Dominik Narnhofer

Inverse Problems with Generative Neural Networks

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Thomas Pock
Institute for Computer Graphics and Vision

Graz, Austria, Mar. 2019

Abstract

One of the most important and challenging mathematical problems in science and industry is recovering model parameters from indirect noisy observations, which is referred to as inverse problem. In this thesis, generative adverserial networks (GANs) are used to address common inverse problems in the imaging domain, such as denoising, inpainting, super-resolution, deblurring and MRI reconstruction. GANs provide a strong implicit prior that encourages the solution of an inverse problem to stay in the domain of source images. Thus, the complex and time-consuming task of finding an appropriate analytical regularizer is no longer required. However, this implicit prior is still not completely optimal, since the immanent representation space of the GAN does not cover the entire data domain, in contrast to the theoretic assumption. To overcome this issue, a method is proposed which is able to manipulate the approximated distribution of the GAN by introducing an additional optimization task, ultimately leading to an improved representation space. This additional adjustment of the implicit prior enables close to exact representations of unseen samples and thereby also improves reconstruction results of inverse problems.

Keywords: Generative Adverserial Networks, Inverse Problems, Denoising, Inpainting, Super-Resolution, Deblurring, MRI Reconstruction, Deep Learning, Unsupervised Learning, Image Processing

Kurzfassung

Eines der wichtigsten und herausforderndsten mathematischen Probleme in Wissenschaft und Industrie ist es Modellparameter von indirekten und verrauschten Beobachtungen zu rekonstruieren, was bekanntermaßen ein inverses Problem darstellt. In dieser Arbeit wurden häufig auftretende inverse Probleme im Bereich der Bildgebung, wie Denoising, Inpainting, Super-Resolution, Deblurring und MRT Rekonstruktion durch den Einsatz von Generative Adversarial Networks (GANs) gelöst. Diese besitzen einen starken impliziten Prior, der dafür sorgt, dass Lösungen im Definitionsbereich der Quellbilder begünstigt werden, wodurch die komplexe und zeitaufwändige Suche eines geeigneten analytischen Regularisierers vermieden werden kann. Jedoch ist dieser implizite Prior nicht gänzlich optimal, da der inhärente Repräsentationsraum des GANs, im Gegensatz zur theoretischen Annahme, nicht den gesamten Definitionsberich der Quellbilder abdeckt. Um dieses Problem zu überwinden wird eine Methode vorgestellt, die in der Lage ist die approximierte Verteilung des GANs zu manipulieren, indem ein zusätzlicher Optimierungsschritt eingeführt wird, was letztendlich zu einem verbesserten Repräsentationsraum des GANs führt. Diese zusätzliche Anpassung des impliziten Priors ermöglicht eine nahezu exakte Darstellung von ungesehenen Daten und verbessert dadurch auch die Rekonstruktionsergebnisse von inversen Problemen.

Acknowledgments

I would first like to thank my supervisor and mentor Professor Thomas Pock from the Institute of Computer Graphics and Vision. I am extremely grateful and indebted to him for providing his expertise and continuous support. His sincere encouragement and valuable guidance largely helped to bring this work to success.

My sincere gratitude also goes to Patrick Knöbelreiter who I had the pleasure to share an office with. I have greatly benefited from our insightful discussions and his helpful advice. Moreover, I am particularly grateful for the assistance given by Alexander Effland. His constructive feedback and enlightening suggestions were invaluable for the completion of this work.

Furthermore, my deepest gratitude goes to my parents and my sister for their unconditional love and encouragement. I am forever grateful for your support.

Finally, I would like to show my greatest heartfelt appreciation to Lydia for encouraging me in all of my pursuits and inspiring me to follow my passion. Thank you for always believing in me.

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Motivation

Digital images tend to play an increasingly important role in our every-day life. In many domains, such as autonomous driving, surveillance, industrial manufacturing, security, transport and medicine, critical decisions are based on digital images and their underlying content. Images where crucial information is obscured due to corruption may result in drawing incorrect and harmful conclusions. A self-driving car that does not recognize a pedestrian crossing the street due to a corrupted measurement, or a radiologist who misinterprets a pathology due to noise in data acquired by magnetic resonance imaging (MRI) are notable examples. Therefore, retrievability of information must be guaranteed, even in corrupted images. An example of noisy MRI data and its corresponding reconstruction is illustrated in figure 1.1. One way of reconstructing corrupted information is to mathematically formulate the cause and try to eliminate the effect in the framework of inverse problems. The cause of corruption can often be found in the imaging system, e.g. the camera or MRI-scanner, and can mathematically be expressed as an operator $\mathcal{F} : X \to Y$ that maps from a space of source images X to the space of measurements Y. Usually in the imaging domain, $\mathcal{F}$ can be expressed as a linear function afflicted with additive noise $\nu \in \mathcal{N}(0, \sigma)$:

$$y = \mathcal{F}(x) + \nu, \tag{1.1}$$

where $x \in X$ describes the desired source image, $y \in Y$ denotes the observed measurement and $\mathcal{F}$ is a linear operator that describes the underlying physical data acquisition process.

In most cases, $\mathcal{F}$ is assumed to be known, which wrongly implies that the desired source image x can be found by simply computing the inverse of $\mathcal{F}$. In general, however, this is not possible since imaging related problems are mostly of ill-posed nature and hence $\mathcal{F}$ cannot be inverted.

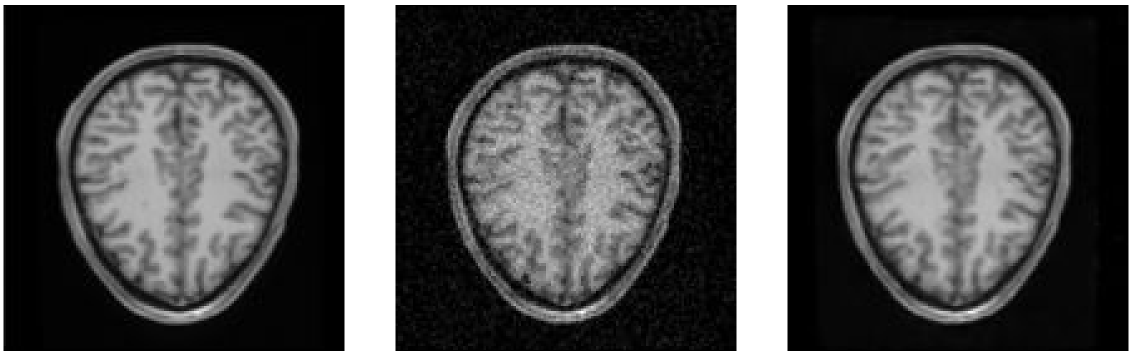
(a) Source **(b)** Corrupted **(c)** Reconstruction

Figure 1.1: Illustration of an MRI image corrupted with noise, along with the corresponding source and reconstructed image.

Standard approaches to solving ill-posed inverse problems (discussed in chapter 2) are iterative reconstruction methods that aim to minimize a distance function between the observation y and $\mathcal{F}(x)$, which is penalized by a regularizer $\mathcal{R} : X \to \mathbb{R}$ that can be used to impose smoothness assumptions on x:

$$x^* \in \arg\min_{x \in X} \frac{1}{2}\|y - \mathcal{F}(x)\|_2^2 + \mathcal{R}(x) \tag{1.2}$$

For the majority of inverse problems in the imaging domain distinct smoothness priors, such as those based on total variation [1], have been under heavy research in the past decade. More recent methods are based on the promising framework of deep learning, where most of those techniques directly approximate the inverse $\mathcal{F}^{-1}$ of the measurement function. Nevertheless, a drawback of deep learning-based methods is the requirement of a large collection of source-observation pairs as well as the need of a separate model for each different problem. In this thesis, several linear inverse problems including denoising, inpainting, super-resolution, deblurring and compressed sensing are solved by the use of GANs, which naturally provide an implicit prior that constrains the reconstruction to the domain of source images. Thus, they overcome the need of regularizing the reconstruction [2].

1.2 Contributions

Considering the standard formulation of inverse problems, it is required to preliminary define a regularizer that should ideally incorporate a priori information, which restricts the resulting solution to the domain of source images X. In practice, however, finding analytical regularizers that improve the solution quality is a difficult and time-consuming task. In this thesis, the issue of finding an appropriate analytical regularizer is overcome by using GANs, which constitute a type of neural network-based generative machine learning algorithm with a strong implicit prior on X. Unfortunately, the provided prior is not completely sufficient as it restricts the solution space $\tilde{X}$ to the representation space of the GAN, which in practice does not exactly cover the entire domain of X, as opposed to the theoretic assumption. In this thesis, a method is proposed that allows to overcome this issue by introducing an additional optimization task, which manipulates the approximated distribution of the GAN in order to obtain an improved representation space. This additional refinement of the domain covered by the GAN enables close to exact representations of unseen samples and hence also improves reconstruction results of inverse problems.

1.3 Outline

Chapter 2 provides an introduction to inverse problems and the associated mathematical formulation in general. Furthermore, all inverse problems that are addressed in this thesis are described. In chapter 3, a comprehensive summary of the basic ideas behind deep learning is provided, which focuses on the individual components of artificial neural networks as well as on the underlying mathematical framework. Moreover, this chapter discusses important deep learning concepts that are of particular interest in the field of image processing. The theoretical background of GANs, which is build upon two competing deep neural networks, is described in chapter 4. Here underlying principles and important conceptual extensions are outlined along with the idea of using GANs in an inverse manner. Chapter 5 then provides an overview of work related to the topic of this thesis. Beyond that, chapter 6 introduces the methods used to solve the addressed inverse problems, including a novel approach for improving the obtained results. Additionally, this chapter describes the different datasets used to evaluate the performance of the proposed techniques. Experimental results obtained in this work are presented in chapter 7. The outcome of each individual dataset is presented qualitatively in form of illustrated reconstructions as well as quantitatively by providing the peak signal to noise ratio (PSNR). Finally, chapter 8 concludes the findings of this thesis and gives an outlook on future work and potential fields of application.

2

Inverse Problems

Contents

2.1 Introduction to Inverse Problems

A key aspect of many scientific and industrial fields, such as geophysics, finance, astronomy and medicine, is to recover the parameters of an underlying model from indirect noisy observations, which is referred to as an inverse problem. While there is no universal formal defintion that describes inverse problems, a frequently quoted statement on this issue was provided by J.B. Keller [3]:

"We call two problems inverses of another if the formulation of each involves all or part of the solution of the other. Often, for historical reasons, one of the two problems has been studied extensively for some time, while the other has never been studied and is not so well understood. In such case, the former is called the direct problem, while the latter is the inverse problem".

A direct problem is associated with calculating the effects of given causes while the corresponding inverse problem deals with the inversion of this cause-effect relationship and therefore involves finding the unknown causes of known effects [4]. Typically, inverse problems are ill-posed in contrast to well-posed problems, which have to satisfy the following three conditions suggested by Jacques Hadamard:

- **Existence:** the solution of the problem exists for any data
- **Uniqueness:** the solution of the problem is unique
- **Stability:** the solution of the problem continuously depends on the data

If one of the conditions listed above is not satisifed, the problem is called ill-posed, and consequently has either no solution, two or more solutions, or the solution is unstable [5].

A vivid example is the following: Considering a question-answer problem, the question "What is the capital city of Austria?" is the direct problem to which the answer is "Vienna". The corresponding inverse problem would be: What is the question to which the answer is "Vienna". Since infinitely many questions could yield this answer, this problem is obviously ill-posed. Having prior knowledge about the context of the question, e.g. geography, limits the amount of possible solutions. Incorporating such domain knowledge to favor certain solutions is called regularization and will be explained in the following section.

In science and engineering solving inverse problems mostly refers to estimating model parameters from observed data. An example would be linear regression where the goal is to find the slope $k \in \mathbb{R}$ and bias $d \in \mathbb{R}$ that fits the underlying relation between a set of N observations $\{x_i, y_i \in \mathbb{R}\}_{y=1}^{N}$. If more than two observations (i.e. $N \geq 2$) are available, this problem satisfies all Hadamard conditions and is therefore well posed.

2.2 General Formulation

The aim in most cases of ill-posed inverse problems is to recover the true source signal $x \in X$ from a noisy observation $y \in Y$ that can be obtained from an operator $\mathcal{F} : X \to Y$, known as the measurement or direct process. Mathematically, this can be expressed as:

$$y = \mathcal{F}(x) + \nu, \tag{2.1}$$

where ν refers to the measurement noise.

Due to the ill-posed nature of the problem and measurement noise that causes serve perturbations in the solution, the source signal cannot be obtained by simply inverting $\mathcal{F}$. Nevertheless, a solution to the inverse problem can be acquired by finding an estimate

$x^* \in X$ that yields y when undergoing the process of $\mathcal{F}$. Under the assumption that the parameters of the measurement process are known, this can be expressed as the following optimization problem:

$$x^* \in \operatorname*{argmin}_{x \in X} \mathcal{D}(y, \mathcal{F}(x)) + \lambda \mathcal{R}(x), \tag{2.2}$$

where $\mathcal{D} : Y \times Y \to \mathbb{R}$ describes the distance between y and $\mathcal{F}(x)$, i.e. the misfit between the observations produced by x and the given observations y, also known as data fidelity term. Naively solving (2.2) without considering $\mathcal{R}(x)$ is likely to yield a result outside the domain of X, as the set of possible solutions can be infinitely large. Therefore, prior knowledge about the domain of the source signal is required to solve the inverse problem in a feasible manner. This prior knowledge is incorporated by constraining the solution space with the help of a regularizer $\mathcal{R} : X \to \mathbb{R}$ that aims to keep the solution in the domain of X. The regularization parameter λ weights the importance of the regularization relatively to the objective. Usually, the choice of the data fidelity term $\mathcal{D}$ is straightforward, e.g. to define it as an ℓ^p-norm, for example, the squared ℓ^2-Norm, whereas the choice of the regularizer is less intuitive and highly dependant on the problem at hand [6].

