Bayesian plug and play methods for inverse problems in imaging

Mario Esteban Gonzalez Olmedo

Université de Paris
Laboratoire de Mathématiques Appliquées à Paris 5 (MAP5-UMR 8145)
École doctorale de Sciences Mathématiques de Paris Centre (ED386)

Universidad de la República
Facultad de Ingeniería
Instituto de Ingeniería Eléctrica

Bayesian Plug & Play Methods for Inverse Problems in Imaging

Par Mario González Olmedo

Thèse de Doctorat de Mathématiques Appliquées
(Université de Paris)
Tesis de Doctorado en Ingeniería Eléctrica
(Universidad de la República)

Dirigée par Andrés Almansa
et par Pablo Musé

Présentée et soutenue publiquement le 15 décembre 2021

Andrés Almansa	DR, CNRS & Université de Paris	Directeur de thèse
Jean-François Aujol	PU, Université de Bordeaux	Examinateur
Pierre Chainais	PU, Ecole Centrale de Lille	Rapporteur
Julie Delon	PU, Université de Paris	Examinatrice
Ricardo Fraiman	PU, Universidad de la República	Examinateur
Pablo Musé	PU, Universidad de la República	Directeur de thèse
Pablo Sprechmann	Ingénieur de Recherche, Deep Mind	Examinateur
Gabriele Steidl	PU, Technische Universität Berlin	Rapportrice

Acknowledgements

Although the PhD degree is awarded to one person, the effort required to achieve it is shared among the members of a team. For this, I would like to thank all the people without whom this work would not have been possible.

I am very grateful to my thesis supervisors, Andrés Almansa and Pablo Musé for trusting me and dedicating countless hours in my academic training. As excellent scientists, they have taught me by example how to conduct quality research. If I deserve to call myself a researcher today, it is mainly because of them. I also thank the reviewers and other members of the committee for taking the time to read the manuscript, participate in the defense of the thesis, and for giving encouraging opinions on our work.

Nine years ago, my family and I decided to move from Montevideo to Salto to work in the *Departmento de Matemática y Estadística del Litoral (DMEL, Universidad de la República*). We have no intention of returning to Montevideo, largely because of the reception we had here. My colleagues have helped me to feel very comfortable and have covered my teaching assignments the times that I needed to travel to Paris several times during the completion of the PhD. Many thanks to them for allowing me to work there, and I hope I can give back with my work all that I received over the years.

It is not easy to adapt to a different culture than the one one is used to since birth, especially when the language is not known beforehand. But my colleagues from the *Université de Paris* during the first year of my PhD (2017-2018) have made me feel like one of them from the first day, and they have patiently helped me to learn French (at least a little). The very stimulating environment that there is there made me feel at home despite being thousands of kilometers away from home. *Merci beaucoup à vous tous d'avoir été si gentils avec moi.*

Finally, the most important thanks goes to my family for their unconditional support. My wife Dalma and my son Samuel have been alone for several weeks while I traveled to Paris on the last years. Without your patience and sacrifice, I would not have been able to dedicate the time necessary to meet the requirements for this achievement. And I thank my mother for doing everything she could so that my brother and I could study and work on what we are passionate about. My deepest thanks to you all.

Abstract

This thesis deals with Bayesian methods for solving ill-posed inverse problems in imaging with learnt image priors. The first part of this thesis (Chapter 3) concentrates on two particular problems, namely joint denoising and decompression and multi-image super-resolution. After an extensive study of the noise statistics for these problem in the transformed (wavelet or Fourier) domain, we derive two novel algorithms to solve this particular inverse problem. One of them is based on a multi-scale self-similarity prior and can be seen as a transform-domain generalization of the celebrated non-local bayes algorithm to the case of non-Gaussian noise. The second one uses a neural-network denoiser to implicitly encode the image prior, and a splitting scheme to incorporate this prior into an optimization algorithm to find a MAP-like estimator.

The second part of this thesis concentrates on the Variational AutoEncoder (VAE) model and some of its variants that show its capabilities to explicitly capture the probability distribution of high-dimensional datasets such as images. Based on these VAE models, we propose two ways to incorporate them as priors for general inverse problems in imaging:

- The first one (Chapter 4) computes a joint (space-latent) MAP estimator named Joint Posterior Maximization using an Autoencoding Prior (JPMAP). We show theoretical and experimental evidence that the proposed objective function satisfies a weak bi-convexity property which is sufficient to guarantee that our optimization scheme converges to a stationary point. Experimental results also show the higher quality of the solutions obtained by our JPMAP approach with respect to other non-convex MAP approaches which more often get stuck in spurious local optima.

- The second one (Chapter 5) develops a Gibbs-like posterior sampling algorithm for the exploration of posterior distributions of inverse problems using multiple chains and a VAE as image prior. We show how to use those samples to obtain MMSE estimates and their corresponding uncertainty.

Keywords: Inverse problems, Bayesian statistics, image processing, optimization

Resumen

En esta tesis se estudian métodos bayesianos para resolver problemas inversos mal condicionados en imágenes usando distribuciones a priori entrenadas. La primera parte de esta tesis (Capítulo 3) se concentra en dos problemas particulares, a saber, el de eliminación de ruido y descompresión conjuntos, y el de superresolución a partir de múltiples imágenes. Después de un extenso estudio de las estadísticas del ruido para estos problemas en el dominio transformado (wavelet o Fourier), derivamos dos algoritmos nuevos para resolver este problema inverso en particular. Uno de ellos se basa en una distribución a priori de autosimilitud multiescala y puede verse como una generalización al dominio wavelet del célebre algoritmo Non-Local Bayes para el caso de ruido no Gaussiano. El segundo utiliza un algoritmo de eliminación de ruido basado en una red neuronal para codificar implícitamente la distribución a priori de las imágenes y un esquema de relajación para incorporar esta distribución en un algoritmo de optimización y así encontrar un estimador similar al MAP.

La segunda parte de esta tesis se concentra en el modelo Variational AutoEncoder (VAE) y algunas de sus variantes que han mostrado capacidad para capturar explícitamente la distribución de probabilidad de conjuntos de datos en alta dimensión como las imágenes. Basándonos en estos modelos VAE, proponemos dos formas de incorporarlos como distribución a priori para problemas inversos genéricos en imágenes:

- El primero (Capítulo 4) calcula un estimador MAP conjunto (espacio imagen y latente) llamado *Joint Posterior Maximization using an Autoencoding Prior (JPMAP)*. Mostramos evidencia teórica y experimental de que la función objetivo propuesta satisface una propiedad de biconvexidad débil que es suficiente para garantizar que nuestro esquema de optimización converge a un punto estacionario. Los resultados experimentales también muestran la mayor calidad de las soluciones obtenidas por nuestro enfoque JPMAP con respecto a otros enfoques MAP no convexos que a menudo se atascan en mínimos locales espurios.

- El segundo (Capítulo 5) desarrolla un algoritmo de muestreo tipo Gibbs para la exploración de la distribución a posteriori de problemas inversos utilizando múltiples cadenas y un VAE como distribución a priori. Mostramos cómo usar esas muestras para obtener estimaciones de MMSE y su correspondiente incertidumbre.

Palabras Clave: Problemas inversos, estadística Bayesiana, procesamiento de imágenes, optimización

Résumé

Cette thèse traite des méthodes bayésiennes pour résoudre des problèmes inverses mal posés en imagerie avec des distributions a priori d'images apprises. La première partie de cette thèse (Chapitre 3) se concentre sur deux problèmes particuliers, à savoir le débruitage et la décompression conjoints et la super-résolution multi-images. Après une étude approfondie des statistiques de bruit pour ces problèmes dans le domaine transformé (ondelettes ou Fourier), nous dérivons deux nouveaux algorithmes pour résoudre ce problème inverse particulie. L'un d'eux est basé sur une distributions a priori d'auto-similarité multi-échelle et peut être vu comme une généralisation du célèbre algorithme de Non-Local Bayes au cas du bruit non gaussien. Le second utilise un débruiteur de réseau de neurones pour coder implicitement la distribution a priori, et un schéma de division pour incorporer cet distribution dans un algorithme d'optimisation pour trouver un estimateur de type MAP.

La deuxième partie de cette thèse se concentre sur le modèle Variational AutoEncoder (VAE) et certaines de ses variantes qui montrent ses capacités à capturer explicitement la distribution de probabilité d'ensembles de données de grande dimension tels que les images. Sur la base de ces modèles VAE, nous proposons deux manières de les incorporer comme distribution a priori pour les problèmes inverses généraux en imagerie:

- Le premier (Chapitre 4) calcule un estimateur MAP conjoint (espace-latent) nommé Joint Posterior Maximization using an Autoencoding Prior (JPMAP). Nous montrons des preuves théoriques et expérimentales que la fonction objectif proposée satisfait une propriété de bi-convexité faible qui est suffisante pour garantir que notre schéma d'optimisation converge vers un point stationnaire. Les résultats expérimentaux montrent également la meilleure qualité des solutions obtenues par notre approche JPMAP par rapport à d'autres approches MAP non convexes qui restent le plus souvent bloquées dans des minima locaux.

- Le second (Chapitre 5) développe un algorithme d'échantillonnage a posteriori de type Gibbs pour l'exploration des distributions a posteriori de problèmes inverses utilisant des chaînes multiples et un VAE comme distribution a priori. Nous montrons comment utiliser ces échantillons pour obtenir des estimations MMSE et leur incertitude correspondante.

Mots-clés: Problèmes inverses, statistiques bayésiennes, traitement d'images, optimisation.

Long résumé

Motivation

Les problèmes inverses sont omniprésents en science et en ingénierie. En général, il y a un événement (cause) qui déclenche des données observables (effet), et il faut extraire des informations sur cet événement à partir des données observées. Dans ce cas, le problème direct consiste à générer des données à partir de l'événement et le problème inverse est celui de récupérer l'événement à partir des données, ou du moins autant que possible. Par exemple, supposons que nous ayons besoin d'obtenir des informations sur la structure intérieure de la Terre. Au lieu de prélever des échantillons de sol coûteux, on peut essayer d'approcher cette structure à partir de mesures prises en surface. En médecine moderne, il est important de disposer d'outils d'assistance spécialisés qui renseignent le plus possible sur l'état du patient. Idéalement, ces méthodes devraient être non invasives. Cette exigence rend nécessaire d'obtenir une approximation de l'état réel de l'organisme à partir de mesures externes.

Dans certains cas, un problème inverse particulier peut conduire à un algorithme d'inversion directe. Malheureusement, les erreurs de discrétisation et le bruit présent dans les mesures peuvent conduire à des erreurs de reconstruction importantes. Dans d'autres cas, des mesures incomplètes ne sont pas suffisantes pour déterminer de manière unique le signal souhaité, nous devons donc envisager un large éventail de solutions compatibles avec les données. Pour faire face à ces problèmes dits mal posés on peut prendre plus de mesures afin de réduire l'ambiguïté des solutions. Par exemple, nous pouvons prendre une rafale d'images au lieu d'une seule, ou nous pouvons combiner plusieurs sources d'informations (appelées fusion de capteurs). Cependant, il est parfois souhaitable de prélever le moins d'échantillons possible car ils sont coûteux (en termes de temps ou d'argent) ou causent des dommages (par exemple, irradiation du corps du patient au scanner à rayons X).

Pour forcer les propriétés souhaitées sur le signal reconstruit, il est impératif de développer des méthodes qui réduisent ou pèsent l'espace de solution. A cet effet, différentes méthodes de régularisation ont été proposées afin d'intégrer des informations a priori sur la solution recherchée puis d'estimer avec précision le signal inconnu:

- L'approche variationnelle consiste en une large classe de méthodes de reconstruction basées sur l'optimisation. Fondamentalement, on construit une fonction objectif qui pénalise à la fois un terme de fidélité des données et un terme de régularisation. Ce dernier terme pénalise les images invraisemblables avec de grandes valeurs, favorisant ainsi des propriétés souhaitables sur la solution. Pour calculer la solution, une procédure de minimisation itérative est appliquée.

- Les réseaux de neurones (NN) sont un sous-ensemble d'algorithmes d'apprentissage automatique inspirés du cerveau humain. Récemment, des algorithmes de reconstruction basés sur NN ont été proposés, qui mappent les données directement à une estimation du signal inconnu. Dans la plupart des problèmes d'imagerie difficiles, les méthodes basées sur NN surpassent considérablement les méthodes variationnelles précédentes.

- Grâce à la modélisation statistique, nous pouvons définir notre croyance a priori sur le signal inconnu sous la forme d'une distribution de probabilité, appelée distribution a priori. Cette distribution, combinée aux données observées, donne lieu aux méthodes dites Bayésiennes. La distribution postérieure résultante contient des informations importantes sur le signal inconnu. En particulier, nous pouvons calculer différentes estimations ponctuelles telles que les estimateurs MAP ou MMSE, mais aussi effectuer des estimations d'incertitude sous forme d'intervalles de confiance.

Défis

Les méthodes variationnelles sont intéressantes car la même méthode de régularisation peut être utilisée pour différents problèmes inverses en ne changeant que le terme de fidélité des données. Cependant, la question principale est de savoir comment choisir le terme de régularisation. En imagerie, un choix courant pour ce terme est la semi-norme de variation totale (TV) qui favorise des gradients parcimonieux sur l'image restaurée, ce qui est cohérent avec le fait que la plupart des images naturelles sont lisses par morceaux. Un autre choix classique est de favoriser la parcimonie des coefficients d'image dans un espace de représentation. Par exemple, il peut s'agir de la représentation des patchs d'images sur un dictionnaire appris, ou d'une décomposition en ondelettes. En raison de la nature complexe des images naturelles, ces termes de régularisation fabriqués à la main ne caractérisent pas complètement toutes les propriétés que nous voulons voir dans la solution calculée. De plus, nous devons être prudents dans le choix des termes convexes afin que le problème d'optimisation résultant puisse être résolu. Ces limitations affectent la qualité des reconstructions, ce qui conduit souvent à des artefacts comme l'effet d'escalier lors de l'utilisation de la régularisation TV. De plus, les algorithmes itératifs pour résoudre le problème d'optimisation résultant sont généralement lents, de sorte que des méthodes d'inférence plus efficaces sur le plan informatique, telles que les réseaux de neurones, sont souvent préférées.

La propriété d'approximation universelle fait de NN un outil puissant pour approximer l'inversion des problèmes inverses qui surviennent en imagerie. Les premières tentatives d'utilisation de réseaux de neurones pour les problèmes de restauration d'images remontent à 1988. À cette époque, le manque de puissance de calcul ne permettait de former que des architectures simples avec peu de paramètres, et ces méthodes étaient donc largement dépassées par les méthodes variation-

nelles. Le domaine des réseaux de neurones a continué de croître jusqu'en 2012, lorsqu'un article révolutionnaire a montré qu'il était possible de former un réseau de neurones convolutifs (CNN) profond avec 60 millions de paramètres en utilisant 1,2 million d'images haute résolution sur une unité de traitement graphique (GPU). Cette approche a réussi à surpasser de manière significative les précédents algorithmes de l'état de l'art lors du concours de classification ImageNet LSVRC-2010. Depuis que des deep CNNs ont montré sa supériorité dans les tâches de classification d'images, les chercheurs ont commencé à rechercher de nouvelles façons d'utiliser cet outil pour résoudre également des problèmes inverses.

Les méthodes basées sur NN sont intéressantes car elles peuvent être construites de manière agnostique, c'est-à-dire sans aucune connaissance préalable du processus de dégradation. Ceci est bénéfique lorsque nous n'avons pas beaucoup de connaissances sur le problème d'intérêt ou qu'il est très difficile de le modéliser analytiquement. En conséquence, ces méthodes d'apprentissage supervisé nécessitent d'entraîner un réseau à l'aide d'un grand ensemble de données contenant des paires d'images cibles et dégradées pour le problème spécifique à traiter. Cependant, il s'agit généralement d'un processus coûteux et sujet aux erreurs qui doit être répété pour chaque problème inverse d'intérêt, ou même lorsqu'une composante du problème réel change. Malheureusement, les CNN profonds de l'état de l'art ont tendance à être énormes (c'est-à-dire des milliards de paramètres) et les ressources de calcul nécessaires pour entraîner ces modèles dans un délai raisonnable ne sont pas disponibles pour la plupart des chercheurs. Un autre inconvénient est qu'ils sont souvent vulnérables aux attaques adverses, c'est-à-dire que des perturbations à peine perceptibles à l'entrée peuvent amener le réseau à produire des réponses très différentes. Ces discontinuités soulèvent plusieurs inquiétudes quant à la robustesse de ces modèles, principalement dans des applications critiques telles que les reconstructions d'images médicales et le diagnostic clinique.

Un réseau formé sur un problème particulier apprend implicitement des informations sur la distribution des images cibles. Inspirés des méthodes variationnelles, certains travaux proposent d'exploiter ces informations pour régulariser d'autres problèmes inverses. Cette classe de méthodes comprend l'approche dite Plug-and-Play. Plus précisément, l'opérateur proximal du terme de régularisation de ce nouveau problème peut être considéré comme un opérateur de débruitage, nous pouvons donc utiliser un débruiteur de l'état de l'art comme étape de régularisation dans la procédure de minimisation alternative itérative. Par conséquent, l'image apprise basée sur NN est découplée du modèle de dégradation, de sorte qu'elle peut être utilisée pour résoudre de nombreux problèmes inverses différents sans recyclage. Le principal inconvénient de ces méthodes est la difficulté de prouver des propriétés de convergence. Un axe de recherche actif est dédié à imposer des restrictions sur le réseau de débruiteur considéré utilisé afin d'obtenir des résultats de convergence, sans dégrader les performances du modèle.

D'autre part, des modèles génératifs tels que Variational AutoEncoders, Generative Adversarial Networks (GAN) et Normalizing Flow sont proposés qui se rapprochent directement de la distribution de probabilité des images d'intérêt. Intuitivement, l'ensemble des images plausibles est supposé être contenu dans une variété de dimension beaucoup plus petite que l'espace ambiant. En utilisant des modèles à variables latentes, on peut effectuer une réduction de dimensionnalité et, en même temps, approximer la distribution de probabilité au sein de la variété. Ces modèles sont entraînés une fois de manière non supervisée sur un grand ensemble de données, indépendamment de tout problème particulier. En raison de la taille croissante des architectures de réseau profond de l'état de l'art, recycler un modèle génératif entier pour l'adapter à un algorithme de régularisation particulier peut être prohibitif pour la plupart des praticiens. Par conséquent, une direction de recherche prometteuse est de savoir comment combiner efficacement les méthodes bayésiennes avec des distributions a priori basées sur NN pour régulariser différents problèmes inverses.

Cette conjonction entre modèles génératifs et méthodes bayésiennes ouvre la porte à un large éventail d'algorithmes d'estimation. Historiquement, les méthodes statistiques classiques devaient être relativement simples, comme la restriction de travailler avec des a priori conjugués, pour construire des estimateurs calculables analytiquement des distributions postérieures. L'avancée des ressources de calcul permet de travailler avec des modèles plus complexes au prix de l'utilisation de méthodes MCMC. Par conséquent, de nouvelles méthodes bayésiennes doivent être développées qui utilisent des modèles génératifs et qui permettent d'explorer la distribution postérieure. Cependant, la mise en œuvre efficace des méthodes d'échantillonnage est une tâche difficile dans les espaces de grande dimension.

Contributions

- *Plug and Play methods for inverse problems in imaging*: Les schémas de compression d'ondelettes peuvent conduire à des artefacts visuels très spécifiques dus à la quantification des coefficients d'ondelettes bruitées. De tels artefacts sont particulièrement gênants dans le cas des compresseurs à ondelettes comme JPEG2000 et la recommandation CCSDS, qui sont largement utilisés pour compresser le cinéma numérique et les images de télédétection haute résolution. Comme ces artefacts ont une structure fortement corrélée dans l'espace, il est difficile de les supprimer avec des algorithmes de débruitage standard. Nous utilisons un terme d'attachment aux données probabiliste basé sur le modèle de formation d'images compressées bruitées qui est non linéaire mais convexe. Ensuite, nous proposons une méthode conjointe de débruitage et de décompression qui découple ce terme d'ajustement des données et un a priori implicite appris à l'aide d'un CNN de débruitage de l'état de l'art via l'algorithme d'optimisation ADMM. De plus, nous développons également un algorithme Plug and Play pour la régularisa-

tion d'un problème de superrésolution multi-images, en utilisant l'algorithme d'optimisation de Chambolle-Pock.

- *Joint Posterior Maximization with Autoencoding Prior (JPMAP)*: Nous abordons le problème de la résolution de problèmes inverses généraux mal posés en imagerie où le prior est donné par un autoencodeur variationnel (VAE). Alors que les approches précédentes basées sur MAP pour ce problème conduisent à des algorithmes d'optimisation hautement non convexes, notre approche calcule la MAP conjointe (latente-spatiale) qui conduit naturellement à des algorithmes d'optimisation alternée et à l'utilisation d'un encodeur stochastique pour accélérer les calculs. La technique résultante (JPMAP) effectue *Joint Posterior Maximization using an Autoencoding Prior*. Nous montrons des preuves théoriques et expérimentales que la fonction objectif proposée est assez proche de bi-convexe. En effet, il satisfait une propriété de bi-convexité faible qui est suffisante pour garantir que notre schéma d'optimisation converge vers un point stationnaire. Nous soulignons également l'importance d'entraîner correctement la VAE à l'aide d'un critère de débruitage, afin de s'assurer que l'encodeur généralise bien aux images hors distribution, sans affecter la qualité du modèle génératif. Cette simple modification est essentielle pour assurer la robustesse de l'ensemble de la procédure. Enfin, nous montrons comment notre méthodologie MAP conjointe est liée à des approches MAP plus courantes, et nous proposons un schéma de continuation qui utilise notre algorithme JPMAP pour fournir des estimations MAP plus robustes. Les résultats expérimentaux montrent également la meilleure qualité des solutions obtenues par notre approche JPMAP par rapport à d'autres approches MAP non convexes qui restent le plus souvent bloquées dans des optima locaux parasites. Ce travail a été soumis pour être publié sur SIAM Journal on Imaging Sciences et est en cours de révision.

- *Posterior sampling using a VAE prior*: La plupart des travaux utilisant des modèles génératifs comme distribution a priori se concentrent sur le calcul d'estimations ponctuelles. D'autre part, les méthodes MCMC pour l'échantillonnage à partir de la distribution a posteriori permettent d'explorer l'espace des solutions souhaitées et de calculer des estimations ponctuelles ainsi que d'autres statistiques sur les solutions telles que les estimations d'incertitude. Cependant, la performance des méthodes largement utilisées comme Metropolis-Hastings dépend d'avoir des distributions de propositions précises qui peuvent être difficiles à définir sur des espaces de grande dimension. Notre travail tente de rapprocher les deux domaines de recherche des méthodes MCMC et des modèles génératifs. En utilisant des techniques d'augmentation de données, nous développons un algorithme d'échantillonnage de la distribution a posteriori de type Gibbs qui exploite la nature bidirectionnelle des modèles VAE. Grâce à la capacité de parallélisation du GPU, nous exécutons efficacement plusieurs chaînes qui explorent plus rapidement la distribution postérieure et donnent également des tests de convergence plus

précis. Pour accélérer la période de rodage, nous explorons l'adaptation de l'échantillonnage d'importance recuit avec la méthode de rééchantillonnage. Il s'agit d'un travail en cours qui sera bientôt soumis pour publication dans une revue.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The reward of the young scientist is the emotional thrill of being the first person in the history of the world to see something or to understand something. Nothing can compare with that experience.

– Cecilia Payne-Gaposchkin

In this chapter, we introduce the main problem of interest, *inverse problems in imaging*, and highlight some of the challenges of state-of-the-art approaches for solving them. Next, we list the contributions made on this thesis and outline the rest of the document.

1.1 Motivation

Inverse problems are ubiquitous in science and engineering. In general, there is an event (cause) that triggers observable data (effect), and one needs to extract information about that event from the observed data. In this case, the forward problem consists in generating data from the event and the inverse problem is that of retrieving the event from the data, or at least as much as possible. For example, suppose that we need to get information on the interior structure of the Earth. Instead of taking soil samples that are costly and time-consuming, we can try to approximate this structure from measurements taken at the surface [211]. In modern medicine, it is important to have specialist assistance tools that provide as much information as possible about the patient's condition. Ideally, such methods should be non-invasive. This requirement makes it necessary to obtain an approximation of the actual condition of the organism from external measurements [73].

In some cases, a particular inverse problem can lead to a direct inversion algorithm. Unfortunately, discretization errors and noise present in the measurements can lead to large reconstruction errors. In other cases, incomplete measurements are not sufficient to uniquely determine the desired signal, so we must consider a broad set of solutions that is compatible with the data. To deal with these so-called *ill-posed problems* one can take more measurements in order to reduce the ambiguity of the solutions. For example, we can take a burst of images instead of only one, or we can combine multiple sources of information (known as sensor fusion [90]). However, it is sometimes desirable to take as few samples as possible because they are costly (in terms of time or money) or cause damage (e.g. radiation to patient's body on X-ray CT).

To force desired properties on the reconstructed signal, it is imperative to develop methods that reduce or weigh the solution space. To this end, various *regularization methods* have been proposed in order to incorporate prior information about the desired solution and then accurately estimate the unknown signal:

- The *variational approach* consists in a broad class of reconstruction methods based on optimization. Basically, one constructs an objective function which penalizes both a data fidelity term and a regularization term. This last term penalizes implausible images with large values, so promoting some kind of smoothness on the solution. To compute the solution, an iterative minimization procedure is applied.

- *Neural Networks* (NN) are a subset of machine learning algorithms inspired by the human brain [148, 188]. Recently, NN-based reconstruction algorithms have been proposed, which map the data directly to an estimate of the unknown signal. In most of the challenging imaging problems, NN-based methods significally outperform previous variational methods.

- Through statistical modeling, we can define our prior belief about the unknown signal in the form of a probability distribution, called *prior distribution*. This distribution, when combined with the observed data, gives rise to the so-called *Bayesian methods*. The resulting *posterior distribution* contains significant information about the unknown signal. In particular, we can compute different point estimates such as the MAP or MMSE estimators, but also to perform uncertainty estimation in the form of confidence intervals.

1.2 Challenges

Variational methods are appealing because the same regularization method can be used for different inverse problems only changing the data fidelity term. However, the main question is how to choose the regularization term. In imaging, a common choice for this term is the Total Variation (TV) seminorm [159, 38, 37, 142] which promotes sparse gradients on the restored image, which is consistent with the fact that most natural images are piece-wise smooth. Other classical choice is to promote sparsity of the image coefficients in some representation space. For example, it can be the representation of the image patches on a learned dictionary [69], or a wavelet decomposition [13]. Due to the complex nature of natural images, these hand-crafted regularization terms do not fully caracterize all the properties that we want to see in the computed solution. In addition, we must be careful in choosing convex terms so that the resulting optimization problem can be solved. These limitations affect the quality of the reconstructions, which often lead to artifacts like the staircase effect when using TV regularization [198]. Furthermore, iterative algorithms to solve the resulting optimization problem are generally slow, so more computationally efficient inference methods such as neural networks are often preferred.

The universal approximation property makes NN a powerful tool for approximating the inversion of inverse problems that arise in imaging. The earliest attempts to use neural networks for image restoration problems date back to 1988 [248]. At that time, the lack of computational power only allowed to train simple architectures with few parameters, and therefore these methods were largely outperformed by variational methods. The field of neural networks continued to grow until 2012, when a groundbreaking paper [125] showed that it is possible to train a deep *Convolutional Neural Network (CNN)* with 60 million parameters using 1.2 million high-resolution images on a *Graphics Processing Unit (GPU)*. This approach managed to significantly outperform previous state-of-the-art algorithms on the ImageNet LSVRC-2010 classification contest. Since deep CNN showed their superiority in image classification tasks, researchers started to look for new ways to use this tool to solve inverse problems too.

NN-based methods are appealing because they can be constructed *agnostically*,

that is without any prior knowledge of the degradation process. This is beneficial when we do not have much knowledge of the problem of interest or it is very hard to model it analytically. As a consecuence, these *supervised learning methods* require training a network using a large dataset containing pairs of target and degraded images for the specific problem at hand. However, this is generally a costly and error prone process that must be repeated for each inverse problem of interest, or even when one component of the actual problem changes. Unfortunately, state-of-the-art deep CNN tend to be huge (i.e. billions of parameters) and the computational resources necessary to train these models in a reasonable period of time are not available to most researchers. Another disadvantage is that they are often vulnerable to *adversarial attacks*, i.e., hardly perceptible perturbations at the input may cause the network to output very different responses [212]. These discontinuities raise several concerns about the robustness of these models, mainly in critical applications such as reconstructions of medical images and clinical diagnosis [72].

A network trained on a particular problem implicitly learns some information about the distribution of target images. Inspired by variational methods, some works propose to leverage this information to regularize other inverse problems. This class of methods includes the so-called *Plug-and-Play approach* [227]. Especifically, the proximal operator of the regularization term of this new problem can be seen as a denoising operator, so we can use a state-of-the-art denoiser as the regularization step in the iterative alternate minimization procedure. Hence, the NN-based learned image prior is *decoupled* from the degradation model, so it can be used to solve many different inverse problems without retraining. The main drawback of these methods is the difficulty of proving convergence properties. An active line of research is dedicated to imposing restrictions on the considered denoiser network used in order to achieve convergence results, without degrading the performance of the model.

On the other hand, *generative models* such as Variational AutoEncoders [121], Generative Adversarial Networks (GAN) [88] and Normalizing Flows [165] have been proposed which directly approximate the probability distribution of the images of interest. Intuitively, the set of plausible images is assumed to be contained in a manifold of much smaller dimension than the ambient space. Using latent variable models one can perform dimensionality reduction and, at the same time, approximate the probability distribution within the manifold. These models are trained once in an unsupervised manner on a large data set, regardless of any particular problem. Due to the growing size of state-of-the-art deep network architectures, to retrain an entire generative model to fit a particular regularization algorithm can be prohibitive for most practitioners. Hence, a promising direction of research is how to efficiently combine Bayesian methods with NN-based prior distributions to regularize different inverse problems.

This conjunction between generative models and Bayesian methods opens the

door for a broad set of estimation algorithms. Historically, classical statistical methods had to be relatively simple, like the restriction of working with conjugate priors, to construct analytically computable estimators of posterior distributions. The advancement of computational resources makes it possible to work with more complex models at the cost of the use of MCMC methods [183]. Consequently, new Bayesian methods need to be developed that use generative models and that allow the posterior distribution to be explored. However, the efficient implementation of sampling methods is a difficult task in high-dimensional spaces.

1.3 Contributions

Plug and Play methods for inverse problems in imaging

Wavelet compression schemes may lead to very specific visual artifacts due to quantization of noisy wavelet coefficients. Such artifacts are particularly annoying in the case of wavelet-based compressors like JPEG2000 and the CCSDS recommendation, which are extensively used to compress digital cinema and high-resolution remote sensing images. As these artifacts have highly spatially-correlated structure, it is difficult to remove them with standard denoising algorithms. We use a probabilistic data-fitting term based on the formation model of noisy compressed images which is non-linear but convex. Then, we propose a joint denoising and decompression method that decouples this data-fitting term and an implicit prior learnt using a state-of-the-art denoising CNN throught the ADMM optimization algorithm (see [86]). Additionally, we also develop a Plug and Play algorithm for the regularization of a multi-image superresolution problem, using the Chambolle-Pock optimization algorithm [39].

Joint Posterior Maximization with Autoencoding Prior

We address the problem of solving general ill-posed inverse problems in imaging where the prior is given by a variational autoencoder (VAE). Whereas previous MAP-based approaches to this problem lead to highly non-convex optimization algorithms, our approach computes the joint (space-latent) MAP that naturally leads to alternate optimization algorithms and to the use of a stochastic encoder to accelerate computations. The resulting technique (JPMAP) performs Joint Posterior Maximization using an Autoencoding Prior. We show theoretical and experimental evidence that the proposed objective function is quite close to bi-convex. Indeed, it satisfies a weak bi-convexity property which is sufficient to guarantee that our optimization scheme converges to a stationary point (see [85]). We also highlight the importance of correctly training the VAE using a denoising criterion, in order to ensure that the encoder generalizes well to out-of-distribution images, without affecting the quality of the generative model. This simple modification

is key to providing robustness to the whole procedure. Finally we show how our joint MAP methodology relates to more common MAP approaches, and we propose a continuation scheme that makes use of our JPMAP algorithm to provide more robust MAP estimates. Experimental results also show the higher quality of the solutions obtained by our JPMAP approach with respect to other non-convex MAP approaches which more often get stuck in spurious local optima. This work was submitted to be published on SIAM Journal on Imaging Sciences and is under review.

Posterior sampling using a VAE prior

Most of the work using generative models as image priors focus on computing point estimates. On the other hand, MCMC methods for sampling from the posterior distribution permit to explore the space of desired solutions and to compute point estimates as well as other statistics about the solutions such as uncertainty estimates. However, the performance of widely used methods like Metropolis-Hastings depends on having precise proposal distributions which can be challenging to define on high-dimensional spaces. Our work attempts to bridge the gap between the two research areas of MCMC methods and generative models. Using data augmentation techniques, we develop a Gibbs-like posterior sampling algorithm that exploits the bidirectional nature of VAE networks. Thanks to the GPU's parallelization capability, we efficiently run multiple chains which explore more rapidly the posterior distribution and also give more accurate convergence tests. To accelerate the burn-in period we explore the adaptation of the annealed importance sampling with resampling method. This is an ongoing work that will be soon submitted for journal publication.

List of publications

- González, M., Preciozzi, J., Musé, P., & Almansa, A. (2017). *Joint denoising and decompression: A patch-based Bayesian approach.* In Proceedings of the **IEEE International Conference on Image Processing - ICIP (pp 1252-1256)**.

- González, M., Preciozzi, J., Musé, P., & Almansa, A. (2018). *Joint denoising and decompression using CNN regularization.* In Proceedings of the **IEEE Conference on Computer Vision and Pattern Recognition Workshops - CVPRW (pp. 2598- 2601)**.

- González, M., Musé, P., Delbracio, M. & Almansa, A. (2019). *Solving Linear Inverse Problems by Joint Posterior Maximization with a VAE Prior.* Poster presentation at **KHIPU Latin American Meeting In Artificial Intelligence** (poster id: 2-36).

- González, M., Almansa, A., & Tan, P. (2021). *Solving Inverse Problems by Joint Posterior Maximization with Autoencoding Prior*. arXiv:2103.01648. Under review at **SIAM Journal on Imaging Sciences**

- González, M., Almansa, A., Musé, P. (2021). *Posterior Sampling with Autoencoding Prior*. In preparation.

List of presentations

- González, M., Almansa, A. and Musé, P. *Joint Image Denoising and Decompression*. Invited presentation at **First ICT4V Workshop on Big and Complex Data Theory and Applications**. Montevideo, May 2018.

- González, M., Preciozzi, J., Musé, P., & Almansa, A. *Joint denoising and decompression using CNN regularization*. Poster presentation at **SIAM Conference on Imaging Sciences**. Bologne, June 2018.

- González, M., Almansa, A., Delbracio, M. Lezama, J. Musé, P. & Tan, P. *Solving Inverse Problems in Imaging with neural regularizers and convergence guarantees*, Invited presentation at **ALGORITMY 2020**, Podbanske, Sep 14th 2020.

- González, M., Almansa, A., Delbracio, M. Lezama, J. Musé, P., & Tan, P. *Solving Inverse Problems by Joint Posterior Maximization with VAE Prior*. Invited oral presentation at **GTTI – Centre Borelli**, ENS Paris-Saclay, Oct 23rd 2020.

- González, M., Almansa, A., Delbracio, M. Lezama, J. Musé, P., & Tan, P. *Solving Inverse Problems by Joint Posterior Maximization with Autoencoding Prior*. Invited oral presentation at **SIAM OneWorld IMAGINE Seminars**, Feb 25th 2021.

- González, M., Laumont, R., De Bortoli, V., Delbracio, M., Delon, J., Durmus, A., Lezama, J., Musé, P., Pereyra, M. & Almansa, A. (2021). *Bayesian Estimators for Inverse Imaging Problems with Decoupled Learned Priors*, Invited presentation at **Bath-LMS workshop on ANALYTIC AND GEOMETRIC APPROACHES TO MACHINE LEARNING**, Bath, July 27th 2021.

Other contributions

- Abergel, R., Almansa, A., González, M., Marzoa, M. & Moisan L. (2018), *Traitement de super-résolution d'images spatiales THR*. Septembre 2018. Rapport final du contrat de recherche CNES # 170862/00.

1.4 Outline

To start, in Chapter 2 we review the main concepts that appear on this thesis. First, we introduce the problem of interest, *inverse problems in imaging*, and ilustrate the ill-posedness nature of some of them. Next, we describe the most common *reconstruction methods* encountered in the literature, in particular deep learning techniques. We also review some of the modern *generative models* and briefly describe how they approximate the probability distribution of images.

In Chapter 3, we focus on *Plug and Play* (PnP) methods for decoupling the data fidelity term from the prior term using denosing CNNs. We first develope an ADMM-based algorithm for the restoration of compressed noisy images by standard schemes such as JPEG2000. We also develop a PnP method for the regularization of a multi-image superresolution problem, using the Chambolle-Pock optimization algorithm for the decoupling.

In Chapter 4, we fully describe the *Variational AutoEncoder* model and some of its variants that show its capabilities to capture the probability distribution of high-dimensional datasets such as images. Then, we propose a joint (space-latent) MAP estimator for inverse problems, named *Joint Posterior Maximization using an Autoencoding Prior (JPMAP)*.

In Chapter 5 we first review classical sampling theory for the computation of Monte Carlo estimates, in particular MCMC algorithms. Next, we develop a Gibbs-like *posterior sampling algorithm* for the exploration of posterior distributions of inverse problems using multiple chains and a VAE as image prior.

Finally, Chapter 6 closes with the conclusions of this thesis.

Chapter 2

Background

If I have seen further it is by standing on the shoulders of Giants.

– Issac Newton

In this chapter we review the main concepts that appear on this thesis. We start by describing common *inverse problems* arising in imaging and show the ill-posed behaviour one generally encounters in most of them. Next, we describe the most common *reconstruction methods* encountered in the literature, in particular deep learning techniques. We also review some of the modern *generative models* and briefly describe how they approximate the probability distribution of images.

2.1 Inverse problems

The term "inverse problem" sugests the existence of a "forward problem". Mathematically speaking, there is a duality between forward and inverse problems. For example, to calculate $f'(x)$ from $f(x)$ (differentiation) or $f(x)$ from $f'(x)$ (integration) are one problem the inverse to the other. Or to compute $y = Ax$ and $x = A^{-1}y$ when A is an invertible matrix. Despite these theoretical discussions, in practice, the inverse problem arises naturally: we are interested in finding the source of an observed effect or estimating the parameters of a model related to that process given measured data. Hence, starting from a mathematical model of the data generating process called *forward problem* we are asked to estimate the signal or parameters which give rise to the observed data, that is to solve the *inverse problem*.

Historically, one of the first inverse problems to be deeply investigated is the *inverse Sturm-Liouville problem*, a class of inverse spectral problem [74]. As an application, it can be briefly described as the problem of infering the density of a string or the shape of a membrane given the frequencies of vibrations it can produce [113, 35]. Also, in *inverse scattering problems* the direct problem consists in describing how radiation or particles are scattered by an object based on its internal properties. Hence, the inverse problem is that of determining the characteristics of the object based on data showing how it scatters incoming radiation or particles [36]. Examples of inverse scattering problems arise in seismic prospecting [119], in remote sensing of the earth [61], in nondestructive evaluation of materials [5], and in medical imaging [10].

In the latter case, the first developed medical imaging system with enormous clinical impact is Computed Tomography [75, 18]. Here, a thin beam of X-rays traverse a slice of the patient body. On the other side, an array of sensors measure the intensity of the rays which are attenuated by anatomic structures. This process is repeated rotating the sensing system, therefore generating projections along multiple directions. Hence, the inverse problem consists in estimating the internal density distribution by inverting this procedure. With some assumptions to simplify the mathematical formulation, this forward problem can be modeled using the Radon Transform: for a function $f \in L^2(\mathbb{R}^2)$ which in this case represents the density of the internal tissues, we define $\mathcal{R}f \colon \mathbb{R} \times [0, \pi) \to \mathbb{R}$ as

$$\mathcal{R}f(t, \theta) = \int f(t\boldsymbol{n}_\theta + s\boldsymbol{n}_\theta^\perp)\, ds \tag{2.1}$$

where $\boldsymbol{n}_\theta = (\cos\theta, \sin\theta)$ and $\boldsymbol{n}_\theta^\perp = (-\sin\theta, \cos\theta)$ (see Figure 2.1). The resulting map $\mathcal{R}f$ is known as the *sinogram*. Hence, the inverse problem that arise is that of retrieving f given the sinogram $\mathcal{R}f$, so inverting the Radon transform. Following this invention, other medical imaging systems have been developed, like magnetic resonance, positron emission tomography and single photon emission computed tomography [17].

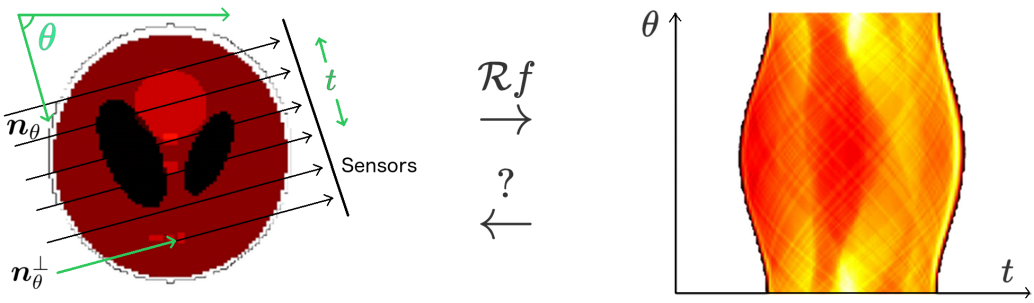


Figure 2.1: Given function f, the Radon transform $\mathcal{R}f$ is computed as the integral of f along lines having direction $\boldsymbol{n}_\theta^\perp$. The result (shown on the right) is called the *sinogram*. The inverse problem that arise is that of retrieving f given the sinogram $\mathcal{R}f$, so inverting the Radon transform.

2.1.1 Problem statement

In this work we focus on inverse problems arising in *digital imaging*. That is, the unknown signal to be estimated is a vector of dimension d. There are mainly common two ways of representing an image $\boldsymbol{x}$ with a finite number of parameters:

- *Pixel-wise representation*: $\boldsymbol{x} \in \mathbb{R}^{H\times W\times C}$ where H and W are the heigth and width of the image respectively, C is the number of channels ($C = 1$ for monochromatic images, $C = 3$ for RGB images and $C > 1$ for general hyperspectral images) and $d = HWC$.
- *Coordinate-based representation*: $\boldsymbol{x} = u_\theta$ where $u_\theta \colon (\Omega \subset \mathbb{R}^2) \to \mathbb{R}^C$ is a function parameterized by $\theta \in \mathbb{R}^d$ and $u_\theta(x_1, x_2)$ are the intensity values of the C channels at spatial position $u_\theta(x_1, x_2)$.

The vast majority of work on imaging and deep learning assume the discrete pixel-wise representation of images, so that is the way we go. Alternatively, recent work uses neural networks to construct continuous maps from spatial locations to intensity values which can accurately model natural scenes and show improvements in the representation of high frequencies (see for example [202] and the references therein). For simplicity, in what follows we denote $\boldsymbol{x} \in \mathcal{X}$.

We continue by formalizing the main problem we are interested in solving. Let $\boldsymbol{x}^* \in \mathcal{X}$ be the target image to be captured. Due to physical constraints, hardware architecture design and other limitations described earlier, we are only able to capture data $\boldsymbol{y} \in \mathcal{Y}$ which is different from (but related to) $\boldsymbol{x}^*$. A general formulation of this forward model is

$$\boldsymbol{y} = \mathcal{D}(\boldsymbol{A}\boldsymbol{x}^*) \tag{2.2}$$

where $\boldsymbol{A}$ is a matrix accounting for a *linear operator*, and $\mathcal{D}$ is a *degradation operator* which may be non-linear and includes a source of randomness in the data generating process like noise. The data domain $\mathcal{Y}$ depends on the whole degradation process and is explicitly known.

In a wide range of practical applications, most part of the degradation process (except for the noise) can be modeled by a linear operator. Furthermore, the noise is commonly assumed to be additive, Gaussian, and independent of the signal. Hence, as a particular case, we can define a *linear inverse problem* as

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^* + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2 I) \tag{2.3}$$

where σ^2 is the noise variance.

Although other types of noise can be considered, they are somehow related to the Gaussian noise (e.g. by means of the Central Limit Theorem) or can be reduced to it in some cases. For example, photon counting errors produced by charge-coupled-device (CCD) cameras are typically modeled by a Poisson distribution [203]. By applying a variance-stabilizing procedure like the *Anscombre transform*, Poisson noise is transformed into (approximately) Gaussian white noise [7]. Then, after restoring this new image assuming additive Gaussian noise, an inverse Anscombe transformation is applied [210]. In the following, unless explicitly stated otherwise, we assume the degradation model (2.3) as is done in most of the image processing literature.

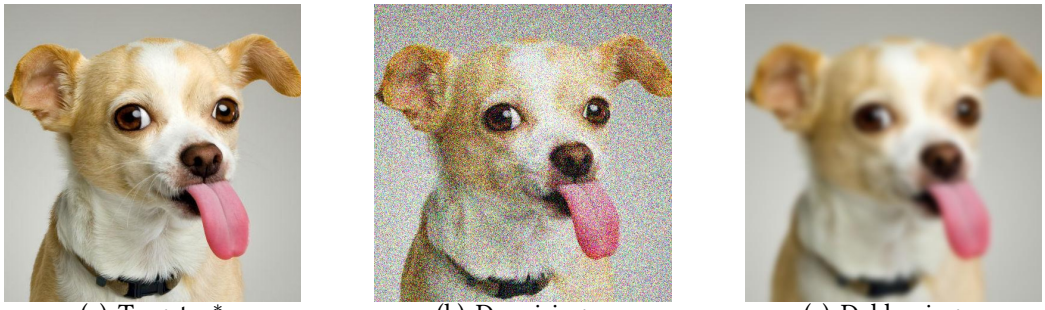
Examples of linear inverse problems in low-level image processing are listed below (and shown on Figure 2.2):

- *Denoising*: $\boldsymbol{A} = I$ (identity matrix) and $\eta \neq 0$.
- *Deblurring* (or *deconvolution*): $\boldsymbol{A}\boldsymbol{x} = h * \boldsymbol{x}$, a convolution with kernel h.
- *Superresolution*: $\boldsymbol{A}\boldsymbol{x}$ is a blurred and subsampled version of $\boldsymbol{x}$.
- *Interpolation*: $\boldsymbol{A}$ is a masking operator containing some (often random) rows of the identity matrix.
- *Inpainting*: related to intepolation but here an entire region of the image (e.g. a patch) is missing.

We refer the reader to [18] and the references therein for a precise mathematical treatment of inverse problems in imaging.

2.1.2 Ill-posed problems

In many applications, the particular structure of the problem can be leveraged to construct ad-hoc restoration algorithms, like the Filtered Back Projection (FBP) algorithm for the CT reconstruction problem mentioned above [162]. Unfortunately,



(a) Target $\boldsymbol{x}^*$ (b) Denoising $\boldsymbol{y}$ (c) Deblurring $\boldsymbol{y}$

(d) Superresolution $\boldsymbol{y}$ (e) Interpolation $\boldsymbol{y}$ (f) Inpainting $\boldsymbol{y}$

Figure 2.2: Examples of common linear inverse problems in imaging.

incomplete data, discretization errors and/or noise present in the measurements can lead to large errors in estimating the solution of the inverse problem. This instability is known as *ill-posedness*.

The well-posedness of a problem, in the sense of Hadamard [96, 112] can be formalized as follows. Let $\mathcal{F}\colon \mathcal{X} \to \mathcal{Y}$ be a map between metric spaces, and $\mathcal{F}(\boldsymbol{x}) = \boldsymbol{y}$ the inverse problem of interest. We say that this problem is well-posed if it satisfies

1. *Existence*: $\mathcal{F}(\mathcal{X}) = \mathcal{Y}$ (i.e. for all $\boldsymbol{y} \in \mathcal{Y}$ exists $\boldsymbol{x} \in \mathcal{X}$ with $\mathcal{F}(\boldsymbol{x}) = \boldsymbol{y}$).
2. *Uniqueness*: $\mathcal{F}$ is invertible (i.e. the solution in 1. is unique).
3. *Stability*: $\mathcal{F}^{-1}$ is continuous.

If an inverse problem does not satisfy some of these conditions, it is said to be *ill-posed*.

Observe that, in most practical applications, it is not sufficient to have existence and uniqueness of the solution. Indeed, if the inverse operator is not bounded, then a little amount of noise, discretization or rounding errors can lead to unstable reconstructions. This last stability requirement implies that if the measurements $\boldsymbol{y}$ and $\boldsymbol{y}'$ are *close*, then the corresponding solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$ are *close* too, therefore errors are not amplified naturally by the inversion process. Hence, we need that the solution depends continually on the data $\boldsymbol{y}$ in order to construct stable restoration algorithms.

For example, in the simple case of a linear equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ on Euclidean spaces, there always exists a solution if the columns of $\boldsymbol{A}$ span $\mathcal{Y}$, uniqueness is equivalent to $\boldsymbol{A}$ being non-singular, and in numerical applications stability requires the condition number of $\boldsymbol{A}$ to be small. For the imaging inverse problems defined in the previous section (without noise, $\eta = 0$), it is clear that superresolution, interpolation and inpainting problems are ill-posed because the matrix $\boldsymbol{A}$ is not invertible, so we have infinite possible solutions.

The problem of deconvolution requires more analysis. Suppose we have $\boldsymbol{y} = h * \boldsymbol{x}$. This degradation may be due to the Point Spread (PSF) function of the imaging system, motion blur, or because the scene is out of focus. As $\hat{\boldsymbol{y}} = \hat{h}\,\hat{\boldsymbol{x}}$, in order to reconstruct $\boldsymbol{x}$ from $\boldsymbol{y}$, we deduce that the support of $\hat{h}$ must contain the one of $\hat{\boldsymbol{x}}$. If not, there will be some frequencies of $\boldsymbol{x}$ that are not retrievable from $\boldsymbol{y}$. Next, to simplify the analysis we assume now a continuous version of the problem and choose the kernel $h(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$. Then, as $\hat{h}(w) = e^{-w^2/2}$, a direct inversion gets

$$\hat{\boldsymbol{x}}(w) = \hat{\boldsymbol{y}}(w) e^{w^2/2}. \tag{2.4}$$

However, if we add Gaussian noise $\boldsymbol{y} = h * \boldsymbol{x} + \eta$ as before, we obtain

$$\hat{\boldsymbol{y}}(w) e^{w^2/2} = \hat{\boldsymbol{x}}(w) + \hat{\eta}(w) e^{w^2/2} \tag{2.5}$$

and hence the direct inversion (2.4) amplifies the high frequencies of the noise η, leading to an inaccurate reconstruction. We conclude that this deconvolution problem is ill-posed too.

We note that, in some cases, the kernel h can also be unknown, or known up to some parameters. This problem is known as *blind deconvolution*. In that situation, we need even more information to estimate also the kernel. In this thesis, we focus on inverse problems where the degradation operator $\boldsymbol{A}$ is completely known or at least accurately approximated.

To deal with the ill-posedness of a problem, one can take more measurements in order to avoid the ambiguity of the solutions. For example, we can take a burst of images instead of only one or combine multiple sources of information (known as sensor fusion [90]). However, it is sometimes desirable to take as few samples as possible because they are costly (in terms of time or money) or cause damage (e.g. radiation to patient's body on X-ray CT). Hence, to force desired properties on the estimators, it is imperative to develop methods that reduce or weigh the solution space. For a complete survey on ill-posed inverse problems, see [112] and the references therein.

2.2 Reconstruction methods

The set of acceptable responses depends on the particular problem at hand. In some cases, a good-looking picture may be enough (e.g. for posting on social net-

works). On the other hand, in medical applications, a data-compatible image can be recovered in which a tumor is not seen in a particular area of interest. But, is it possible to find another solution where a tumor does appear? If both cases are consistent with the available data, giving just either solution can lead to wrong clinical decisions.

There are several ways to solve an inverse problem: we can compute point estimates, confidence intervals, posterior distributions, samples from this posterior, and so on. In this section, we describe the most common reconstruction methods encountered in the literature.

2.2.1 Variational approach

The dominant approach for image reconstruction consists in a broad class of methods based on optimization, known as the *variational approach*. As $\mathcal{Y} \subseteq \mathbb{R}^l$ in most applications, a first attempt to retrieve the target image $\boldsymbol{x}^*$ from data $\boldsymbol{y}$ is to compute

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{2.6}$$

which corresponds to a least squares solution. This is consistent with the assumption that the acquisition noise is Gaussian (see equation (2.19)). In practice, in ill-posed problems this direct inversion does not perform well due to incomplete data ($l < \dim(\mathcal{X})$) and/or measurement errors that lead to poor reconstructions for ill-conditioned matrices $\boldsymbol{A}$.

Simple regularization methods for solving systems of linear equations based on matrix factorization like truncating singular values of $\boldsymbol{A}$ has been proposed [98]. As the image size grows, these algorithms are practically unfeasible, so iterative methods are often prefered (e.g. a Krylov least squares solver [193] or a Landweber iteration [241, 126]). However, this type of methods is not appropriate when it comes to problems that have a significant amount of missing information.

To alleviate this problem, Tikhonov suggested a way to weight the solution space and therefore to promote one type of solution over another [218, 219]. The following is a general *variational formulation* for solving linear inverse problems:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \{f(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{y}) + \lambda R(\boldsymbol{x})\}\,. \tag{2.7}$$

Basically, this objective function is the sum of a *data fidelity term* $f(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{y})$ and a *regularization term* $\lambda R(\boldsymbol{x})$, both positive. For the first one, we can simply choose the Euclidean distance as in (2.6) but other options are also possible.

Adding a regularization term is an effective way to incorporate additional information into the inverse problem, therefore reducing the ambiguity in the solutions present in the data. This term encodes the properties that we want to retrieve

on the resulting image, punishing unwanted solutions with high values. In this way, we are replacing an ill-posed inverse problem with a nearby (conditionally) well-posed one, whose solution is stably computable and provides a reasonable answer to the original problem. Next, we list many regularization methods encountered in the literature:

- **Total Variation (TV)**: Rudin, Osher, and Fatemi published the widely known *Total Variation (TV)* regularization method [159, 38, 37, 142]: they argue that natural images have *bounded variation*. Therefore, they define the regularization term as

$$R(\boldsymbol{x}) = \sum_{i,j} \|\nabla_{i,j}\boldsymbol{x}\|_p, \qquad \nabla_{i,j}\boldsymbol{x} = (\boldsymbol{x}_{i+1,j} - \boldsymbol{x}_{i,j}, \boldsymbol{x}_{i,j+1} - \boldsymbol{x}_{i,j}) \tag{2.8}$$

  where $\boldsymbol{x}_{i,j}$ represents the pixel value of $\boldsymbol{x}$ at position (i, j) and $p = 1$ or 2. This regularization promotes piecewise constant solutions.

- **Sparsity**: $R(\boldsymbol{x}) = \|L\boldsymbol{x}\|_1$ for an appropiate matrix L, so promoting sparsity of the coefficients of the image computed in some representation space. For example, $L\boldsymbol{x}$ can be the decomposition of $\boldsymbol{x}$ on a learned dictionary [69] or a wavelet decomposition [13]. This regularization method is commonly known as *basis pursuit* [41].

- **Combined regularization**: $R(\boldsymbol{x})$ can be the sum of various regularization terms, combining the goodness of different assumptions into the same solution, such as first/second order penalties [164] or spatial/spectral constraints.

- **Expected Patch Log-Likelihood (EPLL)** [249]: they propose to learn a prior on the space of image patches $P_i\boldsymbol{x}$ and to promote every patch on the image to be coherent with this prior: $R(\boldsymbol{x}) = -\sum_i \log p(P_i\boldsymbol{x})$.

Except for simple cases (e.g. quadratic data fitting and regularization terms) the minimization problem (2.7) has no analytical solution, and hence iterative optimization algorithms are needed. Moreover, a naive gradient descent algorithm is only applicable when the objective function (2.7) is smooth, and even in that case, it may be extremely slow in high-dimensional spaces $\mathcal{X}$. Therefore, efficiency is another requirement when designing optimization methods. Splitting methods that can solve a wide range of non-differentiable optimization problems are the *Alternating Direction Method of Multipliers (ADMM)* [25] and the *Primal-Dual Hybrid Gradient (or Chambolle-Pock) algorithm* [39]. A detailed description of these methods is presented in Chapter 3.

The parameter $\lambda > 0$ balances between the data fidelity term and the regularization term. Intuitively, larger values of λ are needed when the amount or quality of the data $\boldsymbol{y}$ decreases. The choice of the λ value can be adjusted manually (e.g. the one who leads to the best mean result on a given set of images), or jointly with the

optimization algorithm. This can be done observing that (2.7) is the unconstrained Lagrangian formulation of the following constrained minimization problem:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in C_\delta} R(\boldsymbol{x}), \qquad C_\delta = \{\boldsymbol{x} \in \mathcal{X} \ : \ f(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{y}) \leq \delta\} \tag{2.9}$$

where $\delta > 0$ prescribes a target fidelity to the data $\boldsymbol{y}$. For example, when f is the Euclidean distance and $\boldsymbol{y} = A\boldsymbol{x} + \eta$ with $\eta \sim \mathcal{N}(0, \sigma^2)$, as in this case we have $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 \simeq l\sigma^2$, a usual choice is $\delta = l\sigma^2$. Hence, starting with an initial value $\lambda_0 > 0$ and after computing the current iteration $\boldsymbol{x}^{(k)}$, we update λ_k using the actual value of the residual $f(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{y}) - \delta$ (e.g. larger values of λ_k for larger residual values). Examples of these type of parameter selection methods are the *exponential multiplier method* [124] and the *Morozov's discrepancy principle* [156]. Many principles and algorithms exist to select a suitable value of lambda. In this work we shall concentrate on the discrepancy principle (which better fits our MAP framework in Chapters 3 and 4), but we may use a likelihood maximisation principle in the future for the approach presented in Chapter 5.

Due to the complex nature of natural images, these hand-crafted regularization terms do not fully caracterize all the properties that we want to see in the computed solution. In addition, we must be careful in choosing convex terms so that the resulting optimization problem can be solved. These limitations affect the quality of the reconstructions, which often lead to artifacts like the staircase effect when using TV regularization [198]. For more accurate solutions we need more realistic priors. Furthermore, iterative algorithms to solve the resulting optimization problem are generally slow, so more computationally efficient inference methods such as neural networks are often preferred.

2.2.2 Learning approach

Artificial Neural Networks (ANN or simply NN) are a subset of machine learning algorithms inspired by the human brain [148, 188]. They generally consist of a set of nodes called *neurons* grouped in different *layers* and connections between them. Every continuous function can be approximated (as close as desired) using such a network, when properly constructed. In particular, this *universal approximation property* makes neural networks a powerful tool to approximate the inversion of problems that arise in imaging. In this section, we briefly review some of the components of state-of-the-art *deep learning* algorithms for solving inverse problems.

End-to-end learning

Given some input $\boldsymbol{z}$, the neurons in the first layer perform some computations using a set of *weights* associated to each neuron of the layer, and then pass the result as the input of the next layer in the network. This process is repeated until

the last layer returns the final output $\boldsymbol{F}_\theta(\boldsymbol{z})$, where θ is the set of parameters that collects all the weights associated to the neurons of the neural network.

