samples. In practice, optimizing both types of objectives together is challenging, and is another focus of our work.

1.4 Contributions to generative modelling

The contributions presented in this manuscript focus on the problems of over-generalisation, mode-dropping, posterior collapse, and successful training of hybrid models. These problems intertwine, and they all relate to the goal of obtaining models that cover the full variety of the data while producing compelling samples. To put contributions into context, sections 2.1 to 2.8 will introduce several successful approaches in generative modelling. A discussion on some limitations of these models is then given in Section 2.9. This is followed by refinements on the basic models in sections 2.10 to 2.11, and a presentation of evaluation procedures in Section 2.12 to conclude the background section.

Variational autoencoders with autoregressive decoders. These models are among the most successful approaches to generative modelling of natural image collections. These approaches have complementary strengths as VAEs handle global structure well, while autoregressive models excel at modelling local image statistics. This motivates hybrid models that encode global image structure using an encoder while autoregressively modeling low level detail. Naive constructions of this type unfortunately yield models that are unable to use the encoder. In Chapter 3, we propose a training procedure relying on an auxiliary loss function that controls which information is captured by the encoder and what is left to the autoregressive decoder. Unlike previous approaches to such hybrid models which restricted the capacity of the autoregressive decoder, we are able to use arbitrarily expressive decoders. Our approach achieved state-of-the-art quantitative performance among models with latent variables at the time of publication, and generates qualitatively convincing samples.

Training GANs by assessing batches of data. Generative adversarial networks evaluate the quality of generated samples using a discriminator network that distinguishes between real and generated images. In existing models the discriminator assesses individual samples. This prevents the discriminator from accessing global distributional statistics of generated samples and leads the generator to mode-drop. In Chapter 4 we propose to use the discriminator to assess *batches* of data that mix both true and fake samples. The task of the discriminator becomes to predict the proportion of real and generated images in the batch. By doing that, variability in the generated data becomes something that can be explicitly evaluated by the discriminator. We show that our approach reduces mode collapse in GANs on two synthetic datasets, and obtains good results on the CIFAR10 and CelebA datasets, both qualitatively and quantitatively.

Adversarial and maximum likelihood hybrid training. In Chapter 5, we show that parametric assumptions commonly made about the output density of maximum-likelihood based models are a source of tension with adversarial training, making successful hybrid models non trivial. We propose using an invertible model to learn an abstract feature space to which targets and predictions are mapped. In that space the distance between them can be computed without hindering hybrid training. We show that compared to existing hybrid models, our model offers improved sample quality and likelihood scores.

1.5 Publications

- Chapter 3 is based on the paper "Auxiliary Guided Autoregressive Variational Autoencoder", Thomas Lucas & Jakob Verbeek, European Conference on Machine Learning (ECML) 2018. See [[Lucas and Verbeek, 2018b](#)].
- Chapter 4 is based on the paper "Mixed batches and symmetric discriminators for GAN training", Thomas Lucas, Corentin Tallec (equal contribution), Jakob Verbeek and Yann Ollivier, International Conference on Machine Learning (ICML) 2018. See [[Lucas et al., 2018b](#)]
- Chapter 5 is based on the paper "Adaptive Density Estimation for Generative Models", Thomas Lucas, Konstantin Shmelkov (equal contribution), Karteek Alahari, Cordelia Schmid, and Jakob Verbeek, conference on Neural Information Processing Systems (NeurIPS) 2019. See [[Lucas et al., 2019](#)].
- A research project initiated during my master internship and completed during the first semester of this Ph.D, led to the paper "Areas of Attention for Image Captioning", Marco Pedersoli, Thomas Lucas, Cordelia Schmid and Jakob Verbeek, International conference on computer vision (ICCV) – see also [[Pedersoli et al., 2017](#)]. It proposes an attention mechanism for image captioning - the task of describing input images with sentences - that allows the model to attend relevant parts of the image while describing it. This requires high level representations of complex image content, as with image generation. However because the topic differs from the other contributions, we left it out of the main body of the manuscript and it can be found in Appendix B.

Chapter 2

A primer on deep generative modelling

Contents

2.1	Introduction to generative modelling	12
2.2	Generative latent variable models	13
2.3	Linear latent variable models	14
2.4	Deep generative modelling	16
2.5	Variational auto-encoders	17
2.6	Deep invertible transformations	21
2.7	Generative adversarial networks	23
2.8	Autoregressive density estimation	25
2.9	Challenges in generative modelling	25
2.10	Details and refinements on Variational auto-encoders	32
2.11	Lipschitz continuity for generative adversarial networks	38
2.12	Evaluation metrics for generative models	42

In this chapter some background on generative modelling is first recalled, which also serves to set the notations for the rest of the dissertation. Latent variable generative models are introduced in sections [2.2](#) and [2.3](#), starting from linear models such as the Gaussian Mixture Model. The extension to deep generative models is presented in Section [2.4](#), with the main difficulties that arise when going to the deep regime. The foundations of existing approaches to deep generative modelling are discussed in sections [2.5](#) to [2.8](#). This is sufficient to discuss, in Section [2.9](#) some of the challenges that will be presented in the rest of the dissertation. We then go into a more detailed presentation of existing work in Section [2.10](#) and Section [2.11](#). Finally, we discuss the evaluation of generative models in Section [5.5](#).

2.1 Introduction to generative modelling

The goal of unconditional density modelling is to learn the distribution p^* underlying samples provided in a dataset $D = \{x_1, \dots, x_M\}$ and seen as realisations of a random variable X on some space \mathcal{X} , with density at x equal to $p^*(X = x)$ ¹. In what follows, the dataset is composed of N -dimensional real valued vectors, $x \in \mathbb{R}^N$. Thus, $\mathcal{X} \subset \mathbb{R}^N$ and \mathbb{R}^N is called the *ambient* space. In practice, we focus on natural images datasets, and a vector x contains per pixel intensity values across colour channels. The parametric approach to density estimation is to select a family of parametric densities, $\mathcal{P}_\Theta = \{p_\theta | \theta \in \Theta\}$, with Θ the set of admissible parameters. Learning then seeks to select the 'best' parameter θ^* in Θ , which requires a measure of performance for θ . Since the goal is to recover p^* , this measure, or "loss", will typically be, or behave like, a distance between p_θ and p^* , and can be denoted $\mathcal{L}(p_\theta, p^*)$.

There are different approaches to designing the loss \mathcal{L} . We will focus on unconditional density modelling, meaning that the estimator $\hat{\theta}$ for θ^* is obtained from unlabelled data. Intuitively, there are two things we expect from a "good" model:

- The model p_θ assigns high density to samples taken from the true distribution p^* :

$$x \sim p^*(x) \implies p_\theta(x) \text{ is "high"}.$$

- Samples taken from the model p_θ behave similarly to real samples from p^* :

$$x \sim p_\theta(x) \implies p^*(x) \text{ is "high"}.$$

These two properties are closely coupled through the fact that a density model is *normalised*. However, focusing on one goal or the other leads to different choices for $\mathcal{L}(p_\theta, p^*)$, and different behaviours for the model p_θ . This distinction coarsely separates existing models. The first type of objective, which we will refer to as "coverage driven" is more convenient to work with as it only requires samples from p^* . It is the logic behind most learning algorithms, and notably leads to maximum-likelihood-estimation (MLE). This class of models is discussed in sections 2.5, 2.6 and 2.8 and is the focus of the work presented in Chapter 3. The second type of objective, which we will refer to as "quality driven" is less obvious to design, as a straightforward implementation would require access to p^* . A popular instance of this approach is the Generative Adversarial Network (GAN), which is presented in Section 2.7. GANs are also the main object of the work presented in Chapter 4. Complementarities and some conflictual aspects of coverage driven and quality driven approaches are discussed in Section 2.9.2, and are a key aspect of the work presented in Chapter 5. As regards practical experiments, this presentation is focused on modelling natural images, though in theory all models presented can be applied to other types of data. Modelling of natural images is a good sandbox research problem, as the data is complex, diverse, and at the same time easy to collect in large amounts. Such generative models also have interesting applications such as compression or automatic editing of images.

¹ Usually there is no ambiguity on the random variable under consideration and we write $p(x)$ instead of $p(X = x)$. Depending on context, $p(x)$ can be either a discrete or a continuous density.

Aside from the training objective and procedures, a family of parametric densities has to be specified. The data lives in a high dimensional space, is highly non-linear and can be very diverse which makes it challenging to model. Thus, "deep" approaches are very successful: \mathcal{P}_Θ is implemented by a very flexible, over-parametrised and non-convex function approximator, and $\hat{\theta}$ is selected by performing gradient descent on the loss. In sections 2.2 to 2.4, deep models and the challenges they raise are introduced. In sections 2.5 to 2.8, the main successful deep models are described, followed in Section 2.9 by limitations of these models that motivate the rest of this dissertation.

2.2 Generative latent variable models

High-dimensional real-valued data such as images are naturally modelled by a joint probability distribution over scalar values (in the case of images, three scalars per pixel for RGB colours channels). A popular approach to define that density is to use latent variables. Suppose that an image $\tilde{x} \in \mathbb{R}^N$ is to be sampled from a model p_θ and that we wish to sample all pixels simultaneously. To obtain an image that is globally coherent, it is clear that the pixels can not be sampled independently from each other. Indeed once one pixel is sampled, this partly determines others; if the top-left pixel belongs to a car, neighbouring ones likely belong to a car as well. That means $p(x)$ cannot simply be decomposed as a product of marginals $\prod_{i=1}^N p(x_i)$, which would be the naive way of allowing simultaneous sampling.

Latent variables can be seen as a solution to this issue. In latent variable generative models, the assumption is made that "most" of the variability in the data can be explained by a certain number of factors of variations. Such factors could be object classes and locations, or the angle taken for camera projection. When generating an image, a latent representation z is chosen first, and \tilde{x} is generated given z . It becomes possible to sample all pixels simultaneously, as their global coherence is controlled by z :

$$p(x|z) = \prod_{i=1}^N p(x_i|z). \quad (2.1)$$

This assumption is called *conditional independence*. With this model, most of the relationship between the pixels are captured through z , and the rest of the variability in the signal is modelled as independent per-pixel "noise". This is very different from full independence: in the extreme case, $p(x|z)$ can tend towards a deterministic function of z , i.e. there exists some f such that $p(x|z) \approx \delta(x - f(z))$.

Intuitively, a latent variable generative model p_θ is considered representative of the data D if for all x in D , there are some probable settings of z that are likely to generate x . Formally, assume latent variable vectors z to be realisations of a random variable Z in a high² dimensional space \mathcal{Z} , with a probability density $p(z)$ defined over \mathcal{Z} , then one can integrate z out to measure $p(x)$:

$$p(x) = \int_{z \in \mathcal{Z}} p(x|z)p(z)dz \quad (2.2)$$

² High, but still probably lower than N which is "very" high.

Given \mathcal{Z} and $p(z)$, a deterministic function $f_\theta : \mathcal{Z} \rightarrow \mathbb{R}^N$ can be used to specify the relation between z and \mathbf{x} , and obtain a random variable in image space by computing $Y = f_\theta(Z)$. In that case $p_Y(\mathbf{x}|z) = \delta(\mathbf{x} - f_\theta(z))$. Such a construction is hard to use directly in practice as is unlikely to pass through all points in the dataset exactly. Unless f is already perfect there exists \mathbf{x} such that $p(\mathbf{x}) = 0$ and in turn $p(D) = 0$. So $p(\mathbf{x})$ cannot be used to train f unless f is already perfect. We say that $f_\theta(Z)$ has a *degenerate support*. Instead, f_θ is usually used to *parametrize* a density that has non-degenerate support. For instance using $f_\theta(z)$ as the mean of a standard Gaussian density:

$$p(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}|f_\theta(z); \mathbb{I}_N). \quad (2.3)$$

Now, a single fixed z is mapped to a density which is strictly greater than 0 in the entire ambient space \mathbb{R}^N , as for any fixed (\mathbf{x}, z) , $p(X = \mathbf{x}|Z = z) > 0$. The Gaussian density is unimodal and has light tails, so it typically cannot cover the full data-set well³. When integrating z , on the other hand, a rich mixture of densities, weighted by $p(z)$, is obtained:

$$p(\mathbf{x}) = \int_z \mathcal{N}(\mathbf{x}|f_\theta(z); \mathbb{I}_N) p(z) dz. \quad (2.4)$$

This shows how to build complex densities over high dimensional vectors using latent variables and the conditional independence assumption to model the dependencies between each scalar dimension. An other option to build a density estimator is to drop the requirements that all scalar values x_i be sampled simultaneously. Intuitively, in that case, there is no longer a need for latent variables: one can sample x_1 , look at the result, then sample x_2 given x_1 and continue. Conditioning on some global information is no longer needed. What this costs is that sampling will be slow and sequential, and that a (possibly arbitrary) ordering needs to be introduced. This construction is discussed in Section 2.8.

2.3 Linear latent variable models

A simple case of latent variable generative model is when f_θ is a linear function of z . A classical example is the Gaussian Mixture Model (GMM). In that case the latent variable z is a discrete one-hot vector that selects a component in a mixture of K component. Denote $\mathbf{1}_k = (\delta_k^1, \dots, \delta_k^K)$ where δ_i^j is the Kroenecker symbol, then:

$$p(z = \mathbf{1}_k) = \pi_k, \text{ and } \mu_k = Wz + \mu \quad (2.5)$$

$$p(\mathbf{x}|z) = \mathcal{N}(\mathbf{x}; \mu_k, \sigma I_D). \quad (2.6)$$

In this case the integration over z is a finite sum: $p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}|z)$. It is illustrated in Figure 2.1, and shows how simple densities can be combined into a more complex one.

A similar recipe can be followed to obtain Probabilistic Principal Component Analysis (Probabilistic PCA Roweis [1997], Tipping and Bishop [1999]). In that case, z is sampled from a

³ The density decreases exponentially with the squared euclidean distance to $f_\theta(z)$, and most data points \mathbf{x} are far from $f_\theta(z)$

standard Gaussian density, and also linearly transformed:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I_d) \quad (2.7)$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu + W\mathbf{z}, \sigma I_D). \quad (2.8)$$

In that case $p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ has a closed form solution and is likewise Gaussian with

$$\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \mu, WW^t + \sigma I_D). \quad (2.9)$$

The number of columns in W corresponds to the number of *principal components* that will be fitted by the model. Note that for both models, the negative log-likelihood gives the ℓ_2 “reconstruction” loss of deterministic PCA and k -means:

$$-\ln p(\mathbf{x}|f_\theta(\mathbf{z})) = \|\mathbf{x} - f_\theta(\mathbf{z})\|_2^2, \quad (2.10)$$

with $f_\theta(\mathbf{z}) = \mu + W\mathbf{z}$. Sampling $\mathbf{z} \sim p(\mathbf{z})$ and mapping it to $p(\mathbf{x}|\mathbf{z})$ yields samples $\tilde{\mathbf{x}} \sim p_\theta(\mathbf{x})$. Note that the mode of $p(\mathbf{x}|\mathbf{z})$ is often used as reconstruction rather than taking a sample, especially with isotropic Gaussian noise or other isotropic densities that cannot capture any structure.

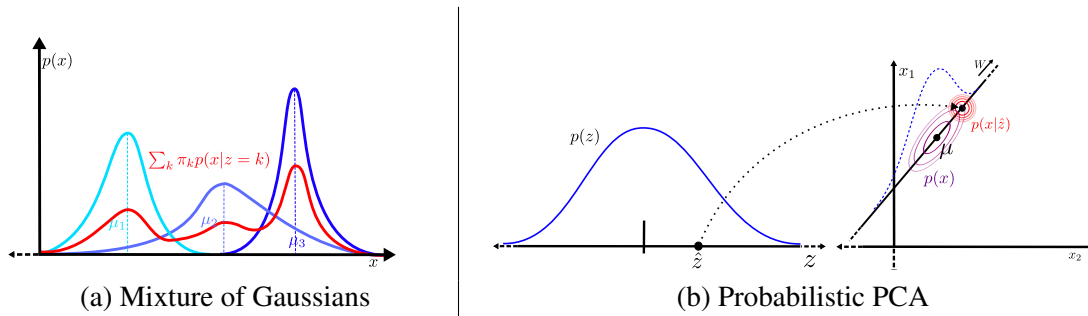


Figure 2.1: (a) A Gaussian Mixture Model is a convex combination of Gaussian densities, which allows for greater flexibility. The parameters of each component can be learned, as well as the weighting coefficients. This can be seen as a latent variable model, where \mathbf{z} selects a component: $p(\mathbf{z} = \mathbf{1}_k) = \pi_k$ which is then linearly mapped to $\mu_k = W\mathbf{z} + \mu$. (b) Probabilistic PCA transforms a standard Gaussian through a linear mapping. Figures adapted from Bishop [2006].

In both cases, the Gaussian noise adds volume around (points on) a linear manifold to make the support non-degenerate in the data space. This works well for many applications, but natural images are highly complex and linear manifolds may not be flexible enough to cover their support well. Therefore a lot of volume must be added, wasting density mass and yielding poor samples and poor likelihoods. Instead, it is desirable to build more flexible, non-linear manifolds f_θ to better fit complex data. This is illustrated in Figure 2.2, and motivates the next section.

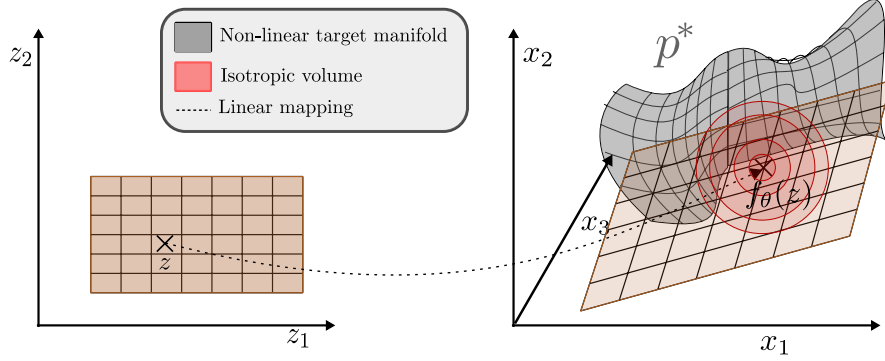


Figure 2.2: The linear model f_θ maps Z to the linear manifold (in brown) that best fits $p^*(x)$ (in grey). It is necessary to add volume (in red) around $f_\theta(z)$ to obtain a non-degenerate density $p_\theta(x|z)$ that does cover the support of p^* . Because the data manifold is non-linear, the model is far from the target in some places and a lot of volume must be added, thus wasting probability mass.

2.4 Deep generative modelling

For highly complex data such as natural images, it is natural to consider models that can learn non-linear manifold and thus more closely fit the data. The idea remains similar: a simple distribution $p(z)$ on latent variable z , e.g. a standard Gaussian, is transformed through a non-linear function $x = f_\theta(z)$ that maps latent variables to data space. For this purpose, a deep neural network can be used⁴. It will induce a complex marginal distribution $p_\theta(x) = \int_z p(z)p(x|f_\theta(z))$ when integrating out z . The non-linear manifold $f_\theta(z)$ is now arbitrarily flexible and can in theory approximate any function. If enough data is provided, or if proper regularization is used to avoid over-fitting the data, very expressive networks can be considered, and closely fit the data.

Using a non-linear f_θ also raises a thorny issue: the evaluation of $p_\theta(x)$ becomes intractable. Indeed the integral involving non-linear deep net $f_\theta(\cdot)$ can no longer be computed in closed form. Various solutions to side-step this problem exist, and lead to different classes of models which we will introduce in the following sections. The first solution is to use a *lower bound* that is tractable, as in Variational Auto-Encoders (VAE), presented in sections 2.5 and 2.10. It is also possible to *constrain* the parametric family for f_θ so that we can compute $p_\theta(x)$ for instance using a flow based method as presented in Section 2.6. Another solution is to train generative models *without* using the integral but rather by evaluating sample quality, as in Generative Adversarial Networks (GAN) (sections 2.7 and 2.11). Finally, it is possible to design models that do not use latent variables at all but rely on chaining many uni-variate conditional distributions instead, as

⁴ Manually specified basis expansion is prohibitively expensive in high-dimension. Kernel methods provide well understood and powerful basis expansions, but decouple learning from feature construction and still require kernel design.

discussed in Section 2.8. All of these basic models are used in various parts of this manuscript.

2.5 Variational auto-encoders

Using expressive non-linear classes of functions for f_θ holds the promise of more accurate models, but also makes computations challenging. For latent variable models, one problem made more complicated by the use of deep functions is the problem of inference. Given an \mathbf{x} and a model $p_\theta(\mathbf{x}|\mathbf{z})$, inference consists in finding a (set of) latent variables \mathbf{z} that are "good explanations" for \mathbf{x} , in the sense that they are likely to generate \mathbf{x} . Unfortunately, $p_\theta(\mathbf{z}|\mathbf{x})$ is completely intractable for feed-forward networks in general. We now study how to use of an inference network to compute and optimise a lower-bound on Equation 2.2.

2.5.1 Deterministic auto-encoders

We have seen in Section 2.3 that PCA can interpreted as a probabilistic model. In PCA, one seeks matrices V and W that minimize the loss [Baldi and Hornik, 1989]:

$$\min_{V,W} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}_n - VW\mathbf{x}_n\|^2. \quad (2.11)$$

The matrix W can be seen as projecting \mathbf{x} into a latent space and V tries to reconstruct \mathbf{x} from \mathbf{z} . To go to the non-linear regime, V and W can be replaced by deep networks g_ϕ and f_θ by stacking many linear layers with non-linearities in between. The problem then becomes:

$$\min_{\phi,\theta} \sum_{n=1}^N \|\mathbf{x} - f_\theta(g_\phi(\mathbf{x}))\|^2. \quad (2.12)$$

The first network g_ϕ maps \mathbf{x} to a space of lower dimensionality, and is called an *encoder*. The second network f_θ maps the latent code $\mathbf{z} = g_\phi(\mathbf{x})$ to reconstruction $\tilde{\mathbf{x}}$ and is called a *decoder*. This construction is referred to as the auto-encoder model. It is a generalization of PCA (PCA is recovered in the special case of g_ϕ and f_θ linear) that tries to capture the main factors of variations in \mathbf{x} in a non-linear way. It does not provide a generative model that can be sampled from, or an evaluation of likelihood. However, the decoder resembles a generative model: the L_2 loss can be seen as putting isotropic Gaussian noise around f_θ , yielding a latent variable model $p_\theta(\mathbf{x}|\mathbf{z})$. The encoder, on the other hand, tries to *infer* a latent variable \mathbf{z} for a given \mathbf{x} i.e. to extract the main factors of variations in \mathbf{x} . This kind of inference network will prove useful in what follows.

It is important to note that some *bottleneck* is needed on the amount of information about \mathbf{x} that goes into \mathbf{z} . This can be achieved by limiting the dimension of \mathbf{z} . Otherwise, g_ϕ and p_θ can simply *both* learn the identity function, which is a trivial solution to Equation 2.12 but does not learn anything useful. Note that a good inference network learns to approximately invert f_θ , such that $f_\theta \circ g_\phi \approx Id$. This is *different* from having $f_\phi \approx f_\theta \approx Id$, which is not useful. A probabilistic extension can be given to the auto-encoder model, yielding a generative model called the Variational Auto-Encoder Kingma and Welling [2014a] which we describe below.

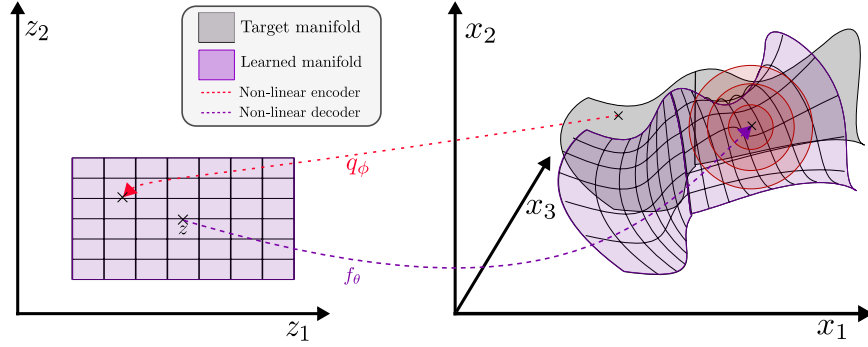


Figure 2.3: If f_θ is implemented with a deep network, $f_\theta(z)$ is distributed on a non-linear manifold. It can better fit p^* , and less isotropic volume is needed. The encoder q_ϕ takes target data and maps it to latent space, and the decoder f_θ maps latent variables to data space.

2.5.2 Variational auto-encoders

To build a stochastic auto-encoder, the decoder f_θ implements $p_\theta(\mathbf{x}|\mathbf{z})$ by mapping a latent code \mathbf{z} to a density over observations \mathbf{x} :

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_\theta^\mu(\mathbf{z}), f_\theta^\sigma(\mathbf{z})) \quad (2.13)$$

The encoder g_ϕ computes an approximation of the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ as:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})). \quad (2.14)$$

In that context, another type of bottleneck can be used: a prior $p_\theta(\mathbf{z})$ can be defined, and the amount of information going into \mathbf{z} can be measured as:

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \left(\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \right), \quad (2.15)$$

with D_{KL} standing for Kullback-Liebler divergence. Intuitively, the Kullback–Leibler divergence measures the difference in the expected number of bits required to encode \mathbf{z} using $q_\phi(\mathbf{z}|\mathbf{x})$ rather than $p_\theta(\mathbf{z})$, i.e. how much information (in the sense of Shannon) about \mathbf{z} we gain by looking at \mathbf{x} using our inference network. With this bottleneck the dimension bottleneck is no longer needed, and the model offers a probabilistic interpretation. In particular, \mathbf{z} can be sampled from the prior $p(\mathbf{z})$ and mapped to data space with f_θ . This construction is detailed in what follows. The quantity of interest that we wish to optimise is the marginal likelihood, $p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$, also called the “evidence”. Because f_θ is now non-linear, this integral can not be carried out in closed form.

Natural solutions to the intractability of the evidence are numerical approximations and Monte-Carlo estimation. The later has the benefit of spending computational budget only on likely values

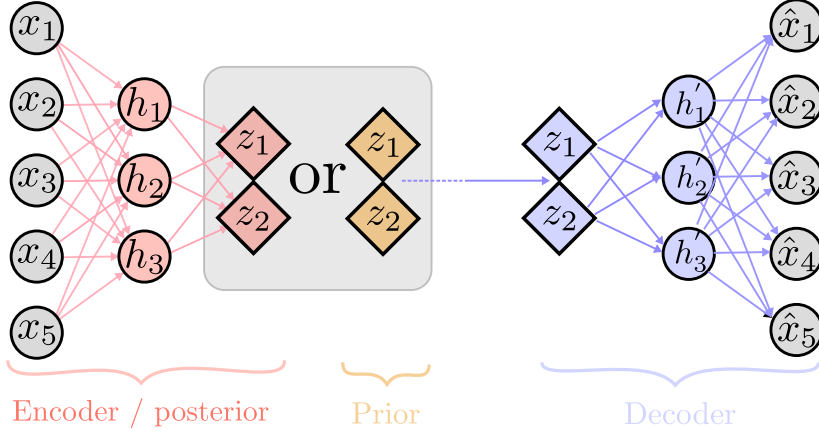


Figure 2.4: The encoder (in red) yields a distribution $q_\phi(z|\mathbf{x})$. Using this distribution rather than the prior (in yellow) $p_\theta(z)$ to sample \mathbf{z} incurs a cost $D_{KL}(q_\phi(z|\mathbf{x})||p_\theta(z|\mathbf{x}))$. Given \mathbf{z} , a deep decoder (in blue) defines a distribution $p_\theta(\mathbf{x}|\mathbf{z})$.

of \mathbf{z} rather than on a fixed grid covering \mathbb{R}^N . However, the latent space is typically very high dimensional, and obtaining an accurate approximation requires a high number of samples per the curse of dimensionality.

To alleviate this issue, a first step is to use weighted sampling. The intuition is that when computing $\int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$ for a fixed \mathbf{x} , most of the latent variables \mathbf{z} are very unlikely to generate \mathbf{x} , and most of the mass in the integral is contributed by a small subset of the entire latent space. To that end, an inference network q_ϕ can be used. For a given \mathbf{x} , q_ϕ tries to guess which part of the latent space is likely to generate \mathbf{x} and is used to reweight the integral:

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}d\mathbf{z}. \quad (2.16)$$

When approximating $p_\theta(\mathbf{x})$ with an empirical estimator $\hat{p}_\theta(\mathbf{x})$ by sampling \mathbf{z} from $q_\phi(\mathbf{z}|\mathbf{x})$, one obtains an estimator that has faster convergence and lower variance than when sampling from $p(\mathbf{z})$, while remaining unbiased. However, we are typically interested in computing $\log(p_\theta(\mathbf{x}))$. Because $\log[\mathbb{E}(\hat{p}_\theta(\mathbf{x}))] \neq \mathbb{E}[\log(\hat{p}_\theta(\mathbf{x}))]$, the empirical estimator $\log(\hat{p}_\theta(\mathbf{x}))$ is a biased estimator of $\log(p_\theta(\mathbf{x}))$. This brings Jensen's inequality to mind: because \log is concave, $\log(p_\theta(\mathbf{x})) = \log[\mathbb{E}(\hat{p}_\theta(\mathbf{x}))] \geq \mathbb{E}[\log(\hat{p}_\theta(\mathbf{x}))]$. By replacing the expectation by Monte-Carlo sampling, we can thus obtain an unbiased empirical estimator of a lower bound on the evidence.

Deriving a variational evidence lower bound. Once again, samples used in the Monte-Carlo estimation of $\mathbb{E}[\log(p_\theta(\mathbf{x}))]$ can be taken from $q_\phi(\cdot|\mathbf{x})$ rather than from $p_\theta(\cdot)$. This yields a so-called *variational lower bound*, that can be derived by combining reweighted sampling (first

line) with Jensen inequality (second line):

$$\ln(p_\theta(\mathbf{x})) = \ln \left(\int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x}|\mathbf{z}) \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \right) \quad (2.17)$$

$$\geq \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \ln \left(p_\theta(\mathbf{x}|\mathbf{z}) \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right) d\mathbf{z} \quad (2.18)$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\ln(p_\theta(\mathbf{x}|\mathbf{z}))] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \quad (2.19)$$

This yields an evidence lower bound (ELBO), denoted $\mathcal{L}_{\theta,\phi}^{elbo}$. The ELBO offers an *auto-encoder* interpretation:

$$\mathcal{L}_{\theta,\phi}^{elbo}(\mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \ln p_\theta(\mathbf{x}|\mathbf{z})}_{\text{Reconstruction}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{Regularization}}. \quad (2.20)$$

Sampling \mathbf{z} from the learned posterior knowing \mathbf{x} , $\mathbf{z} \sim q_\phi(\cdot|\mathbf{x})$, can be seen as encoding \mathbf{x} into \mathbf{z} , while $p_\theta(\cdot|\mathbf{z})$ seeks to reconstruct \mathbf{x} from \mathbf{z} . Putting a lot of information about \mathbf{x} in \mathbf{z} makes reconstruction trivial, but is heavily penalised by the regularization term. Therefore, the regularization term acts as an information bottleneck, analogous to the dimension bottleneck in deterministic auto-encoders, so a balance between both terms must be found.

An other insightful form of the ELBO can be obtained. Using Bayes' rule, $p(\mathbf{z}) = \frac{p(\mathbf{x})p(\mathbf{z}|\mathbf{x})}{p(\mathbf{x}|\mathbf{z})}$ and injecting it in Equation 2.19:

$$\mathcal{L}_{\theta,\phi}^{elbo}(\mathbf{x}) = \mathbb{E}_{q_\phi} \ln p_\theta(\mathbf{x}|\mathbf{z}) - \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \left(\frac{q_\phi(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x}|\mathbf{z})}{p(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x})} \right) d\mathbf{z} \quad (2.21)$$

$$= \mathbb{E}_{q_\phi} \frac{\ln p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{x})}{\ln p_\theta(\mathbf{x}|\mathbf{z})} - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \quad (2.22)$$

$$= \ln p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \quad (2.23)$$

The Kullback-Leibler divergence, D_{KL} is non-negative. Therefore, the second form in Equation 2.23 immediately shows that the ELBO is a *lower bound* on the log-likelihood. It is also clearly tight if and only if $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$. Intuitively, if the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ perfectly matches the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$, there is no 'approximation cost' caused by improper selection of \mathbf{z} . This form shows that maximizing $\mathcal{L}_{\theta,\phi}^{elbo}(\mathbf{x})$ can be done by either increasing $p_\theta(\mathbf{x})$ (training the generator) or by making the bound tighter (training the inference network).

Important remarks. (i) The second form, Equation 2.23 can not be used to train in practice as it requires computation of $p_\theta(\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$, and both quantities are intractable. In contrast, Equation 2.19 requires evaluating two deep feed-forward networks, $q_\phi(\cdot|\mathbf{x})$ and $p_\theta(\cdot|\mathbf{z})$, which is both tractable and efficient.

(ii) When looking at Equation 2.19, it is tempting to think that setting $D_{KL}(q_\phi(\mathbf{x}|\mathbf{z})||p(\mathbf{z}))$ to 0 should improve the bound. That is *not* the case, unlike for the KL divergence in Equation 2.23. Indeed, it would mean that $q_\phi(\mathbf{x}|\mathbf{z}) = p(\mathbf{z})$, so \mathbf{z} becomes independent of \mathbf{x} and the model does not use the encoder at all. This will strongly degrade the reconstruction term, $\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{x}|\mathbf{z})} [\ln(p_\theta(\mathbf{x}|\mathbf{z}))]$. The optimal setting is $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$; this incurs a non-zero $D_{KL}(q_\phi(\mathbf{x}|\mathbf{z})||p(\mathbf{z}))$, but in this cost is exactly compensated by

the improvement of the reconstruction term due to sampling from $q_\phi(z|x)$ rather than $p_\theta(z)$. (iii) In practice the optimal value of $D_{KL}(q_\phi(x|z)||p(z))$ in Equation 2.19 is a balance between its two terms: the network will keep putting more information into z as long as the gain in reconstruction is greater than the cost of that information. Thus, the optimal amount of information depends on the quality of both the encoder and the decoder and cannot be predicted in general. In particular, a model can have poorer reconstructions but better likelihood performance if it uses very little information to reconstruct x .

2.6 Deep invertible transformations

We have seen how to lower-bound the integral $p_\theta(x) = \int_z p(x|z)p(z)dz$. An other approach is to restrain the class of functions \mathcal{F}_Θ that can be used to build p_θ in a way that the integral is easy to compute. To see which properties can be desirable from such a class of functions, we recall the difficulties that arise when training a VAE. First, it requires an approximate posterior $q_\phi(\cdot|x)$ that matches the intractable true posterior $p_\theta(\cdot|x)$. Mistakes made by q_ϕ have a cost: Equation 2.23, is tight if and only if $q_\phi(z|x)$ matches $p_\theta(z|x)$ exactly. Thus, *exact inference* is a desirable property.

An other difficulty is that $\mathbb{E}_{z \sim q_\phi(z|x)}$ remains continuous and high dimensional despite the use q_ϕ . This requires Monte Carlo approximations which trade-off variance of the estimator against computation efficiency. If the posterior density were to be a Dirac, the integral could be computed exactly with a single sample. Thus, *deterministic inference* is a desirable property. To summarize, we wish to have a function f_θ such that $q_\phi(z|x) = p_\theta(z|x) = \delta(z - f_\theta(x))$.

This analysis points to the class of *invertible* functions. Indeed, an invertible generative model offers *exact inference* obtained by computing it's inverse. It also maps a single input to a single output so it offers *deterministic inference*. In the context of natural images, Dinh et al. [2017] proposed 'Non Volume Preserving' transformations (NVP). A latent space of the same dimensionality as the data space is specified, with a simple prior (e.g. unit Gaussian $\mathcal{N}(\mathbb{O}_n, \mathbb{I}_n)$). The simple distribution is then progressively reshaped into a more complex one by successive invertible transformations, $x = f_\theta(z) = f_{\theta_n} \circ f_{\theta_{n-1}} \circ \dots \circ f_{\theta_1}(z)$. Denote Z the random variable obtained by mapping X through f_θ , the density of natural images in the training set under the reshaped distribution can be estimated using the change of variable formula:

$$p_X(x) = p_Z(f_\theta(x)) \times \left| \det \left(\frac{\partial f_\theta(x)}{\partial x^\top} \right) \right|. \quad (2.24)$$

which provides the training loss of the model. This formula follows from the fact that the probability mass contained by a differential area does not depend on our arbitrary parametric choices, i.e.: $|p_Z(z)dz| = |p_X(x)dx|$. To generate new images, latent variable vectors can be samples from $p_Z(z)$, and mapped to image space with the inverse transformation f_θ^{-1} . Figure 2.5 illustrates the mapping from image space to latent space.

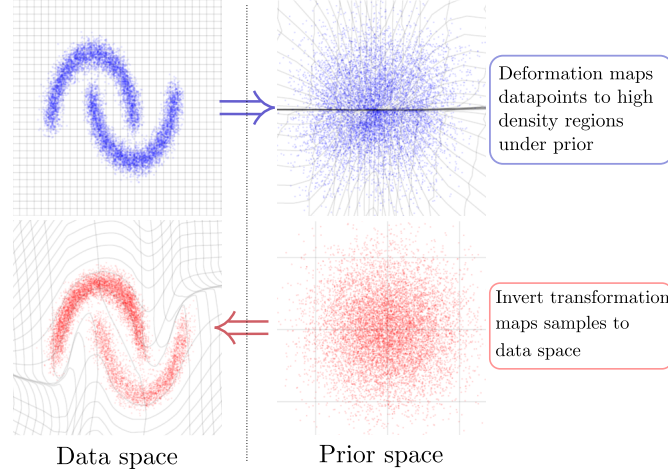


Figure 2.5: At inference, f_θ is used to smoothly and invertibly transform $p_X(x)$ into an isotropic Gaussian $p_Z(z)$. At sampling time, f_θ^{-1} is used for the reverse, thus generating image samples from a simple distribution over latent variables. Notice how the coordinates are transformed in each cases, as visualized by the grids: to displace the points in the desired manner, the model distorts the space. Figure adapted from Dinh et al. [2017].

Practical construction: affine coupling layers. In practice, the class of invertible functions used must be restricted to ensure efficient computations of the quantities involved in Equation 2.24: $y = f(x)$ as well as the determinant $\left| \det \left(\frac{\partial f_\theta(x)}{\partial x} \right) \right|$. One way to ensure tractable and easy to invert blocks f_{θ_i} is to partition the dimensions of the variables in two groups, by applying a permutation σ_i to x and splitting $\sigma_i(x)$ in two such that $\sigma_i(x) = (\sigma_i(x)^1, \sigma_i(x)^2)$. To simplify notations, assume the permutation $x \leftarrow \sigma_i(x)$ has been performed and write x again. One group is kept unchanged, but is used to transform the other group via translation and scaling, as illustrated in Figure 2.6. Then $f_{\theta_i}(x) = (y^1, y^2)$ where:

$$y^1 = x^1 \quad (2.25)$$

$$y^2 = t(x^1) + x^2 \odot \exp(s(x^1)) \quad (2.26)$$

This transformation is trivial to invert:

$$x^1 = y^1 \quad (2.27)$$

$$x^2 = (y^2 - t(x^1)) \odot \exp(-s(x^1)) \quad (2.28)$$

The functions $s(\cdot)$ and $t(\cdot)$ can be arbitrarily flexible, and implemented as complex non-invertible functions, e.g. deep CNNs. There is no need to invert them when computing $f_{\theta_i}^{-1}$ as shown in Equation 2.28.

This construction yields blocks f_{θ_i} that are easy to invert. Each block is not very expressive, being limited to an affine transformation of half the variables. To make f_θ more flexible, many

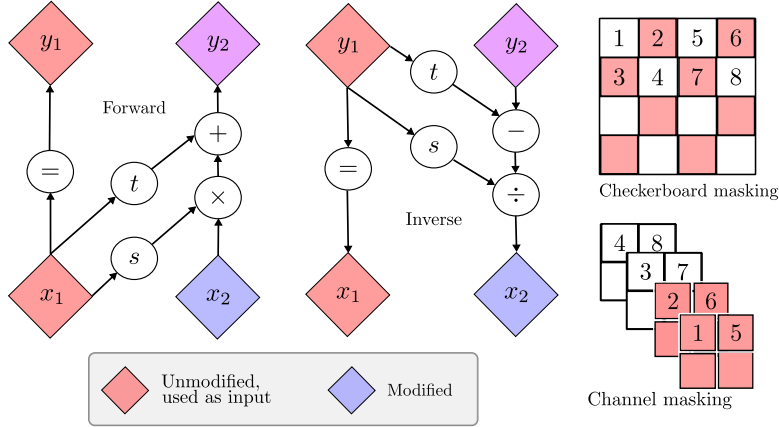


Figure 2.6: During forward propagation, the scalar variables are separated into two groups using masking schemes. The first group x_1 (in red) is used as input to expressive functions s and t that yield translation and scaling parameters for the second group (in blue), yielding a results that depends on both groups (in purple). To invert the function, it is not necessary to invert s and t (as depicted in the centre). To avoid always modifying the same subset of the scalar variables, checkerboard partitioning patterns are used together with partitioning along the channel axis (depicted on the right). Figure adapted from Dinh et al. [2017]

such blocks are composed, and their log-determinants are summed. The permutation σ_i is here to avoid always modifying the same features: because of it, the partitioning scheme changes from one block to the next. To do this, Dinh et al. [2017] proposes two variable partitioning schemes, illustrated in Figure 2.6. One uses a checkerboard mask, which is very natural: looking at one pixel out of two gives a good idea of what the image looks like. The second uses channel-wise masking and the two masks are used alternatively. This construction also offers efficient computations of the log-determinant of the Jacobians. Indeed the Jacobian has a triangular structure:

$$\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}^\top} = \begin{bmatrix} \mathbb{I}_d & \mathbb{O}_d \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1^\top} & \text{diag}(\exp(s(\mathbf{x}_1))) \end{bmatrix}. \quad (2.29)$$

Therefore the determinant is given by the product of the diagonal terms of the Jacobian, $\ln \det \left(\frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}^\top} \right) = \mathbf{1}^\top s(\mathbf{x}_1)$. The log-likelihood of the model is thus also easy and tractable to compute, and can be optimized by stochastic gradient descent on the negative log-likelihood:

$$-\ln p_X(\mathbf{x}) = -\ln p_Z(f_\theta(\mathbf{x})) - \mathbf{1}^\top s(\mathbf{x}_1). \quad (2.30)$$

2.7 Generative adversarial networks

We have presented two classes of deep latent variable generative models built to have tractable (approximation of) $p_\theta(\mathbf{x})$. Another approach is to train the model without maximizing $p_\theta(\mathbf{x})$. Indeed, while inference can be challenging in latent variable models, computing $f_\theta(\mathbf{z})$ is simple

and efficient. Thus given an image quality metric, the model can be trained by assigning scores to samples \tilde{x} taken from the model. Generative Adversarial Networks (GANs), introduced by Goodfellow et al. [2014], propose to use a classifier D_ϕ to evaluate samples. A latent variable vector z is sampled from a prior $p(z)$, and mapped to image space using a deep network: $x = G_\theta(z)$, inducing an implicit density $p_\theta(x)$. A second deep network, termed the ‘discriminator’ is used to evaluate the quality of sampled images by outputting $D_\phi(x) \in [0, 1]$, the estimated probability of x being real. Intuitively, if D_ϕ is well trained, it is able to identify real looking images, and its score can be used as a reward. The objective of the discriminator is a function of the parameters of the generator θ and those of the discriminator, ϕ . It is the expected log-likelihood of correct classification:

$$V(\phi, \theta) = \mathbb{E}_{x \sim p^*(x)} [\ln D_\phi(x)] + \mathbb{E}_{x \sim p_\theta(x)} [\ln (1 - D_\phi(G_\theta(z)))]. \quad (2.31)$$

The discriminator is trained to maximize classification accuracy for a given generator, i.e. ϕ is optimized to reach $\phi^* = \max_\phi V(\phi, \theta)$. Simultaneously, the generator is trained to degrade the classification of a given discriminator, i.e. θ is optimized to reach $\theta^* = \min_\theta V(\phi, \theta)$. Solving this adversarial problem corresponds to finding ϕ^* and θ^* such that:

$$V(\phi^*, \theta^*) = \min_\theta \max_\phi V(\phi, \theta). \quad (2.32)$$

The discriminator can be seen as a *trainable loss function* that can focus on the mistakes typically made by the generator. Assume a fixed generator G_θ , the optimal discriminator maximizes

$$V_\theta(\phi) = \int_x [p^*(x) \ln D_\phi(x) + p_\theta(x) \ln(1 - D_\phi(x))] dx. \quad (2.33)$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$ the function $y \mapsto a \ln(y) + b \ln(1 - y)$ achieves its maximum in $[0, 1]$ at $y = a/(a + b)$ so for a fixed θ , the optimal discriminator $D_{\phi^*(\theta)}$ is the Bayes classifier: $D_{\phi^*(\theta)}(x) = \frac{p^*(x)}{p^*(x) + p_\theta(x)}$. Now assuming that D_ϕ is trained to optimality for a given θ , $D_{\phi^*(\theta)}$ can be plugged in $V(\theta, \phi)$:

$$V(\phi^*(\theta), \theta) + \ln 4 = D_{KL} \left(p^* \parallel \frac{p^* + p_\theta}{2} \right) + D_{KL} \left(p_\theta \parallel \frac{p^* + p_\theta}{2} \right) \propto D_{JS}(p^* \parallel p_\theta). \quad (2.34)$$

Assume the regime of infinite data, infinite model capacity, and under the assumption that the optimal discriminator is reached at each training iteration of the generator. In that context, Equation 2.34 shows, by convexity of D_{KL} ⁵, that there is a unique global optimum for G_θ , at the data distribution $p_\theta = p^*$, which can be recovered by gradient descent. In practice however, the ideal assumptions made above are not met. To optimize the networks, the expectation is replaced with sample average over mini-batches, and parallel stochastic gradient descent is used on ϕ and θ .

GANs are known to be difficult to train, for several reasons. The mini-max objective formulation between two networks can lead to oscillations between solutions, because the optimality assumption is not met. It is also necessary to pick ‘compatible’ generator and discriminator architectures,

⁵ with respect to p_θ , not to θ .

as training is known to fail if the discriminator is too powerful, as explained in [Arjovsky et al. \[2017\]](#). The GAN training objective also elicit models that produce good quality images but fail to capture the full support of the training data, a phenomenon known as mode-collapse and detailed in Section 2.9.2. If training is successful, however, GANs can produce outstanding samples.