2.3 Linear Inverse Problems

When the direct process $\mathcal{F} : X \to Y$ can be expressed as a linear operator $A : X \to Y$, the observation $y \in Y$ following from a source signal $x \in X$ can be obtained as:

$$y = Ax + \nu, \tag{2.3}$$

where the matrix A models the underlying physics of the data acquisition process and ν denotes the measurement noise. In theory, the reconstruction can simply be obtained by the inversion of A. However, in most inverse problems, the matrix A is not a square matrix and therefore, the problem is either under determined (no solution exists) or over determined (many solutions exist), which clearly violates the first and second Hadamard condition. Still, this issue can be overcome by observing the least squares solution, also known as the pseudo inverse (shown for an overdetermined problem):

$$y = (A^* A)^{-1} Ax, \tag{2.4}$$

where A^* denotes the adjoint operator defined through the identity $\langle Ax, y \rangle \equiv \langle x, A^* y \rangle$. Even though this solution is in accordance with the first two Hadamard conditions, the pseudo inverse of A could still be unstable and therefore violate the third condition of Hadamard. Historically, multiple techniques, such as Thikonov regularization [7], have been developed to avoid this issue so that the solution depends smoothly on the data and fulfills all Hadamard conditions.

2.4 Denoising

The main challenge in denoising is to recover the corresponding uncorrupted version of an image that is corrupted by noise. The main reason for noise in digital photography is insufficient lighting during image acquisition, which leads to a low signal to noise ratio (SNR) in the data obtained from the camera sensor. In the medical imaging domain, noise is an inherent property of all acquisition systems, which generally tends to reduce image quality and lowers the diagnostic value. The noise occurring in medical images is usually Gaussian, Poisson and Rician distributed [8].

The general formulation of the direct process, stated in equation (2.3), already assumes additive noise in the measurement process, and therefore, the forward operator can simply be modeled as the identity transform. Accordingly, the observed image with height M, width N and channels C, $y \in \mathbb{R}^{M \times N \times C}$ is the source image $x \in \mathbb{R}^{M \times N \times C}$ corrupted with element-wise additive noise $\nu \in \mathbb{R}^{M \times N \times C}$, drawn from a parametrized distribution $\nu \in \mathcal{N}(0, \sigma)$:

$$y = x + \nu \tag{2.5}$$

An example of an image corrupted by Gaussian noise and the corresponding source image can be seen in figure 2.1.

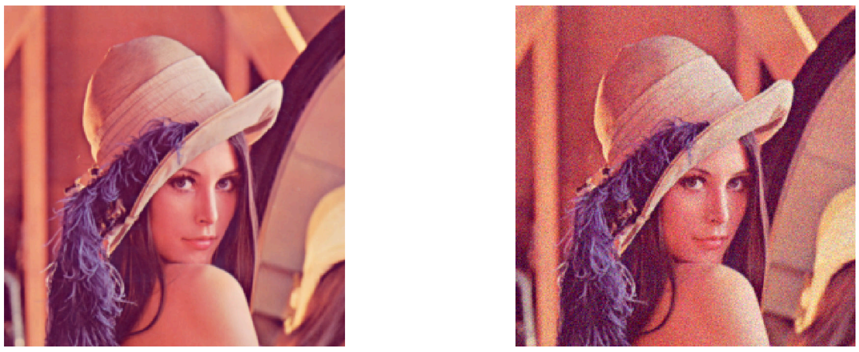
(a) (b)

Figure 2.1: (a) shows the source image. (b) shows image corrupted with Gaussian noise.

2.5 Inpainting

The aim of image inpainting is to restore damaged and missing areas of which the underlying information is unknown. Furthermore, the goal is to make the reconstructed areas look naturally and indistinguishable from the rest of the image. Corrupted areas can arise due to multiple reasons, such as text-overlays, object removal or damaged regions in digitalized analog photos. The ill-posedness in image inpainting arises due to the fact that the underlying image information of a corrupted area is unknown, which makes the solution space infinite. The forward operator for the inpainting problem is provided as follows:

$$y = M \odot x + \nu, \tag{2.6}$$

where $\odot$ denotes an element-wise multiplication and $M \in \mathbb{R}^{M \times N \times C}$ represents a mask $M_{i,j,c} \in \{0, 1\}$ of the same size as the original image x. The resulting image $y \in \mathbb{R}^{M \times N \times C}$ is simply the original image where known parts stay unchanged and unknown parts are set to 0, as demonstrated in figure 2.2.

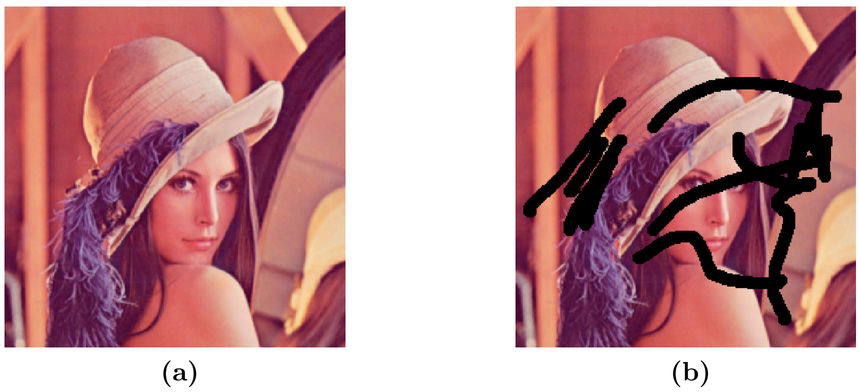

Figure 2.2: (a) shows the source image. (b) shows image with corrupted regions.

2.6 Deblurring

Deblurring describes the task of reconstructing a latent sharp image from a corresponding blurred version with knowledge about the blur kernel Φ. Blur in images is one of the most common reasons of degradation and is usually caused by motion of objects in the image (motion blur), motion of the optical system (camera shake blur), blur of regions not focused by camera (focus blur) and atmospheric particles in wide focus images (atmospheric blur)[9]. An example of an image blurred by a Gaussian filter and the corresponding source image can be seen in figure 2.3.

Formally, the blurred image $y \in \mathbb{R}^{M \times N \times C}$ can be obtained by convolving the source image $x \in \mathbb{R}^{M \times N \times C}$ with a blur kernel $h \in \mathbb{R}^{m \times n \times C}$ and adding the measurement noise $\nu \in \mathbb{R}^{M \times N \times C}$:

$$y = x * h + \nu \tag{2.7}$$

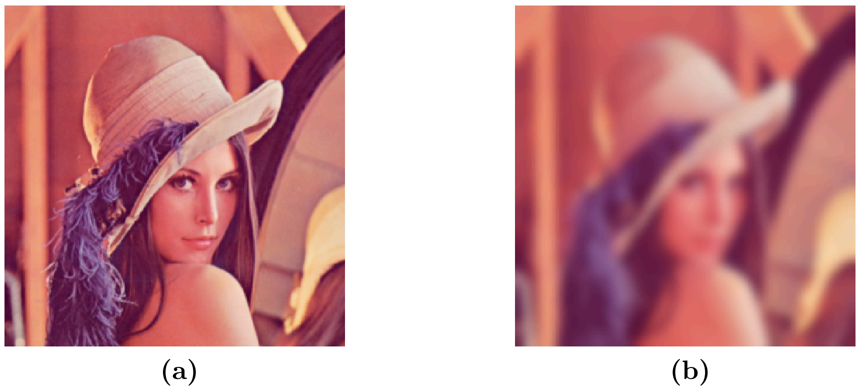

Figure 2.3: (a) shows the source image. (b) shows image convolved with a Gaussian blur kernel.

2.7 Super-Resolution

Image super-resolution approaches are used to obtain a high resolution version given a single or multiple low resolution images. In multi image super-resolution approaches, the non-redundant information of multiple low resolution images is combined to obtain a high resolution image. The additional information is provided by sub-pixel shifts in the image set, which helps to make the ill-conditioned upsampling problem better conditioned [10]. Single image super-resolution, which is performed in this thesis, is always an ill-posed problem since high resolution information can only be estimated in order to make the resulting image look naturally in higher resolution.

For the task of super-resolution, the forward model is observed by a down sample blur operator $DB : \mathbb{R}^{M \times N \times C} \to \mathbb{R}^{m \times n \times C}$, where the source image $x \in \mathbb{R}^{M \times N \times C}$ needs to be low-pass filtered before down-sampling in order to avoid aliasing in the low resolution observation $y \in \mathbb{R}^{m \times n \times C}$.

$$y = DB(x) + \nu \tag{2.8}$$

An example for an up-scaled low resolution image and its corresponding high resolution version can be seen in figure 2.4.

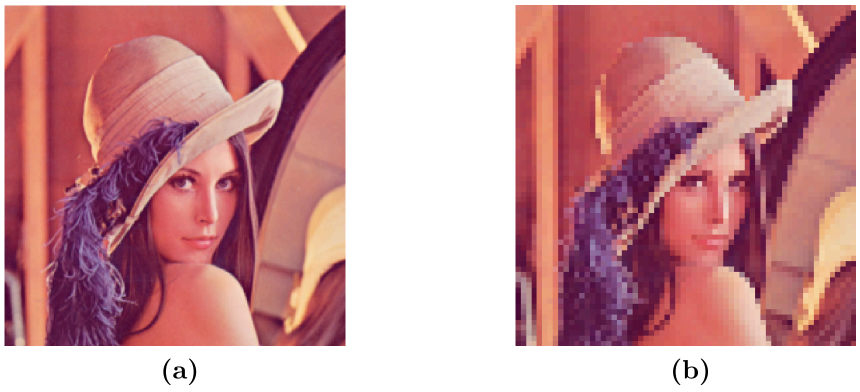

Figure 2.4: (a) shows the source image. (b) shows upscaled low resolution image.

2.8 MRI reconstruction

The mathematical framework of compressed sensing (CS) is used to reconstruct data from under-sampled measurements. An important field of application for CS is magnetic resonance imaging (MRI), which utilizes the ability of protons to absorb and emit radio frequency when exposed to an external magnetic field [11]. The emitted signal is measured using radio frequency coils, placed near the investigated tissue [12]. This spatial frequency information is stored in the so-called k-space, which corresponds to the two-dimensional Fourier transform of the desired image.

A major drawback of MR imaging is the slow image acquisition rate, which results in high expenses and long treatment duration for the patient. One approach to save time in MRI is to under-sample the k-space, which, however, leads to aliasing artifacts as illustrated in figure 2.5. The aim of compressed sensing in MR imaging is the estimation of the missing information in k-space and thereby reconstruct the original image. The mathematical expression is given as follows:

$$\hat{y} = S \odot \mathscr{F}(x) + \nu \tag{2.9}$$

where $\mathscr{F}$ represent the two-dimensional Fourier transform. Furthermore, $x \in \mathbb{R}^{M \times N}$ denotes the desired image, $y \in \mathbb{C}^{M \times N}$ indicates the complex observation and $S \in \mathbb{C}^{M \times N}$ denotes a complex mask $S_{i,j} \in \{0, 1\}$ that blanks out the parts that are lost in Fourier domain due to under-sampling.

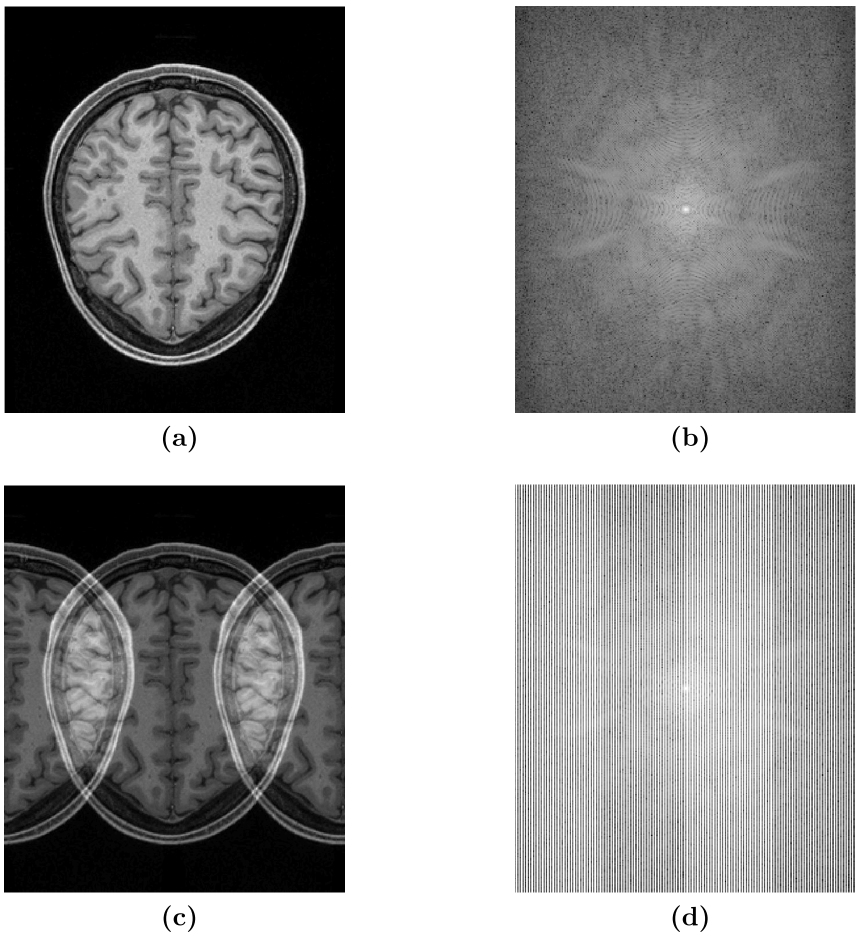

Figure 2.5: (a) shows the source image. (b) shows the k-space representation of the source image, i.e. the 2D Fourier transform. (c) shows the observed image after inverse Fourier transform of the under-sampled k-space. (d) shows the under-sampled k-space.