The usual method to *train* a neural network $\boldsymbol{F}_\theta$ is the so-called *supervised learning* which consists in estimating its weights θ from a set of input-output examples. Formally, given a dataset $\mathcal{D} = \{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1}^N$ of N *training examples* where $\boldsymbol{x}_i$ is the desired (target) solution of the problem at hand given input $\boldsymbol{y}_i$, the training procedure corresponds to finding

$$\theta^* = \arg\min_{\theta\in\Theta} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{F}_\theta(\boldsymbol{y}_i)) \tag{2.10}$$

where Θ is the set of valid parameters values for θ, and f is a quality metric between output vectors $\boldsymbol{x}$. This search is generally done using gradient descent (GD) variants like *stochastic GD (SGD)* [87] and *Adam* [120]. To compute gradients of neural networks we can use the *backpropagation algorithm* which was popularized by [191] but similar ideas can be found on earlier works [237]. Once the training is finished, assuming we found a good optimum of (2.10) and that the dataset $\mathcal{D}$ represents the entire distribution of images of interest, then we can use $\boldsymbol{F}_{\theta^*}(\boldsymbol{y})$ to approximate the solution of the inverse problem at hand given data $\boldsymbol{y}$ never seen before, that is to *generalize*.

Image quality metrics

As William Thomson Kelvin said, *"what is not defined cannot be measured; what is not measured, cannot be improved"*. When optimizing (2.10) we are searching for the best approximation $\boldsymbol{F}_{\theta^*}$ of the inverse operator of the degradation process, as measure by f. Hence, the definition of f *is* the definition of image quality for our restoration algorithm.

As $\mathcal{X} \subseteq \mathbb{R}^d$ in most applications, a straighforward way to measure image quality is to use the *Mean Squared Error (MSE)*:

$$\mathrm{MSE}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{d}\|\boldsymbol{x} - \boldsymbol{x}'\|^2. \tag{2.11}$$

Similarly, the *Peak Signal-to-Noise Ratio (PSNR)* is defined as

$$\mathrm{PSNR}(\boldsymbol{x}, \boldsymbol{x}') = 10\log_{10}\left(\frac{M}{\mathrm{MSE}(\boldsymbol{x}, \boldsymbol{x}')}\right) \tag{2.12}$$

where M is the *dynamic range* or the maximum fluctuation in the input image (e.g. $M = 1$ when we are working with $\mathcal{X} = [0, 1]^d$). Hence, minimizing $\mathrm{MSE}(\boldsymbol{x}, \boldsymbol{x}')$ is equivalent to maximizing $\mathrm{PSNR}(\boldsymbol{x}, \boldsymbol{x}')$.

The popularity of MSE and PSNR in the literature for evaluating image quality may be due to their simplicity to work with and its statistical interpretation [233].

On the other hand, it has been reported that they perform poorly when used to predict human perception of image fidelity. For example, a shift on the original pixel values $\boldsymbol{x}' = \boldsymbol{x} + c$ can lead to a high MSE distance although the difference is practically unnoticed visually. Other popular choices are the *Structural SIMilarity (SSIM) Index* [235] and its multiscale variant *MS-SSIM* [234] which compute an error measure simulating known properties of human visual system (based on luminance, contrast and structure of the compared images). In addition, other perceptual quality metrics can be used, as the *Normalized Laplacian Pyramid (NLP) distance* [127] or the NN-based *Learned Perceptual Image Patch Similarity (LPIPS)* [246].

Neural network architectures

The choice of the network family $\{\boldsymbol{F}_\theta\}_{\theta\in\Theta}$ determines the type of functions $\boldsymbol{y} \mapsto \boldsymbol{x}$ we can obtain for solving our inverse problem of interest, so we need to pay close attention to the architecture design. There are several types of layers that can be stacked to build a wide variety of neural networks. Except for the input and output layers, the rest of them are called *hidden layers*.

A simple type of layer is the so-called *fully connected layer*: each neuron on the layer is connected to each neuron on the previous layer (see Figure 2.3) and can be expressed as

$$\boldsymbol{z}' = g(W\boldsymbol{z}) \tag{2.13}$$

where $\boldsymbol{z}$ and $\boldsymbol{z}'$ are the input and output of the layer respectively, W is the matrix that contains all the weights of the corresponding neurons of the layer, and g is a (generally non-linear) *activation function* like the sigmoid (or logistic) function and the Rectified Linear Unit (ReLU) [87]. When built only by concatenating this type of layer, $\boldsymbol{F}_\theta$ is generally called *fully connected network* or *Multi Layer Perceptron (MLP)*. The *Universal Approximation Theorem* [49, 104] ensures that a fully connected network can arbitrarily accurately approximate any continuous function defined in a compact domain, even if it only has one hidden layer.

Although this result suggests that *shallow* networks (i.e., those with only a few hidden layers) are sufficient for our purposes, the required number of neurons to reach a good approximation may be prohibitively large. It can be shown that a *deep* fully connected network can reach the same expressivity of the shallow ones but with much fewer parameters [155]. However, the number of connection weights grows exponentially with the number of layers on a fully connected network. So, as highly performant models generally need a large number of layers, efficient training of these networks is also out of reach.

A *convolutional layer* [132] reduces the parameter space requiring (a) that each neuron can only combine output values from a local region of the previous layer, and (b) the weights performing these combinations are shared between neurons

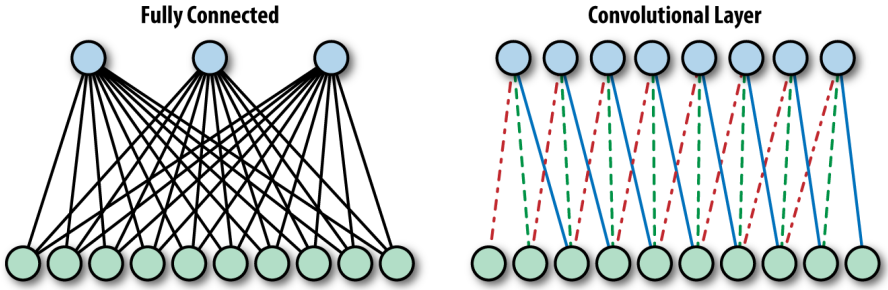


Figure 2.3: In a fully connected layer (left), each neuron is connected to all neurons of the previous layer. In a convolutional layer (middle), each neuron is connected to a fixed number of neurons in a local region of the previous layer. Furthermore, the neurons all share the weights for these connections, as represented by the color lines. Figure retrieved from [103].

on the layer (see Figure 2.3). This process corresponds to convolving the input with a small filter, so computing a kind of *feature extractor* (e.g. a edge detector), and can be related to how simple cells in the primary cortex of the mammalian visual system work [107, 106]. This is a especial case of a fully connected layer (2.13) where W is restricted to be a *Toeplitz matrix*. When a neural network has at least one convolutional layer, it is generally called *Convolutional Neural Network (CNN)*.

Assuming that the previous layer has dimension n, we can add a convolutional layer with filter of size k (so k is the number of parameters of this new layer) and then the layer output may also have size n (e.g. with zero padding). A fully connected layer can do the same but adding n^2 parameters. In addition to reducing the number of parameters and hence simplifying the learning procedure, convolutional layers are *shift-invariant* operators. When working with images, this is a nice property to have because important features can appear anywhere on the input plane. Hence, using sliding filters we can detect relevant features on the image regardless of its spatial location, so making the learning procedure more data-efficient. On the other hand, fully connected networks need to see images showing the same feature repeated on every spatial location to allow the neurons on each region to learn it, and then they need far more data to reach this translation invariant property.

Using small convolutional filters allow to construct deeper CNN (i.e. with a large number of convolutional layers) for a given number of parameters. This permits to get rid of the restricted local dependence of a particular layer and to enlarge the *receptive field* of the network (i.e. the number of pixels of the input image a single neuron has access to, see Figure 2.4). In addition, a CNN computes a hierarchical representation of the input, in which neurons on latter layers can recognize more abstract and complex features than those on the first layers [132]. The name *Deep Learning* usually means training a deep CNN.

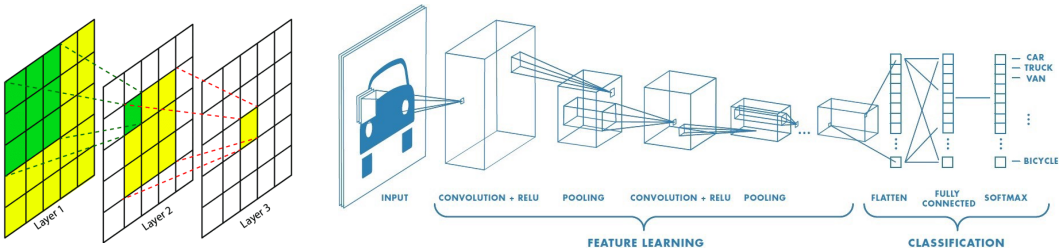


Figure 2.4: *Left*: On images, a convolutional layer computes a local feature detector filter. If we stack many convolutional layers, the receptive field of the network grows. *Right*: a common CNN arquitecture used for classification tasks, with fully connected layers at the end.

Additional building blocks are available in the literature to add to our network architecture: batch normalization [110], Dropout [209], other activation functions [177], weight initialization and regularization techniques [87], residual blocks and skip connections [100], and the list continues to grow. For a complete reference about Deep Learning, see [87].

Neural networks in imaging problems

The earliest attempts to use neural networks for image restoration problems date back to 1988 [248], where fully connected networks were used to represent binary images in a deblurring task. At that time, the lack of computational power only allowed to train simple architectures with few parameters, and therefore these methods were largely outperformed by variational methods. The field of neural networks continued to grow until 2012, when a groundbreaking paper [125] shows that it is possible to train a deep CNN with 60 million parameters using 1.2 million high-resolution images on a *Graphics Processing Unit (GPU)* which significantly outperforms previous state-of-the-art algorithms on the ImageNet LSVRC-2010 classification contest. Since deep CNN showed their superiority in image classification tasks researchers started to look for new ways to use this tool to solve inverse problems too.

The dominant approach is to *agnostically* train CNNs end-to-end for inverse problems in imaging. That is, we train a generic network $\boldsymbol{F}_\theta$ without incorporating any domain-specific knowledge of the problem at hand (e.g. the forward operator $\boldsymbol{A}$) using (2.10). This was done with empirical success for super-resolution [63], denoising [244, 243], deblurring [76], etc. Also, this method can be used to implement the entire camera image signal processing (ISP) pipeline [81, 199]. Other related problems are the so-called *image-to-image translations*. It consists in "translating" an input image belonging to one domain onto an output image from another domain (e.g. to transform a horse photo into a zebra photo). This particular prob-

lem can be very challenging to model analytically but there are many CNN-based methods that have shown impressive performance. See [163] for a recent review on this subject.

Solving inverse problems in imaging with end-to-end approaches is appealing because:

- In order to define and train neural networks, we do not need to know the forward degradation model (e.g. $\boldsymbol{A}$, noise variance σ^2). This is benefical when we do not have knowledge of the problem of interest or it is very hard to model analytically.
- After training, neural networks perform fast inference in the testing phase (as opposed to variational approaches which are computationally demanding).

On the other hand, employing end-to-end approaches presents some challenges:

- The universal approximation theorem ensures what NN *can* learn, not what they *actually* learn. In general, it is difficult to diagnose the convergence of the learning algorithm, which is a highly non-convex optimization process. Although some works conjecture that there might not be as many *bad* local minima as one imagines [229, 137], in practice, the training procedure of large models is so full of tips and tricks that it often feels like more of an art than a science [16].
- Training a deep CNN is a costly process and error prone, which has to be repeated on the particular inverse problem of interest before inference. Unfortunately, state-of-the-art deep CNN tend to be very big (i.e. billions of parameters) and the computational resources necessary to train these models in a reasonable period of time are not available to most researchers except those belonging to large companies (i.e. Google, Facebook, NVIDIA, OpenAI, etc).
- They are often vulnerable to *adversarial attacks*, i.e., hardly perceptible perturbations at the input may cause the network to output very different responses [212]. These instabilities raise several concerns about the robustness of these models, primarily in critical applications such as medical image reconstructions and clinical diagnostics [72, 8] or self-driving cars.
- When no prior knowledge is included in the architecture design, the network must learn all aspects of the inverse problem to be solved from the available data. Therefore, the network will be more "data hungry", i.e. the training dataset must be large to obtain good results. To be more data-efficient, if we incorporate somehow all the available information into the learning process (e.g. degradation model $\boldsymbol{A}$) the network only needs to learn what really

needs to be learned. A clear example of a generic prior information put into network design is the use of convolutional layers instead of fully connected ones.

Incorporating prior knowledge

A direct way to incorporate the degradation operator $\boldsymbol{A}$ is to *post-process* its direct inversion method, if available. For example, one can compute $\tilde{\boldsymbol{x}} = \boldsymbol{A}^T\boldsymbol{y}$ or $\tilde{\boldsymbol{x}} = \boldsymbol{A}^+\boldsymbol{y}$ (where $\boldsymbol{A}^+$ is the Moore-Penrose pseudo inverse of $\boldsymbol{A}$) to put data $\boldsymbol{y}$ back into the image domain. This first reconstruction may be prone to artifacts, hence we train a neural network $\boldsymbol{F}_\theta(\tilde{\boldsymbol{x}})$ to correct this naive reconstruction [111]. Since the map $\tilde{\boldsymbol{x}} \mapsto \boldsymbol{x}$ is close to the identity map, we are learning a residual network that is reportedly easier to train [100].

Another interesting approach for including the degradation model in the network architecture is the use of *unrolled optimization techniques* [91, 42, 60, 84]. They basically consist in implementing a fixed number of steps of an iterative optimization method using a neural network. This way, the data fidelity step is included in the network as a layer with known parameters (that do not need to be learned) and only the prior step becomes a trainable layer. For this reason, unrolled approaches tend to require much less training data to reach the same accuracy as agnostic ones.

Plug and Play methods

The main drawback of neural networks regression is that they require to retrain the neural network each time a single parameter of the degradation model changes. Another family of approaches seeks to *decouple* the NN-based learned image prior from the degradation model, so they can be used to solve many different inverse problems without retraining. A popular approach within this methodology are *Plug and Play (PnP)* methods. Instead of directly learning the regularization term $G(\boldsymbol{x})$, these methods seek to learn an approximation of its gradient ∇G [21, 20] or its proximal operator prox_G [227, 149, 245, 40, 114, 192], by replacing it by a denoising NN. Then, these approximations are used in an iterative optimization algorithm to find some sort of consensus equilibrium among the data fitting term and the prior [32]. Taking a similar approach, the Regularization by Denoising (RED) algorithm [187] uses a denoiser D_σ to construct an explicit regularizer $G(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T(\boldsymbol{x} - D_\sigma(\boldsymbol{x}))$. Under certain conditions, its gradient $\nabla G = Id - D_\sigma$ can be conveniently computed in terms of the denoiser, leading to a gradient descent scheme which is very easy to implement.

Implicit regularization methods

Finally, another way to regularize the inverse problem is using a constrained representation model for the image $\boldsymbol{x}$. An interesting work suggests that the architecture of an untrained (i.e. randomly initialized) CNN acts as a kind of implicit regularization method, called *Deep Image Prior (DIP)* [223]. Here, one trains the network only using the corrupted data $\boldsymbol{y}$ to map a fixed random vector $\boldsymbol{z}_0$ to an image $\boldsymbol{y}$ using gradient descent:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} f(\boldsymbol{A}\boldsymbol{F}_\theta(\boldsymbol{z}_0), \boldsymbol{y}) \tag{2.14}$$

where f is a data fidelity measure like in the variational formulation (2.7). Once trained, the output of the network is the reconstructed image $\hat{\boldsymbol{x}} = \boldsymbol{F}_{\hat{\theta}}(\boldsymbol{z}_0)$. This algorithm can be written as a variational (regularized) method:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \{f(\boldsymbol{A}\boldsymbol{x}, \boldsymbol{y}) + G(\boldsymbol{x})\} \tag{2.15}$$

where

$$G(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \boldsymbol{x} = \boldsymbol{F}_\theta(\boldsymbol{z}_0) \text{ for some } \theta \in \Theta, \\ +\infty, & \text{if not.} \end{cases} \tag{2.16}$$

Other approach is to use a coordinate-based image representation $\boldsymbol{x}_{i,j} = \Phi_\theta(i, j)$, where Φ_θ is a neural network with sinusoidal activation functions like SIREN [202]. These architectures are built by forcing all the spectral components of the image $\boldsymbol{x}$ to be present, using a *Fourier features* layer [213], and therefore facilitating the restoration of high frequencies on the retrieved image. This could compensate for the oversmoothing behavior of a NN trained using MSE as the image quality measure, a kind of *spectral bias* [33, 175]. As with DIP, the regularization is imposed on the class of reachable images using these architectures as representation models.

2.2.3 Bayesian approach

Statistical modeling permits to make the restoration algorithms more flexible, allowing us to consider more than one possible solution compatible with the data, which is appropriate for working with ill-posed problems. In particular, we can define our prior belief about the unknown signal in the form of a probability distribution, called *prior distribution*. This distribution, when combined with the observed data in the form of the *likelihood function*, gives rise to the so-called *posterior distribution*.

In this way, we are naturally characterizing the entire solution space, and we have a solid theory to develop different estimators for the target image with desirable properties. Hence, we can calculate different point estimates, such as the

MAP or MMSE estimators, but we can also compute uncertainty estimates in the form of confidence intervals.

From a probabilistic point of view, we now consider a stochastic model

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \eta, \qquad \boldsymbol{x} \sim p_{\mathcal{X}}(\boldsymbol{x}), \qquad \eta \sim \mathcal{N}(0, \sigma^2 I), \tag{2.17}$$

where $p_{\mathcal{X}}(\boldsymbol{x})$ is the probability distribution of the images of interest. Note that in (2.17) $\boldsymbol{x}$ and $\boldsymbol{y}$ are now random variables defined on probability spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively.[1]

Likelihood function

Let $\boldsymbol{x}^* \in \mathcal{X}$ be the unknown image to be retrieved, and $\boldsymbol{y}$ the observed value for the measurements, as dictated by the stochastic model (2.17). The problem now is to infer as much as possible about $\boldsymbol{x}^*$. The *likelihood function* measures the fitness of a value $\boldsymbol{x}$ to the observed data $\boldsymbol{y}$ by

$$\ell(\boldsymbol{x} \mid \boldsymbol{y}) = p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{A}\boldsymbol{x}, \sigma^2) \propto e^{-\frac{\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2}}. \tag{2.18}$$

The classical *Maximum Likelihood Estimator (MLE)* corresponds to

$$\hat{\boldsymbol{x}}_{\text{MLE}} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \ell(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \{-\log \ell(\boldsymbol{x} \mid \boldsymbol{y})\} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \frac{1}{\sigma^2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 \right\}. \tag{2.19}$$

Therefore, this estimator $\hat{\boldsymbol{x}}_{\text{MLE}}$ maximizes the agreement of an image $\boldsymbol{x}$ with the measurements $\boldsymbol{y}$. Observe that this estimator is equivalent to the least squares solution of (2.6). As with variational methods, when dealing with ill-posed problems, there may be a large set of compatible solution of (2.19) and/or $\hat{\boldsymbol{x}}_{\text{MLE}}$ may be far away from $\boldsymbol{x}^*$. Hence, to obtain appropiate restorations $\hat{\boldsymbol{x}}$ we need to incorporate more information about $\boldsymbol{x}^*$ to the inverse problem.

Prior distribution

The *prior distribution* is a way to describe in probabilistic terms our belief about the unknown image $\boldsymbol{x}^*$, before observing the measurements $\boldsymbol{y}$. When combined with the likelihood function, this distribution will help us find the correct solution despite the ambiguity left by incomplete and/or noisy data $\boldsymbol{y}$. Note that this distribution does not depend on the particular inverse problem at hand and therefore

[1] We do not use the usual probabilistic convention that capital letters X represent random variables, as in $P(X \leq \boldsymbol{x})$. Instead, we use $\boldsymbol{x}$ to refer to the random variable as well as its observed value, and $\boldsymbol{x} \sim f(\boldsymbol{x})$ to denote its density function $f(\boldsymbol{x})$, the meaning being clear for the context.

it can be used to regularize any inverse problem for which the unknown image $\boldsymbol{x}^*$ comes from the same source (e.g. satellite imagery, faces, etc).

In (2.17) we have already assumed that $\boldsymbol{x}^*$ is distributed according to a probability distribution $p_{\mathcal{X}}(\boldsymbol{x})$, which in the Bayesian framework is the so-called *prior distribution*. This is a convention we have adopted for the source of images of interest: they are i.i.d. realizations of a (continuous) random variable with density function $p_{\mathcal{X}}(\boldsymbol{x})$. Quoting from [182], *"this is an axiomatic reduction from the notion of unknown to the notion of random"*. In order to include this distribution in our restoration methods, the problem that arises is how to choose this distribution. As it is the central concern of this restoration method, we leave this discussion for the end of this section (and for the rest of the thesis!), after finishing our description of the Bayesian approach.

Posterior distribution

The *Bayes' formula* can be seen as a way of updating our prior belief about $\boldsymbol{x}^*$ after observing the measurements $\boldsymbol{y}$ (through the forward process modeled by (2.17)):

$$p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})\, p_{\mathcal{X}}(\boldsymbol{x})}{\int_{\boldsymbol{x}' \in \mathcal{X}} p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x}')\, p_{\mathcal{X}}(\boldsymbol{x}')\; d\boldsymbol{x}'}. \tag{2.20}$$

Here, $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$ is known as the *posterior distribution* of $\boldsymbol{x}$ given $\boldsymbol{y}$.

This resulting distribution collects all the uncertainty we have about the unknown signal or process we would like to infer, and therefore contains significant information to retrieve $\boldsymbol{x}^*$ from $\boldsymbol{y}$. An illustration of this approach is shown in Figure 2.5. In the Bayesian framework, the posterior distribution is the complete solution to the inverse problem. In practice, some summaries of it (like point estimates and confidence intervals) are computed.

Bayesian estimators

From a decision-theoretic point of view, we need to condensate all the information included in the posterior distribution into one solution $\hat{\boldsymbol{x}}$. The question to which we are now confronted is: *How do we select $\hat{\boldsymbol{x}}$?* To this end, we have to define a cost function and calculate the optimal solution as the one that minimizes the expectation of that cost under the posterior distribution [118].

As always, let $\boldsymbol{x}^*$ be the unknown image to be retrieved from the measurements $\boldsymbol{y}$ and an estimator $\hat{\boldsymbol{x}}$. Given a *cost function* $\mathcal{C}(\epsilon)$ that assigns the cost of the error $\epsilon = \boldsymbol{x} - \hat{\boldsymbol{x}}$ for (a sample of) the random variable $\boldsymbol{x}$, the corresponding *Bayes risk*

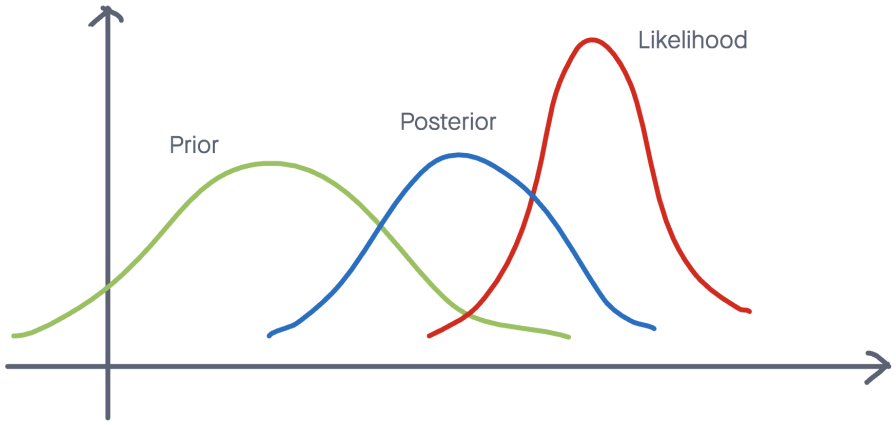


Figure 2.5: *Main distributions of the Bayesian approach*: The prior distribution is updated using the likelihood function, giving rise to the posterior distribution.

is defined by

$$\mathcal{R}_{\mathcal{C}}(\hat{\boldsymbol{x}}) = \int_{(\boldsymbol{x},\boldsymbol{y})\in(\mathcal{X},\mathcal{Y})} \mathcal{C}(\boldsymbol{x}-\hat{\boldsymbol{x}}) p_{\mathcal{X},\mathcal{Y}}(\boldsymbol{x},\boldsymbol{y})\; d\boldsymbol{x}\, d\boldsymbol{y} \tag{2.21}$$

$$= \int_{\boldsymbol{y}\in\mathcal{Y}} \underbrace{\left(\int_{\boldsymbol{x}\in\mathcal{X}} \mathcal{C}(\boldsymbol{x}-\hat{\boldsymbol{x}}) p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x}\,|\,\boldsymbol{y})\; d\boldsymbol{x} \right)}_{I_{\boldsymbol{y}}(\hat{\boldsymbol{x}})} p_{\mathcal{Y}}(\boldsymbol{y})\; d\boldsymbol{y}. \tag{2.22}$$

Each choice of the cost function $\mathcal{C}$ leads to an optimal *Bayes estimator* which minimizes the Bayes risk:

$$\hat{\boldsymbol{x}}_{\mathcal{C}} = \arg\min_{\hat{\boldsymbol{x}}} \mathcal{R}_{\mathcal{C}}(\hat{\boldsymbol{x}}). \tag{2.23}$$

Observe that if we can construct an estimator that minimizes $I_{\boldsymbol{y}}(\hat{\boldsymbol{x}})$ for all $\boldsymbol{y}\in\mathcal{Y}$, then it will be the Bayes estimator $\hat{\boldsymbol{x}}_{\mathcal{C}}$. The main examples are (see Figure 2.6):

- **MMSE estimator**: When $\mathcal{C}(\epsilon) = \|\epsilon\|^2$ we obtain the *Minimum Mean Squared Error (MMSE)* estimator $\hat{\boldsymbol{x}}_{\text{MMSE}}$. Deriving $I_{\boldsymbol{y}}(\hat{\boldsymbol{x}})$:

$$\frac{\partial I_{\boldsymbol{y}}}{\partial \hat{\boldsymbol{x}}}(\hat{\boldsymbol{x}}) = \int_{\boldsymbol{x}\in\mathcal{X}} \frac{\partial}{\partial \hat{\boldsymbol{x}}} \|\boldsymbol{x}-\hat{\boldsymbol{x}}\|^2 p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x}\,|\,\boldsymbol{y})\; d\boldsymbol{x} \tag{2.24}$$

$$= \int_{\boldsymbol{x}\in\mathcal{X}} -2(\boldsymbol{x}-\hat{\boldsymbol{x}}) p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x}\,|\,\boldsymbol{y})\; d\boldsymbol{x} \tag{2.25}$$

$$= 2\hat{\boldsymbol{x}} - 2\int_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{x}\; p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x}\,|\,\boldsymbol{y})\; d\boldsymbol{x} = 0. \tag{2.26}$$

  Hence, the MMSE estimator is the posterior mean (or conditional expectation):

$$\hat{\boldsymbol{x}}_{\text{MMSE}} = \mathbb{E}_{p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x}\,|\,\boldsymbol{y})}[\boldsymbol{x}] = \int_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{x}\; p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x}\,|\,\boldsymbol{y})\; d\boldsymbol{x}. \tag{2.27}$$

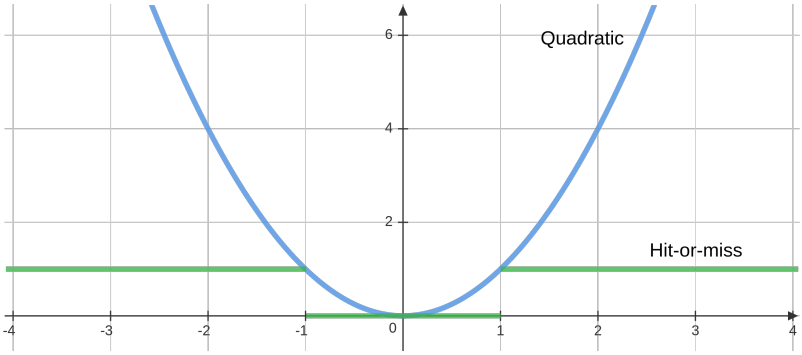


Figure 2.6: *Bayesian cost functions*: The quadratic function (blue) penalizes more heavely the larger errors, while the hit-and-miss function (green) penalizes all errors above the threshold (in this example, $\delta = 1$) with cost 1.

- **MAP estimator**: Another common choice is the *hit-or-miss* cost function. It does not penalizes absolute errors below a predefined threshold $\delta > 0$ and assigns cost 1 for all errors above this threshold:

$$\mathcal{C}(\epsilon) = \begin{cases} 0 & \text{if } \|\epsilon\| < \delta, \\ 1 & \text{if } \|\epsilon\| \geq \delta. \end{cases} \tag{2.28}$$

  Then, the Bayes risk (2.22) becomes

$$\mathcal{R}_{\mathcal{C}}(\hat{\boldsymbol{x}}) = 1 - \int_{\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|<\delta} p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x} \,|\, \boldsymbol{y}\right)\, d\boldsymbol{x}. \tag{2.29}$$

  As $\delta \to 0$, this risk is minimum when $\hat{\boldsymbol{x}}$ maximizes the posterior distribution, leading to the so-called *Maximum A Posteriori (MAP)* estimator:

$$\hat{\boldsymbol{x}}_{\text{MAP}} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x} \,|\, \boldsymbol{y}\right). \tag{2.30}$$

MAP vs MMSE

In general, to approximate the $\hat{\boldsymbol{x}}_{\text{MMSE}}$ estimator in (2.27) it is necessary to use sampling algorithms to calculate Monte Carlo (i.e. numerical) integrations, or to use variational methods to calculate analytical (i.e. exact) expectations of approximate distributions of $p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x} \,|\, \boldsymbol{y}\right)$. On the other hand, the optimization problem (2.30) for computing the $\hat{\boldsymbol{x}}_{\text{MAP}}$ estimator is often easier to solve than the expectation in (2.27). For this reason, the $\hat{\boldsymbol{x}}_{\text{MAP}}$ estimator is more popular in the literature than the $\hat{\boldsymbol{x}}_{\text{MMSE}}$ estimator.

Observe that (2.30) is equivalent to compute

$$\hat{\boldsymbol{x}}_{\text{MAP}} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\log p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x} \mid \boldsymbol{y}\right) \right\} \tag{2.31}$$

$$= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ -\log p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right) - \log p_{\mathcal{X}}\left(\boldsymbol{x}\right) \right\} \tag{2.32}$$

$$= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \frac{1}{\sigma^2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 + R(\boldsymbol{x}) \right\} \tag{2.33}$$

which is closely related to the variational formulation (2.7). Here, the negative log-likelihood function acts as the data fitting term and the negative log-prior function as the regularization term.

An interesting property of the posterior distribution $p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x} \mid \boldsymbol{y}\right)$ is that it naturally balances the weights put on the data and on the prior. For example, as we take more measurements (i.e. $l = \dim(\mathcal{Y})$ grows) and the noise vanishes ($\sigma \to 0$), the first term of (2.33) dominates (i.e. the prior term $R(\boldsymbol{x})$ diminishes) and $\hat{\boldsymbol{x}}_{\text{MAP}} \to \hat{\boldsymbol{x}}_{\text{MLE}}$. In particular, when there is less ambigüity on the data, the posterior distribution gets more concentrated around the target value $\boldsymbol{x}^*$. Therefore, we do not need to manually set a regularization parameter such as λ in the variational approach (2.7).

We conclude by observing that, although in general $\hat{\boldsymbol{x}}_{\text{MMSE}} \neq \hat{\boldsymbol{x}}_{\text{MAP}}$ for the *same* prior $p_{\mathcal{X}}\left(\boldsymbol{x}\right)$, these estimators can be related for *different* priors, as the conditional expectation (2.27) for one prior can also be interpreted as a MAP estimator for the other prior. In other words, it can be shown that, for Gaussian denoising (i.e. $\boldsymbol{A} = I$), the posterior mean (2.27) coincides with the solution of the optimization problem (2.33) for an appropriate penalty function $R_{\text{MMSE}}(\boldsymbol{x})$ [92, 93].

¿How to choose the prior distribution?

Here we discuss some popular ways to estimate the probability density function $p_{\mathcal{X}}\left(\boldsymbol{x}\right)$. Clearly, this is not an easy task in high-dimensional spaces, and general methods such as kernel density estimation (e.g. Parzen windows) require a huge amount of data points to be accurate.

As we mentioned before, the MAP formulation (2.33) corresponds to a variational problem with regularization penalty term $R(\boldsymbol{x}) = -\log p_{\mathcal{X}}\left(\boldsymbol{x}\right)$. Hence, the reconstruction algorithms presented in Section 2.2.1 can be interpreted as MAP estimators under the prior $p_{\mathcal{X}}\left(\boldsymbol{x}\right) \propto e^{-R(\boldsymbol{x})}$.[2] For example, setting the Tikhonov regularization $R(\boldsymbol{x}) = \|L\boldsymbol{x}\|_2^2$ seen on Section 2.2.1 corresponds to choosing a (possibly degenerate) Gaussian prior on $\boldsymbol{x}$, and the L^1 regularization term $G(\boldsymbol{x}) =$

[2] As mentioned above, the MAP formulation of a variational problem may not be the only Bayesian interpretation for (2.33), because in order to define Bayesian estimators we need to define the prior but also the cost function.

$\|L\boldsymbol{x}\|_1$ to a Laplacian prior distribution. When $e^{-R(\boldsymbol{x})}$ are non-integrable functions, we are working with *improper priors*.

As with variational methods, it is very difficult to define a precise hand-crafted model $p_{\mathcal{X}}(\boldsymbol{x})$ for natural images. A common approach is to fit a parametric model to the set of image *patches*, that is, all the small square portions of the image (say 7×7). We then restore each patch separately and merge all the results to form the final result of the restoration algorithm. This is the case with the widely known *Non-Local Means (NLM)* [29], *Non-Local Bayes (NLB)* [131] and *Block-Matching and 3D filtering (BM3D)* [50] denoising algorithms. In particular, we will adapt the NLB algorithm to the wavelet domain in Section 3.2.5.

The parametric models described above are fitted to the set of patches of the noisy image to be restored. Also, it can be fitted once in a large database of image patches (e.g. ImageNet [55]) and then used to restore any image that follows the same distribution. In [69] the authors learn a dictionary of representative patches and then impose a sparsity ℓ^1 prior on the corresponding coefficients. Also, in the *Expected Patch Log Likelihood (EPLL)* method [249] a Gaussian mixture model is first fitted to image patches and then used to define a prior for the whole image (i.e. without the need of combining different patch restorations at the end).

A modern approach is to fit deep learning-based distributions on whole images, giving rise to the *generative models*. As these models are an important component of this thesis, we dedicate the following section to its description and give some motivation on why to use them to regularize inverse problems in imaging.

2.3 Generative models

Recently, some generative models based on neural networks have shown their outstanding capability to approximate the complex and high-dimensional image distribution in a data-driven fashion. In particular, *Variational Autoencoders (VAE)* [121] combine variational inference to approximate unknown posterior distributions of latent variable models with the ability of neural networks to learn such approximations. In this section, we review the most performant methods to learn the distribution of images.

2.3.1 Latent variable models

Latent generative models simulate the way images are generated and interpreted by humans: images are flat projections of 3D objects on which there are some high-level concepts or structures present in the image. We do not interpret images as arrays of independent pixels as they are stored on computers, but as the overlapping of global shapes with long-range pixel dependencies, each of which has its

own fine (local) details or textures. For us, for example, lighting and perspectives are latent variables that cause non-trivial global modifications to the entire image in the pixel space.

It is easy for humans to extrapolate knowledge from one domain to another because we form high-level internal (i.e. latent) representations of the objects we usually see. In this way we can learn to solve problems in new domains with little labeled data, with the help of such representations. Most of the available data is unlabeled, so an end-to-end approach which is data-hungry is not feasible on some domains (e.g. medical imaging). Therefore, it is desirable to develop latent representation models so that they can be reused in different inverse problem where the images follow the same distribution. Additionally, using this latent representations we can easily modify the prior distribution conditioning on high-level abstract features to reduce ambigüity (e.g. if we know that the solution must be the picture of a woman face). How to automatically factorize these sources of variations from samples is an active area of research [57, 136, 135].

Training latent generative models on generic data and then using them to regularize other inverse problems enables data efficiency in the new domain and generalization, as well as providing robustness to reconstruction algorithms. By training on a large image database (e.g. ImageNet [55]), ideally, we can capture the entire image distribution. In practice, a reconstruction algorithm is not trained on huge datasets but on relatively smaller ones. In this way, "gaps" can remain in the trained model (that is, areas of the image space where we do not have reconstruction examples). Using the larger model mentioned above, we can generalize better on those gaps than with the smaller model. This approach is related to the *transfer learning* technique used in classification tasks, where a pre-trained model for image classification is fine-tuned to include new categories [221].

Dimensionality reduction

The image representation space $\mathcal{X} \subseteq \mathbb{R}^d$ has very high dimension for all practical purposes. For example, for 1 megapixel RGB images (e.g. 1000×1000 color pictures) which are small ones considering modern cameras, we have $d = 3 \times 10^6$. However, not all the points $\boldsymbol{x}$ but only a very small subset $\mathcal{X}_0$ of $\mathcal{X}$ represent images of interest. Moreover, the *manifold hypothesis* [71] states that natural images lie near a low dimensional manifold. Therefore, it is desirable to define a prior distribution or density $p_{\mathcal{X}}(\boldsymbol{x})$ only on this subset so that the inverse problem could be posed in a lower-dimensional space. A first step towards an efficient representation of this subset $\mathcal{X}_0$ is to apply a *dimensionality reduction* transform, also known as *feature extraction*. Formally, we seek for a function $\mathsf{E}\colon \mathcal{X} \to \mathcal{Z} \subseteq \mathbb{R}^p$ with $p < d$ which we call *encoder* such as $\mathsf{E}(\mathcal{X}_0)$ retains as much information about $\mathcal{X}_0$ as possible. Ideally, if $\boldsymbol{x} \in \mathcal{X}_0$ then we want to be able to reconstruct $\boldsymbol{x}$ from $\boldsymbol{z} = \mathsf{E}(\boldsymbol{x})$ using a *decoder* function $\mathsf{D}\colon \mathcal{Z} \to \mathcal{X}$ such that $\tilde{\boldsymbol{x}} := \mathsf{D}(\boldsymbol{z}) \simeq \boldsymbol{x}$.

Principal Component Analysis (PCA)

The simplest case is to choose linear (affine) transforms for E and D and the squared Euclidean distance as the reconstruction error. That is

$$\boldsymbol{z} = \mathsf{E}(\boldsymbol{x}) = U\boldsymbol{x} + u, \qquad \text{where } U \in \mathbb{R}^{p\times d} \text{ and } u \in \mathbb{R}^p, \quad (2.34)$$

$$\tilde{\boldsymbol{x}} = \mathsf{D}(\boldsymbol{z}) = V\boldsymbol{z} + v, \qquad \text{where } V \in \mathbb{R}^{d\times p} \text{ and } v \in \mathbb{R}^d, \quad (2.35)$$

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2 = \|\boldsymbol{x} - (W\boldsymbol{x} + w)\|^2 \qquad \text{where } W = VU \text{ and } w = Vu + v. \quad (2.36)$$

Geometrically, $\tilde{\boldsymbol{x}}$ corresponds to project $\boldsymbol{x}$ into the (affine) subspace of $\mathbb{R}^d$ given by the columns of V which has rank at most p. As $VU = (VC)(C^{-1}U)$ for every invertible matrix $C \in \mathbb{R}^{p\times p}$ then U and V are not unique. Also, it is clear that better reconstructions can be achieved if V has full rank p because in that case the latent code $\boldsymbol{z}$ can capture as much information about $\boldsymbol{x}$ as possible (otherwise some dimensions of $\boldsymbol{z}$ are wasted because linear dependence). Hence, without loss of generallity, we can assume V to have orthonormal columns.

Having a sample dataset $\mathcal{D} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \mathcal{X}_0$ we try to find the best pair (E, D) that minimizes

$$\mathcal{L}_{\mathcal{D}}(\mathsf{E}, \mathsf{D}) = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|^2 = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{x}_i - (W\boldsymbol{x}_i + w)\|^2. \quad (2.37)$$

It can be shown [12] that the optimal solution for the columns of V (up to coordinate changes through C) is to choose the eigenvectors corresponding to the p larger eigenvalues of the sample covariance matrix:

$$\Sigma_X = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \quad \text{where } \bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}. \quad (2.38)$$

In this case, we obtain

$$\boldsymbol{z} = V^T(\boldsymbol{x} - \bar{\boldsymbol{x}}), \quad \tilde{\boldsymbol{x}} = V\boldsymbol{z} + \bar{\boldsymbol{x}}. \quad (2.39)$$

This is the well known *Principal Component Analysis (PCA)* method [22].

Factor Analysis and Probabilistic PCA (pPCA)

From a stochastic perspective, we can assume that a latent variable $\boldsymbol{z}$ generates an observation $\boldsymbol{x}$, which both follow probability distributions, and therefore we can try to fit a statistical model to the data $\mathcal{D}$. Closely related to PCA, the *Factor Analysis* model [43] that assumes $\boldsymbol{z}$ to follow a Gaussian distribution and a linear (conditional) observation model with Gaussian noise:

$$p_{\mathcal{Z}}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mu, \Sigma) \quad (2.40)$$

$$p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; V\boldsymbol{z} + v, \Psi). \quad (2.41)$$

In this case, it is well known that the marginal $p_{\mathcal{X}}(\boldsymbol{x})$ and posterior $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$ distributions are also Gaussian and have the following expressions ([22], equations (2.113) to (2.117)):

$$p_{\mathcal{X}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; V\mu + v, \Psi + V\Sigma V^T) \tag{2.42}$$
$$p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; M(V^T\Psi^{-1}(x - v) + \Sigma^{-1}\mu), M) \tag{2.43}$$
$$\text{where } M = (\Sigma^{-1} + V^T\Psi^{-1}V)^{-1}. \tag{2.44}$$

The *Probabilistic PCA (pPCA)* model [220] is a particular case of factor analysis where the observation noise is assumed to have an spherical covariance $\gamma^2 I$ instead of a full covariance matrix Ψ. In addition, as all the parameters of this model has to be learned from data and we only have observed values for $\boldsymbol{x}$, then ν and Σ can be absorbed by V and v. Hence we can assume $\mu = 0$, $\Sigma = I$ and $\Psi = \gamma^2 I$ which lead to the following simplified pPCA model

$$p_{\mathcal{Z}}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; 0, I) \tag{2.45}$$
$$p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; V\boldsymbol{z} + v, \gamma^2 I). \tag{2.46}$$

and

$$p_{\mathcal{X}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; v, VV^T + \gamma^2 I) \tag{2.47}$$
$$p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; MV^T(x - v), \gamma^2 M) \tag{2.48}$$
$$\text{where } M = (V^T V + \gamma^2 I)^{-1}. \tag{2.49}$$

In this context, $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$ and $p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z})$ can be interpreted as *stochastic* encoder and decoder respectively.

Given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathcal{X}_0$ the maximum likehood estimation (MLE) of the model parameters V, v and γ^2 (maximizing $p_{\mathcal{X}}(\boldsymbol{x})$ on the dataset $\mathcal{D}$) can be computed in closed form [220]:

$$\hat{v} = \bar{\boldsymbol{x}}, \tag{2.50}$$
$$\hat{\gamma}^2 = \frac{1}{d-p}\sum_{j=p+1}^{d} \lambda_j, \tag{2.51}$$
$$\hat{V} = U_p(\Lambda_p - \hat{\gamma}^2 I)^{1/2} R. \tag{2.52}$$

Here, U_p corresponds to the first p principal components of the data with corresponding eigenvalues $\lambda_1, \ldots, \lambda_p$ (as in PCA) stored in the $p \times p$ diagonal matrix Λ_p, and R is an arbitrary rotation matrix. Observe that $\hat{\gamma}^2$ can be interpreted as the average variance of the data lost in the projection of $\mathcal{D}$ into the span of the columns of V. Unlike pPCA, the maximum likelihood estimation of a generic factor analysis model (2.42) can not be computed in closed form and therefore they must be approximated using an iterative procedure like the *Expectation-Maximization (EM)* algorithm [82].

2.3.2 Variational AutoEncoders

The models described in the previous section assume linear encoder and decoder functions. However, in most practical applications, the data $\mathcal{X}_0$ of interest does not lie in a linear manifold but in a highly non-linear one. For example, the set of 3×3 high-contrast patches is better described using a Klein bottle [134, 34]. Hence, to better capture the nature of real data and therefore construct more accurate prior distributions, we need to fit more flexible statistical models. In this section, we explore the use of neural networks to perform dimensionality reduction in a data-driven fashion.

AutoEncoders (AE)

We revisit the deterministic encoder/decoder approach defined in the previous section, this time considering more general parametric functions:

$$\mathsf{E}_\phi \colon \mathcal{X} \to \mathcal{Z}, \quad \mathsf{E}_\phi(\boldsymbol{x}) = \boldsymbol{z} \qquad \text{(Encoder)} \tag{2.53}$$

$$\mathsf{D}_\theta \colon \mathcal{Z} \to \mathcal{X}, \quad \mathsf{D}_\theta(\boldsymbol{z}) = \tilde{\boldsymbol{x}} \qquad \text{(Decoder).} \tag{2.54}$$

Here, ϕ and θ collect all the encoder and decoder parameters respectively. In the PCA case of equations (2.34) and (2.35), $\phi = (U, u)$ and $\theta = (V, v)$. When E_ϕ and D_θ are parameterized by neural networks, the pair (E, D) is called *AutoEncoder (AE)*. The training of the autoencoder given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ is generally done using a stochastic gradient descent algorithm on the L^2 reconstruction loss:

$$(\hat{\phi}, \hat{\theta}) = \arg\min_{\phi,\theta} \left\{ \mathcal{L}_{\mathcal{D}}(\phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \mathsf{D}(\mathsf{E}(\boldsymbol{x}_i))\|^2. \right\}. \tag{2.55}$$

After the training is finished, we (hopefully) obtain a parameterization of the data manifold $\mathsf{D}_{\hat{\theta}}(\mathcal{Z}) \subset \mathcal{X}$ of dimension $p < d$. In Figure 2.7 we show a comparison of the AE and PCA solutions to dimensionality reduction on the MNIST dataset [56] from $d = 28 \times 28 = 784$ to $p = 30$.

As we can not compute closed form expressions for the parameters ϕ and θ, we are not forced to use the L^2 reconstruction norm which generally leads to blurry reconstructions $\tilde{\boldsymbol{x}}$. Instead, we can use any differentiable metric $d(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ such as distances defined on wavelet or Laplacian pyramid coefficients [23] or perceptual distances (see Section 2.2.2).

Using the great flexibility of neural networks, autoencoders greatly expand the class of data sets from which we can learn a representation in smaller-dimensional spaces, losing as little information as possible. Next, we endow the autoencoder model with probability distributions to construct a density estimation algorithm known as *Variational AutoEncoder (VAE)*.

Figure 2.7: *AE and PCA solutions on MNIST*. From top to bottom: original digits $\boldsymbol{x}_i$, autoencoder reconstructions $\mathsf{D}_\theta(\mathsf{E}_\phi(\boldsymbol{x}_i))$ using 2-hidden layers fully connected networks for encoder and decoder, and PCA reconstructions (equation (2.39)), with $p = 30$ latent dimensions in both cases. The $\mathcal{X}$ space has dimension $d = 28 \times 28 = 784$.

Stochastic Decoder

Let $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathcal{X} \subseteq \mathbb{R}^d$ be samples drawn from an unknown probability distribution p_{data} supported on a manifold of dimension p. The objective is to construct a parametric approximation $p_{\mathcal{X}}(\boldsymbol{x}) \simeq p_{data}(\boldsymbol{x})$. As before, we assume that there exists a latent variable $\boldsymbol{z}$ of dimension p with a tractable distribution that generates $\boldsymbol{x}$. For $p = d$, if the p_{data} has density

$$p_{data}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p) = p_1(\boldsymbol{x}_1) p_2(\boldsymbol{x}_2|\boldsymbol{x}_1) \ldots p_p(\boldsymbol{x}_p|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{p-1}) \tag{2.56}$$

where p_i is the conditional density of $\boldsymbol{x}_i$ given $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1})$ then we can sample $\boldsymbol{z} \sim \text{Uniform}[0,1]^p$ and transform $\boldsymbol{z}$ into a sample of p_{data} applying the *inversion method* to each conditional density p_i of (2.56). This procedure is known as the *conditional distribution method* [58]. For $p < d$ this can be composed with a parameterization of the manifold supporting p_{data}.

It is common to choose $\boldsymbol{z} \sim \mathcal{N}(0, I)$. We can make this assumption because the stochastic decoder $p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z})$ may eventually capture the relationship between our variable $\boldsymbol{z}$ and the *real* latent variable. Hence, we are heavily relying on the universal approximation property of neural networks to learn the parameterization of the manifold supporting p_{data} and, at the same time, the transform that maps samples from a Gaussian distribution into p_{data}.

As we did with the pPCA decoder (2.46), we define a stochastic decoder D_θ which represents the Gaussian conditional distribution $p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z})$ using a neural network with parameters θ:

$$p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}; \theta) = \mathcal{N}(\boldsymbol{x}; \mu_\theta(\boldsymbol{z}), \gamma^2 I), \qquad \mathsf{D}_\theta(\boldsymbol{z}) = (\mu_\theta(\boldsymbol{z}), \gamma) \tag{2.57}$$

where $\gamma \in \theta$ is a trainable parameter but independent of $\boldsymbol{z}$. In what follows, to simplify notation, we write $p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}; \theta)$ and $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) p_{\mathcal{Z}}(\boldsymbol{z})$.

Hence, the marginal likelihood of $\boldsymbol{x}$ is

$$p_\theta(\boldsymbol{x}) = \int_{\mathcal{Z}} p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})\; d\boldsymbol{z}. \tag{2.58}$$

This integral is intractable so we can not directly compute the maximum likelihood estimation for θ. We can not even compute the gradient $\nabla_\theta\, p_\theta(\boldsymbol{x})$ to train the decoder network. This intractability is related to that of the posterior

$$p_\theta(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})}{p_\theta(\boldsymbol{x})}. \tag{2.59}$$

Variational Inference and Stochastic Encoder

Given $\boldsymbol{x}$, to provide an analytical approximation to the posterior distribution $p_\theta(\boldsymbol{z} \mid \boldsymbol{x})$ of the latent variables $\boldsymbol{z}$, we choose a tractable *variational model* $q(\boldsymbol{z})$. Using the concept of *amortized inference* this model is constructed from $\boldsymbol{x}$ using another neural network, the stochastic encoder $\mathsf{E}_\phi(\boldsymbol{x})$, which shares the network parameters ϕ between all values of $\boldsymbol{x}$:

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \mu_\phi(\boldsymbol{x}), \Sigma_\phi(\boldsymbol{x})), \qquad \mathsf{E}_\phi(\boldsymbol{x}) = (\mu_\phi(\boldsymbol{x}), \Sigma_\phi(\boldsymbol{x})). \tag{2.60}$$

In general, a diagonal covariance $\Sigma_\phi(\boldsymbol{x}) = \operatorname{diag}(\sigma^2_\phi(\boldsymbol{x}))$ is used, which is called *mean-field approximation*. For every pair (θ, ϕ) we have

$$\log p_\theta(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}[\log p_\theta(\boldsymbol{x})] = \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}\right] \tag{2.61}$$

$$= \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\left[\log \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\left[\log \frac{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}{p_\theta(\boldsymbol{z} \mid \boldsymbol{x})}\right] \tag{2.62}$$

$$= \mathcal{L}_{\theta,\phi}(\boldsymbol{x}) + D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x})). \tag{2.63}$$

As the Kullback-Leibler divergence $D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x})) \geq 0$ then the term

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}, \boldsymbol{z}) - \log q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\right] \tag{2.64}$$

is a lower bound of the *evidence* $\log p_\theta(\boldsymbol{x})$ and it is called *Evidence Lower Bound (ELBO)*.

Observe that the KL divergence $D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))$ measures two *gaps*:

- The approximation between the variational model $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and the true posterior distribution $p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$. The smaller this divergence value, the better the match between these distributions.
- The difference between the ELBO $\mathcal{L}_{\theta,\phi}(x)$ and the marginal log likelihood $\log p_\theta(\boldsymbol{x})$. The smaller this divergence value, the tighter this lower bound.

To train the variational autoencoder $(\mathsf{D}_\theta, \mathsf{E}_\phi)$, we maximize the ELBO with respect to (θ, ϕ) using the dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$:

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{\theta,\phi}(\boldsymbol{x}_i). \tag{2.65}$$

The ELBO (2.64) can also be written as

$$\begin{aligned} \mathcal{L}_{\theta,\phi}(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) + \log \frac{p_{\mathcal{Z}}(\boldsymbol{z})}{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \right] && (2.66) \\ &= \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \right] - D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_{\mathcal{Z}}(\boldsymbol{z})) && (2.67) \end{aligned}$$

which can be interpreted as the sum of a reconstruction term (as that of plain autoencoders) and a regularization term forcing the approximate posterior $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ match the prior $p_{\mathcal{Z}}(\boldsymbol{z})$.

Optimizing the ELBO using SGD: The Reparameterization Trick

For a fixed point $\boldsymbol{x}$, the computation of $\nabla_\theta \mathcal{L}_{\theta,\phi}(\boldsymbol{x})$ is straightforward:

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\theta,\phi}(\boldsymbol{x}) &\underset{(2.67)}{=} \nabla_\theta \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \right] - \nabla_\theta D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_{\mathcal{Z}}(\boldsymbol{z})) && (2.68) \\ &= \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \left[\nabla_\theta \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \right] && (2.69) \\ &\simeq \nabla_\theta \log p_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{z}}). && (2.70) \end{aligned}$$

Here, the unbiased estimator $\nabla_\theta \log p_\theta(\boldsymbol{x} \mid \tilde{\boldsymbol{z}})$ of the expectation is computed using Monte Carlo with only one sample $\tilde{\boldsymbol{z}} \sim q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ as is done in [121], but a better estimator can be computed using more samples. Also, in [31] they propose to use importance sampling using $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ as a proposal distribution which relaxes the requeriment that the variational distribution must exactly match the true posterior, and then obtain a tighter ELBO as the number of Monte Carlo samples grows.

The computation of $\nabla_\phi \mathcal{L}_{\theta,\phi}(\boldsymbol{x})$ is more challenging:

$$\nabla_\phi \mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = \nabla_\phi \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})} \left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) \right] - \nabla_\phi D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_{\mathcal{Z}}(\boldsymbol{z})). \tag{2.71}$$

We can not directly compute (or estimate) the first term because the expectation is computed with respect to $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ which depends on the parameters ϕ. The *reparameterization trick* [121] consists in expressing $\boldsymbol{z}$ as a deterministic function of $\boldsymbol{x}$, ϕ and a random variable $\epsilon \sim \mathcal{N}(0, I)$ as follows:

$$\boldsymbol{z}(\boldsymbol{x}, \phi, \epsilon) = \mu_\phi(\boldsymbol{x}) + \sigma_\phi(\boldsymbol{x}) \odot \epsilon \tag{2.72}$$

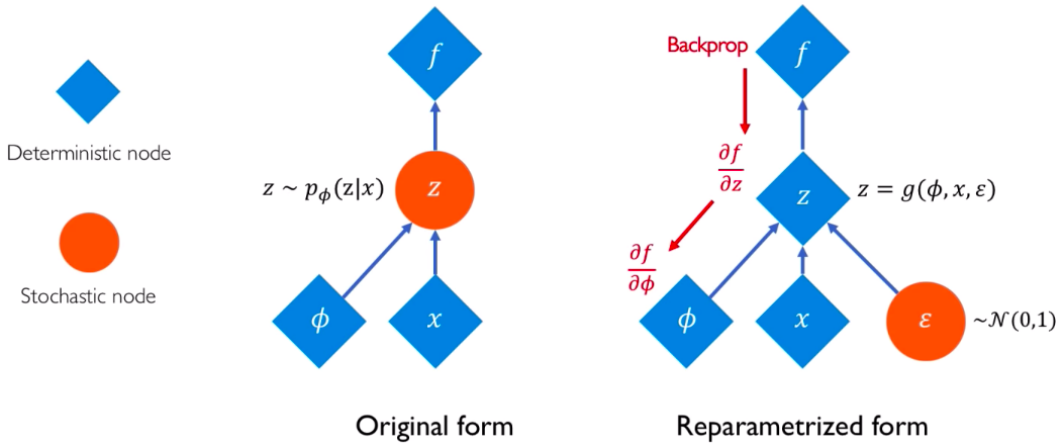


Figure 2.8: *Reparameterization trick.* The stochastic variable $\boldsymbol{z}$ depends on ϕ in the original form. By expressing $\boldsymbol{z}$ as a deterministic function of $\boldsymbol{x}$, ϕ and a random variable ϵ in the reparameterized form we can now backpropagate gradients through $\boldsymbol{z}$ to ϕ.

where $\odot$ means coordinate-wise product. Hence, defining $f(\boldsymbol{z}) = \log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})$ then

$$\nabla_\phi \mathbb{E}_{q_\phi(\boldsymbol{z} \mid \boldsymbol{x})}[f(\boldsymbol{z})] = \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(\boldsymbol{z}(\boldsymbol{x}, \phi, \epsilon))] \tag{2.73}$$

$$= \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(\boldsymbol{z}(\boldsymbol{x}, \phi, \epsilon))] \tag{2.74}$$

$$\simeq \nabla_\phi f(\boldsymbol{z}(\boldsymbol{x}, \phi, \tilde{\epsilon})) \tag{2.75}$$

again using only one Monte Carlo sample $\tilde{\epsilon} \sim \mathcal{N}(0, I)$ to estimate the expectation with respect to ϵ.

Recalling the decoder model (2.57), we have

$$\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) = \sum_{i=1}^{d} \log \mathcal{N}(\boldsymbol{x}_i;\, \mu_\theta(\boldsymbol{z})_i, \gamma^2) \tag{2.76}$$

$$= \sum_{i=1}^{d} \left[\frac{(\boldsymbol{x}_i - \mu_\theta(\boldsymbol{z})_i)^2}{2\gamma^2} + \log \gamma + \frac{1}{2} \log(2\pi) \right] \tag{2.77}$$

and the gradient in (2.75) can be easily computed by backpropagation.