2.8 Autoregressive density estimation

We have presented three approaches to train deep latent-variable generative models. A fourth important class of model is obtained by dropping the conditional independence requirement. In that case, latent variables are no longer needed to ensure global coherence of the image, as explained in Section 2.2. Instead, one can rely on the standard chain rule to factorize $p_\theta(x)$ into a joint probability distribution:

$$p(\mathbf{x}_{1:D}) = p(x_1) \prod_{i=2}^D p(x_i | \mathbf{x}_{<i}), \quad (2.35)$$

with $\mathbf{x}_{<i} = (x_1, \dots, x_{i-1})$. Latent variables are no longer needed to ensure global coherence in this case: a given pixel x_i is modelled after seeing $\mathbf{x}_{j<i}$, so it can adapt to previous pixels to be coherent with them. Then, following pixels $\mathbf{x}_{>i}$ are modelled conditioned on x_i , so they can adapt to x_i , and all pixels are coherent.

In the regime of deep auto-regressive models, the idea is to use (deep) neural networks to model the dependencies in this chain, $p(x_i | \mathbf{x}_{<i})$. To implement this with a CNN-like architecture, masked convolutions can be used. The idea, illustrated in Figure 2.7, is to compute for each scalar variable x_i , features of increasing complexity f_i that will be used to parametrise $p(x_i | \mathbf{x}_{j<i})$. Thus, at a given layer l , a feature f_i^l can only depend on $\mathbf{x}_{j<i}$. This can be built recursively like so: at layer 1, f_i^1 is computed with a masked kernel that only looks at $\mathbf{x}_{j<i}$ as in Figure 2.7, top left. Given the feature at a layer l , $f_{i \leq N}^l$, a new feature f_i^{l+1} can be computed with a masked kernel convolution as in Figure 2.7 (center top), because these features only depend on $\mathbf{x}_{<i}$. Notice the central feature is available except at the first layer. This model, introduced in [Oord et al. \[2016\]](#) is called Pixel-CNN.

This yields tractable exact likelihood computations and does not require complex integral over latent variables. The main drawback of this method is that it suffers from slow sequential sampling. An other drawback is that the fact that the model cannot rely on latent variables to couple pixels makes it harder for it to model global structure, especially given the fact that stacked convolutions already devote more flexibility to modelling local structure. Finally, the absence of latent variables makes it unsuitable for representation learning.

2.9 Challenges in generative modelling

We have discussed the main existing families of deep generative models used in the literature. In this section, we present in greater detail two challenges in generative modelling that will be

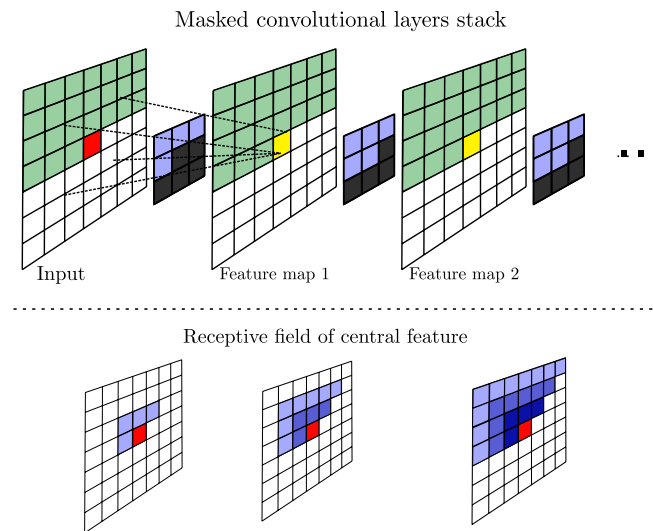


Figure 2.7: To predict the red pixel in the first map (top left), the model is allowed to look at the green pixels. When computing the second feature map (top centre) the yellow feature has been computed without looking at the red pixel, and therefore becomes observable. This explains why the first convolutional filter is different from the following ones. The second half of the figure shows how the receptive field of the central feature evolves through the successive layers.

central in the rest of this dissertation. The first is elaborated upon in Chapter 3 and the second in Chapter 4. The two issues, which can intertwine, are tackled together in Chapter 5. Some key related work also addressing these challenges is presented in Section 2.10 and Section 2.11.

2.9.1 Understanding the conditional independence assumption

In this section, we will take a closer look at the theoretical and practical implications of the conditional independence assumption, which we presented in Section 2.2. Tackling the shortcomings of this assumption is key to the work presented in Chapter 3 and Chapter 5. We present several complementary points of view. As we have seen the standard way of defining latent generative models is to make a conditional independence assumption, which yields a density of the form:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^N p_\theta(x_i|\mathbf{z}). \quad (2.36)$$

Recall that in Section 2.2, we have seen that mapping a latent variable z from some latent space \mathcal{Z} to an image space \mathcal{X} using a non-linear function f_θ yields an implicit density $\rho_\theta(\mathbf{x}) = P(f_\theta(\mathbf{z}) = \mathbf{x})$. The support of this density belongs to a low dimensional manifold of the ambient space, as \mathbf{z} is of low dimensionality and f_θ is deterministic. Therefore ρ_θ has a degenerate support in the sense that it (almost surely) puts 0 mass on the observed dataset:

$$P(\{\exists \mathbf{x} \in D | \rho_\theta(\mathbf{x}) = 0\}) = 1. \quad (2.37)$$

To measure a density and train our model, it is thus necessary to add volume around this low-dimensional manifold. This is typically done by using f_θ as the mean of a parametric distribution and adding isotropic noise around it, for instance using a Gaussian density with isotropic variance:

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} \mathcal{N}(\mathbf{x} | f_\theta(\mathbf{z}), \sigma \mathbb{I}_n) p(\mathbf{z}) d\mathbf{z} = \int_{\mathbf{z}} \prod_{i=1}^N \mathcal{N}(x_i | f_\theta(\mathbf{z})_i, \sigma) d\mathbf{z}. \quad (2.38)$$

Because of the isotropic nature of this volume, it *cannot be used to model any structure*, and all the structure has to be captured by $f_\theta(\mathbf{z})$. In what follows, ρ_θ denotes a density with degenerate support, and p_θ denotes the density obtained by adding isotropic volume to ρ_θ , which thus has a non-degenerate support. As f_θ becomes more flexible and accurate, less volume is needed, and σ decreases. At the optimal limit and with infinite data ρ_θ fits p^* perfectly, and $\sigma \rightarrow 0$ such that:

$$p_\theta(\mathbf{x}|\mathbf{z}) \rightarrow \delta(\mathbf{x} - f_\theta(\mathbf{z})) = \rho_\theta(\mathbf{x}|\mathbf{z}) \text{ and } \rho_\theta(\mathbf{x}) = p^*(\mathbf{x}). \quad (2.39)$$

In practice $f_\theta(\mathbf{z})$ is not flexible enough, so volume will always be necessary to model what f_θ cannot capture.

Limitations of isotropic noise. To take a look at the practical implications, let us first clarify what we mean by «[the volume] cannot be used to model any structure». Suppose the mean of the distribution $f_\theta(\mathbf{z})$ is a good image, then sampling an image from $p_\theta^*(\mathbf{x}|\mathbf{z})$ means adding 'salt and pepper' per-pixel noise around $f_\theta(\mathbf{z})$, which leads to a poor noisy image. The best we can hope

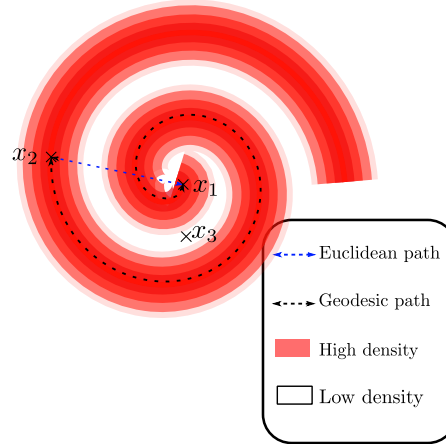


Figure 2.8: Interpolating from point x_1 to point x_2 using the blue arrow requires going through a region of low density, which illustrates that the Euclidean distance is a bad notion of distance for points on a non-linear manifold. Instead, the geodesic distance (black arrow) should be used. Importantly, x_3 is in a region of low density and should thus be "far" from any point in the red area. That is not the case when using the Euclidean distance.

for is that σ is small enough that we can't notice the difference. In terms of density modelling performance, we are *wasting* the probability mass of $p_\theta(\mathbf{x})$ on unlikely points around $f_\theta(\mathbf{x})$, i.e we are *over-generalizing* and taking samples from $\mathbf{x} \sim p_\theta(\mathbf{x})$ will yield unrealistic images. This is particularly problematic if f_θ lacks "a lot" of flexibility, as a lot of mass has to be wasted to compensate. This illustrates that the conditional independence assumption is a poor assumption for $p_\theta(\mathbf{x})$.

Considering these limitations, which concern $p_\theta(\mathbf{x})$, it is reasonable to see ρ_θ as the real quantity of interest, as is the case in GANs. In this view, the noise σ is just here to train ρ_θ and can be considered as belonging to the training loss rather than to the model. This point of view is reasonable in the sense that it is consistent: the loss is zero for $\rho_\theta(\mathbf{x}) = p^*$, and ρ_θ is a universal density approximator. However this view is also problematic: we care about one density, ρ_θ , but *optimize and evaluate* another density. It is therefore reasonable to ask: *what will the impact be on the training of ρ_θ ?*

Impact on the training of $\rho_\theta(\mathbf{x})$. The point of view that ρ_θ is the object of interest, and σ is just "here to help", nevertheless has implications on the training loss of ρ_θ . We can see it in terms of "reconstruction losses": maximizing the log-likelihood of an isotropic Gaussian with fixed variance corresponds to minimizing an L_2 reconstruction loss,

$$-\log(p_\theta(\mathbf{x}|\mathbf{z})) \propto \|\mathbf{x} - f_\theta(\mathbf{z})\|_2. \quad (2.40)$$

Therefore, to measure the distance between an image and a prediction, the *Euclidean norm* in pixel space is used. It is sensible in that it is a distance, so it is minimised for $\mathbf{x} = f_\theta(\mathbf{z})$.

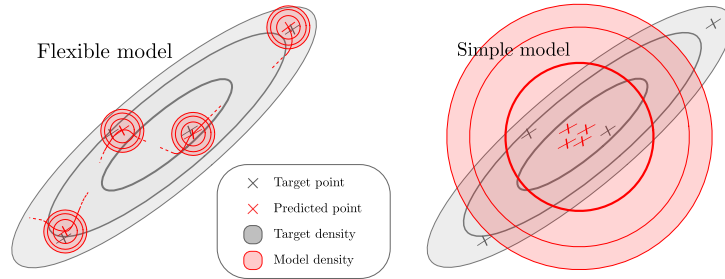


Figure 2.9: If a model is assumed to have isotropic variance, two situations are possible. i) The model is flexible enough to fit all points, and will reduce the variance almost to a Dirac, which can lead to a model that is significantly more complex than it should be, and over-fitting. Indeed in the example on the left, the ground truth has one non-isotropic mode, while the model is significantly more complex. ii) The model is not flexible enough to fit all points closely, and has to substantially increase its variance, thus wasting probability mass and yielding unrealistic samples.

However, it is clear that the Euclidean norm is not a good measure of similarity between images. For instance, take \mathbf{x} and translate it by two pixels to obtain $\tilde{\mathbf{x}}$. Most humans will state, reasonably, that the two images are *almost identical* and yet, the Euclidean norm *will be huge*. The same goes for rotations and other low level transformations of \mathbf{x} . Conversely, two images at a small L_2 distance of each other can seem visually very different.

What would a better distance look like? A reasonable assumption is that images lie (close to) a low-dimensional, non-linear, manifold of \mathbb{R}^N . A desirable distance could be the geodesic distance⁶ between the projection of \mathbf{x} and $\tilde{\mathbf{x}}$ on that manifold, $d_g(\text{proj}(\mathbf{x}), \text{proj}(\tilde{\mathbf{x}}))$. While following the geodesic to minimize d_g , $\text{proj}(\tilde{\mathbf{x}})$ would move on the manifold, so its pixels would change in a coherent manner, and so d_g would better satisfy human judgement. This is illustrated in Figure 2.8. Unfortunately this manifold is precisely what we are trying to learn, so it is impossible to leverage the geodesic distance. However, it highlights the shortcomings of the Euclidean norm.

Pathological example. What happens when using the Euclidean distance anyway? Suppose a subset $\mathcal{T} \subset \mathcal{X}$ in which p^* is approximately unimodal, and take two samples $\mathbf{x}^1, \mathbf{x}^2 \sim p^*(\mathbf{x} | \mathbf{x} \in \mathcal{T})$. If $p_\theta(\mathbf{x} | \mathbf{z})$ is assumed isotropic, two cases may happen: either f_θ is flexible enough to go close to both points, and the simple unimodal distribution is replaced by a more complicated distribution with two modes (and the model overfits), or f_θ is not flexible enough and it will fit both points with a single, blurry mode between the two points and high variance around it, as illustrated in Figure 2.9.

⁶ Assuming a smooth manifold structure – a topology and a smooth atlas are given – and a metric, the geodesic between two points is the path that has the shortest length while staying on the manifold.

Going beyond the conditional independence assumption. Given the discussion above, it appears that building models that go beyond the conditional independence assumption is an interesting research direction. It can be seen as building better density models for p_θ , or equivalently better training losses for ρ_θ . In Chapter 3, we construct such a model by using autoregressive decoders, presented in Section 2.8, in a VAE model. The decoder is conditioned on some latent variable z produced by an inference network, and can be written:

$$p_\theta(\mathbf{x}|z) = \prod_{i=1}^N p_\theta(x_i | \mathbf{x}_{j < i}, z). \quad (2.41)$$

With this construction, the pixels are sampled sequentially, which is slower at sample time, but an arbitrarily complex dependencies can be learned around $f_\theta(z)$. In Chapter 5, a different approach is taken. An invertible model f_ψ is used to build an abstract feature space in which the Euclidean distance is a better measure of similarity. In other words we lift the targets of p_θ in a feature space:

$$p_\theta(f_\psi(\mathbf{x})|z) = \prod_{i=1}^N p_\theta(f_\psi^i(\mathbf{x})|z) \quad (2.42)$$

and use the invertibility of f_ψ to go back to image space, both for likelihood evaluations and for sampling.

2.9.2 Understanding mode-dropping in adversarial networks

Adversarial networks have the ability to generate very compelling samples, far beyond the sample quality of a VAE with an identical computational budget. On the other hand, a known issue is that they tend to model only a strict subset of the support of the true data distribution p^* , which is called mode-dropping. Intuitively, this is a direct consequence of the way the model is trained as we show in the following paragraphs. This issue is made worse by the fact that GANs do not provide a clear way of measuring this phenomenon. Indeed there is no simple way of measuring the density of a given real data point \mathbf{x} under the model, $p_\theta(\mathbf{x})$, because the model has a degenerate support.

There are two types of failure cases for a GAN generator. The first is to produce poor looking images, i.e. put mass outside of the support of p^* . The second is to produce good quality images with too little variability, i.e. miss some modes of the support and put too much mass on others. We now discuss the way the discriminator penalises these two undesirable behaviours. To train p_θ , a sample $\tilde{\mathbf{x}}$ is taken from the model, and it is fed to the discriminator which has to distinguish it from real images $\mathbf{x} \sim p^*(\mathbf{x})$, by computing $D_\phi(\tilde{\mathbf{x}})$. If a sample is of poor visual quality, the discriminator can reject it based on that. In other words it has an approximation of p^* , denote it $\widehat{p^*}$, and can estimate if " $p^*(\mathbf{x})$ is high". Thus it is clear that the discriminator can *explicitly* evaluate the *quality* of the image. What *does not* happen when training a GAN, is taking an image from the training set, $\mathbf{x} \sim p^*$, and explicitly asking if it is well covered by the model p_θ , i.e.

"is $p_\theta(\mathbf{x})$ high ?". This is what MLE, for instance does. However if only $p^*(\mathbf{x})$ was evaluated, p_θ would always collapse to the mode of p^* , which does not usually happen. Therefore some mechanism also evaluates variability.

Implicit variability evaluation. Let us discuss the extreme case where the learned density collapses to a single image. Suppose that the generator learns a Dirac on a *single* real looking image \mathbf{x}_0 from the training set:

$$p_\theta(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_0). \quad (2.43)$$

The discriminator can not reject the image based on its quality, as it is a real image, i.e. evaluating $\widehat{p}^*(\mathbf{x})$ will yield a good score. Yet intuitively it can succeed by always predicting *fake* when seeing this image. Most of the time it will be correct, and very rarely the image will come from the dataset and it will be wrong. This is because the discriminator also holds an approximation of p_θ , that we denote \widehat{p}_θ . If an image is "seen too often", it's score is degraded, and the discriminator determines "too often" by comparing $\widehat{p}_\theta(\mathbf{x})$ and $\widehat{p}^*(\mathbf{x})$. If $p_\theta(\tilde{\mathbf{x}})$ is "too high" and the discriminator pushes it down, the mass must go somewhere. Going outside of the support is heavily penalised, so it should go on the support. Thus, by *explicitly* evaluating the mass of *samples* $\tilde{\mathbf{x}}$, the density mass is *implicitly* pushed towards the training set, due to the fact that the total mass is fixed. This shows an informal 'consistency' of adversarial training.

Practical consequences. We have seen that adversarial training *explicitly* rewards quality, and *implicitly* rewards coverage. In practice the consequence is that training is biased towards a *mode-seeking* behaviour. To see this, assume that p^* is more flexible than p_θ . To avoid going into low density regions of the space, p_θ has to drop some modes of the distribution. Conversely, when training with MLE, p_θ has to go through regions of low density to cover the full support of the distribution. This phenomenon is illustrated in Figure 2.10. From this perspective, the fact that GANs tend to produce good-looking samples but drop part of the support is not surprising. To build conditional models that produce compelling outputs, used for instance in photo editing products, this may be fine. But from a density modelling perspective, coverage and quality are two sides of the same coin: the goal is to learn p^* . This motivates the goal of reducing mode collapse in GANs.

Reducing mode-collapse in GANs. The discussion above motivates training procedures that fight mode-collapse by more closely evaluating support coverage. In Chapter 4, a construction to explicitly evaluate variability is proposed. This idea is as follows: variability is hard to evaluate from i.i.d samples, because it is evaluated implicitly. Instead, one can consider *batches* of images, fake and real, as input to the discriminator. To fool the discriminator, it is necessary to produce a batch with *as much variability* as a batch of fake images. Thus, variability becomes a feature that can be explicitly computed as a batch statistic, rather than implicitly evaluated. In Chapter 5 a more direct approach is taken, based on the symmetrical properties of adversarial training and MLE: by combining the two, one is able to explicitly optimize both coverage and quality.

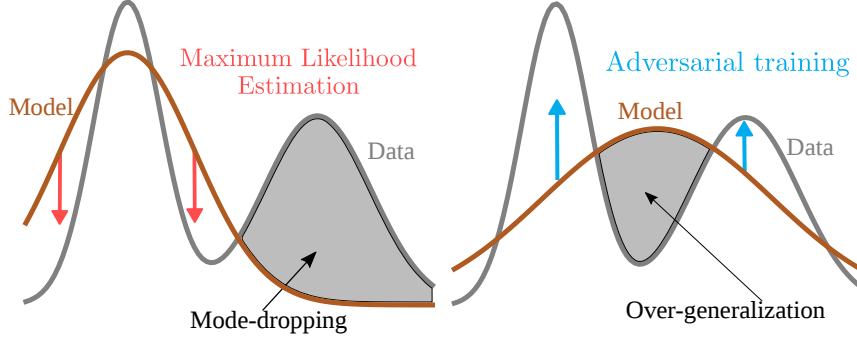


Figure 2.10: With adversarial training (left), a sample x is taken from the model, which is penalised if x has low density under the data distribution. Thus, adversarial training pushes the model to avoid low density regions, and p_θ has to drop modes of p^* if it is not flexible enough. With MLE, the reverse happens: the model is penalised for not assigning mass to samples from the data.

2.10 Details and refinements on Variational auto-encoders

In sections 2.5 and 2.7, the core of the VAE and GAN models have been presented. These two models are at the center of this thesis, thus we now present them more thoroughly. We first discuss additional details on the practical training of VAEs in Section 2.10.1. In sections 2.10.2 to 2.10.4, refinements of the VAE framework that aim at improving the accuracy of the approximate posterior are discussed. In Section 3.3.2, we present the idea of using flexible autoregressive decoders. Finally, a variant of the VAE using quantized latent vectors is presented in Section 2.10.5.

2.10.1 Practical training algorithm

In Section 2.5 we presented the basics of the VAE model. Here we present technicalities and refinements that are necessary to make it work in practice. Recall the key VAE objective function:

$$\mathcal{L}_{\theta, \phi}^{elbo}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\ln(p_\theta(\mathbf{x}|\mathbf{z}))] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \quad (2.44)$$

In practice, parametric families remain to be specified for $p_\theta(\mathbf{z})$, $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$. One usual choice is to assume Gaussianity for $p(\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$, in which case the D_{KL} term can be computed in closed form: with $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$ and $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x}))$,

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2} \left[1 + \ln g_\phi^\sigma(\mathbf{x}) - g_\phi^\mu(\mathbf{x}) - g_\phi^\sigma(\mathbf{x}) \right]. \quad (2.45)$$

For the reconstruction term, empirical estimation is typically used: with $\mathbf{z}_s \sim q_\phi(\mathbf{z}|\mathbf{x})$, $\mathbb{E}_{q_\phi} \ln p_\theta(\mathbf{x}|\mathbf{z}) \approx \frac{1}{S} \sum_{s=1}^S \ln p_\theta(\mathbf{x}|\mathbf{z}_s)$. This empirical estimation raises an issue: the estimator becomes non-differentiable because of the sampling operator. A first solution can be to use the *Reinforce*

algorithm: under assumptions compatible with the Leibniz Integral rule,

$$\nabla_{\phi} \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} [\ln p_{\theta}(\mathbf{x}|z)] = \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} [\ln p_{\theta}(\mathbf{x}|z) \nabla_{\phi} \log q_{\phi}(z|\mathbf{x})] \quad (2.46)$$

$$\approx \frac{1}{N} \sum_{i=1}^N \ln p_{\theta}(\mathbf{x}|z_i) \nabla_{\phi} \log q_{\phi}(z_i|\mathbf{x}), \quad (2.47)$$

where the first line comes from swapping \mathbb{E} and ∇_{ϕ} , and from the log-derivative trick: $\nabla_{\phi} q_{\phi}(z) = q_{\phi}(z) \nabla_{\phi} \log q_{\phi}(z)$. Unfortunately, this approach suffers from high variance. Intuitively, it is in fact too generic: it enables differentiation w.r.t. to *any* sampling operation. We are in a special case where we do need to sample, which is inherently non-differentiable, but from a parametric distribution that is differentiable w.r.t. its parameters. This leads to the idea of treating the randomness as an additional input of the network, sampled from a standard Gaussian, and learning a transformation that reshapes this standard Gaussian into an other distribution. This approach is called the *re-parametrization trick* [Kingma and Welling \[2014a\]](#) and is illustrated in Figure 2.11. With $\varepsilon \sim q(\varepsilon)$ and $z = g_{\phi}(\varepsilon|\mathbf{x})$:

$$\nabla_{\phi} \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} [\ln p_{\theta}(\mathbf{x}|z)] = \nabla_{\phi} \mathbb{E}_{\varepsilon \sim q(\varepsilon)} [\ln(p_{\theta}(\mathbf{x}|g_{\phi}(\varepsilon|\mathbf{x})))] \quad (2.48)$$

$$= \mathbb{E}_{\varepsilon \sim q(\varepsilon)} [\nabla_{\phi} \log(p_{\theta}(\mathbf{x}|g_{\phi}(\varepsilon|\mathbf{x})))] \quad (2.49)$$

$$\approx \frac{1}{N} \sum_{i=1}^N \nabla_{\phi} \ln p_{\theta}(\mathbf{x}|g_{\phi}(\varepsilon_i|\mathbf{x})). \quad (2.50)$$

Both methods are unbiased, but this approach offers gradients with lower variance. In practice, $g_{\phi}(z|\mathbf{x})$ is chosen Gaussian still to enable closed form computation of the KL: $z_s \sim q_{\phi}(z|\mathbf{x}) = \mathcal{N}(z; g_{\phi}^{\mu}(\mathbf{x}), g_{\phi}^{\sigma}(\mathbf{x}))$. With the re-parametrization trick, this is written as: $z_s = g_{\phi}^{\mu}(\mathbf{x}) + g_{\phi}^{\sigma}(\mathbf{x}) \odot \epsilon_s$, with $\epsilon_s \sim \mathcal{N}(\epsilon_s; 0, I)$.

In this set-up, the empirical estimator of the ELBO reduces to:

$$\mathcal{L}_{\theta, \phi}(\mathbf{x}) \approx \frac{1}{S} \sum_{s=1}^S \ln p_{\theta}(\mathbf{x}|g_{\phi}^{\mu}(\mathbf{x}) + g_{\phi}^{\sigma}(\mathbf{x}) \odot \epsilon_s) - \frac{1}{2} \left[1 + \ln g_{\phi}^{\sigma}(\mathbf{x}) - g_{\phi}^{\mu}(\mathbf{x}) - g_{\phi}^{\sigma}(\mathbf{x}) \right] \quad (2.51)$$

2.10.2 Top down sampling

When optimizing $\mathcal{L}_{\theta, \phi}^{elbo}(\mathbf{x}) = \mathbb{E}_{q_{\phi}(z|\mathbf{x})} [\ln(p_{\theta}(\mathbf{x}|z))] - D_{KL}(q_{\phi}(z|\mathbf{x})||p(z))$, the flexibility of $q_{\phi}(z|\mathbf{x})$ is crucial: approximating $p_{\theta}(z|\mathbf{x})$ is a difficult task. One limitation of the vanilla VAE approximate posterior is the Gaussianity restriction with diagonal covariance, commonly referred to as the mean-field assumption. More precisely, equation Equation 2.23 shows that the optimal variational distribution verifies $q_{\phi}(z|\mathbf{x}) = p_{\theta}(z|\mathbf{x})$. With restricted density families for $q(\cdot)$ this is not realizable. One possible solution is to use a hierarchy of latent vectors, rather than a single one. In the generative model p_{θ} , latent variables z can be split into L groups, each one at a

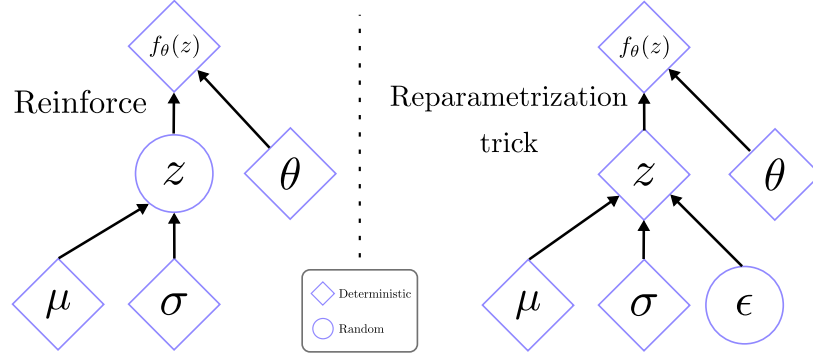


Figure 2.11: The re-parametrization trick consists in treating the "randomness" as an input to the network, denoted ϵ . This allows differentiation with respect to σ and μ , and reduces the variance of the estimator compared to sampling z directly. Quantities inside a circle are stochastic, while those inside a diamond are deterministic functions of their possibly stochastic inputs.

different layer, and the density over \mathbf{z} can be written as:

$$q(\mathbf{z}) = q(\mathbf{z}_L) \prod_{i=1}^{L-1} q(\mathbf{z}_i | \mathbf{z}_{i+1}). \quad (2.52)$$

At depth i , \mathbf{z}_{i-1} can be treated as a standard input feature map, and assume $\mathbf{z}_i | \mathbf{z}_{<i}$ to be Gaussian, as in the standard construction. Yet non-linear functions of \mathbf{z}_{i-1} can be used to implement the conditioning. Therefore when integrating over \mathbf{z} the posterior is no longer Gaussian:

$$\int_{\mathbf{z}} p_\theta(\mathbf{x} | \mathbf{z}) = \int_{\mathbf{z}_1} \dots \int_{\mathbf{z}_n} p_\theta(\mathbf{x} | \mathbf{z}) p(\mathbf{z}_n | \mathbf{z}_{j < n}) \dots p(\mathbf{z}_1) \quad (2.53)$$

with $p(\mathbf{z}_n | \mathbf{z}_{<n})$ implemented using a deep functions, for instance $p(\mathbf{z}_n | \mathbf{z}_{<n-1}) = \mathcal{N}(\cdot | \mu_{\theta_n}(\mathbf{z}_{<n}), \sigma_n I)$ with μ_{θ_n} the output of a deep network. Additionally, to allow the chain of latent variables to be sampled in the same order when encoding and when sampling, top-down sampling is used [Bachman, 2016, Kingma et al., 2016b, S nderby et al., 2016]. With top-down sampling, the encoder (symmetric to the decoder) extracts deterministic features h_i at different levels as the image is being encoded, constituting the bottom-up deterministic pass. While decoding the image, these previously extracted deterministic features h_i are used for top-down sampling and help determining the posterior over latent variables at different depths in the decoder. These posteriors are also conditioned on the latent variables sampled at lower feature resolutions, using normal densities as follows:

$$q_\phi(\mathbf{z}_1 | \mathbf{x}) = \mathcal{N}(\mathbf{z}_1 | \mu_1(\mathbf{x}, h_1), \sigma_1^2(\mathbf{x}, h_1)), \quad (2.54)$$

$$q_\phi(\mathbf{z}_i | \mathbf{z}_{i-1}) = \mathcal{N}(\mathbf{z}_i | \mu_i(\mathbf{x}, \mathbf{z}_{i-1}, h_{i-1}), \sigma_i^2(\mathbf{x}, \mathbf{z}_{i-1}, h_{i-1})). \quad (2.55)$$

This constitutes the stochastic top-down pass, and is illustrated in Figure 2.12. We refer the reader to [Bachman, 2016, Kingma et al., 2016b, S nderby et al., 2016] for more detail.

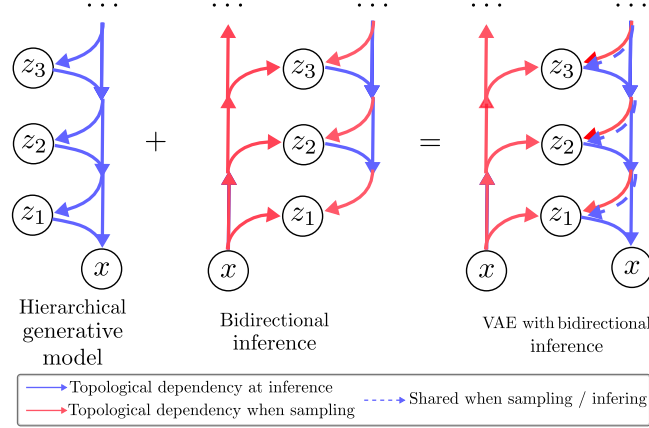


Figure 2.12: With top-down sampling, deterministic information is hierarchically extracted during the encoding. During decoding, latent variables z_i are sampled top-down, using the information extracted at corresponding layer i during encoding to determine the distribution.

2.10.3 Normalizing Flows for flexible posteriors

There are other solution to go beyond simplistic parametric families and mean-field approximations. The principle of normalizing flows (Tabak & Turner, 2013; Tabak & VandenEijnden, 2010) can be used to obtain more flexible posteriors. The key idea is to apply a sequence of invertible transformations to an initially simple probability density. Applying this sequence yields a valid probability distribution, and its density can be evaluated exactly by repeatedly applying the change of variables formula. As the initial density 'flows' through the sequence it becomes increasingly flexible. Consider an invertible, smooth mapping $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$, with inverse $f^{-1} = g$. Given a random variable z with distribution $q(z)$, we can use this mapping to transform z into $f(z)$. The change of variable formula (or inverse function theorem) yields the density of z' :

$$q(z') = q(z) \left| \det \frac{\partial f^{-1}}{\partial z'} \right| = q(z) \left| \det \frac{\partial f}{\partial z} \right|. \quad (2.56)$$

Arbitrarily complex densities can be constructed by successively applying Equation 2.56. After transforming a random variable z_0 with distribution q_0 through a chain of K invertible transformations f_k , one obtains:

$$z_K = f_K \circ \dots \circ f_2 \circ f_1(z_0) \quad (2.57)$$

and

$$\ln q_K(z_K) = \ln q_0(z_0) - \sum_{k=1}^K \ln \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|. \quad (2.58)$$

Expectations w.r.t. the transformed density q_K can be computed without explicitly knowing q_K as for any function h :

$$\mathbb{E}_{q_K}[h(z)] = \mathbb{E}_{q_0}[h(f_K \circ f_{K-1} \circ \dots \circ f_1(z_0))]. \quad (2.59)$$

Note that in the case where $h(\mathbf{z})$ does not depend on q_K (which is not the case in maximum-likelihood estimation), this does not even require computation of the log det-Jacobian terms. The invertible flows can be interpreted as applying a series of expansions or contractions onto the initial density function. The map $\mathbf{z}' = f(\mathbf{z})$ can push points towards the interior of a region in \mathbb{R}^d , increasing the density inside the region and decreasing it outside (contraction). It can also pull the points \mathbf{z} away from a region, thus reducing the density in that region (expansion). An initially simple prior, e.g. factorized Gaussian, can be turned into increasingly flexible and multi-modal distributions by applying normalizing flows with an increasing number of transformations.

In Kingma et al. [2016b], a type of normalizing flow called inverse autoregressive flow (IAF) is introduced. The main benefits of this normalizing flow are its scalability to high dimensions, and its ability to leverage autoregressive neural network, such as those introduced in van den Oord et al. [2016b]. First, a latent variable vector is sampled using the re-parametrization trick Kingma and Welling [2014b], $\mathbf{z}_0 = \mu_0 + \sigma_0 \epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$. Then, new mean and variance parameters μ_1 and σ_1 are computed as functions of \mathbf{z}_0 using autoregressive models, and a new latent variable \mathbf{z}_1 is obtained:

$$\mathbf{z}_1 = \mu_1(\mathbf{z}_0) + \sigma_1(\mathbf{z}_0)\mathbf{z}_0. \quad (2.60)$$

Since σ_1 and μ_1 are implemented by autoregressive networks, the Jacobian $\frac{d\mathbf{z}_1}{d\mathbf{z}_0}$ is triangular with the values of σ_1 on the diagonal, and the density under the new latent variable remains efficient to compute. This transformation can be repeated an arbitrary number of times for increased flexibility in theory, but in practice a single step is used by Kingma et al. [2016b].

2.10.4 Importance weighted VAE

Overly simplistic assumptions about the posterior can lead to overly simple latent representations, that do not use the full modelling capacity of the network. Importance weighting can be used to improve inference by deriving strictly tighter lower bounds on the log-likelihood using several samples in latent space to approximate the posterior, without otherwise modifying the VAE framework. This yields increased flexibility to model posteriors that go beyond the standard VAE modelling assumptions, and can improve likelihood performance, at train or test time.

To present it, we write the ELBO differently:

$$\log p_\theta(\mathbf{x}) = \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] = \mathcal{L}_{\theta, \phi}^{elbo}(\mathbf{x}). \quad (2.61)$$

The gradients of $\mathcal{L}_{\theta, \phi}^{elbo}$ are obtained using the re-parametrization trick, using $\mathbf{z} = \mu + \sigma \epsilon$. Because the distribution on ϵ does not depend on θ , the differentiation and expectation on ϵ can be swapped:

$$\nabla_\phi \mathcal{L}_{\theta, \phi}^{elbo}(\mathbf{x}) = \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[\nabla_\phi \log \frac{p_\theta(\mathbf{x}, \mathbf{z}(\epsilon, \phi))}{q_\phi(\mathbf{z}(\epsilon, \phi)|\mathbf{x})} \right]. \quad (2.62)$$

The gradient inside the expectation can be approximated via Monte Carlo estimation, by generating k samples of ϵ and using standard back-propagation. Let $w_{\theta,\phi}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}, \mathbf{z})/q_{\phi}(\mathbf{z}|\mathbf{x})$, then

$$\nabla_{\phi} \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right] \approx \frac{1}{k} \sum_{i=1}^k \nabla_{\phi} \log w_{\theta,\phi}(\mathbf{x}, \mathbf{z}(\epsilon_k, \phi)). \quad (2.63)$$

This yields an unbiased estimate of $\nabla_{\phi} \mathcal{L}(\mathbf{x})$, with a variance that decreases as k increases. In the case where $k > 1$, the additional samples can also be used to construct a tighter lower-bound using the Jensen inequality:

$$\mathcal{L}_k^{elbo}(\mathbf{x}, \theta, \phi) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_k \sim q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\ln \frac{1}{k} \sum_{i=1}^k w_{\theta,\phi}(\mathbf{x}, \mathbf{z}_i) \right] \quad (2.64)$$

$$\leq \ln \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_k \sim q_{\phi}(\mathbf{z}|\mathbf{x})} \frac{1}{k} \sum_{i=1}^k w(\mathbf{x}, \mathbf{z}_i) \quad (2.65)$$

$$= \ln \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} w_{\theta,\phi}(\mathbf{x}, \mathbf{z}) \quad (2.66)$$

$$= \ln p_{\theta}(\mathbf{x}). \quad (2.67)$$

In the above expression, the standard VAE lower bound is recovered for $k = 1$. For all k , the lower bounds tighten as k increases, i.e. $F_k \leq F_{k+1} \leq \ln p(\mathbf{x})$. [Burda et al. \[2016\]](#) showed that if the weights are bounded, then $\mathcal{L}_k \rightarrow \ln p(\mathbf{x})$ as $k \rightarrow \infty$. These tighter bounds can be used in two ways: i) to improve the training objective, in which case $k \approx 10$ is typically used as a trade-off between tightness and computational efficiency, and ii) only at test time, to define a more accurate bound on the log-likelihood of an already trained VAE, in which case $k \approx 10^3$ can be used. The gradients of the importance weighted lower bound can be expressed as:

$$\nabla \mathcal{L}_k^{elbo}(\mathbf{x}, \theta, \phi) = \mathbb{E}_{\mathbf{z}_{1:k} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} \sum_{i=1}^k \widetilde{w_{\theta,\phi}^i} \nabla (\ln p(\mathbf{x}, \mathbf{z}_i) - \ln q_{\phi}(\mathbf{z}_i|\mathbf{x})). \quad (2.68)$$

This update rule is similar to that of the VAE, but the samples weighted w.r.t. the true posterior. Normalized importance weights are computed as: $\widetilde{w_{\theta,\phi}^i} = w_{\theta,\phi}(\mathbf{x}, \mathbf{z}_i) / \sum_{j=1}^k w_{\theta,\phi}(\mathbf{x}, \mathbf{z}_j)$. This allows for more accurate models with complex posteriors: poor choices of \mathbf{z} will be penalised proportionally to the importance weights. This is to be contrasted with the mode-seeking behaviour of the standard ELBO, which will strongly penalise the model for putting mass in low density regions of the true posterior rather than in high density ones. Richer approximate posteriors can be learned owing to this relaxation.

2.10.5 Quantized variational auto-encoders

The VAE model can be used with a discrete latent space rather than a continuous one, as pioneered by [van den Oord et al. \[2017\]](#) and further extended by [Razavi et al. \[2019b\]](#) and [Fauw et al. \[2019\]](#). To do so, a finite number K of latent vectors e_i of dimension D are learned by gradient

descent, constituting a codebook, $e = \{e_1 \dots e_K\}$. Given an input x , the encoder f_ϕ yields $f_\phi(x)$, which is then discretized by a nearest neighbour look-up, using the codebook e :

$$q_\phi(z = k|x) = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|f_\phi(x) - e_j\|_2, \\ 0 & \text{otherwise} \end{cases}, \quad (2.69)$$

The discretized encoding is then the input to the decoder, let us denote it $z_{q_\phi(x)}$. The embedding space e is trained together with the parameters of the encoder and decoder. Note that this can be extended to use more than one element of the codebook per image, for instance by using a latent vector field with a spatial resolution of $H \times H$, and with D seen as a channel dimension. In that case the codebook can yield $K^{(H \times H)}$ different combinations of shape (H, H, D) .

This formulation does not break the variational interpretation of the auto-encoder, in the sense that $\log p(x)$ is bounded with an ELBO. Indeed $q_\phi(z = k|x)$ is deterministic, and a uniform prior over z yields a constant KL divergence, equal to $\log K$, per latent embedding. The likelihood of the model is

$$\log p(x) = \log \sum_k p(x|z_k)p(z_k). \quad (2.70)$$

This can be bounded using Jensen's inequality: $\log p(x) \geq \log p(x|z_{q_\phi(x)})p(z_{q_\phi(x)})$, which is the bound reported to evaluate the model. Equation 2.69 is not differentiable, so training of the encoder is done by copying the gradients from the discretized input of the decoder $z_{q_\phi(x)}$ to the encoder output $f_\phi(x)$. This does not provide signal to train the embedding vectors e_i . Therefore an l_2 distance measuring the error introduced by the nearest-neighbour lookup is also minimized. The full loss function is summarized in Equation 2.71 where "sg" stands for the stop-gradient operator and β is an hyper-parameter to be tuned.

$$\mathcal{L}_{vq-vae} = \log p_\theta(x|z_{q_\phi(x)}) + \|\operatorname{sg}[f_\phi(x)] - e\|_2^2 + \beta \|f_\phi(x) - \operatorname{sg}[e]\|_2^2, \quad (2.71)$$

One interesting property of this approach is that because the KL cost of the encoder is constant, it can be used together with very expressive autoregressive decoders without suffering from posterior collapse, as shown by Fauw et al. [2019].

Auto-regressive priors. As in standard VAEs, there is a miss-match between the prior and true posterior. Therefore samples from the prior do not perfectly match samples seen by the decoder at training time, which degrades sample quality. To improve on this, it is possible to train an additional prior with the posterior distribution as target, which will hopefully be closer to the posterior than the constant, uniform prior. Autoregressive models, which perform well on discrete data, can be used, and Razavi et al. [2019b] has shown this to be beneficial to sample quality.

2.11 Lipschitz continuity for generative adversarial networks

We now make a more in depth presentation of adversarial models, introduced in Section 2.7. A lot of research has been devoted to developing more stable training procedures, through the

design of better losses. In Section 2.11, we present a connection of adversarial training with optimal transport that has led to a now almost ubiquitous variant of GAN regularization, based on Lipschitz continuity. Generative adversarial networks are notoriously unstable to optimize and research has been devoted to improving adversarial training procedures. One variation that has emerged is the use of Lipschitz continuity as a regularizer, which can be motivated with an interesting connection to optimal transport and the Wasserstein distance. We chose to present this development in this chapter because it may be, at present, the most widely used GAN variant.

Limitations of the vanilla GAN. One source of difficulty in training GANs is that strong discriminators lead to *vanishing gradients* of $\mathbb{E}_{p_z} \ln(1 - D_\phi(G_\theta(z)))$ w.r.t. θ [Arjovsky et al., 2017]. This phenomenon is especially prone to happen early in training, when the generator is essentially random and it is easy for the discriminator to be very accurate. This makes it important to tune the capacity and training regime of the discriminator. To mitigate vanishing gradients early in training, Goodfellow et al. [2014] proposed to train the generator by minimizing $-\mathbb{E}_{p_z} \ln(D(G(z)))$ to boost the gradient signal in early training. This loss has the same stable points in the minimax optimization of $V(\theta, \phi)$ and helps in early training, but the problem remains: if at some point D_ϕ becomes too good, its gradients vanish and no signal remains to train G_θ . This is a motivation for designing better adversarial losses.

Both maximum-likelihood estimation and standard GAN objective rely on Kullback-Liebler divergence minimization. Indeed recall that the optimal loss approximated by D_ϕ is the Jensen-Shannon divergence, composed of two KL divergences. An important property of $D_{KL}(p||q)$ is that it is *infinite* if p has a zero in the support of q , which is easy to check from the definition $D_{KL}(p||q) = \int_{\mathcal{X}} p(\mathbf{x}) [\ln q(\mathbf{x}) - \ln p(\mathbf{x})] d\mathbf{x}$. This means it can not be used to train a model that does not cover the full support of the training set. Because in GANs latent variable vectors \mathbf{z} live in a lower dimensional space than natural images, $G_\theta(\mathbf{z})$ is a low-dimensional manifold of \mathbb{R}^N , and has degenerate support almost surely [Arjovsky et al. [2017]]. Therefore, the optimal loss approximated by D_ϕ is degenerate. Note that in maximum-likelihood estimation this problem is handled by adding volume around the low dimensional manifold $G_\theta(\mathbf{z})$, for instance with Gaussian noise, to obtain a density with non-degenerate support, see Section 2.9.1 for detail.

The earth-mover distance as an alternative adversarial loss. Alternative adversarial losses can be derived using optimal transport. Let us consider a joint distribution $\gamma(\mathbf{x}, \mathbf{y})$ with marginals $p(\mathbf{x}) = \gamma(\mathbf{x})$ and $q(\mathbf{y}) = \gamma(\mathbf{y})$. The conditional $\gamma(\mathbf{y}|\mathbf{x})$ can be seen as “moving mass” to transform $p(\cdot)$ into $q(\cdot)$. A notion of cost associated with a given transformation γ can be defined as:

$$T(\gamma) = \int_{\mathbf{x}, \mathbf{y}} \gamma(\mathbf{x}, \mathbf{y}) \|\mathbf{x} - \mathbf{y}\| = \int_{\mathbf{x}} p(\mathbf{x}) \int_{\mathbf{y}} \gamma(\mathbf{y}|\mathbf{x}) \|\mathbf{x} - \mathbf{y}\| \quad (2.72)$$

This definition is rather intuitive: if $\gamma(\mathbf{y}|\mathbf{x})$ is high, γ is likely to transform \mathbf{x} into \mathbf{y} , and thus to require moving by $\|\mathbf{x} - \mathbf{y}\|$. $T(\gamma)$ is thus the *cost of transformation using γ* . One can then seek to find a γ that minimizes the cost of transportation. This cost is called the Wasserstein distance D_{WS} , and is the *cost of optimal transportation*: $D_{WS}(p||q) = \inf_{\gamma \in \Gamma(p, q)} T(\gamma)$. Intuitively, it is based on a notion of distance between p and q , rather than a notion of overlap as is the case for

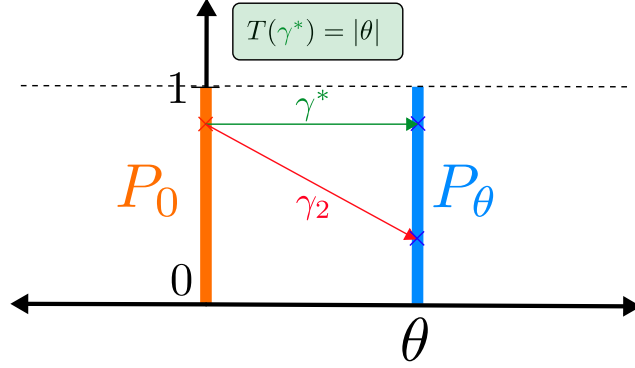


Figure 2.13: If p_0 is the uniform distribution on the orange line, and p_θ is uniform on the blue line, the optimal transport from p_0 to p_θ is γ^* and has cost $T(\gamma^*) = |\theta|$. The cost can be computed despite the fact that p_0 and p_θ are disjoint. Figure adapted from [Arjovsky et al. \[2017\]](#).