3 Neural Networks

Contents

This chapter provides a conceptual overview of artificial neural networks (ANNs), which represent a data-driven machine learning technique that gathers information by detecting distinct relationships and patterns in data [13].

3.1 Biological Interpretation

Artificial neural networks are computational models that are inspired by the structure of biological neural networks in the human brain and the way they process information [14]. The brain, in its function as flexible computational machine, is able to discriminate between an enormous amount of events in a continous stream of environmental information. Structurally simple elements, called neurons, are the basic computaional units, which organize this information into perceptions and trigger appropriate behavioral responses [15]. The full potential of these simple neurons comes into effect when they are interconnected into large neural networks. The human brain, for example, consists of a dense network containing approximately 10^{11} neurons that are interconnected via roughly 10^{14} synapses [16]. The artificial neuron or node, which constitutes the main computationl unit of ANNs, is mathematically represented by consecutive multiplication, summation and activation of information [17]. An illustration of both, a biological and an artificial neuron can be seen in figure 3.1. Loosely speaking, dendrites correspond to network inputs, the cell body fullfills the task of summation and activation, and axons represent the output of the ANN. Furthermore, weights used in ANNs fulfill the task of synapses [18].

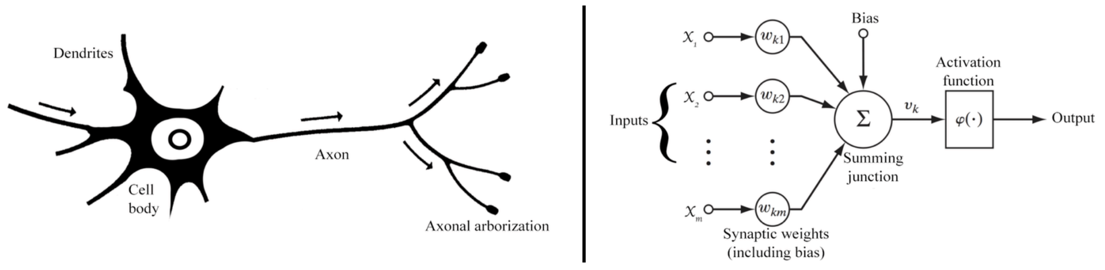

Figure 3.1: Comparison of biological neuron (left) and artificial neuron (right)

Although neuroscience has continued to play a part in the development of ANNs, many advances in this field were obtained by focusing on the mathematics of efficient optimization, rather than neuroscientific research [19].

3.2 The Perceptron

The most simple ANN that can be implemented was introduced by F. Rosenblatt and is known as perceptron, or single-layer perceptron [20]. As can be seen in figure 3.2, the main components of a single-layer perceptron include a vector of inputs $x \in \mathbb{R}^M$, the corresponding weights $w \in \mathbb{R}^M$ and the bias $b \in \mathbb{R}$.

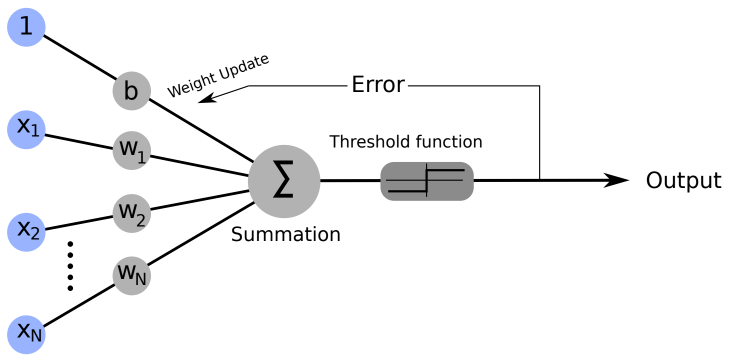

Figure 3.2: Structure of a perceptron.

Observing an input vector x, the neurons output y can be obtained by:

$$y(x, w) = f\big(\sum_i w_i x_i + b\big), \tag{3.1}$$

where $f(\cdot)$ represents a threshold function so that $f(x) = \begin{cases} 1 & \text{if } \sum\limits_i w_i x_i + b > 0, \\ 0 & \text{otherwise} \end{cases}$.

As the perceptron performs a binary decision, the aim is to correctly assign the input x to one of two classes, C_1 or C_2, where the decision rule states that x either belongs to class C_1 if $y(x, w) = 1$ or to class C_2 if $y(x, w) =$ -1.

The decision boundary, i.e. the border that describes the transition from C_1 and C_2, is parametrized by w and b and is defined by a hyperplane in $\mathbb{R}^D$, which corresponds to a line in $\mathbb{R}^2$. Hence, logical operations like AND, NOT and OR can be solved. However, it is not possible for a single-layer perceptron to solve the XOR problem, which is obviously a drawback [21]. In order to obtain a suitable hyperplane, the parameters are initialized to small non-zero values and adapted only when a misclassification occurs. This parameter modification is peformed iteratively until all samples are correctly classified and the algorithm converges [22].

By changing the activation $f(\cdot)$ to an arbitrary differentiable function and subsequently connecting and stacking these nodes, multi layer neural networks are obtained, which are further discussed in the following section.

3.3 Multi Layer Neural Network

A multi layer neural network, also known as multi layer perceptron (MLP), contains additional intermediate nodes between input and output layer. These intermediate units are known as hidden neurons, which form one or more hidden layers. While a single-layer perceptron is only able to learn linear functions, a MLP can also learn arbitrarily complex non-linear functions. Hence, it can be seen as a generalization of the previously described single-layer perceptron. The computational power of MLPs arises from the non-linear activation functions (see section 3.3.3) used within the nodes [22].

Both single layer perceptrons and multi layer perceptrons are examples of feedforward neural networks, where the information flows in only one direction from input neurons, through hidden neurons and finally to the output neurons without any cycles or loops [23]. In figure 3.3, an illustration of a MLP with three hidden layers and full interconnection is shown.

Hornik et al. [24] proved that a feedforward neural network with one hidden layer, containing a finite number of neurons, can approximate any bounded continuous function on a compact set of $\mathbb{R}^n$ to an arbitrary small error, under mild assumptions on the activation to be non-constant, monotonically-increasing and continuous. This statement is known as the universal approximation theorem. Furthermore, Leshno et al. [25] showed that a feedforward neural network with two hidden layers can approximate any desired function to any accuracy.

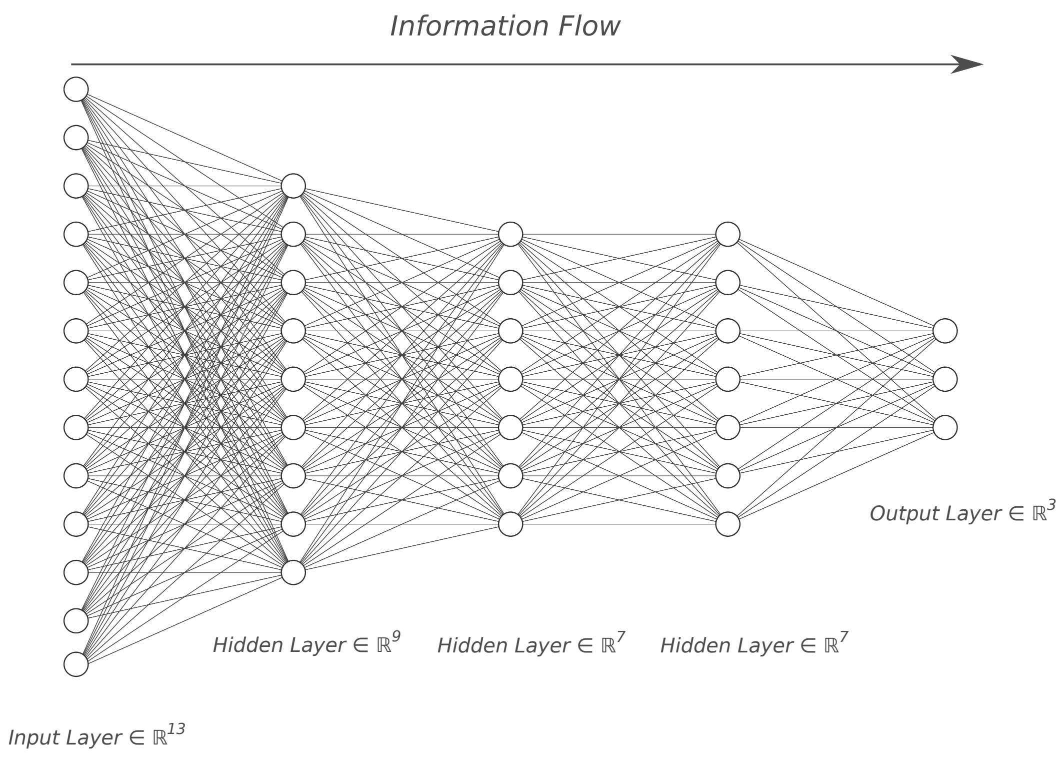

Figure 3.3: Illustration of a Multi Layer Perceptron (MLP) with three hidden layers and full interconnection.

Input Layer:

A layer of neurons that obtains information in form of specific features directly from the data. The received information is processed by a linear transformation and subsequently passed to the following hidden layer.

Hidden Layer:

Hidden neurons receive information from preceding input or hidden units. In such layers, the received information is processed by a non-linear transformation and subsequently passed to the following hidden or output layer. When a neural network is considered to be a black box, hidden units are not visible from the outside, hence the term "hidden" is used.

Output Layer:

Neurons in this layer receive already processed information from preceding hidden units. In this layer, a task specific non-linear transformation is performed, which yields the final result of the network.

The composition function of a neural network with one hidden layer is defined in the following way:

$$y_m(x, \theta) = f^{(2)} \left(\sum_{n=1}^{N} w_{mn}^{(2)} \, f^{(1)} \left(\sum_{k=1}^{K} w_{nk}^{(1)} \, x_k + b_n^{(1)} \right) + b_m^{(2)} \right), \tag{3.2}$$

where $\theta = \{w^{(1)} \in \mathbb{R}^{N \times K},\ w^{(2)} \in \mathbb{R}^{M \times N},\ b^{(1)} \in \mathbb{R}^{N},\ b^{(2)} \in \mathbb{R}^{M}\}$ represents the network parameters, $x \in \mathbb{R}^K$ describes the input and $y \in \mathbb{R}^M$ indicates the output.

When a larger number of hidden layers is employed, each representing a level of abstraction of the data, the neural network is considered deep. When using such deep neural networks, the associated framework is referred to as "deep learning".

For a number of hidden layers (L-1), the resulting composition function is provided as follows (bias was omitted for simplicity):

$$y(x, \theta) = \left(f^{(L)} \circ w^{(L)} \circ f^{(L-1)} \circ w^{(L-1)} \circ ... \circ f^{(1)} \circ w^{(1)} \right)(x) \tag{3.3}$$

3.3.1 Network Training

The composition function shown in equation 3.3 can be expressed in short form as:

$$y = \hat{f}(x, \theta), \tag{3.4}$$

where $\hat{f}(\cdot)$ is defined by the network architecture. Usually, network parameters are initialized randomly, which causes the network to produce no meaningful output, which is independent of the processed data. Therefore, the aim of the training procedure is to optimize the adjustable parameters θ so that $\hat{f}(x, \theta)$ approximates a desired function that is defined by the task to solve. This means that parameters θ have to be found, which minimize the difference between the desired output and the actual output [14]. Hence, a suitable error measure $E(\theta)$, also known as cost function, loss function (minimization) or objective function is required, which analytically describes the performance of the model for a specific task.

As shown in figure 3.4, a loss function $E(\theta)$ can be viewed as a surface defined on the parameter space, which yields high values in regions of bad performance and low values in regions of good performance. Since $E(\theta)$ is a smooth continuous function of network parameters θ, the global optimum (smallest error) is obtained at a point in parameter space where the gradient vanishes, i.e. $\nabla E(\theta) = 0$. However, there are usually also local optima for which the gradient vanishes as well. In general, it may not be necessary to find the global minimum of the loss function to successfully apply a neural network. Instead it is enough to find a sufficiently good local minimum [26].