For the second term of (2.71), as the variational (encoder) distribution was defined as $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z};\, \mu_\phi(\boldsymbol{x}), \mathrm{diag}(\sigma^2_\phi(\boldsymbol{x})))$ then the KL term of the ELBO (2.67) can be analitically computed without resorting to the reparameterization trick:

$$D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_{\mathcal{Z}}(\boldsymbol{z})) = D_{KL}(\mathcal{N}(\boldsymbol{z}; \mu_\phi(\boldsymbol{x}), \mathrm{diag}(\sigma^2_\phi(\boldsymbol{x}))) \,\|\, \mathcal{N}(\boldsymbol{z}; 0, I)) \tag{2.78}$$

$$= \frac{1}{2} \sum_{i=1}^{d} \left[\sigma^2_\phi(\boldsymbol{x})_i + \mu_\phi(\boldsymbol{x})_i^2 - 1 - \log(\sigma^2_\phi(\boldsymbol{x})_i) \right]. \tag{2.79}$$

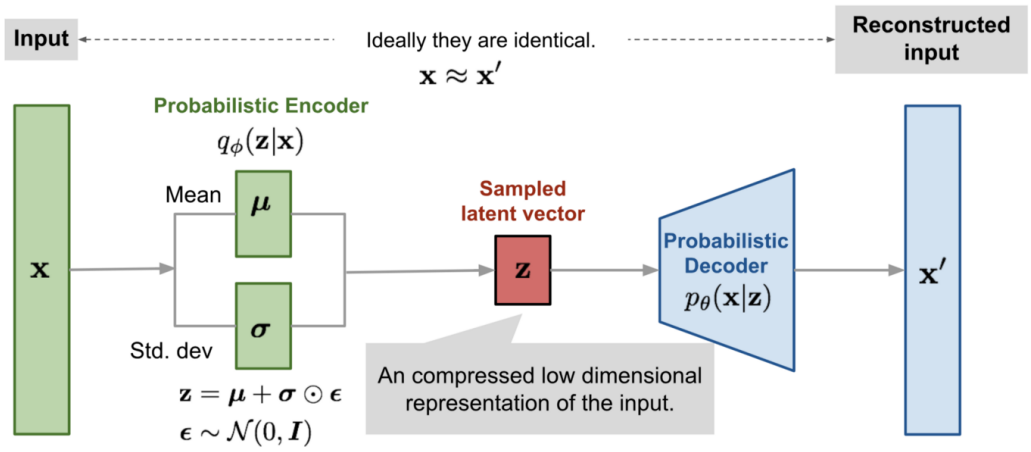


Figure 2.9: *Variational AutoEncoder model.*

Hence, the gradient with respect to ϕ can also be computed by backpropagation.

More expressive variational distributions

The minimum value that the KL divergence $D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))$ can reach depends on the flexibility of the variational distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$. When using a variational model with limited capacity, as we train both encoder and decoder at the same time, minimizing $D_{KL}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))$ can reduce the quality of the generative network $p_\theta(\boldsymbol{z} \mid \boldsymbol{x})$ to match the variational distribution.

A simple way to do so is to choose a full covariance matrix for the encoder in (2.60) using a Cholesky decomposition $\Sigma_\phi(\boldsymbol{x}) = L(\boldsymbol{x})L(\boldsymbol{x})^T$ with lower triangular matrix $L(\boldsymbol{x})$. Also, to obtain more expressive variational models and then to reduce the gap between $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and $p_\theta(\boldsymbol{z} \mid \boldsymbol{x}))$ one can reparameterize $\boldsymbol{z} = g(\boldsymbol{x}, \phi, \epsilon)$ defining g as a composition of non-linear simple functions with analytic Jacobian matrices (the so-called *Normalizing Flows*, see Section 2.3.4) [121, 122].

Are VAEs state-of-the-art generative models?

It is generally very hard to evaluate or to compare generative models on high-dimensional spaces. For example, a model may reach very high likelihood values $p_\theta(\boldsymbol{x})$ but to generate visually unpleasing random samples. On the other hand, we can construct a generative model which only selects images from a small subset of the training data (hence visually pleasant examples) but it will have a low likelihood value. Hence, the quality assessment of a generative model depends on the application at hand [216].

Figure 2.10: *State-of-the-art VAE model. Top*: Random samples from VDVAE trained on the FFHQ dataset [116]. *Bottom*: Multi-scale generation using several layers of stochastic latent variables. Figure retrieved from [44].

However, VAEs with impressive generation capabilities have recently been built. We have reviewed VAE models with independent latent variables $\boldsymbol{z} \sim \mathcal{N}(0, I)$ but state-of-the-art models are generally constructed using several layers of stochastic latent variables, each dependent on the previous one. For example, we can generate a low-resolution image from some latent variables $\boldsymbol{z}_0$, upsample the result (for example using bilinear interpolation) and generate more realistic finer details using additional latent variables $\boldsymbol{z}_1$, conditioning the generation on the previous image, and so on. Moreover, in the *top-down* VAE model [205] both the decoder and encoder generate latent variables in the same order:

$$p_\theta\left(\boldsymbol{x} \mid \boldsymbol{z}\right) = p_\theta\left(\boldsymbol{z}_0\right) p_\theta\left(\boldsymbol{z}_1 \mid \boldsymbol{z}_0\right) \ldots p_\theta\left(\boldsymbol{z}_N \mid \boldsymbol{z}_{<N}\right) p_\theta\left(\boldsymbol{x} \mid \boldsymbol{z}_N\right) \tag{2.80}$$

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = q_\phi(\boldsymbol{z}_0 \mid \boldsymbol{x}) q_\phi(\boldsymbol{z}_1 \mid \boldsymbol{z}_0, \boldsymbol{x}) \ldots q_\phi(\boldsymbol{z}_N \mid \boldsymbol{z}_{<N}, \boldsymbol{x}) \tag{2.81}$$

where $\boldsymbol{z}_{<i} = (\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{i-1})$. Examples of this type of deep latent variable models are NVAE [225] and VDVAE [44] (see Figure 2.10 for random samples generated with this last model).

2.3.3 Generative Adversarial Networks

Another widely used generative model is the so-called *Generative Adversarial Networks (GAN)* model [88]. It consists of a pair of neural networks which are trained in a competitive manner: a *generator network* $G\colon \mathbb{R}^p \to \mathbb{R}^d$ maps (latent) noise vectors $\boldsymbol{z} \sim \mathcal{N}(0, I)$ to images $G(\boldsymbol{z})$ and the *discriminator network* $D\colon \mathbb{R}^d \to [0, 1]$ tries to distinguish between real images $D(\boldsymbol{x}) = 1$ from the dataset $\boldsymbol{x} \in \mathcal{D}$ and *fake* images $D(G(\boldsymbol{z})) = 0$ generated by G. This is done by optimizing the following loss function:

$$\min_G \max_D \left\{ \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{X}}(\boldsymbol{x})} \log D(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z}\sim p_{\mathcal{Z}}(\boldsymbol{z})} \log(1 - D(G(\boldsymbol{z}))) \right\}. \tag{2.82}$$

Although GAN models can generate images of high visual quality, in general they are hard to train [195, 189] and present some problems to correctly capture the density of the data, eg. *mode dropping* [9, 189] which means that some of the modes of the data distribution are not covered by the generator network. Moreover, they do not provide an efficient way to retrieve the latent code $\boldsymbol{z}$ for a given image $\boldsymbol{x}$ like the encoder does in the VAE setting.

2.3.4 Normalizing Flows

Finally, if we choose $p = d$ (latent vector and image dimensions are equal) one can train an invertible network known as *Normalizing Flow*. [179, 165, 123]. It maps images $\boldsymbol{x}$ to a latent representation $\boldsymbol{z} = T(\boldsymbol{x})$ (usually normally distributed) and is constructed composing a chain of simpler transformations which all have triangular Jacobians. If we choose $p_{\mathcal{X}}(\boldsymbol{x})$ to be the push-forward measure

$$p_{\mathcal{X}}(\boldsymbol{x}) = p_{\mathcal{Z}}(T(\boldsymbol{x})) \left|\det J_T(\boldsymbol{x})\right| \tag{2.83}$$

then the term $\left|\det J_T(\boldsymbol{x})\right|$ can be efficiently computed. Despite the fact that here we have an explicit expression for the prior distribution $p_{\mathcal{X}}(\boldsymbol{x})$, some works suggest that to compute MAP estimators directly in image space $\mathcal{X}$ is empirically more challenging that in latent space $\mathcal{Z}$ [11]. Observe that, as in this case the relationship between $\boldsymbol{x}$ and $\boldsymbol{z}$ is deterministic, the conditional distribution $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x}) = \delta(\boldsymbol{z} - T(\boldsymbol{x}))$ is degenerated and then it is not straightforward to apply the sampling algorithms proposed in this paper. We leave this adaptation to future work.

Chapter 3

Plug and Play methods for Inverse Problems in Imaging

This chapter deals with two particular inverse problems, namely Joint Denoising and Decompression (JDD) and Multi-Image Super-Resolution (MISR). In the first case, after an extensive study of the noise statistics in the transformed wavelet domain, we derive two novel algorithms to solve this particular inverse problem. One of them is based on a multi-scale self-similarity prior and can be seen as a transform-domain generalization of the celebrated Non-Local Bayes (NLB) algorithm to the case of non-Gaussian noise. The second one uses a neural-network denoiser to implicitly encode the image prior, and an ADMM scheme to incorporate this prior into an optimization algorithm to find a MAP-like estimator. For the

MISR problem, we derive a related algorithm but using a Chambolle-Pock alternate minimization scheme which also leads to a Plug-and-Play algorithm.

3.1 Plug and Play methods

Denoising is one of the most studied inverse problems in imaging, and several methods have been developed, in particular under a white Gaussian noise assumption [30, 131, 244, 243]. We denote by $\mathcal{G}(\tilde{u}, \sigma^2)$ a denoising method $\mathcal{G}$ applied to a noisy image $\tilde{u}$ assuming noise variance σ^2. To show impressive performance, a denoiser has to model or implicitly learn something about the data manifold it was trained on. For example, one must obtain

$$\mathcal{G}(u + n, \sigma^2) \simeq u, \quad \forall n \sim \mathcal{N}(0, \sigma^2) \tag{3.1}$$

at least for u in the set of images of interest and hence the denoiser $\mathcal{G}$ acts as a projector maping $u + n$ onto the data manifold.

Recently, some effort has been put on leveraging the implicit information captured in state-of-the-art denoising methods to regularize several inverse problems. This is the so-called *Plug-and-Play approach*. In this section, we summarize the main ideas behind this approach and some of the denoising methods used in the rest of the chapter.

3.1.1 Denoising algorithms

Non-Local Bayes (NLB) denoising algorithm

The Non-Local Bayes algorithm [131] allows to recover a clean image u from noisy measurements $u_n = u + n$ in the case where n is a white Gaussian noise of 0 mean and variance σ^2. The algorithm works with patches $p_x(u)$ of size $p \times p$ centered at pixel x and extracted from an image u. Noisy patches will be denoted by $\tilde{P}_x := p_x(u_n)$ whereas clean patches will be denoted by $P_x := p_x(u)$. For the moment, we concentrate on a fixed patch P_x and then we drop the index x for the sake of clarity (we refer to the patch as $P_x = P$). We can estimate the clean patch $\hat{P}$ using a *maximum a posteriori* (MAP) estimation and the Bayes' rule:

$$\begin{aligned} \hat{P} &= \arg\max_P \mathbb{P}(P|\tilde{P}) \\ &= \arg\max_P \mathbb{P}(\tilde{P}|P)\mathbb{P}(P). \end{aligned}$$

The conditional distribution is known for a white Gaussian noise to be

$$-\log \mathbb{P}\left[\ \tilde{P}\ \middle|\ P\ \right] = \frac{\|\tilde{P} - P\|^2}{2\sigma^2} + C_1. \tag{3.2}$$

As a prior model for P we assume that it follows a multivariate Gaussian model with mean μ_P and covariance matrix Σ_P to be estimated, *i.e.*

$$-\log \mathbb{P}[P] = \frac{1}{2}(P - \mu_P)^T \Sigma_P^{-1}(P - \mu_P) + C_2. \tag{3.3}$$

From the three previous equations and considering that the normalization constants C_1 and C_2 do not depend on P we see that our MAP estimation reduces in this case to a simple quadratic minimization problem

$$\hat{P} = \arg\min_P \frac{\|\tilde{P} - P\|^2}{\sigma^2} + (P - \mu_P)^T . \Sigma_P^{-1} (P - \mu_P) \tag{3.4}$$

Since we do not have an expression for the covariance matrix Σ_P because we do not have the real patches P, we build an empirical covariance matrix $\Sigma_{\tilde{P}}$ using noisy patches $p_y(u_n)$ that are similar to $\tilde{P}$ (we obtain also the empirical mean $\mu_{\tilde{P}}$). This empirical covariance matrix is a noisy estimate of the ideal one, which satisfies

$$\mathbb{E}\left[\Sigma_{\tilde{P}}\right] = \Sigma_P + \sigma^2 I.$$

Therefore we substitute the covariance matrix in (3.4) by its unbiased estimate, i.e.

$$\hat{P} = \arg\min_P \frac{\|\tilde{P} - P\|^2}{\sigma^2} + (P - \mu_{\tilde{P}})^T (\Sigma_{\tilde{P}} - \sigma^2 I)^{-1} (P - \mu_{\tilde{P}}) \tag{3.5}$$

Differentiating and equating to zero yields the solution:

$$\hat{P} = \mu_{\tilde{P}} + (\Sigma_{\tilde{P}} - \sigma^2 I) \Sigma_{\tilde{P}}^{-1} (\tilde{P} - \mu_{\tilde{P}}) \tag{3.6}$$

Still in [131], the authors propose a second step to improve the results: from the restored patches, we compute a new version of the covariance matrix, that we note $\Sigma_{\hat{P}}$ ($\mu_{\hat{P}}$ the mean obtained from those unbiased examples). This matrix is obtained from denoised patches and then, we can consider that it is not affected by noise. Taking this into consideration, the second step can be expressed as follows:

$$P^{final} = \mu_{\hat{P}} + \Sigma_{\hat{P}} (\Sigma_{\hat{P}} + \sigma^2 I)^{-1} (\tilde{P} - \mu_{\hat{P}}). \tag{3.7}$$

The minimization problem stated in the previous section is based on patches. That means that the solution for each minimisation problem is a patch. Since each pixel value of u belongs to several patches, we have at our disposal several estimators of the same pixel to be aggregated. In the classic NLBayes or NLMeans algorithm, this aggregation is done by taking the mean value of all patches:

$$\hat{u}(x) = \frac{1}{p^2} \sum_i \hat{P}_{x_i}(x - x_i). \tag{3.8}$$

Denoising with Residual Neural Networks

In recent years, classical denoising methods as the celebrated Non-Local Bayes algorithm have been surpased by deep learning approaches. In particular, the so-called *Residual Networks* [100] are trained end-to-end on pairs $(\tilde{u}, u)$ of noisy (input) and target (output) images, to compute only de noise present in the image, so that

$$\mathcal{G}(u + n, \sigma^2) = n. \tag{3.9}$$

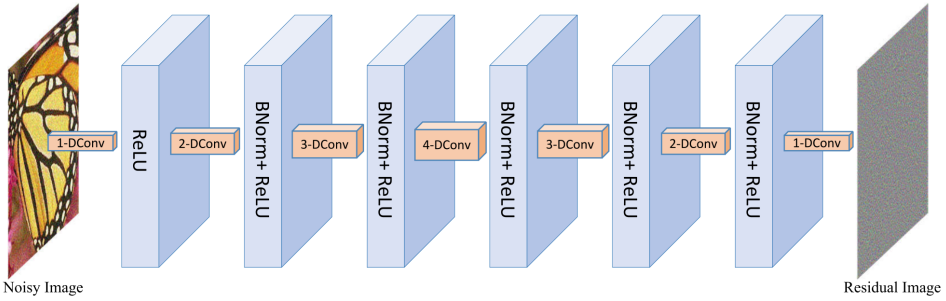


Figure 3.1: *DnCNN architecture.* Figure retrieved from [245].

This particular networks showed improved performance with respect to classical approaches. A widely used model is the *Denoising Convolutional Neural Network* [244, 245] which consists of a cascade of convolutional layers, Batch Normalization and ReLU's (see Figure 3.1).

3.1.2 Denoising as proximal operator

Recall the MAP formulation of a denoising problem:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 + R(\boldsymbol{x}) \right\} \tag{3.10}$$

where $R(\boldsymbol{x}) \propto -\log p_{\mathcal{X}}(\boldsymbol{x})$ is the regularization term given by the prior over $\boldsymbol{x}$. The right hand side is the *proximal operator* of the function $\sigma^2 R(\boldsymbol{x})$ [186]. This operator can be written as before as

$$\hat{\boldsymbol{x}} = \mathcal{G}(\boldsymbol{y}, \sigma^2). \tag{3.11}$$

Alternatively, given *any* denoising operator $\mathcal{G}(\boldsymbol{y}, \sigma^2)$ we can see it as computing the solution of (3.10) for some *implicit prior* $p_{\mathcal{X}}(\boldsymbol{x})$, even those denoisers based on neural networks.

The *Plug-and-Play* approach first appears in [227] using an ADMM optimization scheme [25] but here we present the main idea on simpler form. First observe that a generic inverse problem can be rewritten as a constrained one as

$$\arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2 + R(\boldsymbol{x}) \right\} \tag{3.12}$$

$$= \arg\min_{\boldsymbol{x},\boldsymbol{u}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2 + R(\boldsymbol{u}) \right\} \quad \text{s.a. } \boldsymbol{x}=\boldsymbol{u} \tag{3.13}$$

which in turn can be relaxed as

$$\arg\min_{\boldsymbol{x},\boldsymbol{u}} \left\{ \frac{1}{2\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2 + \frac{\beta}{2}\|\boldsymbol{x}-\boldsymbol{u}\|^2 + R(\boldsymbol{u}) \right\} \tag{3.14}$$

for $\beta \to +\infty$. This procedure is known as *Half-Quadratic Splitting (HQS)* [245]. A direct alternating minimization scheme leads to

$$\begin{cases} \boldsymbol{x}^{(i+1)} = \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\sigma^2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 + \frac{\beta}{2} \|\boldsymbol{x} - \boldsymbol{u}^{(i)}\|^2 \right\} \\ \boldsymbol{u}^{(i+1)} = \arg\min_{\boldsymbol{u}} \left\{ \frac{\beta}{2} \|\boldsymbol{x}^{(i+1)} - \boldsymbol{u}\|^2 + R(\boldsymbol{u}) \right\}. \end{cases} \tag{3.15}$$

Hence, we have *decoupled* the prior/regularization term from the likelihood/data fit term. The first subproblem is quadratic in $\boldsymbol{x}$ and thus its solution can be computed in closed form. The second one is similar to (3.10) so it can be interpreted as a denoising step:

$$\boldsymbol{u}^{(i+1)} = \mathcal{G}(\boldsymbol{x}^{(i+1)}, 1/\beta) \tag{3.16}$$

which can be computed using any state-of-the-art denoising method.

Using this procedure we can apply our favorite denoiser to (implicitly) regularize any inverse problem. This way we can *modularize* the algorithm and easily modify the inverse problem or the prior only changing the corresponding subproblem.

3.2 Joint Denoising and Decompression (JDD)

Image compression through transform coding consists of applying a linear invertible transform that sparsifies the data (like block-wise Discrete Cosine Transform for JPEG compression or a Wavelet Transform for JPEG2000) followed by quantization of the transformed coefficients, which are finally compressed by a lossless encoder. This family of compression schemes may achieve very high compression ratios but may lose some details in the quantization step. This lossy quantization is also responsible for well-known artifacts that may appear in the compressed image in the form of texture loss or Gibbs effects near edges. Many solutions have been proposed in the literature to remove some of these artifacts. Most of them are variational and involve the minimization of the total variation (to minimize ringing) over all images that would lead to the observed quantized image [67, 4, 236].

Surprisingly, little attention has been paid in previous works to the fact that the image to be compressed may contain noise, and that noise may interact in subtle ways with the compressor, producing new kinds of artifacts that we call *outliers* (see Figure 3.7). These artifacts cannot be removed by the previously cited works, which only aim at removing compression artifacts but not noise or its complex interactions with the compressor. However, such artifacts are particularly annoying in the case of wavelet-based compressors like JPEG2000 and the CCSDS recommendation [206], which are extensively used to compress digital cinema and high-resolution remote sensing images.

3.2.1 Lossy compression by quantization

Lossy compression is based on an irreversible process called *quantization*. We can model a generic quantizer with a function that maps the set of values that a coefficient can take (in most cases, the real line $\mathbb{R}$), in a *finite* set of values:

$$Q\colon \mathbb{R} \to C_Q, \qquad C_Q = \{c_1, c_2, \ldots, c_m\}. \tag{3.17}$$

In general, the quantizer maps a whole interval into a unique integer value that represents this interval (see Figure 3.2). Althrough in practice one always has a *floating-point* representation of the coefficients that is already quantized (that is, we do not really have continuous but discrete numerical values), these values need 32 or 64 bits to be represented but one usually wants to represent them with fewer bits. For example, the *mid-rise quantizer* that maps every interval $[k\Delta, (k+1)\Delta)$ to its midpoint (for $k \in \mathbb{Z}$) can be defined as

$$Q(x) = \Delta\left(\left\lfloor \frac{x}{\Delta} \right\rfloor + \frac{1}{2}\right) = \frac{k+1}{2}\Delta \quad \text{if } x \in [k\Delta, (k+1)\Delta), \tag{3.18}$$

where $\lfloor\cdot\rfloor$ stands for the classical *floor* operator (that is, $\lfloor x \rfloor$ is the largest integer less than or equal to x). This quantizer is uniform (that is, the quantization intervals all have size Δ) but also non-uniform quantizers can be used (see Figure 3.2).

The quantization is an irreversible process because it maps different values x to the same value $Q(x)$, so given $Q(x) = \tilde{x}$ is not possible to recover x exactly but only to know $x \in [\tilde{x} - \frac{\Delta}{2}, \tilde{x} + \frac{\Delta}{2})$ for the quantizer of (3.18). For this reason, the quantization is a lossy compression scheme. Ideally, we want to discard only the less relevant information of the signal. For example, in an acoustic signal, there are frequencies less audible to the human ear that we do not want to keep, or in an image, small variations in homogeneous areas are less important that sharp edges. Also, if we can do this using only few possible values c_i, and set most of the signal coefficients to the same value (for example, c_0) then we can get high compression ratios by means of a lossless compression scheme such as Huffman coding [108].

Most commonly used image compression algorithms implement pipelines similar to the one shown in Figure 3.3. The best known image compression scheme is the *JPEG* format[1]. JPEG uses a lossy form of compression based on the *Discrete Cosine Transform (DCT)*, which is related to the Fourier transform. This scheme converts each 8×8 patch of the image from the spatial domain into the frequency domain, and there a perceptual model based on the human visual system discards high-frequency information. Most high-frequency coefficients contribute less to the overall picture than other coefficients (such as those representing sharp edges

[1] *JPEG* stands for Joint Photographic Experts Group, the name of the committee that created the JPEG standard and also other still picture coding standards. https://jpeg.org/

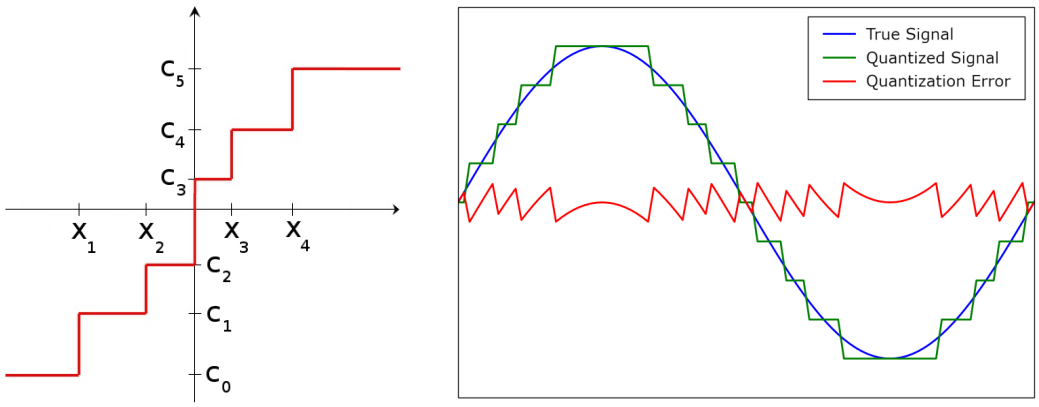


Figure 3.2: *Lossy compression by signal quantization. Left*: Graph of a generic non-uniform quantizer of the form (3.17). *Right*: example of a quantized signal and their corresponding coefficients and quantization errors.

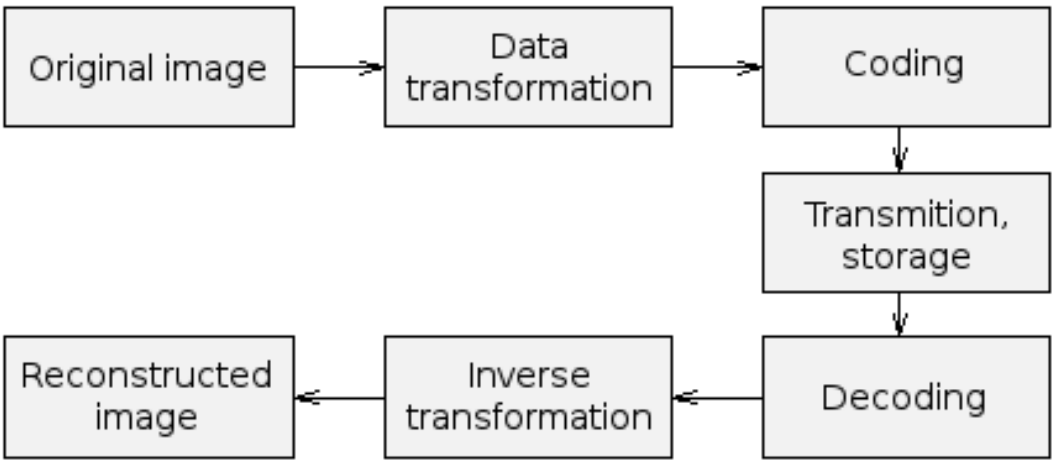


Figure 3.3: Compression/Decompression pipeline.

that are distributed across all frequencies), thus they have generally the smallest values that are quantized to zero, achieving high compression ratios. *JPEG2000* is another standard of image compression based on the wavelet transform. It is similar to the above JPEG standard as it transforms the whole image using a wavelet transform and quantizes the small coefficients to zero, in general in the high-frequency subbands. We will describe this compression scheme as well as the closely related *CCSDS recommended standard* [207] in more detail in Section 3.2.3. For a complete treatment of compression schemes, see [196], [197].

3.2.2 Wavelet transform

The search for basis functions for *analyzing* a function f, that is, to decompose f as a combination of basis elements that describes the behavior of the funtion, goes back at least as far as Jean-Baptiste Joseph Fourier (1768-1830) who used complex sinusoids [172]. The Fourier transform of a continuous time signal $f \in L^2(\mathbb{R})$ can be defined by

$$\mathcal{F}(f)(\omega) = \hat{f}(\omega) = \langle f, e^{i\omega\cdot} \rangle = \int_{\mathbb{R}} f(t) e^{-i\omega t}\, dt. \tag{3.19}$$

A difficulty that has often been pointed out with this approach is that, because of the infinite extent of the basis function, any time-local information (e.g., an abrupt change in the signal) is spread out over the whole frequency axis. For that reason, the above transform (3.19) is not adequate for frequency analysis localized in time. This is like knowing which notes are present in a song, but not at what moments they sound.

On the other hand, it is possible to construct *short waves* or *wavelets* $\psi \in L^2(\mathbb{R})$ so that the set

$$W = \{\psi_{j,k}\}_{j,k\in\mathbb{Z}}, \qquad \psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k) \tag{3.20}$$

is orthonormal, that is

$$\langle \psi_{j,k}, \psi_{j',k'} \rangle = \delta_{jj'}\delta_{kk'} \tag{3.21}$$

where δ_{ij} is the Kronecker delta. A classic example is the *Haar basis* generated by:

$$\psi(t) = \begin{cases} 1 & \text{if } 0 \le t < 1/2 \\ -1 & \text{if } 1/2 \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

(see Figure 3.4). The orthonormality is easily verified, and the fact that it is a basis of $L^2(\mathbb{R})$ is proved in [169]. Haar [95] used these functions to give an example of an orthonormal system for the space of square-integrable functions on the unit interval $[0, 1]$, and it is now recognised as the first known wavelet basis. However, the Haar function is not continuous, and this is not generally appropriate for signal processing.

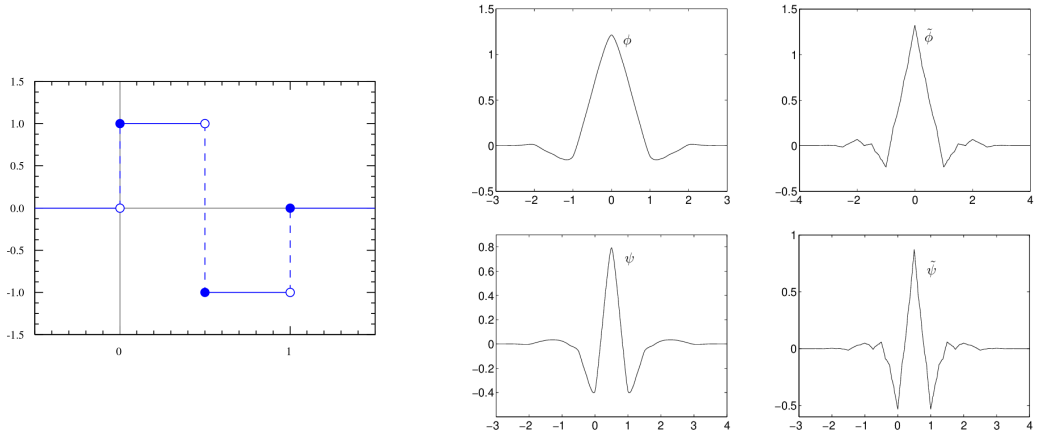


Figure 3.4: *Wavelets. Left*: Haar (orthogonal) wavelet defined by (3.22). *Right*: CDF 9/7 (biorthogonal) scaling funtion ϕ, wavelet ψ and corresponding dual functions $\tilde{\phi}, \tilde{\psi}$.

Orthonormal and biorthogonal wavelet basis

It can be shown [145] that we can construct other wavelets with more regularity (that is, continuity, smoothness, etc) than the Haar wavelet. This is done by the concept of *Multiresolution Analysis (MRA)* due to Mallat [146] and Meyer [154], that basically consists in a nested set of subspaces $\{V_j, W_j\}_{j\in\mathbb{Z}}$ of $L^2(\mathbb{R})$ such that

$$V_j \subset V_{j-1} \quad \forall j \in \mathbb{Z} \tag{3.23}$$

$$V_j \oplus W_j = V_{j-1} \quad \forall j \in \mathbb{Z} \tag{3.24}$$

$$\bigcup_j V_j \text{ is dense in } L^2(\mathbb{R}), \quad \bigcap_j V_j = \{0\} \tag{3.25}$$

and there exists functions ϕ and ψ such that $\{\phi_{j,k}\}_{k\in\mathbb{Z}}$ and $\{\psi_{j,k}\}_{k\in\mathbb{Z}}$ are basis of V_j and W_j respectively, where $\phi_{j,k}$ and $\psi_{j,k}$ are defined as in (3.20). Here, ψ is the wavelet and ϕ is called the *scaling function*. A function $f \in L^2(\mathbb{R})$ can be projected onto an approximation space V_j and then we can add details (which consist in the projection of f onto W_j) to obtain a higher resolution version in V_{j-1}.

There are two important cases to be distinguished:

- $V_j \perp W_j$ and $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ is an *orthonormal* basis of $L^2(\mathbb{R})$, thus

$$\langle \psi_{j,k}, \psi_{j,l} \rangle = \delta_{kl} \qquad \forall j,k,l \in \mathbb{Z} \tag{3.26}$$

$$f = \sum_{(j,k)\in\mathbb{Z}^2} \langle f, \psi_{j,k} \rangle \psi_{j,k} \qquad \forall f \in L^2(\mathbb{R}) \tag{3.27}$$

- There exists a *dual wavelet* $\tilde{\psi}$ such that $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ and $\{\tilde{\psi}_{j,k}\}_{j,k\in\mathbb{Z}}$ are *biorthog-*

onal Riesz basis of $L^2(\mathbb{R})$, thus

$$\langle \psi_{j,k}, \tilde{\psi}_{j,l} \rangle = \delta_{kl} \qquad \forall j,k,l \in \mathbb{Z} \tag{3.28}$$

$$f = \sum_{(j,k)\in\mathbb{Z}^2} \langle f, \psi_{j,k} \rangle \tilde{\psi}_{j,k} = \sum_{(j,k)\in\mathbb{Z}^2} \langle f, \tilde{\psi}_{j,k} \rangle \psi_{j,k} \qquad \forall f \in L^2(\mathbb{R}) \tag{3.29}$$

We are mainly interested in processing finite-length signals such as images. Also, we want to represent these signals using no more wavelet coefficients than the signal length. Otherwise, we are increasing the amount of coefficients needed to represent the same signal, which is obviously undesirable on a compression task, and this can be achieved analyzing a periodization or symmetrization of the original signal. It can be shown [224] that in this case, orthogonal wavelet filters cause border artifacts because they are non-symmetric (the only symmetric orthogonal wavelet basis is the Haar basis [54]), but we can still obtain biorthogonal wavelet basis with this desirable property. This is why biorthogonal basis are preferable to orthogonal basis when processing images, as we see below.

In the orthogonal case, one can construct discrete filters h and g related to the basis functions ϕ and ψ of the MRA such that the decompositions and reconstructions between two resolutions j and $j-1$ are performed by convolutions and downsampling/upsampling with these filters [145]. In the biorthogonal case, we have filters $\tilde{h}$ and $\tilde{g}$ related to the dual functions $\tilde{\phi}$ and $\tilde{\psi}$ that replace h and g in the reconstruction. The *Cohen-Daubechies-Feauveau* (CDF) wavelets [47] are historically the first family of biorthogonal wavelets, and the CDF 9/7 is experimentally the better suited for image analysis [230]. Although this is not an orthogonal wavelet, it is *almost orthogonal* in the sense that is almost energy preserving, so for practical purposes it can be considered as orthogonal [3].

Finally, we can decompose a signal by applying this process several times to the samples of the signal. A *wavelet decomposition* of a function $f \in V_L$, which is caracterized by the coefficients $a_L = \{\langle f, \phi_{L,k} \rangle\}_{k\in\mathbb{Z}}$, is composed of detail wavelet coefficients $d_j = \{\langle f, \psi_{j,k} \rangle\}_{k\in\mathbb{Z}}$ of f at scales $L < j \leq J$ plus the remaining approximation at the largest scale J:

$$[a_J, d_J, d_{J-1}, \ldots, d_{L-1}] \tag{3.30}$$

(recall that the resolution increases as the scale parameter j decreases). In Figure 3.5 we show an example of multi-level decomposition of a signal using the decomposition (3.30) for several values of J. For images, we can apply the decomposition scheme of (3.30) presented above in a separable fashion: first we apply a one-level decomposition to each row of the image and then in the columns, obtaining a *1-level wavelet decomposition of the image*. This result consists in four *subbands* that we call *summary* (LL) and *detail* (LH, HL, HH) subbands. If we repeat the decomposition to the LL subband, we get a *multi-level wavelet decomposition* (see Figure 3.6).

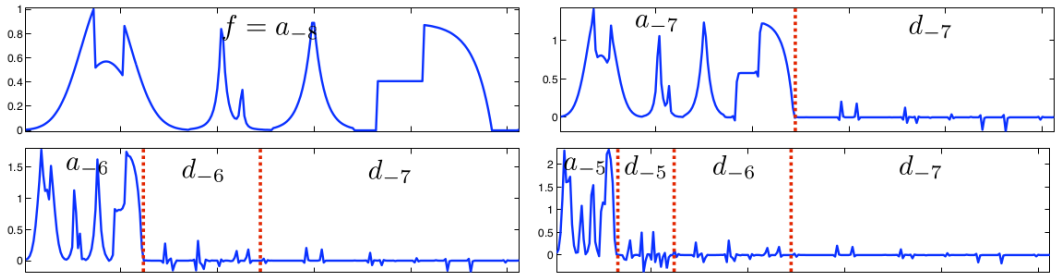


Figure 3.5: Decomposition of a signal in multiple approximation levels. The a_j and d_j are the coefficients of the expansion of f in the basis of V_j and W_j respectively. Figure retrieved from [168].

3.2.3 Problem statement

CCSDS Recommendation and JPEG2000 format

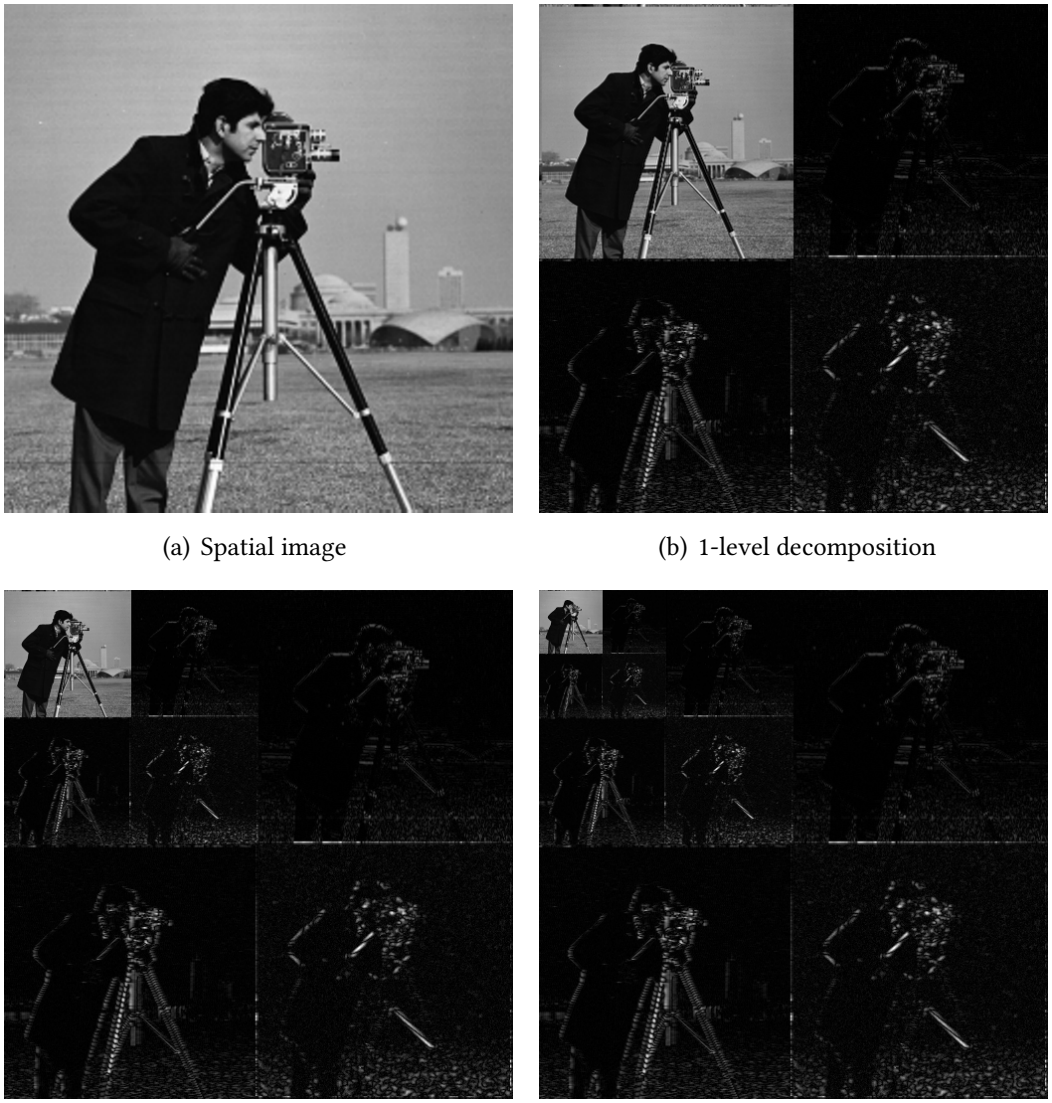
The *Consultative Committe for Space Data Systems (CCSDS)* has issued various recommended standards for image data compression [207] which were implemented and used in more than 1000 space missions[2]. Basically, the CCSDS standard compression scheme, as well as the JPEG2000 format, are based on the quantization of wavelets coefficients. A wavelet transform, such as the biorthogonal CDF 9/7 described in Section 3.2.2, expresses the image on another basis with the same number of coefficients as before, but using *floating point* values (requiring 32 or 64 bits each) instead of *integer* values (as is common in spatial images of 8 or 12 bits gray values). Thus, storing the wavelet coefficients of the image as float values will lead to an enlargement of the original storage space of the image, which is obviously undesired, as we want to *compress* the data. This problem is called *Dynamic Range Expansion* in [207] (Green book).

A possible solution to truncate high precision coefficients and to represent them with fewer bits is the use of the so-called *quantization*. Let u be an image of size d that we want to capture. When we use one of the schemes mentioned above, the compressed image we receive is usually

$$u_{qn} = W^{-1} \underbrace{Q(W(u+n))}_{w_{qn}} \tag{3.31}$$

where n is the image noise, W is a wavelet (invertible) transform and $Q = (Q_k)_k$ is the quantizer that gives, for each number w in the range of possible kth-wavelet coefficients, a reference coefficient $Q_k(w)$ of the interval $[a, b]$ in which w lies. For example, in the CCSDS standard, each wavelet coefficient $w_n(k)$ is quantized by

[2] https://public.ccsds.org/



(a) Spatial image

(b) 1-level decomposition

(c) 2-level decomposition

(d) 3-level decomposition

Figure 3.6: Multi-level wavelet decomposition of an image (detail coefficients have been enhanced for better visualization).

setting to 0 its $m(k)$ least significant bits:

$$Q(w_n(k)) := \text{sign}(w_n(k)) \left\lfloor \frac{|w_n(k)|}{2^{m(k)}} \right\rfloor 2^{m(k)} \tag{3.32}$$

to null most of the high frequency coefficients and also to deal with the dynamic range expansion problem.

Noise model on the wavelet domain

As usual, we assume that our image u is corrupted by additive white Gaussian noise $n \sim N(0, \sigma^2 I)$. Even though sensors usually produce a mixture of additive and multiplicative noise [1, Chapter 2], a variance stabilizing transform is usually applied before compression, making our noise model a valid approximation. The first step of the CCSDS and JPEG2000 compressors apply a wavelet transform W to the noisy image. Hence the corresponding wavelet coefficients are corrupted by Gaussian noise

$$n_w := \underbrace{W(u+n)}_{w_n} - \underbrace{W(u)}_{w} = Wn \sim N(0, \sigma^2 WW^T) \approx \mathcal{N}(0, \sigma^2 I). \tag{3.33}$$

If the wavelet transform were orthogonal, then $WW^T = I$ and n_w would be also Gaussian white noise. Most compression algorithms use, however, the CDF 9/7 *biorthogonal* wavelet transform, but even in that case, as we mentioned before, $WW^T \simeq I$ is a good approximation [3].

Bit allocation

The number of bits $m(k)$ allocated to each coefficient $w(k)$ in equation (3.32) are chosen by the compression algorithm to optimize the rate/distortion trade-off, and can be recovered from the compressed image. From these values we can recover the quantization intervals $Q^{-1}(w_n(k)) = [a_k, b_k]$ of length $q(k) = 2^{m(k)}$ except for the case $Q(w_n(k)) = 0$ where the quantization interval is of length $q(k) = 2^{m(k)+1}$. For a compressed image, the *bit rate* achieved by the compressor, measured in *bits/pixel (bpp)*, is defined as the number of bits used in the compressed representation of the image divided by the number of pixels in the image. Thus, a compression of 0.5 bpp means that we can store an image of d pixels in a compressed image using $d/2$ bits overall (in contrast with $8d$ bits needed to store the gray-level image in an 8-bit representation).

The only issue to be resolved is the number of bits to be assigned to each individual coefficient to give the best performance. We can think that we have a bit budget for the whole image (given by the compression bit rate) that we must

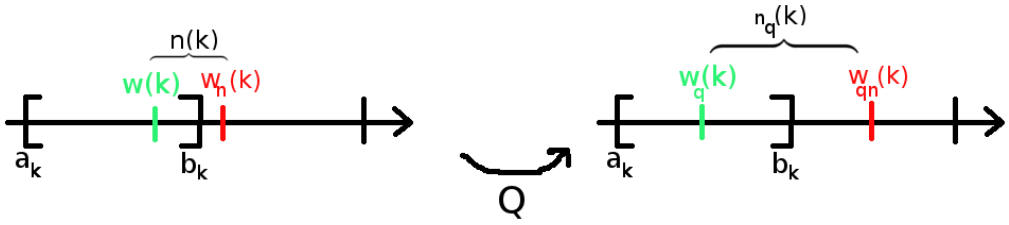


Figure 3.7: *Wavelet outliers.* When the noise $n(k)$ added to a coefficient $w(k)$ make it belongs to another quantization interval of the original one (left), the noise may be amplified (right) and we see a wavelet shaped artifact on the spatial domain.

distribute over all the coefficients of the wavelet domain. This is refered as *bit allocation* in the literature, and each compression scheme implements its own allocation procedure. We assume that the quantization intervals $Q^{-1}(w_n(k)) = [a_k, b_k]$ corresponding to each coefficient $w_n(k)$ are provided.

Artifacts introduced by quantization of wavelet coefficients

High compression ratios can be achieved by setting a large number of the smallest wavelet coefficients to zero. However, the coefficients erroneously treated by the compressor causes basically two types of artifacts:

1. Small coefficients which correspond to edges and other details (microtextures) in the image that are difficult to distinguish from the noise, and generate pseudo-Gibbs oscillations and erased microtextures.

2. Coefficients which are highly contaminated by noise (*outliers*) which modify the correct interval of quantization and generate an artifact with the shape of a wavelet basis function ψ.

As in the truncation of the Fourier series, in our case, some coefficients are needed to correctly represent the sharp edges and setting them to zero causes artifacts related to the Gibbs phenomena. To understand the artifacts caused by the second problem, consider the situation shown in Figure 3.7. If we add noise to a wavelet coefficient

$$w_n(k) = w(k) + n(k)$$

and this noise causes $w_n(k)$ to move to a different quantization interval than $w(k)$, we call $w_n(k)$ an *outlier*. The result is that the quantization of these two coefficients will be different:

$$w_{qn}(k) = Q(w_n(k)) = Q(w(k)) + n_q(k) = w_q(k) + n_q(k)$$

where $w_q(k)$ is the quantization of the original (noiseless) coefficient $w(k)$ and $n_q(k)$ is the noise introduced by the outlier. The effect that this error causes when we apply the inverse wavelet transform is

$$W^{-1}(w_{qn}(k)) = W^{-1}(w_q(k)) + W^{-1}(n_q(k)).$$

If there was no noise we would obtain the target quantized coefficients $w_q = Q(w)$. When the noise level $\sigma \ll q(k)$ is relatively small, the noisy quantized coefficient $w_{qn} := Q(w + n)$ is most often equal to w_q, and hence the quantizer has a denoising effect. However, occasionally the noise may be large enough to change the quantization interval. In that case quantization may amplify the noise

$$|n_q(k)| = |w_{qn}(k) - w_q(k)| > |w_n(k) - w(k)| = |n(k)| \tag{3.34}$$

(see Figure 3.7) and we get a visible (wavelet ψ shaped) artifact that we call an *outlier*. Outliers are particularly annoying when they are isolated. In Figure 3.8 we show an example of this type of artifacts. When the noise level $\sigma \gtrsim q(k)$ is similar to or larger than the quantization level then outliers occur everywhere and they appear indistinguishable from white noise.

Recall that the quantization is done in the wavelet domain, so the errors introduced by the quantization process are (almost) decorrelated in wavelet domain but highly correlated in spatial domain. Therefore it is more natural to work in wavelet domain because it is there where we can formulate an adequate noise model for subsequent image denoising and decompression. The remainder of this section is organized as follows. In Subsection 3.2.4 we define a carefully designed likelihood function that takes into account the degradation process above, which will appear as a data fit term in a MAP estimate. In section 3.2.5 we propose a patch-based approach to estimate the original image u from its noisy, quantized observations, inspired in the NLB algorithm presented in Subsection 3.1.1. Finally, in Subsection 3.2.6 we propose a Plug and Play approach using state-of-the-art denoising algorithms based on convolutional neural networks.

3.2.4 Likelihood function

Let $w = Wu$ be the coefficients of the original (unknown) image, and $w_{qn} = Q(w_n)$ the quantized wavelet coefficients of the noisy image. As stated before, the quantization intervals of each of these coefficients can be retrieved as $[a_k, b_k] = Q^{-1}(w_{qn}(k))$. Using this notation, and given that the noise model in the wavelet domain is $N(0, \sigma^2 I)$ (equation (3.33)), the conditional probability of the corrupted

Figure 3.8: *Artifacts of compressed noisy images by quantization in the wavelet domain.* Original image (left) and a noisy/quantized image (right) with $\sigma = 10/255$ and a 0.30 bpp compresion bit rate. Note the presence of pseudo-Gibbs effects, the loss of microtextures and some outliers.

coefficients given the original ones w is

$$p(w_{qn}|w) = \prod_k p(w_{qn}(k)|w(k)) \tag{3.35}$$

$$= \prod_k p(Q(w(k) + n(k)) = w_{qn}(k)) \tag{3.36}$$

$$= \prod_k p(w(k) + n(k) \in [a_k, b_k]) \tag{3.37}$$

$$= \prod_k p\left(\frac{n(k)}{\sigma} \in \left[\frac{a_k - w(k)}{\sigma}, \frac{b_k - w(k)}{\sigma}\right]\right). \tag{3.38}$$

But $n(k)/\sigma \sim N(0,1)$, then each of the last factors is the probability that a standarn normal variable falls into a given interval $[a_k, b_k]$. So

$$p(w_{qn}|w) = \prod_k \left[\Phi\left(\frac{b_k - w(k)}{\sigma}\right) - \Phi\left(\frac{a_k - w(k)}{\sigma}\right)\right] \tag{3.39}$$

where

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2}\, dt$$

is the normal cumulative distribution function (cdf).

In the following we consider the log-likelihood function

$$\begin{aligned} D(w) &= -\log p(w_{qn}|w) \\ &= -\sum_k \log\left[\Phi\left(\frac{b_k - w(k)}{\sigma}\right) - \Phi\left(\frac{a_k - w(k)}{\sigma}\right)\right]. \end{aligned} \tag{3.40}$$

This data fit term in the wavelet domain carefully takes into account the quantization process of the coefficients. Although this term is not quadratic as in most (linear) inverse problems, it is convex:

Proposition 1. *[250] The data fitting functional D in (3.40) is convex.*

Proof. Consider

$$h(z) = \Phi(\alpha - z) - \Phi(\beta - z) = \frac{1}{\sqrt{2\pi}} \int_{\alpha - z}^{\beta - z} e^{-t^2}\, dt.$$

This function is the convolution of two log-concave functions (the Gaussian density and the indicator function in $[\alpha, \beta]$) which is log-concave too (see [26], section 3.5.2), hence D is convex (is a sum of convex functions of the form $-\log h(z/\sigma)$). □

3.2.5 Wavelet Non-Local Bayes (WNLB)

The combination of acquisition noise and compression may severely damage the acquired image. The main objective of this chapter is to obtain a restored image from the quantised wavelets coefficients. We have decided to face the image compression and denoising jointly. This is not a classic approach (denoising and outliers removal are usually performed separately). However, to our opinion this approach enables a better global optimization and a more natural way to model the problem. It also enables the use of the most powerful denoising methods developed so far, those based on local gaussian models of patches: Non-Local Bayes (NLB) [131] or Piecewise Linear Estimators (PLE) [242].

In order to use these methods, we have to consider:

- Patches are defined on the wavelets domain instead of the spatial domain, which is the classic implementation of NLB.
- The data-fitting term must take into account both the noise generated by the sensor and the one derived from the quantisation process.
- The search of similar patches that are contaminated by a non-gaussian and variable noise.
- The denoising of empirical covariance matrices obtained from patches that were contaminated by a non-gaussian and variable noise.

These adaptations as well as the joint denoising and decompression procedure are condensed in a new method that we name *Wavelets Non Local Bayes* (WLNB) which is presented below.

Proposed method

In order to extend the idea of NLBayes presented in the previous section to take into account all the considerations explained before, several adaptations are needed:

- We have to take into account the exact acquisition model which does no longer lead to a white Gaussian noise.
- The exact model leads to a more involved minimization problem (it is no longer quadratic) that requires special techniques to be solved.
- Since the noise model can only be expressed exactly in the wavelet domain, patches should be constructed in that domain.

- The subsampling in the wavelet transform means that (as opposed to the spatial domain) we do not have access to all integer shifts. This has to be addressed by the use of patches in a redundant (tight) frame.

The joint effect of noise and compression results in a highly correlated noise in the spatial domain, with correlations at several scale levels. This renders patch-based MAP estimators intractable in the spatial domain. However, when analyzed in the wavelet domain the noise affecting each wavelet coefficient is nearly decorrelated from the others. For this reason our algorithm works with patches in the wavelet domain.

Patch shape

A patch

$$P_k = p_k(w) = \{w(k+l) \,:\, l \in \Omega_p\}$$

provides a context of neighboring coefficients $w(k+l)$ that should be as tightly correlated as possible to the central coefficient $w(k)$ when w is the wavelet transform of a natural image. In order to maximize this correlation the shape Ω_p of the patch should be chosen to consider neighboring coefficients along the spatial, subband and scale dimensions. Scale interactions can be considered via the multiscale procedure described below, so they are excluded from the definition of the patch shape, and three alternatives remain:

- $p \times p$: it is simply a patch of size p, centered at $w(k)$, without taking into consideration the other subbands.
- $p \times p \times 3$: We can group the three subbands at the same level, since they are strongly correlated.
- $p \times p \times (1+3)$: In addition to the previous case, we can add the summary of the same level. This means that the wavelet transform is applied one level at a time.

We have found experimentally that the option that gives the best results is the patch with a $p \times p \times (1+3)$ shape. The optimal size p depends on the signal to noise ratio and will be specified in the experiments.

Coarse-to-fine

When compression is done by a wavelet transform at L levels, then we restore one level at a time in a coarse-to-fine fashion. First the summary is restored along with level L subbands. A one level inverse wavelet transform permits to recover

the summary at level $L-1$, which will be restored again with the three $L-1$ level subbands. We proceed in this fashion until we recover the "summary" at level 0 which is the restored image u.

Prior Model

Like in the NLB method (Section 3.1.1) the prior model for clean patches is assumed to be a multivariate Gaussian

$$P_k \sim \mathcal{N}(\mu_k, \Sigma_k)$$

where the mean and covariance parameters (μ_k, Σ_k) are robustly estimated from a set of noisy patches similar to $\tilde{P}_k$ (or from a set of restored patches similar to the restored $\hat{P}_k$ in the second step). Estimating a Gaussian model under such conditions is a difficult problem and the details will be discussed in the following sections.

For the moment we shall assume that (μ_k, Σ_k) are known. Then the prior model is (in $-\log$ scale)

$$R_k(z) := -\log \mathbb{P}\left[P_k = z\right] = (z-\mu_k)^T \Sigma_k^{-1} (z-\mu_k) + C \tag{3.41}$$

where C is a constant that does not depend on z.

Numerical Optimization

Summarizing the two previous sections our Bayesian MAP estimator leads to the following optimization problem

$$\hat{P}_k = \arg\max_z \mathbb{P}\left[\, P_k = z \,\middle|\, \tilde{P}_k \,\right] \tag{3.42}$$

$$= \arg\min_z D_k(z) + R_k(z) \tag{3.43}$$

where the regularization term R_k is quadratic and the degradation model D_k is strictly convex with known gradient and diagonal Hessian. Normally one would solve such a problem with a Newton algorithm. But in this case the strict convexity of D_k is only theoretical, since the second derivative can become very close to zero when $q \gg \sigma$, and also R_k can be degenerate. In view of this, and to obtain a more tractable problem, we propose to add an auxiliary variable (Split-Bregman) and solve instead the following functional:

$$\min_{z_1} \min_{z_2} F_k(z_1, z_2) = D_k(z_1) + R_k(z_2) + \frac{\beta^2}{2} \|S_k^{-1}(z_1 - z_2)\|_2^2 \tag{3.44}$$

where

$$S_k = \mathrm{diag}((\sigma(k+l))_{l\in\Omega_p}).$$

Alternate minimization on z_1 and z_2 provides a convenient and convergent numerical scheme to solve this kind of optimization problems and its solution tends to the one of the original problem when $\beta \to \infty$. In practice we choose β to be an optimal compromise between two requirements: It should be large enough to ensure that $\|z_1 - z_2\|$ is as small as possible, and not too large to prevent the condition number to exceed $10^{-8} = \sqrt{\varepsilon_{\text{mach}}}$. The details of the optimization algorithm are given in [171].

3.2.6 JDD using CNN regularization

More recently, joint denoising and decompression procedures have been considered to remove both artifacts due to the compressor and its interaction with noise. Such methods use either TV regularization or patch-based Gaussian models in combination with relaxed versions of the quantization constraint, in order to take the effects of noise into account [68, 170, 171]. However the TV based approaches could only reliably remove isolated outliers in relatively constant areas, and patch-based approaches could only marginally improve the performance of standard denoising techniques like Non-Local Bayes [131].

In this section, we propose a novel method for joint denoising and decompression. Our method uses a probabilistic data-fitting term based on the formation model of noisy compressed images, coupled with a CNN-based regularization which captures natural image statistics more closely than previously reported patch-based methods. The proposed method is described in Section 3.2.6.

Motivation via MAP estimation

The maximum a posteriori estimation of the non-degraded image u knowing its degraded version u_{qn} is stated as

$$\hat{u} = \arg\max_u p(u|u_{qn}) = \arg\max_u \{p(u_{qn}|u)p(u)\} \tag{3.45}$$

$$= \arg\min_u \{-\log(p(u_{qn}|u)) - \log(p(u))\}, \tag{3.46}$$

where $\hat{u}$ is the MAP estimator of u. Finding $\hat{u}$ amounts to solve the optimization problem

$$\hat{u} = \arg\min_u \{D(u) + \lambda R(u)\}, \tag{3.47}$$

where $D(u)$ is a data-fitting term (defined on pixel space) that depends on the forward operator and the noise model, R is the regularization ($-\log$(prior)) to be used in the restoration, and the parameter $\lambda > 0$ is the strength of the regularization.

Minimization with ADMM

In Section 3.2.4 we construct a likelihood function on wavelet space. Hence, we can split problem (3.47) as

$$\min_{w,u}\{D(w)+\lambda R(u)\} \qquad \text{s.t. } W^{-1}w=u \tag{3.48}$$

where W^{-1} is the inverse (synthesis) wavelet transform. The ADMM algorithm [27] becomes

$$w_{k+1}=\arg\min_{w}\left\{D(w)+\frac{\rho}{2}\|W^{-1}w-u_k+\rho^{-1}y_k\|^2\right\} \tag{3.49}$$

$$u_{k+1}=\arg\min_{u}\left\{\lambda R(u)+\frac{\rho}{2}\|W^{-1}w_{k+1}-u+\rho^{-1}y_k\|^2\right\} \tag{3.50}$$

$$y_{k+1}=y_k+\rho(W^{-1}w_{k+1}-u_{k+1}) \tag{3.51}$$

where subscripts k indicate the iteration number.

For the first subproblem (3.49), let $v_k=-u_k+\rho^{-1}y_k$, then define

$$F(w):=D(w)+\frac{\rho}{2}\|W^{-1}w+v_k\|^2. \tag{3.52}$$

The first and second derivatives of $F(w)$ are given by

$$\nabla F(w)=\nabla D(w)+\rho W^{-T}(W^{-1}w+v_k)$$
$$\nabla^2 F(w)=\nabla^2 D(w)+\rho W^{-T}W^{-1}.$$

As pointed out in Section 3.2.3, for the biorthogonal wavelet CDF 9/7 the term WW^T can be fairly approximated by the identity matrix I, yielding

$$\nabla^2 F(w)\simeq\nabla^2 D(w)+\rho I. \tag{3.53}$$

Now, since $D(w)$ is separable in terms of the elements $w(k)$ of w, it follows that $\nabla^2 D(w)$ is a diagonal matrix. It is also positive semidefinite, since function $D(w)$ is convex. It follows that $\nabla^2 F(w)$ is a diagonal, positive definite matrix, and therefore the minimization of $F(w)$ can be computed very efficiently using a Newton-like minimization algorithm [26].