D_{KL} , and so is defined between densities that have non-overlapping support.

Consider an example where \mathbb{R}^2 is the ambient space, with $\mathbf{x} = (x_1, x_2)$. Assume two densities p_0 and p_θ with low dimensional support: p_0 is uniform on $x_2 \in [0, 1]$ for $x_1 = 0$, and p_θ is uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$. This is illustrated in Figure 2.13. Let us compare D_{KL} , D_{JS} and D_{WS} in this special case. If $\theta = 0$, all measures are zero as $p_\theta = p_0$. For $\theta \neq 0$, we have $D_{KL}(p_0||p_\theta) = \infty$ and $D_{JS}(p_0||p_\theta) = \ln 2$. Although $D_{JS}(p_0||p_\theta)$ can be computed, it can not be used for training: it does not depend on θ , and so can not be used to bring p_θ closer. But the Wasserstein distance, based on a measure of the proximity of supports can be used: in this case, D_{WS} is easy to compute, and intuitive. Indeed the optimal transport is an horizontal translation by $(0, \theta)$, i.e. $\gamma(\mathbf{x}|\mathbf{y}) = \delta(\mathbf{x} - \mathbf{y} - \begin{pmatrix} 0 \\ \theta \end{pmatrix})$ with constant cost $|\theta|$ for every pair (\mathbf{x}, \mathbf{y}) so $D_{WS}(p_0||p_\theta) = |\theta|$.

Dual definition and approximation. In general, the Wasserstein distance is challenging to compute. For high dimensional data an arbitrary γ can not be integrated out in closed form, so it is impossible to find the infimum on γ . However, the Kantorovich-Rubinstein duality Theorem [\[Villani, 2009\]](#) states that D_{WS} can equivalently be defined as:

$$D_{WS}(p^*||q) = \inf_{\gamma \in \Gamma(p, q)} T(\gamma) = \frac{1}{k} \max_{\|D_\phi\|_L \leq k} \mathbb{E}_{p^*} D_\phi(\mathbf{x}) - \mathbb{E}_{p_z} D_\phi(G_\theta(\mathbf{z})), \quad (2.73)$$

In the previous equations, $\|\cdot\|_L$ denotes the *Lipschitz* norm. Intuitively, this theorem says that finding the optimal transport between p and q is the same as finding a Lipschitz function D_ϕ that maximally separates the two. This result is similar in spirit to the duality in linear programming. The constant k can be viewed as a scaling factor and safely set to 1.

This formulation points to approximation schemes for D_{WS} . Suppose taking the supremum over k -Lipschitz function is intractable. Then, perhaps the supremum can be taken on a strict subset

$\mathcal{F}_k \subset \{f \mid \|f\|_L \leq k\}$. If \mathcal{F}_k "covers the full set well" in some sense, then the approximation should be good. In the context of deep learning, the *universal approximation theorem* [Cybenko, 1989] can be invoked: with enough flexibility (number of "neurons"), any k -Lipschitz function can in theory be approximated to arbitrary precision. So in practice, D_ϕ is restricted to some deep network architecture, parametrized by ϕ , such that $\|D_\phi\|_L \leq k$, and ϕ is optimized by gradient descent to find the supremum.

From theory to practice.

The dual formulation of the loss in Equation 2.73 now looks very similar to the training loss of a GAN discriminator:

$$\mathcal{L}_{wgan} = \frac{1}{k} \max_{\|D_\phi\|_L \leq k} \{\mathbb{E}_{p^*}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D_\phi(G_\theta(\mathbf{z}))]\} \quad (2.74)$$

$$\mathcal{L}_{gan} = \frac{1}{k} \max_{D_\phi} \{\mathbb{E}_{p^*}[\log(D_\phi(\mathbf{x}))] + \mathbb{E}_{p_z}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]\} \quad (2.75)$$

There are two differences: i) \log has disappeared, and ii) the Lipschitz norm of D_ϕ is constrained. Theoretically i) should not matter: the universal approximation theorem states that \exp can be approximated, so the \log can be inverted by setting $\bar{D}_\phi = \exp \circ D_\phi$. Nevertheless, this difference can have a big impact in practice: the squashing effect of the \log function can strongly aggravate the vanishing gradient problem when points are far from the decision boundary.

Constraining the Lipschitz norm is easy in practice. In the case where ReLu [Glorot et al., 2011] non-linearities are used, they do not modify the Lipschitz norm. It is thus enough to control the norm of each layer l_i , and use the bound:

$$\|D_\phi\|_L \leq \prod_{i=1}^L \|l_i\|_L. \quad (2.76)$$

Without loss of generality, k can be set to 1, so if $\|l_i\| \leq 1$ for all i , $\|D_\phi\| \leq 1$. Because each l_i is a linear operation, $\|l_i\|$ is the largest singular value of l_i , $\sigma(l_i)$. So ensuring $\|D_\phi\| \leq 1$ can be done by bounding $\sigma(l_i)$ by 1 for all i . This can be enforced by clipping the weights of the discriminator, as originally proposed in Arjovsky et al. [2017]. An other option is to add a penalty on the magnitude of gradients [Gulrajani et al., 2017a]:

$$G_{\text{pen}} = \lambda \mathbb{E}_{\mathbf{x}} [(\|\nabla_{\mathbf{x}} D(\mathbf{x})\|_2 - 1)^2]. \quad (2.77)$$

A third possibility is to estimate the largest singular value of each layer l_i , and normalize: $\tilde{l}_i \leftarrow l_i / \sigma(\hat{l}_i)$. This can be done using the power iteration method, see Miyato et al. [2018b]. Controlling the Lipschitz norm of the discriminator has been shown to be of great help in practice [Miyato et al., 2018b], and is now common practice.

The use of D_{WS} instead of D_{KL} , which seems crucially different at first, leads to a loss that is rather similar to the vanilla GAN loss. How relevant is the analysis in Section 2.11 in practice, and could we simply say that penalizing the Lipschitz norm is a good regularizer? First, it is

important to note that the analysis in Section 2.11 regards the *ideal losses*, D_{KL} vs. D_{WS} . In practice, both are (loosely) approximated by similar discriminators, and a property that is true of the limits of sequences of functions is not necessarily true of the functions themselves. In particular, note that in the standard GAN, having distributions with non-overlapping supports does not break D_ϕ , in the sense that a finite value is computed by the discriminator. This ability to handle support that don't overlap is precisely the reason why it is possible to train without adding volume around the low-dimensional manifold learned by the model, unlike in VAEs where this breaks likelihood computations. Overall, this analysis guarantees that the optimal solution $D_\phi^*(\theta)$ is well behaved in the Wasserstein framework, but may be less critical when considering how to approximate it with a discriminator. However, it does provide an insightful connection to optimal transport, and has led to the now standard use of Lipschitz regularization.

2.12 Evaluation metrics for generative models

Evaluating generative models applied to complex natural images is still an open research problem. Several metrics exist, with their complementarities and respective drawback. The classical evaluation is to report the log-likelihood performance of the model, normalised by the number of pixel values, which yields the Bits-per-dim metric (BPD). By reporting it on unseen data, one can detect over-fitting and know how well the model generalises to the full support of p^* . In practice, image data is discretized, which can complicate log-likelihood evaluations. In sections 2.12.1 and 2.12.2, we present two ways of dealing with it, namely data dequantization and practical discrete parametric densities, which have an impact on both training and evaluation. One drawback of BPD is that it favours models with good coverage of the support over models that produce good looking images but do not cover the full support. Other metrics that better evaluate the visual quality of samples are therefore also desirable. Several of these have been developed with GAN evaluation in mind, because i) GANs do not readily offer log-likelihood evaluations and ii) this type of metric is more aligned with the training objective of GANs. We present them in Section 2.12.3.

2.12.1 Data dequantization

In practice, image datasets such as CIFAR10 or ImageNet, are discrete representations of continuous signals that has been quantized (e.g. 8-bits colors). Though it is natural to assume continuous density models for natural images, fitting such models to discrete data is degenerate. This is because the probability of a singleton, which is of measure 0, under a continuous model is always 0. Training on discrete data anyway produces a degenerate solution that collapses probability mass to discrete datapoints, and evaluating $p_\theta(x)$ will yield very high values, that in theory tend to $+\infty$. The first solution, called *dequantization*, is to convert the discrete data distribution into a continuous distribution. The second is to use discrete output distributions, despite the continuous nature of real images.

Dequantization can be performed by adding uniform noise to the discrete data. For instance if x has D dimensions, and each discrete component is encoded with 8-bits, (i.e. takes values

in $\{0, 1, \dots, 255\}$) then the data can be dequantized by applying $\mathbf{y} = \mathbf{x} + \mathbf{u}$, with \mathbf{u} sampled uniformly from $[0, 1)^D$. This is equivalent to using rectangular approximations of the continuous density function to integrate it. As shown by Theis et al. [2016], training a continuous p_θ on \mathbf{y} can be interpreted as training another, related and discrete, model P_θ on the discrete data \mathbf{x} by maximizing a lower bound on its log-likelihood. Indeed, let

$$P_\theta(\mathbf{x}) := \int_{[0,1)^D} p_\theta(\mathbf{x} + \mathbf{u}) d\mathbf{u} \quad (2.78)$$

If P^* is the true density of the discrete data (i.e. the density obtained by discretizing the true, continuous, data distribution), and p^* is the distribution of uniformly dequantized data, Jensen's inequality yields

$$\mathbb{E}_{\mathbf{y} \sim p^*} [\log p_\theta(\mathbf{y})] = \sum_{\mathbf{x}} P^*(\mathbf{x}) \int_{[0,1)^D} \log p_\theta(\mathbf{x} + \mathbf{u}) d\mathbf{u} \quad (2.79)$$

$$\leq \sum_{\mathbf{x}} P^*(\mathbf{x}) \log \int_{[0,1)^D} p_\theta(\mathbf{x} + \mathbf{u}) d\mathbf{u} \quad (2.80)$$

$$= \mathbb{E}_{\mathbf{x} \sim P^*} [\log P_\theta(\mathbf{x})] \quad (2.81)$$

Thus p_θ can not collapse onto the discrete data, because its objective is bounded above by the log-likelihood of a discrete model and so bounded by a finite value. Uniform dequantization is correct but can be harmful to performance. Intuitively, the data is corrupted with random noise from which no signal can be extracted, and the model needs to learn to be invariant to that noise. Indeed p_θ needs to assign uniform density to unit hypercubes $\mathbf{x} + [0, 1)^D$, centered around the data \mathbf{x} . Smooth function approximators are not well suited to that task that requires using high-frequencies, which is in turn usually discouraged by regularization. Variational inference can be used instead to perform dequantization, as proposed in Ho et al. [2019]. To build a better dequantization noise distribution $q(\mathbf{u})$, still with support over $\mathbf{u} \in [0, 1)^D$, q can be made a function of \mathbf{x} , $q(\mathbf{u}|\mathbf{x})$. Then for all q :

$$\mathbb{E}_{\mathbf{x} \sim P^*} [\log P_\theta(\mathbf{x})] = \mathbb{E}_{\mathbf{x} \sim P^*} \left[\log \int_{[0,1)^D} q(\mathbf{u}|\mathbf{x}) \frac{p_\theta(\mathbf{x} + \mathbf{u})}{q(\mathbf{u}|\mathbf{x})} d\mathbf{u} \right] \quad (2.82)$$

$$\geq \mathbb{E}_{\mathbf{x} \sim P^*} \left[\int_{[0,1)^D} q(\mathbf{u}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x} + \mathbf{u})}{q(\mathbf{u}|\mathbf{x})} d\mathbf{u} \right] \quad (2.83)$$

$$= \mathbb{E}_{\mathbf{x} \sim P^*} \mathbb{E}_{\mathbf{u} \sim q(\cdot|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x} + \mathbf{u})}{q(\mathbf{u}|\mathbf{x})} \right] \quad (2.84)$$

Using an expressive q allows p_θ to place density in each hypercube $\mathbf{x} + [0, 1)^D$ according to flexible distribution $q(\mathbf{u}|\mathbf{x})$ which is more natural for p_θ . In particular, a uniform distribution on \mathbf{u} is within the family of distributions that can be learned by q . Therefore by taking the optimum over that family we necessarily get a tighter bound than by using a uniform q . In practice, q is implemented using a small network and optimized jointly with p_θ using gradient descent, estimating the gradients by monte-carlo approximation.

2.12.2 Discrete parametric densities

Rather than dequantize the discrete data, it is possible to use discrete parametric densities. One possibility, proposed by Oord et al. [2016] is to use a softmax operator: given scores s_i for each color class,

$$p_{\text{soft}}(k) = \frac{e^{s_k}}{\sum_i e^{s_i}} \quad (2.85)$$

This has the advantage that no parametric assumptions about the distributions predicted need to be made: a softmax can be arbitrarily multimodal, skewed, peaked or long tailed. There is also no need to worry about parts of the distribution mass that may be outside the interval 0, 255. A drawback is that no prior information is embedded about the relations between the 256 color categories. The notion that values i and $i + 1$ are neighbors is lost and has to be learned by the model, and small mistakes are penalised as much as big ones.

Another option is model color intensity with a continuous distribution, but taken from a family of functions that can be discretized exactly. Kingma et al. (2016) use the logistic density which is parametrised by a location parameter μ_i , a scale parameter σ_i , and resembles a Gaussian density. It integrates to the sigmoid function σ , so it is easy to discretize:

$$P(x_i|\pi, \mu, s) = \sigma((x_i + 0.5 - \mu_i)/s_i) - \sigma((x_i - 0.5 - \mu_i)/s_i) \quad (2.86)$$

Salimans et al. [2017a], extend this approach using mixtures of logistic distributions, to have multiple modes as with the softmax. In practice a relatively small number of mixture components (5 to 10), is enough for natural images datasets.

2.12.3 GAN evaluations

Quantitative evaluation of Generative Adversarial Networks is challenging, in part due to the absence of log-likelihood but also because GANs are primarily optimised for image quality rather than support coverage. To measure the quality of samples, the ideal would be to directly evaluate $p^*(x)$. That is of course impossible, since learning p^* is precisely the problem we are trying to solve. A simple solution could be to use the dataset to approximate p^* by fitting parzen windows on data points [Goodfellow et al., 2014]. In high dimensions, however, this provides very poor evaluations [Theis et al., 2016].

It is also important to note that any metric evaluating quality only would be degenerate as an evaluation for p_θ , as collapsing p_θ to the mode of the distribution would maximize it. Such a metric would need to be complemented with an other measure.

The most popular solutions to automate the qualitative evaluation of samples rely on the idea of relying on a pretrained image classifier to provide the evaluation. Intuitively, we know how to train classifiers that are very accurate on unseen data, which means that they have some "understanding" of what real images should look like. For instance, one can consider that a classifier should assign a low entropy distribution over classes to a good sample, and be confused

and assign a high-entropy distribution over classes for a poor sample. This is the idea behind the Inception Score (IS), proposed by [Salimans et al. \[2016b\]](#). Another possibility is to compare the behaviour of a classifier when applied to a set of samples *vs.* a set of real images, which is the intuitive idea behind the Fréchet Inception distance (FID), proposed in [Heusel et al. \[2017\]](#).

Inception score (IS). The inception score is a statistic of the generated images, based on an external inception network [[Szegedy et al., 2015](#)], trained for classification on ImageNet. The score is given by:

$$\text{IS}(p_\theta) = \exp(\mathbb{E}_{x \sim p_\theta} D_{KL}(p(y|x) || p(y))), \quad (2.87)$$

where $p(y|x)$ is the conditional class distribution obtained by applying the pre-trained classification network to the generated images, and $p(y) = \int_x p(y|x)p_\theta(x)$ is the class marginal over the generated images. To understand this score intuitively, decompose the Kullback–Liebler divergence in two:

$$\mathbb{E}_{x \sim p_\theta} [D_{KL}(p(y|x) || p(y))] = \mathbb{E}_{x \sim p_\theta} \left[\int_y p(y|x) \log p(y|x) dy - \int_y p(y|x) \log(p(y)) dy \right] \quad (2.88)$$

The first term inside the expectation is minus the entropy of the classifier, for a fixed x . This captures the idea that an image is considered "good" if it can be well identified as a known object by the classifier, in which case the distribution over classes has low entropy. When taking the expectation over x , the average entropy should be low. The second term is the cross-entropy between $y|x$ and y , and captures the idea that $y|x$ should have as much variability as y and thus the entropy of averages should be high.

Fréchet Inception distance (FID). The Fréchet Inception distance compares the distributions of Inception embeddings i.e., activations from the penultimate layer of the Inception network, of real ($p_r(x)$) and generated ($p_g(x)$) images. Both of these distributions are modelled as multi-dimensional Gaussians parametrized by their respective mean and covariance. Finally, the score is computed as the Fréchet distance between the two Gaussian densities, which can be reduced to:

$$d^2((\mathbf{m}_r, \mathbf{C}_r), (\mathbf{m}_g, \mathbf{C}_g)) = \|\mathbf{m}_r - \mathbf{m}_g\|^2 + \text{Tr}(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r \mathbf{C}_g)^{\frac{1}{2}}), \quad (2.89)$$

where $(\mathbf{m}_r, \mathbf{C}_r)$, $(\mathbf{m}_g, \mathbf{C}_g)$ denote the mean and covariance of the real and generated image distributions respectively.

Unlike the Inception Score, the Fréchet Inception distance does compare distributions obtained with real and fake images, and in that sense is similar to a distance. This more directly captures a notion of coverage: to be similar, the distributions of both real and fake inception activations must have the same "coverage". In the case of IS, this is more indirect, and works similarly to the GAN training loss: images must have variability, so the mass must be spread out. Because images must also look good, implicitly the mass must spread to the training support. In practice, however, both metrics are heavily dominated by quality as we now demonstrate.

Practical use of IS and FID. IS and FID correlate predominantly with the quality of samples. In GAN literature, for instance [[Miyato et al., 2018a](#)], they are considered to correlate well with

human judgement of quality. An empirical indicator of that is that state-of-the-art likelihood-based models have very low IS/FID scores despite having good coverage, which shows that the low quality of their samples dominates. Conversely, state-of-the-art adversarial models have high IS/FID scores, despite suffering from mode dropping (which strongly degrades BPD). So the score is determined mainly by the high quality of their samples.

Split size	IS	FID
50k (full)	11.3411	0.00
40k	11.3388	0.13
30k	11.3515	0.35
20k	11.3458	0.79
10k	11.3219	2.10
5k	11.2108	4.82
2.5k	11.0446	10.48

Table 2.1: IS and FID scores obtained by the ground truth when progressively dropping parts of the CIFAR10 dataset. The metrics are largely insensitive to removing most of the dataset, unlike BPD. For reference, a reasonable GAN could get around 8 IS and 20 FID.

To obtain a quantitative estimation of how much entropy/coverage impacts the IS and FID measures, we evaluate the scores obtained by random subsamples of the dataset, such that the quality is unchanged but coverage progressively degrades (see details of the scores below). Table 2.1 shows that when using the full set of images (50k) the FID is 0 as the distributions are identical. Notice that as the number of images decreases, IS is very stable (it can even increase, but by very low increments that fall below statistical significance, with a typical standard deviation of 0.1). This is because the entropy of the distribution is not strongly impacted by sub-sampling, even though coverage is. FID is more sensitive, as it behaves more like a measure of coverage (it compares the two distributions). Nonetheless, the variations remain extremely low even when dropping most of the dataset. For instance, when removing 80 percent of the dataset (i.e., using 10k images), FID is at 2.10, to be compared with typical GAN values that are around 20.

These measurements demonstrate that IS and FID scores are heavily dominated by the quality of images. From this, we conclude that IS and FID can be used as reasonable proxies to assess sample quality, even though they are also slightly influenced by coverage. One should bear in mind, however, that a small increase in these scores may come from better coverage rather than improved sample quality.

Chapter 3

Auxiliary Guided Autoregressive Variational Autoencoder

Contents

3.1	Introduction	48
3.2	Related work	48
3.3	Auxiliary guided autoregressive variational autoencoders	50
3.4	Experimental evaluation	56
3.5	Conclusion	63

Successful approaches to generative modelling of high-dimensional data, in particular natural image collections, include latent variable models as discussed in Section 2.5, and autoregressive models as discussed in Section 2.8. The complementary strengths of these approaches, to model global and local image statistics respectively, suggest hybrid models that encode global image structure into latent variables while autoregressively modeling low level detail. The complementary strengths of these approaches, suggest hybrid models that encode global image structure into latent variables while auto-regressively modelling low level detail. However, a naive construction of this type yields models that ignore the latent variables and only rely on autoregressive modelling, a phenomenon known as the information preference property. Previous approaches to such hybrid models [Chen et al., 2017, Gulrajani et al., 2017c] circumvent this problem by restricting the capacity of the autoregressive decoder to prevent degenerate models. In contrast we present a training procedure relying on an auxiliary loss function that controls which information is captured by the latent variables and what is left to the autoregressive decoder. Our approach can leverage arbitrarily powerful autoregressive decoders, achieves state-of-the art quantitative performance among models with latent variables, and generates qualitatively convincing samples. The material presented in this chapter is based on the paper "Auxiliary Guided Autoregressive Variational Autoencoder", Thomas Lucas & Jakob Verbeek, European Conference on Machine Learning (ECML) 2018.

3.1 Introduction

Latent variable approaches can learn disentangled and concise representations of the data [Bengio et al., 2013], which are useful for compression [Gregor et al., 2016] and semi-supervised learning [Kingma et al., 2014, Rasmus et al., 2015]. Autoregressive models on the other hand, are very effective to model low-level image statistics, and obtain state-of-the-art likelihoods on test data. One interesting property of VAEs is that can learn latent variable representations that abstract away from low-level details, but model pixels as conditionally independent given the latent variables. This renders the generative model computationally efficient, but the lack of low-level structure modeling leads to overly smooth and blurry samples Section 2.9.1. Autoregressive models, such as pixelCNNs [Oord et al., 2016], on the other hand, estimate complex translation invariant conditional distributions among pixels. They are effective to model low-level image statistics, and yield state-of-the-art likelihoods on test data [Salimans et al., 2017a]. This is in line with the observations of Kolesnikov and Lampert [2017] that low-level image details account for a large part of the likelihood. These autoregressive models, however, do not learn a latent variable representations to support, *e.g.*, semi-supervised learning.

The complementary strengths of VAEs and pixelCNNs, modeling global and local image statistics respectively, suggest hybrid approaches combining the strengths of both. Prior work on such hybrid models needed to limit the capacity of the autoregressive decoder to prevent degenerate models that completely ignore the latent variables and rely on autoregressive modeling only [Chen et al., 2017, Gulrajani et al., 2017c]. In this chapter we describe Auxiliary Guided Autoregressive Variational autoEncoders (AGAVE), an approach to train such hybrid models using an auxiliary loss function that controls which information is captured by the latent variables and what is left to the AR decoder. See Figure 3.1 for a schematic illustration of our approach. Using high-capacity VAE and autoregressive components allows our models to obtain quantitative results on held-out data that are on par with the state of the art. Our models generate samples with both global coherence and low-level details. In Section B.2, related work on generative modelling of natural images is presented. In Section 3.3.2, the general motivation for using autoregressive decoders is given, followed by a discussion on why these models, when implemented naively, lead to posterior collapse, and finally by a presentation of our approach that circumvents this problem. Section B.4 contains experimental evaluation of the proposed approach, and ablations study to demonstrate the importance of each components.

3.2 Related work

A variety of approaches leverage deep neural networks to learn complex density models. These include the variational autoencoders and autoregressive models that form the basis of our work, but also generative adversarial networks (GANs) [Arjovsky et al., 2017, Goodfellow et al., 2014] and variable transformation with invertible functions [Dinh et al., 2017]. While GANs produce visually appealing samples, they suffer from mode dropping and their likelihood-free nature prevents measuring how well they model held-out test data, see Section 2.9.2. In particular, GANs can only generate samples on a non-linear manifold in the data space with dimension

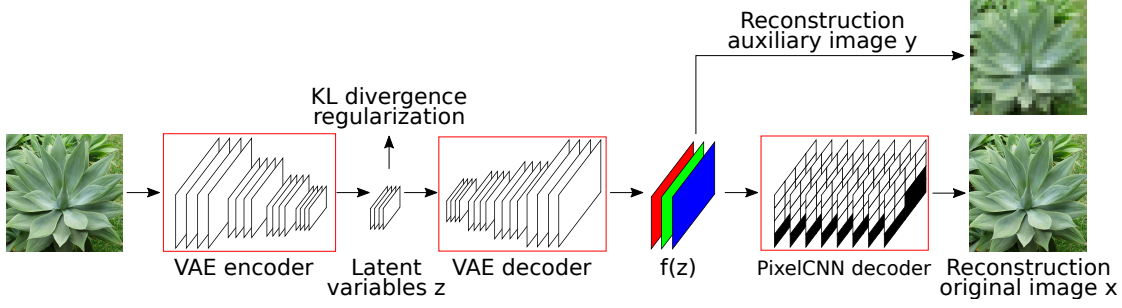


Figure 3.1: Schematic illustration of our auxiliary guided autoregressive variational autoencoder (AGAVE). The objective function has three components: KL divergence regularization, per-pixel reconstruction with the VAE decoder, and autoregressive reconstruction with the pixelCNN decoder.

equal to the number of latent variables. In contrast, this chapter focuses on probabilistic models that generalize to the entire data space, and provide likelihoods of held-out data that can be used for compression, and to quantitatively compare different models. To improve the quality of samples, a structured noise model is proposed, to reduce over-generalization in a MLE setting, see Section 2.9.1. The non-volume preserving (NVP) transformation approach of Dinh et al. [2017] chains together invertible transformations to map a basic (*e.g.* unit Gaussian) prior on the latent space to a complex distribution on the data space. This method offers tractable likelihood evaluation and exact inference, but obtains likelihoods on held-out data below the values reported using state-of-the-art VAE and autoregressive models. Moreover, it is restricted to use latent representations with the same dimensionality as the input data, and is thus difficult to scale to model high-resolution images.

Autoregressive density estimation models, such as pixelCNNs [Oord et al., 2016], admit tractable exact likelihood evaluation, while for variational autoencoders [Kingma and Welling, 2014a, Rezende et al., 2014] accurate approximations can be obtained using importance sampling [Burda et al., 2016]. Naively combining powerful pixelCNN decoders in a VAE framework results in a degenerate model which ignores the VAE latent variable structure, as explained through the lens of bits-back coding by Chen et al. [2017]. To address this issue, the capacity of the autoregressive component can be restricted. This can, for example, be achieved by reducing its depth and/or field of view, or by giving the pixelCNN only access to grayscale values, *i.e.* modeling $p(x_i | x_{<i}, z) = p(x_i | \text{gray}(x_{<i}), z)$ [Chen et al., 2017, Gulrajani et al., 2017c]. This forces the model to leverage the latent variables z to model part of the dependencies among the pixels. This approach, however, has two drawbacks. (i) Curbing the capacity of the model is undesirable in unsupervised settings where training data is abundant and overfitting unlikely, and is only a partial solution to the problem. (ii) Balancing what is modeled by the VAE and the pixelCNN by means of architectural design choices requires careful hand-design and tuning of the architectures. This is a tedious process, and a more reliable principle is desirable. To overcome these drawbacks, we propose to instead control what is modeled by the VAE and pixelCNN with an auxiliary loss on the VAE decoder output before it is used to condition the autoregressive

decoder. This allows us to “plug in” powerful high-capacity VAE and pixelCNN architectures, and balance what is modeled by each component by means of the auxiliary loss.

In a similar vein, [Kolesnikov and Lampert \[2017\]](#) force pixelCNN models to capture more high-level image aspects using an auxiliary representation y of the original image x , *e.g.* a low-resolution version of the original. They learn a pixelCNN for y , and a conditional pixelCNN to predict x from y , possibly using several intermediate representations. This approach forces modeling of more high-level aspects in the intermediate representations, and yields visually more compelling samples. [Reed et al. \[2017\]](#) similarly learn a series of conditional autoregressive models to upsample coarser intermediate latent images. By introducing partial conditional independencies in the model they scale the model to efficiently sample high-resolution images of up to 512×512 pixels. [Gregor et al. \[2016\]](#) use a recurrent VAE model to produce a sequence of RGB images with increasing detail derived from latent variables associated with each iteration. In [\[Denton et al., 2015\]](#), adversarially generated images are progressively refined using a Laplacian pyramid framework. Like our work, all these models work with intermediate representations in RGB space to learn accurate generative image models.

3.3 Auxiliary guided autoregressive variational autoencoders

For self-containedness, we give a brief overview of variational autoencoders and relevant limitations in Section 3.3.1, before we present our approach to learning variational autoencoders with autoregressive decoders in Section 3.3.2.

3.3.1 Variational autoencoders with autoregressive decoders

Variational autoencoders [\[Kingma and Welling, 2014a, Rezende et al., 2014\]](#) learn deep generative latent variable models using two neural networks. The “decoder” network implements a conditional distribution $p_\theta(x|z)$ over observations x given a latent variable z , with parameters θ . Together with a basic prior on the latent variable z , *e.g.* a unit Gaussian, the generative model on x is obtained by marginalizing out the latent variable:

$$p_\theta(x) = \int p(z)p_\theta(x|z) dz. \quad (3.1)$$

The marginal likelihood can, however, not be optimized directly since the non-linear dependencies in $p_\theta(x|z)$ render the integral intractable. To overcome this problem, an “encoder” network is used to compute an approximate posterior distribution $q_\phi(z|x)$, with parameters ϕ . The approximate posterior is used to define a variational bound on the data log-likelihood, by subtracting the Kullback-Leibler divergence between the true and approximate posterior:

$$\ln p_\theta(x) \geq \mathcal{L}(\theta, \phi; x) = \ln(p_\theta(x)) - D_{\text{KL}}(q_\phi(z|x) || p_\theta(z|x)) \quad (3.2)$$

$$= \underbrace{\mathbb{E}_{q_\phi}[\ln(p_\theta(x|z))]}_{\text{Reconstruction}} - \underbrace{D_{\text{KL}}(q_\phi(z|x) || p(z))}_{\text{Regularization}}. \quad (3.3)$$

The decomposition in Equation 3.3 interprets the bound as the sum of a reconstruction term and a regularization term. The first aims to maximize the expected data log-likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ given the posterior estimate $q_\phi(\mathbf{z}|\mathbf{x})$. The second term prevents $q_\phi(\mathbf{z}|\mathbf{x})$ from collapsing to a single point, which would be optimal for the first term. See Section 2.5 for a more detailed presentation.

Variational autoencoders typically model the dimensions of \mathbf{x} as conditionally independent,

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^D p_\theta(x_i|\mathbf{z}), \quad (3.4)$$

for instance using a factored Gaussian or Bernoulli model, see *e.g.* Kingma and Welling [2014a], Kingma et al. [2016a], Yan et al. [2016]. The conditional independence assumption makes sampling from the VAE efficient: since the decoder network is evaluated only once for a sample $\mathbf{z} \sim p(\mathbf{z})$ to compute all the conditional distributions $p_\theta(x_i|\mathbf{z})$, the x_i can then be sampled in parallel. This comes at the price of over-generalisation and blurrier samples. Indeed, a result of relying on the latent variables to account for all pixel dependencies, however, is that all low-level variability must be modeled by the latent variables. Consider, for instance, a picture of a dog, and variants of that image shifted by one or a few pixels, or in a slightly different pose, with a slightly lighter background, or with less saturated colors, *etc.* If these factors of variability are modeled using latent variables, then these low-level aspects are confounded with latent variables relating to the high-level image content. If the corresponding image variability is not modeled using latent variables, it will be modeled as independent pixel noise. In the latter case, using the mean of $p_\theta(\mathbf{x}|\mathbf{z})$ as the synthetic image for a given \mathbf{z} results in blurry samples, since the mean is averaged over the low-level variants of the image. Sampling from $p_\theta(\mathbf{x}|\mathbf{z})$ to obtain synthetic images, on the other hand, results in images with unrealistic independent pixel noise.

3.3.2 Autoregressive decoders in variational autoencoders

Autoregressive density models, see *e.g.* [Germain et al., 2015, Larochelle and Murray, 2011], rely on the basic factorization of multi-variate distributions,

$$p_\theta(\mathbf{x}) = \prod_{i=1}^D p_\theta(x_i|\mathbf{x}_{<i}), \quad (3.5)$$

with $\mathbf{x}_{<i} = x_1, \dots, x_{i-1}$, and model the conditional distributions using a (deep) neural network. For image data, PixelCNNs [Oord et al., 2016, van den Oord et al., 2016c] use a scanline pixel ordering, and model the conditional distributions using a convolution neural network. The convolutional filters are masked so as to ensure that the receptive fields only extend to pixels $\mathbf{x}_{<i}$ when computing the conditional distribution of x_i . PixelCNNs can be used as a decoder in a VAE by conditioning on the latent variable \mathbf{z} in addition to the preceding pixels, leading to a variational bound with a modified reconstruction term:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi} \left[\sum_{i=1}^D \ln p_\theta(x_i|\mathbf{x}_{<i}, \mathbf{z}) \right] - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \quad (3.6)$$

The regularization term can be interpreted as a “cost” of using the latent variables. To effectively use the latent variables, the approximate posterior $q_\phi(z|\mathbf{x})$ must differ from the prior $p(z)$, which increases the KL divergence. In practice, combining a VAE with a flexible decoder (for instance an autoregressive one) leads to the latent code being ignored. This problem could be attributed to optimization challenges: at the start of training $q(z|\mathbf{x})$ carries little information about \mathbf{x} , the KL term pushes the model to set it to the prior to avoid any penalty, and training never recovers from falling into that local minimum. In fact in [Chen et al., 2017, Zhao et al., 2017] extensive explanations have been proposed, showing that the problem is deeper: for the loss in Equation 3.6 and a decoder with enough capacity, it is optimal to encode no information about x in z by setting $q(z|\mathbf{x}) = p(z)$. This is problematic: as \mathbf{x} becomes independent from z , the encoder is not used at all and meaningful latent representations cannot be learned. This behaviour is often referred to as the *information preference* property.

The information preference problem. Intuitively, the information preference property arises because the miss-match between the approximate posterior and the true, inaccessible, posterior has a cost given by the KL regularization term in the ELBO. It is worth paying that cost only if the decoder benefits even more to compensate. In practice autoregressive decoders already predict \mathbf{x} well without using any information from a latent variable z , and using an encoder q_ϕ does not improve the reconstruction cost enough to compensate the regularization cost of the information in z . To show this more formally, let us take the expectation over x of the ELBO:

$$-\mathbb{E}_{x \sim p^*}[\mathcal{L}_{\theta, \phi}^{elbo}(\mathbf{x})] = \mathbb{E}_{x \sim p^*}[-\log(p_\theta(\mathbf{x})) + D_{KL}(q_\phi(z|\mathbf{x})||p_\theta(z|\mathbf{x}))]. \quad (3.7)$$

The cross-entropy of p_θ is lower-bounded by the Shannon entropy,

$$\mathcal{H}(p^*) = \mathbb{E}_{x \sim p^*}[-\log(p^*(\mathbf{x}))] \quad (3.8)$$

so we have:

$$-\mathbb{E}_{x \sim p^*}[\mathcal{L}_{\theta, \phi}^{elbo}(\mathbf{x})] \geq \mathcal{H}(p^*) + \mathbb{E}_{x \sim p^*}[D_{KL}(q_\phi(z|\mathbf{x})||p_\theta(z|\mathbf{x}))]. \quad (3.9)$$

Let \mathcal{F}_Θ a family of generative models, $\mathcal{F}_\Theta = \{p_\theta(\mathbf{x}), \theta \in \Theta\}$. Denote by $\theta|_z$ any $\theta \in \Theta$ such that $p_{\theta|_z}$ uses it’s latent variables, and denote $\Theta|_z$ the set of such θ s, *i.e.*

$$\theta|_z \in \{\theta \in \Theta | p_\theta(z|\mathbf{x}) \neq p(z)\} = \Theta|_z.$$

Suppose a given set \mathcal{F}_Φ of inference networks, $\mathcal{F}_\Phi = \{q_\phi(z|\mathbf{x}), \phi \in \Phi\}$, does not have sufficient capacity to offer perfect inference, such that

$$\forall \phi \in \Phi, D_{KL}(q_\phi(z|\mathbf{x})||p_{\theta|_z}(z|\mathbf{x})) > 0.$$

This yields for any $\theta \in \Theta|_z$ so:

$$\underbrace{\mathcal{H}(p^*)}_A < \underbrace{\mathcal{H}(p^*) + \mathbb{E}_{x \sim p^*}[D_{KL}(q_\phi(z|\mathbf{x})||p_\theta(z|\mathbf{x}))]}_B \leq \underbrace{-\mathbb{E}_{x \sim p^*}[\mathcal{L}_{\theta, \phi}^{elbo}(\mathbf{x})]}_C. \quad (3.10)$$

This shows that the performance C for $\theta|_z \in \Theta|_z$ can not be better than B . Now assume F_θ contains models that do not use their latent variables and with enough capacity to "squeeze" between A – which is the best value any model can attain – and B . In our context, this model would typically be autoregressive so let us denote it with subscript $\theta_{<j, \perp z}$. This yields:

$$\underbrace{\mathcal{H}(p^*)}_A \leq \underbrace{\mathbb{E}_{x \sim p^*} [-\log(p_{\theta_{<j, \perp z}}(x|x_{j < i}))]}_{< C} < \underbrace{\mathcal{H}(p^*) + \mathbb{E}_{x \sim p^*} [D_{KL}(q_\phi(z|x)||p_{\theta|z}(z|x))]}_B \leq \underbrace{-\mathbb{E}_{x \sim p^*} [\mathcal{L}_{\theta, \phi}^{elbo}(x)]}_C \quad (3.12)$$

Thus, any use of z that any model p_θ might do will degrade the optimal performance that can be attained, and it is better not to use z at all and rely only on the autoregressive component of the model. If the family \mathcal{F}_Θ contains sufficiently expressive autoregressive models, these will always be preferred to the models in \mathcal{F}_Θ that do use latent variables. This is not just a theoretical problem, as in practice, using flexible autoregressive decoders naively systematically leads to posterior collapse.

3.3.3 Auxiliary Guided Autoregressive Variational Autoencoder

Several attempts have been made to circumvent the information preference property. One successful approach is to restrict the number of layers of the auto-regressive component of the decoder to a few layers. The receptive field of a given pixel x_i is then a strict subset of those in the image. Any dependence between x_i and a pixel outside of the receptive field has to go through z , or the two will be sampled independently. This is the approach taken by [Chen et al. \[2017\]](#), [Gulrajani et al. \[2017c\]](#). The autoregressive component is in charge of modelling low level detail, which it excels at, while the VAE component models global structure. One drawback is that this is only a partial solution as it does not enable the use of powerful autoregressive models, since these would again suffer from the information preference property. Other solutions to force information into z include [Zhao et al. \[2017\]](#), which compute an estimation of the mutual information between z and x , and more recently [Razavi et al. \[2019a\]](#) which proposes to constrain the posterior distribution to be at a minimal distance from the prior. Finally, when using auto-regressive decoders together with quantized latent variables, the issue of ignored latent variables can be avoided as presented in [Section 2.10.5](#).

In this section we present our approach relying on an auxiliary reconstruction loss to control the information that goes into z . To ensure meaningful latent representation learning [Chen et al. \[2017\]](#) and [Gulrajani et al. \[2017c\]](#) restrict the capacity of the pixelCNN decoder. In our approach, in contrast, it is always optimal for the autoregressive decoder, regardless of its capacity, to exploit the information on x carried by z . We rely on two decoders in parallel: the first one reconstructs an auxiliary image y from an intermediate representation $f_\theta(z)$ in a non-autoregressive manner. The auxiliary image can be either simply taken to be the original image ($y = x$), or a compressed version of it, *e.g.* with lower spatial resolution or with a coarser color quantization. The second decoder is a conditional autoregressive model that predicts x conditioned on $f_\theta(z)$. Modeling y in a non-autoregressive manner ensures a meaningful representation z and renders x and z

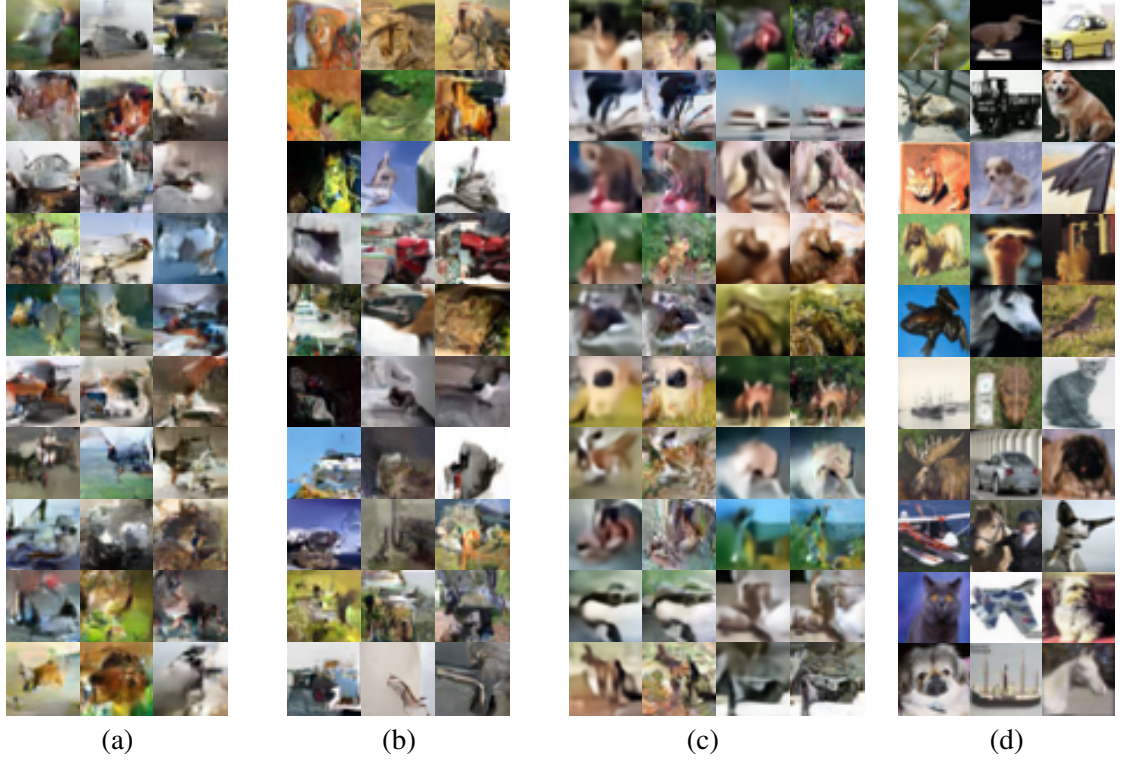


Figure 3.2: Randomly selected samples from unsupervised models trained on 32×32 CIFAR10 images: (a) IAF-VAE [Kingma et al. \[2016a\]](#), (b) pixelCNN++ [Salimans et al. \[2017a\]](#), (c) our hybrid AGAVE model and (d) real CIFAR10 images for comparison. For our model, we show the intermediate high-level representation based on latent variables (left), that conditions the final sample based on the pixelCNN decoder (right).

dependent, inducing a certain non-zero KL “cost” in (3.6). The uncertainty on \mathbf{x} is thus reduced when conditioning on \mathbf{z} , and there is no longer an advantage in ignoring the latent variable for the autoregressive decoder. We provide a more detailed explanation of why our auxiliary loss ensures a meaningful use of latent variables in powerful decoders in Section 3.3.2. To train the model we combine both decoders in a single objective function with a shared encoder network:

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{y}) = \underbrace{\mathbb{E}_{q_\phi} \left[\sum_{i=1}^D \ln p_\theta(x_i | \mathbf{x}_{<i}, \mathbf{z}) \right]}_{\text{Primary Reconstruction}} + \underbrace{\mathbb{E}_{q_\phi} \left[\sum_{j=1}^E \ln p_\theta(y_j | \mathbf{z}) \right]}_{\text{Auxiliary Reconstruction}} - \underbrace{\lambda D_{\text{KL}}(q_\phi(\mathbf{z} | \mathbf{x}) || p(\mathbf{z}))}_{\text{Regularization}}. \quad (3.13)$$

Treating \mathbf{x} and \mathbf{y} as two variables that are conditionally independent given a shared underlying latent variable \mathbf{z} leads to $\lambda = 1$. Summing the lower bounds in Eq. (3.3) and Eq. (3.6) of the marginal log-likelihoods of \mathbf{y} and \mathbf{x} , and sharing the encoder network, leads to $\lambda = 2$. Larger

values of λ result in valid but less tight lower bounds of the log-likelihoods. Encouraging the variational posterior to be closer to the prior, this leads to less informative latent variable representations.

Sharing the encoder across the two decoders is the key of our approach. The factored auxiliary VAE decoder can only model pixel dependencies by means of the latent variables, which ensures that a meaningful representation is learned. Now, given that the VAE encoder output is informative on the image content, there is no incentive for the autoregressive decoder to ignore the intermediate representation $f_\theta(\mathbf{z})$ on which it is conditioned. The choice of the regularization parameter λ and auxiliary image \mathbf{y} provide two levers to control *how much* and *what type* of information should be encoded in the latent variables.

3.3.4 It is optimal for the autoregressive decoder to use the latent variable

Using the auxiliary reconstruction term presented above, it is always optimal for the autoregressive decoder, no matter how expressive, to use the information contained in the latent variable we now show.