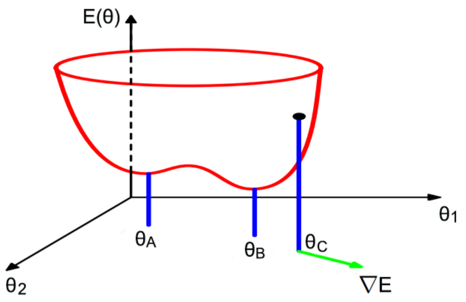

Figure 3.4: Illustration of an objective function E(θ) as a surface sitting over the parameter space, where θ_A indicates a local minimum and θ_B is the global minimum. The local gradient of $E(\theta)$ is given by vector ∇E, for an arbitrary point θ_C in the parameter space. Adapted from [26]

Various objective functions can be considered when training a neural network, depending on the task at hand. The mean squared error loss, for example, is most commonly used for regression tasks, whereas the cross-entropy loss is often used for classification.

Mean Squared Error loss:

$$E(\theta) = \frac{1}{2} \sum_{n=1}^{N} \left(\hat{f}(x_n, \theta) - t_n \right)^2, \tag{3.5}$$

where x denotes the training data, t represents the corresponding target values, N indicates the number of observations and $y = \hat{f}(x, \theta)$ corresponds to the network output parametrized with θ.

Generalized Cross-Entropy loss:

$$E(\theta) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log(\hat{f}_k(x_n, \theta)), \tag{3.6}$$

where x represents the training data, $t_{nk} \in \{0, 1\}$ denotes the one-hot-encoded target, N indicates the number of observations, $y = \hat{f}(x, \theta)$ corresponds to the output of the neural network representing a discrete probability distribution over K classes and θ denotes the model parameters.

3.3.1.1 Optimization of Network Parameters

As already mentioned, the network parameters θ should be adapted in a way that minimizes the cost $E(\theta)$ and leads to a desired behavior, i.e.

$$\min_{\theta} \quad E(\theta) \tag{3.7}$$

In contrast to many linear models where the associated parameters can be computed in closed form, the parameters in ANNs need to be adapted iteratively. This is the case because of the highly non linear function described by the network, which leads to a non-linear non-convex cost function $E(\theta)$. The update of the parameters θ for iteration step k can be computed by the gradient of the cost function $E(\theta)$ with respect to the parameters θ, i.e.

$$\theta^{k+1} = \theta^k - \alpha \; \nabla_\theta \; E(\theta^k) \tag{3.8}$$

As the direction of a gradient points towards an increase of the function, the parameters are changed in the direction of the negative gradient with a certain step size $\alpha \in \mathbb{R}_+$. The optimization algorithm shown in equation (3.8) is known as gradient descent. In practice, however, more sophisticated gradient descent based methods are used, such as those discussed in section 3.3.1.3.

3.3.1.2 Backpropagation

Backpropagation is a well-known algorithm used to efficiently compute the gradient updates of the network parameters θ. First proposed by Hinton et al. [27], its name refers to the fact that an error is calculated at the network output and is passed backwards through all network layers to obtain the individual partial derivatives required to perform the parameter updates. Consequently, for an L-layer neural network, the output of the l-th layer is provided by:

$$a^{(l)} = f(\underbrace{w^{(l)} \; a^{(l-1)} + b^{(l)}}_{z^{(l)}}) \tag{3.9}$$

With an objective function E, the gradient of the parameters in the l-th layer can be computed using the chain rule:

$$\frac{\partial E}{\partial w^{(l)}} = \frac{\partial E}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(l+1)}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial w^{(l)}}, \tag{3.10}$$

where

$$\frac{\partial a^{(l+1)}}{\partial a^{(l)}} = \frac{\partial a^{(l+1)}}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial a^{(l)}} = f^{'}(z^{(l)})(w^{(l+1)})^T \tag{3.11}$$

For the sake of efficiency, partial computations from a subsequent layer, known as error message $\delta^{(l+1)}$, are used to calculate the gradient of its preceding layer [28]. The error message $\delta^{(l)}$ is computed by:

$$\delta^{(l)} = \frac{\partial a^{(l+1)}}{\partial z^{(l+1)}} \odot \left(\frac{\partial z^{(l+1)}}{\partial a^{(l)}} \cdot \delta^{(l+1)} \right) \tag{3.12}$$

Finally, the gradient update of the l-th layer is computed by:

$$\frac{\partial E}{\partial w^{(l)}} = \delta^{(l+1)} \frac{\partial z^{(l)}}{\partial w^{(l)}} = \delta^{(l+1)} (a^{(l)})^T, \tag{3.13}$$

where $a^{(0)}$ represents the input data x of the neural network.

The parameter update then follows according to equation (3.8). A graphical representation of the principle of the backpropagation algorithm is illustrated in figure 3.5.

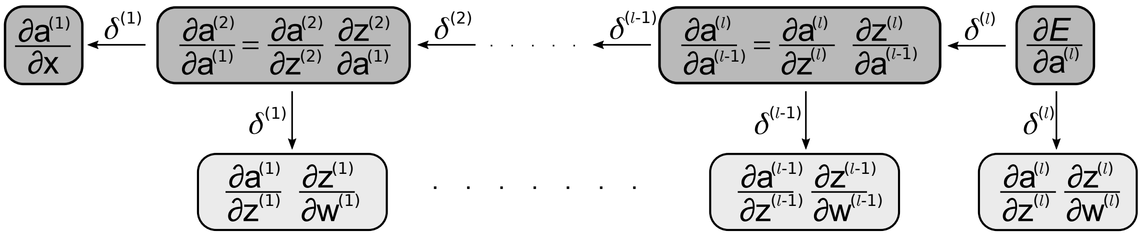
$\frac{\partial a^{(1)}}{\partial x}$
$\delta^{(1)}$
$\frac{\partial a^{(2)}}{\partial a^{(1)}} = \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}}$
$\delta^{(2)}$
$\delta^{(l-1)}$
$\frac{\partial a^{(l)}}{\partial a^{(l-1)}} = \frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial a^{(l-1)}}$
$\delta^{(l)}$
$\frac{\partial E}{\partial a^{(l)}}$
$\delta^{(1)}$
$\delta^{(l-1)}$
$\delta^{(l)}$
$\frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w^{(1)}}$
$\frac{\partial a^{(l-1)}}{\partial z^{(l-1)}} \frac{\partial z^{(l-1)}}{\partial w^{(l-1)}}$
$\frac{\partial a^{(l)}}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial w^{(l)}}$

Figure 3.5: Concept of the backpropagation algorithm.

3.3.1.3 Gradient-based Optimization Algorithms

An unconstrained optimization problem is defined as follows:

$$\min_{\theta} E(\theta), \tag{3.14}$$

where $\theta \in \mathbb{R}^n$ is the optimization variable and $E : \mathbb{R}^n \to \mathbb{R}$ is a continuous differentiable function, for example an objective function for training a neural network. The aim of gradient based methods is to minimize $E(\theta)$ by iteratively updating θ with respect to the gradient of the objective function. Many large scale applications, such as deep learning, base their optimization on gradient information. In the following, all variants of gradient methods used in this thesis will be discussed.

Gradient Descent:

In order to move towards the point of minimum function value $E(\theta^*)$, gradient descent (also known as steepest descent) iteratively updates the optimization variable θ by subtracting the gradient of the objective function in the current step k:

$$\theta^{k+1} = \theta^k - \alpha^k \nabla_\theta E(\theta^k), \tag{3.15}$$

where $\alpha \in \mathbb{R}_+$ denotes the step size of the update step.

Gradient Descent with Momentum:

Since the steps performed by vanilla gradient descent always point towards the steepest gradient direction, oscillations can be observed in regions where the error surface has significantly different curvature along different dimensions, which slows down convergence speed. To overcome this issue Polyak [29] introduced the "heavy ball method", also known as momentum. With this method, oscillations are damped by adding an additional term, which accumulates a velocity vector in gradient directions that is persistent over iterations:

$$\theta^{k+1} = \theta^k - \alpha^k \, \nabla_\theta E(\theta^k) + \beta^k \, (\theta^k - \theta^{k\text{-}1}) \tag{3.16}$$

where $\beta^k \in [0, 1)$ describes the influence of the momentum term and $\alpha \in \mathbb{R}_+$ denotes the step size. A comparison between vanilla gradient descent and gradient descent with momentum is visualized in figure 3.6.

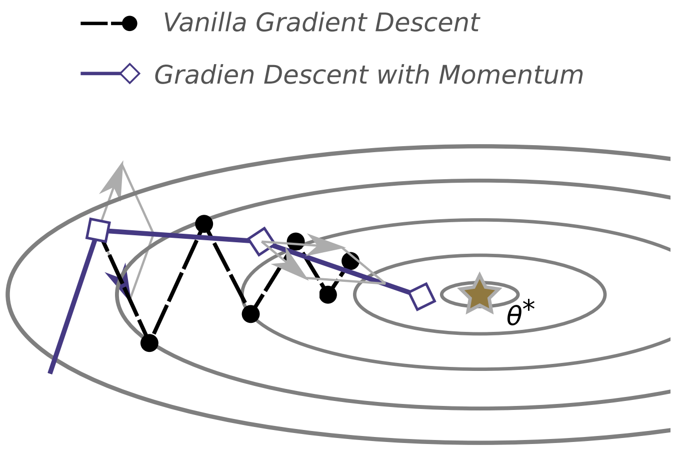

Figure 3.6: Vanilla gradient descent vs gradient descent with momentum.

Nesterov Momentum:

In 1983 J. Nesterov [30] proposed a method closely related to that of classical momentum. The difference between those methods is that classical momentum evaluates the gradient in the current parameters θ^k, while in Nesterov momentum the gradient is obtained at approximate future parameters $\bar{\theta}^k$:

$$\begin{aligned}\bar{\theta}^k &= \theta^k + \beta^k(\theta^k - \theta^{k-1}) \\ \theta^{k+1} &= \bar{\theta}^k - \alpha^k \nabla_\theta E(\bar{\theta}^k)\end{aligned} \tag{3.17}$$

where $\beta^k \in [0, 1)$ describes the influence of the Nesterov momentum term and $\alpha \in \mathbb{R}$ denotes the step size.

Updating θ in this manner has the advantage, of changing v in a quicker and more responsive way, which leads to the optimal convergence rate of $\mathcal{O}(1/k^2)$ for convex functions with Lipschitz-continuous gradient [31].

Adam:

Adaptive Moment Estimation (Adam) [32] is an optimization algorithm that has its main application area in the field of deep learning. It is based upon AdaGrad [33], an algorithm that works well if sparse gradients are obtained, as well as RMSProp [34], which is used in on-line and non-stationary settings.

Similar to momentum, Adam computes an exponential average over past gradients m_t, but also keeps track of the squared gradients s_t (second order moments) via exponential averaging:

$$\begin{aligned}m^k &= \beta_1 \, m^{k-1} + (1 - \beta_1) \, \nabla E(\theta^k) \\ s^k &= \beta_2 \, s^{k-1} + (1 - \beta_2) \, \nabla E(\theta^k)^2,\end{aligned} \tag{3.18}$$

where $\beta_1, \beta_2 \in \mathbb{R}_+$ weight the exponential decay rates.

As m and s are initialized as 0-vectors, they are significantly biased towards 0. This is especially the case for high values of β_1, β_2 as well as in the beginning of the optimization. Therefore, m and s are bias corrected:

$$\begin{aligned} \hat{m}^k &= \frac{m^k}{1-(\beta_1)^k} \\ \hat{s}^k &= \frac{s^k}{1-(\beta_2)^k} \end{aligned} \tag{3.19}$$

with the bias-corrected $\hat{m}$ and $\hat{s}$ the Adam update scheme is as follows:

$$\theta^{k+1} = \theta^k - \frac{\alpha}{\sqrt{\hat{s}^k} + \epsilon} \hat{m}^k, \tag{3.20}$$

where $\epsilon \in \mathbb{R}$ is a small value that ensures numerical stability and $\alpha \in \mathbb{R}_+$ denotes the step size. Dividing by the estimated second order moments has the effect that updates in parameters with high s tends to approach 0. This behavior is desirable, as s is an estimate of the second order moment (the uncentered variance) that may be seen as describing the uncertainty about whether m points towards the true direction of the optimum. Therefore, gradient update steps obtained with Adam tend to reliable move towards the optimal direction.

3.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep neural network, primarily used for image processing tasks, which use convolution instead of general matrix multiplication in at least one of their layers, hence the name [23]. Traditional deep neural networks are not particularly suitable for image processing since the computational complexity increases strongly with increasing size of the input image, due to the full connectivity (all neurons are connected to every neuron in the previous layer) [35]. Furthermore, a key property of images, which is the presence of spatial information, is ignored in traditional neural network approaches [26]. CNNs incorporate several architectural ideas to allow a certain level of shift and distortion invariance and to reduce computational effort, namely local sparse connectivity, shared weights and pooling [36]. An example of a simple CNN architecture is illustrated in figure 3.7. The basic building blocks of CNNs comprise three main types of layers, which are convolutional layers, pooling layers and fully-connected layers [35].