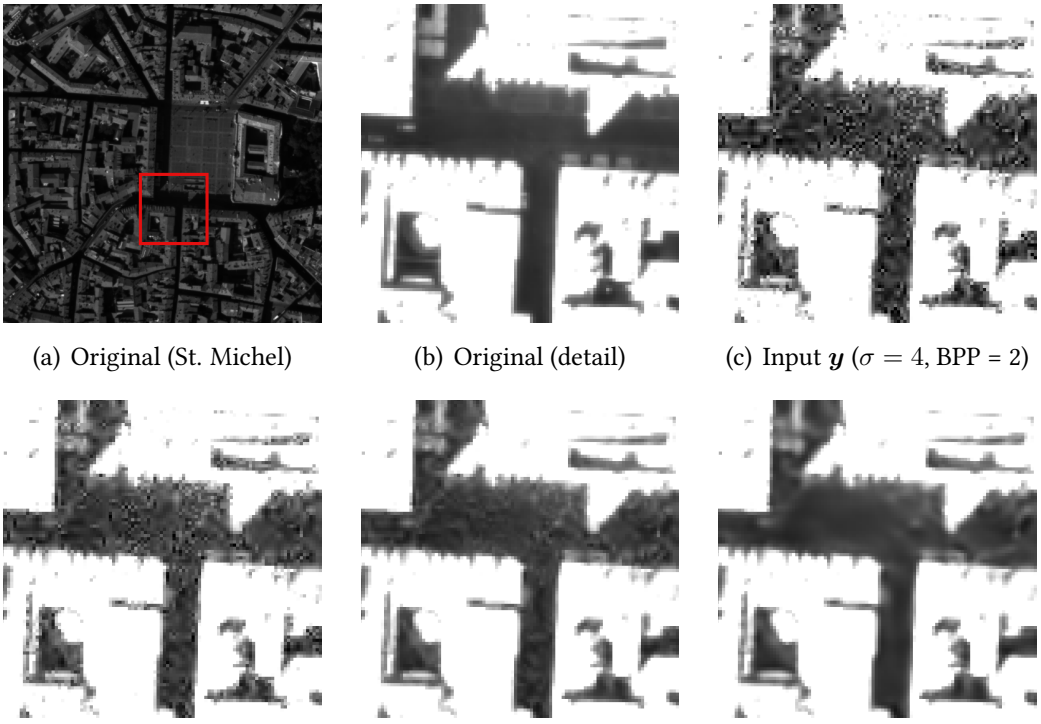
Regularizing by denoising

The second subproblem (3.50) can be rewritten as

$$u_{k+1}=\arg\min_{x}\left\{\frac{1}{2(\lambda/\rho)}\left\|u-(W^{-1}w_{k+1}+\rho^{-1}y_k)\right\|^2+R(u)\right\}. \tag{3.54}$$

As described in Section 3.1.2, this step can be seen as performing a denoising of

$$\tilde{u}_k=W^{-1}w_{k+1}+\rho^{-1}y_k \tag{3.55}$$



(a) Original (St. Michel) (b) Original (detail) (c) Input $\boldsymbol{y}$ ($\sigma = 4$, BPP = 2)

(d) WNLB [171] (e) Zhang et al. [245] (f) Proposed method

Figure 3.9: Top: original and noisy compressed images. Below: results of three restoration methods. Dynamic range has been saturated for better visualization.

with noise variance $\sigma_{\mathcal{G}}^2 = \lambda/\rho$. Hence, the solution can be approximated using a state-of-the-art denoiser $\mathcal{G}(\tilde{u}, \sigma_{\mathcal{G}}^2)$ as the proximal operator of an *implicit prior* $R(u)$ [150]:

$$u_{k+1} = \mathcal{G}(\tilde{u}_k, \quad \sigma_{\mathcal{G}}^2 = \lambda/\rho). \tag{3.56}$$

In our experiments, we choose $\mathcal{G}$ to be the residual network of [245], which was a state-of-the-art Gaussian denoising algorithm at the time of publication of our work [86].

Experimental results

Figure 3.9 illustrates the artifacts that result from noisy compressed images, and compares different restoration methods. The original image was corrupted with white Gaussian noise of $\sigma = 4/255$, then compressed at 2 BPP using the CCSDS compressor. Two different phenomena can be distinguished in the noisy compressed image: a loss of details resulting from wavelet coefficients truncation, and wavelet shaped artifacts resulting from wavelet coefficients outliers. In regions where the variable quantization step $q(k)$ is such that $\sigma > q(k)$, most wavelet coefficients actually become outliers and the structure is very close to white Gaus-

Image	PSNR	SSIM
Corrupted ($\sigma = 4/255$, 2 BPP)	35.92	0.8320
WNLB [171]	36.67	0.8537
Zhang et al. [245]	**39.59**	0.9169
Proposed method	39.52	**0.9241**

Table 3.1: *Experimental results for the proposed JDD algorithm.* For PSNR and SSIM, higher is better.

sian noise. In this case, the Gaussian denoiser [245] and our method exhibit similar performances. However, on the other side, when $\sigma \ll q(k)$ the wavelet shaped artifacts become more isolated and the degradation strongly deviates from white Gaussian noise. In this case, [245] cannot get its full potential and many of these artifacts are not removed, while our method performs particularly well.

Table 3.1 presents a quantitative analysis of the proposed approach by comparing its corresponding PSNR and SSIM [235] to those of WNLB and [245]. Even though [245] exhibits slightly better PSNR, our method performs the best in the more subjective quality metric SSIM, which is consistent with the quality evaluation by visual inspection showing that the proposed approach removes more outliers while better preserving image details.

3.3 Multi-Image Super Resolution

3.3.1 Modeling and simplifications

In this section, we concentrate on another inverse problem, namely, multi-image super-resolution. We assume that the observed ideal high-resolution scene u is acquired K times by a frame or push-broom captor, producing K degraded, low-resolution images (the sensors' gain factors are corrected at the L1a level):

$$\tilde{u}_i = S_{\mathbb{Z}^2}(h * (u \circ \phi_i)) + n_i, \quad i = 1, \dots, K, \tag{3.57}$$

where:

- S_Γ denotes the sampling operator on the grid $\Gamma = \mathbb{Z}^2$. In other words, the row l of this sampling grid consists of a cardinal sine centered at point $\gamma_l \in \Gamma$ of the grid: $S_\Gamma(l, m) = \mathrm{sinc}(m - \gamma_l)$.
- h is the blurring kernel (the point spread function of the instrument), considered to be constant in the absence of active optics.

- $\phi_i : \mathbb{R}^2 \to \mathbb{R}^2$ is a geometric deformation associated to the i-th image acquisition.

- $n_i : \mathbb{Z}^2 \to \mathbb{R}$ is a realization of a Gaussian noise, independent between different pixel locations and different acquisitions. The noise variance typically depends on the observed luminance, *i.e.* $n_i(x) \sim N(0, a_i + b_i L_i(x))$ where $L_i(x) = (h_i * (u \circ \phi_i)))(x)$ is the luminance at pixel x, and a_i and b_i are sensor parameters.

The goal of super-resolution is to restore as accurately as possible the ideal convolved scene $h * u$ (of size $M \times N$ pixels) from its K degraded, low-resolution acquisitions $\tilde{u}_i$ (of sizes $m \times n$ pixels). The sub-sampling factor is $s = \sqrt{\frac{mn}{MN}}$, and the zoom factor is its inverse value $z = 1/s$.

Model linearization for the case of low steroscopic ratio

As a consequence of the geometric deformations involved in the acquisition, the problem we address here is non-linear and hard to solve in its original form. Nevertheless, under the (realistic) hypothesis that the geometric deformations are smooth enough, we can consider a first order affine approximation (translation, rotation, tilt and zoom):

$$\phi_i(x) \approx t_i + A_i x + o(x^2). \tag{3.58}$$

Under this approximation, the image formation model can be written as

$$\tilde{u}_i \approx S_{\phi_i(\mathbb{Z}^2)}(h_i * u) + n_i, \quad i = 1, \dots, K \tag{3.59}$$

where the blurring kernel $h_i(x) = h(A_i^{-1}x)$ is deformed by the affine term of the geometric deformation, and the sampling grid $\Gamma_i = \phi_i(\mathbb{Z}^2)$ becomes an irregular grid in the image sensor plane.

If the affine terms A_i of the deformations are not too far from a similarity transform (a realistic hypothesis in the case of low enough stereoscopic factor B/H between all K acquisitions), then all the blurring kernels will be approximately equal: $h_i \approx h$. In this case the system resolution becomes simpler and is given by

$$\tilde{u} = S_\Gamma(\underbrace{h * u}_{u_{HR}}) + \eta, \tag{3.60}$$

where $\tilde{u} = (\tilde{u}_1; \dots; \tilde{u}_K)$ is the concatenation of all the observations and $\Gamma = \cup_i \Gamma_i$ is the superposition of all registered sampling grids $\Gamma_i = \phi_i(\mathbb{Z}^2)$.

In the following, we assume that the acquisition conditions are such that the model given by (3.60) holds. We also assume that the shifts ϕ_i are known.

Numerical methods for spatial super-resolution

As previously mentioned, the problem of spatial multi-image super-resolution can be stated as the inversion of system of non-linear equations, which under favorable and reasonable conditions that hold in practice, can be well approximated by a linear formulation. The inversion of such linear system can be challenging due to the following factors:

- Ill-posedness of the linear operator S_Γ to be inverted, when the number of acquisitions to be fused is not large enough, or when the distribution of the corresponding shifts is not uniform enough:
- The presence of instrumental noise η;
- The size of the linear operator S_Γ to be inverted is too large to be stored exhaustively in memory.

3.3.2 CNN-based regularization with ℓ^2 data misfit term

The Plug & Play method by Meinhardt *et al.* [150]

We first consider the variational problem

$$\arg\min_u \frac{1}{\lambda}\left\{\|Au-u_0\|^2 + R(u)\right\} \tag{3.61}$$

where A is a linear operator ant u_0 are the measurements, that is the low-resolution image stack. Let $f(v) = \frac{1}{\lambda}\|v-u_0\|^2$, the inverse problem (3.61) can be written as

$$\arg\min_u \left\{f(Au) + R(u)\right\}, \tag{3.62}$$

and solved with Chambolle-Pock splitting algorithm [39], which consists in the following iterative scheme:

$$\begin{cases} q^{k+1} = \underset{q=(q_1,\dots,q_K)}{\arg\min}\left\{\frac{1}{2}\|q-(q^k+\sigma A\bar{u}^k)\|^2 + \sigma f^\star(q)\right\} \\ u^{k+1} = \underset{u}{\arg\min}\left\{\frac{1}{2}\|u-(u^k-\tau A^* q^{k+1})\|^2 + \tau R(u)\right\} \\ \bar{u}^{k+1} = 2u^{k+1}-u^k. \end{cases} \tag{3.63}$$

The condition $\sigma\tau K < 1$ has to be met to ensure convergence in the case where R is a convex, lower semi-contiuous function.

The Legendre-Fenchel transform $f^\star(q)$ considered in (3.63) can be computed analytically as

$$f^\star(q) = \lambda\left\|\frac{q}{2} + \frac{u_0}{\lambda}\right\|^2 - \frac{1}{\lambda}\|u_0\|^2. \tag{3.64}$$

Hence, it can be shown that the update step for the dual variable writes

$$q^{k+1} = \frac{q^k + \sigma A\overline{u}^k - \sigma u_0}{1 + \sigma\lambda/2}\,. \tag{3.65}$$

Following the plug and play approach, the second minimization problem (the MAP estimation in the Bayesian framework) as a denosing of the image $u^k - \tau A^* q^{k+1}$, corrupted with noise of variance $1/\tau$. Hence, substituting the primal stage in (3.63) by a Gaussian denoiser (here a CNN-based denoiser) leads to Algorithm 1, where

$$\mathcal{G}(\tilde{u}, \sigma_{\mathcal{G}}^2) = \arg\min_u \left\{ \frac{1}{2\sigma_{\mathcal{G}}^2} \|u - \tilde{u}\|_2^2 + R(u) \right\} \tag{3.66}$$

denotes a Gaussian denoising of image $\tilde{u}$ for a noise variance $\sigma_{\mathcal{G}}^2$.

Algorithm 1: Plug & Play approach associated to (3.63)

Initialisation: *choose* $\tau, \sigma > 0$ *s.t.* $\tau\sigma K < 1$, $q^0 = 0$, $u^0 \in \mathbb{R}^\Omega$ *and* $\overline{u}^0 = u^0$.

Itérations: *for* $k \geq 0$, *update* q^k, u^k *and* $\overline{u}^k$ *as follows*

$$\begin{aligned}
q^{k+1} &= \frac{q^k + \sigma(A\overline{u}^k - u_0)}{1 + \sigma\lambda/2} && \textbf{complexity:} \quad \mathcal{O}(Kmn\log(mn)) \\
u^{k+1} &= \mathcal{G}(u^k - \tau A^* q^{k+1},\ \sigma_{\mathcal{G}}^2 = 1/\tau) && \textbf{complexity:} \quad \mathcal{O}(MN \times 1,9 \times 10^5) \\
\overline{u}^{k+1} &= 2\,u^{k+1} - u^k && \textbf{complexity:} \quad \mathcal{O}(MN)
\end{aligned}$$

Choice of parameters and experiments

The convergence conditions of the algorithm (in the convex case) require to fix the steps τ and σ such that $\sigma\tau K < 1$. Moreover, the value of $\tau = 1/\sigma_{\mathcal{G}}^2$ is related to the noise variance for which the network was trained. However, in our case, the network was trained for $\sigma_{\mathcal{G}} \geq 2$. Therefore, we are constained to use a step $\tau \geq 1/4$. To obtain steps as large as possible while still ensuring convex convergence, we have subsequently chosen $\sigma = \frac{0.9}{\tau K} \approx 0.011$.

For some experiments (such as the ones shown in Figures 3.10 to 3.12 and Table 3.2), this led to very promising results. When we sought to reproduce the same results on a larger set of images, we have faced difficulties regarding convergence of Algorithm 1. In particular, we have observed a strong oscillatory behaviour along the iterations produced by the scheme. Several causes may explain this observed behavior:

Table 3.2: Reconstruction results obtained with Algorithm 1 on two image stacks provided by CNES.

(a) SNR@L1 = 45 dB

♯ images	PSNR
3	38.30 dB
5	**49.26 dB**
20	46.54 dB

(b) SNR@L1 = 8 dB

♯ images	PSNR
20	37.54

1. The criterion for the parameters' choice produces a strong disequilibrium between the primal step τ and the dual step σ. Typically, we observa a factor of 200 between them. Normally one would seek to balance this two steps, but this would require retraining the network for a much lower value of $\sigma_{\mathcal{G}}$, what was not possible to carry out in the framework of this study.

2. The convergence condition that was considered holds for the convex case, while here we are probably on a non-convex setting.

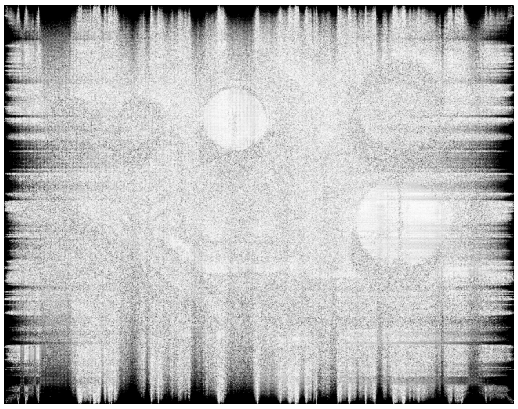
3. More fundamentally, nothing guarantees that the denoising CNN actually computs the proximal operator of any energy R, convex or not. If the denoiser is not the proximal operator of an energy, then less elements we have to analyze the convergence of this algorithm.

3.4 Conclusions

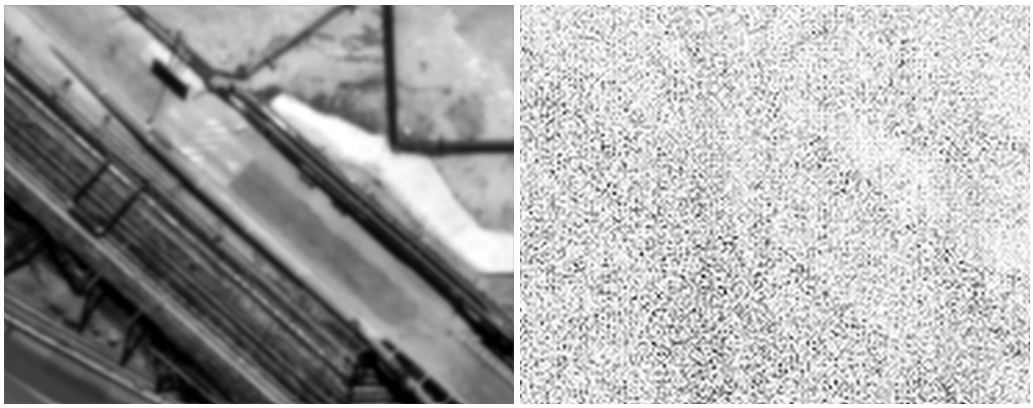
In this chapter we reviewed the Plug-and-Plug approach to use denoising algorithms to regularized different inverse problems. We applied this approach on two problems, namely, joint denoising and decompression and multi-image multiresolution. In the second one, the obtained results were not published in an academic journal but contributed to an accepted patent.

A more systematic study should be necessary to improve the convergence guarantees of this algorithm. First works on Plug-and-Play methods such as [227] rely on empirical success and do not provide convergence guarantees. After the publication of our first work, several convergence proofs has been developed (e.g. [192]) but at that time we followed other research lines as explained in the following chapters.

(a) Restored image and its spectrum

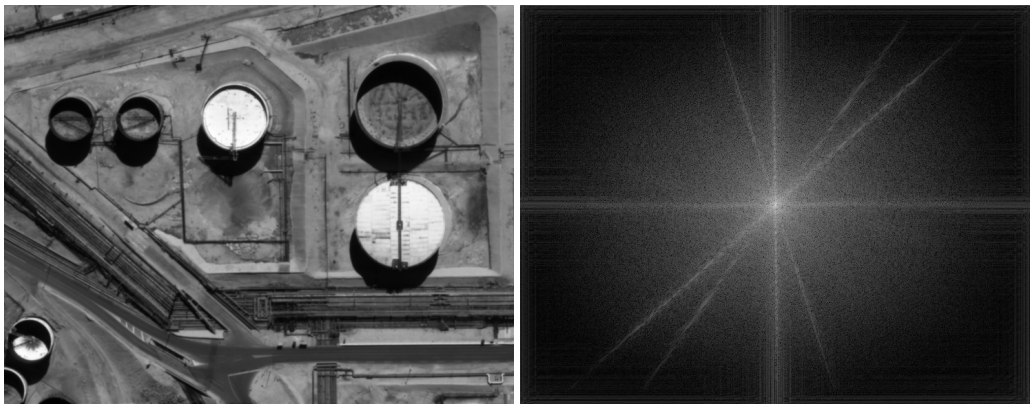


(b) Difference with ground truth

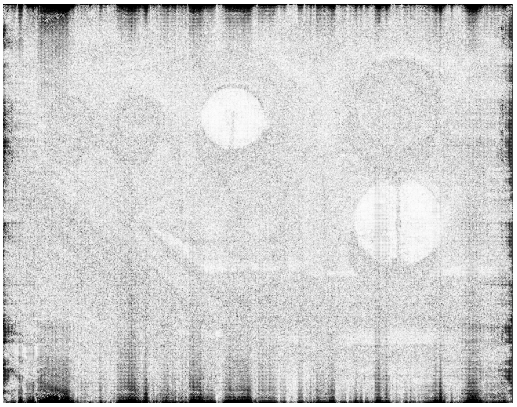


(c) Zoom of (a) and (b)

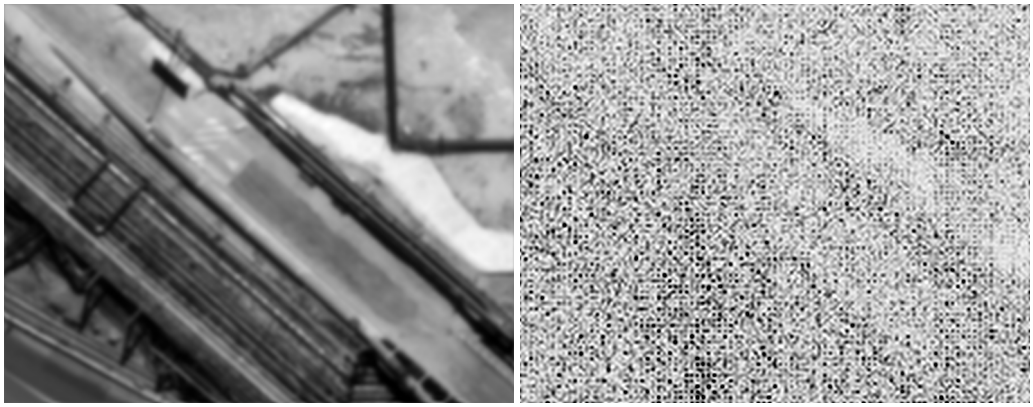
Figure 3.10: Restoration results obtained with Algorithm 1 on a stack of 20 images with SNR@L1 = 45 dB (PSNR = 46,5 dB)



(a) Restored image and its spectrum



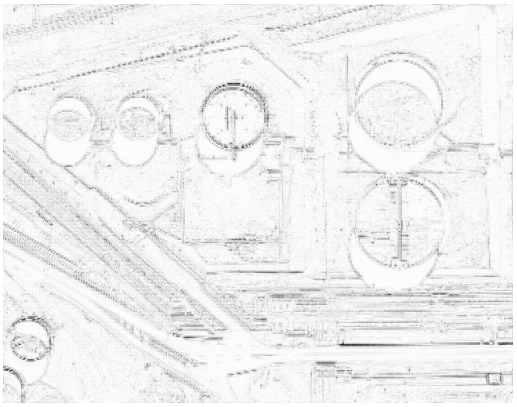
(b) Difference with ground truth



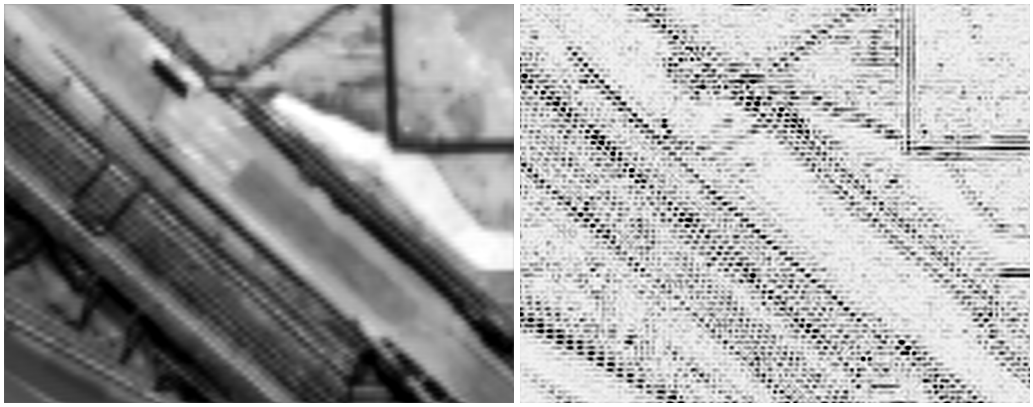
(c) Zoom of (a) and (b)

Figure 3.11: Restoration results obtained with Algorithm 1 on a stack of 5 images with SNR@L1 = 45 dB (PSNR = 49,3 dB)

(a) Restored image and its spectrum



(b) Difference with ground truth



(c) Zoom of (a) and (b)

Figure 3.12: Restoration results obtained with Algorithm 1 on a stack of 3 images with SNR@L1 = 45 dB, (PSNR = 38.3 dB)

Chapter 4

Joint Posterior Maximization with Autoencoding Prior

In this chapter we address the problem of solving ill-posed inverse problems in imaging where the prior is a variational autoencoder (VAE). Specifically we consider the decoupled case where the prior is trained once and can be reused for many different log-concave degradation models without retraining. Whereas previous MAP-based approaches to this problem lead to highly non-convex optimization algorithms, our approach computes the joint (space-latent) MAP that naturally leads to alternate optimization algorithms and to the use of a stochastic encoder to accelerate computations. The resulting technique (JPMAP) performs Joint Posterior Maximization using an Autoencoding Prior. We show theoretical and experimental evidence that the proposed objective function is quite close to bi-convex. Indeed it satisfies a weak bi-convexity property which is sufficient to guarantee that our optimization scheme converges to a stationary point. We also highlight the importance of correctly training the VAE using a denoising criterion, in order to ensure that the encoder generalizes well to out-of-distribution images,

without affecting the quality of the generative model. This simple modification is key to providing robustness to the whole procedure. Finally we show how our joint MAP methodology relates to more common MAP approaches, and we propose a continuation scheme that makes use of our JPMAP algorithm to provide more robust MAP estimates. Experimental results also show the higher quality of the solutions obtained by our JPMAP approach with respect to other non-convex MAP approaches which more often get stuck in spurious local optima.

4.1 Introduction

Since deep neural networks (NN) showed their superiority in image classification tasks [125] researchers started to look for ways to use this tool to solve inverse problems too. The most straightforward attempts employed neural networks as *regressors* to learn a risk minimizing mapping $\boldsymbol{y} \mapsto \boldsymbol{x}$ from many examples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ either agnostically [63, 244, 243, 81, 199, 76] or including the degradation model in the network architecture via unrolled optimization techniques [91, 42, 60, 84].

Implicitly decoupled priors The main drawback of neural networks regression is that they require to retrain the neural network each time a single parameter of the degradation model changes. To avoid the need for retraining, another family of approaches seek to *decouple* the NN-based learned image prior from the degradation model. A popular approach within this methodology are *Plug & Play* (or PnP) methods. Instead of directly learning the log-prior $-\log p_X(\boldsymbol{x}) = G(\boldsymbol{x}) + C$, these methods seek to learn an approximation of its gradient ∇G [21, 20] or proximal operator prox_G [227, 149, 245, 40, 114, 192], by replacing it by a denoising NN. Then, these approximations are used in an iterative optimization algorithm to find the corresponding MAP estimator in Equation (4.2) or more generally some sort of consensus equilibrium among the data fitting term and the priors [32].

Taking an apparently different approach Romano *et al.* introduced the regularization by denoising (RED) algorithm [187] which uses a denoiser D_σ to construct an explicit regularizer $G(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T(\boldsymbol{x} - D_\sigma(\boldsymbol{x}))$. Under certain conditions (see below) its gradient $\nabla G = Id - D_\sigma$ can be conveniently computed in terms of the denoiser, leading to a gradient descent scheme for the associated MAP estimator, which is very easy to implement.

Explicitly decoupled generative priors In another series of works pioneered by Bora *et al.* [24] and followed by [200, 176, 151, 105, 97] the Plug & Play prior is explicitly provided by a generative model, most often a generative adversarial network D that maps a latent variable $\boldsymbol{z} \sim \mathcal{N}(0, Id)$ to an image $\boldsymbol{x} = \mathsf{D}(\boldsymbol{z})$ with the desired distribution p_X as represented by the learning dataset. More precisely these methods solve an optimization problem on the latent variable $\boldsymbol{z}$

$$\hat{\boldsymbol{z}}_{\text{MAP}} = \arg\min_{\boldsymbol{z}} \left\{ F(\mathsf{D}(\boldsymbol{z}), \boldsymbol{y}) + \frac{1}{2}\alpha\|\boldsymbol{z}\|^2 \right\} \tag{4.1}$$

and the reconstructed image is provided by $\hat{\boldsymbol{x}}_{\text{MAP}} = \mathsf{D}(\hat{\boldsymbol{z}}_{\text{MAP}})$. As we show in the following sub-section and in appendix 4.5.2, this corresponds (when $\alpha = 1$) to the Maximum A Posteriori (MAP) estimator with respect to the $\boldsymbol{z}$ variable. In this work we adopt this framework with some extensions that help avoid getting trapped in spurious critical points of the non-convex objective function.

Empirical success of Plug & Play and RED Plug & Play and RED approaches became very popular because they allow to repurpose very powerful state of the art denoisers as regularizers of a large family of inverse problems in a quite straightforward manner. They have been successfully applied to many different problems in imaging and they have thus empirically proven their superiority (in terms of achievable reconstruction quality with respect to more classical regularization techniques), and opened the way for the solution of more difficult inverse problems in imaging.

Theoretical questions The success of Plug & Play and RED approaches largely outpaced our understanding of why and when these techniques lead to algorithms that provably converge to well-posed statistical estimators with well known properties. This is not surprising because obtaining convergence guarantees for non-convex optimization problems under realistic conditions is quite challenging.

A notable exception where strong convergence results have been obtained is the particular case of compressed sensing, where the lines of the degradation operator (or sensing matrix) $\boldsymbol{A}$ are independent realizations of a zero-mean Gaussian distribution. In this particular case Hand *et al.* [97, 105] show that the optimization objective (4.1) has almost no spurious stationary points. As a consequence, a minor modification to the gradient descent algorithm in [24] converges with high probability to the global optimum.

In this work we are interested in more general inverse problems, where the sensing matrix $\boldsymbol{A}$ is not necessarily random but deterministic and highly structured most often dictated by our modeling of the acquisition device. In this more general setting the hypotheses of the CS results are not necessarily satisfied, and the kind of convergence guarantees that could be established for PnP algorithms with non-convex priors are much weaker (typically only convergence to a stationary point or fixed point is provided, not necessarily a global optimum), and most works concentrate in the implicit case, where the prior is not explicitly provided by a generative model, but implicit in a denoising algorithm.

newIn such a case the actual prior is unknown, the existence of a density whose gradient or proximal operator is well approximated by a neural denoiser is most often not guaranteed [178], and the convergence of the algorithm is not guaranteed either unless the denoiser is retrained with specific constraints like idempotence [94, 200], contractive residual [192] or exact, invertible, smooth MMSE denoisers [240].

The effect of such training constraints on the quality of the denoisers and the associated priors is yet to be explored in detail. But even when these constraints are satisfied, convergence conditions can be quite restrictive, either *(a)* requiring the data-fitting term F to be strongly convex [192] (thus excluding many important problems in computational imaging where $\boldsymbol{A}$ is not full rank like interpolation,

super-resolution, deconvolution with a non-invertible kernel or compressive sensing), and/or *(b)* constraining the regularization parameter λ outside of its useful range [192, 240].[1]

Similarly, an early analysis of the RED approach [178] provides a convergence proof, but only under quite restrictive conditions (locally homogeneous denoisers with symmetric Jacobian) which exclude most state of the art denoisers like DnCNN, BM3D, NLMeans. A more recent analysis of a stochastic variant of the RED algorithm [130] (called PnP-SGD) significantly expands the family of denoisers that provide convergence guarantees, including in particular DnCNN in addition to the doubly-stochastic variant of NLM [208]. These guarantees come, however, at the expense of a very small descent step which leads to a very computationally expensive algorithm with slow convergence. In addition, the experiments with PnP-SGD show that this algorithm is extremely sensitive to the initial condition, and it can be easily get stuck on spurious local minima if not initialized very carefully.

Focus of this work Very recent works focused on developing MAP estimation algorithms with convergence guarantees under more realistic conditions. The convergence analysis of the RED framework, and its RED-PRO variant was further refined under a demicontractive condition for the denoiser [48]. This condition is, however, difficult to verify according to [167] who provides an alternative convergence analysis framework based on firmly non-expansive denoisers for which explicit training procedures exist [215]. In this work we explore alternative new ways to bring theory and practice closer together, by proposing novel Plug & Play algorithms to compute the MAP estimator of an inverse problem with a neural regularizer. Unlike previous approaches which were based on implicit priors, or on GAN-based explicit priors, our approach is based on an explicit generative prior that has been trained as a Variational AutoEncoder (VAE). As we shall see later, the additional VAE structure provides: *(i)* powerful mechanisms to avoid getting stuck in spurious local minima of the associated non-convex functional, and *(ii)* convergence guarantees under much less restrictive conditions on the inverse problem F and regularization parameter λ.

The next Section 4.1.1 reviews previous work on similar approaches to compute a MAP estimator from a generative prior that was trained either as a VAE or a GAN. Section 4.1.2 briefly introduces our approach and how it relates to previous work. The section finishes with an overview of the rest of the paper.

[1] In [130] the PnP-ADMM and PnP-FBS algorithms introduced in [192, 240] are reported to: *(i)* Converge in practice quite far beyond the conditions of the theorem, but *(ii)* Require (to obtain optimal performance) the regularization parameter λ to be tuned to values that are far outside the region where convergence is guaranteed. *(iii)* Performance is significantly degraded if λ is constrained to the range where convergence is guaranteed.

4.1.1 Maximum a Posteriori meets Generative Models

Our approach focuses on PnP algorithms where the prior is provided by a generative model. For instance one could use a generative adversarial network (GAN) to learn a generative model for $X = \mathsf{D}(\mathcal{Z})$ with $\mathcal{Z} \sim N(0, I)$ a latent variable. The generative model induces a prior on X via the push-forward measure $p_{\mathcal{X}} = \mathsf{D}\sharp p_{\mathcal{Z}}$, which following [165, section 5] can be developed as

$$p_{\mathcal{X}}(\boldsymbol{x}) = \frac{p_{\mathcal{Z}}\left(\mathsf{D}^{-1}(\boldsymbol{x})\right)}{\sqrt{\det S(\mathsf{D}^{-1}(\boldsymbol{x}))}}\delta_{\mathcal{M}}(\boldsymbol{x})$$

where $S = \left(\frac{\partial \mathsf{D}}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial \mathsf{D}}{\partial \boldsymbol{z}}\right)$ is the squared Jacobian and the manifold $\mathcal{M} = \{\boldsymbol{x} : \exists \boldsymbol{z},\ \boldsymbol{x} = \mathsf{D}(\boldsymbol{z})\}$ represents the image of the generator D. With such a prior $p_{\mathcal{X}}$, the $\boldsymbol{x}$-optimization

$$\hat{\boldsymbol{x}}_{\text{MAP}} = \arg\max_{\boldsymbol{x}} p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \{F(\boldsymbol{x}, \boldsymbol{y}) + \lambda G(\boldsymbol{x})\} \tag{4.2}$$

required to obtain $\hat{\boldsymbol{x}}_{\text{MAP}}$ becomes intractable (in general), for various reasons:

- the computation of $\det S$,
- the inversion of D, and
- the hard constraint $\boldsymbol{x} \in \mathcal{M}$.

These operations are all memory and/or computationally intensive, except when they are partially addressed by the use of a normalizing flow like in [101, 238].

Current attempts to use such a generative model as a prior, like the one proposed by [24] for GANs, circumvent these difficulties by performing an optimization on $\boldsymbol{z}$ (in the latent domain) instead of $\boldsymbol{x}$. Instead of solving Equation (4.2), they solve

$$\begin{aligned} \hat{\boldsymbol{z}}_{\text{MAP}} &= \arg\max_{\boldsymbol{z}} \left\{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \mathsf{D}(\boldsymbol{z}))\, p_{\mathcal{Z}}(\boldsymbol{z})\right\} \\ &= \arg\min_{\boldsymbol{z}} \left\{F(\mathsf{D}(\boldsymbol{z}), \boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2\right\}, \end{aligned} \tag{4.3}$$

by assuming a standard Gaussian prior. This problem is much more tractable, and the corresponding $\boldsymbol{x}$-estimate is obtained as

$$\hat{\boldsymbol{x}}_{\text{MAP}-\boldsymbol{z}} = \mathsf{D}(\hat{\boldsymbol{z}}_{\text{MAP}}). \tag{4.4}$$

As we show in appendix 4.5.2, this new estimator does not necessarily coincide with $\hat{\boldsymbol{x}}_{\text{MAP}}$ but it does correspond to the MAP-estimator of $\boldsymbol{x}$ after the change of variable $\boldsymbol{x} = \mathsf{D}(\boldsymbol{z})$, namely

$$\hat{\boldsymbol{x}}_{\text{MAP}-\boldsymbol{z}} = \mathsf{D}\left(\arg\max_{\boldsymbol{z}} \left\{p_{\mathcal{Z}|\mathcal{Y}}(\boldsymbol{z} \mid \boldsymbol{y})\right\}\right).$$

Since D is non-linear, this problem (or its equivalent formulation (4.3)) is highly non-convex and difficult to solve with global optimality guarantees. Nevertheless, in the particular case where $\boldsymbol{A}$ is a random Gaussian matrix (compressed sensing case) or when F is strongly convex, recent work shows that the global optimum can be reached with linear convergence rates by a small modification of a gradient descent algorithm [105, 97], or by an ADMM algorithm with non-linear constraints [128, 14, 226]. To the best of our knowledge, these results do not extend, however, to the more general case we are interested in here, where $\boldsymbol{A}$ is deterministic and rank-deficient, and F is consequently not strongly convex. In this more general setting, convergence guarantees for this optimization problem remain extremely difficult to establish, as confirmed by experimental results presented in Section 4.3.

A common technique to solve difficult optimization problems like the one in Equation (4.3) is to use (Half Quadratic) splitting methods

$$\hat{\boldsymbol{x}}_\beta = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{z}} \underbrace{\left\{ F(\boldsymbol{x}, \boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{x} - \mathsf{D}(\boldsymbol{z})\|^2 + \frac{1}{2}\|\boldsymbol{z}\|^2 \right\}}_{J_{1,\beta}(\boldsymbol{x},\boldsymbol{z})} \tag{4.5}$$

combined with a continuation scheme, namely:

$$\hat{\boldsymbol{x}}_{\text{MAP}-\boldsymbol{z}} = \lim_{\beta \to \infty} \hat{\boldsymbol{x}}_\beta. \tag{4.6}$$

The convergence of the continuation scheme in the last line is a standard result in Γ-convergence (see [52] and appendix 4.5.3). The corresponding splitting algorithm is presented in Algorithm 2.

Algorithm 2: MAP-$\boldsymbol{z}$ splitting

Require: Measurements $\boldsymbol{y}$, Initial condition $\boldsymbol{x}_0$, maxiter, $k_{\max}$, $\{\beta_0, \ldots, \beta_{k_{\max}}\}$
Ensure: $\hat{\boldsymbol{x}} = \mathsf{D}\left(\arg\max_{\boldsymbol{z}} p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z} \mid \boldsymbol{y}\right)\right)$

1: **for** $k := 0$ **to** $k_{\max}$ **do**
2: $\quad \beta := \beta_k$
3: $\quad$ **for** $n := 0$ **to** maxiter **do**
4: $\quad\quad \boldsymbol{z}_{n+1} := \arg\min_{\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x}_n, \boldsymbol{z})$ // Nonconvex
5: $\quad\quad \boldsymbol{x}_{n+1} := \arg\min_{\boldsymbol{x}} J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z}_{n+1})$ // Quadratic
6: $\quad$ **end for**
7: $\quad \boldsymbol{x}_0 := \boldsymbol{x}_{n+1}$
8: **end for**
9: **return** $\boldsymbol{x}_{n+1}$

However, unlike most cases of HQS which include a linear constraint between the two variables, this splitting algorithm still contains (line 4) a difficult non-convex optimization problem[2].

[2]In another context a primal-dual optimization algorithm was proposed to solve a similar opti-

4.1.2 Proposed method: Joint MAP$_{x,z}$

In this work we propose to address this challenge by substituting the difficult non-convex sub-problem by a local quadratic approximation provided by the encoder of a variational autoencoder.

Indeed, as we show in Section 4.2, a variational autoencoder allows to interpret the splitting Equation (4.5) as the negative logarithm of the joint posterior density $p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{y})$. Therefore, solving Equation (4.5) amounts to compute a joint MAP$_{x,z}$ estimator that we denote by $\hat{\boldsymbol{x}}^{\beta}_{\text{MAP}_{\boldsymbol{x},\boldsymbol{z}}}$. Moreover if the same joint conditional density $p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{y})$ is decomposed in a different manner, it leads to an approximate expression that makes use of the encoder, and is quadratic in $\boldsymbol{z}$. If this approximation is good enough then the maximization of the joint log-posterior becomes a bi-concave optimization problem or approximately so. And in that case, an extension of standard bi-convex optimization results [89] shows that the algorithm converges to a stationary point.

We also highlight the importance of correctly training the VAE in such a way that the encoder generalizes well to noisy values of $\boldsymbol{x}$ outside of the support of $p_{\mathcal{X}}(\boldsymbol{x})$. This can be achieved by training the VAE to reconstruct their clean inputs with noise injected at the input level, as proposed by [109]. We observe that this modified training does not degrade the quality of the generative model, but makes our quasi-bi-convex optimization procedure much more robust.

Finally we show that a continuation scheme allows to obtain the MAP$_{\boldsymbol{z}}$ estimator as the limit of a series of joint MAP$_{\boldsymbol{x},\boldsymbol{z}}$ optimizations. This continuation scheme, in addition to the quasi-bi-convex optimization, and the initialisation heuristic provided by the denoising encoder leads to a much more robust non-convex optimization scheme which more often converges to the right critical point than a straightforward gradient descent of the MAP$_{\boldsymbol{z}}$ model.

The remainder of this paper is organized as follows. In Section 4.2 we derive a model for the joint conditional posterior distribution of space and latent variables $\boldsymbol{x}$ and $\boldsymbol{z}$, given the observation $\boldsymbol{y}$. This model makes use of a generative model, more precisely a VAE with Gaussian decoder. We then propose an alternate optimization scheme to maximize the joint posterior model, and state convergence guarantees. Section 4.3 presents first a set of experiments that illustrates the convergence properties of the optimization scheme. We then test our approach on classical image inverse problems, and compare its performance with state-of-the-art methods. Concluding remarks are presented in Section 4.4.

mization problem [14], but this approach was not explored in the context where D is a generative model.

4.2 From Variational Autoencoders to Joint Posterior Maximization

Recently, some generative models based on neural networks have shown their capability to approximate the complex image distribution in a data-driven fashion. In particular, *Variational Autoencoders (VAE)* [121] combine variational inference to approximate unknown posterior distributions of latent variable models with the ability of neural networks to learn such approximations.

Consider a graphical model $\boldsymbol{z} \to \boldsymbol{x}$ in which we assume that a latent variable $\boldsymbol{z}$ is responsible of the observed image $\boldsymbol{x}$. For example, in an image of a handwritten digit we can imagine which digit is represented in the image, width, angle (and so on) as latent variables. We choose a generative model

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_{\mathcal{Z}}(\boldsymbol{z})$$

where $p_{\mathcal{Z}}(\boldsymbol{z})$ is some simple distribution (which we can easily sample from) and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the approximation of the probability distribution of $\boldsymbol{x}$ given $\boldsymbol{z}$ parameterized by a neural network (with weights θ) known as *stochastic decoder*.

The intractability of $p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_{\mathcal{Z}}(\boldsymbol{z})\, d\boldsymbol{z}$ is related to the posterior distribution $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ by

$$p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_{\mathcal{Z}}(\boldsymbol{z})}{p_\theta(\boldsymbol{x})}. \tag{4.7}$$

The *variational inference* approach consists in approximating this posterior with another model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ which, in our case, is another neural network with parameters ϕ, called a *stochastic encoder*.

Following [121], we consider the *Evidence Lower BOund (ELBO)* as

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) := \log p_\theta(\boldsymbol{x}) - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z}|\boldsymbol{x})) \leq \log p_\theta(\boldsymbol{x}) \tag{4.8}$$

where KL is the Kullback-Leibler divergence. Thus, given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of image samples, maximizing the averaged ELBO on $\mathcal{D}$ means maximizing $\log p_\theta(\mathcal{D})$ which is the maximum likelihood estimator of weights θ *and* also minimizing $KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{z}|\boldsymbol{x}))$ which enforces the approximated posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to be similar to the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$.

It can be shown [121] that the ELBO can be rewritten as

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - KL(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \,||\, p_{\mathcal{Z}}(\boldsymbol{z})). \tag{4.9}$$

The first term in (4.9) is a *reconstruction loss* similar to the one of plain autoencoders: it enforces that the code $\boldsymbol{z} \sim q_\phi(\cdot|\boldsymbol{x})$ generated by the encoder q_ϕ can be used by the decoder p_θ to reconstruct the original input $\boldsymbol{x}$. The second term is

a *regularization term* that enforces the distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ of the latent code $\boldsymbol{z}$ (given $\boldsymbol{x}$) to be close to the prior distribution $p_{\mathcal{Z}}(\boldsymbol{z})$. It is common to choose an isotropic Gaussian as the prior distribution of the latent code:

$$p_{\mathcal{Z}}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z} \mid 0, I) \propto e^{-\|\boldsymbol{z}\|^2/2}$$

and a Gaussian encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z} \mid \mu_\phi(\boldsymbol{x}), \Sigma_\phi(\boldsymbol{x}))$, so that the KL divergence in (4.9) is straightforward to compute. For the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ a Gaussian decoder is the most common choice and as we will see we benefit from that.

4.2.1 Learning approximations vs. encoder approximations

In this work we construct an image prior using a Variational Autoencoder (VAE). Like any machine learning tool VAEs make different kinds of approximations. Let's distinguish two types of approximations that shall be important in the sequel:

Learning approximation: The ideal prior $p^*_{\mathcal{X}}$ can only be approximated by our VAE due to its architectural constraints, finite complexity, truncated optimization algorithms, finite amount of data and possible biases in the data. Due to all these approximations, after learning we have only access to an approximate prior $p_{\mathcal{X}} \approx p^*_{\mathcal{X}}$. VAEs give access to this approximate prior $p_{\mathcal{X}}$ via a generative model: taking samples of a latent variable $\boldsymbol{z}$ with known distribution $\mathcal{N}(0, I)$ in $\mathbb{R}^p$ (with $p \ll d$), and feeding these samples through a learned decoder network, we obtain samples of $\boldsymbol{x} \sim p_{\mathcal{X}}$. The approximate prior itself

$$p_{\mathcal{X}}(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})\, d\boldsymbol{z} \tag{4.10}$$

is intractable because it requires computing an integral over all possible latent codes $\boldsymbol{z}$. However the approximate joint distribution is readily accessible

$$p_{\mathcal{X},\mathcal{Z}}(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})$$

thanks to $p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})$ which is provided by the decoder network.

Encoder approximation: In the previous item we considered the VAE as a generative model without making use of the encoder network. The encoder network

$$\tilde{p}_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x}) := q_\phi(\boldsymbol{z}|\boldsymbol{x}) \approx p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$$

is introduced as an approximate way to solve the intractability of $p_{Z|X}(\boldsymbol{z} \mid \boldsymbol{x}) = p_\theta(\boldsymbol{z}|\boldsymbol{x})$ (which is related to the intractability of $p_\theta(\boldsymbol{x})$ as observed in equation (4.7)).

Using the encoder network we can provide an alternative approximation for the joint distribution

$$\tilde{p}_{X,Z}(\boldsymbol{x}, \boldsymbol{z}) := q_\phi(\boldsymbol{z}|\boldsymbol{x})\, p_{\mathcal{X}}(\boldsymbol{x}) \approx p_{\mathcal{X},\mathcal{Z}}(\boldsymbol{x}, \boldsymbol{z})$$

which shall be useful in the sequel.

Put another way, the ideal joint distribution $p^*_{\mathcal{X},\mathcal{Z}}$ is inaccessible, but can be approximated in two different ways:

The first expression denoted $p_{\mathcal{X},\mathcal{Z}}(\boldsymbol{x},\boldsymbol{z})$ only uses the decoder and is only affected by the *learning approximation*

$$p^*_{X,Z}(\boldsymbol{x},\boldsymbol{z}) \approx p_{\mathcal{X},\mathcal{Z}}(\boldsymbol{x},\boldsymbol{z}) := p_\theta(\boldsymbol{x}|\boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})\,.$$

The second expression denoted $\tilde{p}_{X,Z}(\boldsymbol{x},\boldsymbol{z})$ uses both encoder and decoder and is affected both by the *learning approximation* and by the *encoder approximation*

$$p_{\mathcal{X},\mathcal{Z}}(\boldsymbol{x},\boldsymbol{z}) \approx \tilde{p}_{X,Z}(\boldsymbol{x},\boldsymbol{z}) := q_\phi(\boldsymbol{z}|\boldsymbol{x})\, p_{\mathcal{X}}(\boldsymbol{x})$$

In the following subsection we shall forget about the ideal prior p^*_X and joint distribution $p^*_{X,Z}$ which are both inaccessible. Instead we accept $p_{\mathcal{X}}$ (with its learning approximations) as our prior model which shall guide all our estimations. The approximation symbol shall be reserved to expressions that are affected by the encoder approximation *in addition to* the learning approximation.

4.2.2 Variational Autoencoders as Image Priors

To obtain the Maximum a Posteriori estimator (MAP), we could plug in the approximate prior $p_{\mathcal{X}}$ in equation (4.2), but this leads to a numerically difficult problem to solve due to the intractability of $p_{\mathcal{X}}$. Instead, we propose to maximize the joint posterior $p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}(\boldsymbol{x},\boldsymbol{z}\,|\,\boldsymbol{y})$ over $(\boldsymbol{x},\boldsymbol{z})$ which is equivalent to minimizing

$$\begin{aligned} J_1(\boldsymbol{x},\boldsymbol{z}) &:= -\log p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}(\boldsymbol{x},\boldsymbol{z}\,|\,\boldsymbol{y}) \\ &= -\log p_{\mathcal{Y}|\mathcal{X},\mathcal{Z}}(\boldsymbol{y}\,|\,\boldsymbol{x},\boldsymbol{z})\, p_\theta(\boldsymbol{x}\,|\,\boldsymbol{z}) p_{\mathcal{Z}}(\boldsymbol{z}) \\ &= F(\boldsymbol{x},\boldsymbol{y}) + H_\theta(\boldsymbol{x},\boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2. \end{aligned} \tag{4.11}$$

Note that the first term is quadratic in $\boldsymbol{x}$, the third term is quadratic in $\boldsymbol{z}$ and all the difficulty lies in the coupling term $H_\theta(\boldsymbol{x},\boldsymbol{z}) = -\log p_\theta(\boldsymbol{x}\,|\,\boldsymbol{z})$. For Gaussian decoders [121], the latter can be written as

$$\begin{aligned} H_\theta(\boldsymbol{x},\boldsymbol{z}) = \frac{1}{2}\Big(& d\log(2\pi) + \log\det\boldsymbol{\Sigma}_\theta(\boldsymbol{z}) \\ & + \|\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x}-\boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2\Big). \end{aligned} \tag{4.12}$$

which is also convex in $\boldsymbol{x}$. Hence, minimization with respect to $\boldsymbol{x}$ takes the convenient closed form:

$$\begin{aligned} \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x},\boldsymbol{z}) = & \left(\boldsymbol{A}^T\boldsymbol{A} + \sigma^2\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{z})\right)^{-1} \\ & \times \left(\boldsymbol{A}^T\boldsymbol{y} + \sigma^2\boldsymbol{\Sigma}_\theta^{-1}(\boldsymbol{z})\boldsymbol{\mu}_\theta(\boldsymbol{z})\right). \end{aligned} \tag{4.13}$$

Unfortunately the coupling term H and hence J_1 is a priori non-convex in $\boldsymbol{z}$. As a consequence the $\boldsymbol{z}$-minimization problem

$$\arg\min_{\boldsymbol{z}} J_1(\boldsymbol{x}, \boldsymbol{z}) \tag{4.14}$$

is a priori more difficult. However, for Gaussian encoders, VAEs provide an approximate expression for this coupling term which is quadratic in $\boldsymbol{z}$. Indeed, given the equivalence

$$\begin{aligned} p_\theta(\boldsymbol{x} \,|\, \boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z}) &= p_{\mathcal{X},\mathcal{Z}}(\boldsymbol{x}, \boldsymbol{z}) \\ &= p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \,|\, \boldsymbol{x})\; p_{\mathcal{X}}(\boldsymbol{x}) \\ &\approx q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x})\, p_{\mathcal{X}}(\boldsymbol{x}) \end{aligned} \tag{4.15}$$

we have that

$$H_\theta(\boldsymbol{x}, \boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2 \approx K_\phi(\boldsymbol{x}, \boldsymbol{z}) - \log p_{\mathcal{X}}(\boldsymbol{x})\,. \tag{4.16}$$

where $K_\phi(\boldsymbol{x}, \boldsymbol{z}) = -\log q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x})$. Therefore, this new coupling term becomes

$$\begin{aligned} K_\phi(\boldsymbol{x}, \boldsymbol{z}) &= -\log \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\Sigma}_\phi(\boldsymbol{x})) \\ &= \frac{1}{2}\big[p\log(2\pi) + \log\det \boldsymbol{\Sigma}_\phi(\boldsymbol{x}) \\ &\quad + \|\boldsymbol{\Sigma}_\phi^{-1/2}(\boldsymbol{x})(\boldsymbol{z} - \boldsymbol{\mu}_\phi(\boldsymbol{x}))\|^2\big], \end{aligned}$$

which is quadratic in $\boldsymbol{z}$. This provides an approximate expression for the energy (4.11) that we want to minimize, namely

$$J_2(\boldsymbol{x}, \boldsymbol{z}) := F(\boldsymbol{x}, \boldsymbol{y}) + K_\phi(\boldsymbol{x}, \boldsymbol{z}) - \log p_{\mathcal{X}}(\boldsymbol{x}) \approx J_1(\boldsymbol{x}, \boldsymbol{z}). \tag{4.17}$$

This approximate functional is quadratic in $\boldsymbol{z}$, and minimization with respect to this variable yields

$$\arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}). \tag{4.18}$$

In the case of linear VAEs,

$$p_{\mathcal{Z}}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; 0, I) \tag{4.19}$$

$$p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \,|\, \boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; V_\theta \boldsymbol{z} + v_\theta, \boldsymbol{\Sigma}_\theta). \tag{4.20}$$

it is easily shown that the posterior is also Gaussian [143], namely

$$p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \,|\, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; M V_\theta^T (x - v_\theta), \boldsymbol{\Sigma}_\theta M) \qquad \text{where } M = (V_\theta^T V_\theta + \boldsymbol{\Sigma}_\theta)^{-1}. \tag{4.21}$$

Hence, the linear encoder $q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x})$ that minimizes the ELBO is that of equation (4.21) so the approximation (4.15) is exact and then $J_1 = J_2$.

4.2.3 Alternate Joint Posterior Maximization

The previous observations suggest to adopt an alternate scheme to minimize the joint posterior $-\log p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{y})$ in order to solve the inverse problem. We begin our presentation by a simple version of the proposed algorithm, which aims at managing the case where the approximation of J_1 by J_2 is exact (at least in the sense given in Assumption 1 below); then we propose an adaptation for the more realistic non-exact case and we explore its convergence properties.

When $J_1 = J_2$ is a bi-convex function, Algorithm 3 is known as *Alternate Convex Search*. Its behavior has been studied in [89, 2]. Here we shall consider the following (strong) assumption, which includes the strictly bi-convex case ($J_1 = J_2$):

Assumption 1. *For any $\boldsymbol{x}$, if $\boldsymbol{z}^*$ is a global minimizer of $J_2(\boldsymbol{x}, \cdot)$, then $\boldsymbol{z}^*$ is a global minimizer of $J_1(\boldsymbol{x}, \cdot)$.*

The proposed alternate minimization takes the simple and fast form depicted in Algorithm 3, which can be shown to converge to a stationary point of J_1 under Assumptions 1 and 2, as stated in Proposition 2 below. Note that the minimization in step 2 of Algorithm 3 does not require the knowledge of the unknown term $-\log p_{\mathcal{X}}(\boldsymbol{x})$ in Equation (4.17) since it does not depend on $\boldsymbol{z}$.

Algorithm 3: Joint posterior maximization - exact case

Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial condition $\boldsymbol{x}_0$
Ensure: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{x},\boldsymbol{z}} p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{y})$
1: **for** $n := 0$ **to** maxiter **do**
2: $\quad \boldsymbol{z}_{n+1} := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$ // Quadratic approx
3: $\quad \boldsymbol{x}_{n+1} := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}_{n+1})$ // Quadratic
4: **end for**
5: **return** $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

The convergence analysis of the proposed schemes requires some general assumptions on the functions J_1 and J_2 :

Assumption 2. *$J_1(\cdot, \boldsymbol{z})$ is convex and admits a unique minimizer for any $\boldsymbol{z}$. Moreover, J_1 is coercive and continuously differentiable.*

The unicity of the minimizers of the partial function $J_1(\cdot, \boldsymbol{z})$ can be dropped. In this case, the proof of the convergence of Algorithm 3 has to be slightly adapted.

The convergence property of Algorithm 3 will be investigated in a wider framework below (Proposition 2). Note that all the properties required in Assumption 2 are satisfied if we use a differentiable activation function like the Exponential

Linear Unit (ELU) [46] with $\alpha = 1$, instead of the more common ReLU activation function. More details can be found in Appendix 4.5.1.

4.2.4 Approximate Alternate Joint Posterior Maximization

When the autoencoder approximation in (4.17) is not exact (Assumption 1), the energy we want to minimize in Algorithm 3, namely J_1 may not decrease. To ensure the decay, some additional steps can be added. Noting that the approximation provided by J_2 provides a fast and accurate heuristic to initialize the minimization of J_1, an alternative scheme is proposed in Algorithm 4.

Algorithm 4: Joint posterior maximization - approximate case

Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial conditions $\boldsymbol{x}_0, \boldsymbol{z}_0$

Ensure: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{x},\boldsymbol{z}} p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y}\right)$

for $n := 0$ **to** maxiter **do**
  $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$ // Equation (4.18)
  $\boldsymbol{z}^2 := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}^1$
  $\boldsymbol{z}^3 := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}_n$
  for $i := 1$ **to** 3 **do**
    $\boldsymbol{x}^i := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^i)$ // Equation (4.13)
  end for
  $i^* := \arg\min_{i\in\{1,2,3\}} J_1(\boldsymbol{x}^i, \boldsymbol{z}^i)$
  $(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) := (\boldsymbol{x}^{i^*}, \boldsymbol{z}^{i^*})$
end for
return $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

In Algorithm 4, GD is a gradient descent scheme such that for any starting point $\boldsymbol{z}_0$, the output $\boldsymbol{z}^+$ satisfies

$$\frac{\partial J_1}{\partial \boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{z}^+) = 0 \quad \text{and} \quad J_1(\boldsymbol{x}, \boldsymbol{z}^+) \leq J_1(\boldsymbol{x}, \boldsymbol{z}_0)$$

Hence, one can consider for instance a gradient descent scheme which finds a local minimizer of $J_1(\boldsymbol{x}, \cdot)$ starting from $\boldsymbol{z}_0$.

Our experiments with Algorithm 4 (Section 4.3.5) show that during the first few iterations (where the approximation provided by J_2 is good enough) $\boldsymbol{z}^1$ and $\boldsymbol{z}^2$ reach convergence faster than $\boldsymbol{z}^3$. After a critical number of iterations the opposite is true (the initialization provided by the previous iteration is better than the J_2 approximation, and $\boldsymbol{z}^3$ converges faster).

These observations suggest that a faster execution, with the same convergence properties, can be achieved by the variant in Algorithm 5, which avoids the costly

computation of $\boldsymbol{z}^2$ and $\boldsymbol{z}^3$ when unnecessary. Hence, in practice, we will use Algorithm 5 rather than Algorithm 4. However, Algorithm 4 provides a useful tool for diagnostics. Indeed, the comparison of the evaluation of $J_1(\boldsymbol{x}^i, \boldsymbol{z}^i)$ for $i = 1, 2, 3$ performed in step 8 permits to assess the evolution of the approximation of J_1 by J_2.