Intuitively, the auxiliary term requires some information to go into the latent variable, which is then available 'for free' to the autoregressive decoder. Recall the quantities involved in Equation 3.3.2:

$$\underbrace{\mathcal{H}(p^*)}_A < \underbrace{\mathcal{H}(p^*) + \mathbb{E}_{\mathbf{x} \sim p^*} [D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))]}_B \leq \underbrace{-\mathbb{E}_{\mathbf{x} \sim p^*} [\mathcal{L}_{\theta,\phi}^{elbo}(\mathbf{x})]}_C. \quad (3.14)$$

To obtain the AGAVE setting, an autoregressive decoder is added. Using again the fact that the Shannon entropy lower-bounds the cross-entropy between the autoregressive decoder and the true distribution, we obtain:

$$C + \mathcal{H}(X|Z) \leq C + \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [-\sum_i \log(p(x_i|\mathbf{z}, \mathbf{x}_{j < i}))] = C_{\text{AGAVE}}. \quad (3.15)$$

The entropy of a random variable decreases when it is conditioned on another, i.e. $\mathcal{H}(X|Z) \leq \mathcal{H}(X)$. Therefore, the theoretical lower-bound on the expected code length in our setup is always better when the autoregressive component takes Z into account, no matter its expressivity. In the limit case of an infinitely expressive autoregressive decoder, denoted by $*$, the lower bound is attained and so $C_{\text{AGAVE}}^* = C + \mathcal{H}(X|Z) \leq C + \mathcal{H}(X)$. In non-degenerate cases, the VAE is optimized to encode information about X into a meaningful Z , with potentially near perfect reconstructions, and there exists $\epsilon > 0$ such that $\mathcal{H}(X|Z) < \mathcal{H}(X) - \epsilon$, making the lower bound stricly better by a possibly big margin:

$$C_{\text{AGAVE}}^* = C + \mathcal{H}(X|Z) < C + \mathcal{H}(X). \quad (3.16)$$

More generally, even if the optimum value C_{AGAVE}^* is not attained, taking the latent variables into account is the only way for the AGAVE model to squeeze between $C + \mathcal{H}(X|Z)$ and $C + \mathcal{H}(X)$. This analysis shows that in our setup it is theoretically always better for the autoregressive model

Model		BPD	$\cdot z$	$\cdot x_{j<i}$
NICE	[Dinh et al., 2015]	4.48	✓	
Conv. DRAW	[Gregor et al., 2016]	≤ 3.58	✓	
Real NVP	[Dinh et al., 2017]	3.49	✓	
MatNet	[Bachman, 2016]	≤ 3.24	✓	
PixelCNN	[Oord et al., 2016]	3.14		✓
VAE-IAF	[Kingma et al., 2016a]	≤ 3.11	✓	
Gated pixelCNN	[van den Oord et al., 2016c]	3.03		✓
Pixel-RNN	[Oord et al., 2016]	3.00		✓
Aux. pixelCNN	[Kolesnikov and Lampert, 2017]	2.98		✓
Lossy VAE	[Chen et al., 2017]	≤ 2.95	✓	✓
AGAVE , $\lambda = 12$	(this paper)	≤ 2.92	✓	✓
pixCNN++	[Salimans et al., 2017a]	2.92		✓

Table 3.1: Bits per dimension (lower is better) of models on the CIFAR10 test data.

to make use of the latent and auxiliary representation it is conditioned on. That is true no matter how expressive the model is. It also shows that in theory our model should learn meaningful latent structure.

3.4 Experimental evaluation

In this section we describe our experimental setup, and present results on the CIFAR10 dataset.

3.4.1 Dataset and implementation

The CIFAR10 dataset [Krizhevsky, 2009] contains 6,000 images of 32×32 pixels for each of the 10 object categories *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, *truck*. The images are split into 50,000 training images and 10,000 test images. We train all our models in a completely unsupervised manner, ignoring the class information.

We implemented our model based on existing architectures, in particular we use the VAE architecture of Kingma et al. [2016a]. To deal with the fact that the data is discrete we use a logistic distributions over the RGB color values as in Kingma et al. [2016a]. We let the intermediate representation $f(z)$ output by the VAE decoder be the per-pixel and per-channel mean values of the logistics, and learn per-channel scale parameters that are used across all pixels. The cumulative density function (CDF), given by the sigmoid function, is used to compute probabilities across the 256 discrete color levels, or fewer if a lower quantization level is chosen in \mathbf{y} . Using RGB values $y_i \in [0, 255]$, we let b denote the number of discrete color levels and define $c = 256/b$. The probabilities over the b discrete color levels are computed from the logistic mean and variance μ_i and s_i as

$$p(y_i|\mu_i, s_i) = \sigma(c + c\lfloor y_i/c \rfloor|\mu_i, s_i) - \sigma(c\lfloor y_i/c \rfloor|\mu_i, s_i). \quad (3.17)$$

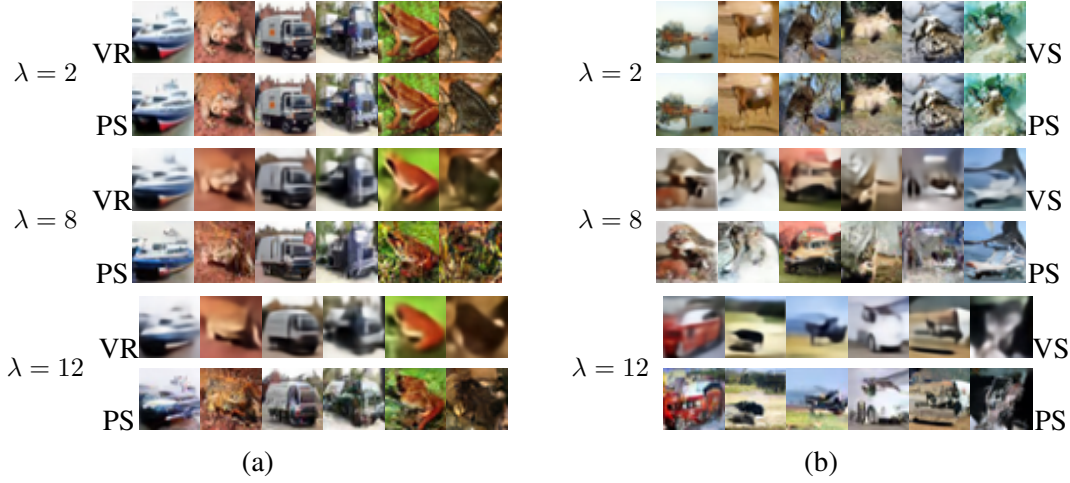


Figure 3.3: Effect of the regularization parameter λ . Reconstructions (a) and samples (b) of the VAE decoder (VR and VS, respectively) and corresponding conditional samples from the pixelCNN (PS).

For the pixelCNN we use the architecture of [Salimans et al. \[2017a\]](#), and modify it to be conditioned on the VAE decoder output $f(z)$, or possibly an upsampled version if y has a lower resolution than x . In particular, we apply standard non-masked convolutional layers to the VAE output, as many as there are pixelCNN layers. We allow each layer of the pixel-CNN to take additional input using non-masked convolutions from the feature stream based on the VAE output. This ensures that the conditional pixelCNN remains autoregressive. To speed up training, we independently pretrain the VAE and pixelCNN in parallel, and then continue training the full model with both decoders. We use the Adamax optimizer [[Kingma and Ba, 2015b](#)] with a learning rate of 0.002 without learning rate decay.

3.4.2 Quantitative performance evaluation.

Following previous work, we evaluate models on the test images using the bits-per-dimension (BPD) metric: the negative log-likelihood divided by the number of pixels values ($3 \times 32 \times 32$). It can be interpreted as the average number of bits per RGB value in a lossless compression scheme derived from the model. The comparison in Table B.4 shows that our model performs on par with the state-of-the-art results of the pixelCNN++ model [[Salimans et al., 2017a](#)]. Here we used the importance sampling-based bound of [Burda et al. \[2016\]](#) with 150 samples to compute the BPD metric for our model.¹ We refer to Figure 3.2 for qualitative comparison of samples from our model and pixelCNN++, the latter generated using the publicly available code.

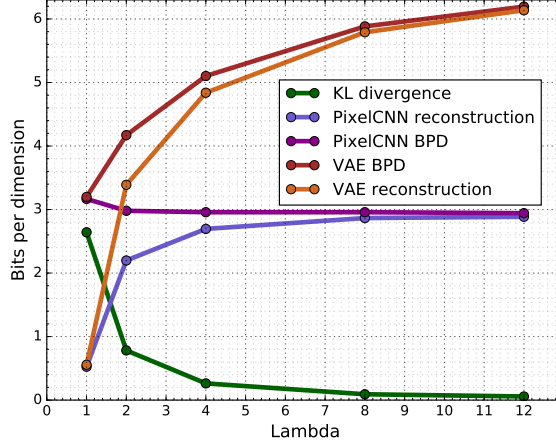


Figure 3.4: Bits per dimension of the VAE and AGAVE model, as well as decomposition in KL regularization and reconstruction terms.

3.4.3 Effect of KL regularization strength.

In Figure 3.3 we show reconstructions of test images and samples generated by the VAE decoder, together with their corresponding conditional pixelCNN samples for different values of λ . As expected, the VAE reconstructions become less accurate for larger values of λ due to the heavier weighting of the KL term, mainly by lacking details while preserving the global shape of the input. At the same time, the samples become more appealing for larger λ , suppressing the unrealistic high-frequency detail in the VAE samples obtained at lower values of λ . Note that the VAE samples and reconstructions become more similar as λ increases, which makes the input to the pixelCNN during training and sampling more consistent.

For both reconstructions and samples, the pixelCNN clearly takes into account the output of the VAE decoder, demonstrating the effectiveness of our auxiliary loss to condition high-capacity pixelCNN decoders on latent variable representations. Samples from the pixelCNN faithfully reproduce the global structure of the VAE output, leading to more realistic samples, in particular for higher values of λ . For $\lambda = 2$ the VAE reconstructions are near perfect during training, and the pixelCNN decoder does not significantly modify the appearance of the VAE output. For larger values of λ , the pixelCNN clearly adds significant detail to the VAE outputs.

Figure 3.4 traces the BPD metrics of both the VAE and pixelCNN decoder as a function of λ . We also show the decomposition in regularization and reconstruction terms. By increasing λ , the KL divergence can be pushed closer to zero. As the KL divergence term drops, the reconstruction term for the VAE rapidly increases and the VAE model obtains worse BPD values, stemming from the inability of the VAE to model pixel dependencies other than via the latent variables. The reconstruction term of the pixelCNN decoder also increases with λ , as the amount

¹ The graphs in Figure 3.4 and Figure 3.8 are based on the bound in Eq. (3.13) to reduce the computational effort.

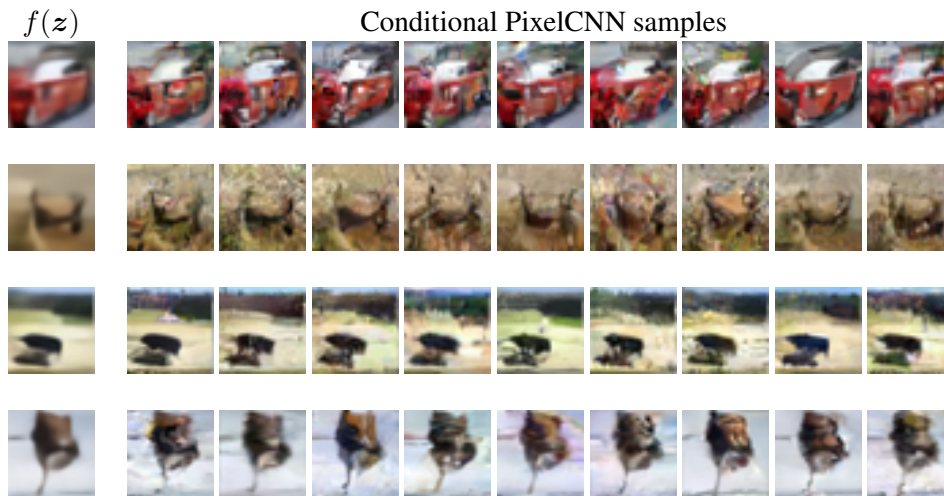


Figure 3.5: The column labeled $f(z)$ displays auxiliary representations, with z sampled from the unit Gaussian prior $p(z)$, accompanied by ten samples of the conditional pixelCNN.

of information it receives drops. However, in terms of BPD which sums KL divergence and pixelCNN reconstruction, a substantial gain of 0.2 is observed increasing λ from 1 to 2, after which smaller but consistent gains are observed.

3.4.4 Role of the auxilliary representation

The auxilliary variables are taken into account Section 3.3.2 shows that in theory it is always optimal for the autoregressive decoder to take the latent variables into account. Figure 3.5 demonstrates this empirically by displaying auxiliary representations $f(z)$ with z sampled from the prior $f(z)$ as well as nine different samples from the autoregressive decoder conditioned on $f(z)$. This qualitatively shows that the low level detail added by the pixelCNN, which is crucial for log-likelihood performance, always respects the global structure of the image being conditioned on. The VAE decoder used to generate that figure was trained with $\lambda = 8$, and in that case the KL divergence weighs very little. Yet it controls the global structure of the samples, which shows that our setup can be used to get the best of both worlds. In Figure 3.6, samples are obtained by encoding ground truth images, then interpolating the latent variables obtained, decoding them with the decoder of the VAE and adding low level detail with the pixelCNN. This demonstrates that the model has learned to use the latent variables z produced by the encoder.

The auxilliary loss is necessary: The fact that the autoregressive decoder ignores the latent variables could be attributed to optimization challenges, as explained in Section 3.3.2. In that case, the auxilliary loss could be used as an initialization scheme only, to guide the model towards a good use of the latent variables. To evaluate this we perform a control experiment where during training we first optimize our objective function in Eq. (3.13), *i.e.* including the



Figure 3.6: The first and last columns contain auxilliary reconstructions, images in between are obtained from interpolation of the corresponding latent variables. Odd rows contain auxilliary reconstructions, and even rows contain outputs of the full model.

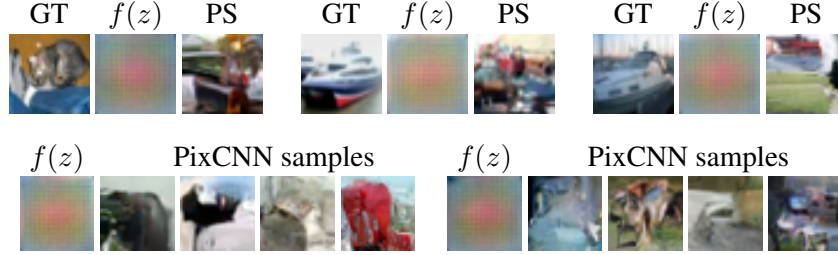


Figure 3.7: Auxiliary reconstructions obtained after dropping the auxilliary loss. (GT) denotes ground truth images unseen during training, $f(z)$ is the corresponding intermediate reconstruction, (PS) denotes pixelCNN samples, conditioned on $f(z)$.

auxiliary reconstruction term, and then switch to optimize the standard objective function of Eq. (3.6) without the auxiliary term. We proceed by training the full model to convergence then removing the auxiliary loss and fine-tuning from there. Figure 3.7 displays ground-truth images, with corresponding auxiliary reconstructions and conditional samples, as well as pure samples. The reconstructions have become meaningless and independent from the ground truth images. The samples display the same behavior: for each auxiliary representation four samples from the autoregressive component are displayed and they are independent from one another. Quantitatively, the KL cost immediately drops to zero when removing the auxiliary loss, in approximately two thousand steps of gradient descent. The approximate posterior immediately collapses to the prior and the pixel CNN samples become independent of the latent variables. This is the behavior predicted by the analysis of [Chen et al. \[2017\]](#): the autoregressive decoder is sufficiently expressive that it suffers from using the latent variables.

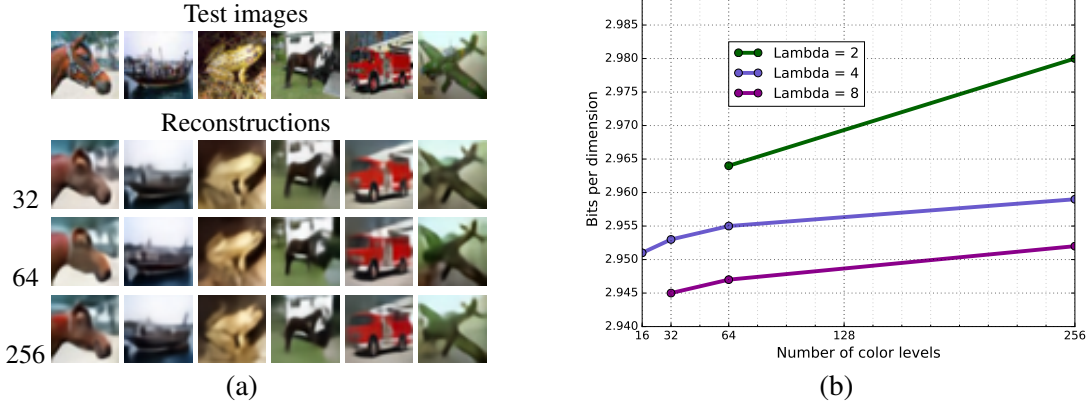


Figure 3.8: Impact of the color quantization in the auxiliary image. (a) Reconstructions of the VAE decoder for different quantization levels ($\lambda = 8$). (b) BPD as a function of the quantization level.

3.4.5 Effect of different auxiliary images.

We assess the effect of using coarser RGB quantizations, lower spatial resolutions, and grayscale in the auxiliary image. All three make the VAE reconstruction task easier, and transfer the task of modelling color nuances and/or spatial detail to the pixelCNN. The VAE reconstructions in Figure 3.8 (a) obtained using coarser colour quantization carry less detail than reconstructions based on the original images using 256 colour values, as expected. To understand the relatively small impact of the quantization level on the reconstruction, recall that the VAE decoder outputs the continuous means of the logistic distributions regardless of the quantization level. Only the reconstruction loss is impacted by the quantization level via the computation of the probabilities over the discrete color levels in Eq. (3.17). In Figure 3.8 (b) we observe small but consistent gains in the BPD metric as the number of color bins is reduced, showing that it is more effective to model color nuances using the pixelCNN, rather than the latent variables. We trained models with auxiliary images down-sampled to 16×16 and 8×8 pixels, which yield 2.94 and 2.93 BPD, respectively. This is comparable to the 2.92 BPD obtained using our best model at 32×32 . We also trained models with 4-bit per pixel grayscale auxiliary images, as in Kolesnikov and Lampert [2017]. While the grayscale auxiliary images are subjectively the ones that have the best global structure, the results are still qualitatively inferior to those obtained by Kolesnikov and Lampert [2017] with a pixelCNN modelling grayscale images. Our model does, however, achieve better quantitative performance at 2.93 BPD. In Figure 3.9 (a) we show samples obtained using models trained with 4-bit per pixel grayscale auxiliary images, in Figure 3.9 (b) with 32 color levels in the auxiliary image, and in Figure 3.9 (c) and (d) with auxiliary images of size 16×16 and 8×8 . The samples are qualitatively comparable, showing that in all cases the pixelCNN is able to compensate the less detailed outputs of the VAE decoder and that our framework can be used with a variety of intermediate reconstruction losses.



Figure 3.9: Samples from models trained with grayscale auxiliary images with 16 color levels (a), 32×32 auxiliary images with 32 color levels (b), and at reduced resolutions of 16×16 (c) and 8×8 pixels (d) with 256 color levels. For each model the auxiliary representation $f(z)$, with z sampled from the prior, is displayed above the corresponding conditional pixelCNN sample.

3.5 Conclusion

We presented a new approach to training generative image models that combine a latent variable structure with an autoregressive model component. Unlike prior approaches, it does not require careful architecture design to trade-off how much is modelled by latent variables and the autoregressive decoder. Instead, this trade-off can be controlled using a regularization parameter and choice of auxiliary target images. We obtain quantitative performance on par with the state of the art on CIFAR10, and samples from our model exhibit globally coherent structure as well as fine details. The construction proposed goes beyond the conditional independence assumption, alleviating some of its limitations as presented in Section 2.9.1. It does so at the cost of slow, sequential sampling and so this construction cannot be used in an adversarial setting. In Chapter 5, another approach to go beyond conditional independence which is compatible with adversarial training is presented.

Chapter 4

Mixed batches and symmetric discriminators for GAN training

Contents

4.1 Outline of this chapter	66
4.2 Related work	68
4.3 Adversarial learning with permutation-invariant batch features	69
4.4 Permutation invariant networks	72
4.5 Experiments	76
4.6 Conclusion	80
4.7 Optimal discriminator for general beta prior (*)	81
4.8 Universal approximation theorem for symmetric functions (*)	85

Generative adversarial networks (GANs) are powerful generative models based on providing feedback to a generative network via a discriminator network. However, the discriminator usually assesses individual samples. This prevents the discriminator from accessing global distributional statistics of generated samples, and often leads the generator to model only part of the target distribution. This phenomenon is known as mode-dropping. In this chapter, we propose to feed the discriminator with *batches* that mix both true and fake samples, and train it to predict the ratio of true samples in the batch. The latter score does not depend on the order of samples in a batch. Rather than learning this invariance, we introduce a generic permutation-invariant discriminator architecture. This architecture is provably a universal approximator of all symmetric functions. Experimentally, our approach reduces mode collapse in GANs on two synthetic datasets, and obtains good results on the CIFAR10 and CelebA datasets, both qualitatively and quantitatively. The material presented in this chapter is based on the paper "Mixed batches and symmetric discriminators for GAN training", Thomas Lucas, Corentin Tallec, Jakob Verbeek and Yann Ollivier, International Conference on Machine Learning (ICML) 2018.

4.1 Outline of this chapter

Several approaches relying on latent variables have been proposed to learn flexible density estimators together with efficient sampling such as generative adversarial networks (GANs) [Goodfellow et al., 2014], variational auto-encoders [Kingma and Welling, 2014a, Rezende et al., 2014], iterative transformation of simple distributions [Sohl-Dickstein et al., 2015], or non-volume preserving transformations [Dinh et al., 2017]. In this chapter we focus on GANs, currently the generative model that produces the most convincing natural image samples [Karras et al., 2018]. GANs consist of a generator and a discriminator network. The generator maps samples from a latent random variable with a basic prior, such as a multivariate Gaussian, to the observation space. This defines a probability distribution over the observation space. A discriminator network is trained to distinguish between generated samples and true samples in the observation space. The generator, on the other hand, is trained to fool the discriminator. In an idealized setting with unbounded capacity of both networks and infinite training data, the generator should converge to the distribution from which the training data has been sampled.

In most adversarial setups, the discriminator classifies individual data samples. Consequently, it cannot directly detect discrepancies between the *distribution* of generated samples and global statistics of the training distribution, such as its moments or quantiles. For instance, if the generator models a restricted part of the support of the target distribution very well, this can fool the discriminator at the level of individual samples, a phenomenon known as *mode dropping*. In such a case there is little incentive for the generator to model other parts of the support of the target distribution. A more thorough explanation of this effect is described in Salimans et al. [2016a] and in Section 2.9.2. In order to access global distributional statistics, imagine a discriminator that could somehow take full probability distributions as its input. This is impossible in practice. Still, it is possible to feed large batches of training or generated samples to the discriminator, as an approximation of the corresponding distributions. The discriminator can compute statistics on those batches and detect discrepancies between the two distributions. For instance, if a large batch exhibits only one mode from a multimodal distribution, the discriminator would notice the discrepancy. Even though a single batch may not encompass all modes of the distribution, it will still convey more information about missing modes than individual samples.

Training the discriminator to discriminate “pure” batches with only real or only synthetic samples makes its task too easy, as a single bad sample reveals the whole batch as synthetic. Instead, we introduce a “mixed” batch discrimination task in which the discriminator needs to predict the ratio of real samples in a batch. This use of batches differs from traditional minibatch learning. The batch is not used as a computational trick to increase parallelism, but as an approximate distribution, on which to compute global statistics. A naive way of doing so would be to concatenate the samples in the batch, feeding the discriminator a single tensor containing all the samples. However, this is parameter-hungry, and the computed statistics are not automatically invariant to the order of samples in the batch. To compute functions that depend on the samples only through their distribution, it is necessary to restrict the class of discriminator networks to *permutation-invariant* functions of the batch. For this, we adapt and extend the architecture

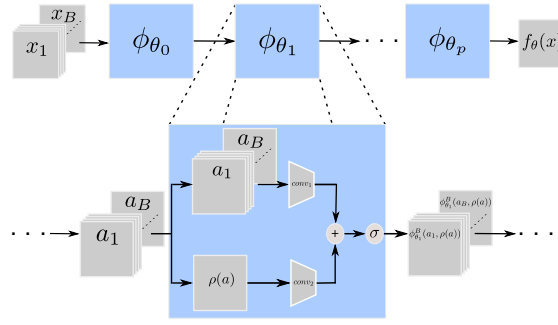


Figure 4.1: Graphical representation of our discriminator architecture. Each convolutional layer of an otherwise classical CNN architecture is modified to include permutation invariant batch statistics, denoted $\rho(x)$. This is repeated at every layer so that the network gradually builds up more complex statistics.

of [McGregor](#), also found in [Zaheer et al. \[2017\]](#) in the context of deep learning, to compute symmetric functions of the input. We show this can be done with minimal modification to existing architectures, at a negligible computational overhead w.r.t. ordinary batch processing.

Discriminating between distributions at the batch level provides an equally principled alternative to approaches to GANs based on duality formulas [[Arjovsky et al., 2017](#), [Gulrajani et al., 2017b](#), [Nowozin et al., 2016](#)]. Section [B.2](#) covers related work, followed by a presentation of the proposed training procedure in Section [4.3](#). Section [4.4](#), describes how to build networks that have the required permutation invariance properties, and experimental results are presented in Section [4.5](#). The last two sections of this chapter, Section [4.7](#) and Section [4.8](#), contain proofs and technical details about results used in the rest of the chapter, and the reader can decide to skip them .

In summary, the contributions presented in this chapter are the following:

- Naively training the discriminator to discriminate “pure” batches with only real or only synthetic samples makes its task way too easy. A discrimination loss based on *mixed* batches of true and fake samples, that avoids this pitfall is presented. We also derive the associated optimal discriminator.
- A principled way of defining neural networks that are permutation-invariant over a batch of samples is provided. We formally prove that the resulting class of functions comprises all symmetric continuous functions, and only symmetric functions.
- We apply these insights to GANs, with good experimental results, both qualitatively and quantitatively.

4.2 Related work

The training of generative models via distributional rather than pointwise information has been explored in several recent contributions. Batch discrimination [Salimans et al., 2016a] uses a specially designed layer to compute batch statistics which are then combined with sample-specific features to enhance individual sample discrimination. Karras et al. [2018] directly compute the standard deviation of features and feed it as an additional feature to the last layer of the network. Both methods use a single layer of handcrafted batch statistics, instead of letting the discriminator learn arbitrary batch statistics useful for discrimination as in our approach. Moreover, in both methods the discriminator still assesses single samples, rather than entire batches. Radford et al. [2016] reported improved results with batch normalization in the discriminator, which may also be due to reliance on batch statistics.

Other works, such as Li et al. [2015] and Dziugaite et al. [2015], replace the discriminator with a fixed distributional loss between true and generated samples, the *maximum mean discrepancy*, as the criterion to train the generative model. This has the advantage of relieving the inherent instability of GANs, but lacks the flexibility of an adaptive discriminator. The discriminator we introduce treats batches as *sets* of samples. Processing sets prescribes the use of permutation invariant networks. There has been a large body of work around permutation invariant networks, e.g. McGregor, 2008], Qi et al. [2016], Vaswani et al. [2017a], Zaheer et al. [2017]. Our processing is inspired by McGregor, 2008] which designs a special kind of layer that provides the desired invariance property. The network from McGregor is a multi-layer perceptron in which the single hidden layer performs a batchwise computation that makes the result equivariant by permutation. Here we show that stacking such hidden layers and reducing the final layer with a permutation invariant reduction, covers the entire space of continuous permutation invariant functions.

Zaheer et al. [2017] first process each element of the set independently, then aggregate the resulting representation using a permutation invariant operation, and finally process the permutation invariant quantity. Qi et al. [2016] process 3D point cloud data, and interleave layers that process points independently, and layers that apply equivariant transformations. The output of their networks are either permutation equivariant for pointcloud segmentation, or permutation invariant for shape recognition. In our approach we stack permutation equivariant layers that combine batch information and sample information at every level, and aggregate these in the final layer using a permutation invariant operation. More complex approaches to permutation invariance or equivariance appear in [Guttenberg et al.]. We prove, however, that our simpler architecture already covers the full space of permutation invariant functions.

Improving the training of GANs has received a lot of recent attention. For instance, Arjovsky et al. [2017], Gulrajani et al. [2017b] and Miyato et al. [2018b] constrain the Lipschitz constant of the network and show that this stabilizes training and improves performance. Karras et al. [2018] achieved impressive results by gradually increasing the resolution of the generated images as training progresses. These research directions, orthogonal to our work, can be adapted to work with batches of data rather than i.i.d. samples.

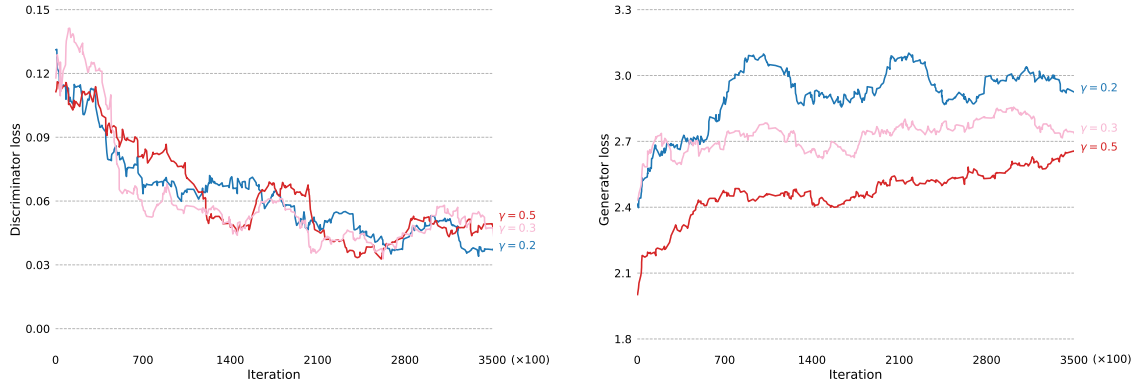


Figure 4.2: Effect of batch smoothing with different γ 's on the generator and discriminator losses.

4.3 Adversarial learning with permutation-invariant batch features

Using a batch of samples rather than individual samples as input to the discriminator can provide global statistics about the distributions of interest. Such statistics could be useful to avoid mode dropping. Adversarial learning [Goodfellow et al., 2014] can easily be extended to the batch discrimination case. For a fixed batch size B , the corresponding two-player optimization procedure becomes

$$\min_G \max_D \mathbb{E}_{\mathbf{x}_1, \dots, \mathbf{x}_B \sim \mathcal{D}} [\log D(\mathbf{x}_1, \dots, \mathbf{x}_B)] + \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_B \sim p(\mathbf{z})} [\log(1 - D(G(\mathbf{z}_1), \dots, G(\mathbf{z}_B)))] \quad (4.1)$$

with \mathcal{D} the empirical distribution over data, $p(\mathbf{z})$ a distribution over the latent variable that is the input of the generator, G a pointwise generator and D a batch discriminator.¹ This leads to a learning procedure similar to the usual GAN algorithm, except that the loss encourages the discriminator to output 1 when faced with an entire batch of real data, and 0 when faced with an entire batch of generated data.

Unfortunately, this basic procedure makes the work of the discriminator too easy. As the discriminator is only faced with batches that consist of either only training samples or only generated samples, it can base its prediction on any subset of these samples. For example, a single poorly generated sample would be enough to reject a batch. To cope with this deficiency, we propose to sample batches that mix both training and generated data. The discriminator's task is to predict the *proportion* of real images in the batch, which is clearly a permutation invariant quantity.

¹ The generator G could also be modified to produce batches of data, which can help to cover more modes per batch, but this deviates from the objective of learning a density estimator from which we can draw i.i.d. samples.

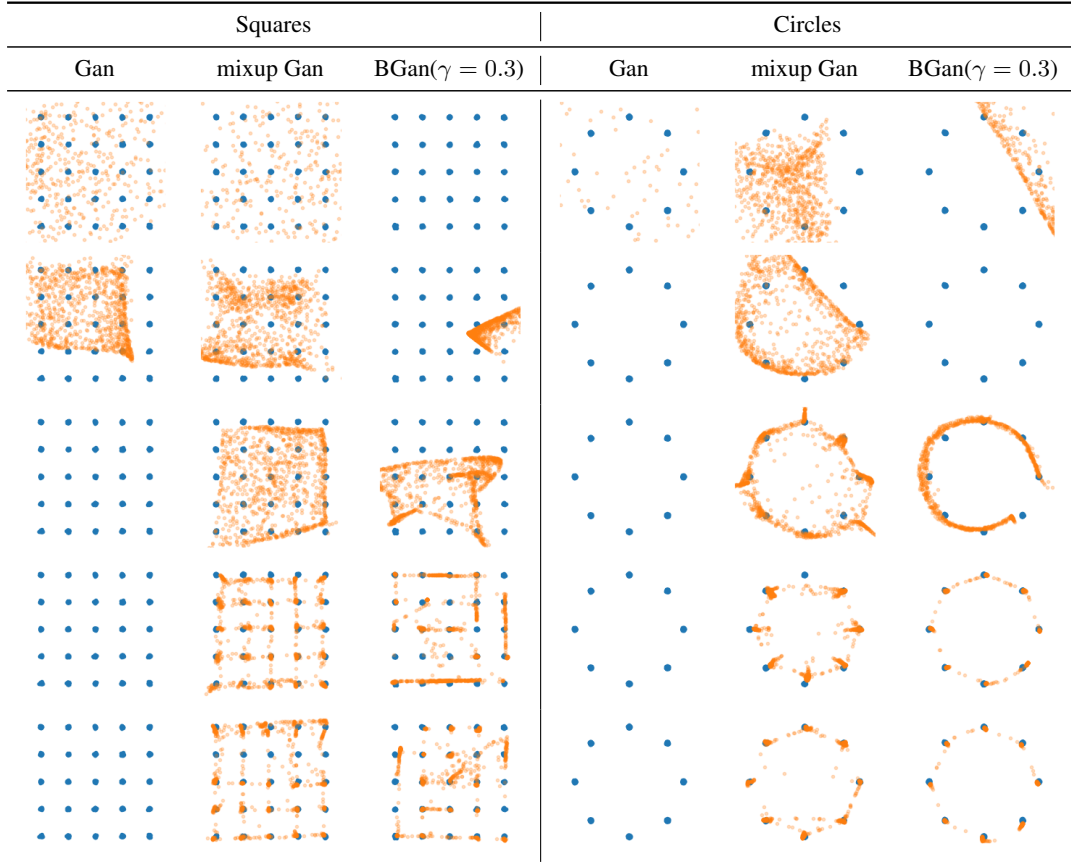


Figure 4.3: Comparison between standard, mixup and batch smoothing GANs on a 2D experiment. From top to bottom: result after training for 10, 100, 1000, 10000 and 20000 iterations.

4.3.1 Batch smoothing as a regularizer

A naive approach to sampling mixed batches would be, for each batch index, to pick a datapoint from either real or generated images with probability $\frac{1}{2}$. This is necessarily ill behaved: as the batch size increases, the ratio of training data to generated data in the batch tends to $\frac{1}{2}$ by the law of large numbers. Consequently, a discriminator always predicting $\frac{1}{2}$ would achieve very low error with large batch sizes, and provide no training signal to the generator. Instead, for each batch we sample a ratio p^* from a distribution \mathcal{P} on $[0, 1]$, and construct a batch by picking real samples with probability p^* and generated samples with probability $1 - p^*$. This forces the discriminator to predict across an entire range of possible values of p^* .

Formally, suppose we are given a batch of training data $\mathbf{x} \in \mathbb{R}^{B \times N}$ and a batch of generated data $\tilde{\mathbf{x}} \in \mathbb{R}^{B \times N}$. To mix \mathbf{x} and $\tilde{\mathbf{x}}$, a binary vector β is sampled from $\mathcal{B}(p)^B$, a B -dimensional

Bernoulli distribution with parameter p . The mixed batch with mixing vector β is denoted

$$m_\beta(\mathbf{x}, \tilde{\mathbf{x}}) := \mathbf{x} \odot \beta + \tilde{\mathbf{x}} \odot (1 - \beta) \quad (4.2)$$

where \odot denotes pointwise multiplication along the batch axis, with β being broadcasted across the other dimensions. This apparently wastes some samples, but we can reuse the discarded samples by using $1 - \beta$ in the next batch.

The discriminator has to predict the ratio of real images, $\frac{\#\beta}{B}$ where $\#\beta$ is the sum of the components of β . As a loss on the predicted ratio, we use the Kullback–Leibler divergence between a Bernoulli distribution with the actual ratio of real images, and a Bernoulli distribution with the predicted ratio. The divergence between Bernoulli distributions with parameters u and v is

$$\text{KL}(\mathcal{B}(u) \parallel \mathcal{B}(v)) = u \log \frac{u}{v} + (1 - u) \log \frac{1 - u}{1 - v}. \quad (4.3)$$

Formally, the discriminator D will minimize the objective

$$\mathbb{E}_{p^* \sim \mathcal{P}, \beta \sim \mathcal{B}(p^*)^B} \text{KL}\left(\mathcal{B}\left(\frac{\#\beta}{B}\right) \parallel \mathcal{B}(D(m_\beta(\mathbf{x}, \tilde{\mathbf{x}})))\right), \quad (4.4)$$

where the expectation is over sampling p^* from a distribution \mathcal{P} , typically uniform on $[0, 1]$, then sampling a mixed minibatch. For clarity, we have omitted the expectation over the sampling of training and generated samples. The generator is trained with the loss

$$\mathbb{E}_{p^* \sim \mathcal{P}, \beta \sim \mathcal{B}(p^*)^B} \log(D(m_\beta(\mathbf{x}, \tilde{\mathbf{x}}))). \quad (4.5)$$

This loss, which is not the generator loss associated to the min-max optimization problem, is known to saturate less [Goodfellow et al., 2014].

In some experimental cases, using the discriminator loss of Equation 4.4 with $\mathcal{P} = \mathcal{U}([0, 1])$ made discriminator training too difficult. To alleviate some of the difficulty, we sampled the mixing variable p^* from a reduced symmetric union of intervals $[0, \gamma] \cup [1 - \gamma, 1]$. With low γ , all generated batches are nearly purely taken from either real or fake data. We refer to this training method as *batch smoothing- γ* . Batch smoothing-0 corresponds to no mixing, while batch smoothing-0.5 corresponds to Equation 4.4.

4.3.2 The optimal discriminator for batch smoothing

The optimal discriminator for batch smoothing can be computed explicitly, for $p^* \sim \mathcal{U}([0, 1])$, and covers the usual GAN discriminator when $B = 1$.

Proposition 1. *The optimal discriminator for the loss in Equation 4.4, given a batch $\mathbf{y} \in \mathbb{R}^{B \times N}$, is*

$$D^*(\mathbf{y}) = \frac{1}{2} \frac{p_{\text{unbalanced}}(\mathbf{y})}{p_{\text{balanced}}(\mathbf{y})} \quad (4.6)$$

where the distribution p_{balanced} and $p_{\text{unbalanced}}$ on batches are defined as

$$\begin{aligned} p_{\text{balanced}}(\mathbf{y}) &= \frac{1}{B+1} \sum_{\beta \in \{0,1\}^B} \frac{p_1(\mathbf{y})^\beta p_2(\mathbf{y})^{1-\beta}}{\binom{B}{\#\beta}} \\ p_{\text{unbalanced}}(\mathbf{y}) &= \frac{2}{B+1} \sum_{\beta \in \{0,1\}^B} \frac{p_1(\mathbf{y})^\beta p_2(\mathbf{y})^{1-\beta}}{\binom{B}{\#\beta}} \frac{\#\beta}{B}. \end{aligned} \quad (4.7)$$

in which p_1 is the data distribution and p_2 the distribution of generated samples, and where $p_1(\mathbf{y})^\beta$ is shorthand for $p_1(\mathbf{y}_1)^{\beta_1} \dots p_1(\mathbf{y}_B)^{\beta_B}$.

For non-uniform beta distributions on p , a similar result holds, with different coefficients depending on $\#\beta$ and B in the sum. The proof is technical and is deferred to Section 4.7. These expressions can be interpreted easily. First, in the case $B = 1$, the optimal discriminator reduces to the optimal discriminator for a standard GAN, $D^* = \frac{p_1(\mathbf{y})}{p_1(\mathbf{y}) + p_2(\mathbf{y})}$. Actually $p_{\text{balanced}}(\mathbf{y})$ is simply the distribution of batches \mathbf{y} under our procedure of sampling p uniformly, then sampling $\beta \sim \mathcal{B}(p)^B$. The binomial coefficients put on equal footing contributions with different true/fake ratios.

The generator loss (4.5), when faced with the optimal discriminator, is the Kullback–Leibler divergence between p_{balanced} and $p_{\text{unbalanced}}$ (up to sign and a constant $\log(2)$). Since $p_{\text{unbalanced}}$ puts more weight on batches with higher $\#\beta$ (more true samples), this brings fake samples closer to true ones. Since p_{balanced} and $p_{\text{unbalanced}}$ differ by a factor $2\#\beta/B$, the ratio $D^* = \frac{1}{2} \frac{p_{\text{unbalanced}}(\mathbf{y})}{p_{\text{balanced}}(\mathbf{y})}$ is simply the expectation of $\#\beta/B$ under a probability distribution on β that is proportional to $\frac{p_1(\mathbf{y})^\beta p_2(\mathbf{y})^{1-\beta}}{\binom{B}{\#\beta}}$. But this is the posterior distribution on β given the batch \mathbf{y} and the uniform prior on the ratio p . Thus, the optimal discriminator is just the posterior mean of the ratio of true samples, $D^*(\mathbf{y}) = \mathbb{E}_{\beta|\mathbf{y}} \left[\frac{\#\beta}{B} \right]$. This is standard when minimizing the expected divergence between Bernoulli distributions and the approach can therefore be extended to non-uniform priors on p as shown in Section 4.7.

Note on available statistics. A reasonable handcrafted statistic to use for batch discrimination is the standard deviation. Observe that our discriminator can learn to compute it with $E[(X - E[X])^2]$, which requires 2 layers of computation. Owing to the convolutional nature of the architecture, non-pixelwise statistics are also available, (e.g. covariances).

4.4 Permutation invariant networks

Computing statistics of probability distributions from batches of i.i.d. samples requires to compute quantities that are invariant to permuting the order of samples within the batch. In this section we propose a permutation equivariant layer that can be used together with a permutation invariant aggregation operation to build networks that are permutation invariant.

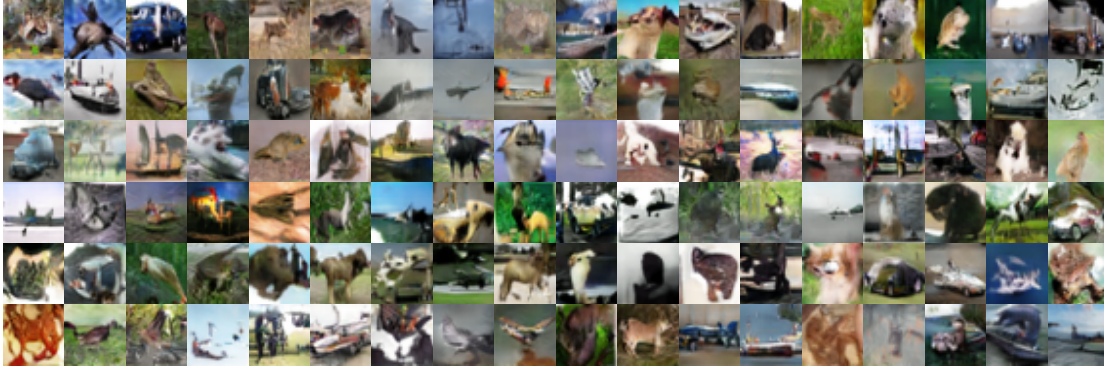


Figure 4.4: Sample images generated by our best model trained on CIFAR10.

4.4.1 Building a permutation invariant architecture

A naive way of achieving invariance to batch permutations is to consider the batch dimension as a regular feature dimension, and to randomly reorder the batches at each step. This multiplies the input dimension by the batch size, and thus greatly increases the number of trainable parameters. Moreover, this only provides approximate invariance to batch permutation, as the network has to infer the invariance based on the training data. Instead, we propose to directly build invariance into the architecture. This method drastically reduces the number of parameters compared to the naive approach, bringing it back in line with ordinary networks, and ensures strict invariance to batch permutation.

Let us first formalize the notion of batch permutation invariance and equivariance. A function f from $\mathbb{R}^{B \times l}$ to $\mathbb{R}^{B \times L}$ is *batch permutation equivariant* if permuting samples in the batch results in the same permutation of the outputs: for any permutation σ of the inputs,

$$f(\mathbf{x}_{\sigma(1)}, \dots, \mathbf{x}_{\sigma(B)}) = f(\mathbf{x})_{\sigma(1)}, \dots, f(\mathbf{x})_{\sigma(B)}. \quad (4.8)$$

For instance, any regular neural network or other function treating the inputs $\mathbf{x}_1, \dots, \mathbf{x}_B$ independently in parallel, is batch permutation equivariant.

A function f from $\mathbb{R}^{B \times l}$ to \mathbb{R}^L is *batch permutation invariant* if permuting the inputs in the batch does not change the output: for any permutation on batch indices σ ,

$$f(\mathbf{x}_{\sigma(1)}, \dots, \mathbf{x}_{\sigma(B)}) = f(\mathbf{x}_1, \dots, \mathbf{x}_B). \quad (4.9)$$

The mean, the maximum or the standard deviation along the batch axis are all batch permutation invariant. Permutation equivariant and permutation invariant functions can be obtained by combining ordinary, parallel treatment of batch samples with an additional *batch-averaging* operation that performs an average of the activations across the batch direction. In our architecture, this averaging is the only form of interaction between different elements of the batch. It is one of our



Figure 4.5: Samples obtained after 66000 iterations on the celebA dataset. From left to right: (a) Standard GAN (b) BGAN, no batch smoothing. (c) BGAN, batch smoothing $\gamma = 0.5$. (d) M-BGAN, batch smoothing $\gamma = 0.5$

main results that such operations are sufficient to recover all invariant functions.

Formally, on a batch of data $\mathbf{x} \in \mathbb{R}^{B \times N}$, our proposed batch permutation invariant network f_θ is defined as

$$f_\theta(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B (\phi_{\theta_p} \circ \phi_{\theta_{p-1}} \circ \dots \circ \phi_{\theta_0}(\mathbf{x}))_b \quad (4.10)$$

where each ϕ_{θ_i} is a batch permutation equivariant function from $\mathbb{R}^{B \times l_{i-1}}$ to $\mathbb{R}^{B \times l_i}$, where the l_i 's are the layer sizes.