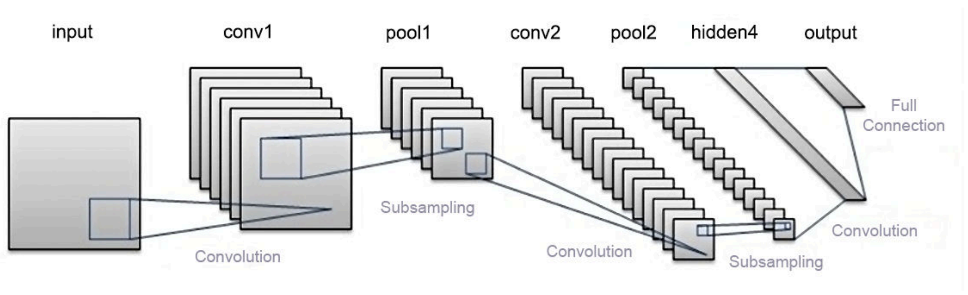

Figure 3.7: Typical architecture of a convolutional neural network.

Convolutional Layer: This layer is the main building block of a CNN, where neurons are structured in feature maps. Each unit in a feature map is connected to a local area (receptive field) of the preceding layer, which means that inputs of layer l are obtained from a subset of neurons in layer l-1. All neurons within the same feature map share an identical weight matrix, also called filter bank, since they aim to recognize the same pattern at different spatial locations of the image. Different feature maps, however, use different filter banks, since they focus on different patterns [37]. Finally, the output of a unit is obtained by applying a non-linear activation function to the local weighted sum. By employing such a local connectivity pattern and sharing weights, the computational effort of a CNN can be largely reduced.

An illustration of the convolution operation can be seen in figure 3.8.

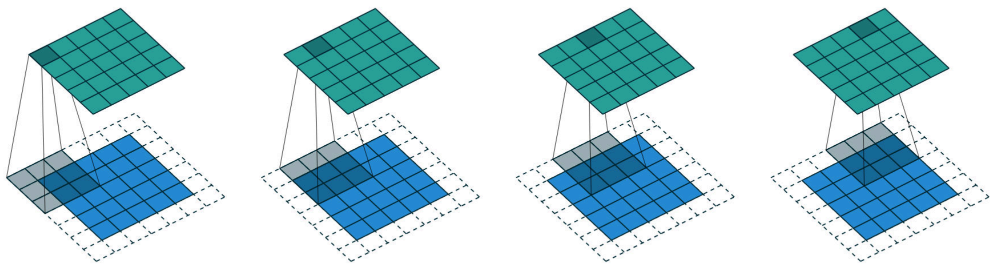

Figure 3.8: Illustration of a convolution operation using a 3×3 kernel on a 5×5 input with stride 1 and same padding, which results in a 5×5 output.

Pooling Layer: Usually, a pooling layer follows a subsequent convolutional layer in order to locally combine pixels and extract information [37]. The max-pooling operation is most frequently used in practice and represents a type of non-linear down-sampling. It can be viewed as a max filter, where a $n \times n$ area is replaced by its maximum pixel value. Less frequently used forms of pooling include average-pooling, stochastic pooling and winner-takes-all pooling [28].

Fully Connected Layer: Fully connected layers, where each neuron is connected to every neuron in the preceding layer, are often used as final layers of CNNs in order to encode position-dependent information and global patterns after several stages of convolution, non-linear activation and pooling [14].

3.3.2.1 Transposed Convolution

Classical pattern recognition networks, such as LeNet [38] or AlexNet [39], use fully connected final layers that produce non-spatial outputs. However, there is an abundance of problems where a spatial output map is desired instead. In order to address such problems, it is necessary to upsample feature maps within the network, for example, by using interpolation. For instance, simple bilinear interpolation considers the closest 2x2 neighborhood surrounding an unknown pixel and computes a weighted average of these four adjacent values. In general, upsampling with factor q corresponds to a convolution operation with a fractional input stride of $1/q$. Therefore, a natural way to perform umpsampling is to use backwards convolution, also known as convolution with fractional strides, transposed convolution or deconvolution [40]. However, the term ‘deconvolution‘ is unfortunate and confusing, as the upsampling operation does not correspond to the actual deconvolution operation described in mathematics. In contrast to bilinear upsampling, transposed convolution kernels do not need to be fixed, which makes the upsampling filter learnable. Even a non-linear upsampling can be learned by stacking transposed convolution layers and activation functions [40]. Figure 3.9 illustrates a 3x3 transposed convolution using stride 2 and pad 1 on a 2x2 input map, resulting in a 4x4 output map.

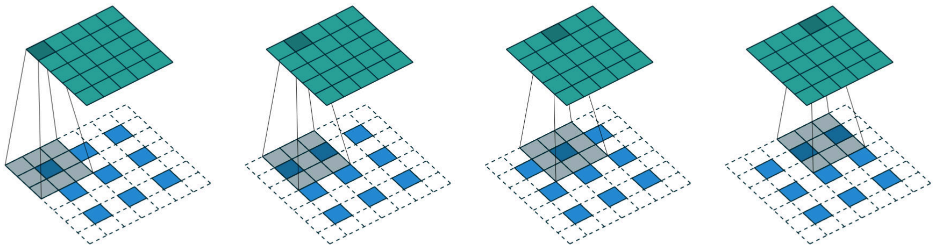

Figure 3.9: Illustration of a 3×3 transposed convolution operation, stride 2 pad 1. The input map is of dimension 2×2 and the resulting output map is of dimension 4×4, accordingly.

3.3.3 Activation Functions

A linear model that maps from feature to output space using solely matrix multiplications can represent, by definition, only linear functions. Therefore, activation functions are an essential building block of neural networks, as they introduce non-linearity in the transformation. Such a non-linear transformation in $\mathbb{R}^2$ is illustrated in figure 3.10 and shows a feature space that is linearly separable after transformation. [23].

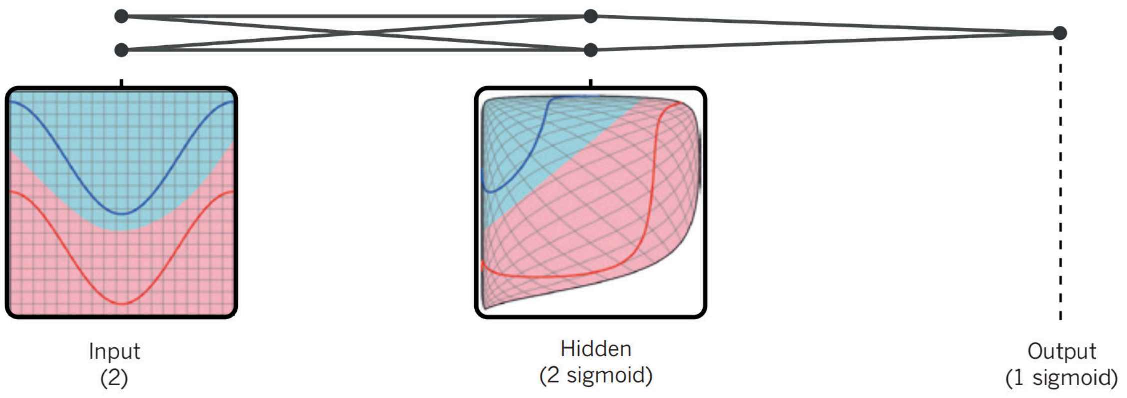

Figure 3.10: Non-linear transformation of feature space, taken from [23]

Some desirable properties of activation functions are discussed below:

- **Non-linearity** – Using non-linear activations enables the network to approximate highly complex non-linear functions, which is usually desired in practice. In contrast, this cannot be achieved by using linear activations.
- **Boundedness** – Bounded activation functions lead to robust optimization during training of deep neural networks. The trend however goes towards using unbounded functions, which show better performance for deep architectures [41].
- **Differentiability** – A necessary condition for gradient descent based optimization methods is that the function $f : \mathbb{R}^n \to \mathbb{R}$ is continously differentiable over $\mathbb{R}^n$ [42].

The choice of the non-linearity, however, is not trivial and depends on the task at hand as well as on the network architecture. The most frequently used activation functions are discussed in the following.

Linear Activation:

The linear or identity activation function is particularly used in the output of neural networks for regression tasks, since $f : \mathbb{R} \to \mathbb{R}$. It is mathematically described by:

$$f(x) = x \tag{3.21}$$

along with its derivative:

$$f'(x) = 1 \tag{3.22}$$

Softmax Activation:

From a probabilistic point of view, the softmax activation function takes a vector and squashes it into a probability distribution. It is therefore frequently used in the output layer of multi-class classification networks. For an output layer with N neurons, it affects all output activations, i.e. $f : \mathbb{R}^N \to [0,1]^N$, and scales the activations such that $\sum_{i=1}^{N} f(x)_i = 1$. It is defined by:

$$f(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \qquad \forall i \in 1..K \tag{3.23}$$

along with its gradient:

$$\frac{\partial f(x)}{\partial x_i} = \begin{cases} f(x)_i(1 - f(x)_j) & , i = j \text{ main diag. of Jacobian} \\ -f(x)_i f(x)_j & , i \neq j \text{ off diag. of Jacobian} \end{cases} \tag{3.24}$$

Hyperbolic tangent Activation (*tanh*):

The hyperbolic tangent function is a typical choice of non-linearity in hidden layers, as well as in output layers if the desired result has to remain in a certain range, i.e. $f : \mathbb{R} \to$ [-1, 1]. The function is defined by:

$$f(x) = \tanh(x) \tag{3.25}$$

along with its derivative:

$$f'(x) = 1 - f(x)^2 \tag{3.26}$$

In neural networks, where many hidden layers use tanh, saturation of the function and its gradient can be observed, which is known as vanishing gradient problem [43]. This issue occurs when lover level parameters receive almost no gradient update due to saturation at the asymptotes of the tanh in higher level units, see figure 3.11.

Rectified Linear Activation (ReLU):

The ReLU activation function is able to overcome the vanishing gradient problem, as it is unbounded $f : \mathbb{R} \to \mathbb{R}_0^+$. If a hidden neuron with this type of activation function receives an input above 0, its derivative becomes 1, which makes vanishing gradients impossible along paths with active hidden neurons [41]. The ReLU function is defined as:

$$f(x) = \max(x, 0) = \begin{cases} x, & x > 0 \\ 0, & else \end{cases} \tag{3.27}$$

along with its derivative:

$$\partial f(x) = \begin{cases} 0, & x < 0 \\ [0, 1], & x = 0 \\ 1, & x > 0 \end{cases} \tag{3.28}$$

As can be seen in equation (3.28) and figure 3.11, the ReLU function is non-differentiable at $x = 0$, as infinitely many slopes can be found in the interval $[0, 1]$. In this case, the gradient is simply set to 0 in practice, which makes the function applicable in gradient-descent based methods [41].

Leaky Rectified Linear Activation (LReLU):

The leaky-ReLU activation function is an adapted version of ReLU that features small non-zero gradients if the neurons input is < 0. The mathematical definition of the LReLU function is:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha x, & else \end{cases} \tag{3.29}$$

where $\alpha \in [0, 1)$, along with its derivative:

$$\partial f(x) = \begin{cases} \alpha, & x < 0 \\ [\alpha, 1], & x = 0 \\ 1, & x > 0 \end{cases} \tag{3.30}$$

In practice, for the LReLU function the gradient at $x = 0$ is set to α.

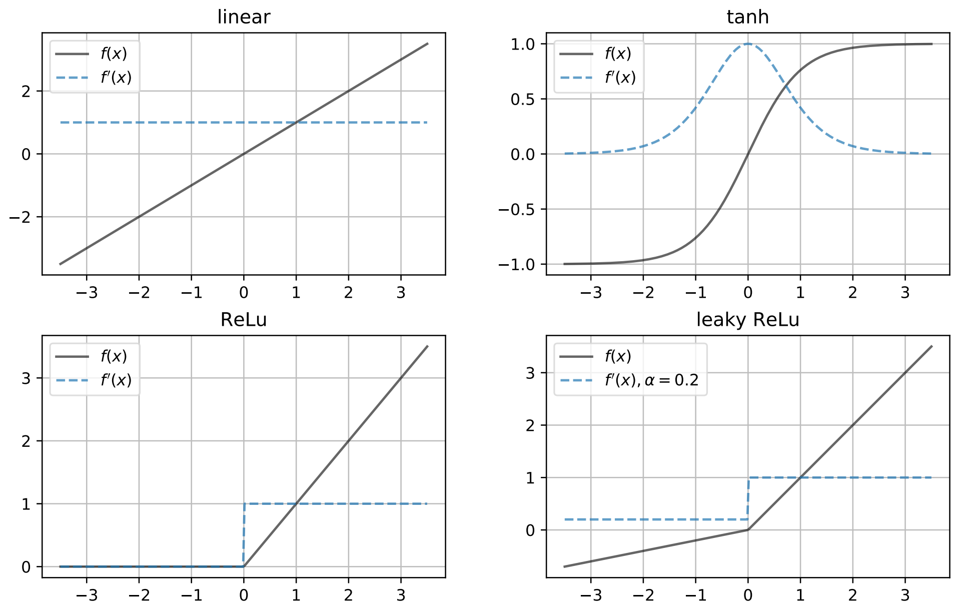

Figure 3.11: Activation functions along with their corresponding gradient functions

3.3.4 Batch Normalization

Batch normalization [44] is a widely used method in deep learning to normalize activations in intermediate network layers, which significantly speeds up training and improves the prediction accuracy. The idea of normalizing the input data to zero-mean and constant standard deviation is a well studied method to improve neural network training. Batch normalization is an extension of this approach, which incorporates this concept in intermediate layers of deep neural networks. However, the normalization is applied on mini batches instead of the full training set for computational speed up [45].