Algorithm 5 is still quite fast when J_2 provides a sufficiently good approximation, since in that case the algorithm chooses $i^* = 1$, and avoids any call to the iterative gradient descent algorithm. Even if we cannot give a precise definition of what *sufficiently good* means, the sample comparison of K_ϕ and H_θ as functions of $\boldsymbol{z}$, displayed in Figure 4.2(a), shows that the approximation is fair enough in the sense that it preserves the global structure of J_1. The same behavior was observed for a large number of random tests.

Note that Algorithm 3 is a particular instance of Algorithm 5 in the case where Assumption 1 holds, and $n_1 = n_2 = 0$ and if grad descent gives a global minimizer of the considered function (in this case, the computation of $\boldsymbol{z}^1$, $\boldsymbol{z}^2$, are skipped and only $\boldsymbol{z}^3$ is computed).

Proposition 2 (Convergence of Algorithm 5)**.** *Let $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ be a sequence generated by Algorithm 5. Under Assumption 2 we have that:*

1. *The sequence $\{J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ converges monotonically when $n \to \infty$.*
2. *The sequence $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ has at least one accumulation point.*
3. *All accumulation points of $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ are stationary points of J_1 and they all have the same function value.*

Proof. Since we are interested in the behaviour for $n \to \infty$, we assume $n > n_2$ in Algorithm 5.

1. Since $n > n_2$ the algorithm chooses $i^* = 3$ and $\boldsymbol{z}_{n+1} = \boldsymbol{z}^3$. According to the definition of grad descent, one has

$$J_1(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

and by optimality one has

$$J_1(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}).$$

Hence, since J_1 is coercive (thus, lowerbounded), Statement 1 is straightforward.

2. Thanks to the coercivity of J_1, the sequences $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$ and $\{(\boldsymbol{x}_n, \boldsymbol{z}_{n+1})\}$ are bounded, thus admit an accumulation point.

Algorithm 5: Joint posterior maximization - approximate case (faster version)

Require: Measurements $\boldsymbol{y}$, Autoencoder parameters θ, ϕ, Initial condition $\boldsymbol{x}_0$, iterations $n_1 \leq n_2 \leq n_{\max}$

Ensure: $\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{x},\boldsymbol{z}} p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y}\right)$

for $n := 0$ **to** $n_{\max}$ **do**
  done := FALSE
  if $n < n_1$ **then**
    $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$ // Equation (4.18)
    $\boldsymbol{x}^1 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^1)$ // Equation (4.13)
    if $J_1(\boldsymbol{x}^1, \boldsymbol{z}^1) < J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$ **then**
      $i^* := 1$ // J_2 is good enough
      done := TRUE
    end if
  end if
  if not done **and** $n < n_2$ **then**
    $\boldsymbol{z}^1 := \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_n, \boldsymbol{z})$
    $\boldsymbol{z}^2 := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}^1$
    $\boldsymbol{x}^2 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^2)$ // Equation (4.13)
    if $J_1(\boldsymbol{x}^2, \boldsymbol{z}^2) < J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$ **then**
      $i^* := 2$ // J_2 init is good enough
      done := TRUE
    end if
  end if
  if not done **then**
    $\boldsymbol{z}^3 := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_n, \boldsymbol{z})$, starting from $\boldsymbol{z} = \boldsymbol{z}_n$
    $\boldsymbol{x}^3 := \arg\min_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}^3)$ // Equation (4.13)
    $i^* := 3$
  end if
  $(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) := (\boldsymbol{x}^{i^*}, \boldsymbol{z}^{i^*})$
end for
return $\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}$

3. Using Fermat's rule and the definition of grad descent, one has

$$\frac{\partial J_1}{\partial \boldsymbol{z}}(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}) = 0 \quad \text{and} \quad \frac{\partial J_1}{\partial \boldsymbol{x}}(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) = 0.$$

Let $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ be an accumulation point of $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}$. By double extraction, one can find two subsequences such that

$$(\boldsymbol{x}_{n_j+1}, \boldsymbol{z}_{n_j+1}) \to (\boldsymbol{x}^*, \boldsymbol{z}^*) \text{ and } (\boldsymbol{x}_{n_j}, \boldsymbol{z}_{n_j+1}) \to (\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*)$$

By continuity of ∇J_1, one gets that

$$\frac{\partial J_1}{\partial \boldsymbol{z}}(\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*) = 0 \quad \text{and} \quad \frac{\partial J_1}{\partial \boldsymbol{x}}(\boldsymbol{x}^*, \boldsymbol{z}^*) = 0$$

In particular, the convexity of $J_1(\cdot, \boldsymbol{z}^*)$ and Assumption 2 ensure that $\boldsymbol{x}^*$ is a global minimizer of $J_1(\cdot, \boldsymbol{z}^*)$. Besides, the inequalities proved in Point 1 above show that

$$J_1(\boldsymbol{x}^*, \boldsymbol{z}^*) = J_1(\hat{\boldsymbol{x}}^*, \boldsymbol{z}^*) = \lim_{n\to\infty} J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

that is, $\hat{\boldsymbol{x}}^*$ is also a global minimizer of $J_1(\cdot, \boldsymbol{z}^*)$. Since $J_1(\cdot, \boldsymbol{z}^*)$ has a unique minimizer, one has $\hat{\boldsymbol{x}}^* = \boldsymbol{x}^*$, and

$$\frac{\partial J_1}{\partial \boldsymbol{z}}(\boldsymbol{x}^*, \boldsymbol{z}^*) = 0$$

namely $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ is a stationary point of J_1. Note that we have also proved that $\boldsymbol{x}_{n_j}$ and $\boldsymbol{x}_{n_j+1}$ have same limit. □

Note that if $n_1 = n_2 = \infty$ we cannot assume that $i^* = 3$. In that case statements 1 and 2 are still valid but the third statement is not. The reason is that for $i^* \in \{1, 2\}$ we cannot guarantee the chain of inequalities

$$J_1(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_n)$$

but only

$$J_1(\boldsymbol{x}_{n+1}, \boldsymbol{z}_{n+1}) \leq J_1(\boldsymbol{x}_n, \boldsymbol{z}_n).$$

This is consistent with the design of the algorithm where iterations $n < n_2$ serve as an heuristic to guide the algorithm to a sensible critical point. However, convergence to a critical point is only guaranteed by the final iterations $n > n_2$.

4.2.5 MAP-z as the limit case for $\beta \to \infty$

If one wishes to compute the MAP-$\boldsymbol{z}$ estimator instead of the joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ from the previous section, one has two options:

1. Use your favorite gradient descent algorithm to solve equation (4.3).

2. Use Algorithm 5 to solve a series of joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ problems with increasing values of $\beta \to \infty$ as suggested in Algorithm 2.

In the experimental section we show that the second approach most often leads to a better optimum.

In practice, in order to provide a stopping criterion for Algorithm 2 and to make a sensible choice of β-values we reformulate Algorithm 2 as a constrained optimization problem

$$\underset{\boldsymbol{x},\boldsymbol{z}\,:\,\|\mathsf{D}(\boldsymbol{z})-\boldsymbol{x}\|^2\leq\varepsilon}{\arg\min} F(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2.$$

The corresponding Lagrangian form is

$$\max_{\beta}\min_{\boldsymbol{x},\boldsymbol{z}} F(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2 + \beta\left(\|\mathsf{D}(\boldsymbol{z})-\boldsymbol{x}\|^2-\varepsilon\right)^+ \tag{4.22}$$

and we use the exponential multiplier method [222] to guide the search for the optimal value of β (see Algorithm 6)

Algorithm 6: MAP-$\boldsymbol{z}$ as the limit of joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$.

Require: Measurements $\boldsymbol{y}$, Tolerance ε, Rate $\rho > 0$, Initial β_0, Initial $\boldsymbol{x}_0$, Iterations $0 \leq n_1 \leq n_2 \leq n_{\max}$

Ensure: $\arg\min_{\boldsymbol{x},\,\boldsymbol{z}\,:\,\|\mathsf{D}(\boldsymbol{z})-\boldsymbol{x}\|^2\leq\varepsilon} F(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2$.

$\beta := \beta_0$
$\boldsymbol{x}^0, \boldsymbol{z}^0 :=$ Algorithm 5 starting from $\boldsymbol{x} = \boldsymbol{x}_0$ with $\beta, n_1,\ n_2,\ n_{\max}$.
converged := FALSE
$k := 0$
while not converged **do**
  $\boldsymbol{x}^{k+1}, \boldsymbol{z}^{k+1} :=$ Algorithm 5 starting from $\boldsymbol{x} = \boldsymbol{x}^k$ with β and $n_1 = n_2 = 0$
  $C = \|\mathsf{D}(\boldsymbol{z}^{k+1}) - \boldsymbol{x}^{k+1}\|^2 - \varepsilon$
  $\beta := \beta\exp(\rho C)$
  converged := $(C \leq 0)$
  $k := k + 1$
end while
return $\boldsymbol{x}^k, \boldsymbol{z}^k$

4.3 Experimental results

4.3.1 Baseline algorithms

To validate our approach, we perform comparisons on several inverse problems with the following algorithms:

- CSGM (Bora et al. [24]) directly computes the $\boldsymbol{z}-$ MAP estimator as defined in Equation (4.3) using gradient descent. We run CSGM using the decoder of a VAE as generator D starting at random $\boldsymbol{z}_0$. In addition, as Bora et al. note that random restarts are important for good performance, we also compute the best result (as measured by (4.3)) among $m = 10$ different random initializations $\boldsymbol{z}_0$ and refer to this variant as mCSGM.
- PULSE [151] is very similar to CSGM but restricts the search of the latent code $\boldsymbol{z}$ to the sphere of radius $\sqrt{p}$, arguing that it concentrates most of the probability mass of a Gaussian distribution $\mathcal{N}(0, I)$ on a high-dimensional space $\mathbb{R}^p$.
- PGD-GAN [200] performs a projected gradient descent of $F(\boldsymbol{x}, \boldsymbol{y})$ wrt $\boldsymbol{x}$:

$$\begin{cases} \boldsymbol{w}_k = \boldsymbol{x}_k - \eta \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) \\ \boldsymbol{x}_{k+1} = \mathsf{D}(\arg\min_{\boldsymbol{z}} \|\boldsymbol{w}_k - \mathsf{D}(\boldsymbol{z})\|). \end{cases} \tag{4.23}$$

- In addition, we implement the splitting method of Algorithm 2 which is a simple continuation scheme for the $\boldsymbol{z}-$ MAP estimator of Equation (4.3).

For a fair comparison we run all algorithms on the same prior, *i.e.* the same generator network $\mathsf{D} = \boldsymbol{\mu}_\theta$ where $\boldsymbol{\mu}_\theta$ is the decoder mean from the VAE model that we trained for JPMAP.

4.3.2 Inverse problems

Here, we briefly describe the inverse problems $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \eta$, $\eta \sim \mathcal{N}(0, \sigma^2 I)$ to be considered for validating our approach:

- *Denoising*: $\boldsymbol{A} = I$ and σ large.
- *Compressed Sensing*: the sensing matrix $\boldsymbol{A} \in \mathbb{R}^{q \times d}$ has Gaussian random entries $\boldsymbol{A}_{ij} \sim \mathcal{N}(0, 1/q)$, where $q \ll d$ is the number of measurements.
- *Interpolation*: $\boldsymbol{A}$ is a diagonal matrix with random binary entries, so masking a percentage p of the image pixels.

- *(Non-blind) Deblurring*: $\boldsymbol{A}\boldsymbol{x} = h * \boldsymbol{x}$ where h is a known convolution kernel.
- *Super-resolution*: $\boldsymbol{A}$ is a downsampling/decimation operator of scaling factor s.

4.3.3 AutoEncoder and dataset

In order to test our joint prior maximization model we first train a Variational Autoencoder like in [121] on the training data of MNIST handwritten digits [133].

The *stochastic encoder* takes as input an image $\boldsymbol{x}$ of $28 \times 28 = 784$ pixels and produces as an output the mean and (diagonal) covariance matrix of the Gaussian distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, where the latent variable $\boldsymbol{z}$ has dimension 8. The architecture of the encoder is composed of 3 fully connected layers with ELU activations (to preserve continuous differentiability). The sizes of the layers are as follows: $784 \to 500 \to 500 \to (8+8)$. Note that the output is of size $8+8$ in order to encode the mean and diagonal covariance matrix, both of size 8.

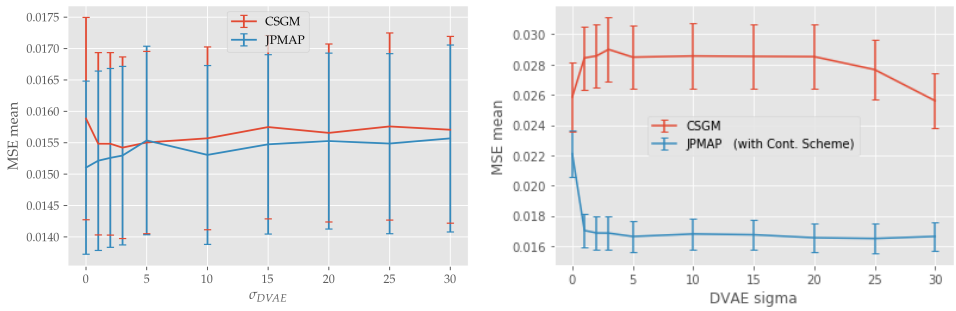
The *stochastic decoder* takes as an input the latent variable $\boldsymbol{z}$ and outputs the mean and covariance matrix of the Gaussian distribution $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. Following [51] we chose here an isotropic covariance $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}) = \gamma^2 I$ where $\gamma > 0$ is trained, but independent of $\boldsymbol{z}$. This choice simplifies the minimization problem (4.14), because the term $\det \Sigma_\theta(\boldsymbol{z})$ (being constant) has no effect on the $\boldsymbol{z}$-minimization. The architecture of the decoder is also composed of 3 fully connected layers with ELU activations (to preserve continuous differentiability). The sizes of the layers are as follows: $8 \to 500 \to 500 \to 784$. Note that the covariance matrix is constant, so it does not augment the size of the output layer which is still $784 = 28 \times 28$ pixels.

We also trained a VAE on CelebA [141] images cropped to $64 \times 64 \times 3$, with latent dimension ranging from 64 to 512. We choose a DCGAN-like [174] CNN architecture as encoder and a symmetrical one as decoder with ELU activations, batch normalization and isotropic covariance as before. We train these architectures using PyTorch [166] with batch size 128 and Adam algorithm for 200 epochs with learning rate 0.0001 and rest of the parameters as default. For more details, see the code[3].

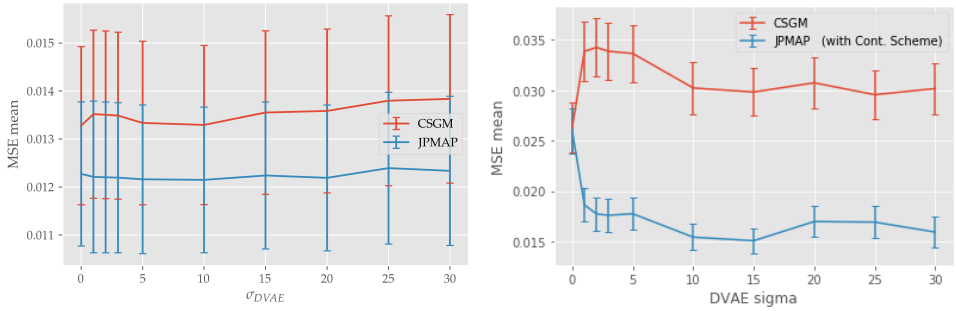
4.3.4 Need to train the VAE with a denoising criterion

It should be noted that when training our Variational Autoencoder we should be more careful than usual. Indeed in the most widespread applications of VAEs they are only used as a generative model or as a way to interpolate between images

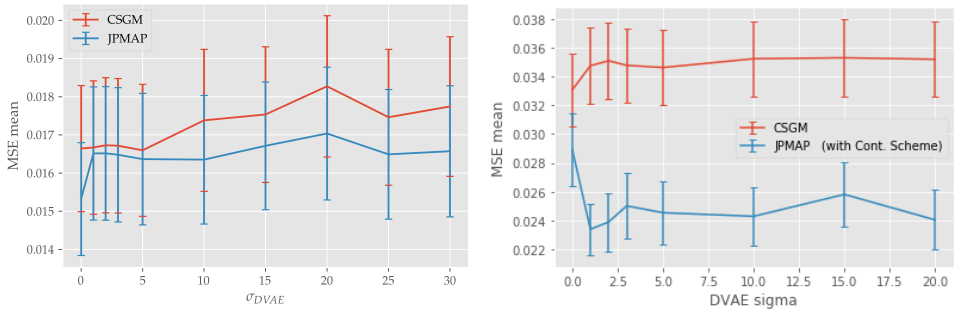
[3]Code available at https://github.com/mago876/JPMAP.



(a) Denoising



(b) Compressed Sensing



(c) Interpolation

Figure 4.1: Evaluating the quality of the generative model as a function of σ_{DVAE}. On (a) Denoising (Gaussian noise $\sigma = 150$), (b) Compressed Sensing ($\sim 10.2\%$ measurements, noise $\sigma = 10$) and (c) Interpolation (80% of missing pixels, noise $\sigma = 10$). Results of both algorithms are computed on a batch of 50 images and initialising on ground truth $\boldsymbol{x}^*$ (for CSGM we use $\boldsymbol{z}_0 = \boldsymbol{\mu}_\phi(\boldsymbol{x}^*)$). Without a denoising criterion $\sigma_{\text{DVAE}} = 0$ (*left*) the JPMAP algorithm may provide wrong guesses $\boldsymbol{z}^1$ when applying the encoder in step 2 of Algorithm 4. For $\sigma_{\text{DVAE}} > 0$ (*right*) however, the alternating minimization algorithm can benefit from the robust initialization heuristics provided by the encoder, and it consistently converges to a better local optimum than the simple gradient descent in CSGM.

that are close to $\mathcal{M}$, *i.e.* the image of the generator $\boldsymbol{\mu}_\theta$. For such applications it is sufficient to train the encoder $\boldsymbol{\mu}_\phi$, $\boldsymbol{\Sigma}_\phi$ on a training set that is restricted to $\mathcal{M}$.

In our case however, we need the encoder to provide sensible values even when its input $\boldsymbol{x}$ is quite far away from $\mathcal{M}$: the encoder has to actually fulfill two functions at the same time:

1. (Approximately) project $\boldsymbol{x}$ to its closest point in $\mathcal{M}$, and

2. compute the encoding of this projected value (which should be the same as the encoding of the original $\boldsymbol{x}$.

Traditional VAE training procedures do not ensure that the encoder generalizes well to $\boldsymbol{x} \notin \mathcal{M}$. In order to ensure this generalization ability we adopt the training procedure of the DVAE (Denoising VAE) proposed by [109], which consists in adding various realizations of zero-mean Gaussian noise of variance σ^2_{DVAE} to the samples $\boldsymbol{x}$ presented to the encoder, while still requiring the decoder to match the noiseless value, *i.e.* we optimize the parameters in such a way that

$$\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\tilde{\boldsymbol{x}})) \approx \boldsymbol{x} \tag{4.24}$$

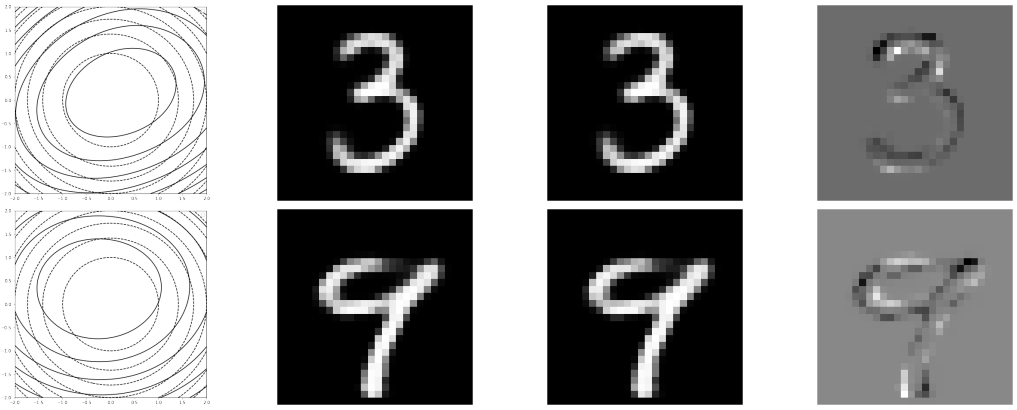
where $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \sigma_{\text{DVAE}}\varepsilon$ and $\varepsilon \sim \mathcal{N}(0, I)$ for all $\boldsymbol{x}$ in the training set and for many realizations of ε.

More specifically, if we take a corruption model $p(\tilde{\boldsymbol{x}} \,|\, \boldsymbol{x})$ like above, it can be shown [109] that

$$\tilde{\mathcal{L}}_{\theta,\phi}(\boldsymbol{x}) = \mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}\left[\mathbb{E}_{q_\phi(\boldsymbol{z}|\tilde{\boldsymbol{x}})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - KL(q_\phi(\boldsymbol{z}|\tilde{\boldsymbol{x}}) \,||\, p_{\mathcal{Z}}(\boldsymbol{z}))\right] \tag{4.25}$$

is an alternative ELBO of (4.9). In practice, using Monte Carlo for estimating the expectation $\mathbb{E}_{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})}$ in (4.25), we only need to add noise to $\boldsymbol{x}$ before passing it to the encoder q_ϕ during training, as mentioned in (4.24).

Our experiments with this denoising criterion confirm the observation by [109] that it does not degrade the quality of the generative model, as long as σ_{DVAE} is not too large (see Figure 4.1). As a side benefit, however, we obtain a more robust encoder that generalizes well for values of $\boldsymbol{x}$ that are not in $\mathcal{M}$ but within a neighbourhood of size $\approx \sigma_{\text{DVAE}}$ around $\mathcal{M}$. This side benefit, which was not the original intention of the DVAE training algorithm in [109] is nevertheless crucial for the success of our algorithm as demonstrated in Figure 4.1. The same figure shows that as long as $\sigma_{\text{DVAE}} \geq 5$ its value does not significantly affect the performance. In the sequel we use $\sigma_{\text{DVAE}} = 15$.



(a) Encoder approximation (b) Decoded exact optimum (c) Decoded approx. optimum (d) Difference (b)-(c)

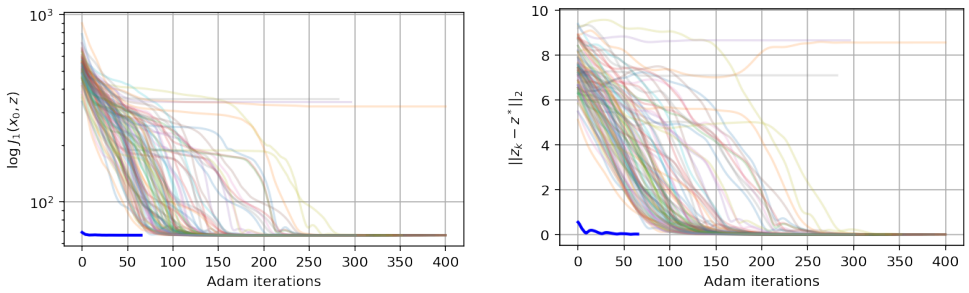
Figure 4.2: Encoder approximation: *(a)* Contour plots of $-\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2$ and $-\log q_\phi(\boldsymbol{z}|\boldsymbol{x})$ for a fixed $\boldsymbol{x}$ and for a random 2D subspace in the $\boldsymbol{z}$ domain (the plot shows $\pm 2\boldsymbol{\Sigma}_\phi^{1/2}$ around $\boldsymbol{\mu}_\phi$). Observe the relatively small gap between the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ and its variational approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. This figure shows some evidence of partial $\boldsymbol{z}$-convexity of J_1 around the minimum of J_2, but it does not show how far is $\boldsymbol{z}^1$ from $\boldsymbol{z}^2$. *(b)* Decoded exact optimum $\boldsymbol{x}_1 = \boldsymbol{\mu}_\theta\left(\arg\max_{\boldsymbol{z}} p_\theta(\boldsymbol{x}|\boldsymbol{z}) e^{\frac{1}{2}\|\boldsymbol{z}\|^2}\right)$. *(c)* Decoded approximate optimum $\boldsymbol{x}_2 = \boldsymbol{\mu}_\theta\left(\arg\max_{\boldsymbol{z}} q_\phi(\boldsymbol{z}|\boldsymbol{x})\right)$. *(d)* Difference betweeen (b) and (c).

4.3.5 Effectiveness of the encoder as a fast approximate minimizer

Proposition 2 shows that the proposed alternate minimization scheme in Algorithm 5 converges to a stationary point of J_1. And so does the gradient descent scheme in [24]. Since both algorithms have to deal with non-convex energies, they both risk converging to spurious local minima. Also both algorithms solve essentially the same model when the variance γ of the coupling term tends to zero.

If our algorithm shows better performance (see next subsection), it is mainly because it relies on a previously trained VAE in two fundamental ways: *(i)* to avoid getting trapped in spurious local minima and *(ii)* to accelerate performance during the initial iterations ($n < n_{\min}$). These two features are only possible if the autoencoder approximation is good enough and if the encoder is able to provide good initializations for the non-convex $\boldsymbol{z}$- optimization subproblem in line 13 of Algorithm 5.

Figures 4.2 and 4.3 illustrate these two properties of our VAE. We do so by selecting a random $\boldsymbol{x}_0$ from MNIST test set and by computing $\boldsymbol{z}^*(\boldsymbol{z}_0) := \text{GD}_{\boldsymbol{z}}\, J_1(\boldsymbol{x}_0, \boldsymbol{z})$ with different initial values $\boldsymbol{z}_0$. These experiments were performed using the ADAM minimization algorithm with learning rate equal to 0.01. Figure 4.3(a) shows that



(a) Energy evolution, initializing with $\mathcal{N}(0, I)$.

(b) Distance to the optimum at each iteration of (a).

Figure 4.3: *Effectiveness of the encoder approximation:* We take $\boldsymbol{x}_0$ from the *test* set of MNIST and minimize $J_1(\boldsymbol{x}_0, \boldsymbol{z})$ with respect to $\boldsymbol{z}$ using gradient descent from random Gaussian initializations $\boldsymbol{z}_0$. The blue thick curve represents the trajectory if we initialize at the encoder approximation $\boldsymbol{z}^1 = \arg\min_{\boldsymbol{z}} J_2(\boldsymbol{x}_0, \boldsymbol{z}) = \boldsymbol{\mu}_\phi(\boldsymbol{x}_0)$. *(a)*: Plots of the energy iterates $J_1(\boldsymbol{x}_0, \boldsymbol{z}_k)$. *(b)*: ℓ^2 distances of each trajectory with respect to the global optimum $\boldsymbol{z}^*$. *Conclusion:* Observe that the encoder initialization allows much faster convergence both in energy and in $\boldsymbol{z}$, and it avoids the few random initializations that lead to a wrong stationary point different from the unique global minimizer.

$\boldsymbol{z}^*(\boldsymbol{z}_0)$ reaches the global optimum for most (but not all) initializations $\boldsymbol{z}_0$. Indeed from 200 random initializations $\boldsymbol{z}_0 \sim \mathcal{N}(0, I)$, 195 reach the same global minimum, whereas 5 get stuck at a higher energy value. However these 5 initial values have energy values $J_1(\boldsymbol{x}_0, \boldsymbol{z}_0) \gg J_1(\boldsymbol{x}_0, \boldsymbol{z}^1)$ far larger than those of the encoder initialization $\boldsymbol{z}^1 = \boldsymbol{\mu}_\phi(\boldsymbol{x}_0)$, and are thus never chosen by Algorithm 5. The encoder initialization $\boldsymbol{z}^1$ on the other hand provides much faster convergence to the global optimum.

In addition, this experiment shows that we cannot assume $\boldsymbol{z}$-convexity: The presence of plateaux in the trajectories of many random initializations as well as the fact that a few initializations do not lead to the global minimum indicates that J_1 may not be everywhere convex with respect to $\boldsymbol{z}$. However, in contrast to classical works on alternate convex search, our approach adopts weaker assumptions and does not require convexity on $\boldsymbol{z}$ to prove convergence in Proposition 2.

In Figure 4.3(b) we display the distances of each trajectory to the global optimum $\boldsymbol{z}^*$ (taken as the median over all initializations $\boldsymbol{z}_0$ of the final iterates $\boldsymbol{z}^*(\boldsymbol{z}_0)$); note that this optimum is always reached, which suggests that $\boldsymbol{z} \mapsto J_1(\boldsymbol{x}_0, \boldsymbol{z})$ has a unique global minimizer in this case. Finally, Figure 4.2 shows that the encoder approximation is quite good both in the latent space (Figure 4.2(a)) and in image space (Figures 4.2(b) and 4.2(c)). It also shows that the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is

pretty close to log-concave near the maximum of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$.

4.3.6 Image restoration experiments

Choice of $\boldsymbol{x}_0$: In the previous section, our validation experiments used a random $\boldsymbol{x}_0$ from the data set as initialization. When dealing with an image restoration problem, Algorithms 4 and 5 require an initial value of $\boldsymbol{x}_0$ to be chosen. In all experiments we choose this initial value as $\boldsymbol{A}^T\boldsymbol{y}$.

Choice of n_1 and n_2: After a few runs of Algorithm 4 we find that in most cases, during the first 10 or 20 iterations $\boldsymbol{z}^1$ decreases the energy with respect to the previous iteration, and this value depends on the inverse problems (for example, for denoising is smaller than for compressed sensing). But after at most 150 iterations the autoencoder approximation is no longer good enough and we need to perform gradient descent on $\boldsymbol{z}_n$ in order to further decrease the energy. Based on these findings we set $n_1 = 25$ and $n_2 = 150$ in Algorithm 5 for all experiments. Note that we could also choose $n_1 = n_2 = n_{\max}$, since in all our experiments we observed that the algorithm auto-regulates itself, evolving from $i^* = 1$ in the first few dozen iterations to $i^* = 3$ when it is close to convergence. Choosing a finite value for n_1 and n_2 is only needed to ensure that $i^* = 3$ when $n \to \infty$, which is a necessary condition to prove statement 3 of Proposition 2.

Figure 4.4 shows the evolution of $\boldsymbol{x}_k$ and $\mathsf{D}(\boldsymbol{z}_k)$ from Algorithm 6 in an interpolation example. Here we can see how the exponential multiplier method in Equation (4.22) updates the values of β_k to ensure $\|\mathsf{D}(\boldsymbol{z}_k) - \boldsymbol{x}_k\|^2 \leq \varepsilon$.

Figures 4.5 and 4.6 show the results of denoising, interpolation, compressed sensing, deblurring and super-resolution experiments on MNIST for different degradation levels using the proposed algorithm (JPMAP) and the baseline algorithms introduced in Section 4.3.1. The metrics used are PSNR and LPIPS[4] [246] mean $\pm$ its standard error computed over 100 random experiments for each problem. Figure 4.7 displays the images of 10 representative interpolation and deblurring experiments from the hundreds of experiments summarized in figures 4.5 and 4.6.

These results show that JPMAP outperforms all other baseline algorithms in terms of PSNR and LPIPS when random restarts are not allowed. When 10 random restarts are allowed for CSGM, but not for JPMAP, then both algorithms (JPMAP and mCSGM) show a similar global performance: JPMAP tends to provide a slightly better result than mCSGM except for the most extremely ill-posed interpolation, super-resolution and compressed sensing experiments (when available measurements are less than 10% the number of pixels). In that case mCSGM outperforms JPMAP by an equally small margin. The latter case can be explained by the fact that the encoder (which is used by JPMAP but not by CSGM) strug-

[4]MNIST images were zero-padded to 32×32 because LPIPS does not accept 28×28 images.

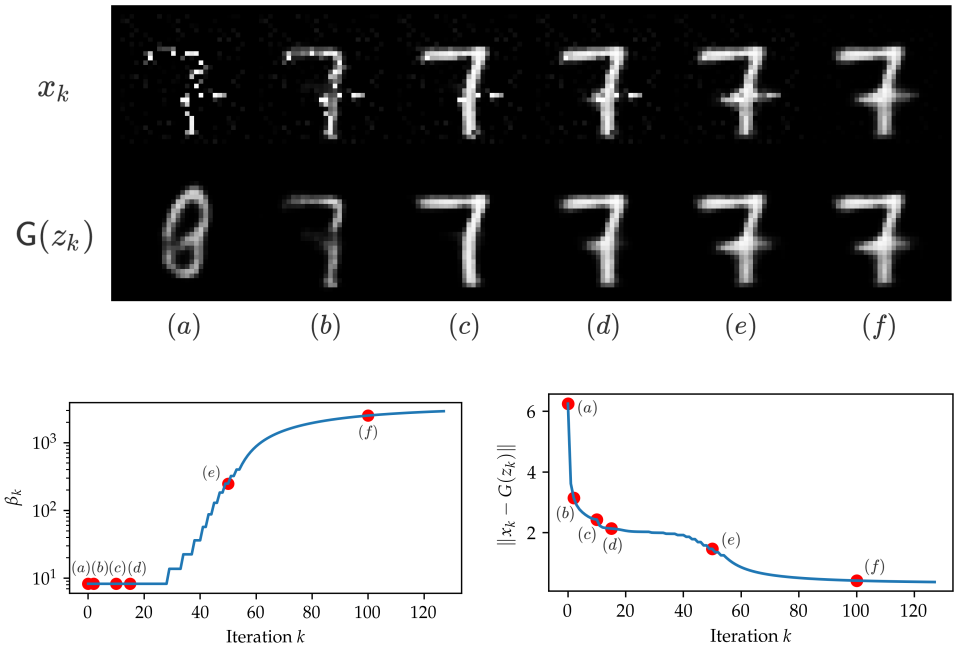
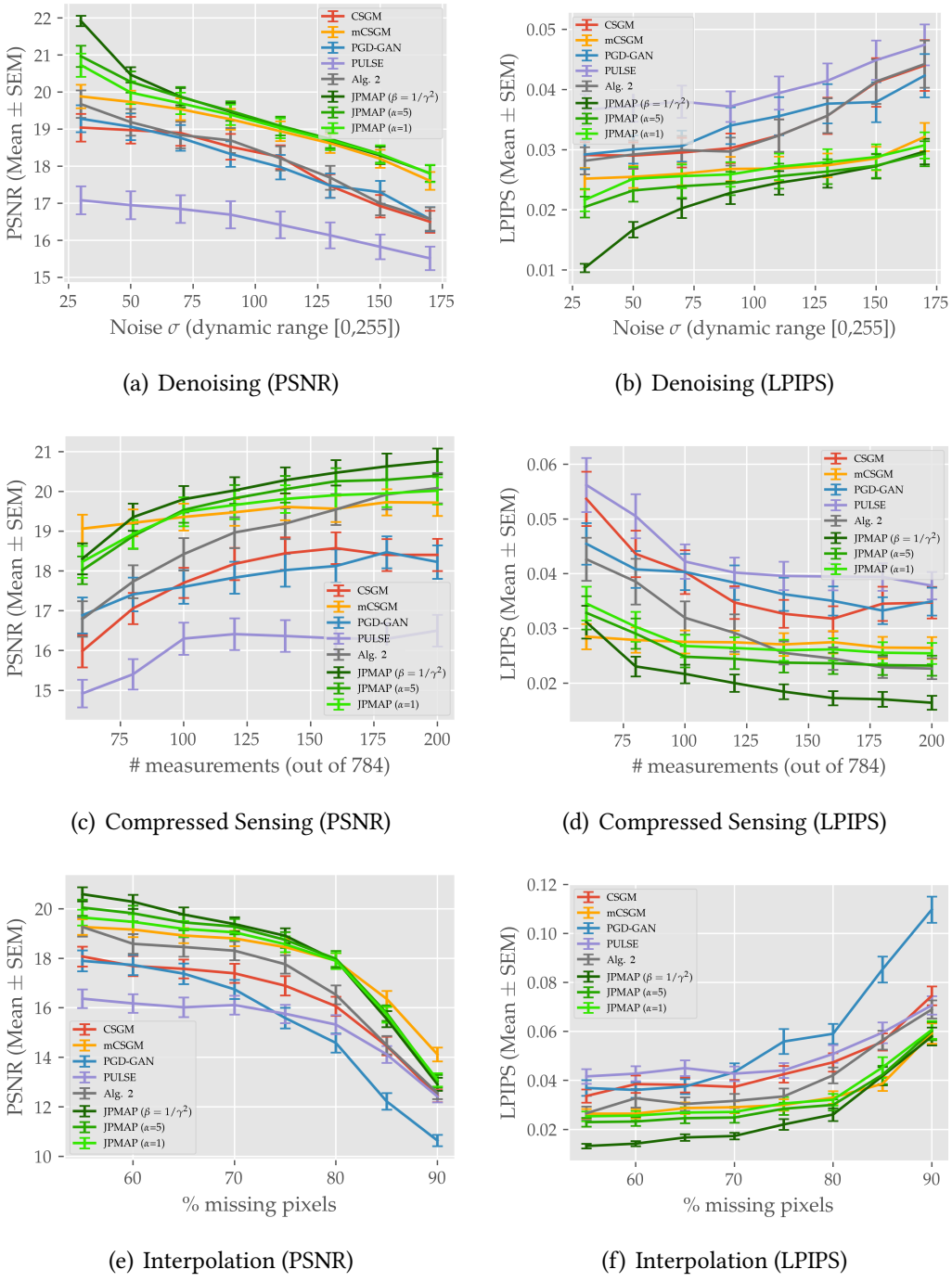


Figure 4.4: *Evolution of Algorithm 6.* In this interpolation example, JPMAP starts with the initialization in (a). During first iterations $(b) - (d)$ where β_k is small, $\boldsymbol{x}_k$ and $\mathsf{D}(\boldsymbol{z}_k)$ start loosely approaching each other at a coarse scale, and $\boldsymbol{x}_k$ only fills missing pixels with the ones of $\mathsf{D}(\boldsymbol{z}_k)$ (in particular the noise of $\boldsymbol{y}$ is still present). By increasing β_k in $(e) - (f)$ we enforce $\|\mathsf{D}(\boldsymbol{z}_k) - \boldsymbol{x}_k\|^2 \leq \varepsilon$. Here we set $\epsilon = \left(\frac{3}{255}\right)^2 d$, that is, MSE of 3 gray levels.



(a) Denoising (PSNR)

(b) Denoising (LPIPS)

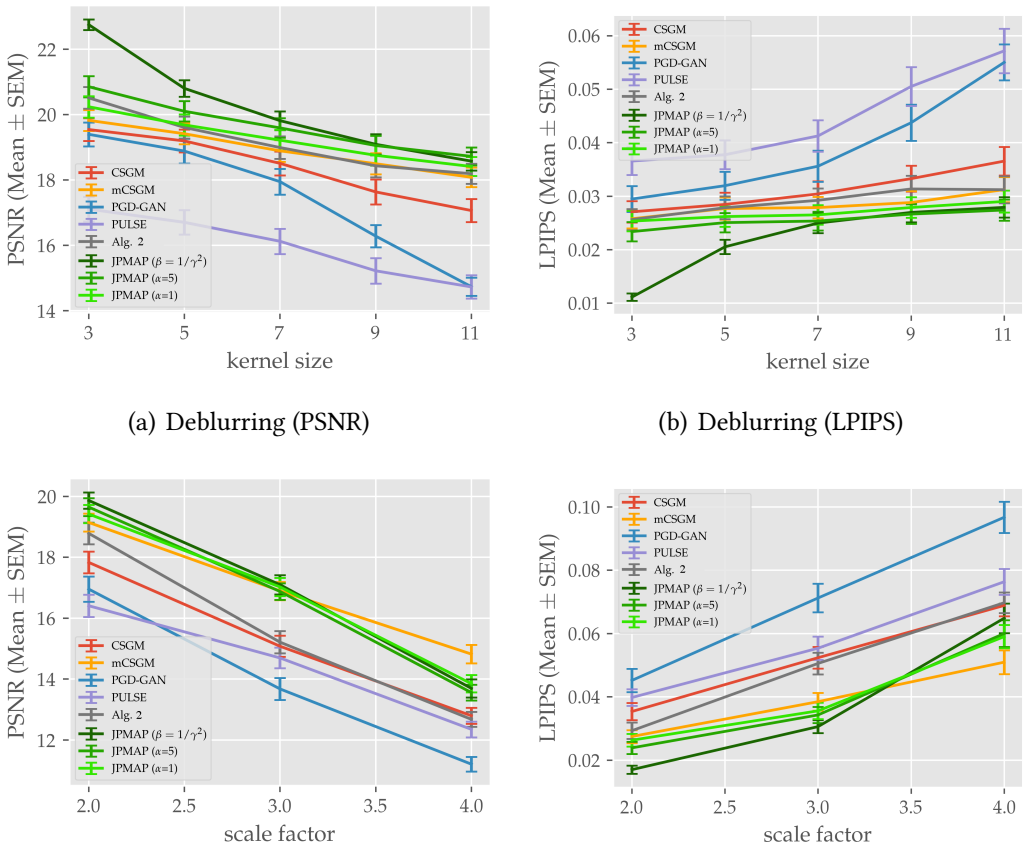
(c) Compressed Sensing (PSNR)

(d) Compressed Sensing (LPIPS)

(e) Interpolation (PSNR)

(f) Interpolation (LPIPS)

Figure 4.5: *Denoising, Compressed Sensing and Interpolation*: Evaluating the effectiveness of Algorithm 5 (fixed β) and Algorithm 6 for different values of $\epsilon = \left(\frac{\alpha}{255}\right)^2 n$, with $\sigma_{\text{DVAE}} = 15$ (metrics were computed on a batch of 100 test images). For PSNR, higher is better and for LPIPS, lower is better. For comparison we provide the results of the baselines introduced in Section 4.3.1 (namely, Algorithm 2, CSGM [24], mCSGM (CSGM with restarts), PGD-GAN [200] and PULSE [151].)



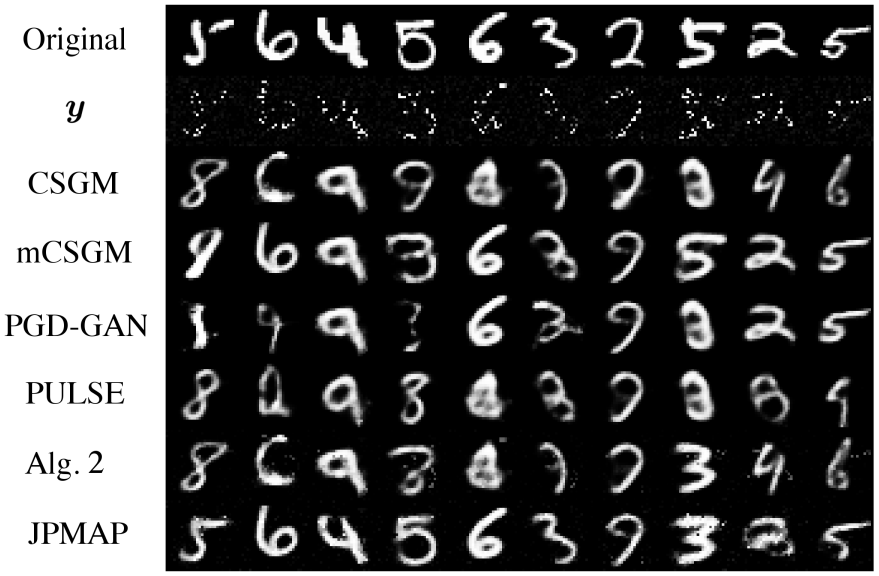
(a) Deblurring (PSNR)

(b) Deblurring (LPIPS)

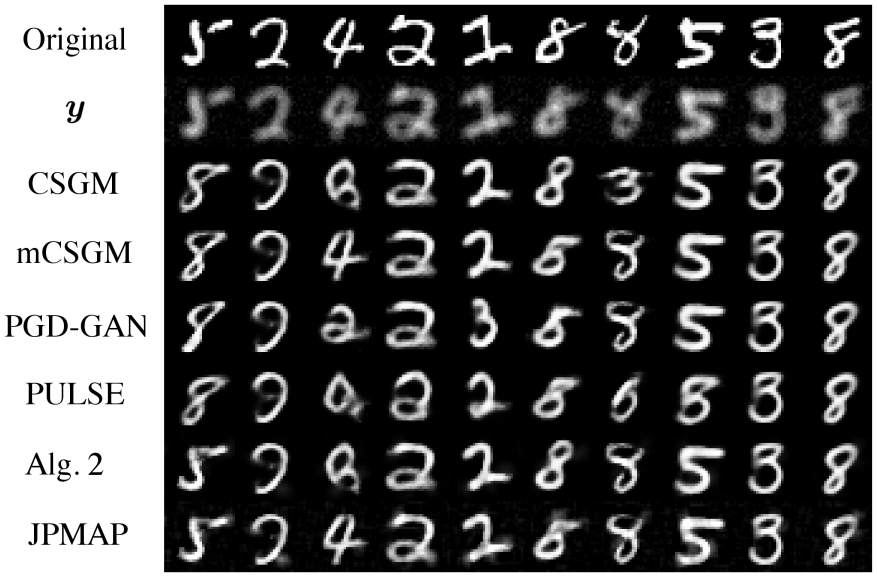
(c) Super-resolution (PSNR)

(d) Super-resolution (LPIPS)

Figure 4.6: *Deblurring and Super-resolution*: Evaluating the effectiveness of Algorithm 5 (fixed β) and Algorithm 6 for different values of $\epsilon = \left(\frac{\alpha}{255}\right)^2 n$, with $\sigma_{\text{DVAE}} = 15$ (metrics were computed on a batch of 100 test images). For PSNR, higher is better and for LPIPS, lower is better. For comparison we provide the results of the baselines introduced in Section 4.3.1 (namely, Algorithm 2, CSGM [24], mCSGM (CSGM with restarts), PGD-GAN [200] and PULSE [151].)

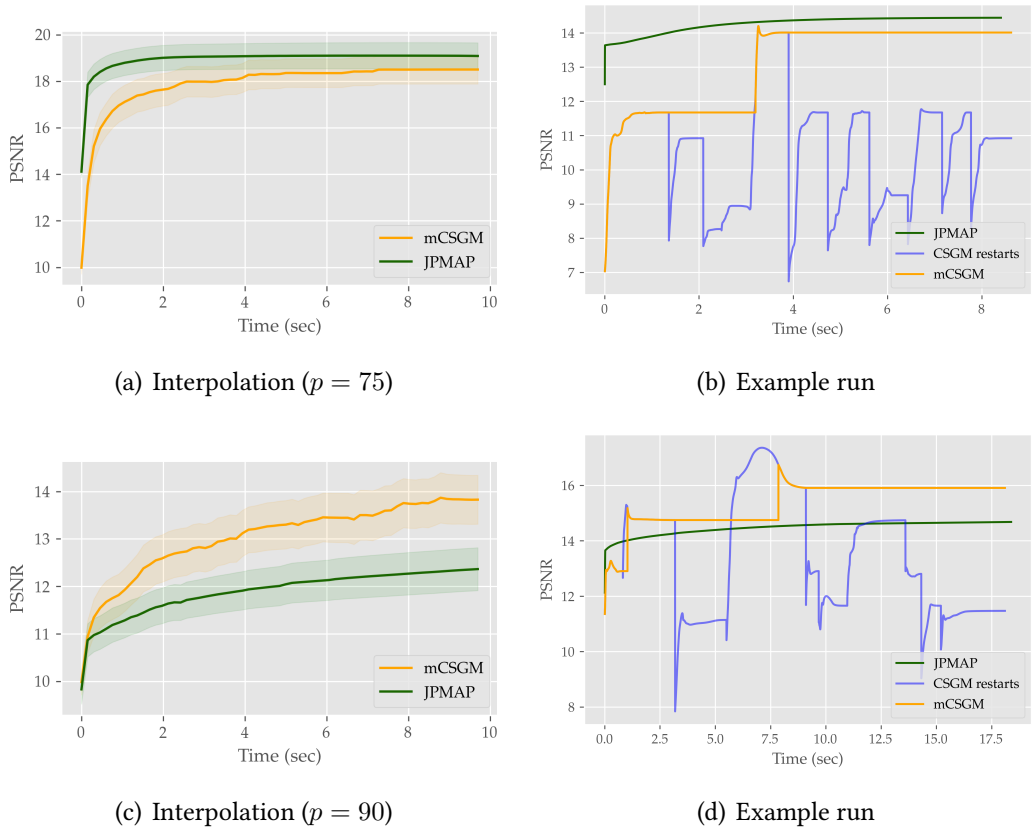


(a) Results on interpolation



(b) Results on deblurring.

Figure 4.7: *Experimental results on MNIST.* Comparison of JPMAP with the baseline algorithms described in Section 4.3.1. *(a)* Some selected results from the interpolation experiment with 80% of missing pixels and Gaussian noise with $\sigma = 10/255$. From top to bottom: original image $\boldsymbol{x}^*$, corrupted image $\boldsymbol{y}$, and the results computed by CSGM, mCSGM, PGD-GAN, PULSE, Algorithm 2 and JPMAP. *(b)* Same as (a) from the deblurring experiment with kernel size 3×3 and Gaussian noise with $\sigma = 10/255$. *Conclusion:* Our algorithm performs generally better than the baseline algorithms, although in some cases it falls behind mCSGM.



(a) Interpolation ($p = 75$)

(b) Example run

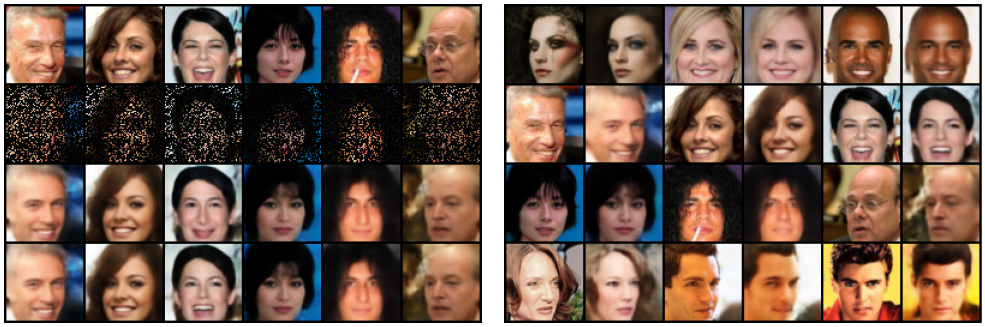
(c) Interpolation ($p = 90$)

(d) Example run

Figure 4.8: *Time/PSNR comparison between mCSGM and JPMAP. Left*: Confidence intervals (for a batch of 100 random experiments) for PSNR vs computing time for both algorithms on the interpolation problem with $p\%$ of missing pixels with noise std $\sigma = 10/255$. *Right*: Detailed view of one of the 100 random experiments on the left. Blue lines represent $m = 10$ random restarts of CSGM and the orange line is the PSNR of the best $\boldsymbol{z}_k$ at iteration k of mCSGM as measured by (4.3).

gles to generalize to images $\boldsymbol{x}$ which are very far away from $\mathcal{M}$ (the range of the generator). Indeed, in Section 4.3.4 we trained the VAE's encoder to generalize to $\mathcal{M} + n$ where $n \sim \mathcal{N}(0, \sigma^2_{\text{DVAE}} Id)$ and $\sigma_{\text{DVAE}} = 15/255$. This value is optimal for moderately ill-posed problems, but more extreme problems may require larger values of σ_{DVAE} or a coarse to fine scheme, where a coarse VAE (with large σ_{DVAE}) is used during the first few iterations and a finer VAE (with smaller σ_{DVAE}) is used later until convergence. Finally one may consider using random restarts for both JPMAP and CSGM for a more fair comparison.

Figure 4.8 performs a more detailed comparison between JPMAP and mCSGM, which also considers running times of both algorithms. For the stopping criteria used in our experiments, one run of JPMAP requires roughly as much time as mCSGM (with $m = 10$ restarts). In addition for moderately ill-posed problems (like



(a) Results of interpolation on CelebA. (b) Reconstructions $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x}))$.

Figure 4.9: *(a)* Some preliminary results on CelebA: 80% of missing pixels, noise std $\sigma = 10/255$. From top to bottom: original image $\boldsymbol{x}^*$, corrupted image $\tilde{\boldsymbol{x}}$, restored by CSGM [24], restored image $\hat{\boldsymbol{x}}$ by our framework. *(b)* Reconstructions $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x}))$ (even columns) for some test samples $\boldsymbol{x}$ (odd columns), showing the over-regularization of data manifold imposed by the trained vanilla VAE. As a consequence, $-\log p_{\mathcal{Z}|\mathcal{Y}}(\boldsymbol{z} \mid \boldsymbol{y})$ does not have as many local minima and then a simple gradient descent yields almost the same result as JPMAP (except on third column of (a)).

interpolation of 75% missing pixels see subfigures (a) and (b)) where JPMAP's performance beats mCSGM, we can observe that JPMAP also converges much faster to that solution. For more extremely ill-posed problems (like interpolation of 90% missing pixels, see subfigures (c) and (d)) the opposite is true.

In the case of CelebA, we did not observe as much difference between JPMAP and CSGM as on MNIST. In Figure 4.9(a) the restorations on an interpolation problem (80% of missing pixels) are very similar to each other, but blurry. Also, although this problem is very ill-posed, both algorithms impressively find a solution $\boldsymbol{z}^*$ very close to the code $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ of the ground truth image $\boldsymbol{x}$, except for the third column where CSGM converges to a local minimum.

We hypothesize that, as CelebA is a substantially more complex dataset than MNIST, a simple model like vanilla VAE is over-regularizing the manifold of samples (underfitting problem). In particular, because of the spectral bias [175] the learned manifold perhaps only contains low-frequency approximations of the true images as we can see in the reconstructions $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x}))$ of test samples (see Figure 4.9(b)). This may cause the posterior $p_{\mathcal{Z}|\mathcal{Y}}(\boldsymbol{z} \mid \boldsymbol{y})$ to have fewer local minima. With more realistic generative models such as VDVAE [45] or NVAE [225], which better represent the true data manifold, we expect the objective function $-\log p_{\mathcal{Z}|\mathcal{Y}}(\boldsymbol{z} \mid \boldsymbol{y})$ to exhibit a much larger number of local minima, thus making it more difficult to optimize by a simple gradient descent scheme. In that situation the proposed JPMAP method would more clearly show its advantages.

4.4 Conclusions and Future work

In this chapter we presented a new framework to solve inverse problems with a convex data-fitting term and a non-convex regularizer learned in the latent space via variational autoencoders. Unlike similar approaches like CSGM [24] , PULSE [151] and PGD-GAN [200] which learn the prior based on generative models, our approach is based on a generalization of alternate convex search to quasi-bi-convex functionals. This quasi-bi-convexity is the result of considering the joint posterior distribution of latent and image spaces. As a result, the proposed approach provides convergence guarantees that extend to a larger family of inverse problems. Experiments on denoising, interpolation, deconvolution, super-resolution and compressed sensing confirm this, since our approach gets stuck much less often in spurious local minima than CSGM, PGD-GAN or PULSE, which are simply based on gradient descent of a highly non-convex functional. This leads to restored images which are significantly better in terms of PSNR and LPIPS.

JPMAP vs related *Plug & Play* approaches When compared to other decoupled *plug & play* approaches that solve inverse problems using NN-based priors, our approach is constrained in different ways:

(a) In a certain sense our approach is *less constrained* than existing decoupled approaches since we do not require to retrain the NN-based denoiser to enforce any particular property to ensure convergence: [192] requires the denoiser's residual operator to be non-expansive, and [94, 200] require the denoiser to act as a projector. The effect of these modifications to the denoiser on the quality of the underlying image prior has never been studied in detail and chances are that such constraints degrade it. Our method only requires a variational autoencoder without any further constraints, and the quality and expressiveness of this prior can be easily checked by sampling and reconstruction experiments. Checking the quality of the prior is a much more difficult task for [192, 94, 200] which rely on an implicit prior, and do not provide a generative model.

(b) Unlike [192] which requires the data-fitting term $F(\boldsymbol{x})$ to be *strongly convex* to ensure convergence, our method admits weakly convex and ill-posed data-fitting terms like missing pixels, compressed sensing and non-invertible blurring for instance.

(c) On the other hand our method is *more constrained* in the sense that it relies on a generative model of a *fixed size*. Even if the generator and encoder are both convolutional neural networks, training and testing the same model on images of different sizes is a priori not possible because the latent space has a fixed dimension and a fixed distribution. As a future work we plan to explore different ways to address this limitation. The most straightforward way is to use our model to learn a prior of image patches of a fixed size and stitch this model via aggrega-

tion schemes like in EPLL [249] to obtain a global prior model for images of any size. Alternatively we can use hierarchical generative models like in [117, 225] or resizable ones like in [15, 238], and adapt our framework accordingly.

MAP-x or MAP-z or joint MAP-x-z In this work we explored and clarified the tight relationships between joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ estimation, splitting and continuation schemes and the more common MAP-$\boldsymbol{z}$ estimator in the context of inverse problems with a generative prior. On the other hand MAP-$\boldsymbol{x}$ estimators (which are otherwise standard in bayesian imaging) remained largely unexplored in the context of generative priors, due to the optimization challenges they impose, until the recent work of [101, 238] showed that a normalizing flow-based generative model allows to overcome those challenges and deems this problem tractable. Similarly [160] use Glow (an invertible normalizing flow) to compare synthesis-based and analysis-based reconstructions. Yet an extensive comparison of the advantages and weaknesses of these three families of estimators under the same prior model is still missing, and so is the link between these MAP estimators and the analysis/synthesis-based estimators in [160]. This will be the subject of future work.

Extension to higher dimensional problems The present paper provides a first proof of concept of our framework, on a very simple dataset (MNIST) with a very simple VAE. More experiments are needed to verify that the framework preserves its qualitative advantages on more high-dimensional datasets (like CelebA, FFHQ, etc.), and a larger selection of inverse problems.

Generalizing our proposed method to much higher dimensional problems implies training much more complex generative models which can match the finer details and higher complexity of such data. We can still use over-simplified generative models in those cases, but our preliminary experiments suggest that in that situation, not only do we obtain relatively poor reconstructions, but the objective function associated to the MAP-$\boldsymbol{z}$ problem presents less spurious local-minima: as a consequence our proposed joint MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ is overkill in that configuration, and does not present such a great competitive advantage.

The big challenge of generalizing our proposed method to much higher dimensional problems is then to train sufficiently detailed and complex generative models. And in this area VAEs traditionally lagged behind GANs in terms of quality of the generated samples, the former producing in general more blurred samples. Nevertheless some studies [194] show that VAEs and Normalizing Flows produce more accurate representations of the probability distribution. In the medium term our work should be able to benefit from recent advances in VAE architectures [45, 225, 51], and adversarial training for VAEs [173, 247] that reach GAN-quality samples with the additional benefits of VAEs. These extensions are however nontrivial, since these VAEs have a huge number of parameters and they need to be

retrained or fine-tuned using a denoising criterion (see section 4.3.4 and [109]) for our method to work properly. In addition, the latent space of the most competitive VAEs is much larger than the image space, which may reduce its regularization capabilities. As an alternative, GAN-based generative models can be augmented with a denoising encoder network [62], and Normalizing Flows can also act as projectors or denoising VAEs if we split the latent space to separate the data manifold from its complement, as suggested in [28, 139]. In combination with relaxation techniques, such augmented GANs or specially tailored Flows may provide SOTA priors that fit our quasi-bi-convex optimization framework.