The equivariant layer operation ϕ_θ with l input features and L output features comprises an ordinary weight matrix $\Lambda \in \mathbb{R}^{l \times L}$ that treats each data point of the batch independently (“non-batch-mixing”), a batch-mixing weight matrix $\Gamma \in \mathbb{R}^{l \times L}$, and a bias vector $\beta \in \mathbb{R}^L$. As in regular neural networks, Λ processes each data point in the batch independently. On the other hand, the weight matrix Γ operates after computing an average across the whole batch. Defining ρ as the batch average for each feature,

$$\rho(\mathbf{x}_1, \dots, \mathbf{x}_B) := \frac{1}{B} \sum_{b=1}^B \mathbf{x}_b \quad (4.11)$$

the permutation-equivariant layer ϕ is formally defined as

$$\phi_\theta(\mathbf{x})_b := \mu\left(\beta + \mathbf{x}_b \Lambda + \rho(\mathbf{x}) \Gamma\right) \quad (4.12)$$

where μ is a non-linearity, b is a batch index, and the parameter of the layer is $\theta = (\beta, \Lambda, \Gamma)$.

4.4.2 A universal approximation theorem for permutation invariant functions

The networks constructed above are permutation invariant by construction. However, it is unclear a priori that all permutation invariant functions can be represented this way: the functions that

can be approximated to arbitrary precision by those networks could be a strict subset of the set of permutation invariant functions. The optimal solution for the discriminator could lie outside this subset, making our construction too restrictive. We now show this is not the case: our architecture satisfies a universal approximation theorem for permutation-invariant functions.

Theorem 1. *The set of networks that can be constructed by stacking as in Eq. (4.10) the layers ϕ defined in Eq. (4.12), with sigmoid nonlinearities except on the output layer, is dense in the set of permutation-invariant functions (for the topology of uniform convergence on compact sets).*

The standard universal approximation theorem for neural networks proves the following: for any continuous function f , we can find a network that given a batch $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_B)$, computes $(f(\mathbf{x}_1), \dots, f(\mathbf{x}_B))$. This is insufficient for our purpose as it provides no way of mixing information between samples in the batch, and we need extend that theorem. The proof is restricted to sigmoid nonlinearities here, for simplicity. It can easily be extended to other types of nonlinearities that yield universal function approximators.

To describe the set of functions that can be approximated by our construction, some structure on that set is needed. For instance, suppose two functions f_1 and f_2 can be approximated, then can we approximate $f_1 + f_2$? Intuitively, if D_1 and D_2 are networks that approximate f_1 and f_2 respectively, then $D_1 + D_2$, summed using a linear operator, should approximate $f_1 + f_2$. We will then need similar results for multiplication by a scalar, and for products. More precisely, we will prove that the set of functions that can be approximated to arbitrary precision by our networks is an algebra, *i.e.*, a vector space stable under products. With this structure, it is possible to go from simple symmetric functions to complex ones. To show that *any* symmetric function can be approximated, we then need to prove that the set of functions we can approximate contains a family of functions that can be extended, using the algebra structure, to all continuous symmetric functions. Precisely, we must show that our algebra contains a generative family of the continuous symmetric functions.

While the case of one-dimensional features is relatively simple, the multidimensional case is more intricate, and the detailed proof is given in Section 4.8. Here we describe the key ideas underlying the proof, without technical details. To prove that we can compute the sum of two functions f_1 and f_2 , compute f_1 and f_2 on different channels (this is possible even if f_1 and f_2 require different numbers of layers, by filling in with the identity if necessary). Then sum across channels, which is possible in (4.12). To compute products, first compute f_1 and f_2 on different channels, then apply the universal approximation theorem to turn this into $\log f_1$ and $\log f_2$, then add, then take the exponential thanks to the universal approximation theorem again. Multiplication by a real scalar α is clearly given by linear layers. This gives us stability by scalar multiplication, sum and product, *i.e.* the algebra structure.

The key point is then the following: the algebra of all permutation-invariant *polynomials* over the components of $(\mathbf{x}_1, \dots, \mathbf{x}_B)$ is generated *as an algebra* by the averages $\frac{1}{B}(f(\mathbf{x}_1) + \dots + f(\mathbf{x}_B))$ when f ranges over all functions of single batch elements. This non-trivial algebraic statement is proved in Section 4.8. By construction, such functions $\frac{1}{B}(f(\mathbf{x}_1) + \dots + f(\mathbf{x}_B))$ are readily available in our architecture, by computing f as in an ordinary network and then applying the

batch-averaging operation ρ in the next layer. Further layers provide sums and products of those thanks to the algebra property. Having obtained the desired result for polynomial symmetric functions, we can extend it and conclude with a symmetric version of the Stone–Weierstrass theorem (polynomials are dense in continuous functions).

4.4.3 Practical architecture

In our experiments, we apply the constructions above to standard, deep convolutional neural networks. In practice, for the linear operations Λ and Γ in (4.12) we use convolutional kernels (of size 3×3) acting over x_b and $\rho(x)$ respectively. Weight tensors Λ and Γ are also reweighted like so that at the start of training $\rho(x)$ does not contribute disproportionately compared with other features: $\tilde{\Lambda} = \frac{|B|}{|B|+1}\Lambda$ and $\tilde{\Gamma} = \frac{1}{|B|+1}\Gamma$ where $|B|$ denotes the size of batch B . While these coefficients could be learned, we have found this explicit initialization to improve training. Figure 4.1 shows how to modify standard CNN architectures to adapt each layer to our method.

In the first setup, which we refer to as BGAN, a permutation invariant reduction is done at the end of the discriminator, yielding a single prediction per batch, which is evaluated with the loss in (4.4). We also introduce a setup, M-BGAN, where we swap the order of averaging and applying the loss.² Namely, letting y be the single target for the batch (in our case, the proportion of real samples), the BGAN case translates into

$$\mathcal{L}((o_1, \dots, o_B), \mathbf{y}) = \ell\left(\frac{1}{B} \sum_{i=1}^B o_i, \mathbf{y}\right) \quad (4.13)$$

while M-BGAN translates to

$$\mathcal{L}((o_1, \dots, o_B), \mathbf{y}) = \frac{1}{B} \sum_{i=1}^B \ell(o_i, \mathbf{y}) \quad (4.14)$$

where \mathcal{L} is the final loss function, ℓ is the KL loss function used in (4.4), (o_1, \dots, o_b) is the output of the last equivariant layer, and \mathbf{y} is the target for the *whole* batch. Both these losses are permutation invariant. A more detailed explanation of M-BGAN is given in Section 4.5.4.

4.5 Experiments

We now present experimental results obtained using our construction. First, results on a synthetic 2D dataset are provided on which mode-dropping can be assessed. Then results are presented on the CIFAR10 and STL10 natural images datasets. In terms of Inception score and Fréchet inception distance, our model was on par with the state of the art at the time of submission on these datasets.

² This was initially a bug that worked.

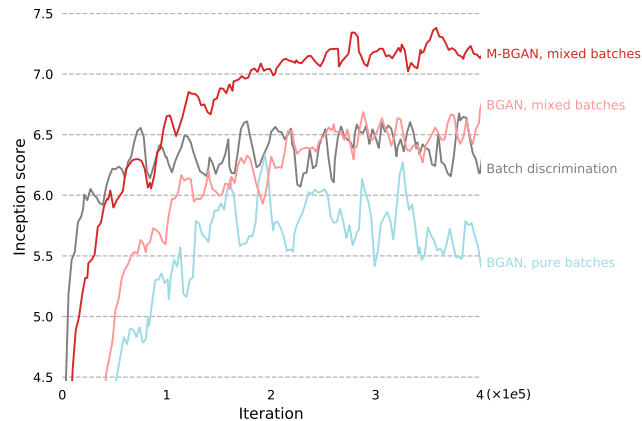


Figure 4.6: Inception score for various versions of BGAN and for batch discrimination [Salimans et al. \[2016a\]](#).

4.5.1 Synthetic 2D distributions

The synthetic dataset of [Zhang et al. \[2017\]](#) is explicitly designed to test mode dropping. The data are sampled from a mixture of concentrated Gaussians in the 2D plane. We compare standard GAN training, “mixup” training [[Zhang et al., 2017](#)], and batch smoothing using the BGAN from Section 4.4.3. In all cases, the generators and discriminators are three-layer ReLU networks with 512 units per layer. The latent variables used as input to the generator are sampled from 2-dimensional standard Gaussians. The models are trained on their respective losses using the Adam [[Kingma and Ba, 2015b](#)] optimizer, with default parameters. The discriminator is trained for five steps for each generator step.

Qualitative results are provided in Figure 4.3. Batch smoothing and mixup have similar effects. Results for BGAN and M-BGAN are qualitatively similar on this dataset and we only display results for BGAN. The standard GAN setting quickly diverges, due to its inability to fit several modes simultaneously, while both batch smoothing and mixup successfully fit the majority of modes of the distribution.

4.5.2 Experimental results on CIFAR10

Next, we consider image generation on the CIFAR10 dataset. We use the simple architecture from [Miyato et al. \[2018b\]](#), minimally modified to obtain permutation invariance thanks to Equation 4.12. All other architectural choices are unchanged. The same Adam hyper-parameters from [Miyato et al. \[2018b\]](#) are used for all models: $\alpha = 2e^{-4}$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, and no learning rate decay. We performed hyper-parameter search for the number of discrimination steps between each generation step, n_{disc} , over the range $\{1, \dots, 5\}$, and for the batch smoothing parameter γ over $[0.2, 0.5]$. All models are trained for 400,000 iterations, counting both generation and

Table 4.1: Comparison to the state of the art in terms of inception score (IS) and Fréchet inception distance (FID) on the CIFAR10 dataset.

Model	IS \uparrow	FID \downarrow
WGP Miyato et al. [2018b]	$6.68 \pm .06$	40.2
GP Miyato et al. [2018b]	$6.93 \pm .08$	37.7
SN Miyato et al. [2018b]	$7.42 \pm .08$	29.3
Models with batch statistics		
Salimans et al.	$7.09 \pm .08$	35.0
BGAN	$7.05 \pm .06$	36.5
M-BGAN	$7.49 \pm .06$	23.7

discrimination steps. We compare smoothed BGAN and M-BGAN, and the same network trained with spectral normalization Miyato et al. [2018b] (SN), and gradient penalty Gulrajani et al., 2017b] on both the Wasserstein Arjovsky et al., 2017] (WGP) and the standard loss (GP). We also compare to a model using the batch-discrimination layer of Gulrajani et al. [2017b], adding a final batch discrimination layer to the architecture of Miyato et al. [2018b]. All models are evaluated by reporting the Inception Score and the Fréchet Inception Distance Heusel et al. [2017] and results are summarized in Table B.4. Figure 4.4 displays sample images generated with our M-BGAN model.

Figure 4.6 highlights the training dynamics of each model³. On this architecture, M-BGAN heavily outperforms both batch discrimination and our other variants, and yields results similar to, or slightly better than Miyato et al. [2018b]. The model trained with batch smoothing display results on par with batch discrimination, and much better than without batch smoothing.

Effect of batch smoothing on the generator and discriminator losses. To check the effect of the batch smoothing parameter γ on the loss, we plot the discriminator and generator losses of the network for different γ 's. The smaller the γ , the purer the batches. We would expect discriminator training to be more difficult with larger γ . In Fig. 4.2) the generator loss –which is the part of the discriminator loss that evaluates samples – is lower for larger γ , revealing the relative advantage of the generator on the discriminator. BGAN and M-BGAN behave similarly and we only report on BGAN in the figure. This suggests to increase γ if the discriminator dominates learning, and to decrease γ if the discriminator is stuck at a high value in spite of poor generated samples.

4.5.3 Results on celebA and STL10 datasets

Finally, on the celebA face dataset, we adapt the simple architecture of Miyato et al. [2018b] to the increased resolution by adding a layer to both networks. For optimization we use Adam with $\beta_1 = 0$, $\beta_2 = 0.9$, $\alpha = 1e - 4$, and $n_{\text{disc}} = 1$. Fig. 4.5 displays BGAN samples with pure batches,

³ For readability, a slight smoothing is performed on the curves.

and BGAN and M-BGAN samples with $\gamma = .5$. The visual quality of the samples is reasonable; we believe that an improvement is visible from pure batches to M-BGAN. We provide results on the STL-10 dataset, in Figure 4.7 and Table 4.2 where M-BGAN yields numerical results slightly worse than Spectral Normalization. Except for the adaptation of the network to 48×48 images, as done in Miyato et al. [2018b], the experimental set-up is left unchanged.

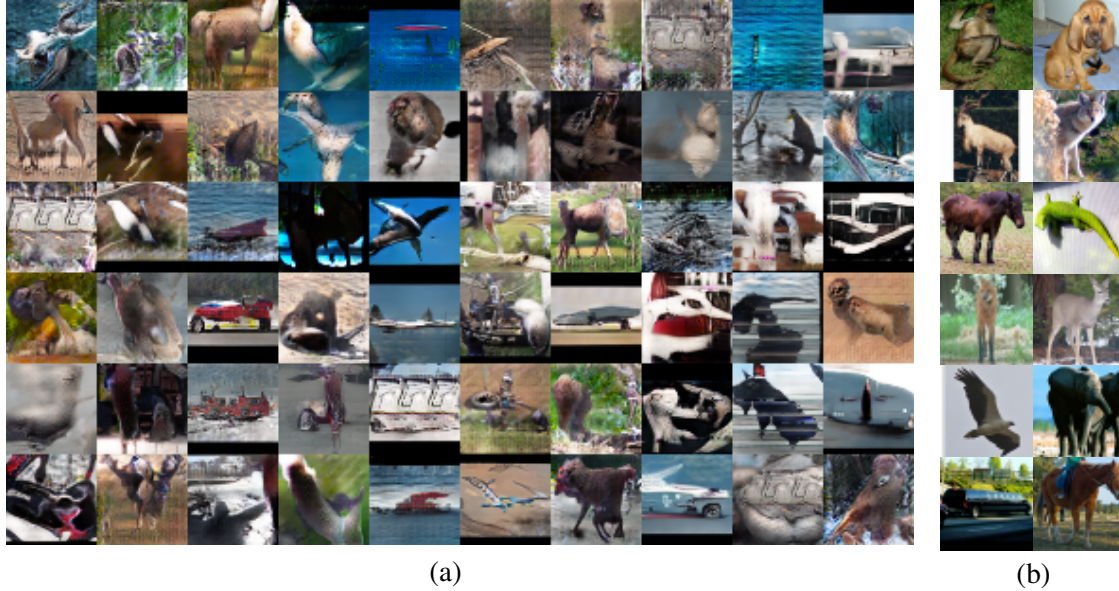


Figure 4.7: (a) Sample images generated by our best model trained on STL10. (b) real STL10 images for comparison

Table 4.2: Comparison to the state of the art in terms of inception score (IS) and Fréchet inception distance (FID) on the STL-10 dataset.

Model	IS	FID
WGP Miyato et al. [2018b]	8.4	55
M-BGAN	8.7	51
SN Miyato et al. [2018b]	8.7	47.5
SN (Hinge loss) Miyato et al. [2018b]	8.8	43.2

4.5.4 Interpretation of M-BGAN as an ensembling method

The experiments show that M-BGAN can quite significantly improve performance. Intuitively, the M-BGAN loss performs a simple ensembling of many strongly dependant permutation invariant discriminators, at no additional cost. In the general case, ensembling of N independent

discriminators D_1, \dots, D_N amounts to training each discriminator independently, and using the averaged gradient signal to train the generator. Ensembling is expected to alleviate some of the difficulties of GAN training: as long as one of the discriminators still provides a significant gradient signal, training of the generator is possible.

With equation (4.14), M-BGAN is an ensemble of B permutation invariant discriminators, with respective outputs $1\text{-th}(o_1, \dots, o_B), \dots, B\text{-th}(o_1, \dots, o_B)$, where $i\text{-th}$ is the function that returns the i -th greatest element of a B dimensional vector. Indeed,

$$\frac{1}{N} \sum_{i=1}^N l(i\text{-th}(o_1, \dots, o_B), y) = \frac{1}{N} \sum_{i=1}^N l(o_i, y). \quad (4.15)$$

which is the M-BGAN loss. The ensembled discriminators of the M-BGAN all share the same weights. This ensembling effect at least partially explains the improved performance of M-BGAN.

4.6 Conclusion

We have shown how to build neural network architectures that are, assuming sufficient capacity, dense in the set of all functions invariant to permutations on the batch axis. This means that our networks can approximate any statistics on the distributions given as input, where "approximate" comes from i) the finite size of the batch, which induces variance for the estimator of the statistic and ii) the fact that in practice the density result may require too much computation for a given statistic.

Because statistics on the *distribution* can be considered and used to reject a batch, the discriminator can explicitly evaluate variability of the images in the batch. With a perfect discriminator, a batch of samples must contain as much variability as a batch of real images, thus mode dropping is explicitly penalised together with image quality. With a perfect discriminator *and* an infinite batch size, the distributions p_θ and p^* must match exactly.

We also observe that in practice, feeding all-fake or all-genuine batches to a discriminator makes its task too easy, and therefore generalize the approach to mixed batches, without breaking the optimality results. Experimentally, this provides a new, alternative method to reduce mode dropping and reach good quantitative scores in GAN training. The vanilla GAN case is trivially recovered as a special case

4.7 Optimal discriminator for general beta prior (*)

We now take a closer look at the optimal discriminator, for a given generator. First, we give a derivation of the optimal discriminator expression when the mixing parameter p is drawn from $\text{Beta}(a, b)$. This extends the result for the uniform distribution given in Eq. (4.7), as $\text{Beta}(1, 1) = \mathcal{U}([0, 1])$. Let us first build an intuition about the result, starting from the vanilla GAN case. Given p_θ and a dataset sampled from p^* , and mixing weights between equal to π_1 and $1 - \pi_1$, one can build a dataset containing samples from p^* in proportion π_1 and from the model in proportion $1 - \pi_1$, and an image \mathbf{y}_i from this (mixed) dataset then has density

$$p_{\text{mix}}(\mathbf{y}) = \pi_1 p^*(\mathbf{y}) + (1 - \pi_1) p_\theta(\mathbf{y}).$$

Denote B the random variable modelling the class of \mathbf{y}_i , taking value $b = 0$ for fake and $b = 1$ for real. The prior on class B is then $p(B = 1) = \pi_1$, $p(B = 0) = 1 - \pi_1$. Using Bayes' rule, the *true* posterior on C is

$$p_{\text{mix}}(b|\mathbf{y}_i) = \frac{p_{\text{mix}}(\mathbf{y}_i|b)p(b)}{p_{\text{mix}}(\mathbf{y}_i)} = \frac{p_b(\mathbf{y}_i)\pi_b}{p_{\text{mix}}(\mathbf{y}_i)}.$$

This gives us a probability for each discrete class c , so values between 0 and 1, that sum to 1, *i.e.* "soft" predictions. If we need to pick a definite answer (classification), we pick the *mode* of p , *i.e.* the most likely class, by looking at which probability is greater than 0.5. This yields the optimal Bayes classifier. We already knew this result: it is exactly the optimal value of the discriminator in the standard GAN case. Indeed with $\pi_1 = 1/2$,

$$p_{\text{mix}}(B = 0|\mathbf{y}_i) = \frac{p_\theta(\mathbf{y}_i)}{p_\theta(\mathbf{y}_i) + p^*(\mathbf{y}_i)}.$$

Now, we have a probabilistic interpretation of this result: the optimal discriminator approximates the Bayes classifier, which is the true posterior. In the i.i.d case with mixed batches this is easy to apply over a batch rather than a single image: if $\beta = (b_1, \dots, b_B)$ is the class predictions for images $(\mathbf{y}_1, \dots, \mathbf{y}_B)$, then

$$p(\beta|\mathbf{y}) = \prod_i p_{\text{mix}}(b_i|\mathbf{y}_i).$$

We are interested in extending this to the non-i.i.d vector case, which is the setting of our model. Looking at the vanilla GAN case, we have the intuition that we can link the optimal discriminator to $p_{\text{mix}}(\beta|\mathbf{y})$ using Bayes' rule. It will require $p(\mathbf{y}|\beta)$, $p(\beta)$ and $p_{\text{mix}}(\mathbf{y})$. Knowing β and p_θ we can easily specify $p(\mathbf{y}|\beta)$. Knowing p_θ and the dataset provides $p(\mathbf{y})$, so we need the prior on β , $p(\beta)$, which we now derive.

Beta prior on batch mixing proportion. Consider mixed batches of samples of size B . The i -th sample of the batch is a real sample if $\beta_i = 1$ and a generated sample if $\beta_i = 0$. Given a certain mixing proportion p , assuming that samples are sampled independantly according to a Bernoulli of parameter p , the probability of a certain β is

$$\mathbb{P}(\beta | p) = \prod_i p^{\beta_i} (1 - p)^{1 - \beta_i}. \quad (4.16)$$

Consider a beta prior distribution $\text{Beta}(a, b)$ on the mixing parameter $p \in [0, 1]$. Because the β s are i.i.d, $\#\beta \sum_i \beta_i$ is a sufficient statistic and the posterior distribution $\#\beta$ contains all the "interesting" information. It is given by the beta-binomial compound distribution:

$$\mathbb{P}(\#\beta) = \int_p \text{Beta}(p | a, b) \mathbb{P}(\#\beta | p) dp \quad (4.17)$$

$$= \binom{B}{\#\beta} \frac{\mathcal{B}(\#\beta + a, B - \#\beta + b)}{\mathcal{B}(a, b)}, \quad (4.18)$$

where $\mathcal{B}(\cdot, \cdot)$ is the beta function. For $a = 1, b = 1$, i.e. a uniform distribution on mixing parameters, the beta-binomial compound distribution reduces to a uniform distribution on β . Because for a given $\#\beta$, all vectors β are equally likely and there are $\binom{B}{\#\beta}$ of them, it follows from the expression of $\mathbb{P}(\#\beta)$ that

$$\mathbb{P}(\beta) = \frac{\mathcal{B}(\#\beta + a, B - \#\beta + b)}{\mathcal{B}(a, b)}. \quad (4.19)$$

Now that we have an explicit expression for $\mathbb{P}(\beta)$, we need the value of the optimal discriminator. As in the vanilla case, we can obtain it by differentiating the B-GAN loss.

Optimal discriminator. Let $\mathbf{y} = m_\beta(\mathbf{x}, \tilde{\mathbf{x}})$ denote a mixed batch of samples. The discriminator minimizes the KL divergence between $D(\mathbf{y})$ and β , averaged over batches and mixing vectors β , see Equation 4.4. This reduces to minimizing the expected cross-entropy. For a given batch and mixing vector β ,

$$L(D(\mathbf{y}), \#\beta) = -\frac{\#\beta}{B} \ln D(\mathbf{y}) - \frac{B - \#\beta}{B} \ln(1 - D(\mathbf{y})).$$

Averaging over batches and mixing vectors,

$$\begin{aligned} \mathbb{E}_{\beta, \mathbf{y}}[L(D(\mathbf{y}), \#\beta)] &= \int_{\mathbf{y}} \mathbb{P}(\mathbf{y}) \sum_{\beta} \mathbb{P}(\beta | \mathbf{y}) L(D(\mathbf{y}), \#\beta) \\ &= - \int_{\mathbf{y}} \mathbb{P}(\mathbf{y}) \left[\mathbb{E}_{\beta | \mathbf{y}} \left[\frac{\#\beta}{B} \right] \ln D(\mathbf{y}) + \mathbb{E}_{\beta | \mathbf{y}} \left[\frac{B - \#\beta}{B} \right] \ln(1 - D(\mathbf{y})) \right] \end{aligned}$$

From the latter it follows that for any \mathbf{y} , the optimal discriminator value $D^*(\mathbf{y})$ is

$$D^*(\mathbf{y}) = \mathbb{E}_{\beta | \mathbf{y}} \left[\frac{\#\beta}{B} \right], \quad (4.20)$$

i.e. the posterior expectation of the fraction of training samples in the batch. This expression of the optimal discriminator is very compact, and reasonable intuitively. However it is too generic: it is an expression over the vectors β and \mathbf{y} and hides the fact that in our case the elements are generated in a i.i.d. manner. We now express the loss in terms of each element of the batch, to better understand how the discriminator acts on *each* \mathbf{y}_i . This gives more insight on how the generator is trained because the generator is i.i.d. The expressions become less compact, precisely because the mapping from elements \mathbf{y} to answer β is not factorized, but it will be manageable because of the invariance to permutations. We proceed by applying Bayes' rule.

Posterior analysis. Through Bayes rule, the posterior expectation yields

$$D^*(\mathbf{y}) = \mathbb{E}_{\beta|\mathbf{y}} \left[\frac{\#\beta}{B} \right] = \frac{\sum_{\beta} \frac{\#\beta}{B} \mathbb{P}(\mathbf{y}|\beta) \mathbb{P}(\beta)}{\mathbb{P}(\mathbf{y})}. \quad (4.21)$$

The marginal on the batch \mathbf{y} is

$$\mathbb{P}(\mathbf{y}) = \sum_{\beta} \mathbb{P}(\mathbf{y} | \beta) \mathbb{P}(\beta) \quad (4.22)$$

$$= \sum_{\beta} \mathbb{P}(\mathbf{y}|\beta) \frac{\mathcal{B}(\#\beta + a, B - \#\beta + b)}{\mathcal{B}(a, b)}. \quad (4.23)$$

The numerator in Eq. (4.21) can be written as a distribution on \mathbf{y} ,

$$\mathbb{Q}(\mathbf{y}) = \sum_{\beta} \mathbb{P}(\mathbf{y}|\beta) \mathbb{Q}(\beta) \quad (4.24)$$

$$\mathbb{Q}(\beta) = \frac{a+b}{a} \mathbb{P}(\beta) \frac{\#\beta}{B}. \quad (4.25)$$

The distribution $\mathbb{Q}(\beta)$ sums to 1, as $\mathbb{E}_{\mathbb{P}(\beta)}[\#\beta] = \frac{Ba}{a+b}$.

This finally yields

$$D^*(\mathbf{y}) = \frac{a}{a+b} \frac{\mathbb{Q}(\mathbf{y})}{\mathbb{P}(\mathbf{y})}, \quad (4.26)$$

which for the uniform beta prior with $a = b = 1$ on p simplifies to

$$D^*(\mathbf{y}) = \frac{1}{2} \frac{\mathbb{Q}(\mathbf{y})}{\mathbb{P}(\mathbf{y})}. \quad (4.27)$$

Expressing $\mathbb{P}(\mathbf{y} | \beta)$. Notice that $m_{\beta}(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbf{y}$ is equivalent to $\forall i \in \{1, \dots, B\}, x_i = y_i$ and $\beta_i = 1$ or $\tilde{x}_i = y_i$ and $\beta_i = 0$. Denote by p_1 (resp. p_2) the distribution of real samples (resp. generated samples).

From the previous observation, it yields that

$$\mathbb{P}(\mathbf{y} | \beta) = \prod_{i=1}^B p_1(\mathbf{y}_i)^{\beta_i} p_2(\mathbf{y}_i)^{1-\beta_i}. \quad (4.28)$$

From the latter and Eq. (4.27) we obtain the optimal discriminator expression in Equation 4.7.

$$\begin{aligned} p_{\text{balanced}}(\mathbf{y}) &= \frac{1}{B+1} \sum_{\beta \in \{0,1\}^B} \frac{p_1(\mathbf{y})^{\beta} p_2(\mathbf{y})^{1-\beta}}{\binom{B}{\#\beta}} \\ p_{\text{unbalanced}}(\mathbf{y}) &= \frac{2}{B+1} \sum_{\beta \in \{0,1\}^B} \frac{p_1(\mathbf{y})^{\beta} p_2(\mathbf{y})^{1-\beta}}{\binom{B}{\#\beta}} \frac{\#\beta}{B}. \end{aligned} \quad (4.29)$$

The next section provides further analysis of these quantities.

4.7.1 p_{balanced} and $p_{\text{unbalanced}}$ are well normalized:

We now show that $p_{\text{unbalanced}}$ is well defined, in the sense that it is normalized. The computation for p_{balanced} is almost identical and left to the reader. First, recall that

$$\int_{\mathbf{y}} p_{\text{unbalanced}}(\mathbf{y}) d\mathbf{y} = \frac{2}{B+1} \sum_{\beta \in \{0,1\}^B} \frac{\#\beta}{BC_B^{\#\beta}} \int_{\mathbf{y}} p_x(\mathbf{y})^\beta p_{\tilde{x}}(\mathbf{y})^{1-\beta} d\mathbf{y} \quad (4.30)$$

Let us take a close look at the integral $\int_{\mathbf{y}} p_x(\mathbf{y})^\beta p_{\tilde{x}}(\mathbf{y})^{1-\beta} d\mathbf{y}$. Recall that $p_x(\mathbf{y})^\beta$ is a shorthand for $p_x(y_1)^{\beta_1} \dots p_x(y_B)^{\beta_B}$, so

$$\int_{\mathbf{y}} p_x(\mathbf{y})^\beta p_{\tilde{x}}(\mathbf{y})^{1-\beta} d\mathbf{y} = \prod_i \int_{y_i} p_x(y_i)^{\beta_i} p_{\tilde{x}}(y_i)^{1-\beta_i} dy_i.$$

Each β_i is either 0 or 1 so each term of the product is the integral of either p_x or $p_{\tilde{x}}$ which are densities. Therefore the integral resolves to 1. We are now ready to solve Equation 4.30:

$$\begin{aligned} \int_{\mathbf{y}} p_{\text{unbalanced}}(\mathbf{y}) d\mathbf{y} &= \frac{2}{B+1} \sum_{\beta \in \{0,1\}^B} \frac{\#\beta}{BC_B^{\#\beta}} \int_{\mathbf{y}} p_x(\mathbf{y})^\beta p_{\tilde{x}}(\mathbf{y})^{1-\beta} d\mathbf{y} \\ &= \frac{2}{B+1} \sum_{\#\beta=1}^B C_B^{\#\beta} \frac{\#\beta}{BC_B^{\#\beta}} \\ &= \frac{2}{(B+1)B} \sum_{\#\beta=1}^B \#\beta \\ &= \frac{2}{(B+1)B} \frac{B(B+1)}{2} \\ &= 1 \end{aligned}$$

Where line two is obtained by remarking that there are $C_B^{\#\beta}$ vectors β of cardinal $\#\beta$. This shows that $p_{\text{unbalanced}}$ is indeed a valid density over \mathbf{y} .

Let us now comment on how $p_{\text{unbalanced}}$ behaves. Suppose a fixed label vector β is given, and we want to evaluate how well it describes batch \mathbf{x} . The term in the sum can be interpreted as a score for β , computed over the batch. Every time an image in the batch \mathbf{x}_i looks fake ($p_x(\mathbf{y}_i)$ is low) but β_i is a "real" label, the score decreases, or similarly if the image looks "real" but the label is 0. These per-image scores are multiplied together, so a single mistake, *i.e.* a single value close to 0, will strongly impact the product of scores of the fixed β . Therefore when summing over all β s, most of the mass will be concentrated on values of β that fit all points well.

The extra term in $p_{\text{unbalanced}}$, $\frac{\#\beta}{BC_B^{\#\beta}}$, is very important: it reweights the scores to penalise mistakes in the case where $\#\beta$ is high more, and thus brings fake samples closer to true ones. It is the only term left *inside* the posterior expectation in Equation 4.20.

4.8 Universal approximation theorem for symmetric functions (*)

In what follows, we aim at proving a universal approximation theorem for the class of permutation invariant neural networks we have defined. To ease readings, products, sums and real function applications are assumed to be broadcasted when need be. Throughout the paper the batch dimension n is constant and omitted from set indices. We begin with precise definitions, and there are then two main steps in the proof. The first is to show that the set of functions that can be approximated to arbitrary precision on a compact set K is an Algebra for some triplet $(+, \cdot, \times)$, and then to show that this algebra contains a generative family of the set of permutation invariant functions.

4.8.1 Definitions

We begin by defining precisely what a *symmetric function* is, which you can read as "invariant to permutations on the batch axis". We then define permutation-equivariant functions. Finally, we define how to build the networks in our construction; we start with equivariant networks, then reduce them to obtain invariant networks.

Definition 1. (Symmetric functions) A function $f: \mathbb{R}^{n \times k} \mapsto \mathbb{R}^l$ is symmetric if for any permutation of indexes σ and for all $x \in \mathbb{R}^{n \times k}$, $f(x_{\sigma(1)}, \dots, x_{\sigma(n)}) = f(x_1, \dots, x_n)$. The set of continuous symmetric functions from $\mathbb{R}^{n \times k}$ to \mathbb{R}^l is denoted by \mathcal{I}_k^l .

Definition 2. (Permutation equivariance) A function $f: \mathbb{R}^{n \times k} \mapsto \mathbb{R}^{n \times l}$ is permutation equivariant if for any permutation of indexes σ and for all $x \in \mathbb{R}^n$, $f(x_{\sigma(1)}, \dots, x_{\sigma(n)}) = f(x)_{\sigma(1)}, \dots, f(x)_{\sigma(n)}$.

The two previous definitions can be summarized by the following mnemonics: i) symmetric functions give the same result for any ordering of their inputs; ii) permutation equivariant functions give the same result, up to re-ordering of their outputs, for any ordering of their inputs. When symmetric functions and permutation equivariant functions are restricted to a compact K , we assume that the compact itself is symmetric, i.e.:

$$(x_1, \dots, x_n) \in K \implies \forall \sigma, (x_{\sigma(1)}, \dots, x_{\sigma(n)}) \in K.$$

Note that a classical network, which treats a batch of inputs in an i.i.d manner, is a trivial case of permutation equivariant function: its outputs are re-ordered when a permutation is applied to the batch. Typically, a loss function is then averaged over the batch, such that the final loss becomes *invariant*. So classical networks, with classical losses are already invariant. However, they are a trivial case that we wish to extend. In what follows, we use ρ as a reducing operator on vectors defined for $x \in \mathbb{R}^{n \times k}$ by

$$\rho(x)_j = \frac{1}{n} \sum_{i=1}^n x_{i,j}. \quad (4.31)$$

Definition 3. (Recursive definition of equivariant networks) Let the sets E_k^l be sets that contain permutation equivariant neural networks from $\mathbb{R}^{n \times k}$ to $\mathbb{R}^{n \times l}$, recursively defined thus:

- For all $k \in \mathbb{N}$, the identity function on $\mathbb{R}^{n \times k}$ belongs to E_k^k .

- For all $f \in E_r^k$, $\Gamma \in \mathbb{R}^{l \times k}$, $\Lambda \in \mathbb{R}^{l \times k}$ and $\beta \in \mathbb{R}^l$, and for act , a sigmoid activation function, g defined as

$$g(x)_{i,j} = \sum_{p=1}^k \Gamma_{j,p} \text{act}(f(x))_{i,p} + \sum_{p=1}^k \Lambda_{j,p} \rho(\text{act}(f(x)))_p + \beta_j \quad (4.32)$$

is in E_r^l .

The number of layers of the network is defined as the induction depth of the previous construction. The set of thus constructed permutation equivariant neural networks with number of layers L is denoted by $E(L)_k^l$. Note that this class of function is trivially stable by composition, i.e. if $g_1 \in E_{l_1}^{l_2}$ and $g_2 \in E_{l_2}^{l_3}$, the $g_2 \circ g_1 \in E_{l_1}^{l_3}$.

Definition 4. (Symmetric neural networks) Let I_k^l be a set containing symmetric neural networks from $\mathbb{R}^{n \times k}$ to \mathbb{R}^l defined as

$$I_k^l = \rho(E_k^l). \quad (4.33)$$

Definition 3 can be thought of as a formal description of how our networks are built. If two sub-networks that are permutation equivariant are given, they can be combined using our permutation equivariant layer. The second definition simply says that taking a permutation equivariant network and reducing it using ρ yields a symmetric network. This is intuitive: an equivariant network produces the same result up to the order of outputs, and an invariant reduction doesn't care about the order of it's inputs.

4.8.2 Algebraic structure of approximable functions

We have constructed sets I_k^l , containing permutation invariant networks. We now show that the way they are constructed is not too restrictive, i.e. that any analytical symmetric function can be approximated with arbitrary precision by a sufficiently expressive network of our construct.

Theorem 2. For all n, k, l and for all compact K , $I_k^l|_K$ is dense in $\mathcal{I}_k^l|_K$.

We begin by showing that the set of functions that can be approximated to arbitrary precision (i.e. the closure of $I_k^l|_K$) can be equipped with an algebraic structure using $+$ and \times operators constructed pointwise with the $+$ and \times in \mathbb{R} , i.e. $f + g = x \mapsto f(x) +_{\mathbb{R}} g(x)$ and $f \times g = x \mapsto f(x) \times_{\mathbb{R}} g(x)$. Intuitively, we have shown how to combine networks E_k^l while retaining permutation equivariance, and want to "lift" this to the set of functions that can be approximated as limits of our networks. The first step of the proof is to show that the closure of $I_k^l|_K$ is a ring, i.e. that it is stable by sum, product and that each element has an inverse for $+$, as well as a vectorial space, making it an algebra.

The main 'technique' to prove this is to pick the right δ s and ϵ s to obtain uniform convergence over K . The second step is to prove that this closure contains a generative family of the set of all polynomials that operate symmetrically on the batch dimension. Finally, because symmetric

polynomials are dense in the set of all symmetric functions this will conclude the proof. We wish to adapt the Stone-Weierstrass theorem, which states that on a compact, any continuous function of a single variable can be approximated with polynomials. We need an extension to symmetric functions, as the Stone-Weierstrass theorem works for univariate functions, or in the i.i.d case only.

Stability by composition "o". We begin by showing that if two equivariant functions can be approximated, then so can their composition (informally, one can think " $\lim(f \circ g) = (\lim f) \circ (\lim g)$ "). Formally, we prove:

Lemma 1. *If $f_1 \in \overline{E_{l_1}^{l_2}}|_K$ and $f_2 \in \overline{E_{l_2}^{l_3}}|_{f_1(K)}$ then $f_2 \circ f_1 \in \overline{E_{l_1}^{l_3}}|_K$.*

Proof. Let $\varepsilon > 0$, f_2 is continuous on a compact set, thus uniformly continuous, and there exists an $\eta > 0$ such that $\|x - x'\| < \eta$ implies $\|f_2(x) - f_2(x')\| < \frac{\varepsilon}{2}$. Now let $g_1 \in \overline{E_{l_1}^{l_2}}|_K$ be such that $\|g_1 - f_1\|_\infty \leq \eta$ and $g_2 \in \overline{E_{l_2}^{l_3}}|_{f_1(K)}$ such that $\|g_2 - f_2\|_\infty \leq \frac{\varepsilon}{2}$, then, for x in K

$$\begin{aligned} \|f_2 \circ f_1(x) - g_2 \circ g_1(x)\| &\leq \|f_2 \circ f_1(x) - g_2 \circ f_1(x)\| + \|g_2 \circ f_1(x) - g_2 \circ g_1(x)\| \\ &\leq \varepsilon \end{aligned}$$

□

Intuitively, this Lemma says: if your approximations of f_1 and of f_2 are good enough, then composing them yields a good enough approximation of $f_1 \circ f_2$.

Stability by concatenation "concat". We now show that limits can be concatenated, i.e. " $\text{concat}(\lim f, \lim g) = \lim \text{concat}(f, g)$ ". Formally:

Lemma 2. *For any continuous functions $g: \mathbb{R}^k \mapsto \mathbb{R}^l$, the restriction of the function $G: \mathbb{R}^{n \times k} \mapsto \mathbb{R}^{n \times l}$, defined as $G(x) = (g(x_1), \dots, g(x_n))$, to a compact K is in $\overline{E_k^l}|_K$. More precisely, for all $L \geq 2$, the restriction of G to K is in $\overline{E(L)_k^l}|_K$.*

Proof. This is a consequence of the neural network universal approximation theorem, as stated e.g. in [Cybenko \[1989\]](#). □

The previous Lemma is simply the application of the universal approximation theorem to each component function independantly. Intuitively, this trivial statement will let us use "many approximations in parallel".

Lemma 3. *If $f_1 \in \overline{E_k^{l_1}}|_K$, $f_2 \in \overline{E_k^{l_2}}|_K$ and f_1 and f_2 have the same number of layers (i.e. they have the same induction depth), then $\text{concat}_1(f_1, f_2) \in \overline{E_k^{l_1, l_2}}|_K$, with*

$$\text{concat}_1(x, y)_{i,j} = \begin{cases} x_{i,j} & \text{if } j \leq l_1 \\ y_{i,j-l_1} & \text{otherwise} \end{cases} \quad (4.34)$$

Proof. By induction on the number of layers L ,

- if $L = 0$, the result is clear.
- if $L > 0$, let g_1, Γ_1, Λ_1 and β_1 as well as g_2, Γ_2, Λ_2 and β_2 be the parameters associated to f_1 and f_2 , then, by induction, $\text{concat}_1(g_1, g_2)$ is a permutation equivariant network, and $\text{concat}_1(f_1, f_2)$ is obtained by setting Γ to be the block diagonal matrix obtained with Γ_1 and Γ_2 , Λ , the block diagonal matrix obtained with Λ_1 and Λ_2 , and β the concatenation of both β 's.

□

Lemma 4. *If $f_1 \in \overline{E_k^{l_1}|_K}$, $f_2 \in \overline{E_k^{l_2}|_K}$, then $\text{concat}_1(f_1, f_2) \in \overline{E_k^{l_1+l_2}|_K}$.*

Proof. Let $\varepsilon > 0$, let $g_1 \in E_k^{l_1}|_K$ and $g_2 \in E_k^{l_2}|_K$ be such that $\|g_1 - f_1\|_\infty \leq \frac{\varepsilon}{4}$ and $\|g_2 - f_2\|_\infty \leq \frac{\varepsilon}{4}$. Denote by L_1 and L_2 the numbers of layers of g_1 and g_2 . We assume $L_1 \geq L_2$ without loss of generality. By lemma 2, there exist $h_1 \in E_{l_1}^{l_1}|_K$ and $h_2 \in E_{l_2}^{l_2}|_K$ with h_1 of depth 2 and h_2 of depth $L_1 - L_2 + 2$ such that $\|h_1 - Id\|_\infty \leq \frac{\varepsilon}{4}$ on $g_1(K)$ and $\|h_2 - Id\|_\infty \leq \frac{\varepsilon}{4}$ on $g_2(K)$. The networks $h_1 \circ g_1$ and $h_2 \circ g_2$ have the same number of layers, consequently, $\text{concat}_1(h_1 \circ g_1, h_2 \circ g_2) \in \overline{E_k^{l_1+l_2}|_K}$. Besides,

$$\|\text{concat}_1(f_1, f_2) - \text{concat}_1(h_1 \circ g_1, h_2 \circ g_2)\|_\infty \quad (4.35)$$

$$\leq \|f_1 - g_1\|_\infty + \|h_1 \circ g_1 - g_1\|_\infty + \|f_2 - g_2\|_\infty + \|h_2 \circ g_2 - g_2\|_\infty \quad (4.36)$$

$$\leq \varepsilon \quad (4.37)$$

yielding the result. □

Lemma 3 is simply a formal statement that you can concatenate equivariant networks, and that you get an equivariant network. It is trivial, but very usefull: if we can concatenate functions f_1 and f_2 we can then, for instance, sum them using an additional layer. Lemma 4 is more interesting: concatenating approximations that are "good enough" yields an approximation of the concatenation.

Stability by sum "+". We now show that we can sum limits, i.e. " $\lim f + \lim g = \lim(f + g)$ ". Formally:

Lemma 5. *If f_1 and f_2 are in $\overline{E_k^l|_K}$, then $f_1 + f_2$ is too.*

Proof. By lemma 3, $\text{concat}_1(f_1, f_2)$ is in $\overline{E_k^{2l}|_K}$. Consider the layer g , with kernels

$$\Gamma_{i,j} = \begin{cases} 1 & \text{if } j = i \text{ or } j = k + i \\ 0 & \text{otherwise} \end{cases}$$

for $1 \leq i \leq l, 1 \leq j \leq 2l, \Lambda = 0, \beta = 0$. By lemma 1, as both $\text{concat}_1(f_1, f_2)$ and g are in closures of permutation equivariant networks, their composition is too. This composition is $\text{act}(f_1 + f_2)$. By the universal approximation theorem act^{-1} is also in the closure so $f_1 + f_2$ is in the closure. □

Thus closures of permutation equivariant networks is stable by $+$, where $+$ is defined pointwise using the $+$ in \mathbb{R} . More generally, following similar reasonings and noting that multiplication by a real scalar $\alpha \in \mathbb{R}$ is possible with linear layers, closures of permutation equivariant networks are vectorial spaces on \mathbb{R} . It follows that closures of permutation invariant networks are vectorial spaces too. Thus we have verified the vectorial space structure on \mathbb{R} .

Broadcasted functions: For the following proofs, we will need the ability to broadcast functions on the batch axis. Intuitively, if f can be approximated, then f copied b times also can.

Lemma 6. *If $f \in \overline{I_k^l|_K}$, then F defined by*

$$F(x)_{i,j} = f(x)_j \quad (4.38)$$

for all i, j , is in $\overline{E_k^l|_K}$

Proof. By definition, for any $\varepsilon > 0$, there exists a G in $E_k^l|_K$ such that f and $\rho(G)$ are at distance at most $\frac{\varepsilon}{2}$. Let α be a non zero real number such that $\text{act}^{-1}(\alpha G(x))$ is well defined for any $x \in K$. Consider the equivariant layer

$$m(x)_{i,j} = \alpha^{-1} \rho(\text{act}(x))_j. \quad (4.39)$$

Let η_1 be a positive real number, and L_{η_1} be a compact set that contains both $\text{act}^{-1}(\alpha G(K))$ and any ball of radius η_1 contained in this set. m is uniformly continuous on L_{η_1} , and consequently there exists an η_2 such that if x and y are at distance at most η_2 , $m(x)$ and $m(y)$ are at distance at most $\frac{\varepsilon}{2}$. Now, by composition and the universal approximation theorem, let $h \in E_k^l$ be such that h and $\text{act}^{-1}(\alpha G)$ are at distance at most $\min(\eta_1, \eta_2)$. Then $m \circ \text{act}^{-1}(\alpha G)$ and $m \circ h$ are at distance at most $\frac{\varepsilon}{2}$, and by triangular inequality, F and $m \circ h$ are at distance at most ε . \square

Stability by product " \times ". We now show that we can multiply limits, *i.e.* " $\lim f \times \lim g = \lim(f \times g)$ ". The ingredients are: (i) the broadcast operation we just defined, for technical reasons. (ii) the ability to compose with a log, (iii) the ability to sum and to compose with an exp. This gives us the product, using $\exp(\log(a) + \log(b)) = ab$.