For a single layer, normalization, i.e. mean $\mu = 0$ and variance $\sigma^2 = 1$ of the k-th dimension in a d-dimensional input $z \in \mathbb{R}^d$ of m batches is observed as:

$$\hat{z}^{(k)} = \frac{z^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{3.31}$$

where a small scalar $\epsilon \in \mathbb{R}$ guarantees numerical stability, $\mu_B = \frac{1}{m}\sum_{i=1}^{m} z_i^{(k)}$ represents the mean of the k-th dimension in a mini-batch with m samples and the according variance $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(z_i^{(k)} - \mu_B)^2$.

Performing normalization can change the activation of a layer in an undesired way, as it is the case for sigmoidal activations, where a normalized input constrains the output to the linear region of the sigmoid function. This issue is overcome by introducing additional trainable parameters $\gamma^{(k)}$ and $\beta^{(k)}$, which scale and shift the normalized value [44]:

$$\tilde{z}^{(k)} = \gamma^{(k)} \hat{z}^{(k)} + \beta^{(k)} \tag{3.32}$$

This allows the transformation, introduced by batch-normalization, to result in an identity transform, i.e. $z^{(k)} = \tilde{z}^{(k)}$ if $\gamma^{(k)} = \sigma_B$ and $\beta^{(k)} = \mu_B$, if no reparametrization is required.

3.3.5 Regularization

The ultimate aim of training a deep neural network is to obtain a model that does not only perform well on the finite training set, but also generalizes well to new and unseen data (test set). To achieve this, several techniques that are especially designed to reduce the test error, while accepting an increased training error, can be used. These specific techniques are commonly referred to as regularization [23].

There are two types of prediction errors, namely the irreducible error and the reducible error. Let f be the true relationship between the data X and the associated target Y, and $\hat{f}$ the corresponding estimate. The mismatch between f and $\hat{f}$ is known as the reducible error, as it can be reduced by adjusting the model. Since Y is not completely determined by X, there also exists an irreducible error (inherent uncertainty), which is associated with unknown or unpredictable variables. Therefore, this error can not be improved. The reducible error consists of two components - bias and variance - that represent two different sources of prediction error in an estimator [23]. On the one hand, the bias provides a measure of the difference between the expected prediction of the model and the true value. A high bias leads to underfitting, which corresponds to an overly simple model that ignores important relationships between input features and target outputs. On the other hand, the variance gives a measure of how much the expected prediction value varies as a function of the data sample. High variance causes the model to capture random noise in the training data, which leads to a reduced generalization ability on unseen data and is known as overfitting [26]. Figure 3.12 illustrates different combinations of both high and low bias and variance. Obviously, an ideal model would have low bias and low variance, in practice, however, there is always a trade-off between the two.

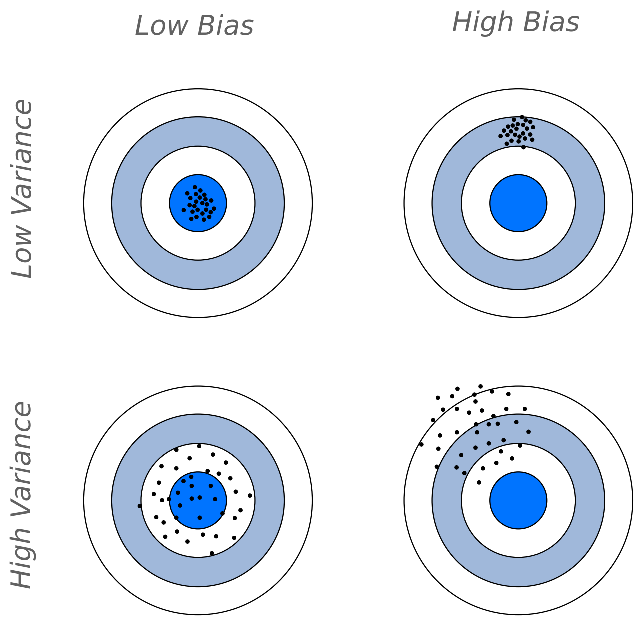

Figure 3.12: Illustration of bias variance trade-off.

A simple remedy for the problem of underfitting is to increase the complexity of the network. In order to address overfitting, more specialized techniques known as regularization methods are required. In the field of optimization, traditionally, the term "regularization" is only used for penalty terms in the loss function [46]. However, it has recently acquired a wider meaning, which is defined as follows by Kukačka et al. [47]:

"Regularization is any supplementary technique that aims at making the model generalize better, i.e. produce better results on the test set. This can include various properties of the loss function, the loss optimization algorithm, or other techniques."

This general definition is in accordance with machine learning literature, whereas more restrictive definitions are found in inverse problems literature.

3.3.5.1 ℓ^2-regularization

One of the most used regularization methods is ℓ^2 regularization, also known as weight decay, Tikhonov regularization or ridge regression. This regularization approach aims to penalize large model weights, since smaller weights are found to be less specialized and more regular (a good model does not only focus on specific features). Penalizing the model based on the size of its weights is also called weight or parameter norm penalty.

A simple way to penalize large model weights is to add a penalty term to the original loss function E. In the case of ℓ^2 regularization, the weights are penalized by additionally computing the Euclidean norm of the model weights:

$$\min_{\theta} E(\theta) + \frac{\lambda}{2} ||\theta||^2 , \tag{3.33}$$

where θ are the model weights and λ indicates the tuning parameter that weights the importance of the regularization term relative to the loss E.

Since the ℓ^2 norm is used as a penalty term, the model weights are constrained to lie in the ℓ^2 norm ball, which is illustrated in figure 3.13. The exact size of the resulting constrained region is not known, however, the regularization strength can be roughly controlled by adjusting the λ parameter. A smaller λ leads to a larger constraint region, whereas a larger λ leads to a smaller constraint region [23]. If too much focus is put on the penalty term (large λ), the model underestimates the weights, which leads to underfitting. In contrast, when the weight penalty is too weak (small λ), overfitting can not be prevented appropriately.

When no regularization is applied, the objective is to find the global optimum of the loss function. By adding the ℓ^2 penalty term, the overall objective changes to minimizing the original loss function while keeping the weights within the ℓ^2 norm ball. This approach encourages the model toward using small weights and thereby prevents overfitting. Therefore, ℓ^2 regularization has the effect of improving the generalization ability of the network.

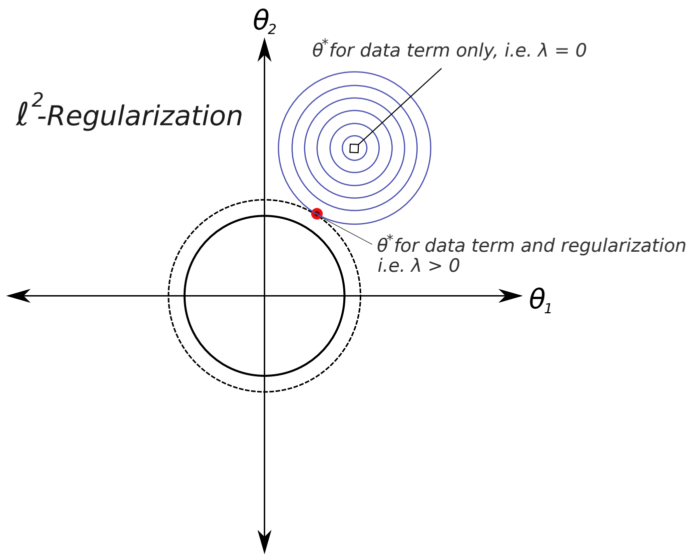
θ_2
θ^* for data term only, i.e. $\lambda = 0$
ℓ^2-Regularization
θ^* for data term and regularization i.e. $\lambda > 0$
θ_1

Figure 3.13: Illustration of the ℓ^2 norm ball.

4 Generative Adverserial Networks

Contents

In general machine learning approaches can be divided in two distinct groups:

- **Discriminative Algorithms:**, which directly learn mappings from the space of inputs to labels $f : \mathcal{X} \to \mathcal{T}$ by modeling the conditional distribution $\mathbb{P}(t|x)$ of the labels t given the data x.

- **Generative Algorithms:**, which try to model the data distribution $\mathbb{P}(x)$ in case of unlabeled data, respectively the joint distribution $\mathbb{P}(x, t)$ of data x and labels t for labeled data. For classification problems, finding the posterior can then simply be done by applying Bayes rule: $\mathbb{P}(t|x) = \frac{\mathbb{P}(x|t)\mathbb{P}(t)}{\mathbb{P}(x)}$ that follows from the product rule of probability $\mathbb{P}(x, t) = \mathbb{P}(x|t)\mathbb{P}(t)$.

For most machine learning tasks, discriminative algorithms are favored since in general less computational resources and data samples are needed. However, generative models explicitly or implicitly model the data distribution and therefore have the additional ability to generate new data by sampling from the learned distribution.

GANs represent a type of generative machine learning algorithm that implicitly models the data distribution $p(x)$. Rather than maximizing the likelihood of a parametric distribution on the data, as done in classical approaches, the algorithm learns a parametric function $G(\cdot)$, called the generator, that directly maps from a latent space $z \in \mathbb{R}^d$ to the data space $x \in \mathbb{R}^{|x|}$, i.e. $G : \mathbb{R}^d \to \mathbb{R}^{|x|}$, where $| \cdot |$ denotes the number of dimensions, usually $d << |x|$ [48].

Initially proposed by I. Goodfellow et al. [49] in 2014, GANs are topic of vast research and have experienced continuously improvement since then. In a variety of applications, such as image synthesis [50], semantic image editing [51], style transfer [52] and image super-resolution [53], state-of-the-art algorithms are based on GANs or include parts of the GAN framework. Furthermore, the generator of a trained GAN serves as an implicit prior that constrains the reconstruction to the domain of source images, which can be utilized to solve inverse problems.

In this chapter the underlying basic concepts and mathematical principles of the GAN framework are discussed.

4.1 Foundations of probability theory

Following Klenke [54], a probability space is composed of the triple $(\Omega, \mathcal{A}, \mathbb{P})$, where $\Omega \neq \emptyset$ defines a space, $\mathcal{A}$ denotes a σ-Algebra in this space and $\mathbb{P}$ is a probability measure in Ω. In detail, throughout this thesis it is assumed that Ω is a compact metric space. Furthermore, a set $\mathcal{A} \subset 2^\Omega$ is a σ-Algebra if $\Omega \in \mathcal{A}$, the complement set $A^c = \Omega \backslash A$ is contained in $\mathcal{A}$ for all $A \in \mathcal{A}$ and any union of sets in $\mathcal{A}$ is again contained in $\mathcal{A}$, i.e. for all $\{A_i\}_{i \in \mathbb{N}} \subset \mathcal{A}$ one has $\bigcup_{i \in \mathbb{N}} A_i \in \mathcal{A}$. If not otherwise stated, we assume that $\mathcal{A}$ is the Borel σ-algebra, which is the smallest σ-algebra that contains all open sets in Ω. The set of probability measures on Ω is further denoted by $\text{Prob}(\Omega)$, which is defined as the space of non-negative measures such that $\mathbb{P}(\Omega) = 1$ for all $\mathbb{P} \in \text{Prob}(\Omega)$. For any random variable Y on the aforementioned probability space, the expectation value $\mathbb{E}[Y]$ of Y is given by

$$\mathbb{E}[Y] = \int_\Omega Y \mathrm{d}\mathbb{P}.$$

Furthermore, if $\mathbb{P}$ is absolutely continuous with respect to the Lebesgue measure and thus a density function f exists, then the expectation value admits the representation

$$\mathbb{E}[Y] = \int_\Omega y f(y) \mathrm{d}y.$$

The expectation value of a random variable undergoing a transformation g is given by

$$\mathbb{E}[g(Y)] = \int_\Omega g(Y) \mathrm{d}\mathbb{P}.$$

In this thesis, the expectation value is frequently approximated as follows

$$\mathbb{E}[g(Y)] \approx \frac{1}{N} \sum_{i=1}^{N} g(Y_i),$$

where $\{Y_i\}_{i=1}^N$ denote independently sampled values drawn from the probability distribution. This approach is justified by the law of large numbers.

Throughout this thesis, multiple different probability spaces are considered. To highlight the probability space under consideration, the current random variable as well as the probability measure are added as a subindex of $\mathbb{E}$.

4.2 Underlying Principles

GANs are generative machine learning algorithms that are composed of two parametric models, which are usually represented as deep neural networks and trained in a competitive manner. Those two models are the generative model G that samples from a learned distribution $\mathbb{P}_\theta$ and the discriminative model D that estimates the probability of a generated sample belonging to the distribution of real data $\mathbb{P}_r$, see figure 4.1.

For the sake of illustration, the generator can be thought of as a counterfeiter that produces fake paintings, while the discriminator can be seen as an art expert that has knowledge about the domain and can distinguish between fake and real paintings. In this context, the overall goal of the generator is to produce fake paintings that can not be distinguished from real paintings by the discriminator.