Towards stronger convergence guarantees under weaker conditions. The proposed Algorithm 6 bears strong similarities with ADMM with non-linear constraints as introduced by Valkonen *et al.* [226, 14] and analyzed by Latorre-Gómez *et al.* [128]. Latorre-Gómez result provides very strong convergence guarantees (linear convergence rates to a global optimum), but requires the data fitting term to be strongly convex or to satisfy a restricted strong convexity property. Our result, on the other hand, provides much weaker convergence guarantees (convergence to a stationary point), but does not require strong convexity. Further exploring these connections might hopefully lead to something closer to the best of both worlds.

4.5 Appendix

4.5.1 Properties of J_1

In this section, we establish that the objective function J_1 fulfills the assumptions required to prove the convergence of Algorithm 3, namely

- $J_1(\cdot, \boldsymbol{z})$ is convex for any $\boldsymbol{z}$;
- $J_1(\cdot, \boldsymbol{z})$ has a unique minimizer for any $\boldsymbol{z}$;
- J_1 is coercive;
- J_1 is continuously differentiable;

We recall that

$$J_1(\boldsymbol{x}, \boldsymbol{z}) = \underbrace{\frac{1}{2\,\sigma^2}\,\|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}\|^2}_{F(\boldsymbol{x},\boldsymbol{y})} + \underbrace{\frac{1}{2}\Big(Z_\theta(\boldsymbol{z}) + \|\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2\Big)}_{H_\theta(\boldsymbol{x},\boldsymbol{z})} + \frac{1}{2}\,\|\boldsymbol{z}\|^2$$

where

$$Z_\theta(\boldsymbol{z}) = d\log(2\pi) + \log\det\boldsymbol{\Sigma}_\theta(\boldsymbol{z}).$$

Thus, it is the sum of three non-negative terms.

Convexity and unicity of the minimizer of $J_1(\cdot, z)$

Let $\boldsymbol{z}$ be fixed. Then there exists a constant $C \in \mathbb{R}$ such that $\forall\, \boldsymbol{x}$

$$J_1(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{2\,\sigma^2} \|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}\|^2 + \|\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2 + C$$

Being the sum of two quadratic forms, $J_1(\cdot, \boldsymbol{z})$ is obviously twice differentiable. Its gradient is given by

$$\begin{aligned}\frac{\partial J_1}{\partial \boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{z}) =& \frac{1}{\sigma^2}\,\boldsymbol{A}^T(\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}) \\ &+ 2\,(\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z}))^T\big(\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\big)\end{aligned}$$

and its Hessian is

$$\text{Hess}_{\boldsymbol{x}} J_1(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{\sigma^2}\,\boldsymbol{A}^T\boldsymbol{A} + 2\,(\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z}))^T\boldsymbol{\Sigma}_\theta^{-1/2}(\boldsymbol{z})$$

Since $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}) = \gamma^2 I$ the Hessian is positive definite (without the need to assume that A is full rank), and we have that

Lemma 1. *$J_1(\cdot, \boldsymbol{z})$ is strictly convex for any $\boldsymbol{z}$.*

An immediate consequence is the unicity of the minimizer of the partial function $J_1(\cdot, \boldsymbol{z})$.

Coercivity of J_1

Lemma 2. *J_1 is coercive.*

Proof. First, let us note that J_1 is the sum of three non-negative terms. If it was not coercive, then we could find a sequence $(\boldsymbol{x}_k, \boldsymbol{z}_k) \to \infty$ such that $J_1(\boldsymbol{x}_k, \boldsymbol{z}_k)$ is bounded. As a consequence all three terms are bounded. In particular the last term $\|\boldsymbol{z}_k\|$ is bounded, which means that $\boldsymbol{x}_k \to \infty$. From Property 1, $\{\boldsymbol{\mu}_\theta(\boldsymbol{z}_k)\}$ and $\{\boldsymbol{\Sigma}_\theta(\boldsymbol{z}_k)\}$ are bounded for bounded $\{\boldsymbol{z}_k\}$. Now, from the definition of the second term of J_1, we get that, $\{\boldsymbol{\mu}_\theta(\boldsymbol{z}_k)\}$ and $\{\boldsymbol{\Sigma}_\theta(\boldsymbol{z}_k)\}$ being bounded and $\boldsymbol{x}_k$ going to ∞ yield that $H_\theta(x_k, z_k)$ goes to infinity, while being bounded. This leads to a contradiction and thus proves that J_1 is coercive.

□

Regularity of J_1

In the sequel we adopt the common assumption that all neural networks used in this work are composed of a finite number d of layers, each layer being composed of: *(a)* a linear operator (*e.g.* convolutional or fully connected layer), followed by *(b)* a non-linear L-Lipschitz component-wise activation function with $0 < L < \infty$.

Therefore we have the following property:

Property 1. *For any neural network f_θ with parameters θ having the structure described above:*
There exists a constant C_θ such that $\forall \boldsymbol{u}$,

$$\|f_\theta(\boldsymbol{u})\|_2 \leq C_\theta \|\boldsymbol{u}\|_2.$$

Concerning activation functions we use two kinds:

- continously differentiable activations like ELU, or
- continuous but non-differentiable activations like ReLU

Hence, by composition, we have that

Lemma 3. *For continuously differentiable activation functions, J_1 is continuously differentiable.*

4.5.2 MAP-x and MAP-z for deterministic generative models

Assume that the stochastic γ-generative model is

$$p_{\mathcal{X}_\gamma|\mathcal{Z}_\gamma}(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}(\mathsf{D}(\boldsymbol{z}), \gamma^2 I)$$

meaning that when $\gamma \to 0$

$$p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) = \delta(\boldsymbol{x} - \mathsf{D}(\boldsymbol{z}))$$

We now analyze the MAP-$\boldsymbol{z}$ and MAP-$\boldsymbol{x}$ estimators for the limit case when $\gamma = 0$. This is what we call a deterministic generative model, and it includes GANs for instance.

MAP-z

By definition the MAP-$\boldsymbol{z}$ estimator is obtained by maximising the posterior with respect to $\boldsymbol{z}$:

$$\begin{aligned}\hat{\boldsymbol{z}}_{\text{MAP}-\boldsymbol{z}} &= \arg\max_{\boldsymbol{z}} \left\{ p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z} \mid \boldsymbol{y}\right) \right\} \\ &= \arg\max_{\boldsymbol{z}} \left\{ p_{\mathcal{Y}|\mathcal{Z}}\left(\boldsymbol{y} \mid \boldsymbol{z}\right) p_{\mathcal{Z}}\left(\boldsymbol{z}\right) \right\}.\end{aligned} \tag{4.26}$$

In the last line we used Bayes rule to rewrite this posterior in more simple terms. However, this expression still involves the unknown conditional $p_{\mathcal{Y}|\mathcal{Z}}\left(\boldsymbol{y} \mid \boldsymbol{z}\right)$. Let us express this maximization in terms of $p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right)$.

To do so we recall the relation between the conditionals and the joint:

$$p_{\mathcal{Y}|\mathcal{Z}}\left(\boldsymbol{y} \mid \boldsymbol{z}\right) p_{\mathcal{Z}}\left(\boldsymbol{z}\right) = p_{\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{y}, \boldsymbol{z}\right) = p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z} \mid \boldsymbol{y}\right) p_{\mathcal{Y}}\left(\boldsymbol{y}\right) \tag{4.27}$$

We can also compute the joint distribution $p_{\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{y}, \boldsymbol{z}\right)$ by marginalization on a third random variable $\mathcal{X}$:

$$\begin{aligned}p_{\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{y}, \boldsymbol{z}\right) &= \int p_{\mathcal{X},\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\right) d\boldsymbol{x} \\ &= \int p_{\mathcal{Y}|\mathcal{X},\mathcal{Z}}\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}\right) p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x} \mid \boldsymbol{z}\right) p_{\mathcal{Z}}\left(\boldsymbol{z}\right) d\boldsymbol{x} \\ &= \int p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right) \delta(\boldsymbol{x} - \mathsf{D}(\boldsymbol{z})) p_{\mathcal{Z}}\left(\boldsymbol{z}\right) d\boldsymbol{x} \\ &= p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \mid \mathsf{D}(\boldsymbol{z})\right) p_{\mathcal{Z}}\left(\boldsymbol{z}\right)\end{aligned} \tag{4.28}$$

The third line follows from our graphical model $\mathcal{Z} \to \mathcal{X} \to \mathcal{Y}$ which implies that once we know $\mathcal{X} = \boldsymbol{x}$, then $\mathcal{Z}$ provides no additional information, therefore

$$p_{\mathcal{Y}|\mathcal{X},\mathcal{Z}}\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}\right) = p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right).$$

The last line follows simply from the integration on $\boldsymbol{x}$ of a delta function.

From equations (4.27) and (4.28) we can derive an expression of $p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z} \mid \boldsymbol{y}\right)$ in terms of $p_{\mathcal{Y}|\mathcal{X}}\left(\cdot \mid \cdot\right)$ and the generator D namely:

$$p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z} \mid \boldsymbol{y}\right) = \frac{1}{p_{\mathcal{Y}}\left(\boldsymbol{y}\right)} p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \mid \mathsf{D}(\boldsymbol{z})\right) p_{\mathcal{Z}}\left(\boldsymbol{z}\right)$$

This proves the main result of this section:

Proposition 3 (MAP-$\boldsymbol{z}$ estimator for deterministic generative models)**.** *Assume we have*

- *a deterministic generative model where* $X = \mathsf{D}(\mathcal{Z})$ *and*
- *an inverse problem characterised by the log conditional distribution* $\log p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x}) = -F(\boldsymbol{x}, \boldsymbol{y})$.

Then the MAP-$\boldsymbol{z}$ estimator is computed as $\hat{\boldsymbol{x}}_{MAP-\boldsymbol{z}} = \mathsf{D}(\hat{\boldsymbol{z}}_{MAP-\boldsymbol{z}})$ *where*

$$\begin{aligned}\hat{\boldsymbol{z}}_{MAP-\boldsymbol{z}} &= \arg\max_{\boldsymbol{z}} \left\{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \mathsf{D}(\boldsymbol{z}))\, p_{\mathcal{Z}}(\boldsymbol{z})\right\} \\ &= \arg\min_{\boldsymbol{z}} \left\{F(\mathsf{D}(\boldsymbol{z}), \boldsymbol{y}) - \log p_{\mathcal{Z}}(\boldsymbol{z})\right\}.\end{aligned} \tag{4.29}$$

MAP-x

The MAP-$\boldsymbol{x}$ estimator is obtained by maximizing the posterior with respect to $\boldsymbol{x}$. The generative model induces a prior on X via the push-forward measure $p_{\mathcal{X}} = \mathsf{D}\sharp p_{\mathcal{Z}}$, which following [165, section 5] can be developed as

$$p_{\mathcal{X}}(\boldsymbol{x}) = \frac{p_{\mathcal{Z}}(\mathsf{D}^{-1}(\boldsymbol{x}))}{\sqrt{\det S(\mathsf{D}^{-1}(\boldsymbol{x}))}}\delta_{\mathcal{M}}(\boldsymbol{x})$$

where $S = \left(\frac{\partial \mathsf{D}}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial \mathsf{D}}{\partial \boldsymbol{z}}\right)$ is the squared Jacobian and the manifold $\mathcal{M} = \{\boldsymbol{x} : \exists \boldsymbol{z},\ \boldsymbol{x} = \mathsf{D}(\boldsymbol{z})\}$ represents the image of the generator D.

With such a prior $p_{\mathcal{X}}$, the $\boldsymbol{x}$-optimization (4.2) required to obtain $\hat{\boldsymbol{x}}_{\text{MAP}}$ becomes intractable (in general), for various reasons:

- the computation of $\det S$,
- the inversion of D, and
- the hard constraint $\boldsymbol{x} \in \mathcal{M}$.

These operations are are all memory and/or computationally intensive, except when they are partially addressed by the use of a normalizing flow like in [101, 238].

4.5.3 Joint MAP-x-z, Continuation Scheme and convergence to MAP-z

The functional $J_{1,\beta}$ introduced in Equation 4.5 can be seen from two different perspectives.

From a machine learning perspective it corresponds to the joint log-posterior J_1 in the case where $\boldsymbol{\Sigma}_\theta(\boldsymbol{z}) = \frac{1}{\beta} I$ and $\boldsymbol{\mu}_\theta = \mathsf{D}$, namely:

$$J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z}) = \underbrace{\frac{1}{2\,\sigma^2} \|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}\|^2}_{F(\boldsymbol{x},\boldsymbol{y})} + \underbrace{\frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))\|^2}_{H_\theta(\boldsymbol{x},\boldsymbol{z}) = \varphi_\beta(\boldsymbol{x},\boldsymbol{z})} + \frac{1}{2}\|\boldsymbol{z}\|^2 + C_\beta.$$

From an optimization standpoint it can be considered as an inexact penalisation procedure: We want to solve the constrained problem

$$\min_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{C}} \underbrace{F(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{z}\|^2}_{=J_{1,0}(\boldsymbol{x},\boldsymbol{z})}$$

with $\mathcal{C} = \{(\boldsymbol{x}, \boldsymbol{z}) \mid \boldsymbol{x} = \boldsymbol{\mu}_\theta(\boldsymbol{z})\}$ whose solution provides the MAP-$\boldsymbol{z}$ estimator

$$(\boldsymbol{x}^*, \boldsymbol{z}^*) \in \operatorname*{arg\,min}_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{C}} J_{1,0}(\boldsymbol{x}, \boldsymbol{z}). \tag{4.30}$$

To do so, we introduced the family of unconstrained problems

$$\min_{\boldsymbol{x},\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z})$$

and their corresponding minimizers

$$(\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{z}}_\beta) \in \operatorname*{arg\,min}_{\boldsymbol{x},\boldsymbol{z}} J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z})$$

which for $\beta = \frac{1}{\gamma^2}$ provide the MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ estimator.

We can show that the MAP-$\boldsymbol{x}$-$\boldsymbol{z}$ estimator converges to the MAP-$\boldsymbol{z}$ estimator when $\beta \to \infty$ (or equivalently $\gamma \to 0$).

Proposition 4. *The unconstrained functional tends to the constrained functional plus the constraint:*

$$J_{1,\beta}(\boldsymbol{x}, \boldsymbol{z}) \xrightarrow{\beta\to\infty} J_{1,\infty}(\boldsymbol{x}, \boldsymbol{z}) = F(\boldsymbol{x}, \boldsymbol{z}) + \iota_{\boldsymbol{x}=\boldsymbol{\mu}_\theta(\boldsymbol{z})}(\boldsymbol{x}, \boldsymbol{z}) + \frac{1}{2}\|\boldsymbol{z}\|^2 \tag{4.31}$$

and the unconstrained minimizers tend to the constrained minimizer as $\beta \to \infty$:

$$\lim_{\beta\to\infty} (\hat{\boldsymbol{x}}_\beta, \hat{\boldsymbol{z}}_\beta) \in \operatorname*{arg\,min}_{(\boldsymbol{x},\boldsymbol{z})\in\mathcal{C}} J_{1,0}(\boldsymbol{x}, \boldsymbol{z}) = \operatorname*{arg\,min}_{\boldsymbol{x},\boldsymbol{z}} J_{1,\infty}(\boldsymbol{x}, \boldsymbol{z}). \tag{4.32}$$

Proof. The pointwise convergence of φ_β to $\iota_{\boldsymbol{x}=\mathsf{D}(\boldsymbol{z})}$ as β goes to ∞ is straightforward.

Let us first prove that for any sequence $(\beta_n)_n$ that goes to ∞, the quantity $\|\hat{\boldsymbol{x}}_{\beta_n} - \mathsf{D}(\hat{\boldsymbol{z}}_{\beta_n})\|$ goes to zero. Otherwise, for any $\varepsilon > 0$, there exists a subsequence $(\beta_{n_j})_j$ such that $\|\hat{\boldsymbol{x}}_{\beta_n} - \mathsf{D}(\hat{\boldsymbol{z}}_{\beta_n})\| > \varepsilon$. In this case, for any $\boldsymbol{z}$, one has by optimality

$$J_{1,0}(\mathsf{D}(\boldsymbol{z}), \boldsymbol{z}) = J_{1,\beta_{n_j}}(\mathsf{D}(\boldsymbol{z}), \boldsymbol{z}) \geq J_{1,\beta_{n_j}}(\hat{\boldsymbol{x}}_{\beta_{n_j}}, \hat{\boldsymbol{z}}_{\beta_{n_j}}) > J_{1,0}(\hat{\boldsymbol{x}}_{\beta_{n_j}}, \hat{\boldsymbol{z}}_{\beta_{n_j}}) + \frac{\beta_{n_j}}{2}\,\varepsilon^2$$

As a result, the nonnegative quantity $J_{1,0}(\hat{\boldsymbol{x}}_{\beta_{n_j}}, \hat{\boldsymbol{z}}_{\beta_{n_j}})$ goes to $-\infty$, which leads to a contradiction. Thus, one has $\hat{\boldsymbol{x}}_\infty = \mathsf{D}(\hat{\boldsymbol{z}}_\infty)$ for any limit point $(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty)$ of $(\hat{\boldsymbol{x}}_{\beta_n}, \hat{\boldsymbol{z}}_{\beta_n})$. Assume that $J_{1,0}(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty) > J_{1,0}(\boldsymbol{x}^*, \boldsymbol{z}^*)$. Since

$$J_{1,\beta_n}(\hat{\boldsymbol{x}}_{\beta_n}, \hat{\boldsymbol{z}}_{\beta_n}) \leq J_{1,\beta_n}(\boldsymbol{x}^*, \boldsymbol{z}^*) = J_{1,0}(\boldsymbol{x}^*, \boldsymbol{z}^*) < J_{1,0}(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty)$$

this leads to another contradiction.

□

The previous result motivates Algorithm 2.

Consider Algorithm 2 in the ideal case (maxiter=∞) where the internal loop converges.

Proposition 5 (Convergence of Algorithm 2)**.** *Let $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)_k$ be a sequence generated by Algorithm 2 when maxiter=∞. If $(\boldsymbol{z}_\infty^k)_k$ is bounded, then any limit point of $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)_k$ is in $\mathcal{C}$. Moreover, any limit point of $(\boldsymbol{z}_\infty^k)_k$ is a stationary point of*

$$f(\boldsymbol{z}) = F(\mathsf{D}(\boldsymbol{z}), \boldsymbol{y}) + \frac{1}{2}\,\|\boldsymbol{z}\|^2 \tag{4.33}$$

Proof. Note that, for any k, $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ is a limit point of the sequence generated by the k-th subloop in Algorithm 2 if it does not converge. Let $(\beta_k)_k$ a sequence that converges to ∞. Let $k \in \mathbb{N}$. We consider the sequence $(\boldsymbol{x}_n^k, \boldsymbol{z}_n^k)_n$ generated by

$$\forall\, n \in \mathbb{N}, \qquad \boldsymbol{z}_{n+1}^k \in \arg\min_{\boldsymbol{z}} J_{1,\beta_k}(\boldsymbol{x}_n^k, \boldsymbol{z}) \text{ and } \boldsymbol{x}_{n+1}^k = \arg\min_{\boldsymbol{x}} J_{1,\beta_k}(\boldsymbol{x}, \boldsymbol{z}_{n+1}^k)$$

with $\boldsymbol{x}_0^k = \boldsymbol{x}_\infty^{k-1}$. Since J_{1,β_k} corresponds to a particular instance of J_1, and since Algorithm 2 can be seen asymptotically as a particular instance of Algorithm 5, one can use all the results established in Proposition 2. In particular, the sequence $(\boldsymbol{x}_n^k, \boldsymbol{z}_n^k)_n$ admits a limit point $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ and we have

$$\frac{\partial J_{1,\beta_k}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) = \frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) + \beta_k(\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)) = 0$$

and

$$\frac{\partial J_{1,\beta_k}}{\partial \boldsymbol{z}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) = \frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) + \beta_k (D\mathsf{D}(\boldsymbol{z}_\infty^k))^* (\mathsf{D}(\boldsymbol{z}_\infty^k) - \boldsymbol{x}_\infty^k) = 0$$

By convexity, $\boldsymbol{x}_\infty^k$ is the (unique) minimizer of $J_{1,\beta_k}(\cdot, \boldsymbol{z}_\infty^k)$.

Assume that the sequence $(\boldsymbol{z}_\infty^k)_k$ is bounded. By optimality, one has

$$\min J_{1,0} \leq J_{1,0}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) \leq J_{1,\beta_k}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) \leq J_{1,\beta_k}(\mathsf{D}(\boldsymbol{z}_\infty^k), \boldsymbol{z}_\infty^k) = J_{1,0}(\mathsf{D}(\boldsymbol{z}_\infty^k), \boldsymbol{z}_\infty^k)$$

Since $(J_{1,0}(\mathsf{D}(\boldsymbol{z}_\infty^k), \boldsymbol{z}_\infty^k))_k$ is bounded, so is $(J_{1,0}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k))_k$. By coercivity, the sequence $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)_k$ is also bounded. Then it admits a limit point denoted $(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty)$. Let $(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j})_j$ be a convergent subsequence of limit $(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty)$. Let us assume that $\hat{\boldsymbol{x}}_\infty \neq \mathsf{D}(\hat{\boldsymbol{z}}_\infty)$. Then, there exists $a > 0$ and $j_0 \in \mathbb{N}$ such that

$$\forall\, j \geq j_0, \qquad \|\boldsymbol{x}_\infty^{k_j} - \mathsf{D}(\boldsymbol{z}_\infty^{k_j})\|^2 > a$$

Hence, one has

$$J_{1,\beta_{k_j}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) \geq J_{1,0}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) + \beta_{k_j}\, a \geq \min J_{1,0} + \beta_{k_j}\, a \underset{j\to+\infty}{\longrightarrow} \infty$$

which leads to a contradiction. This proves that $\hat{\boldsymbol{x}}_\infty = \mathsf{D}(\hat{\boldsymbol{z}}_\infty)$. Otherwise said, $(\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k))_k$ goes to zero.

Since we have for any k

$$\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) + \beta_k(\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)) = 0$$

the continuity of $\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}$ ensures that $\left(\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j})\right)_j$ converges; thus, so is $(\beta_{k_j}(\boldsymbol{x}_\infty^{k_j} - \mathsf{D}(\boldsymbol{z}_\infty^{k_j})))_j$. Then there exists $\lambda^* \in \mathbb{R}^d$ such that

$$\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) = -\beta_{k_j}(\boldsymbol{x}_\infty^{k_j} - \mathsf{D}(\boldsymbol{z}_\infty^{k_j})) \underset{j\to+\infty}{\longrightarrow} \lambda^* = \frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty)$$

and

$$\frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) = -\beta_{k_j}(D\mathsf{D}(\boldsymbol{z}_\infty^{k_j}))^*(\mathsf{D}(\boldsymbol{z}_\infty^{k_j}) - \boldsymbol{x}_\infty^{k_j}) \underset{j\to+\infty}{\longrightarrow} -(D\mathsf{D}(\hat{\boldsymbol{z}}_\infty))^*(\lambda^*)$$

Note that $f(\boldsymbol{z}) = J_{1,0}(\mathsf{D}(\boldsymbol{z}), \boldsymbol{z})$. One can check that f is differentiable and that

$$\nabla f(\boldsymbol{z}) = (D\mathsf{D}(\boldsymbol{z}))^* \left(\frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\mathsf{D}(\boldsymbol{z}), \boldsymbol{z})\right) + \frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\mathsf{D}(\boldsymbol{z}), \boldsymbol{z})$$

Hence, we have proved that

$$\nabla f(\hat{\boldsymbol{z}}_\infty) = 0$$

Conclusion: If $(\boldsymbol{z}_\infty^k)_k$ is bounded, any limit point of $(\boldsymbol{z}_\infty^k)_k$ is a stationary point of (4.33). □

In general, we can only prove that the limit points of the sequences generated by Algorithm 2 are stationary points of 4.33. However, if the growth of β is sufficiently slow, then we obtain the optimality of the limit points. Indeed, given that, in Algorithm 2, each subloop is an exact BCD scheme, one has for any n and any j

$$\forall\, \boldsymbol{z}, \qquad J_{1,\beta_{k_j}}(\boldsymbol{x}_{n-1}, \boldsymbol{z}_n) \leq J_{1,\beta_{k_j}}(\boldsymbol{x}_{n-1}, \boldsymbol{z})$$

By considering the subsequence $(\boldsymbol{x}_{n_\ell}, \boldsymbol{z}_{n_\ell})$, which converges to $\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}$ (we recall that $\boldsymbol{x}_{n_\ell}$ and $\boldsymbol{x}_{n_\ell - 1}$ have same limit), we can prove that

$$\forall\, \boldsymbol{z}, \qquad J_{1,\beta_{k_j}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) \leq J_{1,\beta_{k_j}}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z})$$

that is, $\boldsymbol{z}_\infty^{k_j}$ is a minimizer of $J_{1,\beta_{k_j}}(\boldsymbol{x}_\infty^{k_j}, \cdot)$. Hence, we have

$$\forall\, \boldsymbol{z}, \qquad J_{1,0}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}_\infty^{k_j}) + \frac{\beta_{k_j}}{2} \left\|\boldsymbol{x}_\infty^{k_j} - \mathsf{D}(\boldsymbol{z}_\infty^{k_j})\right\|^2 \leq J_{1,0}(\boldsymbol{x}_\infty^{k_j}, \boldsymbol{z}) + \frac{\beta_{k_j}}{2} \left\|\boldsymbol{x}_\infty^{k_j} - \mathsf{D}(\boldsymbol{z})\right\|^2$$

Assume that $\beta_{k_j} \|\boldsymbol{x}_\infty^{k_j} - \hat{\boldsymbol{x}}_\infty\|^2 \underset{j \to +\infty}{\longrightarrow} 0$**.** By letting j to ∞, we get that, for any $\boldsymbol{z}$ such that $\mathsf{D}(\boldsymbol{z}) = \boldsymbol{x}_\infty^{k_j}$,

$$J_{1,0}(\hat{\boldsymbol{x}}_\infty, \hat{\boldsymbol{z}}_\infty) \leq J_{1,0}(\hat{\boldsymbol{x}}_\infty, \boldsymbol{z}) + \lim_{j \to +\infty} \frac{\beta_{k_j}}{2} \|\boldsymbol{x}_\infty^{k_j} - \boldsymbol{x}_\infty^{k_j}\|^2 = J_{1,0}(\hat{\boldsymbol{x}}_\infty, \boldsymbol{z})$$

that is, $\hat{\boldsymbol{z}}_\infty$ is a minimizer of $J_{1,0}(\hat{\boldsymbol{x}}_\infty, \cdot) + \chi_{\mathcal{C}}(\hat{\boldsymbol{x}}_\infty, \cdot)$. By definition of f, this also means that $\hat{\boldsymbol{z}}_\infty$ is a minimizer of f. However, one has to note that the growth control for β depends on the convergence speed of $\boldsymbol{x}_\infty^k$, which cannot be estimated.

Algorithm 6 is a particular (truncated) case of Algorithm 2 with an adaptive choice of β that does not need to go to ∞.

Proposition 6 (Convergence of Algorithm 6)**.**

Proof. Let us write the Lagrangian of the problem solved in Algorithm 6:

$$\forall\, \lambda \geq 0, \qquad \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \lambda) = J_{1,0}(\boldsymbol{x}, \boldsymbol{z}) + \lambda \left(\|\boldsymbol{x} - \mathsf{D}(\boldsymbol{z})\|^2 - \varepsilon\right)$$

KKT conditions ensure that any solution $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ of the constrained problem is associated to at least one Lagrange multiplier $\lambda^* \geq 0$ such that

$$\frac{\partial \mathcal{L}}{\partial(\boldsymbol{x}, \boldsymbol{z})}(\boldsymbol{x}^*, \boldsymbol{z}^*; \lambda^*) = 0 = \begin{pmatrix} \frac{\partial J_{1,0}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*, \boldsymbol{z}^*) + 2\lambda^* \left(\boldsymbol{x}^* - \mathsf{D}(\boldsymbol{z}^*)\right) \\ \frac{\partial J_{1,0}}{\partial \boldsymbol{z}}(\boldsymbol{x}^*, \boldsymbol{z}^*) + 2\lambda^* \left(D\mathsf{D}(\boldsymbol{z}^*)\right)^* (\boldsymbol{x}^* - \mathsf{D}(\boldsymbol{z}^*)) \end{pmatrix}$$

According to the calculus above, this proves that $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ is a stationary point of $J_{1,2\lambda^*}$. Note that, if $\lambda^* = 0$, then $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ is a minimizer of $J_{1,0}$. Otherwise, one has $\|\boldsymbol{x}^* - \mathsf{D}(\boldsymbol{z}^*)\|^2 = \varepsilon$.

Hence, if we consider Algorithm 2 with the update rule for β_k as in Algorithm 6 and a stopping rule saying that the iterations stop as soon as, for any given k,

$$\|\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)\|^2 \leq \varepsilon$$

there are two possible cases:

1. **case $\lambda^* = 0$:** then $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ is a solution of the constraint problem iff $\nabla J_{1,0}(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k) = 0$ (that is, $\boldsymbol{x}_\infty^k = \mathsf{D}(\boldsymbol{z}_\infty^k)$);
2. **case $\lambda^* > 0$:** unless $\|\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)\|^2$ exactly equals ε, $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ is **not** a solution of the constraint problem

However, in general, $(\boldsymbol{x}_\infty^k, \boldsymbol{z}_\infty^k)$ is a solution of the following constraint problem

$$\min_{\|\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)\|^2 \leq \tilde{\varepsilon}} J_{1,0}(\boldsymbol{x}, \boldsymbol{z})$$

with $\tilde{\varepsilon} = \|\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)\|^2 \leq \varepsilon$. Hence, if we stop the iterations when $\|\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)\|^2 \leq \varepsilon$, we will get a solution of

$$\min_{\|\boldsymbol{x}_\infty^k - \mathsf{D}(\boldsymbol{z}_\infty^k)\|^2 \leq \tilde{\varepsilon}} J_{1,0}(\boldsymbol{x}, \boldsymbol{z}), \qquad \tilde{\varepsilon} \leq \varepsilon$$

which provides an error control as well. □

Chapter 5

Posterior Sampling with Autoencoding Prior

In the previous chapter, we showed how generative models can be used as image priors for computing solutions of generic inverse problems. Most of the work in this line of research actually focuses on approximating MAP estimators, which generally leads to non-convex optimization problems that can get stuck in suboptimal local minima [24, 200, 151]. Although these methods can be applied using different types of generative models, they usually rely on decoder networks (e.g. generators of GAN models) and do not leverage the bidirectional nature of VAE models.

The method we proposed in Chapter 4 computes a MAP estimator using a VAE, with convergence guarantees derived from biconvex optimization results.

In this chapter, we study the use of VAE models to propose sampling algorithms that explore the posterior distribution. In particular, an alternate sampling scheme for unknown and latent variables naturally leads to a Gibbs-like algorithm. The resulting posterior samples can be used for computing different point estimates such as the MAP or MMSE estimators, but also to perform uncertainty estimation in the form of confidence intervals. On the other hand, these algorithms are not optimized yet and hence show higher running times, so further work needs to be done to implement competitive methods. This is an ongoing work that will be soon submitted for journal publication.

5.1 Introduction

5.1.1 Markov Chain Monte Carlo (MCMC)

In Bayesian statistics, a problem we generally want to solve is to calculate expectations such as

$$\mathbb{E}_f[a(\boldsymbol{x})] = \int a(\boldsymbol{x}) f(\boldsymbol{x})\, d\boldsymbol{x}. \tag{5.1}$$

The mean $\mu_{\boldsymbol{x}}$ and variance $\sigma^2_{\boldsymbol{x}}$ of $\boldsymbol{x}$ (with distribution given by f) are special cases if we set $a(\boldsymbol{x}) = \boldsymbol{x}$ and $a(\boldsymbol{x}) = (\boldsymbol{x} - \mu_{\boldsymbol{x}})^2$ respectively. In particular, we are interested in computing expectations with respect to the posterior distribution of the inverse problem defined above:

$$f(\boldsymbol{x}) = p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x} \,|\, \boldsymbol{y}\right) = \frac{p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \,|\, \boldsymbol{x}\right) p_{\mathcal{X}}\left(\boldsymbol{x}\right)}{p_{\mathcal{Y}}\left(\boldsymbol{y}\right)}. \tag{5.2}$$

When the above expression (5.1) does not have an analytic solution, as is the case in most of the real-world applications in signal processing [66], a common strategy is to employ numerical estimates of it. Deterministic integration methods are not suitable when the integration space is high-dimensional, so we need to rely on Monte Carlo techniques [152, 184].

By the Strong Law of Large Numbers, the empirical average

$$\bar{a}_m := \frac{1}{m} \sum_{j=1}^{m} a(\boldsymbol{x}_j), \tag{5.3}$$

where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are i.i.d. samples drawn from distribution f, converges almost surely to $\mathbb{E}_f[a(\boldsymbol{x})]$ when m tends to infinity. Moreover, in the univariate case when the variance σ_a^2 of $a(X)$ (under f) is finite, the variance of the estimator (5.3) can also be approximated by

$$\bar{s}^2_m = \frac{1}{m^2} \sum_{j=1}^{m} [a(\boldsymbol{x}_j) - \bar{a}_m]^2. \tag{5.4}$$

Then, when m is large, the Central Limit Theorem leads to the following approximation:

$$\frac{\bar{a}_m - \mathbb{E}_f[a(\boldsymbol{x})]}{\sqrt{\bar{s}^2_m}} \sim \mathcal{N}(0, I) \tag{5.5}$$

which can be used to construct convergence tests and confidence bounds on the approximation of $\mathbb{E}_f[a(\boldsymbol{x})]$ [184].

Except for a few simple cases, it is not easy to draw samples directly from the target distribution f but only to evaluate $f(\boldsymbol{x})$, at least up to a normalizing constant. In that case, there are mainly two classes of simulation algorithms to get approximate samples $\boldsymbol{x}_j$ which can still be used to estimate $\mathbb{E}_f[a(\boldsymbol{x})]$. With *iterative simulation algorithms* we draw samples from a *sequence of distributions* which converge to the target distribution. Thus, as the number of iterations grows, we obtain samples which are more and more likely to be drawn from the desired distribution. On the other hand, *non-iterative simulation algorithms* draw samples from a *fixed distribution* which is an approximation of the target distribution, and then correct them (eg. by rejecting or weighting) in order to evaluate consistent estimators like (5.3) [78].

Markov Chain Monte Carlo (MCMC)

A widely used class of iterative simulation algorithms for drawing samples of complex distributions f are the so called *Markov Chain Monte Carlo (MCMC)* techniques [184]. Generally speaking, they consist in constructing an *ergodic* Markov chain $(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots)$ having f as its stationary distribution.

Commonly, such chains are generated by a *transition (or Markov) kernel*

$$T \colon \mathbb{R}^d \times \mathcal{B}(\mathbb{R}^d) \to [0, 1] \tag{5.6}$$

such that $T(\boldsymbol{x}, A)$ gives the probability of jumping from $\boldsymbol{x}$ to subset A at each iteration (here we only consider *homogeneous* Markov chains). When the measure $T(\boldsymbol{x}', \cdot)$ has a density, we use the notation $T(\boldsymbol{x}', \boldsymbol{x})$ for it. Formally, we start at $\boldsymbol{x}^{(0)}$ and draw samples $\boldsymbol{x}^{(i+1)} \sim T(\boldsymbol{x}^{(i)}, \boldsymbol{x})$ for $i \geq 0$. The transition kernel is constructed in such a way that the distribution of the generated sequence converges to the target distribution f. In particular, one of the required properties is to leave the target distribution invariant:

$$\int T(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}) \, d\boldsymbol{x} = f(\boldsymbol{x}'). \tag{5.7}$$

A sufficient condition that ensures that a transition kernel verifies this property is *detailed balance*:

$$T(\boldsymbol{x}, \boldsymbol{x}') f(\boldsymbol{x}) = T(\boldsymbol{x}', \boldsymbol{x}) f(\boldsymbol{x}') \quad \forall \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d. \tag{5.8}$$

This can be easily shown by integrating (5.8) on both sides wrt $\boldsymbol{x}$ and using that $\int T(\boldsymbol{x}', \boldsymbol{x}) \, d\boldsymbol{x} = 1$ for all $\boldsymbol{x}'$.

Observe that the identity kernel $T(\boldsymbol{x}, \boldsymbol{x}') = \delta(\boldsymbol{x} - \boldsymbol{x}')$ leaves invariant every distribution but is not useful for constructing a chain that *converges* to the target distribution. We need a convergence property called *ergodicity*. Some sufficient conditions that are generally easy to verify and ensure that the Markov chain converges to the invariant distribution *for all starting points* $\boldsymbol{x}$ are

- *Aperiodic*: $\nexists d \geq 2$ and disjoint subsets $A_1, \ldots, A_d$ such that $\pi(A_j) > 0 \; \forall j$ and
$$T(\boldsymbol{x}, A_{j+1}) = 1 \quad \forall \boldsymbol{x} \in A_j \pmod d$$
- *Harris recurrent*: for all B with $\pi(B) > 0$ and all $\boldsymbol{x}$ the chain will eventually reach B from $\boldsymbol{x}$ with probability 1

(see for example [185, Theorem 4]).

Metropolis-Hastings

The most popular MCMC method is the *Metropolis-Hastings (MH)* algorithm [153, 99], [184, Sect. 7.3]. The main idea is to sample from a generic *proposal distribution* $J(\boldsymbol{x}' \mid \boldsymbol{x})$ from which we can easily draw samples, and then accept or reject the proposed sample after evaluating the target distribution. The algorithm is summarized in Algorithm 7.

If the support of the proposal distribution $J(\cdot \mid \boldsymbol{x})$ contains the support of f for all $\boldsymbol{x}$, then it can be shown that the kernel $T(\boldsymbol{x}, \boldsymbol{x}')$ associated to the chain generated by the MH algorithm satisfies the detailed balance property, so f is the invariant distribution [184, Thm. 7.2]. Moreover, if the proposal distribution satisfies

$$\exists \epsilon > 0, \delta > 0 \quad \text{such that} \quad J(\boldsymbol{x}' \mid \boldsymbol{x}) > \epsilon \quad \forall \boldsymbol{x}, \boldsymbol{x}' \; / \; \|\boldsymbol{x} - \boldsymbol{x}'\| < \delta \tag{5.9}$$

then the chain is ergodic and converges to the target distribution [184, Corolary 7.7]. A Gaussian proposal

$$J(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}'; \boldsymbol{x}, \alpha^2 I) \tag{5.10}$$

is a universal choice which leads to the *random walk Metropolis-Hastings* algorithm. If we set $\alpha > 0$ too high, most of the samples from $J(\boldsymbol{x}' \mid \boldsymbol{x})$ may be rejected, which result in high autocorrelation between samples from the chain. On the other hand, with low values of α most of the proposed $\tilde{\boldsymbol{x}}$ will be accepted but the chain will vary slowly, so we need to run the chain for very long time (N large) to get convergence to the target distribution. The value of α can be set in order to get a predefined *acceptance ratio* [19].

Gibbs sampling with approximations [78]

Here we adopt a *data augmentation* approach [214]: let the input vector be now $(\boldsymbol{x}, \boldsymbol{z})$ instead of $\boldsymbol{x}$, where $\boldsymbol{z}$ is some *latent* or *auxiliary* variable. The main assumption is that sampling from the conditional distribution of $\boldsymbol{z}$ given $\boldsymbol{x}$ (and viceversa) is easier than sampling directly from the joint distribution of $(\boldsymbol{x}, \boldsymbol{z})$. For example,

Algorithm 7: Metropolis-Hastings algorithm

Require: Initial sample $\boldsymbol{x}^{(0)}$ with $f(\boldsymbol{x}^{(0)}) > 0$.

1: **for** $i := 0$ **to** $N - 1$ **do**

2: Draw $\tilde{\boldsymbol{x}}$ from $J(\boldsymbol{x} \mid \boldsymbol{x}^{(i)})$

3: Compute the *Metropolis-Hastings acceptance probability*:

$$\rho(\boldsymbol{x}^{(i)}, \tilde{\boldsymbol{x}}) = \min\left\{\frac{f(\tilde{\boldsymbol{x}})}{f(\boldsymbol{x}^{(i)})}\frac{J(\boldsymbol{x}^{(i)} \mid \tilde{\boldsymbol{x}})}{J(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}^{(i)})}, 1\right\}$$

4: Accept or reject $\tilde{\boldsymbol{x}}$:

$$\boldsymbol{x}^{(i+1)} = \begin{cases} \tilde{\boldsymbol{x}} & \text{with probability } \rho(\boldsymbol{x}^{(i)}, \tilde{\boldsymbol{x}}) \\ \boldsymbol{x}^{(i)} & \text{otherwise.} \end{cases} \tag{5.11}$$

5: **end for**

6: **return** Markov chain $\{\boldsymbol{x}^{(i)}\}_{i=1,\ldots,N}$.

this approach is particularly useful when we have a VAE-based generative model $p_{\mathcal{X},\mathcal{Z}}(\boldsymbol{x}, \boldsymbol{z})$, where sampling from $\boldsymbol{x}|\boldsymbol{z}$ and $\boldsymbol{z}|\boldsymbol{x}$ can be done efficiently using the decoder and encoder respectively.

A particular choice of the proposal distribution function in the Metropolis-Hastings algorithm lets us split the input variable in two and to perform sampling on each one alternatively, conditioning on the other. More specifically, we choose a proposal (conditional) distribution $q_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x})$ where $q_{\boldsymbol{z}}$ can be the true (target) conditional distribution of $\boldsymbol{z}$ given $\boldsymbol{x}$ or an approximation of it.

To construct the chain, given $(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})$ we first draw a sample $\tilde{\boldsymbol{z}} \sim q_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x}^{(i)})$. This corresponds to the following Metropolis-Hastings proposal distribution

$$J_1(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}} \mid \boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}) = \begin{cases} q_{\boldsymbol{z}}(\tilde{\boldsymbol{z}}|\boldsymbol{x}^{(i)}) & \text{if } \tilde{\boldsymbol{x}} = \boldsymbol{x}^{(i)}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.12}$$

Then, we compute the acceptance probability by

$$\rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}) = \min\left\{\frac{f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}})}{f(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})}\frac{J_1(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} \mid \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}})}{J_1(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}} \mid \boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})}, 1\right\} \tag{5.13}$$

$$= \min\left\{\frac{f(\tilde{\boldsymbol{z}}|\boldsymbol{x}^{(i)})}{f(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)})}\frac{q_{\boldsymbol{z}}(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)})}{q_{\boldsymbol{z}}(\tilde{\boldsymbol{z}}|\boldsymbol{x}^{(i)})}, 1\right\} \tag{5.14}$$

and accept or reject $\tilde{\boldsymbol{z}}$ as before:

$$\boldsymbol{z}^{(i+1)} = \begin{cases} \tilde{\boldsymbol{z}} & \text{with probability } \rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}), \\ \boldsymbol{z}^{(i)} & \text{otherwise.} \end{cases} \tag{5.15}$$

The procedure for fixing $\boldsymbol{z}$ and sampling from $q_{\boldsymbol{x}}(\boldsymbol{x}|\boldsymbol{z})$ to generate $\boldsymbol{x}^{(i+1)}$ is analogous. In some situations we can sample from some or all the true conditional distributions:

$$q_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x}) = f(\boldsymbol{z}|\boldsymbol{x}) \quad \text{and / or} \quad q_{\boldsymbol{x}}(\boldsymbol{x}|\boldsymbol{z}) = f(\boldsymbol{x}|\boldsymbol{z}). \tag{5.16}$$

Observe that in the first case $\rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}) = 1 \; \forall i$ and then $\tilde{\boldsymbol{z}}$ is always accepted. This is also true for $\tilde{\boldsymbol{x}}$ in the second case. If both cases are true, the resulting algorithm is the well known *Two-stage Gibbs sampler*. In other case, it is often called *Metropolis-within-Gibbs*.

5.1.2 Common issues of single chain MCMC algorithms

General purpose MCMC algorithms, in particular Metropolis-Hastings with generic transition kernels, are very appealing because they can be used to (asymptotically) generate samples from virtually any target distribution, and also they are easy to implement. In addition, as first samples will become less and less important as the number of iterations grows because ergodicity, the starting distribution from which we draw the initial samples of the chain does not need to be an accurate approximation of the target distribution. So we can say that these kind of generic algorithms are the best option (if not the only one) if one has little or no knowledge about the target distribution. However, there are well-known issues with these generic MCMC algorithms:

- *Mixing time*: although ergodicity ensures that, given enough time (iterations) we will eventually reach stationarity and then the generated samples will cover (asympotically) all the support of the target distribution, in practice it is common to see chains being trapped in modes for a long time and then to wrongly assume convergence [80].

- *Correlated samples*: this causes larger variances than in the i.i.d. case and slow convergence of the corresponding Monte Carlo estimators. Hence we may need to run very long chains in order to obtain accurate results.

In the rest of this chapter, we explore the use of VAE models to alternate sampling between image and latent variables. By leveraging the structure of the particular problem at hand, we can build algorithms for sampling from the posterior distribution with a VAE prior having better convergence behavior than generic methods.

5.2 Gibbs sampling using VAE

5.2.1 Data augmentation and substitution sampling

To compute expectations such as the MMSE estimator

$$\hat{\boldsymbol{x}}_{\text{MMSE}} = \mathbb{E}_{p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x}\,|\,\boldsymbol{y})}[\boldsymbol{x}] = \int_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{x}\; p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}\,|\,\boldsymbol{y}\right)\; d\boldsymbol{x} \tag{5.17}$$

we need to sample from the conditional distribution $p_{\mathcal{X}|\mathcal{Y}}\left(\cdot\,|\,\boldsymbol{y}\right)$. As this distribution is intractable in most practical applications, we follow a data-augmentation approach [214] to sample alternatively from $\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{y}$, which gives rise to a Gibbs sampler (also known as *substitution sampling scheme* [77]).

We begin first by observing that

$$p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}\,|\,\boldsymbol{y}\right) = \int p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}\,|\,\boldsymbol{y},\boldsymbol{z}\right) p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z}\,|\,\boldsymbol{y}\right)\; d\boldsymbol{z} \tag{5.18}$$

$$p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z}\,|\,\boldsymbol{y}\right) = \int p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z}\,|\,\boldsymbol{x}\right) p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}\,|\,\boldsymbol{y}\right)\; d\boldsymbol{x}. \tag{5.19}$$

In (5.19) we used that $p_{\mathcal{Z}|\mathcal{X},\mathcal{Y}}\left(\boldsymbol{z}\,|\,\boldsymbol{x},\boldsymbol{y}\right) = p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z}\,|\,\boldsymbol{x}\right)$ because, in our setting, $\boldsymbol{z}$ and $\boldsymbol{y}$ are conditionally independent given $\boldsymbol{x}$. Combining (5.18) and (5.19) we obtain

$$\begin{aligned} p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}\,|\,\boldsymbol{y}\right) &= \int p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}\,|\,\boldsymbol{y},\boldsymbol{z}\right) \left(\int p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z}\,|\,\boldsymbol{x}'\right) p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}'\,|\,\boldsymbol{y}\right)\; d\boldsymbol{x}' \right) d\boldsymbol{z} \\ &= \int \left(\int p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}\,|\,\boldsymbol{y},\boldsymbol{z}\right) p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z}\,|\,\boldsymbol{x}'\right)\; d\boldsymbol{z} \right) p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}'\,|\,\boldsymbol{y}\right)\; d\boldsymbol{x}' \\ &= \int h(\boldsymbol{x}', \boldsymbol{x})\; p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}'\,|\,\boldsymbol{y}\right)\; d\boldsymbol{x}' \end{aligned} \tag{5.20}$$

where

$$h(\boldsymbol{x}', \boldsymbol{x}) := \int p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}\,|\,\boldsymbol{y},\boldsymbol{z}\right) p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z}\,|\,\boldsymbol{x}'\right)\; d\boldsymbol{z} \tag{5.21}$$

can be seen as a jumping distribution from $\boldsymbol{x}'$ to $\boldsymbol{x}$ that depends on $\boldsymbol{y}$. The above shows that the distribution $p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}\,|\,\boldsymbol{y}\right)$ is a fixed point of the following integral operator:

$$I_h(p)(\boldsymbol{x}) = \int h(\boldsymbol{x}', \boldsymbol{x})\; p(\boldsymbol{x}')\, d\boldsymbol{x}'. \tag{5.22}$$

Thus, it is tempting to propose the following iterative algorithm for estimating $p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}\,|\,\boldsymbol{y}\right)$:

1. Start with an initial guess $p^{(0)}(\boldsymbol{x})$ of $p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}\,|\,\boldsymbol{y}\right)$.

2. For $i \geq 0$:
$$p^{(i+1)} = I_h(p^{(i)}). \tag{5.23}$$

Under mild conditions, this iterative approximations of $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$ has the following properties [77]:

- *Uniqueness*: $p_{\mathcal{X}|\mathcal{Y}}(\cdot \mid \boldsymbol{y})$ is the unique fixed point of (5.22).
- *Convergence*: For almost any $p^{(0)}$, the sequence defined by (5.23) converges monotonically in L^1 to $p_{\mathcal{X}|\mathcal{Y}}(\cdot \mid \boldsymbol{y})$.
- *Rate*: $\int |p^{(i)}(\boldsymbol{x}) - p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})| \, d\boldsymbol{x} \to 0$ geometrically in i.

5.2.2 Pseudo-Gibbs and Metropolis-within-Gibbs

To generate a Markov chain using the iteration proposed in (5.23), from an initial estimate $p^{(0)}(\boldsymbol{x})$ of $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$, we can sample $\boldsymbol{x}^{(0)}$ and then

- sample $\boldsymbol{z}^{(1)}$ from $p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \,\middle|\, \boldsymbol{x}^{(0)}\right)$,
- sample $\boldsymbol{x}^{(1)}$ from $p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x} \,\middle|\, \boldsymbol{y}, \boldsymbol{z}^{(1)}\right)$.

Observe that the conditional distribution of $\boldsymbol{x}^{(1)}$ given $\boldsymbol{x}^{(0)}$ is

$$p(\boldsymbol{x} \mid \boldsymbol{x}^{(0)}) = \int p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) \, p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \,\middle|\, \boldsymbol{x}^{(0)}\right) \, d\boldsymbol{z} = h(\boldsymbol{x}^{(0)}, \boldsymbol{x}) \tag{5.24}$$

and hence the marginal distribution of $\boldsymbol{x}^{(1)}$ is

$$p^{(1)}(\boldsymbol{x}) = \int h(\boldsymbol{x}', \boldsymbol{x}) \, p^{(0)}(\boldsymbol{x}') \, d\boldsymbol{x}' \underset{(5.22)}{=} I_h(p^{(0)})(\boldsymbol{x}). \tag{5.25}$$

Repetition of this cycle will generate a Markov chain $(\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots)$ with $h(\boldsymbol{x}', \boldsymbol{x})$ as the associated transition kernel.

In practice, in order to compute each iteration of (5.23) we need to draw samples from $\boldsymbol{z}|\boldsymbol{x}$ and from $\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{z}$. We recall that the conditional distribution of $\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{z}$ is Gaussian as is shown below.

Proposition 7. *For a linear inverse problem*

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^* + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2 I) \tag{5.26}$$

and a VAE with decoder

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_\theta(\boldsymbol{z}), \gamma^2 I) \tag{5.27}$$

the distribution $p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$ *is Gaussian with mean* ν *and covariance* Σ *given by*

$$\Sigma = (\sigma^{-2} A^T A + \gamma^{-2} I)^{-1}, \qquad \nu = \Sigma(\sigma^{-2} A^T \boldsymbol{y} + \gamma^{-2} \boldsymbol{\mu}_\theta(\boldsymbol{z})). \tag{5.28}$$

Proof. The result is a direct application of [22], Equations (2.113) to (2.117). For the sake of completeness, we give here a simpler proof. Recalling that

- $p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}; A\boldsymbol{x}, L^{-1})$ with $L = \sigma^{-2} I$
- $p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \mu_{\boldsymbol{x}}(\boldsymbol{z}), \Lambda^{-1})$ with $\Lambda = \gamma^{-2} I$
- $p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) = \dfrac{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})\, p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})}{p_{\mathcal{Y}|\mathcal{Z}}(\boldsymbol{y} \mid \boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})} = \dfrac{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})\, p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z})}{p_{\mathcal{Y}|\mathcal{Z}}(\boldsymbol{y} \mid \boldsymbol{z})}$

then

$$\begin{aligned}
-\log & p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) = -\log p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x}) - \log p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) + \log p_{\mathcal{Y}|\mathcal{Z}}(\boldsymbol{y} \mid \boldsymbol{z}) \\
&= \frac{1}{2}(\boldsymbol{y} - A\boldsymbol{x})^T L(\boldsymbol{y} - A\boldsymbol{x}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z}))^T \Lambda(\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z})) + C_1(\boldsymbol{y}, \boldsymbol{z}) \\
&= \frac{1}{2}\boldsymbol{x}^T A^T L A \boldsymbol{x} - \boldsymbol{y}^T L A \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T \Lambda \boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z})^T \Lambda \boldsymbol{x} + C_2(\boldsymbol{y}, \boldsymbol{z}) \\
&= \frac{1}{2}\boldsymbol{x}^T (A^T L A + \Lambda) x - (A^T L \boldsymbol{y} + \Lambda \boldsymbol{\mu}_\theta(\boldsymbol{z}))^T \boldsymbol{x} + C_2(\boldsymbol{y}, \boldsymbol{z}).
\end{aligned}$$

Setting

$$\begin{aligned}
\Sigma &= (A^T L A + \Lambda)^{-1} \\
\nu &= \Sigma(A^T L \boldsymbol{y} + \Lambda \boldsymbol{\mu}_\theta(\boldsymbol{z}))
\end{aligned}$$

we obtain

$$\begin{aligned}
\log p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) &= -\frac{1}{2}\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x} + \nu^T \Sigma^{-1} \boldsymbol{x} - C_2(\boldsymbol{y}, \boldsymbol{z}) \\
&= -\frac{1}{2}(\boldsymbol{x} - \nu)^T \Sigma^{-1} (\boldsymbol{x} - \nu) + C_3(\boldsymbol{y}, \boldsymbol{z}),
\end{aligned}$$

where $C_3(\boldsymbol{y}, \boldsymbol{z})$ does not depend on $\boldsymbol{x}$. This means that $\boldsymbol{x}|(\boldsymbol{y}, \boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{x}; \nu, \Sigma)$. Substituting $L = \sigma^{-2} I$ and $\Lambda = \gamma^{-2} I$ we obtain

$$\begin{aligned}
\Sigma &= (\sigma^{-2} A^T A + \gamma^{-2} I)^{-1} \\
\nu &= \Sigma(\sigma^{-2} A^T \boldsymbol{y} + \gamma^{-2} \boldsymbol{\mu}_\theta(z))
\end{aligned}$$

as required. □

Next, as the posterior $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$ is intractable, we use the encoder approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ for sampling from $\boldsymbol{z}|\boldsymbol{x}$. Without any other modification, this scheme corresponds to a *pseudo-Gibbs sampling* [147] and coincides with Gibbs sampling as long as the approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x}) \simeq p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$ is exact. Figure 4.2 in Chapter 4 gives empirical evidence that the variational distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ can give accurate approximations to $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$ on training data points, but to avoid instability problems or convergence to a distribution other than the target, we need to take this gap into account.

One way of ensure detailed-balance wrt the target distribution is to perform a Metropolis-Hastings step for sampling $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$ using $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ as proposal distribution, so using a Metropolis-within-Gibbs scheme as [147] does for missing data imputation. In our case, the acceptance probability (5.14) leads to

$$\rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}) = \min\left\{\frac{p_{\mathcal{Z}|\mathcal{X}}\left(\tilde{\boldsymbol{z}} \,\middle|\, \boldsymbol{x}^{(i)}\right)}{p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z}^{(i)} \mid \boldsymbol{x}^{(i)}\right)} \frac{q_\phi(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)})}{q_\phi(\tilde{\boldsymbol{z}}|\boldsymbol{x}^{(i)})}, 1\right\} \tag{5.29}$$

$$= \min\left\{\frac{p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x}^{(i)} \,\middle|\, \tilde{\boldsymbol{z}}\right) p_{\mathcal{Z}}(\tilde{\boldsymbol{z}})}{p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x}^{(i)} \mid \boldsymbol{z}^{(i)}\right) p_{\mathcal{Z}}\left(\boldsymbol{z}^{(i)}\right)} \frac{q_\phi(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)})}{q_\phi(\tilde{\boldsymbol{z}}|\boldsymbol{x}^{(i)})}, 1\right\}. \tag{5.30}$$

This acceptance step comes with negligible additional computational cost because all the evaluations of the encoder and decoder needed to compute the terms involved in (5.30) are already done in the pseudo-Gibbs algorithm. The resulting Metropolis-within-Gibbs algorithm is described in Algorithm 8.

Although theoretically this Metropolis-Hastings step ensures that the accepted samples are asymptotically drawn from the target distribution, we found that the acceptance rate can be very small when the encoder approximation is not as close to the true posterior distribution as expected. Hence, we propose to close the gap $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \simeq p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$ correcting the mean of the proposal distribution $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ as is described in Algorithm 9. The resulting pseudo-Gibbs algorithm is described in Algorithm 10.

We illustrate the behaviour of Algorithm 10 in Figure 5.1. Here, we generate several chains initialized at $\boldsymbol{z}^{(0)} \sim \mathcal{N}(0, I)$ and $\boldsymbol{x}^{(0)} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{z}^{(0)}), \gamma^2 I)$ (first row). In this toy inverse problem, $\boldsymbol{x} \in \mathbb{R}^2$, $\boldsymbol{A} = [1, -1]$, $\sigma = 0.01$ and $\boldsymbol{y} = 0$ (so $x_1 \simeq x_2$). First column shows the histogram of the values $\boldsymbol{z}^{(i)}$ at each iteration and central column the data samples they generate. When $\boldsymbol{x}^{(i)}$ is updated after applying the step 4 (using data $\boldsymbol{y}$) then $\boldsymbol{x}^{(i+1)}$ moves in the direction of the set of feasible solutions $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. In the next iteration (second row) the encoder in step 2 projects this new samples to the data manifold, and so on. In last row we see that every chain converged to one of the posterior modes depending on the initialization.