Lemma 7. *If f_1 and f_2 are in $\overline{I_k^l|_K}$, then $f_1 f_2$ is too.*

Proof. Let F_1 and F_2 be the extensions of f_1, f_2 as defined in lemma 6. There exists a $C \in \mathbb{R}$ such that for all $i, j, x \in K$, $F_1(x)_{i,j} + C > 0$, and similarly for F_2 . Consequently, by lemma 1, lemma 2 and lemma 5, $\exp(\log(F_1 + C) + \log(F_2 + C)) = F_1 F_2 + F_1 C + F_2 C + C^2 \in \overline{E_k^l|_K}$. As this closure is a vectorial space, $F_1 F_2 \in \overline{E_k^l|_K}$. Consequently, $f_1 f_2 = \rho(F_1 F_2) \in \overline{I_k^l|_K}$. \square

We proved that $\overline{I_k^l|_K}$ is stable by $+$, and by multiplication by a real scalar $\alpha \in \mathbb{R}$, thus $\overline{I_k^l|_K}$ is a vector space over \mathbb{R} (viewed as a field). We also proved that $\overline{I_k^l|_K}$ is stable by \times . Because $+$ and \times are defined using $+\mathbb{R}$ and $\times\mathbb{R}$, it is clear that \times is bilinear. Thus, $\overline{I_k^l|_K}$ is an algebra.

4.8.3 The closure contains a generative family of all symmetric functions (**)

We have now shown that $\overline{I_k^l|_K}$ is an algebra. We are left to prove that it contains a generative family of the continuous symmetric functions. Let us first exhibit a family of continuous symmetric functions that is contained in the set of interest. We will then show that this family generates all continuous symmetric functions.

The key idea of the proof is that a symmetric function on variables x_1, \dots, x_n can be seen as an other function of one variable, evaluated at n different points. Technically speaking, the difficulty of the proof is to see how to jump from functions of a space X to functions of a space X^n . One can think about the case of univariate functions ($X = \mathbb{R}$ for instance), that we wish to turn into multivariate functions (if $X = \mathbb{R}$, $X^n = \mathbb{R}^n$), though our proof will be more generic: we care about *batches* of *vectors*, so in practice we use $X = \mathbb{R}^d$ (the vectors) and $X^n = \mathbb{R}^d \otimes^n \mathbb{R}^d$ (the batches of vectors). The idea is to use an algebra of functions on X , that we can denote F_X (the algebra structure lets us scale, sum and multiply functions) and use it to construct an algebra of functions on X^n , that we will denote F_{X^n} . We consider F_X provided (we can use the universal approximation theorem to obtain it with neural networks), and we seek to construct F_{X^n} . We begin by establishing "bridges" between the two sets.

Lemma 8. *For all f , restriction of a function from \mathbb{R}^l to \mathbb{R}^k to a compact set K , the symmetric function F , defined on $K^{n \times l}$ by*

$$F(x) = \sum_{i=1}^n f(x_i) \quad (4.40)$$

is in I_k^l .

Proof. By the universal approximation theorem, f is in $\overline{I_k^l|_K}$. By lemma 6, there exists a G in $\overline{E_k^l|_K}$ that replicates f along the batch axis of an equivariant network. Consequently, $\rho(G) = F$ is in $\overline{I_k^l|_K}$. \square

This first "bridge" is intuitive: if you have a function f on X , you can turn it into a function on X^n by evaluating it at several points simultaneously. If you sum the results, you obtain an invariant function. We are going to prove that this family of functions generates the set of all symmetric polynomials. Then, deriving a generalization of Stone Weierstrass theorem to symmetric functions, we obtain the final result.

We now present a tool to turn a function on X^n into a *symmetric* function on X^n :

Symmetrization operator. To keep things general, in what follows, X denotes an arbitrary set, and S is the symmetrization operator on functions of X^n , i.e. for all $(x_1, \dots, x_n) \in X^n$,

$$(Sf)(x_1, \dots, x_n) = \sum_{\sigma} f(x_{\sigma(1)}, \dots, x_{\sigma(n)}) \quad (4.41)$$

where the sum is over all permutations of $[1, n]$. This is well defined: for any function f defined on X^n , $f \circ \sigma$ is also a function of X^n . Since all possible results for all possible permutations

of the inputs are summed, permuting the order of the inputs of Sf will simply reorder the sum. Because the $+$ operation in $f(X^n)$ is commutative, the result is invariant to permutations. An important property of S is that it is linear. Let $\alpha \in \mathbb{R}$ and g a functions of X^n ,

$$(S(\alpha f + g))(x_1, \dots, x_n) = \sum_{\sigma} \alpha f(x_{\sigma(1)}, \dots, x_{\sigma(n)}) + g(x_{\sigma(1)}, \dots, x_{\sigma(n)}) \quad (4.42)$$

$$= \alpha \sum_{\sigma} (f(x_{\sigma(1)}, \dots, x_{\sigma(n)})) + \sum_{\sigma} g(x_{\sigma(1)}, \dots, x_{\sigma(n)}) \quad (4.43)$$

$$= (\alpha(Sf) + (Sg))(x_1, \dots, x_n). \quad (4.44)$$

In what follows, we will build multivariate polynomials (in preparation of the use of a density result). We begin by recalling the definition of a *monomial*.

Definition 5. Monomials (informal) A monomial is a product of powers of variables (note the plural) with non-negative integer exponents. In the classical context of univariate polynomials, it is of the form x^n . In the context of multivariate polynomials, it is of the form $x^a y^b \dots z^c$. The degrees and the number of terms is arbitrary. The point is, there is no $+$.

Example: For instance, $xy + yz$ is a multivariate *polynomial*, because there is a $+$, but $x^{128}y^3z^{29}$ is a monomial. The polynomial $xy + yz$ is composed of monomials xy and yz .

Polynomials from monomials. Let F an algebra of functions on X (we can sum, multiply, and scale the functions of F), and let P be the algebra of functions of X^n generated by the functions $f(x_k): x \rightarrow f(x_k)$ for f in F , with a slight abuse of notations.

We are allowed to define P like this: we give P an algebra structure, so it suffices to provide a family of functions, and the whole set P is generated using composition laws. The definition is valid: $f(x_k)$ takes a value in X^n , and maps it to a constant image. The functions we give are monomials with one variable. Because of the algebra structure, we have a \times and so we can raise the degree, still obtaining monomials of the form $f_1(x_1) \dots f_n(x_n)$. Then, because of the vector space structure, we can generate *polynomials*, of the form $\sum_i \alpha_i f_{i_1}(x_{i_1}) \dots f_{i_n}(x_{i_n})$. Then, it remains to determine if the P that we generate this way is interesting. In fact, we will first symmetrize P (we care about symmetric functions), and then wonder if SP is interesting.

P is linearly generated by the monomials $f_1(x_1) \dots f_n(x_n)$ for f_k arbitrary functions of F , by definition.

We are interested in the symmetrization of P , SP . By linearity of S , SP is generated by the symmetrized monomials,

$$Sf_1(x_1) \dots f_n(x_n) = \sum_{\sigma} \prod_{k=1}^n f_k(x_{\sigma(k)}). \quad (4.45)$$

Here, this result is based on an interesting transfer of structure between P and SP : because S is linear, the base of P yields a base of SP through S .

Now comes the crucial part of the proof: intuitively, because the functions f on X^n are symmetric, you don't need to know how the polynomials evaluate on all x_i , x_1 is enough. Note that this is true of polynomials, not all functions, but the closure remains the same.

Lemma 9. *SP is generated as an algebra by $Sf(x_1)$ for $f \in F$. Notably, $Sf(x_1)$ takes the special form*

$$Sf(x_1) = \sum_{\sigma} f(x_{\sigma(1)}) = (n-1)! \sum_{k=1}^n f(x_k). \quad (4.46)$$

Typically, for our case, $X = \mathbb{R}^l$ for l the number of input features, F is an algebra of functions containing the multivariate polynomials on \mathbb{R}^l , and SP thus contains the set of all polynomials which are symmetric along the batch dimension.

Proof. Call *rank* of a monomial $f_1(x_1) \dots f_n(x_n)$, the number of functions f_k such that $f_k \neq 1$. Let k_1, \dots, k_r be these indices. Up to renaming f_{k_1} to f_1 , etc., the monomial can be written as $f_1(x_{k_1}) \dots f_r(x_{k_r})$.

We will work by induction on r . For $r = 1$ the claim is trivial.

Since S does not care about permuting the variables, we have

$$Sf_1(x_{k_1}) \dots f_r(x_{k_r}) = Sf_1(x_1) \dots f_r(x_r) = \sum_{\sigma \in S_K} \prod_{i=1}^r f_i(x_{\sigma(i)}) \quad (4.47)$$

Intuitively, the magic has already happened: x_1 will take all positions in the "input batch". The values $\sigma(r+1), \dots, \sigma(n)$ have no influence so that

$$Sf_1(x_1) \dots f_n(x_n) = (n-r)! \sum_{\sigma \in \text{Inj}_r^n} \prod_{i=1}^r f_i(x_{\sigma(i)}) \quad (4.48)$$

where Inj_r^n is the set of injective functions from r to n . This extra term comes from the fact that we have n variables but only r usefull ones, and the previous line gets 'rid' of that by focusing on usefull variables.

Assume we can generate all symmetric monomials up to rank r . We need to add a variable (replace one that was "not used" by one that is "used".) By definition we can generate $Sf_{r+1}(x_1)$ for any $f_{r+1} \in F$ (using our construction of P , then SP). Then using the algebra structure, and the recurrence hypothesis, we can generate the product:

$$\begin{aligned} \frac{1}{(n-r-1)!} (Sf_{r+1}(x_1)) \left(\sum_{\sigma \in \text{Inj}_r^n} \prod_{i=1}^r f_i(x_{\sigma(i)}) \right) &= \left(\sum_{k \in n} f_{r+1}(x_k) \right) \left(\sum_{\sigma \in \text{Inj}_r^n} \prod_{i=1}^r f_i(x_{\sigma(i)}) \right) \\ &= \sum_{\sigma \in \text{Inj}_r^n} \sum_{k \in n} f_{r+1}(x_k) \prod_{i=1}^r f_i(x_{\sigma(i)}) \end{aligned}$$

Now, for each σ , we can decompose according to whether $k \in \text{Im } \sigma$ or $k \in n \setminus \text{Im } \sigma$, where $\text{Im } \sigma = \{\sigma(1), \dots, \sigma(r)\}$ is the image of σ . Intuitively, the variable that we added is permuted

around, and can either i) arrive on a variable slot that was not used before, or on a slot that was used. We obtain two terms:

$$\dots = \sum_{\sigma \in \text{Inj}_r^n} \sum_{k \in \text{Im } \sigma} f_{r+1}(x_k) \prod_{i=1}^r f_i(x_{\sigma(i)}) + \sum_{\sigma \in \text{Inj}_r^n} \sum_{k \in n \setminus \text{Im } \sigma} f_{r+1}(x_k) \prod_{i=1}^r f_i(x_{\sigma(i)})$$

But if k is not in $\text{Im } \sigma$, then $(\sigma(1), \dots, \sigma(r), k)$ is an injective function from $r+1$ to n . So summing over σ then on $k \in n \setminus \text{Im } \sigma$ is exactly equivalent to summing over $\sigma \in \text{Inj}_{r+1}^n$. So the second term above is

$$\sum_{\sigma \in \text{Inj}_{r+1}^n} \left(\prod_{i=1}^r f_i(x_{\sigma(i)}) \right) f_{r+1}(\sigma(r+1)) = \sum_{\sigma \in \text{Inj}_{r+1}^n} \prod_{i=1}^{r+1} f_i(x_{\sigma(i)}) = S f_1(x_{k_1}) \dots f_{r+1}(x_{k_{r+1}})$$

which is the one we are interested in (it gives us the recurrence hypothesis we want for $r+1$). So if we prove that we can generate the first term, we are done by subtracting the first term on both sides using the algebra structure. Let us consider the first term, with $k \in \text{Im } \sigma$. Now, since $k \in \text{Im } \sigma$, we can decompose over the cases $k = \sigma(1), \dots, k = \sigma(r)$. The notations are a bit heavy, but the idea of the trick is very simple: where there is a collision between the new variable and the one already used (say, $x_{r+1} = x_i$) use a new function $\tilde{f}_i = f_{r+1} \times f_i$. You can do this because F is an algebra. Let us now do it:

$$\sum_{\sigma \in \text{Inj}_r^n} \sum_{k \in \text{Im } \sigma} f_{r+1}(x_k) \prod_{i=1}^r f_i(x_{\sigma(i)}) = \sum_{\sigma \in \text{Inj}_r^n} \sum_{j=1}^r f_{r+1}(x_{\sigma(j)}) \prod_{i=1}^r f_i(x_{\sigma(i)}) \quad (4.49)$$

$$= \sum_{j=1}^r \sum_{\sigma \in \text{Inj}_r^n} \prod_{i=1}^r \tilde{f}_{ij}(x_{\sigma(i)}) \quad (4.50)$$

where

$$\tilde{f}_{ij} := \begin{cases} f_i & i \neq j \\ f_i f_{r+1} & i = j \end{cases} \quad (4.51)$$

Now since F is a ring, $f_i f_{r+1} \in F$. For each j the term

$$\sum_{\sigma \in \text{Inj}_r^n} \prod_{i=1}^r \tilde{f}_{ij}(x_{\sigma(i)}) \quad (4.52)$$

is equal to $S \tilde{f}_{1j} \dots \tilde{f}_{rj}$ up to a factor $(n - (r+1))!$ that can be inverted (vector space structure). By our induction hypothesis, each term can be generated, and this ends the proof. \square

The hard part is done. Now, we just need density results.

Lemma 10. *For any compact K , any $l \in \mathbb{N}$, the intersection of \mathcal{I}_l^1 with the set of multivariate polynomials is dense in \mathcal{I}_l^1 for the infinity norm.*

Proof. Let $\varepsilon > 0$, and f be in I_l^1 . There exists a multivariate polynomials P such that $\|P - f\|_\infty \leq \varepsilon$. Let us consider the symmetrized polynomial

$$\tilde{P}(x_1, \dots, x_n) = \frac{1}{n!} \sum_{\sigma} P(x_{\sigma(1)}, \dots, x_{\sigma(n)}). \quad (4.53)$$

Then \tilde{P} is in the intersection, and, for $x \in K$,

$$\|\tilde{P}(x) - f(x)\| = \left\| \frac{1}{n!} \sum_{\sigma} (P(x_{\sigma(1)}, \dots, x_{\sigma(n)}) - f(x_{\sigma(1)}, \dots, x_{\sigma(n)})) \right\| \quad (4.54)$$

$$\leq \frac{1}{n!} \sum_{\sigma} \|P(x_{\sigma(1)}, \dots, x_{\sigma(n)}) - f(x_{\sigma(1)}, \dots, x_{\sigma(n)})\| \quad (4.55)$$

$$\leq \varepsilon. \quad (4.56)$$

□

Intuitively, the infinity norm does not change if we change the variables around for both P and f at the same time (think about a change of basis).

We now have all the ingredients to end the proof. We just have to plug in the universal approximation theorem to get our polynomials. For a given compact K of \mathbb{R}^l , for any multivariate polynomial P of \mathbb{R}^l , any $\varepsilon > 0$, there exists an element f of I_k^1 at distance at most ε of $x \rightarrow \sum_{i=1}^n P(x_i)$. This means that the closure of the considered set contains all such functions. As this closure is an algebra (it is both a ring and a vectorial space), by lemma 8, it contains the intersection of \mathcal{I}_l^2 with the set of multivariate polynomials. By lemma 10, it contains \mathcal{I}_l^1 , which ends the proof.

Chapter 5

Adaptive density estimation for generative modelling

Contents

5.1	Outline of this chapter	96
5.2	Related work	97
5.3	Preliminaries on MLE and adversarial training	99
5.4	Adaptive Density Estimation and hybrid adversarial-likelihood training	101
5.5	Experimental evaluation	103
5.6	Complementary evaluations	111
5.7	Qualitative influence of the feature space flexibility	112
5.8	Conclusion	113

Unsupervised learning of generative models has seen tremendous progress over recent years, in particular due to generative adversarial networks (GANs), variational auto-encoders, and flow-based models. GANs have dramatically improved sample quality, but suffer from two drawbacks: (i) they mode-drop, *i.e.*, do not cover the full support of the train data, and (ii) they do not allow for likelihood evaluations on held-out data. In contrast likelihood-based training encourages models to cover the full support of the train data, but yields poorer samples. These mutual shortcomings can in principle be addressed by training generative latent variable models in a hybrid adversarial-likelihood manner. However, we show that commonly made parametric assumptions create a conflict between them, making successful hybrid models non trivial. As a solution, we propose to use deep invertible transformations in the latent variable decoder. This approach allows for likelihood computations in image space, is more efficient than fully invertible models, and can take full advantage of adversarial training. We show that our model significantly improves over existing hybrid models: offering GAN-like samples, IS and FID scores that are competitive with fully adversarial models and improved likelihood scores. The material presented in this chapter is based on the paper "Adaptive Density Estimation for Generative Models", Thomas Lucas, Konstantin Shmelkov, Karteeek Alahari, Cordelia Schmid, and Jakob Verbeek, conference on Neural Information Processing Systems (NeurIPS) 2019.

5.1 Outline of this chapter

Successful recent generative models of natural images can be divided into two broad families, which are trained in fundamentally different ways. The first is trained using likelihood-based criteria, which ensure that all training data points are well covered by the model. This category includes variational auto-encoders (VAEs) [Kingma and Welling, 2014b, Kingma et al., 2016b, Rezende and Mohamed, 2015, Rezende et al., 2014], autoregressive models such as PixelCNNs [Salimans et al., 2017b, van den Oord et al., 2016b], and flow-based models such as real-NVP [Dinh et al., 2017, Ho et al., 2019, Kingma and Dhariwal, 2018]. The second category is trained based on a signal that measures to what extent (statistics of) samples from the model can be distinguished from (statistics of) the training data, *i.e.*, based on the quality of samples drawn from the model. This is the case for generative adversarial networks (GANs) [Arjovsky et al., 2017, Goodfellow et al., 2014, Karras et al., 2018], and moment matching methods [Li et al., 2015].

Despite tremendous recent progress, existing methods exhibit a number of drawbacks. Adversarially trained models such as GANs do not provide a density function, which poses a fundamental problem as it prevents assessment of how well the model fits held out and training data. Moreover, adversarial models typically do not allow to infer the latent variables that underlie observed images. Finally, adversarial models suffer from mode collapse [Arjovsky et al., 2017], *i.e.*, they do not cover the full support of the training data. Likelihood-based model on the other hand are trained to put probability mass on all elements of the training set, but over-generalise and produce samples of substantially inferior quality as compared to adversarial models. The models with the best likelihood scores on held-out data are autoregressive models [Menick and Kalchbrenner, 2019], which suffer from the additional problem that they are extremely inefficient to sample from [Ramachandran et al., 2017], since images are generated pixel-by-pixel. The sampling inefficiency makes adversarial training of such models prohibitively expensive.

In order to overcome these shortcomings, we seek to design a model that (i) generates high-quality samples typical of adversarial models, (ii) provides a likelihood measure on the entire image space, and (iii) has a latent variable structure to allow for efficient sampling, that permits adversarial training. Additionally we show that, (iv) a successful hybrid adversarial-likelihood paradigm requires going beyond simplifying conditional independence assumptions commonly used with likelihood based latent variable models. These simplifying assumptions on the conditional distribution on data x given latent variables z , $p(x|z)$, include full independence across the dimensions of x and/or simple parametric forms such as Gaussian [Kingma and Welling, 2014b], as detailed in Section 2.9.1, or use fully invertible networks [Dinh et al., 2017, Kingma and Dhariwal, 2018]. These assumptions create a conflict between achieving high sample quality and high likelihood scores on held-out data. Autoregressive models, such as PixelCNNs [Salimans et al., 2017b, van den Oord et al., 2016b], do not make factorization assumptions, but are extremely inefficient to sample from. As a solution, we propose learning a non-linear invertible function f_ψ between the image space and an abstract feature space as illustrated in Figures 5.1 and 5.2. Training a model with full support in this feature space induces a model in the image space that does not

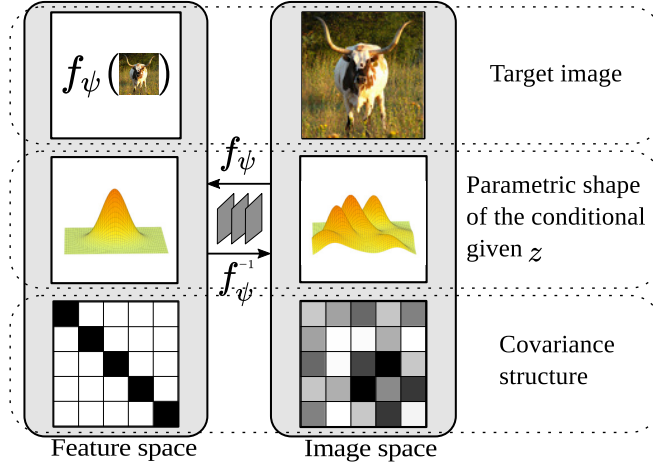


Figure 5.1: An invertible non-linear mapping f_ψ maps an image x to a vector $f_\psi(x)$ in feature space. f_ψ is trained to adapt to modelling assumptions made by a trained density p_θ in feature space.

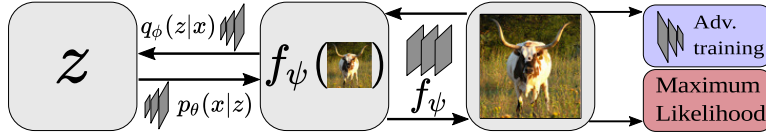


Figure 5.2: Variational inference is used to train a latent variable generative model in feature space. The invertible mapping f_ψ maps back to image space, where adversarial training can be performed together with MLE.

make Gaussianity or independence assumptions in the conditional $p(x|z)$. Trained by MLE, f_ψ adapts to modelling assumptions made by p_θ so we refer to this approach as "Adaptive density estimation".

We experimentally validate our approach on the CIFAR-10 dataset with an ablation study. Our model significantly improves over existing hybrid models, producing GAN-like samples as shown in Figure 5.3, and IS and FID scores that are competitive with fully adversarial models. At the same time, we obtain likelihoods on held-out data comparable to state-of-the-art likelihood-based methods which requires covering the full support of the dataset. We further confirm these observations with quantitative and qualitative experimental results on the STL-10, ImageNet and LSUN datasets.

5.2 Related work

Mode-collapse in GANs has received considerable attention, and stabilizing the training process as well as improved and bigger architectures have been shown to alleviate this issue [Arjovsky



Figure 5.3: Our model yields compelling samples while the optimization of likelihood ensures coverage of all modes in the training support and thus sample diversity, here on LSUN churches (64×64).

et al., 2017, Gulrajani et al., 2017b, Miyato et al., 2018a]. Another line of work focuses on allowing the discriminator to access batch statistics of generated images, as pioneered by Karras et al. [2018], Salimans et al. [2016b], and further generalized by Lin et al. [2018], Lucas et al. [2018a]. This enables comparison of distributional statistics by the discriminator rather than only individual samples. Other approaches to encourage diversity among GAN samples include the use of maximum mean discrepancy [Arbel et al., 2018], optimal transport [Salimans et al., 2018], and determinantal point processes [Elfeki et al., 2018]. In contrast to our work, these models lack an inference network, and do not define an explicit density over the full data support.

An other line of research has explored inference mechanisms for GANs. The discriminator of BiGAN [Donahue et al., 2017] and ALI [Dumoulin et al., 2017a], given pairs (x, z) of images and latent variables, predict if z was encoded from a real image, or if x was decoded from a sampled z . In Ulyanov et al. [2018] the encoder and the discriminator are collapsed into one network that encodes both real images and generated samples, and tries to spread their posteriors apart. In Chen et al. [2018] a symmetrized KL divergence is approximated in an adversarial set-up, and uses reconstruction losses to improve the correspondence between reconstructed and target variables for x and z . Similarly, Rosca et al. [2017] use a discriminator to replace the KL divergence term in the variational lower bound used to train VAEs with the density ratio trick. In Makhzani et al. [2016] the KL divergence term in a VAE is replaced with a discriminator that compares latent variables from the prior and the posterior in a more flexible manner. The VAE-GAN model [Larsen et al., 2016] uses the intermediate feature maps of a GAN discriminator as target space for a VAE. This regularization is more flexible than the standard KL divergence. Unlike ours, these methods do not define a likelihood over the image space.

Likelihood-based models typically make modelling assumptions that conflict with adversarial training, these include strong factorization and/or Gaussianity. In our work we avoid these limitations by learning the shape of the conditional density on observed data given latent variables, $p_\theta(x|z)$, beyond fully factorized Gaussian models. As in our work, Flow-GAN [Grover et al., 2018] also builds on invertible transformations to construct a model that can be trained in a hybrid adversarial-MLE manner, see Figure 5.2. However, Flow-GAN does not use efficient non-invertible layers we introduce, and instead relies entirely on invertible layers. Other approaches combine autoregressive decoders with latent variable models to go beyond typical parametric assumptions in pixel space [Chen et al., 2017, Gulrajani et al., 2017c, Lucas and Verbeek, 2018a]. They, however, are not amenable to adversarial training due to the prohibitively slow sequential

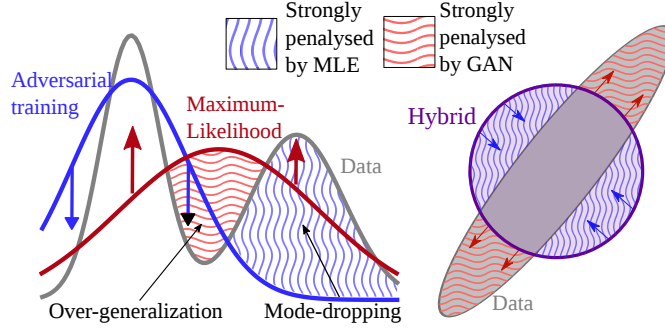


Figure 5.4: Maximum likelihood training pulls probability mass towards high-density regions of the data distribution, while adversarial training pushes mass out of low-density regions. (Right) Independence assumptions become a source of conflict in a joint training setting, making hybrid training non-trivial.

pixel sampling.

5.3 Preliminaries on MLE and adversarial training

Maximum-likelihood and over-generalization. The de-facto standard approach to train generative models is maximum-likelihood estimation. It maximizes the probability of data sampled from an unknown data distribution p^* under the model p_θ w.r.t. the model parameters θ . This is equivalent to minimizing the Kullback-Leibler (KL) divergence, $D_{\text{KL}}(p^* || p_\theta)$, between p^* and p_θ . This yields models that tend to cover all the modes of the data, but put mass in spurious regions of the target space; a phenomenon known as “over-generalization” or “zero-avoiding” [Bishop, 2006], and manifested by unrealistic samples in the context of generative image models, see Figure 5.4. Over-generalization is inherent to the optimization of the KL divergence oriented in this manner. Real images are sampled from p^* , and p_θ is explicitly optimized to cover all of them. The training procedure, however, does not sample from p_θ to evaluate the quality of such samples (ideally using the inaccessible $p^*(x)$ as a score). Therefore p_θ may put mass in spurious regions of the space without being heavily penalized. We refer to this kind of training procedure as “coverage-driven training” (CDT). This optimizes a loss of the form $\mathcal{L}_C(p_\theta) = \int_x p^*(x) s_c(x, p_\theta) dx$, where $s_c(x, p_\theta) = \ln p_\theta(x)$ evaluates how well a sample x is covered by the model. Any score that verifies $\mathcal{L}_C(p_\theta) = 0 \iff p_\theta = p^*$ is equivalent to the log-score, in which case it is called *strictly proper* (see e.g. Dawid and Musio [2014]), which forms a justification for MLE on which we focus.

Explicitly evaluating sample quality is redundant in the regime of unlimited model capacity and training data. Indeed, putting mass on spurious regions takes it away from the support of p^* , and thus reduces the likelihood of the training data. In practice, however, datasets and model capacity are finite, and models *must* put mass outside the finite training set in order to generalize. The maximum likelihood criterion, by construction, only measures *how much* mass goes off the training data, not *where* it goes. In classic MLE, generalization is controlled in two ways: (i)

inductive bias, in the form of model architecture, controls *where* the off-dataset mass goes, and (ii) regularization controls to which extent this happens. An adversarial loss, by considering samples of the model p_θ , can provide a second handle to evaluate and control where the off-dataset mass goes. In this sense, and in contrast to model architecture design, an adversarial loss provides a “trainable” form of inductive bias.

Adversarial models and mode collapse. Adversarially trained models produce samples of excellent quality. As mentioned, their main drawbacks are their tendency to “mode-drop”, and the lack of metric to assess mode-dropping, or their performance in general. The reasons for this are two-fold. First, defining a valid likelihood requires adding volume to the low-dimensional manifold learned by GANs to define a density under which training and test data have non-zero density. Second, computing the density of a data point under the defined probability distribution requires marginalizing out the latent variables, which is not trivial in the absence of an efficient inference mechanism.

When a human expert subjectively evaluates the quality of generated images, samples from the model are compared to the expert’s implicit approximation of p^* . This type of objective may be formalized as $\mathcal{L}_Q(p_\theta) = \int_{\mathbf{x}} p_\theta(\mathbf{x}) s_q(\mathbf{x}, p^*) d\mathbf{x}$, and we refer to it as “quality-driven training” (QDT). To see that GANs [Goodfellow et al. \[2014\]](#) use this type of training, recall that the discriminator is trained with the loss

$$\mathcal{L}_{\text{GAN}} = \int_{\mathbf{x}} p^*(\mathbf{x}) \ln D(\mathbf{x}) + p_\theta(\mathbf{x}) \ln(1 - D(\mathbf{x})) d\mathbf{x}.$$

It is easy to show that the optimal discriminator equals $D^*(\mathbf{x}) = p^*(\mathbf{x}) / (p^*(\mathbf{x}) + p_\theta(\mathbf{x}))$, see [Goodfellow et al. \[2014\]](#). Substituting the optimal discriminator, \mathcal{L}_{GAN} equals the Jensen-Shannon divergence,

$$\mathcal{D}_{\text{JS}}(p^* || p_\theta) = \frac{1}{2} \text{D}_{\text{KL}}(p^* || \frac{1}{2}(p_\theta + p^*)) + \frac{1}{2} \text{D}_{\text{KL}}(p_\theta || \frac{1}{2}(p_\theta + p^*)), \quad (5.1)$$

up to additive and multiplicative constants [[Goodfellow et al., 2014](#)]. This loss, approximated by the discriminator, is symmetric and contains two KL divergence terms. Note that $\text{D}_{\text{KL}}(p^* || \frac{1}{2}(p_\theta + p^*))$ is an integral on p^* , so *coverage driven*. The term that approximates it in \mathcal{L}_{GAN} , *i.e.*, $\int_{\mathbf{x}} p^*(\mathbf{x}) \ln D(\mathbf{x})$, is however independent from the generative model, and disappears when differentiating. Therefore, it cannot be used to perform coverage-driven training, and the generator is trained to minimize $\ln(1 - D(G(\mathbf{z})))$, where $G(\mathbf{z})$ is the deterministic generator that maps latent variables \mathbf{z} to the data space. Assuming $D = D^*$, this yields

$$\int_{\mathbf{z}} p(\mathbf{z}) \ln(1 - D^*(G(\mathbf{z}))) d\mathbf{z} = \int_{\mathbf{x}} p_\theta(\mathbf{x}) \ln \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}) + p^*(\mathbf{x})} d\mathbf{x} = \text{D}_{\text{KL}}(p_\theta || (p_\theta + p^*)/2), \quad (5.2)$$

which is a quality-driven criterion, favoring sample quality over support coverage. An alternative training loss for the generator, also proposed in [Goodfellow et al. \[2014\]](#), is to maximize $\ln D(G(\mathbf{z}))$, which does not modify the optimum but improves early training. As remarked by

Sønderby et al. [2017], it is possible to simultaneously minimize $\ln(1 - D(G(z)))$ and maximize $\ln D(G(z))$ by minimizing

$$\int_z p(z) \ln \left(\frac{1 - D(G(z))}{D(G(z))} \right), \quad (5.3)$$

which under optimality assumption yields $D_{KL}(p_{\theta, \psi} || p^*)$.

The complementarities that appear between maximum likelihood and adversarial training motivate hybrid training of generative models, which is presented in Section 5.4.

5.4 Adaptive Density Estimation and hybrid adversarial-likelihood training

We present a hybrid training approach with MLE to cover the full support of the training data, and adversarial training as a trainable inductive bias mechanism to improve sample quality. Using both these criteria provides a richer training signal, but satisfying both criteria is more challenging than each in isolation for a given model complexity. In practice, model flexibility is limited by (i) the number of parameters, layers, and features in the model, and (ii) simplifying modelling assumptions, usually made for tractability. We show that these simplifying assumptions create a conflict between the two criteria, making successful joint training non trivial. We introduce Adaptive Density Estimation as a solution to reconcile them.

Latent variable generative models, defined as $p_{\theta}(\mathbf{x}) = \int_z p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}) d\mathbf{z}$, typically make simplifying assumptions on $p_{\theta}(\mathbf{x}|\mathbf{z})$, such as full factorization and/or Gaussianity, see *e.g.* Dorta et al. [2018], Kingma and Welling [2014b], Litany et al. [2018]. In particular, assuming full factorization of $p_{\theta}(\mathbf{x}|\mathbf{z})$ implies that any correlations not captured by \mathbf{z} are treated as independent per-pixel noise. This is a poor model for natural images, unless \mathbf{z} captures each and every aspect of the image structure. Crucially, this hypothesis is problematic in the context of hybrid MLE-adversarial training. If p^* is too complex for $p_{\theta}(\mathbf{x}|\mathbf{z})$ to fit it accurately enough, MLE will lead to a high variance in a factored (Gaussian) $p_{\theta}(\mathbf{x}|\mathbf{z})$ as illustrated in Figure 5.4 (right).

This leads to unrealistic blurry samples, easily detected by an adversarial discriminator, which then does not provide a useful training signal. Conversely, adversarial training will try to avoid these poor samples by dropping modes of the training data, and driving the “noise” level to zero. This in turn is heavily penalized by maximum likelihood training, and leads to poor likelihoods on held-out data.

Adaptive density estimation. The point of view of regression hints at a possible solution. For instance, with isotropic Gaussian densities, $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mu(\mathbf{z}), \sigma I)$ with fixed variance, solving the optimization problem $\theta^* \in \max_{\theta} \ln(p_{\theta}(\mathbf{x}))$ is similar to solving $\min_{\theta} ||\mu_{\theta}(\mathbf{z}) - \mathbf{x}||_2$, i.e., ℓ_2 regression, where $\mu_{\theta}(\mathbf{z})$ is the mean of the decoder $p_{\theta}(\mathbf{x}|\mathbf{z})$. The Euclidean distance in RGB space is known to be a poor measure of similarity between images, non-robust to small translations or other basic transformations Mathieu et al. [2016]. One can instead compute the

Euclidean distance in a feature space, $\|f_\psi(\mathbf{x}_1) - f_\psi(\mathbf{x}_2)\|_2$, where f_ψ is chosen so that the distance is a better measure of similarity. A popular way to obtain f_ψ is to use a CNN that learns a non-linear image representation, that allows linear assessment of image similarity. This is the idea underlying GAN discriminators, the FID evaluation measure [Heusel et al., 2017], and the reconstruction loss of VAE-GAN Larsen et al. [2016].

Despite their flexibility, such similarity metrics are in general degenerate in the sense that they may discard information about the data point \mathbf{x} . For instance, two different images \mathbf{x} and \mathbf{y} can collapse to the same points in feature space, i.e., $f_\psi(\mathbf{x}) = f_\psi(\mathbf{y})$. This limits the use of similarity metrics in the context of generative modeling for two reasons: (i) it does not yield a valid measure of likelihood over inputs, and (ii) points generated in the feature space f_ψ cannot easily be mapped to images. To resolve this issue, we chose f_ψ to be a bijection. Given a model p_θ trained to model $f_\psi(\mathbf{x})$ in feature space, a density in image space is computed using the change of variable formula, which yields $p_{\theta,\psi}(\mathbf{x}) = p_\theta(f_\psi(\mathbf{x})) \left| \det \left(\partial f_\psi(\mathbf{x}) / \partial \mathbf{x}^\top \right) \right|$. Image samples are obtained by sampling from p_θ in feature space, and mapping to the image space through f_ψ^{-1} . We refer to this construction as Adaptive Density Estimation. If p_θ provides efficient log-likelihood computations, the change of variable formula can be used to train f_ψ and p_θ together by maximum-likelihood, and if p_θ provides fast sampling adversarial training can be performed efficiently.

MLE with adaptive density estimation. To train a generative latent variable model $p_\theta(\mathbf{x})$ which permits efficient sampling, we rely on amortized variational inference. We use an inference network $q_\phi(\mathbf{z}|\mathbf{x})$ to construct a variational evidence lower-bound (ELBO),

$$\mathcal{L}_{\text{ELBO}}^\psi(\mathbf{x}, \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\ln(p_\theta(f_\psi(\mathbf{x})|\mathbf{z}))] - \text{D}_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})) \leq \ln p_\theta(f_\psi(\mathbf{x})). \quad (5.4)$$

Using this lower bound together with the change of variable formula, the mapping to the similarity space f_ψ and the generative model p_θ can be trained jointly with the loss

$$\mathcal{L}_C(\theta, \phi, \psi) = \mathbb{E}_{\mathbf{x} \sim p^*} \left[-\mathcal{L}_{\text{ELBO}}^\psi(\mathbf{x}, \theta, \phi) - \ln \left| \det \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right| \right] \geq - \mathbb{E}_{\mathbf{x} \sim p^*} [\ln p_{\theta,\psi}(\mathbf{x})]. \quad (5.5)$$

We use gradient descent to train f_ψ by optimizing $\mathcal{L}_C(\theta, \phi, \psi)$ w.r.t. ψ . The $\mathcal{L}_{\text{ELBO}}$ term encourages the mapping f_ψ to maximize the density of points in feature space under the model p_θ , so that f_ψ is trained to match modeling assumptions made in p_θ . Simultaneously, the log-determinant term encourages f_ψ to maximize the volume of data points in feature space. This guarantees that data points cannot be collapsed to a single point in the feature space. We use a factored Gaussian form of the conditional $p_\theta(\cdot|\mathbf{z})$ for tractability, but since f_ψ can arbitrarily reshape the corresponding conditional image space, it still avoids simplifying assumptions in the image space. Therefore, the (invertible) transformation f_ψ avoids the conflict between the MLE and adversarial training mechanisms, and can leverage both.

Adversarial training with adaptive density estimation. To sample the generative model, we sample latents from the prior, $\mathbf{z} \sim p_\theta(\mathbf{z})$, which are then mapped to feature space through $\mu_\theta(\mathbf{z})$,

and to image space through f_ψ^{-1} . We train our generator using the modified objective proposed by [Sønderby et al., 2017], which combines both generator losses considered in [Goodfellow et al., 2014] (see Equation 5.3) and yields:

$$\mathcal{L}_Q(p_{\theta,\psi}) = - \mathbb{E}_{p_\theta(z)} \left[\ln D(f_\psi^{-1}(\mu_\theta(z))) - \ln(1 - D(f_\psi^{-1}(\mu_\theta(z)))) \right]. \quad (5.6)$$

Assuming the discriminator D is trained to optimality at every step, it is easy to demonstrate that the generator is trained to optimize $D_{\text{KL}}(p_{\theta,\psi} \parallel p^*)$. The training procedure, written as an algorithm in Section A.3, alternates between (i) bringing $\mathcal{L}_Q(p_{\theta,\psi})$ closer to its optimal value $\mathcal{L}_Q^*(p_{\theta,\psi}) = D_{\text{KL}}(p_{\theta,\psi} \parallel p^*)$, and (ii) minimizing $\mathcal{L}_C(p_{\theta,\psi}) + \mathcal{L}_Q(p_{\theta,\psi})$. Assuming that the discriminator is trained to optimality at every step, the generative model is trained to minimize a bound on the sum of two symmetric KL divergences:

$$\mathcal{L}_C(p_{\theta,\psi}) + \mathcal{L}_Q^*(p_{\theta,\psi}) \geq D_{\text{KL}}(p^* \parallel p_{\theta,\psi}) + D_{\text{KL}}(p_{\theta,\psi} \parallel p^*) + \mathcal{H}(p^*), \quad (5.7)$$

where the entropy of the data generating distribution, $\mathcal{H}(p^*)$, is an additive constant independent of the generative model $p_{\theta,\psi}$. In contrast, MLE and GANs optimize one of these divergences each.

5.5 Experimental evaluation

We present our evaluation protocol, followed by an ablation study to assess the importance of the components of our model in segmentation. We then show the quantitative and qualitative performance of our model, and compare it to the state of the art on the CIFAR-10 dataset in Section 5.5.3. We present additional results and comparisons on higher resolution datasets in Section 5.5.4.

5.5.1 Evaluation metrics

We evaluate our models with three complementary metrics. To assess sample quality, we report the Fréchet inception distance (FID) [Heusel et al., 2017] and the inception score (IS) [Salimans et al., 2016b], which are the de facto standard metrics to evaluate GANs [Brock et al., 2019, Zhang et al., 2018]. Although these metrics focus on sample quality, they are also sensitive to coverage, see Section 2.12.3 for details. To specifically evaluate the coverage of held-out data, we use the standard bits per dimension (BPD) metric, defined as the negative log-likelihood on held-out data, averaged across pixels and color channels [Dinh et al., 2017].

Due to their degenerate low-dimensional support, GANs do not define a density in the image space, which prevents measuring BPD on them. To endow a GAN with a full support and a likelihood, we train an inference network “around it”, while keeping the weights of the GAN generator fixed. We also train an isotropic noise parameter σ . For both GANs and VAEs, we use the inference network to compute a lower bound to approximate the likelihood, *i.e.*, an upper

bound on BPD.

We conduct an ablation study on the CIFAR-10 dataset with the standard split of 50k/10k train/test images of 32×32 pixels. We evaluate all metrics using held-out data not used during training, which improves over common practice in the GAN literature, where training data is often used for evaluation.

5.5.2 Ablation study and comparison to VAE and GAN baselines

Our GAN baseline uses the non-residual architecture of SNGAN [Miyato et al., 2018a], stable and quick to train, without spectral normalization. The same convolutional architecture is kept to build a VAE baseline.¹ It produces the mean of a factorizing Gaussian distribution. To ensure a valid density model we add a trainable isotropic variance σ . We train the generator for coverage by optimizing $\mathcal{L}_Q(p_\theta)$, for quality by optimizing $\mathcal{L}_C(p_\theta)$, and for both by optimizing the sum $\mathcal{L}_Q(p_\theta) + \mathcal{L}_C(p_\theta)$. The model using Variational inference with Adaptive Density Estimation (ADE) is referred to as V-ADE. The addition of adversarial training is denoted AV-ADE, and hybrid training with a Gaussian decoder as AV-GDE. The bijective function f_ψ ² increases the number of weights by approximately 1.4%, which we compensate for with a slight decrease in the width of the generator for fair comparison.³ Implementation details can be found in Section A.2.

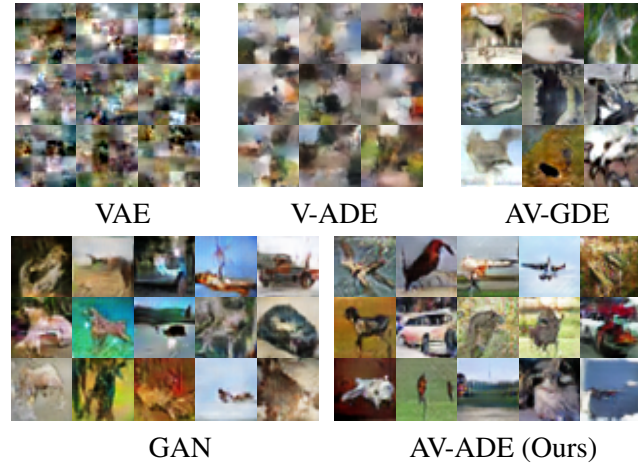


Figure 5.5: Samples from GAN and VAE baselines, our V-ADE, AV-GDE and AV-ADE models, all trained on CIFAR-10.

Experimental results in Table 5.1 confirm that the GAN baseline yields better sample quality (IS and FID) than the VAE baseline, *e.g.*, obtaining inception scores of 6.8 and 2.0, respectively. Conversely, VAE achieves better coverage, with a BPD of 4.4, compared to 7.0 for GAN, which is not

¹ In the VAE model, some intermediate feature maps are treated as conditional latent variables, allowing for hierarchical top-down sampling (see Section A.2). Experimentally, we find that similar top-down sampling is not effective for the GAN model. ² implemented as a small Real-NVP with 1 scale, 3 residual blocks, 2 layers per block. ³ This is too small to make a significant difference in experiments.

	f_ψ	Adv.	MLE	BPD ↓	IS ↑	FID ↓
GAN	×	✓	×	[7.0]	6.8	31.4
VAE	×	×	✓	4.4	2.0	171.0
V-ADE [†]	✓	×	✓	3.5	3.0	112.0
AV-GDE	×	✓	✓	4.4	5.1	58.6
AV-ADE [†]	✓	✓	✓	3.9	7.1	28.0

Table 5.1: Quantitative results. [†] : Parameter count decreased by 1.4% to compensate for f_ψ . [Square brackets] denote that the value is approximated, see Section 5.5.

far from the 8.0 BPD obtained if using a uniform distribution and suggests heavy mode-dropping. An identical generator trained for both quality and coverage, AV-GDE, obtains a sample quality that is in between that of the GAN and the VAE baselines, in line with the analysis in Section 5.4. Samples from the different models in Figure 5.5 confirm these quantitative results. Using f_ψ and training with $\mathcal{L}_C(p_\theta)$ only, denoted by V-ADE in the table, leads to improved sample quality with IS up from 2.0 to 3.0 and FID down from 171 to 112. Note that this quality is still below the GAN baseline and our AV-GDE model.

When f_ψ is used with coverage and quality driven training, AV-ADE, we obtain improved IS and FID scores over the GAN baseline, with IS up from 6.8 to 7.1, and FID down from 31.4 to 28.0. The examples shown in the figure confirm the high quality of the samples generated by our AV-ADE model. Our model also achieves a better BPD than the VAE baseline. These experiments demonstrate that our proposed bijective feature space substantially improves the compatibility of coverage and quality driven training. We obtain improvements over both VAE and GAN in terms of held-out likelihood, and improve VAE sample quality to, or beyond, that of GAN. We further evaluate our model using the recent precision and recall approach of [Sajjadi et al., 2018] in the classification framework of [Shmelkov et al., 2018] in Section 5.6.1. Additional results showing the impact of the number of layers and scales in the bijective similarity mapping f_ψ are presented in Section 5.7, and reconstructions qualitatively demonstrating the inference abilities of our AV-ADE model are presented in Section A.2.1.