In order to generate high quality samples, the distribution learned by the generator $\mathbb{P}_\theta$ should be a close approximation of the data distribution $\mathbb{P}_r$. Instead of directly estimating the density of $\mathbb{P}_r$, a random variable Z with distribution $\mathbb{P}_Z$ is introduced. Passing Z through the generator yields samples that follow the generator distribution $\mathbb{P}_\theta$. Under the assumption of an optimal generator, $\mathbb{P}_\theta$ equals the data distribution $\mathbb{P}_r$, as shown in [49].

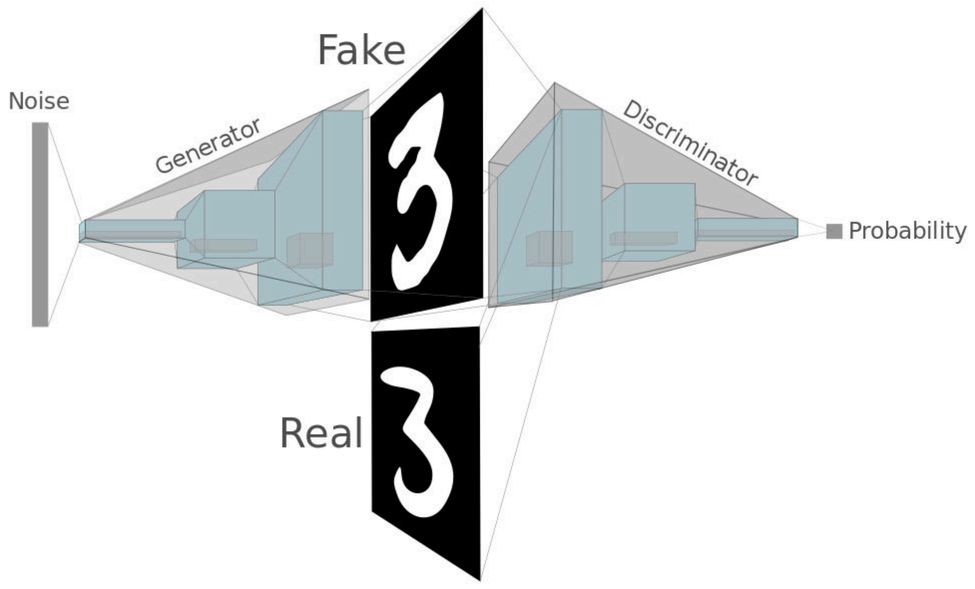

Figure 4.1: Concept of a Generative Adverserial Network

More formally, the aim of the discriminator is to decide whether a sample comes from the data distribution $\mathbb{P}_r$ or the learned distribution $\mathbb{P}_\theta$, i.e. it should assign high probabilities to real and low probabilities to generated samples. The goal of the generator in contrast, is to produce samples, which the discriminator classifies as real data with high probability. Therefore, the objective function of G and D can be observed as the following two-player minimax game:

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathbb{E}_{z\sim\mathbb{P}_Z}[\log(1 - D(G(z))] \tag{4.1}$$

In practice, this minimax game is realized by consecutive optimization, as can be seen in algorithm 1. Furthermore, optimizing the discriminator to completion for every update of the generator would not be feasible and would result in overfitting in case of a finite dataset. Hence, [49] suggests an unbalanced update scheme, where the discriminator is optimized k times for each generator update, which leads to the discriminator being maintained near its optimal solution as long as the generator changes slowly.

According to [49], the unique optimal discriminator D for a given generator G can be observed as:

$$D^*_G(x) = \frac{\mathbb{P}_r(x)}{\mathbb{P}_r(x) + \mathbb{P}_\theta(x)}. \tag{4.2}$$

This leads to the optimal discriminator predicting a value of 0.5 for all samples if $\mathbb{P}_\theta = \mathbb{P}_r$. Furthermore, the authors show that for an optimal discriminator, training the generator is equivalent to minimizing the Jensen-Shanon divergence between $\mathbb{P}_\theta$ and $\mathbb{P}_r$.

Algorithm 1: Training of a Generative Adverserial Network in mini-batch gradient descent mannor, as proposed in [49].

```
Require: $\alpha$ - learning rate; $\theta_0$ - initial $G$ parameter; $w_0$ - initial $D$ parameter;
$m$ - batch size; $k$ - amount of $D$ updates
begin
    for training iterations do
        for $k$ steps do
            Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ batch of training samples
            Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
            $w \longleftarrow w - \alpha \cdot \nabla_w \frac{1}{m} \sum_{i=1}^m [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))]$

        Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
        $\theta \longleftarrow \theta - \alpha \cdot \nabla_\theta \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)}))$
```

A problem that often arises in the classical GAN formulation is that of mode collapse. If the generator overpowers the discriminator, samples resulting from different z all collapse to the same value $G(z)$ and, therefore, the variety of the samples produced by the generator is lost.

Another common issue is that of diminishing gradients, which occurs due to the Jensen-Shannon divergence delivering more reliable but very small gradients as the discriminator improves [55]. This issue has motivated research towards finding novel objective functions based on Wasserstein-distance, which will be discussed in section 4.4.

4.3 DCGAN

The first GAN implementations have been based on fully connected architectures in the generator as well as in the discriminator. This approach works reasonable well for non-complex datasets such as MNIST but delivers noisy and non meaningful images for more complex datasets such as cifar-10 [48].

In order to improve bad image quality, architectures were changed from fully connected to convolutional neural networks. However, Radford et al. [56] stated that common architectures used in the supervised literature do not perform well in the GAN framework. Nevertheless, the authors went through extensive model exploration and found certain architectural constraints that describe a family of convolution based architectures, which can be used for GANs to create higher resolution samples. Their work introduces Deep Convolutional Generative Adversarial Networks (DCGAN), which are build on the following implementation guideline for GANs using convolutional architectures:

- **Strided Convolutions** are used instead of deterministic spatial pooling functions (such as max pooling), allowing the discriminator and the generator to learn their own spatial down- respectively up-sampling.
- **Batch Normalization** is used in both the discriminator and the generator, to improve gradient flow in deeper models and minimize problems that arise due to bad initialization.
- **ReLU activations** are used in the generator, except for the last layer that yields the final image. Here, a *tanh* activation is used in order to force the model output to stay in the range of [-1, 1]. In the discriminator **Leaky-ReLU activations** have shown to be usefull in all convolutional layers.

Figure 4.2 shows the generator architecture proposed in [56]. The generator takes a random vector z that is mapped into a feature volume by applying a fully convolutional layer with reshaped output. Subsequently, the following layers use transposed convolutions to upsample the feature maps by a factor of 2 until the desired image size is obtained. The discriminator architecture basically corresponds to the reversed generator architecture, i.e. it takes an image that is subsequently down-sampled by strided convolutions. The feature maps of the last convolution layer are flattened and processed by a fully connected layer with a single sigmoid output in order to obtain a probability for a given sample.

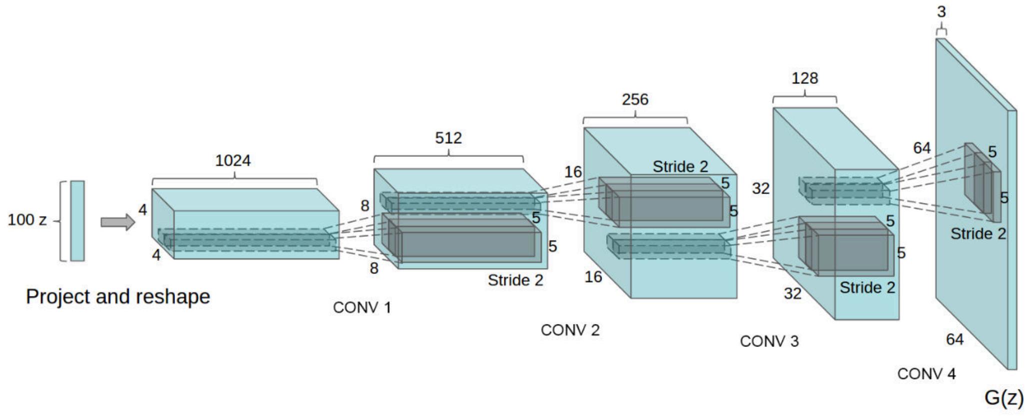

Figure 4.2: DCGAN generator architecture. Adapted from [56].

4.4 Wasserstein GAN

In [55] Arjovsky et al. introduced a novel objective function to train GANs, which is based on the Wasserstein distance. The Wasserstein-1 distance is a measure of how much energy is required to transform one probability distribution into another. An often used analogy is seeing both distributions as heaps of earth. The Wasserstein distance then represents the minimum energy required to move one heap into the other. Accordingly, the minimum energy corresponds to the product of the amount of earth that needs to be transported and the mean distance it has to be moved. Thus, the Wasserstein distance is also known as Earth-Mover(EM) distance for two distributions $\mathbb{P}_m$ and $\mathbb{P}_n$, which is formally defined as:

$$W(\mathbb{P}_m, \mathbb{P}_n) = \inf_{\gamma \in \prod(\mathbb{P}_m, \mathbb{P}_n)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||], \tag{4.3}$$

where $\prod(\mathbb{P}_m, \mathbb{P}_n)$ represents the set of all joint distributions $\gamma(x, y)$ with marginal distributions $\mathbb{P}_m, \mathbb{P}_n$. Since it is highly intractable to compute all joint distributions $\gamma(x, y)$ in order to find the infimum, the Katrovich-Rubenstein duality is used to rewrite 4.3 to its dual form:

$$W(\mathbb{P}_m, \mathbb{P}_n) = \sup_{||f||_L \leqslant 1} \mathbb{E}_{x\sim\mathbb{P}_m}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_n}[f(x)], \tag{4.4}$$

with the supremum over all 1-Lipschitz functions that map from data space X to a scalar, $f : X \to \mathbb{R}$. In practice, a parametrized function $f_w(\cdot)$ where $w \in \mathcal{W}$, such as a neural network, can be considered for use in equation (4.4) as long as the parameters w are lying in a compact space $\mathcal{W}$. This leads to the 1-Lipschitz requirement being replaced by a K-Lipschitz assumption and an approximation of the EM-distance up to a factor K. Furthermore, to return to the framework of GANs a distribution $\mathbb{P}_\theta$ learned by a generator G_θ is from now on considered in the formulation along with the true data distribution $\mathbb{P}_r$.

Therefore, instead of equation (4.4), the following problem can be considered:

$$K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta) = \max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim \mathbb{P}_Z}[f_w(G_\theta(z))] \tag{4.5}$$

To keep the parameters w in a compact space, the weights are simply bounded by a certain ϵ. Furthermore, $f(x)$ in the GAN framework represents the discriminator, which no longer acts as a classifier for real and fake samples but instead serves as a helper for estimating the Wasserstein metric between real and generated data distributions.

Updating the discriminator is implicitly performed when solving equation (4.5), while the generator updates can be obtained by computing the gradient with respect to the generator parameters of the approximated Wasserstein-distance, i.e:

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\nabla_\theta \mathbb{E}_{z \sim \mathbb{P}_Z}[f_w(G_\theta(z))] = -\mathbb{E}_{z \sim \mathbb{P}_Z}[\nabla_\theta f_w(G_\theta(z))], \tag{4.6}$$

where the second equality is proven in [55].The algorithm for training a GAN with the Wasserstein GAN (WGAN) objective function, is based on the standard algorithm proposed in [49] and can be seen in algorithm 2. Again, the discriminator and generator are updated consecutively, where the discriminator is trained for k steps before the generator is updated, in order to obtain a reasonable estimation of the EM-distance. The weights w of the discriminator are clipped after every update step, which should ensure that the K-Lipschitz requirement is fulfilled.

Algorithm 2: Training of a Wasserstein Generative Adverserial Network in mini-batch gradient descent mannor, as proposed in [55].

```
Require: $\alpha$ - learning rate; $\gamma$-RMSProp hyperparameters;
$\theta_0$ - initial $G$ parameter; $w_0$ - initial $D$ parameter; $m$ - batch size;
$k$ - amount of $D$ updates; $c$ - clipping parameter
begin
    while $\theta$ has not converged do
        for $k$ steps do
            Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ batch of training samples
            Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
            $w \longleftarrow \text{RMSProp}(\nabla_w \frac{1}{m} \sum_{i=1}^m [D_w(x^{(i)}) - D_w(G_\theta(z^{(i)}))], \alpha, \gamma)$
            $w \longleftarrow \text{clip}(w, -\epsilon, \epsilon)$

        Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
        $\theta \longleftarrow \text{RMSProp}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m D_w(G_\theta(z^{(i)})), \alpha, \gamma)$
```

Using the EM-distance as an objective function for training GANs has multiple advantages over using the standard formulation. Arjovsky et al. [49] showed that the Jenson-Shannon divergence (JSD), which is optimized in the classical GAN formulation, yields more reliable but very small gradients as the discriminator improves. This is due to the JSD being locally saturated if the discriminator is able to distinguish well between fake and real data. The discriminator of a GAN trained with WGAN objective, however, does not show this behavior and provides reasonable gradients for training the GAN, even if the discriminator is trained until optimality, as can be seen in figure 4.3. Furthermore, the problem of mode collapse is no more observed, as the generator can not overpower a well trained discriminator.