In Figures 5.2 and 5.3 we show the effect of this gap correction on the behavior of the chains generated by the resulting pseudo-Gibbs algorithm: when we do not apply any correction (column (b)) the limit distribution is not the same as the target

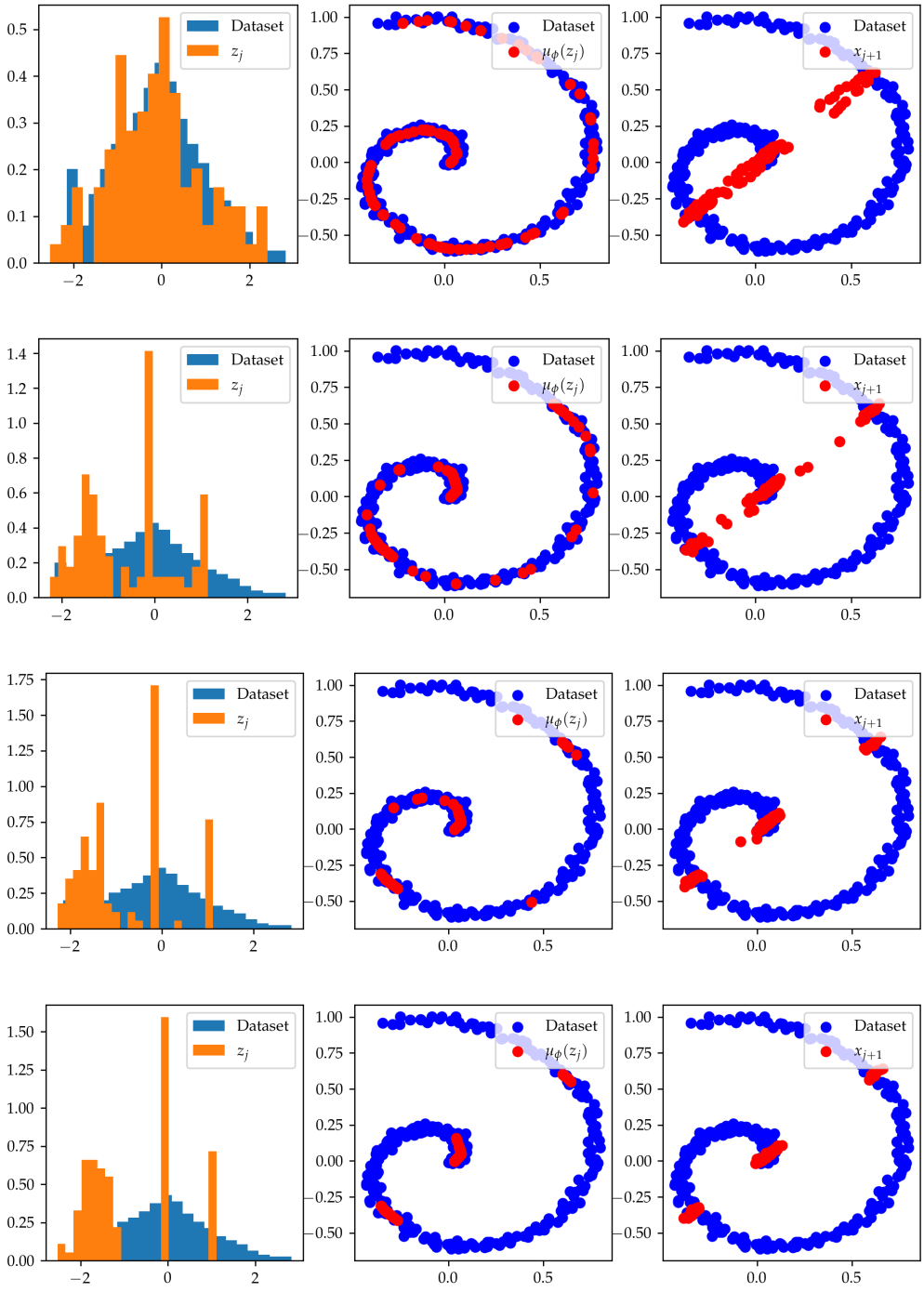


Figure 5.1: Behaviour of Algorithm 10. In this toy inverse problem, $\boldsymbol{x} \in \mathbb{R}^2$, $\boldsymbol{A} = [1, -1]$, $\sigma = 0.01$ and $\boldsymbol{y} = 0$ (so $x_1 \simeq x_2$). First column shows the histogram of the values $\boldsymbol{z}^{(i)}$ at each iteration, central column the data samples they generate and updated points $\boldsymbol{x}^{(i+1)}$ after applying the step 4.

Algorithm 8: Metropolis-within-Gibbs with VAE

Require: Initial sample $\boldsymbol{z}^{(0)} \sim \mathcal{N}(0, I)$ and $\boldsymbol{x}^{(0)} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{z}^{(0)}), \gamma^2 I)$.

1: **for** $i := 0$ **to** $N - 1$ **do**

2: Draw $\tilde{\boldsymbol{z}} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\boldsymbol{x}^{(i)}), \operatorname{diag}(\sigma_\phi^2))$ (proposal for $p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \,\middle|\, \boldsymbol{x}^{(i)}\right)$)

3: Compute the *Metropolis-Hastings acceptance probability*:

$$\rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}) = \min\left\{ \frac{p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x}^{(i)} \,\middle|\, \tilde{\boldsymbol{z}}\right) p_{\mathcal{Z}}\left(\tilde{\boldsymbol{z}}\right)}{p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x}^{(i)} \mid \boldsymbol{z}^{(i)}\right) p_{\mathcal{Z}}\left(\boldsymbol{z}^{(i)}\right)} \frac{q_\phi(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)})}{q_\phi(\tilde{\boldsymbol{z}}|\boldsymbol{x}^{(i)})}, 1 \right\}$$

4: Accept or reject $\tilde{\boldsymbol{z}}$:

$$\boldsymbol{z}^{(i+1)} = \begin{cases} \tilde{\boldsymbol{z}} & \text{with probability } \rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}}) \\ \boldsymbol{z}^{(i)} & \text{otherwise.} \end{cases} \tag{5.31}$$

5: Draw $\boldsymbol{x}^{(i+1)} \sim p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x} \,\middle|\, \boldsymbol{y}, \boldsymbol{z}^{(i+1)}\right)$ (Proposition 7)

6: **end for**

7: **return** Markov chain $\{(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})\}_{i=1,\ldots,N}$.

Algorithm 9: Approximate sampling from $p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \mid \boldsymbol{x}\right)$

Require: $\boldsymbol{x} \in \mathbb{R}^d$, encoder $q_\phi(\cdot \mid \boldsymbol{x})$, n_{Rprop} number of Rprop steps [181].

1: $(\boldsymbol{\mu}_\phi, \sigma_\phi^2) = \text{Encoder}(\boldsymbol{x})$ (corresponds to $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{\mu}_\phi, \operatorname{diag}(\sigma_\phi^2))$)

2: To reduce the gap between $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ and $p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \mid \boldsymbol{x}\right)$, apply n_{Rprop} Rprop iterations to

$$\begin{aligned} F(\boldsymbol{z}) &:= -\log p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x} \mid \boldsymbol{z}\right) - \log p_{\mathcal{Z}}\left(\boldsymbol{z}\right) && (5.32) \\ &= \frac{1}{2\gamma_{\boldsymbol{x}}^2}\|\boldsymbol{x} - \boldsymbol{\mu}_\theta(\boldsymbol{z})\|^2 + \frac{1}{2}\|\boldsymbol{z}\|^2 && (5.33) \\ &\propto -\log p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \mid \boldsymbol{x}\right) && (5.34) \end{aligned}$$

starting at $\boldsymbol{\mu}_\phi$. Let $\widetilde{\mu}$ be the output.

3: **return** $\boldsymbol{z} \sim \mathcal{N}(\widetilde{\mu}, \operatorname{diag}(\sigma_\phi^2))$.

Algorithm 10: Pseudo-Gibbs sampling with VAE - Single Chain

Require: Measurements $\boldsymbol{y}$, initial guess $p^{(0)}(\boldsymbol{x})$ of $p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x} \,|\, \boldsymbol{y}\right)$.

1: $\boldsymbol{x}^{(0)} \sim p^{(0)}(\boldsymbol{x})$
2: **for** $i := 0$ **to** $N-1$ **do**
3: $\quad \boldsymbol{z}^{(i+1)} \sim p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \,\middle|\, \boldsymbol{x}^{(i)}\right)$ (using Algorithm 9)
4: $\quad \boldsymbol{x}^{(i+1)} \sim p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x} \,\middle|\, \boldsymbol{y}, \boldsymbol{z}^{(i+1)}\right)$
5: **end for**
6: **return** Markov chain $\{(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})\}_{i=1,\ldots,N}$.

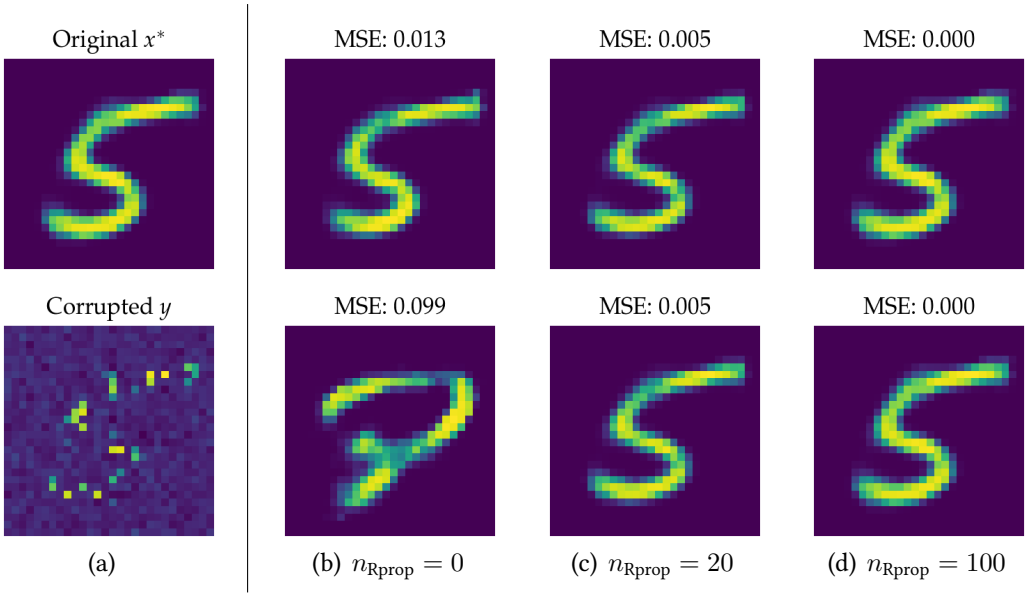
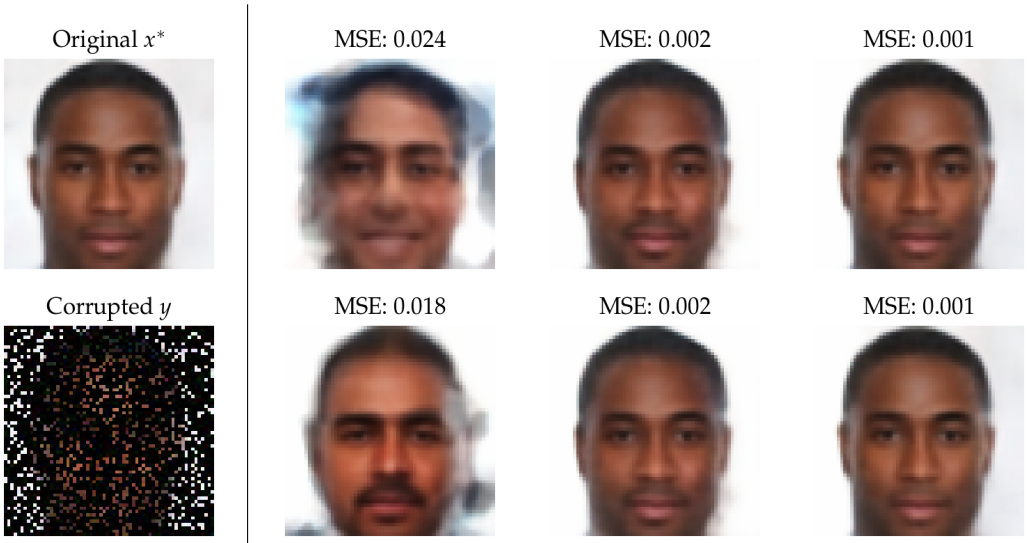


Figure 5.2: Closing the gap $q_\phi(\boldsymbol{z}|\boldsymbol{x}) \simeq p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \,|\, \boldsymbol{x}\right)$: *(a)* Original image $\boldsymbol{x}^*$ (chosen to be on the range of the decoder, that is $\boldsymbol{x}^* = \boldsymbol{\mu}_\theta(\boldsymbol{z}^*)$) and corrupted version $\boldsymbol{y}$ (80% of random missing pixels with Gaussian noise $\sigma = 10/255$). *(b)* We run 1000 iterations of pseudo-Gibbs algorithm (without gap correction) and compute the mean of last 100 iterations. In this column we show the obtained result computed on two independent chains initialized at the *same* $\boldsymbol{x}^{(0)}$. *(c)* Same as (b) but using Algorithm 9 with $n_{\text{Rprop}} = 20$ Rprop steps. *(d)* Same as (b) but using Algorithm 9 with $n_{\text{Rprop}} = 100$ Rprop steps. Note how increasing n_{Rprop} the sampling step in the pseudo-Gibbs algorithm is more stable and generates samples that are more faithful to the target distribution.



(a) (b) $n_{\text{Rprop}} = 0$ (c) $n_{\text{Rprop}} = 20$ (d) $n_{\text{Rprop}} = 100$

Figure 5.3: Closing the gap $q_\phi(\boldsymbol{z}|\boldsymbol{x}) \simeq p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$: Same experiment as in Figure 5.2 but on CelebA with DCGAN-like [174] architectures for encoder and decoder.

distribution but a similar one, as stated in [180, Prop. F.1]. Moreover, starting at the *same* initial sample $\boldsymbol{x}^{(0)}$ we get very different results, showing that there may be instabilities on the iterations of the pseudo-Gibbs algorithm (Algorithm 9, see also Figure 5.4). Using a gap correction only with $n_{\text{Rprop}} = 20$ iterations we get stable and more accurate results. With $n_{\text{Rprop}} = 100$ iterations we get even more accurate results but at the cost of larger runtimes.

Also, in Figure 5.5 we see the effect of this gap correction, applied to the proposal distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, on the acceptance rate of the Metropolis-Hastings step of Algorithm 8. For larger values of n_{Rprop}, the acceptance rate increases due to the fact that the proposal $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is closer to the true posterior $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$, at the expense of higher computational costs.

To empirically validate that the algorithm converges, we compute the following statistics at iteration i:

$$M_j(\boldsymbol{x}_j^{(i)}) = \frac{1}{m}\sum_{j=1}^{m} \boldsymbol{x}_j^{(i)} \tag{5.35}$$

$$V_j(\boldsymbol{x}_j^{(i)}) = \frac{1}{m}\sum_{j=1}^{m} (\boldsymbol{x}_j^{(i)} - M_j(\boldsymbol{x}_j^{(i)}))^2 \tag{5.36}$$

and analogous for $\boldsymbol{z}_j^{(i)}$. After completing N iterations we compare these statistics at iteration i and N for each i. We show the results in Figure 5.6.

It is well known that, when using standard Gibbs samplers, the chain can move slowly when the variables are correlated [22] as in our case between $\boldsymbol{x}$ and $\boldsymbol{z}$. In

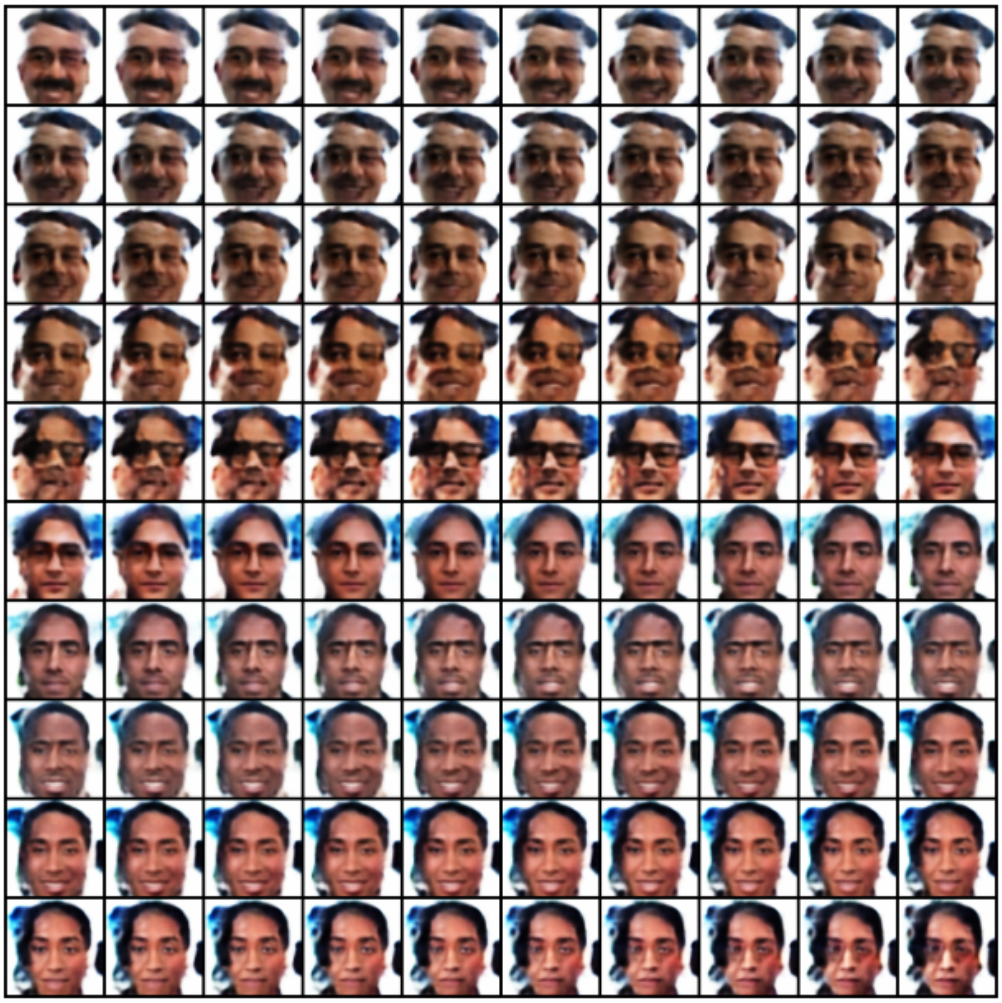
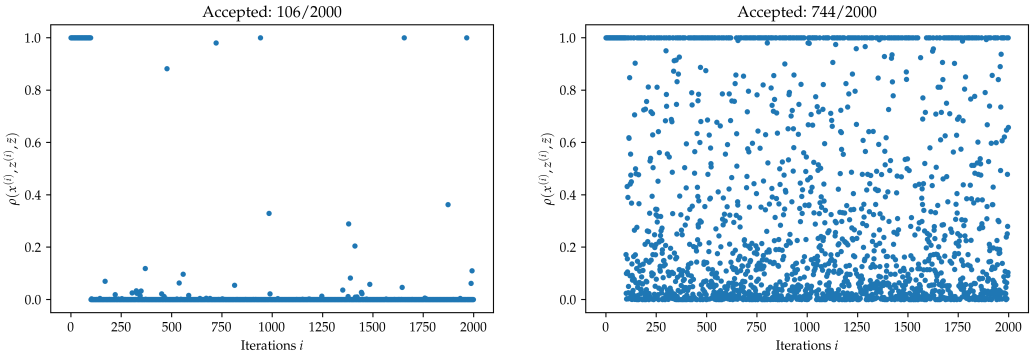
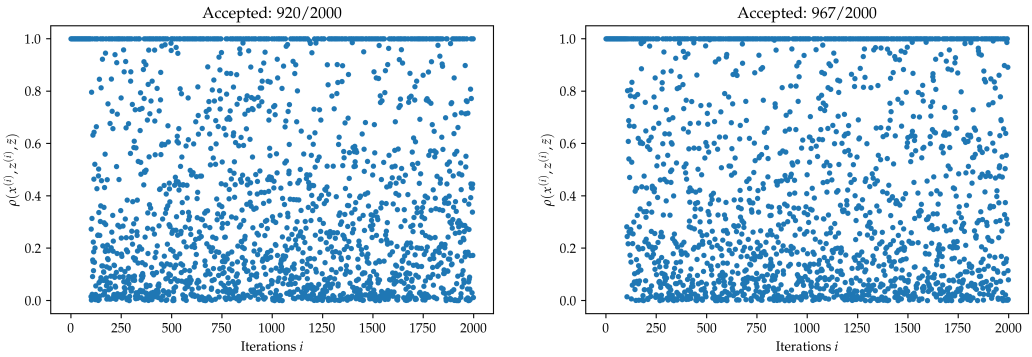


Figure 5.4: Instabilities of the pseudo-Gibbs algorithm without gap correction: 100 last samples (out of 1000) of a chain generated by Algorithm 10 with $n_{\text{Rprop}} = 0$. The mean of these samples is the image shown on column (b), first row of Figure 5.3



(a) $n_{\text{Rprop}} = 0$ (b) $n_{\text{Rprop}} = 10$



(c) $n_{\text{Rprop}} = 20$ (d) $n_{\text{Rprop}} = 100$

Figure 5.5: Plot of $\rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{z}})$ and acceptance rate of Metropolis-within-Gibbs (Algorithm 8) using gap correction of proposal $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ (Algorithm 9), for different values of n_{Rprop}. The first 100 iterations are pseudo-Gibbs samples (to accelerate burn-in and avoid early rejections) and are always accepted. For larger values of n_{Rprop}, the acceptance rate increases as the proposal is closer to the true posterior $p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x})$.

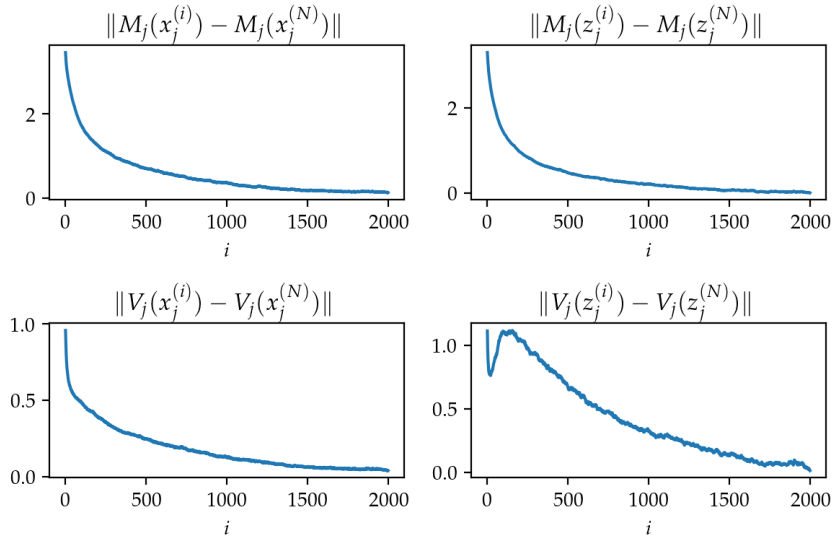


Figure 5.6: *Monitoring convergence of Algorithm 10 using* (5.35) *and* (5.36). We see that the chains generated by the algorithm stabilizes as the number of iterations grows.

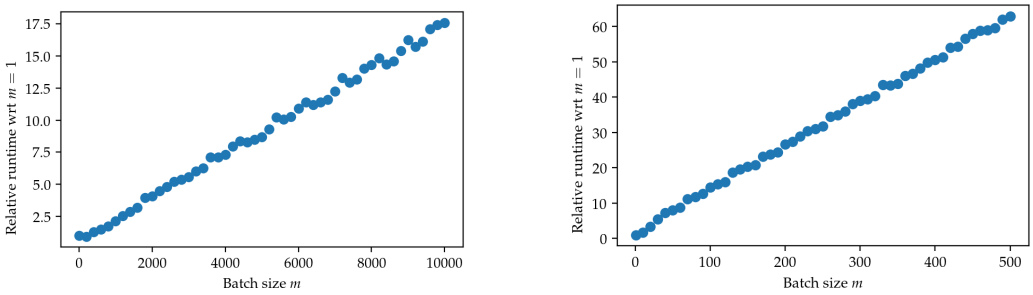
particular, the proposed point $\tilde{\boldsymbol{z}}$ in step 2 of Algorithm 8 brings new information about $\boldsymbol{y}$ from $\boldsymbol{x}^{(i)}$ (sampled on step 5 at the previous iteration) which may not be in the range of the decoder. So $p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x}^{(i)} \,\middle|\, \tilde{\boldsymbol{z}}\right)$ may be small and the new sample $\tilde{\boldsymbol{z}}$ may get rejected. Another approach is to apply Metropolis-Hastings steps directly to $p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{x}, \boldsymbol{z} \,|\, \boldsymbol{y}\right)$. That is, instead of accepting or rejecting $\boldsymbol{z}$ inmediately after step 2 of Algorithm 8, we sample both variables $(\boldsymbol{x}, \boldsymbol{z})$ and *then* accept or reject the whole pair. This idea is similar to the *delayed rejection method* proposed in [217]. In order to do so we need a jumping distribution $J(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})$. We propose to concatenate the previous conditional distributions as follows:

$$J(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}) = q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x}^{(i)}) p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x} \,|\, \boldsymbol{y}, \boldsymbol{z}\right). \tag{5.37}$$

In other words, we first sample $\tilde{\boldsymbol{z}} \sim q_\phi(\boldsymbol{z} \,|\, \boldsymbol{x}^{(i)})$, then $\tilde{\boldsymbol{x}} \sim p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x} \,|\, \boldsymbol{y}, \tilde{\boldsymbol{z}}\right)$ and compute the acceptance ratio as follows:

$$\rho(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}) = \frac{p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}\left(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}} \,|\, \boldsymbol{y}\right)}{p_{\mathcal{X},\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} \,|\, \boldsymbol{y}\right)} \frac{J(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}|\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}})}{J(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}}|\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})} \tag{5.38}$$

$$= \frac{p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \,|\, \tilde{\boldsymbol{x}}\right) p_{\mathcal{X}|\mathcal{Z}}\left(\tilde{\boldsymbol{x}} \,|\, \tilde{\boldsymbol{z}}\right) p_{\mathcal{Z}}\left(\tilde{\boldsymbol{z}}\right)}{p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \,|\, \boldsymbol{x}^{(i)}\right) p_{\mathcal{X}|\mathcal{Z}}\left(\boldsymbol{x}^{(i)} \,|\, \boldsymbol{z}^{(i)}\right) p_{\mathcal{Z}}\left(\boldsymbol{z}^{(i)}\right)} \frac{q_\phi(\boldsymbol{z}^{(i)} \,|\, \tilde{\boldsymbol{x}}) p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}^{(i)} \,\middle|\, \boldsymbol{y}, \boldsymbol{z}^{(i)}\right)}{q_\phi(\tilde{\boldsymbol{z}} \,|\, \boldsymbol{x}^{(i)}) p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\tilde{\boldsymbol{x}} \,|\, \boldsymbol{y}, \tilde{\boldsymbol{z}}\right)}. \tag{5.39}$$



(a) MNIST (b) CelebA

Figure 5.7: Runtimes of 1000 iterations of $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi([\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]))$, relative to $m = 1$, for different values of batch size m. The VAE model trained on MNIST has 2 fully-connected hidden layers as encoder and decoder. The one trained in CelebA has architectures for encoder and decoder similar to DCGAN [174]. The experiment was run on a GTX 1080 Ti GPU. This benchmark was made only varying the batch sizes, but we believe that higher improvements can be achieved with additional optimizations.

5.2.3 Parallel multiple Markov Chains

Theoretically, the ergodicity of the MCMC algorithms described above ensures that the generated chains converge to the target distribution as the number of iterations grows. In practice, however, when truncating the sampling process this asymptotic result can lead to misleading conclusions. As a simple example, when the target distribution has two modes separated by a large region of zero or very small probability, then a chain can get stuck in one mode and not jump to the other in a reasonable amount of time.

To overcome the main issues of single-chain MCMC algorithms, that is slow mixing and correlation of the samples, a possible solution is to generate $m > 1$ independent chains, starting from different points $\boldsymbol{x}_j^{(0)}$, $j = 1, \dots, m$. The appealing of this approach is that we can run multiple chains almost *for free* using GPU parallelization (to some extent). To confirm this intuition, we run a simple benckmark which consists in computing $\boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi([\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]))$ for different values of m. That is, for a batch of m images we compute the encoder mean following the decoder mean. Figure 5.7(a) shows that, on MNIST and in our specific setup, computing the encoder and decoder means of $m \simeq 570$ chains, which are the bottleneck of the algorithms considered in this work, is as cheap as computing only 1. The same is true on CelebA (with a more complex architecture) for $m = 8$.

The idea of using multiple sequences for posterior sampling is equivalent to run gradient descent algorithms for minimization problems, starting from several initializations, where only one initialization may not be sufficient to reach the global

minimum. In both cases, the starting distribution from which we draw the first samples must be *overdispersed*, which means that it must cover the target distribution or have more variance to have a good chance of visiting all its modes of it after several runs. Here, we borrow some ideas from [80] (see also [83, Chapter 8]).

Two (or more) chains are better than one

There are some detractors for the multiple chain approach. For example, Charles Geyer says:

If you can't get a good answer with one long run, then you can't get a good answer with many short runs either.[1]

The main argument is that, if you have a budget of mn iterations (samples) of your favorite sampling scheme, then it is better to spend it all in a single chain than to generate n samples of m independent chains, as the lack of convergence of the first *long* chain ensures that of the *shorter* ones. Although this claim is theoretically coherent and can be supported with simple examples, we respond with the following ones:

1. *There may not be a long enough chain*: Although is relatively easy to construct examples in which a long chain covers the whole target distribution, in modern practical applications which usually involve high-dimensional spaces, to generate a *long enough* chain is not feasible, e.g. as is the case for complex models such as the one involved in realistic climate estimation [204].

2. *We can generate multiple chains for free*: As mentioned before, if we can generate a chain of length n then we also can construct several independent Markov chains at roughly the same cost as constructing only one, and therefore multiplying the number of samples drawn at low computational cost.

This procedure not only permits the generation of more (approximate) samples from the target distribution and hence to accelerate the convergence of the corresponding integral estimators, but also to better diagnose convergence of the sampling algorithm, a fact that may not be done using only one sequence [79]. The main problem is that one can get stuck in an isolated mode of the target distribution for a long time, and to falsely believe that the chain converged. To illustrate the benefits of the multiple chain approach, we consider the following example.

[1] *One Long Run in MCMC* by Charles Geyer: http://users.stat.umn.edu/~geyer/mcmc/one.html

Dimension d	1	2	3	4	5	6	7	8	9	10
# accepted	48839	16862	6266	2627	1041	503	216	140	67	0
# jumps	6993	1768	426	92	17	11	1	1	0	0

Table 5.1: *Number of accepted samples and jumps between modes for the Metropolis-Hastings algorithm for different dimension values d, for the target distribution defined in* (5.40). The number of iterations is $n = 100.000$ in each case. As the dimension d increases, the acceptance rate and the number of jumps between modes decrease rapidly.

Let the target distribution consist of the mixture of two Gaussian modes:

$$f(\boldsymbol{x}) \propto \lambda \exp\left[-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i-0.5)^2}{\sigma_1^2}\right] + (1-\lambda)\exp\left[-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i+0.5)^2}{\sigma_2^2}\right], \quad (5.40)$$

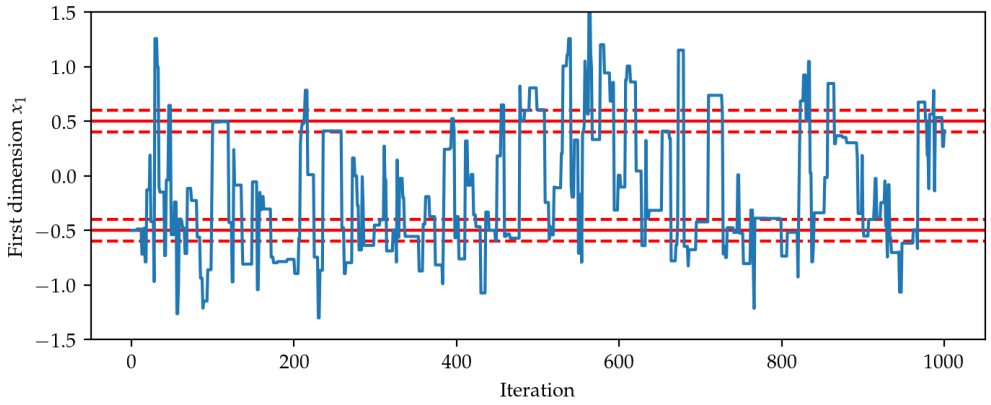
where $\boldsymbol{x} \in \mathbb{R}^d$ and $\lambda \in [0,1]$ is a mixing coefficient. We aim at jumping between the two modes so we initialize a random walk Metropolis-Hastings algorithm (Algorithm (7)) at one mode with a spherical Gaussian distribution as a proposal distribution $J(\boldsymbol{x} \mid \boldsymbol{x}') = \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}', \alpha^2 I)$ as in (5.10). Observe that the distance between the means of the two modes is $\sqrt{d}$. As Gaussian samples drawn from $\mathcal{N}(0, I)$ in a high-dimensional space concentrate in a sphere of radius $\sqrt{d}$ [228] and $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{x};\boldsymbol{x}',I)}[\|\boldsymbol{x}-\boldsymbol{x}'\|^2] = d$, we choose $\alpha = 1$.

In Figure 5.8 we show how the samples generated by this procedure evolve with the iteration. We see that the acceptance rate decreases as the dimension d increases. Also, the number of jumps between modes goes to zero rapidly. In Table 5.1 we show quantitative results for dimensions ranging between 1 and 10 which confirm this behavior. Here, a *jump* is defined as an accepted sample $\boldsymbol{x}^{(i+1)} \sim J(\boldsymbol{x} \mid \boldsymbol{x}^{(i)})$ at distance larger than $\sqrt{d}$ from the starting point $\boldsymbol{x}^{(i)}$ (that is, the nearest mean μ_i changes).

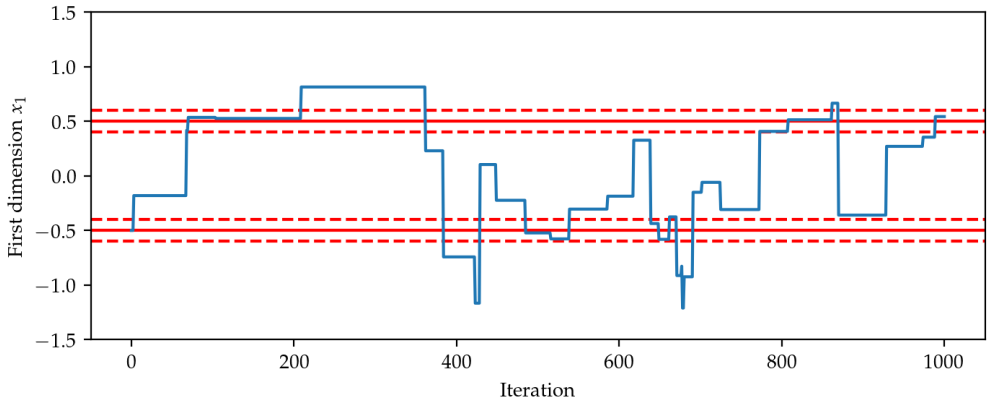
In our particular case, if the posterior distribution has several isolated modes, it will be difficult for the sampling algorithms to explore all of them. A possible solution is to compute *tempered transitions* [158] for sampling from $p_{\mathcal{Z}|\mathcal{Y}}(\boldsymbol{z} \mid \boldsymbol{y})$ using the following intermediate distributions:

$$p_T(\boldsymbol{z} \mid \boldsymbol{y}) \propto p_{\mathcal{Y}|\mathcal{Z}}(\boldsymbol{y} \mid \boldsymbol{z})^T \, p_{\mathcal{Z}}(\boldsymbol{z}) \quad (5.41)$$

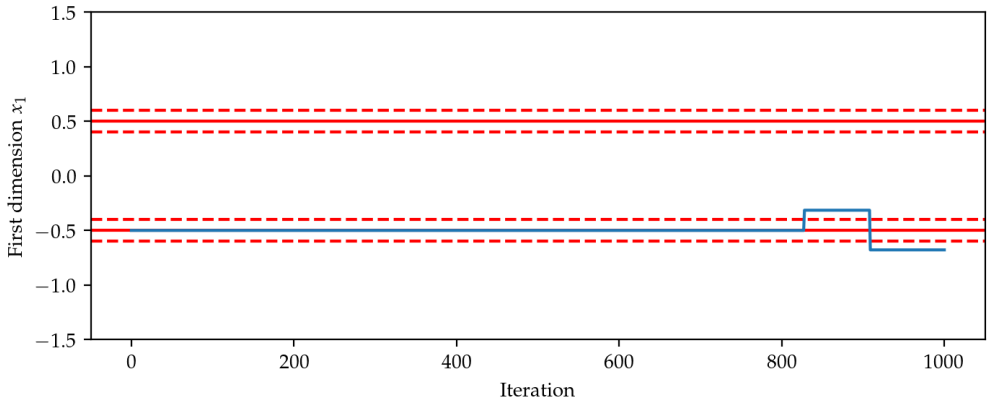
where $T = 0$ gives the (unimodal) prior $p_{\mathcal{Z}}(\boldsymbol{z})$ and for $T = 1$ the full posterior. In [102] the authors propose to run several independent Metropolis-Hastings chains with target distribution $p_T(\boldsymbol{z} \mid \boldsymbol{y})$ for a set of intermediate temperatures $0 \leq T_1 < \ldots < T_p \leq 1$. For $T \simeq 0$ it is easier to explore the whole space $\mathcal{Z}$ and when $T \to 1$ the chains get stuck in one of the posterior modes. To exploit this behaviour, the authors perform swaps between samples from chains at temperatures T_i and T_{i+1}. We explore similar tempering ideas in Section 5.3.



(a) Dimension $d = 2$



(b) Dimension $d = 4$



(c) Dimension $d = 6$

Figure 5.8: *1000 samples generated by a random walk Metropolis-Hastings algorithm for the target distribution described in* (5.40). For this experiment we choose $\lambda = 0.5$, $\sigma_1 = \sigma_2 = 0.1$ and the proposal distribution (5.10) with zero mean and identity covariance for different values of d. Here we only display the first coordinate $x_1^{(i)}$ of each iterate $\boldsymbol{x}^{(i)}$ with blue lines. The red lines represent $\mu_k \pm \sigma_k$. Note that, as the dimension d increases, the acceptance rate and the number of jumps between modes decrease rapidly.

The multiple chain approach to avoid getting stuck in posterior modes for a long time requires to run several independent chains initialized using an overdispersed distribution. The prior distribution $p_{\mathcal{X}}(\boldsymbol{x})$ covers $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$ so is appropiate to use it for initializing the sampling scheme in Algorithm 10. Then, we can repeat m times the algorithm proposed in Section 5.2.2 to generate i.i.d. samples $\{\boldsymbol{x}_j^{(i)}\}_{j=1,\dots,m}$ of $p^{(i)}$ which converge to $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$ when $i \to \infty$. If we run each chain for N iterations, at the end (assuming convergence) we obtain m i.i.d. samples of the target distribution but also the last samples of each chain can be used to compute estimators. The proposed algorithm is summarized in Algorithm 11.

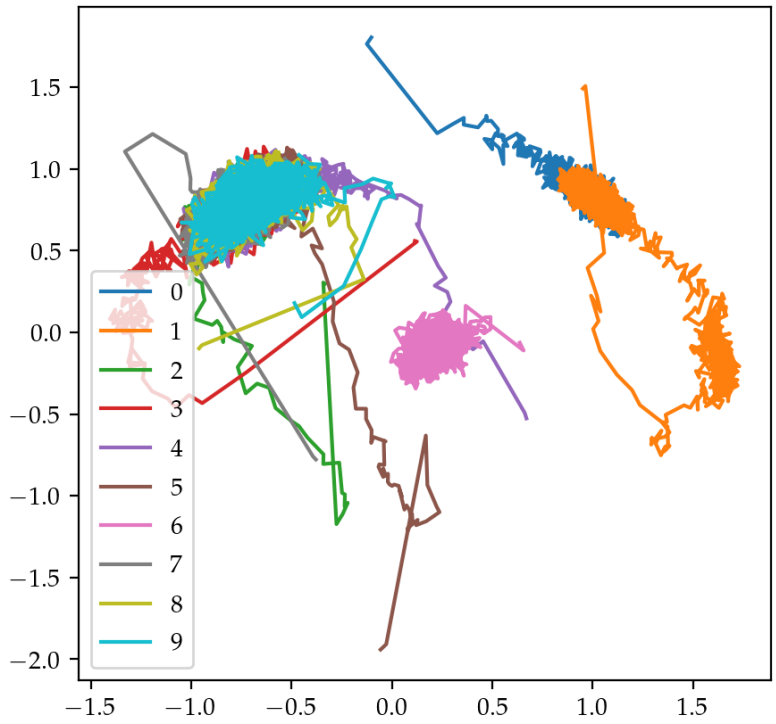
Algorithm 11: Pseudo-Gibbs sampling with VAE - Multiple chains

Require: Measurements $\boldsymbol{y}$, initial guess $p^{(0)}(\boldsymbol{x})$ of $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$.

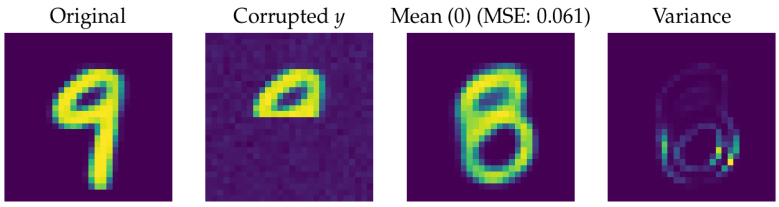
1: $\boldsymbol{x}_1^{(0)}, \dots, \boldsymbol{x}_m^{(0)} \sim p^{(0)}(\boldsymbol{x})$
2: **for** $j := 1$ **to** m **do**
3: **for** $i := 0$ **to** $N - 1$ **do**
4: $\boldsymbol{z}_j^{(i+1)} \sim p_{\mathcal{Z}|\mathcal{X}}\left(\boldsymbol{z} \,\middle|\, \boldsymbol{x}_j^{(i)}\right)$ (using Algorithm 9)
5: $\boldsymbol{x}_j^{(i+1)} \sim p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x} \,\middle|\, \boldsymbol{y}, \boldsymbol{z}_j^{(i+1)}\right)$
6: **end for**
7: **end for**
8: **return** $\{\boldsymbol{x}_j^{(N)}\}_{j=1,\dots,m}$ i.i.d. samples of $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$.

In the Bayesian framework, the posterior distribution is the complete solution of the inverse problem, so intead of giving only point estimates we can output more information about the space of feasible restorations. In less ill-posed problems the posterior distribution will be peaked at only one mode and all the chains will eventually find it in a few iterations. On the other hand, for more ill-posed problems as the one shown in Figure 5.9 the posterior may have multiples modes and several chains may cover them when properly initialized. Summarizing all the modes in only one point estimate (e.g. the MMSE estimator) may not be a good idea. Alternatively, as this procedure permits to detect isolated modes we can output several possible restorations.

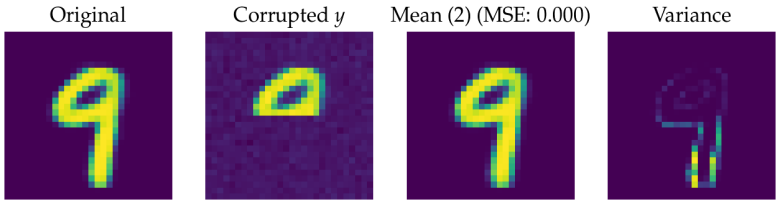
In Figure 5.9 we show how independent chains, although not *converging* to the posterior distribution in the sense of MCMC algoritms, can give multiple solutions (in this example 3 coherent restorations). Here we try to recover the image given the upper half (see *(b)*) using a VAE with latent dimension 8 as the one used in Chapter 4. In *(a)* we show the first 2 dimensions on $\mathcal{Z}$ space for 10 independent chains of length $n = 50.000$ initialized with $\boldsymbol{z} \sim \mathcal{N}(0, I)$ generated by Algorithm 10. As we see, most of them find a posterior mode and do not jump to other ones but explore that they find. We can identify mainly 3 modes, and for each one we show in *(b)*, *(c)* and *(d)* the mean and variance of corresponding chains (after discarding some burn-in iterations at the beginning).



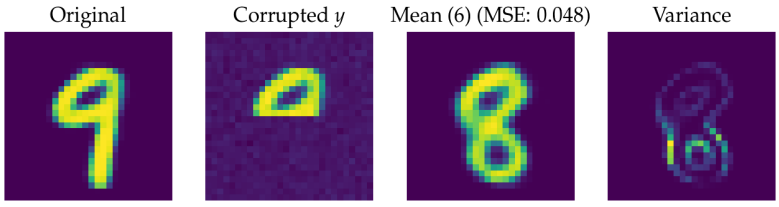
(a) First 2 dimensions of $\mathcal{Z}$



(b) Mean and variance for chain #0 (blue)



(c) Mean and variance for chain #2 (green)



(d) Mean and variance for chain #6 (pink)

Figure 5.9: *Multiple chains for better posterior exploration* (see text for a detailed description).

Another approach is to combine the chains and share information between them as the iterations evolve, as was originally proposed in [214]. In particular, the Sampling-Importance-Resampling (SIR) scheme [190] proposed to weight the samples $\boldsymbol{x}_j^{(i)}$ at iteration i of all the chains and then perform a resampling step. After resampling, the intermediate distribution $p^{(i)}$ of the samples $\boldsymbol{x}_j^{(i)}$ is closer to the target distribution than before resampling, thus it may accelerate the convergence of the whole algorithm. Resampling ideas are explored in the next section.

5.3 Annealed Importance Sampling with Resampling

A common problem with accept/reject algorithms such as Metropolis-Hastings is that they may reject a huge amount of samples before the chain has reached a region with significant posterior mass (burn-in). Instead of throwing away samples, *Importance Sampling* [184] weigths each sample drawn from a proposal distribution by assigning to it a weight (or score) depending on how well it fits the target distribution. The effectiveness of this approach depends on how far the proposal distribution is from the target distribution. First, we begin by describing the importance sampling approach and how it could be combined with annealing techniques to accelerate the burn-in period.

5.3.1 Importance Sampling

When we cannot sample directly from the target distribution f but we have a proposal distribution g with $\operatorname{supp}(f) \subset \operatorname{supp}(g)$ which is easy to sample from and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ are samples from g, then

$$\mathbb{E}_f[a(\boldsymbol{x})] = \mathbb{E}_g\left[\frac{f(\boldsymbol{x})}{g(\boldsymbol{x})}a(\boldsymbol{x})\right] \simeq \frac{1}{m}\sum_{j=1}^{m} w_j a(\boldsymbol{x}_j) \tag{5.42}$$

where

$$w_j = \frac{f(\boldsymbol{x}_j)}{g(\boldsymbol{x}_j)} \tag{5.43}$$

is the *importance weight* associated to sample $\boldsymbol{x}_j$ and compensates the fact that it was sampled from g instead of f.

The accuracy of the resulting estimator (5.42) depends on the variability of the importance weights [184]. Intuitively, we want that w_j does not vary so much so all the samples will have a similar contribution in (5.42). Therefore, we have to choose g close enough to f. In general, how to choose a good proposal distribution g is not straightforward.

Another estimator which has a small bias but is still more stable is

$$\mathbb{E}_f[a(\boldsymbol{x})] \simeq \frac{\sum_{j=1}^m w_j a(\boldsymbol{x}_j)}{\sum_{j=1}^m w_j} = \sum_{j=1}^m w_j^* a(\boldsymbol{x}_j) \tag{5.44}$$

where $w_j^* = w_j / \sum_{k=1}^m w_k$ are the normalized weights. Due to the normalization in (5.44), it is sufficient to calculate each coefficient $w_j = c\tilde{w}_j$ up to a constant c.

5.3.2 Bridging between prior and posterior densities

With the Annealed Importance Sampling (AIS) approach [157], we can construct a family of densities f_i between $f_0 = g$ and $f_n = f$ such that each f_{i-1} is close to f_i. Also, we define transition kernels $T_i(\boldsymbol{x}, \boldsymbol{x}')$ which propagate samples from f_0 to f_n in a sequential manner, mapping samples from f_{i-1} to samples of f_i at each iteration.

Returning to our problem of interest, we need to sample from $f_n(\boldsymbol{x}) = p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$. Choosing a sequence $0 = \beta_0 < \beta_1 < \ldots < \beta_n = 1$ we define

$$f_i(\boldsymbol{x}) := \frac{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_i} p_{\mathcal{X}}(\boldsymbol{x})}{Z_i}, \quad i = 0, \ldots, n \tag{5.45}$$

where $Z_i = \int p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_i} p_{\mathcal{X}}(\boldsymbol{x})\, d\boldsymbol{x}$. In our setting, it is easy to sample from $f_0(\boldsymbol{x}) = p_{\mathcal{X}}(\boldsymbol{x})$ (eg. taking samples from the dataset where the VAE was trained on). Observe that

$$-\log(p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_i}) = \frac{1}{2(\sigma^2/\beta_i)} \|\boldsymbol{y} - A\boldsymbol{x}\|^2 \tag{5.46}$$

and then f_i corresponds to the posterior distribution of the following relaxed inverse problem:

$$\boldsymbol{y} = A\boldsymbol{x}^* + \eta, \quad \eta \sim \mathcal{N}(0, \sigma_i^2), \quad \sigma_i^2 = \frac{\sigma^2}{\beta_i}. \tag{5.47}$$

Now we need to define the transition kernels $T_i(\boldsymbol{x}', \boldsymbol{x})$ that leave the corresponding f_i invariant, and each T_i represents the probability of moving from $\boldsymbol{x}'$ to $\boldsymbol{x}$ at iteration i on a Markov chain setting. We do not need to be able to compute $T_i(\boldsymbol{x}', \boldsymbol{x})$ but to sample $\boldsymbol{x}$ given $\boldsymbol{x}'$. In our case, we adapt the kernels $h(\boldsymbol{x}', \boldsymbol{x})$ from (5.21) to define

$$p_i(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) \propto p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_i} p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z}) \tag{5.48}$$

$$h_i(\boldsymbol{x}', \boldsymbol{x}) = \int p_i(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z}) p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \boldsymbol{x}')\, d\boldsymbol{z}. \tag{5.49}$$

As in equation (5.20), it is easily shown that

$$f_i(\boldsymbol{x}) = \int h_i(\boldsymbol{x}', \boldsymbol{x})\, f_i(\boldsymbol{x}')\, d\boldsymbol{x}, \tag{5.50}$$

so h_i leaves f_i invariant. Hence, we can define T_i as the concatenation of $K_i \geq 1$ sampling steps from h_i so that the resulting Markov chain is invariant with respect to T_i as required. The distribution $p_i(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{z})$ is Gaussian by (5.47) and Proposition 7.

As in [157], we construct a Markov chain as follows

- Sample $\boldsymbol{x}^{(0)}$ from $f_0(\boldsymbol{x}) = p_{\mathcal{X}}(\boldsymbol{x})$
- Sample $\boldsymbol{x}^{(1)}$ from $T_1(\boldsymbol{x}^{(0)}, \boldsymbol{x})$
- ...
- Sample $\boldsymbol{x}^{(n)}$ from $T_n(\boldsymbol{x}^{(n-1)}, \boldsymbol{x})$.

After having generated the sequence described above, we set $\boldsymbol{x}_j = \boldsymbol{x}^{(n)}$ and compute the corresponding importance weight as

$$w_j = \frac{f_n(\boldsymbol{x}^{(n-1)})}{f_{n-1}(\boldsymbol{x}^{(n-1)})} \frac{f_{n-1}(\boldsymbol{x}^{(n-2)})}{f_{n-2}(\boldsymbol{x}^{(n-2)})} \cdots \frac{f_2(\boldsymbol{x}^{(1)})}{f_1(\boldsymbol{x}^{(1)})} \frac{f_1(\boldsymbol{x}^{(0)})}{f_0(\boldsymbol{x}^{(0)})} \tag{5.51}$$

(see [157] for details). Observe that the above ratios

$$\frac{f_{i+1}(\boldsymbol{x})}{f_i(\boldsymbol{x})} = \frac{Z_i}{Z_{i+1}} \frac{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_{i+1}} p_{\mathcal{X}}(\boldsymbol{x})}{p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_i} p_{\mathcal{X}}(\boldsymbol{x})} \propto p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_{i+1} - \beta_i} \tag{5.52}$$

are easily computed (up to a constant) from the likelihood function $p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})$ evaluated at intermediate points $\{\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(n-1)}\}$ of the Markov chain associated to each $\boldsymbol{x}_j$. To avoid numerical problems, it is better to compute

$$\log w_j = \sum_{i=0}^{n-1} \left[\log \frac{f_{i+1}(\boldsymbol{x}^{(i)})}{f_i(\boldsymbol{x}^{(i)})} \right] + C = \sum_{i=0}^{n-1} \left[(\beta_{i+1} - \beta_i) \log p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \mid \boldsymbol{x}^{(i)}\right) \right] + C. \tag{5.53}$$

We can repeat this procedure to generate independent samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ with corresponding (normalized) importance weights $w_1^*, \ldots, w_m^*$. The proposed method is summarized in Algorithm 12. See Figure 5.10 for an illustration of the algorithm. The returned samples can be used to compute estimators of the form (5.44) or to initialize a multiple chain pseudo-Gibbs algorithm (Algorithm 11).

Algorithm 12: Annealed Importance Sampling with VAE

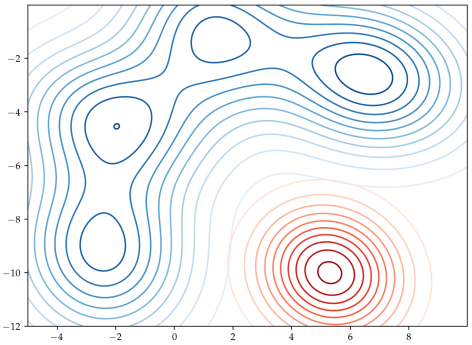
Require: Measurements $\boldsymbol{y}$, sequence $0 = \beta_0 < \beta_1 < \ldots < \beta_n = 1$.

$\boldsymbol{x}_1^{(0)}, \ldots, \boldsymbol{x}_m^{(0)} \sim p_{\mathcal{X}}(\boldsymbol{x})$, $q_1^{(0)} = \ldots = q_m^{(0)} = 0$
for $j := 1$ **to** m **do**
  for $i := 0$ **to** $n - 1$ **do**
    $q_j^{(i+1)} = q_j^{(i)} + (\beta_{i+1} - \beta_i) \log p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \,\middle|\, \boldsymbol{x}_j^{(i)}\right)$ (weight update (5.53))
    $\boldsymbol{x}_j^{(i+1)} \sim T_{i+1}(\boldsymbol{x}_j^{(i)}, \boldsymbol{x})$ (see Algorithm 13)
  end for
end for
Set $\tilde{w}_j = \exp(q_j^{(n)})$ and $w_j^* = \tilde{w}_j / \sum_{k=1}^m \tilde{w}_k$ for $j = 1, \ldots, m$.
return Samples $\{\boldsymbol{x}_1^{(n)}, \ldots, \boldsymbol{x}_m^{(n)}\}$ with corresponding importance weights $w_1^*, \ldots, w_m^*$ of $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$.

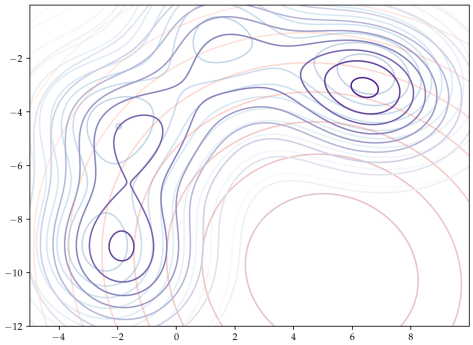
Algorithm 13: Sampling from $T_i(\boldsymbol{x}', \boldsymbol{x})$

Require: Point $\boldsymbol{x}'$, transition kernel T_i, number $K_i \geq 1$ of sampling steps of h_i.

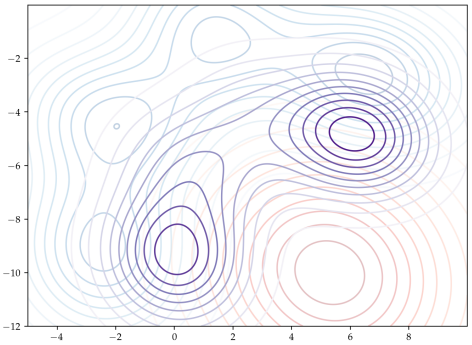
$\tilde{\boldsymbol{x}}_0 = \boldsymbol{x}'$
for $k := 1$ **to** K_i **do**
  $\tilde{\boldsymbol{z}}_k \sim p_{\mathcal{Z}|\mathcal{X}}(\boldsymbol{z} \mid \tilde{\boldsymbol{x}}_{k-1})$ (using Algorithm 9)
  $\tilde{\boldsymbol{x}}_k \sim p_i(\boldsymbol{x} \mid \boldsymbol{y}, \tilde{\boldsymbol{z}}_k)$
end for
return $\tilde{\boldsymbol{x}}_{K_i}$.



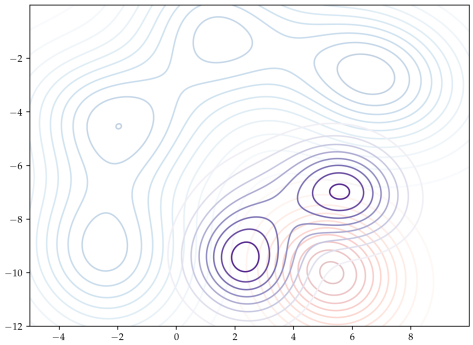
(a) Prior $p_{\mathcal{X}}(\boldsymbol{x})$ (blue) and likelihood $p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})$ (red)



(b) $f_j(\boldsymbol{x})$ for $\beta_j = 0.05$



(c) $f_j(\boldsymbol{x})$ for $\beta_j = 0.30$



(d) $f_n(\boldsymbol{x}) = p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$ for $\beta_n = 1$

Figure 5.10: Interpolation between prior $p_{\mathcal{X}}(\boldsymbol{x})$ and posterior $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$ distributions. *(a)* We first sample from prior $p_{\mathcal{X}}(\boldsymbol{x})$ and incorporate likelihood information smoothly using $p_{\mathcal{Y}|\mathcal{X}}(\boldsymbol{y} \mid \boldsymbol{x})^{\beta_j}$. *(b)* If f_{j-1} is close to f_j then the first one is a good importance sampling proposal for the second. *(c)* As j increase, we propagate the samples through f_j using T_j. *(d)* At last step $\beta_n = 1$ we have (approximate) samples $\boldsymbol{x}^{(i)}$ from $p_{\mathcal{X}|\mathcal{Y}}(\boldsymbol{x} \mid \boldsymbol{y})$ and the corresponding weights $w^{(i)}$.

5.3.3 Weight degeneracy and Resampling

The accuracy of the estimator (5.44) depends on the variance of the importance weights (5.43). A way to estimate this variance is to use the *coefficient of variation* [138]:

$$C^2(w) = \frac{1}{m}\sum_{j=1}^{m}(mw_j^* - 1)^2. \tag{5.54}$$

A common problem arising in AIS which is shared with Sequential Monte Carlo algorithms [65] is the *weight degeneracy*: in general, most of the normalized weights w_j^* vanish and only a few of them concentrate all of the positive values, which causes $C^2(w)$ to be large. As a "rule of thumb" in sampling, it is proposed to calculate the *Effective Sample Size* [138]:

$$\text{ESS}(w) = \frac{m}{1 + C^2(w)}. \tag{5.55}$$

For example, if $m' < m$ coefficients verify $w_j^* = 1/m'$ and the rest verify $w_j^* = 0$, a direct calculation leads to $C^2(w) = \frac{m}{m'} - 1$ and $\text{ESS}(w) = m'$.

Weight updates (5.51) are computed using the likelihood $p_{\mathcal{Y}|\mathcal{X}}\left(\boldsymbol{y} \,\middle|\, \boldsymbol{x}^{(i)}\right)$ evaluated at new samples $\boldsymbol{x}^{(i)}$. If a chain becomes stuck in a posterior mode with little likelihood values, the corresponding weight start to decrease wrt the others which leads to a lower value of $ESS(w)$. To deal with this problem, we can add a resampling step when $ESS(w) < m_0$ for some predefined threshold such as $m_0 = m/2$ (i.e. half of the total number of chains). For the resampling step at iteration i, the naive approach is to sample from $\{\boldsymbol{x}_1^{(i)}, \ldots, \boldsymbol{x}_m^{(i)}\}$ with probabilities given by the normalized weights $\{w_1^*, \ldots, w_m^*\}$. Since this introduces additional variance to the algorithm, other sampling techniques are generally used such as *Multinomial Resampling*. The resampling step stops the generation of the chains with lower fit (as measured by the likelihood function) and replicates the most promising ones. This idea is usually known as *early rejection* in *particle filters* and *bootstrap filters* [64]. An ilustration of resampling is shown in Figure 5.11.