5.5.3 Refinements and comparison to the state of the art

We now consider further refinements to our model, inspired by recent generative modeling literature. Four refinements are used: (i) adding residual connections to the discriminator [Gulrajani et al., 2017b] (rd), (ii) leveraging more accurate posterior approximations using inverse auto-regressive flow [Kingma et al., 2016b] (iaf); see Section A.2, (iii) training wider generators with twice as many channels (wg), and (iv) using a hierarchy of two scales to build f_ψ (s2); see Section 5.7. Table 5.2 shows consistent improvements with these additions, in terms of BPD, IS, FID.

Table 5.3 compares our model to existing hybrid approaches and state-of-the-art generative models on CIFAR-10. In the category of hybrid models that optimize likelihood, denoted by



Figure 5.6: Samples from models trained on CIFAR-10. Our AV-ADE spills less mass on unrealistic samples, owing to adversarial training which controls where off-dataset mass goes.

Refinements	BPD ↓	IS ↑	FID ↓
GAN	[7.0]	6.8	31.4
GAN (rd)	[6.9]	7.4	24.0
AV-ADE	3.9	7.1	28.0
AV-ADE (rd)	3.8	7.5	26.0
AV-ADE (wg, rd)	3.8	8.2	17.2
AV-ADE (iaf, rd)	3.7	8.1	18.6
AV-ADE (s2)	3.5	6.9	28.9

Table 5.2: Model refinements. Brackets [] denote that the values have been estimated using an inference network. (rd) denotes the use of a residual discriminator, (wg) a wide generator, (iaf) the use of inverse-autoregressive flow, and (s2) the use of a hierarchy of two scales in the invertible function. These refinements and upgrades yield consistent improvements.

Hybrid (L) in the table, FlowGAN(H) optimizes MLE and an adversarial loss, and FlowGAN(A) is trained adversarially. The AV-ADE model significantly outperforms these two variants both in terms of BPD, from 4.2 to between 3.5 and 3.8, and quality, *e.g.*, IS improves from 5.8 to 8.2. Compared to models that train an inference network adversarially, denoted by Hybrid (A), our model shows a substantial improvement in IS from 7.0 to 8.2. Note that these models do not allow likelihood evaluation, thus BPD values are absent.

Compared to adversarial models, which are not optimized for support coverage, AV-ADE obtains better FID (17.2 down from 21.7) and similar IS (8.2 for both) compared to SNGAN with residual connections and hinge-loss, despite training on 17% less data than GANs (test split removed). The improvement in FID is likely due to this measure being more sensitive to support coverage than IS. Compared to models optimized with MLE only, we obtain a BPD between 3.5 and 3.7, comparable to 3.5 for Real-NVP demonstrating a good coverage of the support of held-out data. We computed IS and FID scores for MLE based models using publicly released code, with provided parameters (denoted by [†] in the table) or trained ourselves (denoted by [‡]). Despite being smaller (for reference Glow has 384 layers VS at most 10 for our deeper generator), our AV-ADE



Figure 5.7: Samples from our AV-ADE model. Additional samples are given in Section 5.6.2.

model generates better samples, *e.g.*, IS up from 5.5 to 8.2 (samples displayed in Figure 5.6), owing to quality driven training controlling where the off-dataset mass goes.

Hybrid (L)	BPD ↓	IS ↑	FID ↓
AV-ADE (wg, rd)	3.8	8.2	17.2
AV-ADE (iaf, rd)	3.7	8.1	18.6
AV-ADE (S2)	3.5	6.9	28.9
FlowGan(A) [Grover et al., 2018]	8.5	5.8	
FlowGan(H) [Grover et al., 2018]	4.2	3.9	

Hybrid (A)	BPD ↓	IS ↑	FID ↓
AGE [Ulyanov et al., 2018]		5.9	
ALI [Dumoulin et al., 2017a]		5.3	
SVAE [Chen et al., 2018]		6.8	
α -GAN [Rosca et al., 2017]		6.8	
SVAE-r [Chen et al., 2018]		7.0	

Adversarial	BPD ↓	IS ↑	FID ↓
mmd-GAN [Arbel et al., 2018]		7.3	25.0
SNGan [Miyato et al., 2018a]		7.4	29.3
BatchGAN [Lucas et al., 2018a]		7.5	23.7
WGAN-GP [Gulrajani et al., 2017a]		7.9	
SNGAN _(R,H)		8.2	21.7

MLE	BPD ↓	IS ↑	FID ↓
NVP [Dinh et al., 2017]	3.5	4.5 [†]	56.8 [†]
VAE-IAF [Kingma et al., 2016b]	3.1	3.8 [†]	73.5 [†]
Pixcnn++ [Salimans et al., 2017b]	2.9	5.5	
Flow++ [Ho et al., 2019]	3.1		
Glow [Kingma and Dhariwal, 2018]	3.4	5.5[‡]	46.8[‡]

Table 5.3: Performance on CIFAR10, without labels. MLE and Hybrid (L) models discard the test split.

5.5.4 Results on additional datasets

To further validate our model we evaluate it on STL10 (48×48), ImageNet and LSUN (both 64×64). We use a wide generator to account for the higher resolution, without iaf, a single scale in f_{ψ} , and no residual blocks (see Section 5.5.3).

The architecture and training hyper-parameters are the same as before, besides adding one layer at resolution 64×64 , which demonstrates the stability of our approach. Quantitative results on STL10, and ImageNet are reported in Table 5.4, which shows that our AV-ADE improves

STL-10 (48×48)	BPD ↓	IS ↑	FID ↓	ImageNet (64×64)	BPD ↓	IS ↑	FID ↓
AV-ADE (wg, wd)	4.4	9.4	44.3	AV-ADE (wg, wd)	4.90	8.5	45.5
AV-ADE (iaf, wd)	4.0	8.6	52.7	Real-NVP	3.98		
AV-ADE (s2)	3.8	8.6	52.1	Glow	3.81		
NVP	3.7[‡]	4.8 [‡]	103.2 [‡]	Flow++	3.69		
BatchGAN		8.7	51	MMD-GAN		10.9	36.6
SNGAN (Res-Hinge)		9.1	40.1				

Table 5.4: Results on the STL-10 and ImageNet datasets. [‡] denotes models trained and evaluated by us using available source code



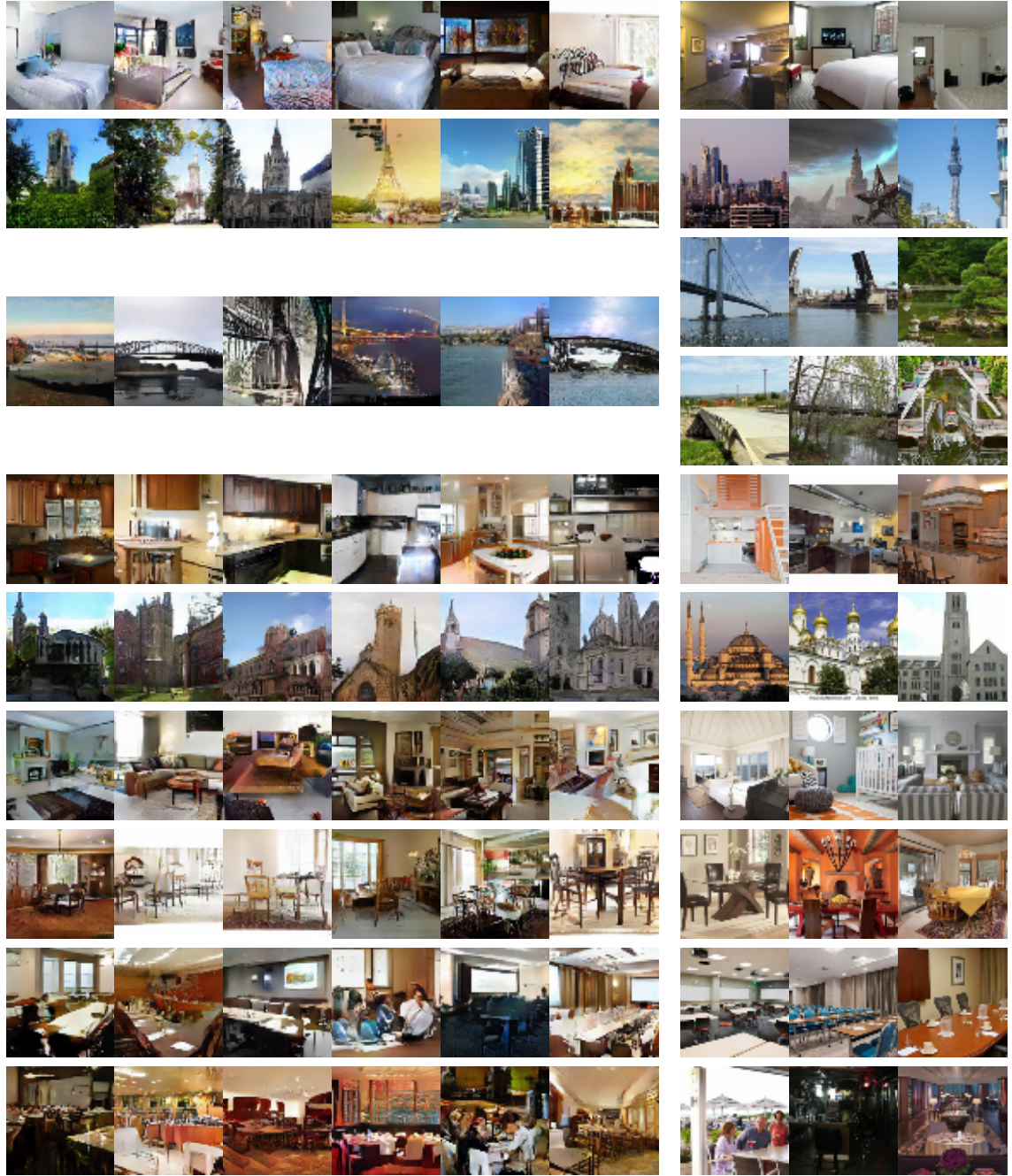
Figure 5.8: Samples from our AV-ADE model trained on STL10, compared to real images. See Figure A.1 for more samples.

inception score over SNGAN, from 9.1 up to 9.4, and is second best in FID, with corresponding samples displayed in Figure 5.8. Our likelihood performance, between 3.8 and 4.4, and close to that of NVP at 3.7, demonstrates good coverage of the support of held-out data. On the ImageNet dataset, maintaining high sample quality, while covering the full support is challenging, due to its very diverse support. Our AV-ADE model obtains a sample quality behind that of MMD-GAN with IS/FID scores at 8.5/45.5 vs. 10.9/36.6. However, MMD-GAN is trained purely adversarially and does not provide support guarantees, unlike our approach.

Figure 5.9 shows samples from our generator trained on a single GPU with 11 Gb of memory on LSUN classes. It yields compelling samples compared to those of the Glow model, despite having more flexibility (over 500 VS 7 layers) showing that Glow spills more mass outside of the training support (while our model mode drops more, hence the inferior BPD). Quantitative performance on the LSUN dataset is reported in Table 5.5. Additional samples and other LSUN categories are presented in Section 5.6.2.



Figure 5.9: Samples from models trained on LSUN Churches (C) and bedrooms (B). Our AV-ADE model over-generalises less and produces more compelling samples. See Section 5.6.2 for more classes and samples.



(a) samples

Figure 5.10: Samples (at resolution 64×64) and quantitative performance obtained by our model AV-ADE, on LSUN classes (order of appearance given by the table).

LSUN	Real-NVP	Glow	Ours
Bedroom	(2.72/×)	(2.38/208.8 [†])	(3.91, 21.1)
Tower	(2.81/×)	(2.46/214.5 [†])	(3.95, 15.5)
Church	(3.08/×)	(2.67/222.3 [†])	(4.3, 13.1)
Classroom	×	×	(4.6, 20.0)
Restaurant	×	×	(4.7, 20.5)

Table 5.5: Quantitative performance measured by (BPD, FID), on LSUN classes at resolution 64×64 .

5.6 Complementary evaluations

In this section we consider other, less standard evaluations. The first is based on the idea of training classifiers using generated data, the second using a precision-recall procedure.

5.6.1 Evaluation using samples as training data for a discriminator

We evaluate our approach using the two measures recently proposed by [Shmelkov et al., 2018]. The first measure, GAN-test, is obtained by training a classifier on natural image data and evaluating it on generated samples. It is sensitive to sample quality only. Our AV-ADE model obtain a slightly higher GAN-test score, evidencing comparable sample quality, which is in line with the results in Section 5.5.2. The second measure, GAN-train, is obtained by training a classifier on generated samples and evaluating it on natural images. Having established similar GAN-test performance, this demonstrates improved sample diversity of the AV-ADE model and shows that the coverage-driven training improves the support of the learned model.

Using these metrics requires a conditional version of AV-ADE. To address the poor compatibility of conditional batch-normalization with VAEs, we propose conditional weight normalization (CWN) (see below for details). Otherwise, the architecture is the same as in Table 5.1. The results in Table 5.6 show that our model benefits from maximum-likelihood estimation in that it obtains better coverage, as evidenced by better GAN-train performance.

model	GAN-test (%)	GAN-train (%)
GAN	71.8	29.7
AV-ADE	76.9	73.4
DCGAN [†]	58.2	65.0

Table 5.6: Performance of our model using the GAN-train and GAN-test metrics. [†]performance of DCGAN, as reported in [Shmelkov et al., 2018], for reference.

Class conditioning with conditional weight-normalization. To perform this evaluation we develop a class conditional version of our AV-ADE model. The discriminator is conditioned using the class conditioning introduced by Miyato and Koyama [2018]. GAN generators are typically made class-conditional using conditional batch normalization [De Vries et al., 2017, Dumoulin et al., 2017b], however batch normalization is known to be detrimental in VAEs [Kingma et al., 2016b], as we verified in practice. To address this issue, we propose conditional weight normalization (CWN). As in weight normalization [Salimans and Kingma, 2016], we separate the training of the scale and the direction of the weight matrix. Additionally, the scaling factor $g(y)$ of the weight matrix \mathbf{v} is conditioned on the class label y :

$$\mathbf{w} = \frac{g(y)}{\|\mathbf{v}\|} \mathbf{v}, \quad (5.8)$$

We also make the network biases conditional on the class label. Otherwise, the architecture is the same one used for the experiments in Table 5.1.

5.6.2 Model evaluation using precision and recall

In this section, we evaluate our models using the precision and recall procedure of [Sajjadi et al., 2018]. This evaluation is relevant as it seeks to evaluate coverage of the support and the quality of the samples separately, rather than aggregating them into a single score.

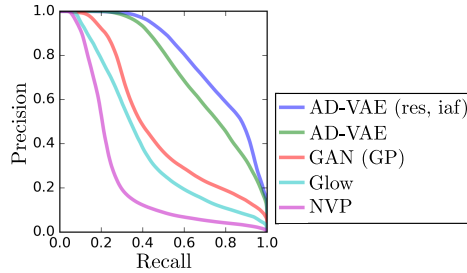


Figure 5.11: Precision-recall curves using the evaluation procedure of [Sajjadi et al., 2018].

Figure 5.11 presents the evaluation of our models in Section 5.5.2, as well as the Glow and NVP models, using the official code provided online by the authors at <https://github.com/msmsajjadi/precision-recall-distributions>. Our AV-ADE model obtains a better area under curve (AUC) than the GAN baseline, and model refinements improve AUC further. For comparison, Glow and NVP have lower AUC than both GAN and our models.

5.7 Qualitative influence of the feature space flexibility

We experiment with different architectures to implement the invertible mapping used to build the feature space as presented in Section 5.4. To assess the impact of the expressiveness of the

invertible model on the behavior of our framework, we modify various standard parameters of the architecture. Popular invertible models such as NVP [Dinh et al., 2017] readily offer the possibility of extracting latent representation at several scales, separating global factors of variations from low level detail. Thus, we experiment with varying number of scales. An other way of increasing the flexibility of the model is to change the number of residual blocks used in each invertible layer. Note that all the models evaluated so far in the main body of the paper are based on a single scale and two residual blocks, except the one denoted with (s2). In addition to our AV-ADE models, we also compare with similar models trained with maximum likelihood estimation (MLE). Models are first trained with maximum-likelihood estimation, then with both coverage and quality driven criteria.

The results in Table 5.12 (a) show that factoring out features at two scales rather than one is helpful in terms of BPD. For the AV-ADE models, however, IS and FID deteriorate with more scales, and so a tradeoff between must be struck. For the V-ADE models, the visual quality of samples also improves when using multiple scales, as reflected in better IS and FID scores. Their quality, however, remains far worse than those produced with the coverage and quality training used for the AV-ADE models. Samples in the maximum-likelihood setting are provided in Figure 5.12 (c). With three or more scales, models exhibit symptoms of overfitting: train BPD keeps decreasing while test BPD starts increasing, and IS and FID also degrade.

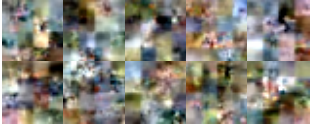
5.8 Conclusion

We presented a generative model that leverages invertible network layers to relax the conditional pixel independence assumption commonly made in VAE decoders. It allows for efficient feed-forward sampling, and can be trained using a maximum likelihood criterion that ensures coverage of the data generating distribution, as well as an adversarial criterion that ensures high sample quality. This is a step towards limiting mode-collapse in GANs, because the coverage of the support is explicitly optimized and can be evaluated. On the other hand adversarial training pushes the model to produce more compelling samples compared to purely maximum-likelihood based models, and this is achieved at little expense in terms of bits per dimension.


Scales	Blocks	BPD ↓	IS ↑	FID ↓	Scales	Blocks	BPD ↓	IS ↑	FID ↓
1	2	3.77	7.9	20.1	1	2	3.52	3.0	112.0
2	2	3.48	6.9	27.7	2	2	3.41	4.5	85.5
2	4	3.46	6.9	28.9	3	2	3.45	4.4	78.7
3	3	3.49	6.5	31.7	4	1	3.49	4.1	82.4

(a) AV-ADE models


(b) V-ADE models




No NVP



NVP 1 scale



NVP 2 scales



NVP 3 scales

(c) samples

Figure 5.12: Evaluation on CIFAR-10 of different architectures of the invertible layers of the model. In Table (a), adversarial training is used, while models in Table (b) are purely maximum-likelihood based. In (c), samples displayed were obtained using VAE models trained with MLE (Table 5.12b), showing qualitative influence of multi-scale feature space. The models include one without invertible decoder layers, and with NVP layers using one, two and three scales. The samples illustrate the impact of using invertible NVP layers in these auto-encoders.

Chapter 6

Conclusion

Contents

6.1 Summary of contributions	115
6.2 Future work	116

6.1 Summary of contributions

The work presented in this manuscript revolves around two main goals. The first is to develop models that avoid parametric assumptions, which are ill suited to natural images modelling, while remaining tractable to train. The second is to obtain generative models that can explicitly evaluate both sample quality and diversity of the learned distribution. The second goal is approached from two different angles: one purely adversarial, and one based on hybrid training. We now summarise our contributions and the extent to which these goals have been attained.

6.1.1 Going beyond conditional independence

We have presented two approaches to training generative image models that go beyond the usual conditional independence assumption. The first, presented in Chapter 3, combines a latent variable structure with an autoregressive decoder. Unlike prior approaches to such models, we use a regularization parameter and auxiliary target images to control what is modelled by latent variables and what is left to model for the autoregressive decoder. This framework avoids the information preference property, without constraining the flexibility of the autoregressive component. We obtained quantitative performance on par with the state of the art on CIFAR10 at the time of publication, and compelling samples demonstrating globally coherent structure and fine details.

While this construction goes beyond the conditional independence assumption, it does so at the cost of slow, sequential sampling which is incompatible with adversarial setting. We also presented another approach, in Chapter 5, that leverages invertible network layers to relax the conditional pixel independence assumption. It allows for efficient feed-forward sampling, and

can thus be trained using an adversarial criterion that ensures high sample quality.

6.1.2 Distributional statistics in adversarial training

In Chapter 4 we have shown how to build adversarial networks architectures that can consider the samples in a batch together rather than independently. This allows the model to approximate statistics on the variability of the learned distribution and use them to reject or accept a batch. The discriminator is thus able to explicitly evaluate the diversity of the samples produced by the generator. Assuming the discriminator has fit the distribution of batches of real images perfectly, fooling it requires two things: i) batches of samples must contain as much variability as batches of real images and ii) each image in a batch has to be of high quality. Our approach thus turns mode dropping into something that is explicitly penalised, together with image quality.

After observing that the problem is invariant to permutations on the batch axis, we proposed architectures that are able to model any function invariant to permutations, assuming sufficient capacity. We also observed that in practice, using pure batches makes the discriminator's task too easy, and generalized the approach to mixed batches. Experimentally, we showed our training method to reduce mode dropping and reach good quantitative scores compared to the GAN literature.

6.1.3 Hybrid adversarial and maximum-likelihood based training

In Chapter 5 we presented a generative model that can be trained using a maximum likelihood criterion to optimize coverage of the data generating distribution, and an adversarial criterion to optimize sample quality. This is a solution to limit mode-collapse in GANs because the coverage of the support is explicitly optimized and can be evaluated. This approach is orthogonal to discriminating at the batch level, but pursues the same goal of explicitly evaluating support coverage.

The key observation to make our hybrid model work was that usual conditional independence assumptions induce a conflict between the two losses. Learning an abstract feature space in which to compute the distance between target and prediction allowed us to avoid conditional independence without incurring slow sampling, and thus to use a hybrid criterion successfully. Our model is able to generate samples of compelling visual quality, on par with purely adversarial methods. It also has inference capabilities - real images can be mapped to the latent space - which is useful for applications such as compression. It also achieves good bits per dimension performance on unseen data, significantly improved over existing hybrid models, which is a guarantee that the support of the dataset is well covered.

6.2 Future work

It is reasonable to think that the recent progress in generative modelling makes it ripe to be used as a building block to solve other problems, and to tackle more challenging types of data. One of

the next frontiers of generative modelling is that of videos – two topics of particular interest to the author are video compression and representation learning from videos. The author believes that in these domains, performance leaps are to be expected in the near future and possible research directions are presented in sections 6.2.1 and 6.2.2. Other research areas include video generation [Clark et al., 2019] and world modelling for curiosity-driven reinforcement learning [Pathak et al., 2017]. There is also still a lot of work to do to scale image models to larger datasets or to images of higher resolutions. Several applications of existing methods to photo and video editing, smartphone photo enhancement [Ledig et al., 2017], video games and virtual environments also have plenty of room for development and improvement.

6.2.1 Video compression - Entropy coding and lossless compression

A natural application of generative models of images is video compression, which is one of the backbones of the internet. Video represents over 60 % of internet traffic¹, thus significant improvements to video compression can have a high impact. Deep models have been shown to be well suited to natural images, and could improve on traditional compression codecs, at least in terms of prediction performance. Two types of compression exist: lossy, and lossless. The cornerstone of lossless compression is entropy coding; the high-level idea is that frequent values should be associated to short codes and rare ones to long codes. The optimal transmission efficiency that can be reached when encoding a message x sampled from a distribution p is the Shannon Entropy, equal to $\mathbb{E}_{x \sim p^*}[-\log_2 p^*(x)]$. With a model p_θ that does not perfectly match the data generating distribution, the efficiency becomes:

$$\mathbb{E}_{x \sim p^*}[-\log_2 p_\theta(x)]. \quad (6.1)$$

The better the model p_θ , the closer we get to the bound. The recent progress of deep generative models holds the promise of greatly improved compression performances, as they provide the tools to get models p_θ that fit the real data distribution very well. Scaling and refining work such as [Lu et al., 2019] in this context is a promising research direction.

If x does not need to be perfectly recovered, one can extract some information about x into z , send z losslessly as above, and reconstruct x from z . To "extract information" and to "reconstruct", auto-encoders are natural candidates: x is encoded into z by a deep network g_ϕ parametrised by weights ϕ , z is sent to the receiver and decoded into $f_\theta(z)$. The metric used to evaluate the model is then a weighted sum between how good the reconstructions are, measured by \mathcal{L} , and how costly it is to send $z = g_\phi(x)$ (losslessly), using density model Q :

$$\underbrace{-\log_2[Q(g_\phi(x))]}_{\text{cost of sending } z} + \underbrace{\beta \mathcal{L}[f_\theta(g_\phi(x)), x]}_{\text{distortion}}. \quad (6.2)$$

Variational auto-encoders provide a very natural replacement to traditional tools for lossy compression of natural images and videos. Different choices for Q and for \mathcal{L} will lead to different models. Likelihood based losses such as the L_2 loss² $\mathcal{L}[f_\theta(g_\phi(x)), x] = \|f_\theta(g_\phi(x)) - x\|^2$ are one form of evaluation. Another possibility is to evaluate the quality of the image using a

¹ see the Global Internet Phenomena Report at <https://www.sandvine.com/phenomena> ² Recall that minimizing the L_2 loss is equivalent to maximizing the log-likelihood of a normal distribution

GAN discriminator D_ψ , then $\mathcal{L}_{\text{adversarial}} = -D_\psi(f_\theta(z))$. can be added to the weighted sum. The discriminator can also be used to train the model to compensate the degradation in quality caused by the lossy compression [Ledig et al., 2017].

Such models can be directly applied to videos (in which case the encoder and decoder may be recurrent, or be 3D-convolutional networks). A strategic approach may be to decompose the problem: given a sequence of frames x_1, \dots, x_n , use an image model to independently compress all images into z_1, \dots, z_n . Then, noting that the redundancy between frames is high, a model that compresses (z_1, \dots, z_n) into \tilde{z} can be built, for instance using an autoregressive model [Kalchbrenner et al., 2016]. The compressed representation can be losslessly transmitted, and each z_i decoded independently.

To summarize, this framework is suitable to lossy compression of videos, potentially at high compression rates. It can leverage many generative modelling concepts – VAEs for lossy compression, any model with a likelihood for lossless compression, and adversarial models to restore low level detail – and their interaction is a rich topic that has yet to be explored.

6.2.2 Representation learning

The spirit of self-supervised training is to use the structure naturally present inside raw data to extract a target signal used for optimisation. In the context of videos, one natural way to do that is the task of next-frame prediction [Lotter et al., 2017]: given part of a video, the model has to predict the rest. This problem can be equivalently formulated as a conditional generative modelling problem. Training this type of models on videos in an unsupervised manner can be expected to yield a powerful representation learning framework [LeCun, 2019]. Indeed, natural videos are complex and highly structured, and because there is a vast amount of data, very big models can be considered.

Given some video frames, multiple futures are usually possible which means that the output of the model needs to be a multi-modal distribution over possible futures. One approach to achieve that is to rely on latent variables, as in a VAE. These will be used to transmit information about which future to choose among possible ones, *i.e.*, which mode of the distribution to select. Using the Euclidean norm as reconstruction loss and noting y the target, x the observed frames, z the latent variable and f_θ the model, parametrised by θ , the optimisation problem being solved is then:

$$\min_{\theta} \min_z \|y - f_\theta(x, z)\|_2^2. \quad (6.3)$$

The latent variable z is here to help predict y . Only part of the signal y can be predicted from x and the other (unpredictable) part has to go through z . If no constraints are put on z , all the information contained in y can go through z making the problem trivial. So it is clear that some bottleneck on z needs to be added. It can be in the form of hard constraints for instance architectural constraints such as limiting its dimensionality, or a penalty term $R(z)$ can be added, yielding:

$$\min_{\theta} \min_z \|y - f_\theta(x, z)\|_2^2 + R(z). \quad (6.4)$$

This research direction is an other potentially impactful application of generative modelling of images. The right type of architectures and regularization is an open research topic. It can yield powerful representations of data, or a way to pre-train networks.

Appendix A

Adaptive density estimation: samples and other results

This appendix contains additional samples from our models as presented in Chapter [5](#) as well as architectural details and a detailed training algorithm.

A.1 Additional Samples

In this section we provide additional qualitative results on CIFAR10, STL10, LSUN (categories: Bedrooms, Towers, Bridges, Kitchen, Church, Living room, Dining room, Classroom, Conference room and Restaurant) at resolutions 64×64 and 128×128 , ImageNet and CelebA. We report IS/FID scores together with BPD.

A.1.1 Additional samples on CIFAR10 and STL10



Figure A.1: Samples from our AV-ADE (wg, rd) model trained on CIFAR10 compared to real images. A significant proportion of samples can be reasonably attributed to a CIFAR10 class, even though the model was trained without labels. Our model is constrained to cover the full support of the data, which translates to diverse samples, as noted in the qualitative results illustrated here.



Figure A.2: Additional Cifar Samples.

A.1.2 Samples on all Lsun datasets



Figure A.3: Samples obtained from our AV-ADE (wg, rd) model trained on LSUN bedrooms (left), compared to training images (right).

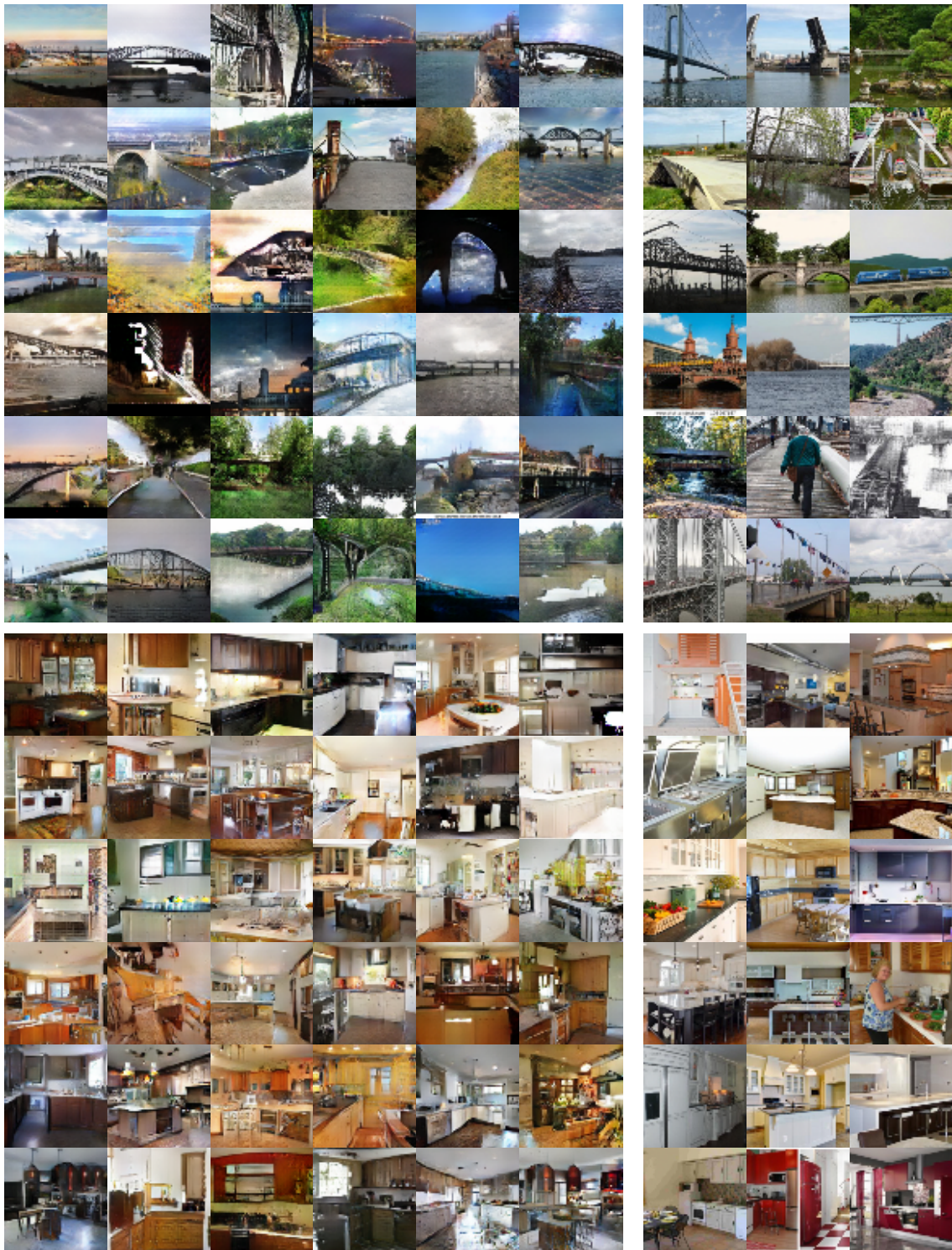


Figure A.4: Samples obtained from our AV-ADE (wg, rd) model trained on LSUN bridges (left), compared to training images (right).



Figure A.5: Samples obtained from our AV-ADE (wg, rd) model trained on LSUN churches (left), compared to training images (right).

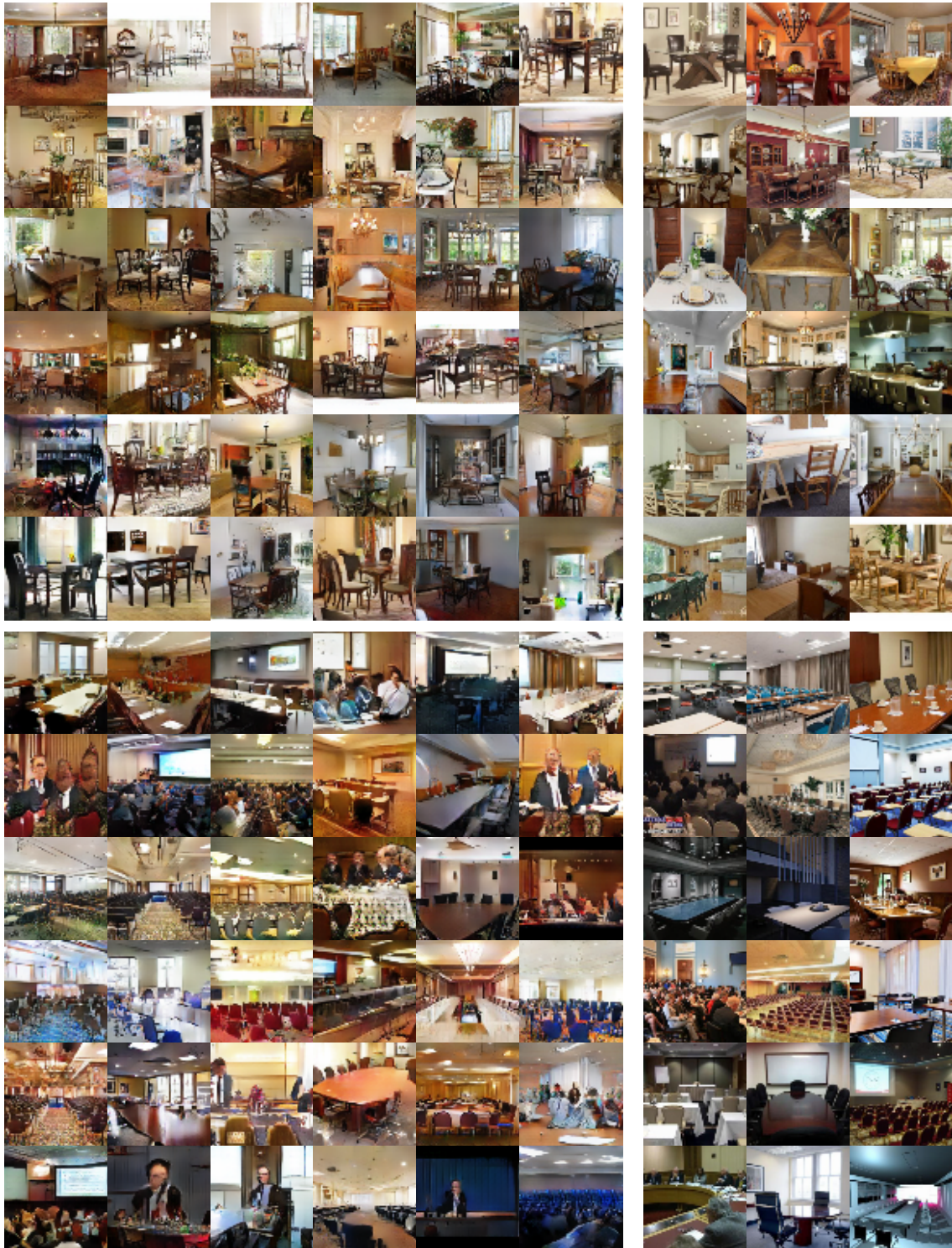


Figure A.6: Samples obtained from our AV-ADE (wg, rd) model trained on LSUN dining rooms (left), compared to training images (right).



Figure A.7: Samples obtained from our AV-ADE (wg, rd) model trained on LSUN restaurants (left), compared to training images (right).

A.2 Implementation details

Architecture and training hyper-parameters. We used Adamax [Kingma and Ba, 2015a] with learning rate 0.002, $\beta_1 = 0.9$, $\beta_2 = 0.999$ for all experiments. All CIFAR-10 experiments use batch size 64, other experiments in high resolution use batch size 32. To stabilize the adversarial training we use the gradient penalty [Gulrajani et al., 2017b] with coefficient 100, and 1 discriminator update per generator update. We experimented with different weighting coefficients between the two loss components, and found that values in the range 10 to 100 on the adversarial component work best in practice. No significant influence on the final performance of the model is observed in this range, though the training dynamics in early training are improved with higher values. With values significantly smaller than 10, discriminator collapses was observed in a few isolated cases. All experiments reported here use coefficient 100.

For experiments with hierarchical latent variables, we use 32 of them per layer. In the generator we use ELU nonlinearity, in discriminator with residual blocks we use ReLU, while in simple convolutional discriminator we use leaky ReLU with slope 0.2.

Unless stated otherwise we use three NVP layers with a single scale and two residual blocks that we train only with the likelihood loss. Regardless of the number of scales, the VAE decoder always outputs a tensor of the same dimension as the target image, which is then fed to the NVP layers. As in the reference implementations, we use both batch normalization and

weight normalization in NVP and only weight normalization in IAF. We use the reference implementations of IAF and NVP released by the authors.

Discriminator	Generator
	conv 3×3 , 16
conv 3×3 , 16	IAF block 32
ResBlock 32	IAF block down 64
ResBlock down 64	IAF block down 128
ResBlock down 128	IAF block down 256
ResBlock down 256	$h \sim \mathcal{N}(0; 1)$
Average pooling	IAF block up 256
dense 1	IAF block up 128
	IAF block up 64
	IAF block 32
	conv 3×3 , 3

Table A.1: Residual architectures for experiments from Table 5.2

A.2.1 Visualisations of reconstructions

We display reconstructions obtained by encoding and then decoding ground truth images with our models (AV-ADE from sectablation) in Figure A.8. As is typical for expressive variational autoencoders, real images and their reconstructions cannot be distinguished visually.



Figure A.8: Real images and their reconstructions with the AV-ADE models.

A.3 Coverage and Quality driven training algorithm

Algorithm 1 Coverage and Quality driven training for our AV-ADE model.

In this section we summarize the training procedure as an algorithm to give an overview of the different quantities and steps involved. **for** number of training steps **do**

- Sample m real images $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from p^* , approximated by the dataset.
- Map the real images to feature space $\{\mathbf{f}(\mathbf{x})^{(1)}, \dots, \mathbf{f}(\mathbf{x})^{(m)}\}$ using the invertible transformation f .
- Encode the feature space vectors using the VAE encoder and get parameters for the posterior $q_\phi(\mathbf{z}|\mathbf{f}(\mathbf{x}))$.
- Sample m latent variable vectors, $\{\hat{\mathbf{z}}^{(1)}, \dots, \hat{\mathbf{z}}^{(m)}\}$ from the posterior $q_\phi(\mathbf{z}|\mathbf{x})$, and m latent variable vectors $\{\tilde{\mathbf{z}}^{(1)}, \dots, \tilde{\mathbf{z}}^{(m)}\}$ from the VAE prior $p_\theta(\mathbf{z})$
- Decode both sets of latent variable vectors using the VAE decoder into the means of conditional Gaussian distributions, $\{\mu(\hat{\mathbf{z}})^{(1)}, \dots, \mu(\hat{\mathbf{z}})^{(m)}\}$ and $\{\mu(\tilde{\mathbf{z}})^{(1)}, \dots, \mu(\tilde{\mathbf{z}})^{(m)}\}$
- Sample from the Gaussian densities obtained, $\{\mathcal{N}(\cdot|\mu(\hat{\mathbf{z}})^{(i)}, \sigma I_n)\}_{i \leq m}$ and $\{\mathcal{N}(\cdot|\mu(\tilde{\mathbf{z}})^{(i)}, \sigma I_n)\}_{i \leq m}$, which yields reconstructions in feature space $\{\widehat{\mathbf{f}(\mathbf{x})}^{(i)}\}_{i \leq m}$ and samples in feature space $\{\widetilde{\mathbf{f}(\mathbf{x})}^{(i)}\}_{i \leq m}$
- Map the samples and reconstructions back to image space using the inverse of the invertible transformation f^{-1} which yields reconstructions $\{\hat{\mathbf{x}}^{(i)}\}_{i \leq m}$ and samples $\{\tilde{\mathbf{x}}^{(i)}\}_{i \leq m}$
- Compute $L_C(p_\theta)$ using ground truth images $\{\mathbf{x}^{(i)}\}_{i \leq m}$ and their reconstructions $\{\hat{\mathbf{x}}^{(i)}\}_{i \leq m}$
- Compute $L_Q(p_\theta)$ by feeding the ground truth images $\{\mathbf{x}^{(i)}\}_{i \leq m}$ together with the sampled images $\{\tilde{\mathbf{x}}^{(i)}\}_{i \leq m}$ to the discriminator
- Optimize the discriminator by gradient descent to bring L_Q closer to L_Q^*
- Optimize the generator by gradient descent to minimize $L_Q + L_C$

end for

Appendix B

Areas of attention for image captioning

Note. The work presented in this appendix, conducted during my research internship and the first semester of my PhD, is separated from the main body of the manuscript because the subject differs substantially from the other contributions. It approaches the topic of image captioning and involves recurrent networks, attention mechanisms and natural language processing. Image captioning, not unlike image generation, requires high level representations of complex image content.

We propose “Areas of Attention”, a novel attention-based model for automatic image captioning. Our approach models the dependencies between image regions, caption words, and the state of an RNN language model, using three pairwise interactions. In contrast to previous attention-based approaches that associate image regions only to the RNN state, our method allows a direct association between caption words and image regions. During training these associations are inferred from image-level captions, akin to weakly-supervised object detector training. These associations help to improve captioning by localizing the corresponding regions during testing. We also propose and compare different ways of generating attention areas: CNN activation grids, object proposals, and spatial transformers nets applied in a convolutional fashion. Spatial transformers give the best results. They allow for image specific attention areas, and can be trained jointly with the rest of the network. Our attention mechanism and spatial transformer attention areas together yield state-of-the-art results on the MSCOCO dataset.

B.1 Introduction

Image captioning, *i.e.* automatically generating natural language image descriptions, is useful for the visually impaired, and for natural language based image search. It is significantly more challenging than classic vision tasks such as object recognition and image classification for two reasons. First, the structured output space of well formed natural language sentences is significantly more challenging to predict over than just a set of class labels. Second, this complex output space allows a finer interpretation of the visual scene, and therefore also requires a more detailed visual analysis of the scene to do well at this task. Figure B.1(top) gives an example of a typical image description that not only refers to objects in the scene, but also the scene type or

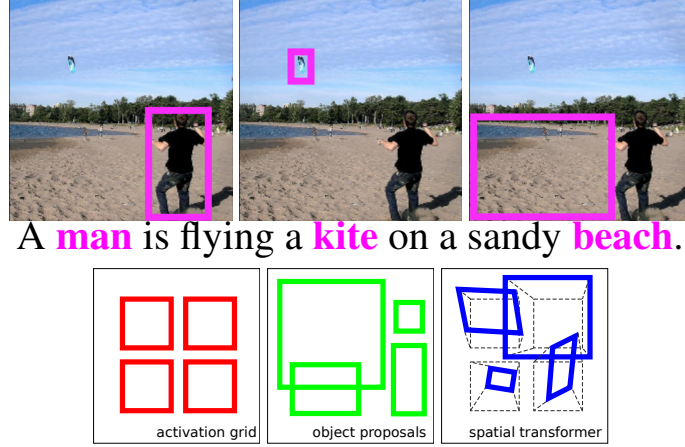


Figure B.1: We propose an attention mechanism that jointly predicts the next caption word and the corresponding region at each time-step given the RNN state (top). Besides implementing our model using attention areas defined over CNN activation grids or object proposals, as used in previous work, we also present a end-to-end trainable convolutional spatial transformer approach to compute image specific attention areas (bottom).

location, object properties, and their interactions.

Neural encoder-decoder based approaches, similar to those used in machine translation [Sutskever et al. \[2014\]](#), have been found very effective for this task, see *e.g.* [Kiros et al. \[2014\]](#), [Mao et al. \[2015\]](#), [Vinyals et al. \[2015\]](#). These methods use a convolutional neural network (CNN) to encode the input image into a compact representation. A recurrent neural network (RNN) is used to decode this representation word-by-word into a natural language description of the image. While effective, these models are limited in that the image analysis is (i) static, *i.e.* does not change over time as the description is produced, and (ii) not spatially localized, *i.e.* describes the scene as a whole instead of focusing on local aspects relevant to parts of the description. Attention mechanisms can address these limitations by dynamically focusing on different parts of the input as the output sequence is generated. Such mechanisms are effective for a variety of sequential prediction tasks, including machine translation [Bahdanau et al. \[2015\]](#), speech recognition [Chorowski et al. \[2015\]](#), image synthesis [Gregor et al. \[2015\]](#), and image captioning [Xu et al. \[2015\]](#). For some tasks the definition of parts of the input to attend to are clear and limited in number: for example the individual words in the source sentence for machine translation. For other tasks with complex inputs, such as image captioning, the notion of parts is less clear. In this work we propose a novel attention model and three different ways to select parts of the image, or areas of attention, for the automatic generation of image captions.

The first contribution of our work is a new attention mechanism that models the interplay between the RNN state, image region descriptors, and word embedding vectors by means of three pairwise interactions. Previous attention approaches model either only the interaction between image regions and RNN state [Jin et al. \[2015\]](#), [Xu et al. \[2015\]](#), or the interaction between

regions and words but with an external representation that is learned off-line, *e.g.* pre-trained object detectors [Fang et al. \[2015\]](#), [Wu et al. \[2016\]](#), [You et al. \[2016\]](#). In contrast, our attention representation explicitly considers, in a single end-to-end trainable system, the direct interaction among caption words, image regions and RNN state. At each time-step, our model jointly predicts the next caption word and the associated image region. Similar to weakly-supervised object localization, the associations between image regions and words are inferred during training from image-level captions. Our experimental results show that our three pair-wise interactions clearly improve the attention focus and the quality of the generated sentences.