5.4 Preliminary results

In Figure 5.12 we show some preliminary results of some of the algorithms described above. The point estimate $\hat{\boldsymbol{x}}_1$ corresponds to the mean of all the generated samples, which gives a blurry output as it combines all the posterior modes. Observe that we have samples $\boldsymbol{z}_j^{(N)} \sim p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z} \,|\, \boldsymbol{y}\right)$. Thus, we can sort the samples

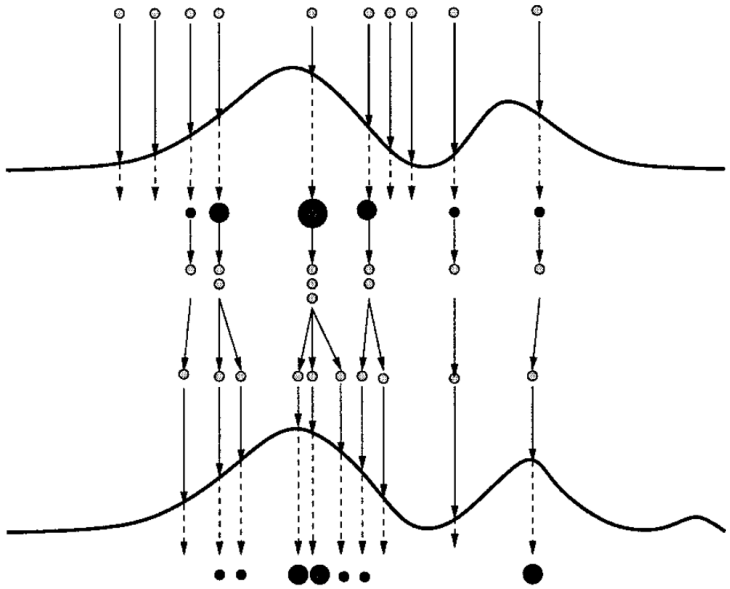


Figure 5.11: *Resampling.* The samples $\boldsymbol{x}_j^{(i)}$ $(j = 1, \ldots, m)$ at iteration i (top) are resampled using the importance weights $w_j^{(i)}$ (bottom) which depend on the likelihood function. Figure adapted from [6].

$\boldsymbol{x}_j^{(N)}$ using

$$p_{\mathcal{X}|\mathcal{Y}}\left(\boldsymbol{x}_i^{(N)} \,\middle|\, \boldsymbol{y}\right) = \int p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}_i^{(N)} \,\middle|\, \boldsymbol{y}, \boldsymbol{z}\right) p_{\mathcal{Z}|\mathcal{Y}}\left(\boldsymbol{z} \,|\, \boldsymbol{y}\right)\, d\boldsymbol{z} \tag{5.56}$$

$$\simeq \frac{1}{m}\sum_{j=1}^{m} p_{\mathcal{X}|\mathcal{Y},\mathcal{Z}}\left(\boldsymbol{x}_i^{(N)} \,\middle|\, \boldsymbol{y}, \boldsymbol{z}_j^{(N)}\right) \tag{5.57}$$

and compute the mean of the better ones (that is, with larger posterior values) as indicated by the dashed line in Figure 5.13. The resulting *trimmed mean* is denoted by $\hat{\boldsymbol{x}}_2$.

In Figure 5.14 we show the effect of the resampling step in the AIS algorithm. Here we can see the weight degeneracy as the number of iterations grows, which leads to a very poor pool of samples $\boldsymbol{x}_j^{(N)}$. By applying resampling when the ESS equals half of the total number of chains, the corresponding weights become more balanced and we end up with a more homogeneous set of samples.

5.5 Conclusions and future work

In this chapter we present a study of sampling algorithms using VAE models. As opposed to generic MCMC algorithms, our approach seeks to leverage the bidirec-

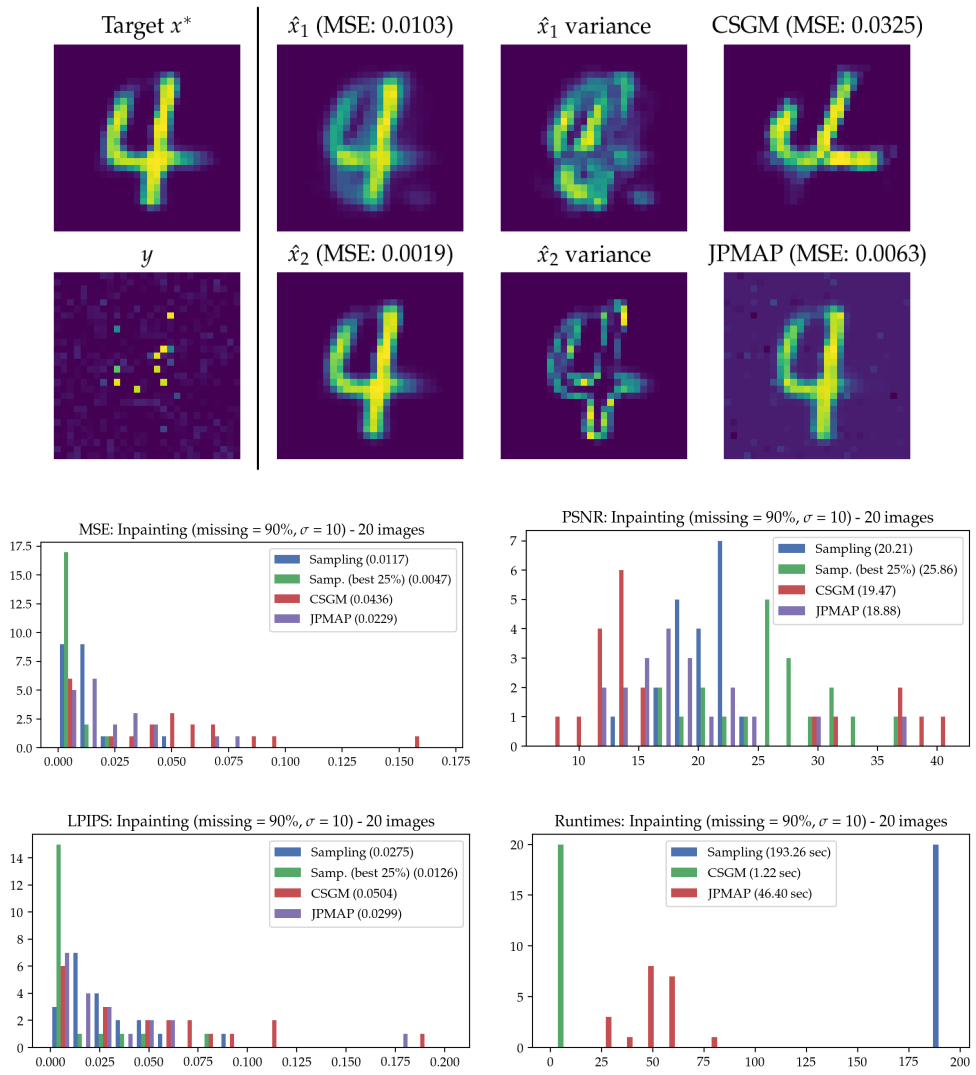


Figure 5.12: *Top:* Result on interpolation (90% of random missing pixels) and noise $\sigma^2 = 10/255$, generated by Algorithm 11. $\hat{\boldsymbol{x}}_1$ corresponds to the mean of all the samples, which gives a blurry output as it combines all the posterior modes. For $\hat{\boldsymbol{x}}_2$ we first sort the samples using (5.57) and compute the mean of the best ones (that is, with larger posterior values) as indicated by the dashed line in Figure 5.13. *Below*: Some preliminary comparisons with CSGM [24] and JPMAP (blue and green bars corresponds to $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$ respectively). In the first three histograms, the horizontal axis represent MSE, PSNR and LPIPS values respectively and the vertical axis is the number of images belonging to each bin (out of 20 images). We also give runtimes of all the algorithms. Algorithm 11 was executed using a prefixed number of iterations as an adecuate stopping criteria for it was not implemented yet.

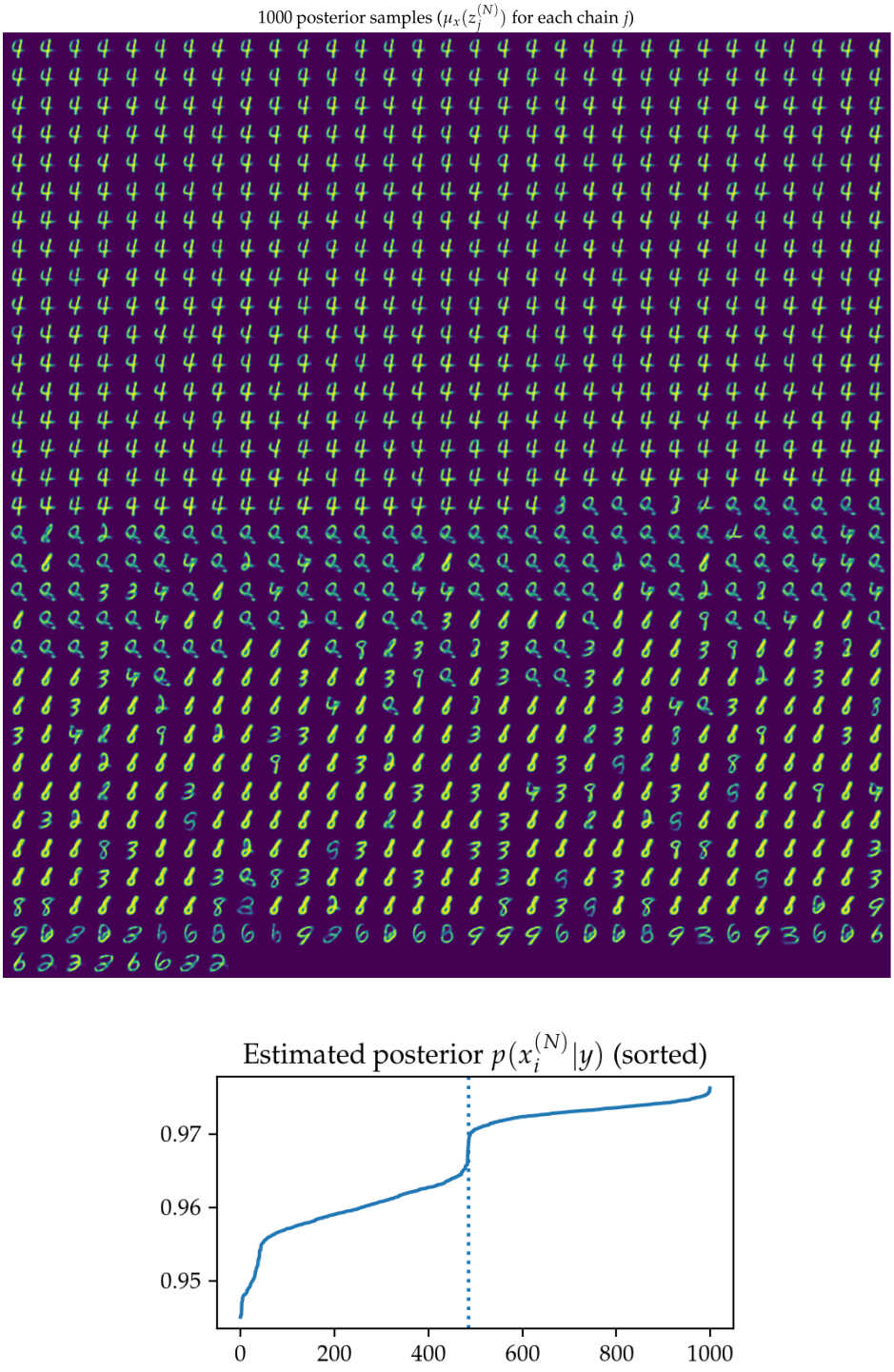


Figure 5.13: Preliminary result on interpolation (90% random missing pixels) with noise $\sigma^2 = 10/255$. *Top:* the last iteration of $m = 1000$ independent chains initialized at random, sorted in decreasing order of their posterior values. *Bottom:* posterior weights for each sample, estimated by (5.57).

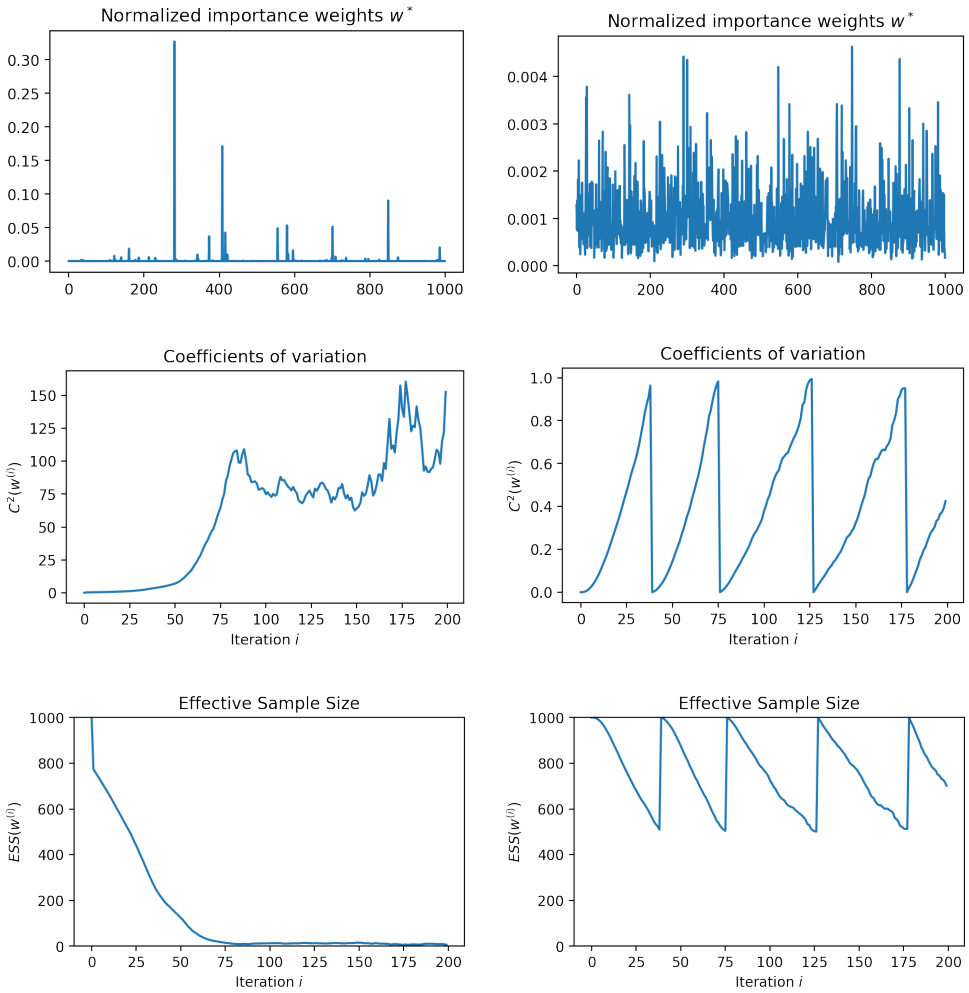


Figure 5.14: *AIS with Resampling*: Result of the AIS algorithm for $m = 1000$ chains, on an interpolation problem with 90% of random missing pixels and noise $\sigma^2 = 10/255$. *Left column:* Without perform resampling, at the last iteration the importance weights concentrate in a few ones, hence the effective sample size (5.55) is very small. *Right column:* We can resample the current iteration of each chain when $C^2(m) \geq 1$ (so $\text{ESS}(w) \leq m/2$) to spend the following iterations on the most promising regions explored by the chains with higher weights.

tional nature of VAEs, in particular the encoder network, to put the $\boldsymbol{y}$ information into the $\mathcal{Z}$ space and hence to construct better proposal distributions for the exploration of the posterior distribution. Also, the multiple chain approach helps to diagnose lack of convergence (e.g. chains get trapped in isolated modes) and to output a more detailed description of the posterior distribution. Finally, resampling techniques permit to reinitialize some chains and hence to spend more resources on exploring promising regions showing larger likelihood values. This can be performed at the initial regime (i.e. with the AIS scheme) or when running the pseudo-Gibbs algorithm.

We leave the following list of topics for future work:

- Although the methods we proposed are based on widely used MCMC algorithms, a complete study of convergence properties (e.g. VAE necesary conditions, convergence rates) is missing.
- Monitoring convergence is an important step to implement efficient stopping criteria. A fully quantitative method is presented in [80] for monitoring convergence of multiple chains. A complete review can be found in [201].
- To initialize the sampling algorithm (e.g. Algorithm 12) we can choose random data points or a representative subset. For example, on MNIST we can choose starting points belonging to all the digit classes. A more general method is to select representative images as in [70]. Also, one can start with a large number of m samples and to progressively select the best fits (e.g. by resampling) as the number of iterations grows.
- More experimental work needs to be done to set the parameters of the algorithms. For example, to set K_i in Algorithm 13 and the number of transition temperatures β_i in Algorithm 12.
- Comparison with other recent sampling algorithms on inverse problems such as [232, 231, 102, 129].
- Extension to other state-of-the-art bidirectional generative models such as Normalizing Flows [179] and Diffusion Models [59].

Chapter 6

Conclusions and future work

In this thesis we reviewed Bayesian methods for solving inverse problems, focusing on regularization methods based on neural networks. To begin, in Chapters 1 and 2 we introduced the main problems of interest and summarized some background material. Next, in Chapter 3 we reviewed Plug-and-Play approaches to leverage implicit priors learned by state-of-the-art denoisers for regularizing generic inverse problems. In particular, we used these methods on two applications, namely, joint denoising and decompression and multi-image super-resolution.

In the second part, we explored the use of Variational AutoEncoder models to learn image priors and then regularize inverse problems. In Chapter 4 we constructed a quasi-biconvex alternating minimization algorithm, JPMAP, to compute MAP estimates with proven convergence guarantees. We showed its performance in several linear inverse problems such as denoising, deblurring, interpolation, compressed sensing and super-resolution. Then, in Chapter 5 we construct several sampling algorithms to explore the posterior distribution of inverse problems. Although this is an ongoing work at the time of the submission of this manuscript, the results obtained by thes methods are already promising, and further investigation is left for near future work. As recent works claim that VAE models can obtain state-of-the-art results on image generation [225, 239, 45], we expect that our algorithms will benefit from these more performant generative models.

The final performance of the techniques proposed in this theses heavly depend on the quality of the learned models used as image priors. In particular, the JPMAP (Chapter 4) and the posterior sampling algorithms (Chapter 5) assume that the images of interest follow a distribution $p_{\mathcal{X}}(\boldsymbol{x}) = \int p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z})\, p_{\mathcal{Z}}(\boldsymbol{z})\, d\boldsymbol{z}$ for a decoder $p_{\mathcal{X}|\mathcal{Z}}(\boldsymbol{x} \mid \boldsymbol{z})$ and latent code $p_{\mathcal{Z}}(\boldsymbol{z}) = \mathcal{N}(0, I)$. But the construction of such a generative model is not an easy task. In addition, most of the existing models were not trained for regularizing inverse problems, and some essencial properties we need may not be present in these models (e.g. pixel-wise fidelity as opposed to good-looking images).

If we can compress an image to around 10% of its size using patches without any noticeable artifact (like JPEG does), can we do the same with AutoEncoders? A possible solution is to train a *fine-level* autoencoder model on image patches, and hence to learn a latent representation which we call *Patch Embedding*. If we turn these patch embeddings into generative models (patch priors) then we may easily solve (not so much) ill-posed inverse problems like interpolation (e.g. with around 80% of missing pixels), deblurring, denoising, superresolution (e.g. with scale factor 2), etc.

Also, this dimensionality reduction model (done once on a large dataset like ImageNet) can be integrated as a first layer to simplify other generative models trained on specific datasets (such as CelebA) that captures global structures. This "2nd level" generative model is trained on the latent code (embedding) associated to each patch of the original image. Thus, we may first solve the 2nd level inverse problem (estimating the code for each patch of the original image) and then fine tune the pixel values solving the inverse problem on each patch domain. A related approach was recently proposed in [53].

Borrowing some ideas from [51, 28, 140] we could split the learning of the generative model into two parts:

1. AutoEncoder training for dimensionality reduction and feature extraction,
2. Density Estimation of the learned latent code.

For the first step, the most common choice for measuring the reconstruction error is the squared Euclidean L^2 distance but it is well known that this causes blurry reconstructions. Alternatively, in [23] a L^1 reconstruction loss computed on a Laplacian Pyramid representation is combined with the L^2 distance on pixel space, where each level of the pyramid is weighted in order to enforce better high-frequency fidelity.

Instead of learning an end-to-end latent representation (as in vanilla AutoEncoders), we can also use well known operators in order to lower the number of the parameters (to *only learn what needs to be learned*), enforce the balanced learning of spectral components with dedicated latent dimensions, and then simplify the training of the autoencoder [144]. For example, it is well known that images are sparser in the Discrete Fourier transform (DFT), Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) domains. So we consider using such transforms as first (last) layers of encoder (decoder) networks, as in [115, 161]. Also, by precisely controlling which frequencies of images are learned we can get rid of the spectral bias [175, 213, 33]. The construction of better generative models to be used in image reconstruction problems is a challenging problem that will also be left to future work.

Bibliography

[1] Cecilia Aguerrebere. "On the Generation of High Dynamic Range Images Theory and Practice from a Statistical Perspective". PhD. Telecom ParisTech, 2014. URL: https://tel.archives-ouvertes.fr/tel-01136641.

[2] Cecilia Aguerrebere et al. "A Bayesian Hyperprior Approach for Joint Image Denoising and Interpolation, With an Application to HDR Imaging". In: *IEEE Transactions on Computational Imaging* 3.4 (2017), pp. 633–646. ISSN: 2333-9403. DOI: 10.1109/TCI.2017.2704439. arXiv: 1706.03261. URL: https://nounsse.github.io/HBE_project/.

[3] Jabran Akhtar. "Optimization of biorthogonal wavelet filters for signal and image compression". MA thesis. Hovedoppgave, University of Oslo, 2001.

[4] F. Alter, S. Durand, and J. Froment. "Adapted total variation for artifact free decompression of JPEG images". In: *JMIV* 23 (2005), pp. 199–211.

[5] Iris Altpeter et al. "Robust solutions of inverse problems in electromagnetic non-destructive evaluation". In: *Inverse problems* 18.6 (2002), p. 1907.

[6] Christophe Andrieu et al. "An introduction to MCMC for machine learning". In: *Machine learning* 50.1 (2003), pp. 5–43.

[7] Francis J Anscombe. "The transformation of Poisson, binomial and negative-binomial data". In: *Biometrika* 35.3/4 (1948), pp. 246–254.

[8] Vegard Antun et al. "On instabilities of deep learning in image reconstruction-Does AI come at a cost?" In: *arXiv preprint arXiv:1902.05300* (2019).

[9] Martin Arjovsky and Léon Bottou. "Towards principled methods for training generative adversarial networks". In: *arXiv preprint arXiv:1701.04862* (2017).

[10] Simon R Arridge. "Optical tomography in medical imaging". In: *Inverse problems* 15.2 (1999), R41.

[11] Muhammad Asim et al. "Invertible generative models for inverse problems: mitigating representation error and dataset bias". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 399–409.

[12] Pierre Baldi and Kurt Hornik. "Neural networks and principal component analysis: Learning from examples without local minima". In: *Neural networks* 2.1 (1989), pp. 53–58.

[13] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[14] Martin Benning et al. "Preconditioned ADMM with nonlinear operator constraint". In: *IFIP Advances in Information and Communication Technology* 494 (2016), pp. 117–126. ISSN: 18684238. DOI: 10.1007/978-3-319-55795-3_10. arXiv: 1511.00425.

[15] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. "Learning Texture Manifolds with the Periodic Spatial GAN". In: *(ICML) International Conference on Machine Learning* 1 (2017), pp. 722–730. arXiv: 1705.06566.

[16] Michael Bernico. *Deep learning quick reference: useful hacks for training and optimizing deep neural networks with TensorFlow and Keras*. Packt Publishing Ltd, 2018.

[17] M Bertero and M Piana. "Inverse problems in biomedical imaging: modeling and methods of solution". In: *Complex systems in biomedicine*. Springer, 2006, pp. 1–33.

[18] Mario Bertero and Patrizia Boccacci. *Introduction to inverse problems in imaging*. CRC press, 2020.

[19] Alexandros Beskos, Gareth Roberts, and Andrew Stuart. "Optimal scalings for local Metropolis–Hastings chains on nonproduct targets in high dimensions". In: *The Annals of Applied Probability* 19.3 (2009), pp. 863–898.

[20] Siavash Arjomand Bigdeli and Matthias Zwicker. *Image Restoration using Autoencoding Priors*. Tech. rep. 2017. arXiv: 1703.09964.

[21] Siavash Arjomand Bigdeli et al. "Deep Mean-Shift Priors for Image Restoration". In: *(NIPS) Advances in Neural Information Processing Systems 30*. 2017, pp. 763–772. arXiv: 1709.03749. URL: http://papers.nips.cc/paper/6678-deep-mean-shift-priors-for-image-restoration.

[22] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN: 978-0387-31073-2. URL: http://research.microsoft.com/en-us/um/people/cmbishop/prml/.

[23] Piotr Bojanowski et al. "Optimizing the latent space of generative networks". In: *arXiv preprint arXiv:1707.05776* (2017).

[24] Ashish Bora et al. "Compressed sensing using generative models". In: *(ICML) International Conference on Machine Learning*. Vol. 2. JMLR. org. 2017, pp. 537–546. ISBN: 9781510855144. arXiv: arXiv:1703.03208v1.

[25] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[27] Stephen Boyd et al. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: 3.1 (2011), pp. 1–122. DOI: 10.1561/2200000016.

[28] Johann Brehmer and Kyle Cranmer. "Flows for simultaneous manifold learning and density estimation". In: (2020). arXiv: 2003.13913. URL: http://arxiv.org/abs/2003.13913.

[29] Antoni Buades, Bartomeu Coll, and J-M Morel. "A non-local algorithm for image denoising". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. IEEE. 2005, pp. 60–65.

[30] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. "A review of image denoising algorithms, with a new one". In: *Multiscale modeling & simulation* 4.2 (2005), pp. 490–530.

[31] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance weighted autoencoders". In: *arXiv preprint arXiv:1509.00519* (2015).

[32] Gregery T Buzzard et al. "Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium". In: *SIAM Journal on Imaging Sciences* 11.3 (2018), pp. 2001–2020.

[33] Yuan Cao et al. "Towards understanding the spectral bias of deep learning". In: *arXiv preprint arXiv:1912.01198* (2019).

[34] Gunnar Carlsson et al. "On the local behavior of spaces of natural images". In: *International journal of computer vision* 76.1 (2008), pp. 1–12.

[35] Khosrow Chadan and Pierre C Sabatier. *Inverse problems in quantum scattering theory*. Springer Science & Business Media, 2012.

[36] Khosrow Chadan et al. *An introduction to inverse scattering and inverse spectral problems*. SIAM, 1997.

[37] A Chambolle. "An algorithm for total variation minimization and applications". In: *Journal of Mathematical Imaging and Vision* 20 (2004), pp. 89–97. DOI: 10.1023/B:JMIV.0000011325.36760.1e.

[38] Antonin Chambolle and Pierre-Louis Lions. "Image recovery via total variation minimization and related problems". In: *Numerische Mathematik* 76.2 (1997), pp. 167–188.

[39] Antonin Chambolle and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of mathematical imaging and vision* 40.1 (2011), pp. 120–145.

[40] S. H. Chan, X. Wang, and O. A. Elgendy. "Plug-and-Play ADMM for Image Restoration: Fixed-Point Convergence and Applications". In: *IEEE Transactions on Computational Imaging* 3.1 (2017), pp. 84–98. ISSN: 2333-9403. DOI: 10.1109/TCI.2016.2629286. arXiv: 1605.01710.

[41] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. "Atomic decomposition by basis pursuit". In: *SIAM review* 43.1 (2001), pp. 129–159.

[42] Yunjin Chen and Thomas Pock. "Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1256–1272. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2596743. arXiv: 1508.02848.

[43] Dennis Child. *The essentials of factor analysis*. Cassell Educational, 1990.

[44] Rewon Child. "Very deep vaes generalize autoregressive models and can outperform them on images". In: *arXiv preprint arXiv:2011.10650* (2020).

[45] Rewon Child. "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images". In: *(ICLR) International Conference on Learning Representations*. 2020, pp. 1–17. arXiv: 2011.10650. URL: https://openreview.net/forum?id=RLRXCV6DbEJ.

[46] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *(ICLR) International Conference on Learning Representations*. 2016. arXiv: 1511.07289.

[47] Albert Cohen, Ingrid Daubechies, and J-C Feauveau. "Biorthogonal bases of compactly supported wavelets". In: *Communications on pure and applied mathematics* 45.5 (1992), pp. 485–560.

[48] Regev Cohen, Michael Elad, and Peyman Milanfar. "Regularization by Denoising via Fixed-Point Projection (RED-PRO)". In: (2020). ISSN: 23318422. arXiv: 2008.00226. URL: http://arxiv.org/abs/2008.00226.

[49] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[50] Kostadin Dabov et al. “Image denoising by sparse 3-D transform-domain collaborative filtering”. In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.

[51] Bin Dai and David Wipf. “Diagnosing and enhancing VAE models”. In: *7th International Conference on Learning Representations, ICLR 2019*. International Conference on Learning Representations, ICLR, 2019. arXiv: 1903.05789.

[52] Gianni Dal Maso. *An Introduction to Γ-Convergence*. Boston, MA: Birkhäuser Boston, 1993. ISBN: 978-1-4612-6709-6. DOI: 10.1007/978-1-4612-0327-8. URL: http://link.springer.com/10.1007/978-1-4612-0327-8.

[53] Giannis Daras et al. “Intermediate layer optimization for inverse problems using deep generative models”. In: *arXiv preprint arXiv:2102.07364* (2021).

[54] Ingrid Daubechies. “Orthonormal bases of compactly supported wavelets”. In: *Communications on pure and applied mathematics* 41.7 (1988), pp. 909–996.

[55] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[56] Li Deng. “The mnist database of handwritten digit images for machine learning research [best of the web]”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[57] Emily Denton and Vighnesh Birodkar. “Unsupervised learning of disentangled representations from video”. In: *arXiv preprint arXiv:1705.10915* (2017).

[58] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.

[59] Prafulla Dhariwal and Alex Nichol. “Diffusion models beat gans on image synthesis”. In: *arXiv preprint arXiv:2105.05233* (2021).

[60] Steven Diamond et al. “Unrolled optimization with deep priors”. In: (2017). arXiv: 1705.08041.

[61] Adrian Doicu, Thomas Trautmann, and Franz Schreier. *Numerical regularization for atmospheric inverse problems*. Springer Science & Business Media, 2010.

[62] Jeff Donahue and Karen Simonyan. “Large Scale Adversarial Representation Learning”. In: (2019). arXiv: 1907.02544. URL: http://arxiv.org/abs/1907.02544.

[63] Chao Dong et al. “Learning a deep convolutional network for image super-resolution”. In: *European conference on computer vision*. Springer. 2014, pp. 184–199.

[64] Randal Douc and Olivier Cappé. “Comparison of resampling schemes for particle filtering”. In: *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.* IEEE. 2005, pp. 64–69.

[65] Arnaud Doucet, Nando De Freitas, and Neil Gordon. “An introduction to sequential Monte Carlo methods”. In: *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 3–14.

[66] Arnaud Doucet and Xiaodong Wang. “Monte Carlo methods for signal processing: a review in the statistical signal processing context”. In: *IEEE Signal Processing Magazine* 22.6 (2005), pp. 152–170.

[67] S. Durand and J. Froment. “Reconstruction of wavelet coefficients using Total Variation minimization”. In: *SIAM, Journal on Scientific Computing* 24.5 (2003), pp. 1754–1767.

[68] Sylvain Durand and Mila Nikolova. “Denoising of Frame Coefficients Using ℓ^1 Data-Fidelity Term and Edge-Preserving Regularization”. In: *SIAM Multiscale Modeling & Simulation* 6.2 (2007), pp. 547–576. ISSN: 15403459. DOI: 10.1137/06065828X. URL: http://hal.archives-ouvertes.fr/hal-00204984/.

[69] Michael Elad and Michal Aharon. “Image denoising via sparse and redundant representations over learned dictionaries”. In: *IEEE Transactions on Image processing* 15.12 (2006), pp. 3736–3745.

[70] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. “See all by looking at a few: Sparse modeling for finding representative objects”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 1600–1607.

[71] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. “Testing the manifold hypothesis”. In: *Journal of the American Mathematical Society* 29.4 (2016), pp. 983–1049.

[72] Samuel G Finlayson et al. “Adversarial attacks against medical deep learning systems”. In: *arXiv preprint arXiv:1804.05296* (2018).

[73] Bernd Fischer and Jan Modersitzki. “Ill-posed medicine—an introduction to image registration”. In: *Inverse Problems* 24.3 (2008), p. 034008.

[74] Gerhard Freiling and Vjacheslav Anatoljevich Yurko. *Inverse Sturm-Liouville problems and their applications*. NOVA Science Publishers Huntington, 2001.

[75] GW Friedland and BD Thurber. "The birth of CT." In: *AJR. American journal of roentgenology* 167.6 (1996), pp. 1365–1370.

[76] Hongyun Gao et al. "Dynamic scene deblurring with parameter selective sharing and nested skip connections". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3848–3856.

[77] Alan E Gelfand and Adrian FM Smith. "Sampling-based approaches to calculating marginal densities". In: *Journal of the American statistical association* 85.410 (1990), pp. 398–409.

[78] Andrew Gelman. "Iterative and Non-iterative Simulation Algorithms". In: *Computing science and statistics* (1993), pp. 433–433.

[79] Andrew Gelman and Donald B Rubin. "A single series from the Gibbs sampler provides a false sense of security". In: *Bayesian statistics* 4 (1992), pp. 625–631.

[80] Andrew Gelman and Donald B Rubin. "Inference from iterative simulation using multiple sequences". In: *Statistical science* 7.4 (1992), pp. 457–472.

[81] Michaël Gharbi et al. "Deep joint demosaicking and denoising". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 191.

[82] Benyamin Ghojogh et al. "Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey". In: *arXiv preprint arXiv:2101.00734* (2021).

[83] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[84] Davis Gilton, Greg Ongie, and Rebecca Willett. "Neumann networks for inverse problems in imaging". In: (2019). arXiv: 1901.03707.

[85] Mario González, Andrés Almansa, and Pauline Tan. "Solving Inverse Problems by Joint Posterior Maximization with Autoencoding Prior". In: *arXiv preprint arXiv:2103.01648* (2021).

[86] Mario González et al. "Joint denoising and decompression using CNN regularization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 2598–2601.

[87] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[88] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[89] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. "Biconvex sets and optimization with biconvex functions: a survey and extensions". In: *Mathematical Methods of Operations Research* 66.3 (2007), pp. 373–407. ISSN: 1432-2994. DOI: 10.1007/s00186-007-0161-1.

[90] Raffaele Gravina et al. "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges". In: *Information Fusion* 35 (2017), pp. 68–80.

[91] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress. 2010, pp. 399–406.

[92] Rémi Gribonval. "Should penalized least squares regression be interpreted as maximum a posteriori estimation?" In: *IEEE Transactions on Signal Processing* 59.5 (2011), pp. 2405–2410.

[93] Rémi Gribonval and Pierre Machart. "Reconciling" priors" and" priors" without prejudice?" In: *Advances in Neural Information Processing Systems*. 2013.

[94] Harshit Gupta et al. "CNN-based projected gradient descent for consistent CT image reconstruction". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1440–1453. DOI: 10.1109/TMI.2018.2832656. arXiv: 1709.01809.

[95] Alfred Haar. "Zur theorie der orthogonalen funktionensysteme". In: *Mathematische Annalen* 69.3 (1910), pp. 331–371.

[96] Jacques Hadamard. *Lectures on Cauchy's Problem in Linear Partial Differential Equations*. Yale University Press, 1923.

[97] Paul Hand and Vladislav Voroninski. "Global Guarantees for Enforcing Deep Generative Priors by Empirical Risk". In: *IEEE Transactions on Information Theory* 66.1 (2020), pp. 401–418. ISSN: 15579654. DOI: 10.1109/TIT.2019.2935447. arXiv: 1705.07576.

[98] Per Christian Hansen. "The truncated SVD as a method for regularization". In: *BIT Numerical Mathematics* 27.4 (1987), pp. 534–553.

[99] W Keith Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: (1970).

[100] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[101] Leonhard Helminger et al. *Blind Image Restoration with Flow Based Priors*. Tech. rep. 2020. arXiv: 2009.04583. URL: http://arxiv.org/abs/2009.04583.

[102] Matthew Holden, Marcelo Pereyra, and Konstantinos C Zygalakis. "Bayesian Imaging With Data-Driven Priors Encoded by Neural Networks: Theory, Methods, and Algorithms". In: *arXiv preprint arXiv:2103.10182* (2021).

[103] Tom Hope, Yehezkel S Resheff, and Itay Lieder. *Learning tensorflow: A guide to building deep learning systems.* " O'Reilly Media, Inc.", 2017.

[104] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks". In: *Neural networks* 3.5 (1990), pp. 551–560.

[105] Wen Huang et al. "A Provably Convergent Scheme for Compressive Sensing under Random Generative Priors". In: (2018). arXiv: 1812.04176. URL: http://arxiv.org/abs/1812.04176.

[106] David H Hubel and Torsten N Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of physiology* 160.1 (1962), pp. 106–154.

[107] David H Hubel and Torsten N Wiesel. "Receptive fields of single neurones in the cat's striate cortex". In: *The Journal of physiology* 148.3 (1959), pp. 574–591.

[108] David A Huffman. "A method for the construction of minimum-redundancy codes". In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101.

[109] Daniel Jiwoong Im et al. "Denoising criterion for variational auto-encoding framework". In: *31st AAAI Conference on Artificial Intelligence, AAAI 2017*. AAAI press, 2017, pp. 2059–2065. arXiv: 1511.06406.

[110] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[111] Kyong Hwan Jin et al. "Deep convolutional neural network for inverse problems in imaging". In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4509–4522.

[112] Sergei Igorevich Kabanikhin. "Definitions and examples of inverse and ill-posed problems". In: (2008).

[113] Mark Kac. "Can one hear the shape of a drum?" In: *The american mathematical monthly* 73.4P2 (1966), pp. 1–23.

[114] Ulugbek S Kamilov, Hassan Mansour, and Brendt Wohlberg. "A plug-and-play priors approach for solving nonlinear imaging inverse problems". In: *IEEE Signal Processing Letters* 24.12 (2017), pp. 1872–1876.

[115] Eunhee Kang, Jong Chul Ye, et al. "Wavelet domain residual network (WavResNet) for low-dose X-ray CT reconstruction". In: *arXiv preprint arXiv:1703.01383* (2017).

[116] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410.

[117] Tero Karras et al. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *(ICLR) International Conference on Learning Representations* 10.2 (2017), pp. 327–331. arXiv: 1710.10196. URL: https://openreview.net/forum?id=Hk99zCeAb.

[118] Steven M Kay. *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, Inc., 1993.

[119] VI Keilis-Borok and TB Yanovskaja. "Inverse problems of seismology (structural review)". In: *Geophysical Journal International* 13.1-3 (1967), pp. 223–234.

[120] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[121] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *(ICLR) International Conference on Learning Representations*. Ml. 2013, pp. 1–14. ISBN: 1312.6114v10. DOI: 10.1051/0004-6361/201527329. arXiv: 1312.6114.

[122] Durk P Kingma et al. "Improved variational inference with inverse autoregressive flow". In: *Advances in neural information processing systems* 29 (2016), pp. 4743–4751.

[123] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. "Normalizing flows: An introduction and review of current methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[124] Barry W Kort and Dimitri P Bertsekas. "A new penalty function method for constrained minimization". In: *Proceedings of the 1972 IEEE Conference on Decision and Control and 11th Symposium on Adaptive Processes*. IEEE. 1972, pp. 162–166.

[125] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *(NIPS) Advances in neural information processing systems* (2012), pp. 1097–1105. ISSN: 10495258. arXiv: 1102.0183.

[126] Louis Landweber. "An iteration formula for Fredholm integral equations of the first kind". In: *American journal of mathematics* 73.3 (1951), pp. 615–624.

[127] Valero Laparra et al. "Perceptual image quality assessment using a normalized Laplacian pyramid". In: *Electronic Imaging* 2016.16 (2016), pp. 1–6.

[128] Fabian Latorre et al. "Fast and Provable ADMM for Learning with Generative Priors". In: *(NeurIPS) Advances in Neural Information Processing Systems*. Ed. by H Wallach et al. Vol. 32. NeurIPS. Curran Associates, Inc., 2019. arXiv: 1907.03343. URL: https://papers.nips.cc/paper/2019/hash/4559912e7a94a9c32b09d894f2bc3c82-Abstract.html.

[129] Rémi Laumont et al. "Bayesian imaging using Plug & Play priors: when Langevin meets Tweedie". In: *arXiv preprint arXiv:2103.04715* (2021).

[130] Rémi Laumont et al. "On Maximum-a-Posteriori estimation with Plug & Play priors and stochastic gradient descent". 2021. URL: https://hal.archives-ouvertes.fr/hal-03348735.

[131] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. "A nonlocal Bayesian image denoising algorithm". In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1665–1688.

[132] Yann LeCun et al. "Generalization and network design strategies". In: *Connectionism in perspective* 19 (1989), pp. 143–155.

[133] Yann Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. ISSN: 00189219. DOI: 10.1109/5.726791. arXiv: 1102.0183.

[134] Ann B Lee, Kim S Pedersen, and David Mumford. "The nonlinear statistics of high-contrast patches in natural images". In: *International Journal of Computer Vision* 54.1 (2003), pp. 83–103.

[135] Hsin-Ying Lee et al. "Diverse image-to-image translation via disentangled representations". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 35–51.

[136] José Lezama. "Overcoming the disentanglement vs reconstruction trade-off via jacobian supervision". In: *International Conference on Learning Representations*. 2018.

[137] Hao Li et al. "Visualizing the Loss Landscape of Neural Nets". In: *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc. 2018, pp. 6391–6401.

[138] Jun S Liu and Rong Chen. "Blind deconvolution via sequential imputations". In: *Journal of the american statistical association* 90.430 (1995), pp. 567–576.

[139] Yang Liu et al. "Disentangling Noise from Images: A Flow-Based Image Denoising Neural Network". In: (2021). arXiv: 2105.04746. URL: https://arxiv.org/abs/2105.04746v1.

[140] Yang Liu et al. "Disentangling Noise from Images: A Flow-Based Image Denoising Neural Network". In: *arXiv preprint arXiv:2105.04746* (2021).

[141] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

[142] Cécile Louchet and Lionel Moisan. "Posterior expectation of the total variation model: Properties and experiments". In: *SIAM Journal on Imaging Sciences* 6.4 (2013), pp. 2640–2684. ISSN: 19364954. DOI: 10.1137/120902276.

[143] James Lucas et al. "Don't blame the ELBO! A linear VAE perspective on posterior collapse". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019. arXiv: 1911.02469. URL: https://arxiv.org/abs/1911.02469.

[144] Andreas K Maier et al. "Learning with known operators reduces maximum error bounds". In: *Nature machine intelligence* 1.8 (2019), pp. 373–380.

[145] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[146] Stephane Mallat. "Multiresolution approximations and wavelet orthonormal bases of $L^2(R)$". In: *Transactions of the American mathematical society* 315.1 (1989), pp. 69–87.

[147] Pierre-Alexandre Mattei and Jes Frellsen. "Leveraging the exact likelihood of deep latent variable models". In: *arXiv preprint arXiv:1802.04826* (2018).

[148] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[149] Tim Meinhardt et al. "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems". In: *(ICCV) International Conference on Computer Vision*. 2017, pp. 1781–1790. DOI: 10.1109/ICCV.2017.198. URL: http://openaccess.thecvf.com/content_iccv_2017/html/Meinhardt_Learning_Proximal_Operators_ICCV_2017_paper.html.

[150] Tim Meinhardt et al. “Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems”. In: (2017). arXiv: 1704.03488. URL: http://arxiv.org/abs/1704.03488.

[151] Sachit Menon et al. “PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2020), pp. 2434–2442. ISSN: 10636919. DOI: 10.1109/CVPR42600.2020.00251. arXiv: 2003.03808.

[152] Nicholas Metropolis and Stanislaw Ulam. “The monte carlo method”. In: *Journal of the American statistical association* 44.247 (1949), pp. 335–341.

[153] Nicholas Metropolis et al. “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[154] Yves Meyer. *Wavelets and operators*. Vol. 1. Cambridge university press, 1995.

[155] Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. “When and why are deep networks better than shallow ones?” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[156] Vladimir Alekseevich Morozov. *Methods for solving incorrectly posed problems*. Springer Science & Business Media, 2012.

[157] Radford M. Neal. “Annealed importance sampling”. In: *Statistics and Computing* 11.2 (2001), pp. 125–139. ISSN: 09603174. DOI: 10.1023/A:1008923215028. arXiv: 9803008 [physics]. URL: http://arxiv.org/abs/physics/9803008.

[158] Radford M Neal. “Sampling from multimodal distributions using tempered transitions”. In: *Statistics and computing* 6.4 (1996), pp. 353–366.

[159] “Nonlinear total variation based noise removal algorithms”. In: *Physica D: Nonlinear Phenomena* 60.1-4 (1992), pp. 259–268. ISSN: 01672789. DOI: 10.1016/0167-2789(92)90242-F.

[160] Thomas Oberlin and Mathieu Verm. “Regularization via deep generative models: an analysis point of view”. In: (2021). arXiv: 2101.08661. URL: http://arxiv.org/abs/2101.08661.

[161] Edouard Oyallon et al. “Scattering networks for hybrid representation learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.9 (2018), pp. 2208–2221.

[162] Xiaochuan Pan, Emil Y Sidky, and Michael Vannier. “Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction?” In: *Inverse problems* 25.12 (2009), p. 123009.

[163] Yingxue Pang et al. “Image-to-Image Translation: Methods and Applications”. In: *arXiv preprint arXiv:2101.08629* (2021).

[164] Konstantinos Papafitsoros and Carola-Bibiane Schönlieb. “A combined first and second order variational approach for image reconstruction”. In: *Journal of mathematical imaging and vision* 48.2 (2014), pp. 308–338.

[165] George Papamakarios et al. “Normalizing flows for probabilistic modeling and inference”. In: *arXiv preprint arXiv:1912.02762* (2019).

[166] Adam Paszke et al. “Automatic differentiation in pytorch”. In: (2017).

[167] Jean-Christophe Pesquet et al. “Learning Maximally Monotone Operators for Image Recovery”. In: (2020). arXiv: 2012.13247. URL: http://arxiv.org/abs/2012.13247.

[168] Gabriel Peyré. “The numerical tours of signal processing”. In: *Computing in Science & Engineering* 13.4 (2011), pp. 94–97.

[169] Mark A Pinsky. *Introduction to Fourier analysis and wavelets*. Vol. 102. American Mathematical Soc., 2002.

[170] Javier Preciozzi. “Two Restoration Problems In Satellite Imaging”. [URL]. PhD thesis. Universidad de la República, 2016.

[171] Javier Preciozzi et al. “Joint denoising and decompression: A patch-based Bayesian approach”. In: *ICIP*. IEEE, 2017, pp. 1252–1256. ISBN: 978-1-5090-2175-8. DOI: 10.1109/ICIP.2017.8296482.

[172] E. Prestini. *The Evolution of Applied Harmonic Analysis: Models of the Real World*. Applied and Numerical Harmonic Analysis. Springer New York, 2016. ISBN: 9781489979872. URL: https://books.google.com.uy/books?id=_Lm6DAEACAAJ.

[173] Yunchen Pu et al. “Adversarial symmetric variational autoencoder”. In: *(NIPS) Advances in Neural Information Processing Systems*. 2017, pp. 4331–4340.

[174] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[175] Nasim Rahaman et al. “On the spectral bias of neural networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5301–5310.

[176] Ankit Raj, Yuqi Li, and Yoram Bresler. “GAN-Based Projector for Faster Recovery With Convergence Guarantees in Linear Inverse Problems”. In: *(ICCV) International Conference on Computer Vision*. IEEE, 2019, pp. 5601–5610. ISBN: 978-1-7281-4803-8. DOI: 10.1109/ICCV.2019.00570.

[177] Prajit Ramachandran, Barret Zoph, and Quoc V Le. “Searching for activation functions”. In: *arXiv preprint arXiv:1710.05941* (2017).

[178] Edward T Reehorst and Philip Schniter. "Regularization by denoising: Clarifications and new interpretations". In: *IEEE Transactions on Computational Imaging* 5.1 (2018), pp. 52–67. DOI: 10.1109/TCI.2018.2880326. arXiv: 1806.02296.

[179] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[180] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[181] Martin Riedmiller and Heinrich Braun. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm". In: *IEEE international conference on neural networks*. IEEE. 1993, pp. 586–591.

[182] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[183] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[184] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Undetermined. Springer Texts in Statistics. 2004. ISBN: 978-1-4419-1939-7. DOI: 10.1007/978-1-4757-4145-2.

[185] Gareth O Roberts and Jeffrey S Rosenthal. "General state space Markov chains and MCMC algorithms". In: *Probability surveys* 1 (2004), pp. 20–71.

[186] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

[187] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The little engine that could: Regularization by denoising (RED)". In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

[188] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[189] Kevin Roth et al. "Stabilizing training of generative adversarial networks through regularization". In: *arXiv preprint arXiv:1705.09367* (2017).

[190] DB Rubin. "A Noniterative Sampling/Importance resampling alternative to data augmentation for creating a few imputations when fractions of missing information are modest: The SIR algorithm". In: *Journal of the American Statistical Association* 82 (1987), pp. 544–546.

[191] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[192] Ernest K. Ryu et al. "Plug-and-Play Methods Provably Converge with Properly Trained Denoisers". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. 2019, pp. 5546–5557. arXiv: 1905.05406. URL: http://proceedings.mlr.press/v97/ryu19a.html.

[193] Youcef Saad and Martin H Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *SIAM Journal on scientific and statistical computing* 7.3 (1986), pp. 856–869.

[194] Mehdi S. M. Sajjadi et al. "Assessing Generative Models via Precision and Recall". In: *(NeurIPS) Neural Information Processing Systems*. 2018. arXiv: 1806.00035.

[195] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016), pp. 2234–2242.

[196] David Salomon. *Data compression: the complete reference*. Springer Science & Business Media, 2004.

[197] Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.

[198] Otmar Scherzer et al. "Variational methods in imaging". In: (2009).

[199] Eli Schwartz, Raja Giryes, and Alex M Bronstein. "DeepISP: Toward learning an end-to-end image processing pipeline". In: *IEEE Transactions on Image Processing* 28.2 (2018), pp. 912–923.

[200] Viraj Shah and Chinmay Hegde. "Solving linear inverse problems using gan priors: An algorithm with provable guarantees". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4609–4613.

[201] Sandip Sinharay. "Assessing convergence of the Markov chain Monte Carlo algorithms: A review". In: *ETS Research Report Series* 2003.1 (2003), pp. i–52.

[202] Vincent Sitzmann et al. "Implicit neural representations with periodic activation functions". In: *Advances in Neural Information Processing Systems* 33 (2020).

[203] Donald L Snyder, Abed M Hammoud, and Richard L White. "Image recovery from data acquired with a charge-coupled-device camera". In: *JOSA A* 10.5 (1993), pp. 1014–1023.

[204] Antti Solonen et al. “Efficient MCMC for climate model parameter estimation: Parallel adaptive chains and early rejection”. In: *Bayesian Analysis* 7.3 (2012), pp. 715–736.

[205] Casper Kaae Sønderby et al. “Ladder variational autoencoders”. In: *Advances in neural information processing systems* 29 (2016), pp. 3738–3746.

[206] The Consultative Committee for Space Data Systems (CCSDS). “Technical report, Image data compression”. In: 122.0-b-1 (2005). [URL].

[207] The Consultative Committee for Space Data Systems (CCSDS). “Technical report, Image data compression”. In: 122.0-b-1 (2005). https://public.ccsds.org/Publications/.

[208] Suhas Sreehari et al. “Plug-and-Play Priors for Bright Field Electron Tomography and Sparse Interpolation”. In: *IEEE Transactions on Computational Imaging* 2.4 (2016), pp. 1–1. ISSN: 2333-9403. DOI: 10.1109/TCI.2016.2599778. arXiv: 1512.07331.

[209] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[210] Jean-Luc Starck, Fionn D Murtagh, and Albert Bijaoui. *Image processing and data analysis: the multiscale approach*. Cambridge University Press, 1998.

[211] William W Symes. “The seismic reflection inverse problem”. In: *Inverse problems* 25.12 (2009), p. 123008.

[212] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[213] Matthew Tancik et al. “Fourier features let networks learn high frequency functions in low dimensional domains”. In: *arXiv preprint arXiv:2006.10739* (2020).

[214] Martin A Tanner and Wing Hung Wong. “The calculation of posterior distributions by data augmentation”. In: *Journal of the American statistical Association* 82.398 (1987), pp. 528–540.

[215] Matthieu Terris et al. “Building firmly nonexpansive convolutional neural networks”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2020-May (2020), pp. 8658–8662. ISSN: 15206149. DOI: 10.1109/ICASSP40776.2020.9054731.

[216] Lucas Theis, Aäron van den Oord, and Matthias Bethge. “A note on the evaluation of generative models”. In: *arXiv preprint arXiv:1511.01844* (2015).

[217] Luke Tierney and Antonietta Mira. "Some adaptive Monte Carlo methods for Bayesian inference". In: *Statistics in medicine* 18.17-18 (1999), pp. 2507–2515.

[218] Andrei Nikolaevich Tikhonov. "Regularization of incorrectly posed problems". In: Soviet Mathematics Doklady. 1963.

[219] Andrei Nikolajevits Tikhonov. "Solution of incorrectly formulated problems and the regularization method". In: *Soviet Math.* 4 (1963), pp. 1035–1038.

[220] Michael E Tipping and Christopher M Bishop. "Probabilistic principal component analysis". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622.

[221] Lisa Torrey and Jude Shavlik. "Transfer learning". In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[222] Paul Tseng and Dimitri P. Bertsekas. "On the convergence of the exponential multiplier method for convex programming". In: *Mathematical Programming* 60.1-3 (1993), pp. 1–19. ISSN: 00255610. DOI: 10.1007/BF01580598. URL: https://link.springer.com/article/10.1007/BF01580598.

[223] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.

[224] Bryan E Usevitch. "A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000". In: *IEEE signal processing magazine* 18.5 (2001), pp. 22–35.

[225] Arash Vahdat and Jan Kautz. "Nvae: A deep hierarchical variational autoencoder". In: *arXiv preprint arXiv:2007.03898* (2020).

[226] Tuomo Valkonen. "A primal-dual hybrid gradient method for nonlinear operators with applications to MRI". In: *Inverse Problems* 30.5 (2014), pp. 1–42. ISSN: 13616420. DOI: 10.1088/0266-5611/30/5/055012. arXiv: 1309.5032.

[227] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. "Plug-and-Play priors for model based reconstruction". In: *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings* (2013), pp. 945–948. DOI: 10.1109/GlobalSIP.2013.6737048.

[228] Roman Vershynin. "Random Vectors in High Dimensions". In: *Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press* 3 (2018), pp. 38–69.

[229] Rene Vidal et al. "Mathematics of deep learning". In: *arXiv preprint arXiv:1712.04741* (2017).

[230] John D Villasenor, Benjamin Belzer, and Judy Liao. "Wavelet filter evaluation for image compression". In: *IEEE Transactions on image processing* 4.8 (1995), pp. 1053–1060.

[231] Maxime Vono, Nicolas Dobigeon, and Pierre Chainais. "Asymptotically Exact Data Augmentation: Models, Properties, and Algorithms". In: *Journal of Computational and Graphical Statistics* (2020), pp. 1–14.

[232] Maxime Vono, Nicolas Dobigeon, and Pierre Chainais. "Split-and-augmented Gibbs sampler—Application to large-scale inference problems". In: *IEEE Transactions on Signal Processing* 67.6 (2019), pp. 1648–1661.

[233] Zhou Wang and Alan C Bovik. "Mean squared error: Love it or leave it? A new look at signal fidelity measures". In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.

[234] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. "Multiscale structural similarity for image quality assessment". In: *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003.* Vol. 2. Ieee. 2003, pp. 1398–1402.

[235] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[236] P. Weiss et al. "Compression artifacts reduction using variational methods : Algorithms and experimental study". In: *ICASSP*. 2008, pp. 1173–1176. DOI: 10.1109/ICASSP.2008.4517824.

[237] Paul Werbos. "Beyond regression: new tools for prediction and analysis in the behavioral sciences". In: *Ph. D. dissertation, Harvard University* (1974).

[238] Jay Whang, Qi Lei, and Alexandros G. Dimakis. "Compressed Sensing with Invertible Generative Models and Dependent Noise". In: *NeurIPS deep-inverse workshop*. 2020. arXiv: 2003.08089v2.

[239] Zhisheng Xiao et al. "Vaebm: A symbiosis between variational autoencoders and energy-based models". In: *International Conference on Learning Representations*. 2020.

[240] Xiaojian Xu et al. "Provable Convergence of Plug-and-Play Priors with MMSE denoisers". In: 4 (2020), pp. 1–10. arXiv: 2005.07685. URL: http://arxiv.org/abs/2005.07685.

[241] WQ Yang et al. "An image-reconstruction algorithm based on Landweber's iteration method for electrical-capacitance tomography". In: *Measurement Science and Technology* 10.11 (1999), p. 1065.

[242] Guoshen Yu, G. Sapiro, and S. Mallat. "Solving Inverse Problems With Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity". In: *Image Processing, IEEE Transactions on* 21.5 (2012), pp. 2481–2499. ISSN: 1057-7149. DOI: 10.1109/TIP.2011.2176743.

[243] Kai Zhang, Wangmeng Zuo, and Lei Zhang. "FFDNet: Toward a fast and flexible solution for CNN-based image denoising". In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4608–4622.

[244] Kai Zhang et al. "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising". In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[245] Kai Zhang et al. "Learning Deep CNN Denoiser Prior for Image Restoration". In: *(CVPR) IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017, pp. 2808–2817. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.300. arXiv: 1704.03264. URL: http://openaccess.thecvf.com/content_cvpr_2017/html/Zhang_Learning_Deep_CNN_CVPR_2017_paper.html.

[246] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[247] Zijun Zhang et al. "Perceptual Generative Autoencoders". In: *(ICLR) International Conference on Learning Representations*. 2019, pp. 1–7. arXiv: 1906.10335. URL: https://github.com/zj10/PGA.

[248] Y-T Zhou et al. "Image restoration using a neural network". In: *IEEE transactions on acoustics, speech, and signal processing* 36.7 (1988), pp. 1141–1151.

[249] Daniel Zoran and Yair Weiss. "From learning models of natural image patches to whole image restoration". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 479–486.

[250] Argyrios Zymnis, Stephen Boyd, and Emmanuel Candes. "Compressed sensing with quantized measurements". In: *IEEE Signal Processing Letters* 17.2 (2009), pp. 149–152.