Our second contribution is to integrate a localization sub-network in our model —similar to spatial transformer networks [Jaderberg et al. \[2015\]](#), but applied in a convolutional fashion— that regresses a set of attention areas from the image content. Earlier attention-based image captioning models used the positions in the activation grid of a CNN layer as attention areas, see *e.g.* [Xu et al. \[2015\]](#); such regions are not adaptive to the image content. Others have used object proposals as attention regions, see *e.g.* [Jin et al. \[2015\]](#), in which case the regions are obtained by an external mechanism, such as Edge boxes [Zitnick and Dollár \[2014\]](#), that is not trained jointly with the rest of the captioning system.

Our third contribution is a systematic experimental study of the effectiveness of these three different areas of attention using a common attention model, see Figure [B.1](#)(bottom). To the best of our knowledge we are the first to present such a comparison. Our experimental results show that the use of image-specific areas of attention is important for improved sentence generation. In particular, our spatial-transformer based approach is a good choice: it outperforms the other approaches, while using fewer regions and not requiring an external proposal mechanism. Using our proposed attention mechanism and the spatial transformer attention areas together we obtain state-of-the-art performance on the MSCOCO dataset.

B.2 Related work

Image captioning with encoder-decoder models has recently been extensively studied, see *e.g.* [Bengio et al. \[2015\]](#), [Donahue et al. \[2015\]](#), [Karpathy and Fei-Fei \[2015\]](#), [Kiros et al. \[2014\]](#), [Mao et al. \[2015\]](#), [Ranzato et al. \[2016\]](#), [Vinyals et al. \[2015\]](#), [Xu et al. \[2015\]](#), [Yang et al. \[2016\]](#). In its basic form a CNN processes the input image to encode it into a vectorial representation, which is used as the initial input for an RNN. Given the previous word, the RNN sequentially predicts the next word in the caption without the need to restrict the temporal dependence to a fixed order, as in approaches based on n-grams. The CNN image representation can be entered into the RNN in different manners. While some authors [Karpathy and Fei-Fei \[2015\]](#), [Vinyals et al. \[2015\]](#) use it only to compute the initial state of the RNN, others enter it in each RNN iteration [Donahue et al. \[2015\]](#), [Mao et al. \[2015\]](#).

[Xu et al. \[2015\]](#) were the first to propose an attention-based approach for image captioning, in which the RNN state update includes the visual representation of an image region. Which image region is attended to is determined based on the previous state of the RNN. They propose a “soft” variant in which a convex combination of different region descriptors is used,

and a “hard” variant in which a single region is selected. The latter is found to perform slightly better, but is more complex to train due to a non-differentiable sampling operator in the state update. In their approach the positions in the activation grid of a convolutional CNN layer is the loci of attention. Each position is described with the corresponding activation column across the layer’s channels.

Several works build upon the approach of Xu *et al.* Xu *et al.* [2015]. You *et al.* You *et al.* [2016] learn a set of attribute detectors, similar to Fang *et al.* Fang *et al.* [2015], for each word of their vocabulary. These detectors are applied to an image, and the strongest object detections are used as regions for an attention mechanism similar to that of Xu *et al.* Xu *et al.* [2015]. In their work the detectors are learned prior and independently from the language model. Wu *et al.* Wu *et al.* [2016] also learn attribute detectors but manually merge word tenses (*walking*, *walks*) and plural/singulars (*dog*, *dogs*) to reduce the set of attributes. Jin *et al.* Jin *et al.* [2015] explore the use of selective search object proposals Uijlings *et al.* [2013] as regions of attention. They resize the regions to a fixed size and use the VGG16 Simonyan and Zisserman [2015] penultimate layer to characterize them. Yang *et al.* Yang *et al.* [2016] improve the attention based encoder-decoder model by adding a reviewer module that improves the representation passed to the decoder. They show improved results for various tasks, including image captioning. Yao *et al.* Yao *et al.* [2015] use a temporal version of the same mechanism to adaptively aggregate visual representations across video frames per word for video captioning. Yeung *et al.* Yeung *et al.* use a similar temporal attention model for temporal action localization.

Visual grounding of natural language expressions is a related problem Karpathy and Fei-Fei [2015], Rohrbach *et al.* [2016], which can be seen as an extension of weakly supervised object localization Bilen and Vedaldi [2016], Cinbis *et al.* [2014], Russakovsky *et al.* [2012]. The goal is to localize objects referred to by natural language descriptions, while only using image-level supervision. Since the goal in visual grounding and weakly supervised localization is precise localization, methods typically rely on object proposal regions which are specifically designed to align well with object boundaries Uijlings *et al.* [2013], Zitnick and Dollár [2014]. Instead of localizing a given textual description, our approach uses image-level supervision to infer a latent correspondence between the words in the caption and image regions.

Object proposal methods were designed to focus computation of object detectors on a selective set of image regions likely to contain objects. Recent state-of-the-art detectors, however, integrate the object proposal generation and recognition into a single network. This is computationally more efficient and leads to more accurate results Liu *et al.* [2016], Ren *et al.* [2015]. Johnson *et al.* Johnson *et al.* [2016] use similar ideas for the task of localized image captioning, which predicts semantically relevant image regions together with their descriptions. In each region, they generate descriptions with a basic non-attentive image captioning model similar to the one used by Vinyals *et al.* Vinyals *et al.* [2015]. They train their model from a set of bounding-boxes with corresponding captions per image. In our work we do not exploit any bounding-box level supervision, we instead infer the latent associations between caption words and image regions. We propose a convolutional variant of the spatial transformer network of Jaderberg *et al.* Jaderberg *et al.* [2015], to place the attention areas in an image-adaptive manner. This module is trained in an integrated end-to-end manner with the rest of our captioning model.

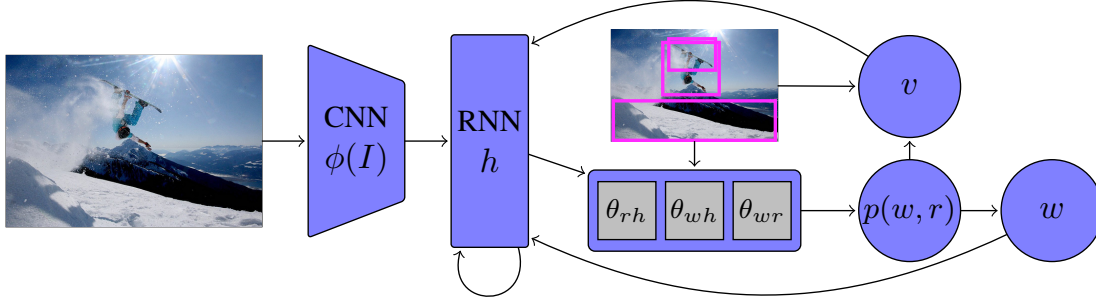


Figure B.2: In our attention-based model the conditional joint distribution, $p(w, r|h)$, over words and regions given the current state h is used to generate a word and to pool region descriptors in a convex combination. Both are then fed back to update the state at the next time-step.

Compared to previous attention models [Jin et al. \[2015\]](#), [Xu et al. \[2015\]](#), [Yang et al. \[2016\]](#), [You et al. \[2016\]](#), our attention mechanism, consisting of a single interaction layer, is less complex yet improves performance. Our approach models a joint distribution over image regions and caption words, generalizing weakly supervised localization methods and RNN language models. It includes a region-word interaction found in weakly supervised localization, as well as a word-state interaction found in RNN language models. In addition, our model includes a region-state interaction which forms a dynamic appearance-based salience mechanism. Our model naturally handles different types of attention regions (fixed grid, object proposals, and spatial transformers), and is applicable to all tasks where attention can model joint distributions between parts of the input data and output symbols. To the best of our knowledge, we propose the first trainable image-adaptive method to define attention regions, and present the first systematic comparison among different region types for attention-based image captioning in a single model.

B.3 Attention in encoder-decoder captioning

In Section [B.3.1](#) we describe our baseline encoder-decoder model. We extend this baseline in Section [B.3.2](#) with our attention mechanism in a way that abstracts away from the underlying region types. In Section [B.3.3](#) we show how we integrate regions based on CNN activation grids, object proposals, and spatial transformers networks in our model.

B.3.1 Baseline CNN-RNN encoder-decoder model

Our baseline encoder-decoder model uses a CNN to encode an image I into a vectorial representation $\phi(I) \in \mathbb{R}^{d_I}$, which is extracted from a fully connected layer of the CNN. The image encoding $\phi(I)$ is used to initialize the state of an RNN language model. Let h_t denote the RNN state vector at time t , then $h_0 = \theta_{hi}\phi(I)$, where $\theta_{hi} \in \mathbb{R}^{d_h \times d_I}$ linearly maps $\phi(I)$ to the RNN state space of dimension d_h .

The distribution over w_t , the word at time t , is given by a logistic regression model over the

RNN state vector,

$$p(w_t|h_t) \propto \exp\left(w_t^\top W \theta_{wh} h_t\right), \quad (\text{B.1})$$

where $w_t \in \{0, 1\}^{n_w}$ is a 1-hot coding over the captioning vocabulary of n_w words, W is a matrix which contains word embedding vectors as rows, and θ_{wh} maps the word embedding space to the RNN state space. For sake of clarity, we omit the dependence on I in Eq. (B.1) and below.

We use an RNN based on gated recurrent units (GRU) [Chung et al. \[2014\]](#), which are simpler than LSTM units [Hochreiter and Schmidhuber \[1997\]](#), while we found them to be at least as effective in preliminary experiments. Abstracting away from the GRU internal gating mechanism (see supplementary material), the state update function is given by a non-linear deterministic function

$$h_{t+1} = g(h_t, W^\top w_t). \quad (\text{B.2})$$

The feedback of w_t in the state update makes that w_{t+1} recursively depends on both $\phi(I)$ and the entire sequence of words, $w_{1:t} = (w_1, \dots, w_t)$, generated so far.

During training we minimize the sum of losses induced by pairs of images I_m with corresponding captions $w_{1:l_m}$,

$$\sum_m L(I_m, w_{1:l_m}, \theta) = - \sum_m \sum_{t=1}^{l_m} \ln p(w_t|h_t, \theta), \quad (\text{B.3})$$

where θ collectively denotes all parameters of the CNN and RNN component. This amounts to approximate maximum likelihood estimation, due to local minima in the loss.

Once the model is trained, captions for a new image can be generated by sequentially sampling $w_t \sim p(w_t|h_t)$, and updating the state $h_{t+1} = g(h_t, w_t)$. Since determining the maximum likelihood sequence is intractable, we resort to beam search if a single high-scoring caption is required.

B.3.2 Attention for prediction and feedback

In the baseline model the image is used only to initialize the RNN, assuming that the memory of the recurrent net is sufficient to retain the relevant information of the visual scene. We now extend the baseline model with a mechanism to attend to different image regions as the caption is generated word-by-word. Inspired by weakly supervised object localization methods, we score region-word pairs and aggregate these scores by marginalization to obtain a predictive distribution over the next word in the caption. The advantage is that this model allows words to be associated with specific image region appearances instead of global image representations, which leads to better generalization to recognize familiar scene elements in novel compositions. Importantly, we maintain the word-state interaction in Eq. (B.1) of the baseline model, to ensure temporal coherence in the generated word sequence by recursive conditioning on all previous words. Finally, a region-state interaction term allows the model to highlight and suppress image regions based on their appearance and the state, implementing a dynamic salience mechanism.

See Figure B.2 for a schematic illustration of our model.

We define a joint distribution, $p(w_t, r_t|h_t)$, over words w_t and image regions r_t at time t given the RNN state h_t . The marginal distribution over words, $p(w_t|h_t)$, is used to predict the next word at every time-step, while the marginal distribution over regions, $p(r_t|h_t)$, is used to provide visual feedback to the RNN state update. Let $r_t \in \{0, 1\}^{n_r}$ denote a 1-hot coding of the index of the region attended to among n_r regions at time t . We write the state-conditional joint distribution on words and regions as

$$p(w_t, r_t|h_t) \propto \exp s(w_t, r_t, h_t), \quad (\text{B.4})$$

$$s(w_t, r_t, h_t) = w_t^\top W \theta_{wh} h_t + w_t^\top W \theta_{wr} R^\top r_t + r_t^\top R \theta_{rh} h_t + w_t^\top W \theta_w + r_t^\top R \theta_r, \quad (\text{B.5})$$

where R contains the region descriptors in its rows. The score function $s(w_t, r_t, h_t)$ is composed of three bi-linear pairwise interactions. The first scores state-word combinations, as in the baseline model. The second scores the compatibility between words and region appearances, as in weakly supervised object localization. The third scores region appearances given the current state, and acts as a dynamic salience term. The last two unary terms implement linear bias terms for words and regions respectively.

Given the RNN state, the next word in the image caption is predicted using the marginal word distribution, $p(w_t|h_t) = \sum_{r_t} p(w_t, r_t|h_t)$, which replaces Eq. (B.1) of the baseline model. The baseline model is recovered for $R = 0$.

In addition to using the image regions to extend the state-conditional word prediction model, we also use them to extend the feedback connections of the RNN state update. We use a mechanism related to the soft attention model of Xu *et al.* [2015]. We compute a convex combination of region descriptors which will enter into the state-update. In contrast to Xu *et al.*, we derive the region weights from the joint distribution defined above. In particular, we use the marginal distribution over regions, $p(r_t|h_t) = \sum_{w_t} p(w_t, r_t|h_t)$, to pool the region descriptors as

$$v_t = \sum_{r_t} p(r_t|h_t) r_t^\top R = p_{rh}^\top R, \quad (\text{B.6})$$

where $p_{rh} \in \mathbb{R}^{n_r}$ stacks all region probabilities at time t . This visual representation is concatenated to the generated word in the feedback signal of the state update, *i.e.* we replace the update of Eq. (B.2) of the baseline model with

$$h_{t+1} = g(h_t, [w_t^\top W \ v_t^\top]^\top). \quad (\text{B.7})$$

In Section B.4, we experimentally assess the importance of the different pairwise interactions, and the use of the attention mechanism in the state update.

B.3.3 Areas of attention

Our attention mechanism presented above is agnostic to the definition of the attention regions. In this section we describe how to integrate three types of regions in our model.

Activation grid. For the most basic notion of image regions we follow the approach of Xu *et al.* [2015]. In this case the regions of attention correspond to the $z = x \times y$ spatial positions in the activation grid of a CNN layer $\gamma(I)$ with c channels. The region descriptors in the rows of $R \in \mathbb{R}^{z \times c}$ are given by the activations corresponding to each one of the z locations of the activation grid. In this case, the receptive fields for the regions is the same as all regions have a fixed shape and size, independent of the image content.

Object proposals. To obtain attention regions that adapt to the image content, we consider the use of object detection proposals, similar to the approach of Jin *et al.* [2015]. We expect such regions to be more effective since they tend to focus on scene elements such as (groups of) objects, and their parts. In particular we use edge-boxes [Zitnick and Dollár, 2014], and max-pool the activations in a CNN layer $\gamma(I)$ over each object proposal to obtain a set of fixed-size region descriptors. To ensure a high-enough resolution of the CNN layer which allows to pool activations for small proposals, we use a separate CNN which processes the input image at a higher resolution than the one used for the global image representation $\phi(I)$. This is similar to Girshick [2015], He *et al.* [2014], but we pool to a single cell instead of using a spatial pyramid. This is more efficient and did not deteriorate performance, as compared to using a pyramid. In this case the number of proposals is not limited by the number of positions in the activation tensor of the CNN layer that is accessed for the region descriptors.

Spatial transformers. We propose a third type of attention region that has not been used in existing attention-based captioning models. It is inspired by recent object detectors and localized image captioning methods with integrated the region proposal networks [Johnson *et al.* [2016], Liu *et al.* [2016], Ren *et al.* [2015]]. In contrast to the latter methods, which rely on bounding-box annotations to learn the region proposal network, we only use image captions for training. Therefore, we need a mechanism that allows back-propagation of the gradient of the captioning loss w.r.t. the region coordinates and the features extracted using them. To this end we use a bilinear sampling approach as in [Jaderberg *et al.* [2015], Johnson *et al.* [2016]]. In contrast to the max-pooling we use for proposals, it enables differentiation w.r.t. the region coordinates.

Our approach is illustrated in Figure B.3. Given an activation map $\gamma(I)$, we use a localization network that consists of two convolutional layers to locally regress an affine transformation $A \in \mathbb{R}^{2 \times 3}$ for each location of the feature map. With each location of the activation map $\gamma(I)$ we associate an “anchor box”, which is centered at that position and covers 3×3 activations. The affine transformations, computed at each location in a convolutional fashion, are applied to the coordinates of the anchor boxes. Locally a 3×3 patch is bilinearly interpolated from $\gamma(I)$ over the area of the transformed anchor box. A 3×3 filter is then applied to the locally extracted patches to compute the region descriptor, which has the same number of dimensions as the activation tensor $\gamma(I)$ has channels. If the local transformations leave the anchor boxes unchanged, then this reduces to the activation grid approach.

As we have no bounding-box annotations, training the spatial transformer can get stuck at poor local minima. To alleviate this issue, we initialize the network with a model that was trained

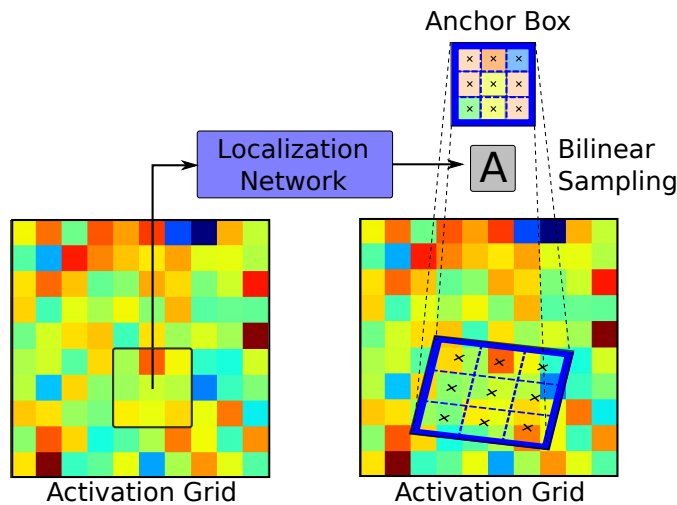


Figure B.3: For our spatial transformer network attention areas, the localization network regresses affine transformations for all feature map positions in a convolutional manner, which are applied to the anchor boxes that are used to re-sample the feature map.

using activation grids. We initialize the transformation layers to produce affine transformations that scale the anchor boxes to twice their original size, to move away from the local optimum of the activation grid model.

B.4 Experimental evaluation

We define the experimental setup in Section B.4.1, and present the experimental results in Section B.4.2.

B.4.1 Experimental setup and implementation details

Dataset and evaluation metrics. For most of our experiments we use the MSCOCO dataset Lin et al. [2014a]. It consists of around 80K training images and 40K development images. Each image comes with five descriptive captions, see Figure B.5 for example images. For sake of brevity we only report the most commonly used metrics, BLEU4, METEOR, and CIDEr-D, in the main paper. BLEU 1, 2 and 3 metrics can be found in the supplementary material. Similar to previous work Wu et al. [2016], Xu et al. [2015], Yang et al. [2016] we use 5K development images to validate the training hyper-parameters based on CIDEr-D and another 5K development images to measure performance. Finally, we also use the visual entity annotations of Plummer *et al.* Plummer et al. [2015] to assess the extent to which the attention model focuses on objects or their context.

CNN image encoder. We use the penultimate layer of the VGG16 architecture Simonyan and Zisserman [2015] to extract the global image representation $\phi(I)$ that initializes the RNN

Method	B4	Meteor	CIDEr
Baseline: θ_{wh}	26.4	22.2	78.9
Ours: θ_{wh}, θ_{wr}	28.0	22.9	83.6
Ours: $\theta_{wh}, \theta_{wr}, \theta_{rh}$	28.4	23.3	85.5
Ours: conditional feedback	28.7	23.7	86.8
Ours: full model	28.8	23.7	87.4

Table B.1: Evaluation of the baseline and our attention model using activation grid regions, including variants with certain components omitted, and word-conditional instead of marginal feedback.

state. The “activation grid” regions are taken from the last convolutional layer. For the “spatial transformer” regions, we use the penultimate convolutional layer to regress the transformations, which are then applied to convolve a locally transformed version of the same layer. For the “object proposal” regions we max-pool features from the last convolutional layer. Similar to [Ren et al. \[2015\]](#), we re-scale the image so that the smaller image dimension is 300 pixels while keeping the original aspect-ratio. When fine-tuning we do not share the parameters of the two CNNs. In all cases, the dimension of the region descriptors is given by the number of channels in the corresponding CNN layer, *i.e.* $d_r = 512$.

Captioning vocabulary. We use all 6,325 unique words in the training captions that appear at least 10 times. Words that appear less frequently are replaced by a special OUT-OF-VOCABULARY token, and the end of the caption is marked with a special STOP token. The word embedding vectors of dimension $d_w = 512$ collected in the matrix W are learned along with the RNN parameters.

Training. We use RNNs with a single layer of $d_h = 512$ GRU units. We found it useful to train our models in two stages. In the first stage, we use pre-trained CNN weights obtained from the ImageNet 2010 dataset [Deng et al. \[2009\]](#). In the second stage, we also update the CNN parameters. We use the Adam stochastic gradient descend algorithm [Kingma and Ba \[2015b\]](#). To speed-up training, we sub-sample the 14×14 convolutional layers to 7×7 when using the activation grid and the spatial transformer regions. For proposal regions, each time we process an image we use 50 randomly selected regions.

B.4.2 Experimental results

In this section we assess the relative importance of different components of our model, the effectiveness of the different types of attention regions, and the effect of jointly fine-tuning the CNN and RNN components. Finally, we compare our results to the state of the art.

Attention and visual feedback. In Table B.1 we progressively add components of our model to the baseline system. Here we use activation grid regions for our attention model. Adding all components improves the CIDEr score of the baseline, 78.9, by 8.5 points to 87.4. The baseline

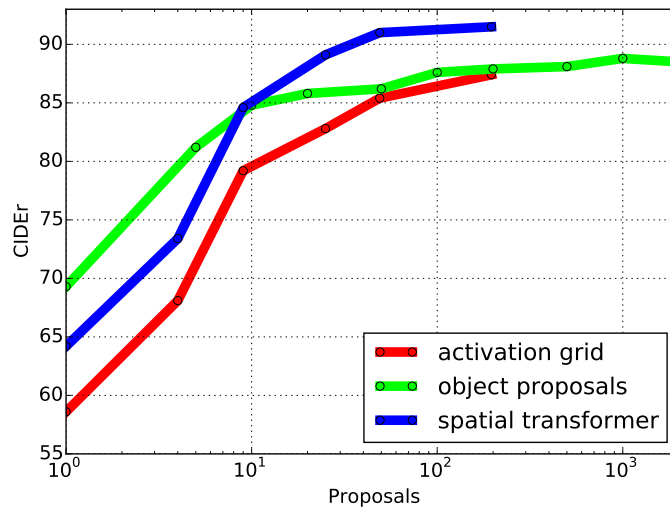


Figure B.4: Image captioning performance in CIDEr-D as a function of the number of regions. Note the log-scale on the horizontal axis.

RNN uses only word-state interaction terms to predict the next word given the RNN state. Adding the word-region interaction term (second row) improves the CIDEr metric by 4.7 points to 83.6. This demonstrates the significance of localized visual input to the RNN. As in weakly-supervised object detection, the model learns to associate caption terms to local appearances. Adding the third pairwise interaction term between regions and the RNN state (third row) brings another improvement of 1.9 points to 85.5 CIDEr. This shows that the RNN is also able to implement a dynamic salience mechanism that favors certain regions over others at a given time-step by scoring the compatibility between the RNN state and the region appearance. Finally we add the visual feedback mechanism to our model (87.4, last row), which drives the CIDEr-D score further up by 1.9 points. We also experimented with a word-conditional version of the visual feedback mechanism (86.8, last but one row), which uses $p(r_t|w_t, h_t)$ instead of $p(r_t|h_t)$ to compute the visual feedback. Although this also improves the CIDEr-D score, as compared to not using visual feedback, it is less effective than using the marginal distribution weights. The visualizations in Figure B.5 suggest that the reason for this is that the marginal distribution already tends to focus on a single semantically meaningful area.

Comparing areas of attention. In our next set of experiments we compare the effectiveness of different attention regions in our model. In Figure B.4 we consider the performance of the three region types as a function of the number of regions that are used when running the trained model on test images. For activation grids and spatial transformers the number of regions are regularly sampled from the original 14×14 resolution using increasing strides. For instance, using a stride of 2 generates $7 \times 7 = 49$ regions. For object proposals we test a larger range, from 1 up to 2,000 regions, sorted by their “objectness” score. For all three region types, performance quickly increases with the number of regions, and then plateaus off. Using four or less regions yields results below the baseline model, probably because strong sub-sampling at

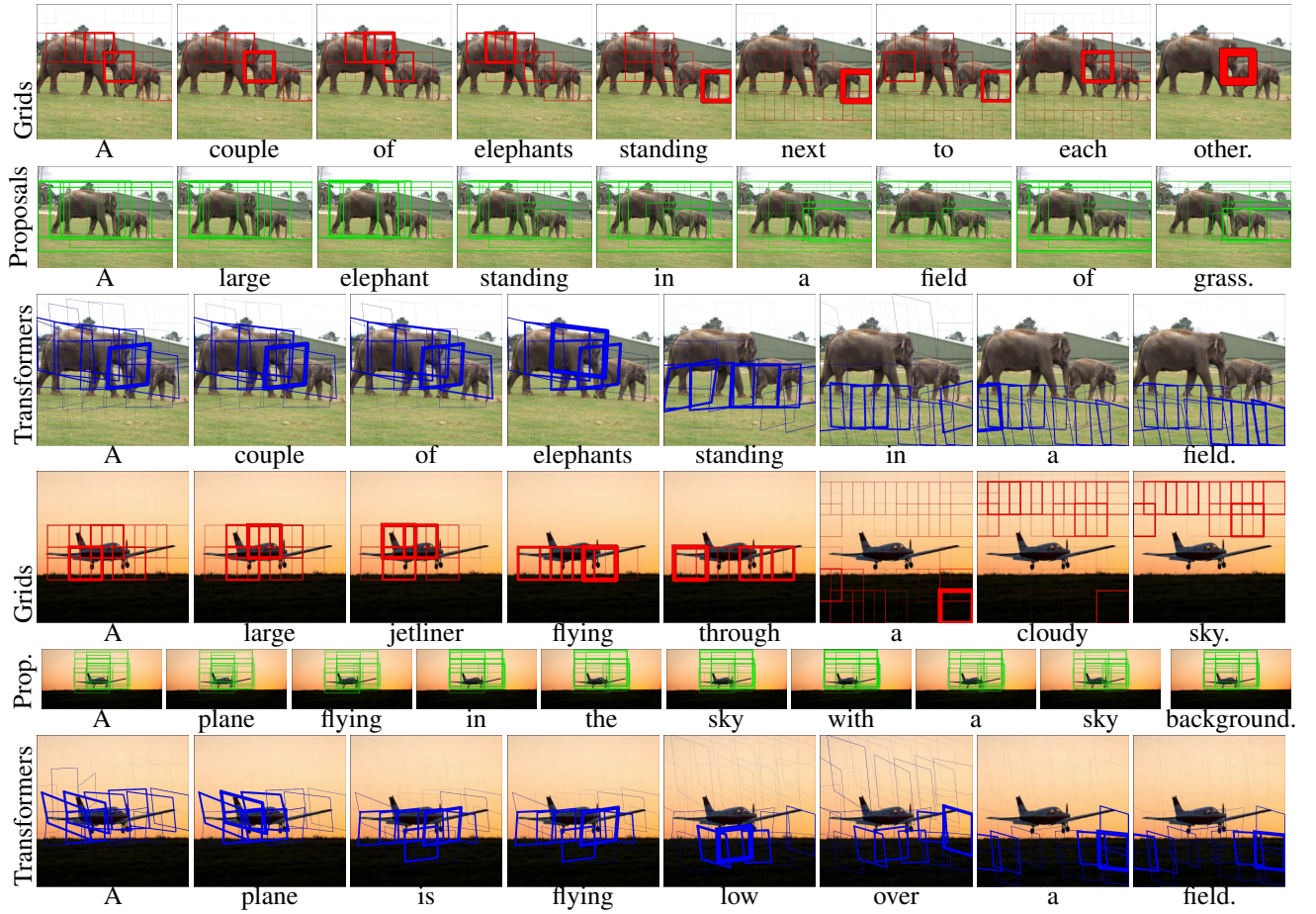


Figure B.5: Visualization of the focus of our attention model during sequential word generation for the three different region types: activation grids, object proposals, and spatial transformers. The attention areas are drawn with line widths directly proportional to weights $p(r_t|h_t)$.

	B4	Meteor	CIDEr
RNN training only			
Baseline	26.4	22.2	78.9
Activation grid	28.8	23.6	87.4
Object proposals	28.9	23.7	89.0
Spatial transformers	30.2	24.2	91.1
CNN-RNN fine-tuning			
Baseline	28.7	23.5	87.1
Activation grid	30.3	24.5	92.6
Object proposals	30.1	24.5	93.7
Spatial transformers	30.7	24.5	93.8

Table B.2: Captioning performance of the baseline and our model using different attention regions, with and without fine tuning.

test-time is sub-optimal for models trained using 7×7 or 50 regions. The spatial transformer regions consistently improve over the activation grid ones, demonstrating the effectiveness of the region transformation sub-network. As compared to object proposals, the spatial transformer regions yield better results, while also being computationally more efficient: taking only 18ms to process an image using 7×7 regions, as compared to 352ms for 50 proposals which is dominated by 320ms needed to compute the proposals. At 6ms per image, fixed 7×7 activation grids are even more efficient, but come with less accurate results. In the remaining experiments, we report performance with the optimal number of regions per method: 1,000 for proposals, and 196 for grids and transformers.

Joint CNN-RNN fine-tuning. We now consider the effect of jointly fine-tuning the CNN and RNN components. In Table B.2 we report the performance with and without fine-tuning for each region type, as well as the baseline performance for reference. All models are significantly improved by the fine-tuning. The baseline improves the most in absolute terms, but its performance remains substantially behind that of our attention models. The two types of image-dependent attention regions improve over fixed activation grids, but the differences between them are reduced after fine-tuning. Spatial transformer regions lead to comparable results as edge-box object proposals, that were designed to align with object boundaries. Spatial transformer regions, however, are more appealing from a modeling perspective since the region module is trainable fully end-to-end and does not rely on an external image processing pipeline, while also being more efficient to compute.

Visualizing areas of attention. In Figure B.5 we provide a qualitative comparison of the attentive focus using different regions in our model. A larger selection, including failure cases, can be found in the supplementary material. We show the attention weights over the image regions at each point in the generated sentences. For the spatial transformers, we show the transformed anchor boxes. For the activation grid regions, we show the back-projection of a 3×3 activation block, which allows for direct comparison with the spatial transformers. Note that in all cases the underlying receptive fields are significantly larger than the depicted areas. For object proposals

Method	GT	Gen.
Liu <i>et al.</i> Liu et al. [2017]	38.4	52.0
Liu <i>et al.</i> Liu et al. [2017], spatial superv.	43.3	57.9
Areas of Attention, MSCOCO	42.4	68.5
Areas of Attention, Flickr30k	40.2	61.1

Table B.3: Attention correctness for ground truth (GT) and generated (Gen.) sentences on the Flickr30k test set.

we directly show the edge-box proposals. The images displayed for the object proposals differ slightly from the others, since the high-resolution network used in that case applies a different cropping and scaling scheme. Proposals accurately capture objects, *e.g.* the elephants and the plane, but in other cases regions for background elements are missing, *e.g.* for the field and the sky. The spatial transformers tend to focus quite well on relational terms. For example, “*standing*” focuses on the area around the legs of the elephants in the first image, and “*low*” on the area between the airplane and the ground in the second image. For the spatial transformers in particular, the focus of attention tends to be stable across meaningful sub-sequences, such as noun phrases (*e.g.* “*A couple of elephants*”) and verb phrases (*e.g.* “*is flying.*”).

Attention correctness. We follow the approach of Liu *et al.* Liu et al. [2017] to quantitatively assess the alignment of attention with image regions corresponding to the generated caption words. Their approach uses the visual entity annotations on the Flickr30k dataset by Plummer *et al.* Plummer et al. [2015]. For caption words that are associated with a ground-truth image region, they integrate the attention values over that region. See Liu *et al.* Liu et al. [2017] for more details. Following the protocol of Liu *et al.*, we measured the attention correctness of our model (based on spatial transformer regions) on MSCOCO for ground truth and generated sentences. As Liu *et al.* reported results with a model trained on Flickr30k, for a fairer comparison, we have also trained a model on Flickr30k using the same hyper-parameters and architecture as for MSCOCO. In terms of caption generation the model obtained a CIDEr of 41.3 and a BLEU4 of 22.2. As shown in Table B.3, when considering the correctness computed on the ground truth sentences, both our models perform better than Liu *et al.* using the attention model of Xu *et al.* Xu et al. [2015], and come close to their model trained with additional spatial supervision. However, when evaluating the attention correctness on the generated sentences, our models perform significantly better than those in Liu *et al.*, including those trained with spatial supervision.

Comparison to the state of the art. We compare our results obtained using the spatial transformer regions to the state of the art in Table B.4; we refer to our method as “Areas of Attention”. We obtain state-of-the-art results on par with Wu *et al.* Wu et al. [2016]. They use a region-based high-level attribute representation instead of a global CNN image descriptor to condition the RNN language model. This approach is complementary to ours. For sake of comparability, we also ensemble our model and compare to ensemble results in the bottom part of Table B.4. For our ensemble, we trained using 30K additional validation images on top of the 80K training images, and use a random horizontal flip of the images during training. We use the same 5K validation images and 5K images for reporting as in the other experiments. We obtain state-of-the-art results, on par with Bengio *et al.* Bengio et al. [2015]. They used “scheduled

	B4	Meteor	CIDEr
Xu <i>et al.</i> Xu et al. [2015], soft	24.3	23.9	—
Xu <i>et al.</i> Xu et al. [2015], hard	25.0	23.0	—
Yang <i>et al.</i> Yang et al. [2016]	29.0	23.7	88.6
Jin <i>et al.</i> Jin et al. [2015]	28.2	23.5	83.8
Donahue <i>et al.</i> Donahue et al. [2015]	30.0	24.2	89.6
Bengio <i>et al.</i> Bengio et al. [2015]	30.6	24.3	92.1
Wu <i>et al.</i> Wu et al. [2016]	31	26	94
Areas of Attention	30.7	24.5	93.8
Ensemble methods			
Vinyals <i>et al.</i> Vinyals et al. [2015]	27.7	23.7	85.5
You <i>et al.</i> You et al. [2016]	30.4	24.3	—
Bengio <i>et al.</i> Bengio et al. [2015]	32.3	25.4	98.7
Areas of Attention	31.9	25.2	98.1

Table B.4: Comparison of our results to the state of the art on the MSCOCO dataset.

sampling”, a modified RNN training algorithm that samples from the generated words during training. With standard training, as for our results, they report 95.7 CIDEr.

B.5 Conclusion

In this paper we made three contributions. (i) We presented a novel attention-based model for image captioning. Our model builds upon the recent family of encoder-decoder models. It is based on a score function that consists of three pairwise interactions between the RNN state, image regions, and caption words. (ii) We presented a novel region proposal network to derive image-specific areas of attention for our captioning model. Our region proposal network is based on a convolutional variant of spatial transformer networks, and is trained without bounding-box supervision. (iii) We evaluated our model with three different region types based on CNN activation grids, object proposals, and our region proposal network. Our extensive experimental evaluation shows the importance of all our model components, as well as the importance of image-adaptive attention regions. This work is a first step towards weakly-supervised learning of objects and relations from captions, *i.e.* short sentences describing the content of an image. Future work will improve these associations for example by training object and relation detectors based on them. We release an open source Theano-Lasagne based implementation of our model: <https://github.com/marcopede/AreasOfAttention>

Acknowledgment. We thank NVIDIA for donating GPUs used in this research. This work was partially supported by the grants ERC Allegro, ANR-16-CE23-0006, and ANR-11-LABX-0025-01.

Bibliography

- M. Arbel, D. J. Sutherland, M. Binkowski, and A. Gretton. On gradient regularizers for mmd gans. 2018.
- M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.
- P. Bachman. An architecture for deep, hierarchical generative models. In *NIPS*, 2016.
- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
- P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 1989.
- T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 1763.
- S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015.
- Y. Bengio, A. C. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, 2012.
- Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *PAMI*, 35(8):1798–1828, 2013.
- H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016.
- C. Bishop. *Pattern recognition and machine learning*. Springer-Verlag, 2006.
- A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.
- Y. Burda, R. Salakhutdinov, and R. Grosse. Importance weighted autoencoders. In *ICLR*, 2016.
- L. Chen, S. Dai, Y. Pu, E. Zhou, C. Li, Q. Su, C. Chen, and L. Carin. Symmetric variational autoencoder and connections to adversarial learning. In *AISTATS*, 2018.

- X. Chen, D. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. In *ICLR*, 2017.
- J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *NIPS*, 2015.
- J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning Workshop*, 2014.
- R. Cinbis, J. Verbeek, and C. Schmid. Multi-fold MIL training for weakly supervised object localization. In *CVPR*, 2014.
- A. Clark, J. Donahue, and K. Simonyan. Efficient video generation on complex datasets. *arXiv*, 2019.
- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.
- A. P. Dawid and M. Musio. Theory and applications of proper scoring rules. 2014.
- H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *NIPS*, 2017.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *NIPS*, 2015.
- L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear independent components estimation. In *ICLR*, 2015.
- L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *ICLR*, 2017.
- C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. *ICCV*, 2015.
- J. Donahue, L. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.
- J. Donahue, P. Krhenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017.
- G. Dorta, S. Vicente, L. Agapito, N. Campbell, and I. Simpson. Structured uncertainty prediction networks. In *CVPR*, 2018.
- A. Dosovitskiy, J. T. Springenberg, M. A. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. *CoRR*, 2014.

- V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. In *ICLR*, 2017a.
- V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017b.
- G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.
- M. Elfeki, C. Couprie, M. Riviere, and M. Elhoseiny. GDPP: Learning diverse generations using determinantal point processes. In *arxiv.org/pdf/1812.00068*, 2018.
- H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. Platt, C. Zitnick, and G. Zweig. From captions to visual concepts and back. In *CVPR*, 2015.
- J. D. Fauw, S. Dieleman, and K. Simonyan. Hierarchical autoregressive image models with auxiliary decoders. *arXiv*, 2019.
- M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: Masked autoencoder for distribution estimation. In *ICML*, 2015.
- R. Girshick. Fast R-CNN. In *ICCV*, 2015.
- X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. *AISTATS*, 2011.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.
- K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, 2015.
- K. Gregor, F. Besse, D. Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression. In *NIPS*, 2016.
- A. Grover, M. Dhar, and S. Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *AAAI Press*, 2018.
- I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of Wasserstein GANs. In *NIPS*, 2017a.
- I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017b.
- 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 *ICLR*, 2017c.
- N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. *CoRR*.

- K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.
- 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 *NIPS*, 2017.
- J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *ICML*, 2019.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- D. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, pp. 1098-1102, 1952.
- M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.
- J. Jin, K. Fu, R. Cui, F. Sha, and C. Zhang. Aligning where to see and what to tell: image caption with region-based attention and scene factorization. *arXiv:1506.06272*, 2015.
- J. Johnson, A. Karpathy, and L. Fei-Fei. DenseCap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016.
- N. Kalchbrenner, A. van den Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. *arXiv*, 2016.
- A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.
- T. Karras, T. Aila, and S. L. abd J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.
- T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019.
- D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015a.
- D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015b.
- D. Kingma and M. Welling. Auto-encoding variational Bayes. In *ICLR*, 2014a.
- D. Kingma, D. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.
- D. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *NIPS*, 2016a.
- D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. 2018.

- D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *ICLR*, 2014b.
- D. P. Kingma, T. Salimans, R. Józefowicz, X. Chen, I. Sutskever, and M. Welling. Improving variational autoencoders with inverse autoregressive flow. In *NIPS*, 2016b.
- R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *ICML*, 2014.
- A. Kolesnikov and C. Lampert. PixelCNN models with auxiliary variables for natural image modeling. In *ICML*, 2017.
- A. Krizhevsky. Learning multiple layers of features from tiny images. Master’s thesis, University of Toronto, 2009.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.
- H. Larochelle and I. Murray. The neural autoregressive distribution estimator. 2011.
- A. Larsen, S. Sonderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.
- Y. LeCun. Energy-based approaches to representation learning, 2019. URL https://www.math.ias.edu/files/special_year_workshops/lecun-20191016-ias.pdf.
- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998.
- C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CVPR*, 2017.
- Y. Li, K. Swersky, and R. S. Zemel. Generative moment matching networks. In *ICML*, 2015.
- T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014a.
- T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: common objects in context. *CoRR*, 2014b.
- Z. Lin, A. Khetan, G. Fanti, and S. Oh. PacGAN: The power of two samples in generative adversarial networks. 2018.
- O. Litany, A. Bronstein, M. Bronstein, and A. Makadia. Deformable shape completion with graph convolutional autoencoders. In *CVPR*, 2018.
- C. Liu, J. Mao, F. Sha, and A. Yuille. Attention correctness in neural image captioning. In *AAAI*, 2017.

- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.
- J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.
- W. Lotter, G. Kreiman, and D. D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *CVPR*, 2017.
- G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao. Dvc: An end-to-end deep video compression framework. *CVPR*, 2019.
- T. Lucas and J. Verbeek. Auxiliary guided autoregressive variational autoencoders. In *ECML*, 2018a.
- T. Lucas and J. Verbeek. Auxiliary guided autoregressive variational autoencoder. *ECML*, 2018b.
- T. Lucas, C. Tallec, Y. Ollivier, and J. Verbeek. Mixed batches and symmetric discriminators for GAN training. In *ICML*, 2018a.
- T. Lucas, C. Tallec, J. Verbeek, and Y. Ollivier. Mixed batches and symmetric discriminators for gan training. *ICML*, 2018b.
- T. Lucas, K. Shmelkov, K. Alahari, C. Schmid, and J. Verbeek. Adaptive density estimation for generative models. *NeurIPS*, 2019.
- A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *ICLR*, 2016.
- J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-RNN). *ICLR*, 2015.
- M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.
- S. McGregor. Neural network processing for multiset data. In *Proceedings of the 17th International Conference on Artificial Neural Networks*.
- S. McGregor. Further results in multiset processing with neural networks. *Neural Networks*, 21(6):830–837, 2008. doi: 10.1016/j.neunet.2008.06.020. URL <https://doi.org/10.1016/j.neunet.2008.06.020>.
- J. Menick and N. Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *ICLR*, 2019.
- T. Miyato and M. Koyama. cGANs with projection discriminator. In *ICLR*, 2018.
- T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018a.

- T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*, 2018b. URL <https://openreview.net/forum?id=B1QRgzIT->. accepted as oral presentation.
- M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, 2016.
- S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.
- A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.
- D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. *ICML*, 2017.
- M. Pedersoli, T. Lucas, C. Schmid, and J. Verbeek. Areas of attention for image captioning. *ICCV*, 2017.
- B. Plummer, L. Wang, C. Cervantes, J. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, 2015.
- C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL <http://arxiv.org/abs/1612.00593>.
- A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.
- P. Ramachandran, T. L. Paine, P. Khorrami, M. Babaeizadeh, S. Chang, Y. Zhang, M. A. Hasegawa-Johnson, R. H. Campbell, and T. S. Huang. Fast generation for convolutional autoregressive models. In *ICLR workshop*, 2017.
- M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. In *ICLR*, 2016.
- A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *NIPS*. 2015.
- A. Razavi, A. van den Oord, B. Poole, and O. Vinyals. Preventing posterior collapse with delta-vaes. In *ICLR*, 2019a.
- A. Razavi, A. van den Oord, and O. Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *NeurIPS*, 2019b.
- S. Reed, A. van den Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, D. Belov, and N. de Freitas. Parallel multiscale autoregressive density estimation. In *ICML*, 2017.

- S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015.
- D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.
- D. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.
- A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele. Grounding of textual phrases in images by reconstruction. In *ECCV*, 2016.
- M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.
- S. Roweis. EM Algorithms for PCA and SPCA. In *NIPS*, 1997.
- O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *ECCV*, 2012.
- M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. 2018.
- T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.
- T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016a.
- T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016b.
- T. Salimans, A. Karpathy, X. Chen, and D. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017a.
- T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017b.
- T. Salimans, H. Zhang, A. Radford, and D. Metaxas. Improving gans using optimal transport. In *ICLR*, 2018.
- C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, vol 27, pp. 379-423, Jul. 1948, 1948.
- C. E. Shannon. Prediction and entropy of printed english. *Bell Syst. Tech. J.*, vol 30, pp. 35-4, Jan. 1951, 1951.
- K. Shmelkov, C. Schmid, and K. Alahari. How good is my GAN? In *ECCV*, 2018.

- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, , and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.
- K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.
- C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *NIPS*, 2016.
- C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised MAP inference for image super-resolution. In *ICLR*, 2017.
- I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.
- C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015.
- L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *ICLR*, 2016.
- M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.
- J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.
- D. Ulyanov, A. Vedaldi, and V. Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *AAAI Press*, 2018.
- A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *SSW*, 2016a.
- A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016b.
- A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *NIPS*, 2016c.
- A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *NeurIPS*, 2017.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*. 2017a.

- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. 2017b.
- C. Villani. Optimal transport, old and new. *Grundlehren der mathematischen Wissenschaften*, Springer, 2009.
- O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.
- O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, Y. Wu, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, , and D. Silver. Alphastar: Mastering the real-time strategy game starcraft ii. 2019.
- Q. Wu, C. Shen, L. Liu, A. Dick, and A. van den Hengel. What value do explicit high level concepts have in vision to language problems? In *CVPR*, 2016.
- K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.
- X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016.
- Z. Yang, Y. Yuan, Y. Wu, R. Salakhutdinov, and W. Cohen. Encode, review, and decode: Reviewer module for caption generation. In *NIPS*, 2016.
- L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015.
- S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos.
- Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In *CVPR*, 2016.
- M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep sets. *NIPS*, 2017.
- H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.
- H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.
- S. Zhao, J. Song, and S. Ermon. InfoVAE: Information maximizing variational autoencoders. *arXiv*, abs/1706.02262, 2017.

C. Zitnick and P. Dollár. Edge boxes: locating object proposals from edges. In *ECCV*, 2014.