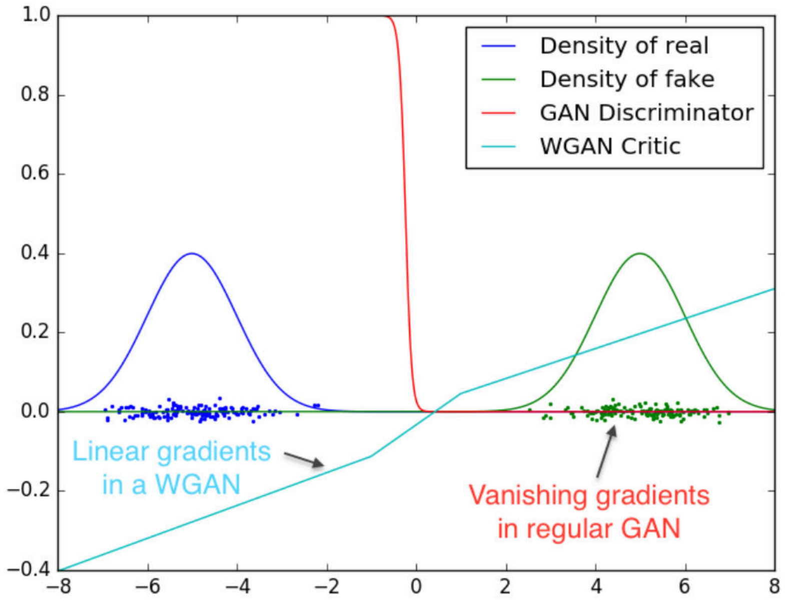

Figure 4.3: Gradients obtained for optimal discriminators in the standard and Wasserstein GAN setting. Adapted from [55].

4.4.1 Wasserstein GAN with gradient penalty

Even tough the usage of EM-distance for training GANs has improved training stability and sample quality, the authors of [57] showed that multiple issues arise due to parameter clipping, which is required to fulfill the K-Lipschitz criterion of the discriminator. In particular, it was found that the capacity of discriminators trained with weight clipping is reduced. As a result, the discriminators fail to capture higher order moments of the data distribution, due to the limited ability to model complex functions. Furthermore, it was observed that without tuning the clipping parameter ϵ, interactions between the cost function and the weight clipping lead to either vanishing or exploding gradients in the update steps. To overcome these issues, Gulrajani et al. [57] defined a novel objective function that penalizes the gradients of the discriminator in order to enforce the 1-Lipschitz constraint:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \underbrace{\mathbb{E}_{x\sim\mathbb{P}_r}[D_w(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[D_w(x)]}_{\text{WGAN loss}} + \underbrace{\lambda\, \mathbb{E}_{\tilde{x}\sim\mathbb{P}_r}[(\|\nabla_{\tilde{x}} D_w(\tilde{x})\|_2 - 1)^2]}_{\text{Gradient Penalty}}, \tag{4.7}$$

where λ describes the penalty coefficient and $\tilde{x}$ represents a linear interpolation between a generated sample and a sample from the dataset. By looking at the gradient penalty term in equation (4.7) it can be noticed that the penalty encourages the norm of the discriminator gradient to go towards 1, instead of staying below 1, which was empirically found to deliver slightly better results [57]. The algorithm for training a Wasserstein-GAN with gradient penalty can be found in 3.

Algorithm 3: Training of a Wasserstein GAN with gradient penalty in mini-batch gradient descent manner, as proposed in [57].

```
Require: $\lambda$ - penalty coefficient; $\alpha$ - learning rate; $\beta_1, \beta_2$ - Adam hyperparameters;
  $\theta_0$ - initial $G$ parameter; $w_0$ - initial $D$ parameter;
$m$ - batch size; $k$ - amount of $D$ updates

Define: $L(w, x, \tilde{x}, \hat{x}) = D_w(x) - D_w(\tilde{x}) + \lambda(\|\nabla_{\hat{x}} D_w(\hat{x})\| - 1)^2$

begin
    while $\theta$ has not converged do
        for $k$ steps do
            Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ batch of training samples
            Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
            $\tilde{x} \longleftarrow G_\theta(z)$
            $\hat{x} \longleftarrow \epsilon x + (1-\epsilon)\tilde{x}$
            $w \longleftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L(w, x^{(i)}, \tilde{x}^{(i)}, \hat{x}^{(i)}), \alpha, \beta_1, \beta_2)$

        Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
        $\theta \longleftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m D_w(G_\theta(z^{(i)})), \theta, \alpha, \beta_1, \beta_2$
```

4.5 Information Maximization GAN

In the usual GAN setting, latent vectors z are sampled from a multivariate standard normal distribution and handed to the generator that produces samples in dependence on z. As no restrictions are set on how the generator processes the noise, it is very likely that individual dimensions of z do not correspond to semantic features, which is referred to as an entangled representation. In the same context, a disentangled representation would mean that specific elements of the latent vector z correspond to specific features in the data domain. An example for disentangled representation regarding the MNIST dataset (handwritten digits) would be that specific parts of the latent vector z incorporate the numerical identity of the digit, the font size or stroke thickness. An illustration of the concept can be seen in figure 4.4.

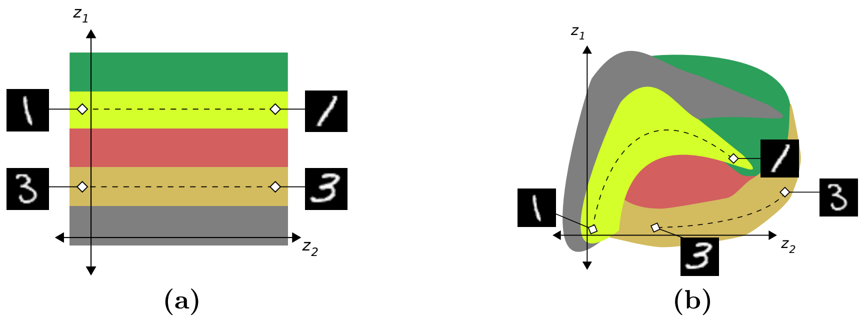

Figure 4.4: Disentangled representation. In the disentangled representation (a), digit classes change with z_1 while the rotation of the digit is encoded in z_2. The entangled representation (b) does not show such behavior, as the rotation and the digit class are not explicitly encoded in a certain element of z.

Chen et al. [58], describe an information theoretic GAN-based approach that is able to learn disentangled representations in a completely unsupervised manner. In the proposed technique, the latent vector is split into an incompressible noise vector z and the latent code c. An extension to the GAN objective maximizes the mutual information between the generated data samples and the latent code c in order to observe disentangled representations. Samples produced by the generator can now be observed by: $x = G(z, c)$. In the standard GAN framework, the latent code would simply be used in the same way as the rest of the latent vector, i.e. $\mathbb{P}_\theta(x|c) = \mathbb{P}_\theta(x)$. To increase the mutual information of the generator distribution and the distribution of the latent vector, an information theoretic entropy regularization was introduced. Therefore, the objective function 4.1 changes to:

$$\min_G \max_D \mathbb{E}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathbb{E}_{x\sim\mathbb{P}_\theta}[\log(1 - D(x)] - I(c; G(z, c)), \tag{4.8}$$

where the additional term $I(c; G(z, c))$ refers to the mutual information that can be intuitively interpreted as the reduction of uncertainty in $G(z, c)$ when c is given. In practice, however, it is hard to maximize $I(c; G(z, c))$ as it requires the unknown posterior $\mathbb{P}(c|x)$. Still, the mutual information can be maximized by incorporating a function $Q(\cdot)$ that estimates the latent code for a generated sample $x = G(c, z)$. In order to maximize $I(c; G(z, c))$, an additional objective function $L(G, Q, c)$ is introduced which measures the difference between the given latent code c and the prediction $\tilde{c} = Q(G(c, z))$ as well as encourages the generator to produce disentangeled samples that incorporate information about c. In the practical implementation of InfoGAN, the function Q is parametrized by a neural network that shares most of the layers with the discriminator. The latent code c

can be further split into a categorial latent code c_{cat}, as well as a continuous latent code c_{cont}. The aim of this approach is to use categorial latent code to represent categorial features, for instance class membership, and continuous latent code c_{cont} to represent continuous features, such as rotation. An example for a latent vector, as used in the InfoGAN framework, can be seen in figure 4.5.

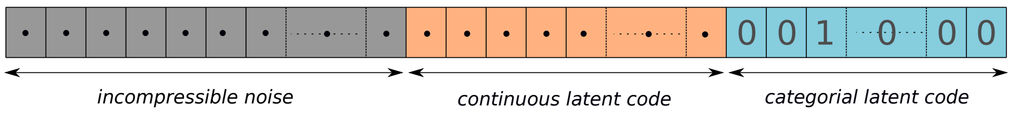

Figure 4.5: Latent vector as observed in InfoGAN framework. The incompressible noise is usually sampled from a standard normal distribution $z_i \sim \mathcal{N}(0, 1)$, the continuous part is sampled from a uniform distribution $c_{con,i} \sim U(-1, 1)$ and the categorial part is sampled from a categorial distribution $c_{cat} \sim Cat(K = 10, p = 0.1)$.

Considering the categorial part of the latent code, the objective function defined by the variational lower bound is simply a cross-entropy loss. For the continuous part the mean-squared-error can be used, as suggested in [60]. This leads to:

$$L(G, Q, c) = \frac{\lambda_{con}}{2} \|Q_{w,con}(x) - c_{con}\|^2 - \lambda_{cat} \sum_{j=1}^{n} c_{cat,j} \log(Q_{w,cat}(x)_j), \tag{4.9}$$

4.6 Inverse Generative Adverserial Networks

As mentioned in section 4.2 GANs are capable of learning the transformation from a lower dimensional latent distribution $\mathbb{P}_z$ to a complex high dimensional data distribution $\mathbb{P}_r$. The mapping is performed by the generator part of the GAN and can be expressed as:

$$G : \mathbb{R}^d \rightarrow \mathbb{R}^{|x|}, \tag{4.10}$$

where $z \in \mathbb{R}^d$ is a sample from the d-dimensional latent distribution, $x \in \mathbb{R}^{|x|}$ is a generated data sample. Going the inverse way, i.e. mapping from sample to latent space is not trivial, as the generator is a highly non-linear model and can therefore not simply be inverted. Nevertheless, the solution can be approximated by means of an optimization problem:

$$\min_{z' \in \mathbb{R}^d} \frac{1}{2} \|G_\theta(z) - G_\theta(z')\|^2, \tag{4.11}$$

which has a value of 0 at the global minimum, since the sample is produced by the generator [61].

This optimization problem can simply be solved by gradient descent based algorithms, since the generator is represented by a differentiable function. In addition, as the distribution of z is known, regularization can be added to the objective function, which penalizes latent vectors that have statistics that are inconsistent with the underlying distribution $\mathbb{P}_z$ [62]. This leads to:

$$\min_{z' \in \mathbb{R}^d} \frac{1}{2} \|G_\theta(z) - G_\theta(z')\|^2 + \mathcal{R}(z), \tag{4.12}$$

where $\mathcal{R}(\cdot)$ describes the regularization of z, e.g. the ℓ^2-norm if the underlying distribution of z is Gaussian. Under the assumption that the data distribution $\mathbb{P}_r$, is well approximated by generator distribution $\mathbb{P}_\theta$, the same formulation as in 4.12 can be used to obtain a latent vector that maps to a specific sample x from the data distribution:

$$\min_{z' \in \mathbb{R}^d} \frac{1}{2} \|x - G_\theta(z')\|^2 + \mathcal{R}(z) \tag{4.13}$$

Due to $\mathbb{P}_\theta$ being an approximation of $\mathbb{P}_r$, the sample generated by z' is only as close to x, as the most similar sample that the generator is able to yield, which means that the value of the objective function (4.13) in the global minimum cannot be obtained at 0.

5 Related Work

Contents

In this chapter, related work as well as current state-of-the-art for solving inverse problems with GANs are discussed briefly. For in-depth insight, reading the given references is highly recommended. Considering GANs in the framework of inverse problems in imaging, two opposed approaches with the aim of reconstruction can be found:

- Direct modeling of the inverse measurement process $\mathcal{F}^{-1}$ in the generator path of the GAN
- Iteratively solving the inverse problem by using the generator of a GAN as an implicit regularizer

In this chapter the first method will be called the direct approach and the second method will be referred to as the indirect approach.

Direct Approach

The aim of the direct approach is to solve a specific inverse problem by directly approximating an inverse of its measurement process $\mathcal{F}$. In this method, the generator takes a corrupted image y and is trained to directly yield the reconstruction x, i.e. a set of corrupted data as well as the corresponding set of uncorrupted data has to be available.

The reconstruction can be formulated as:

$$x = G(y) \approx \mathcal{F}^{\dagger}(y), \tag{5.1}$$

with $G : \mathbb{R}^{|x|} \rightarrow \mathbb{R}^{|x|}$, where $|\cdot|$ denotes the dimensionality of x and $\dagger$ denoting that the inverse usually does not exist. This can be seen as a mapping from a distribution of corrupted images to a distribution of uncorrupted images. Therefore, the type of problem addressed with this approach needs to be known in advance. Furthermore, the trained model is only capable of solving the specific inverse problem it was trained for.

Indirect Approach

Assuming a trained generator, the goal of the indirect approach is to find the latent vector z that maps to the uncorrupted source image closest to the corrupted image when $F(\cdot)$ is applied, i.e.:

$$\min_{z \in \mathbb{R}^d} \frac{1}{2} \|\mathcal{F}(G_\theta(z)) - y\|^2 + \mathcal{R}(z), \tag{5.2}$$

with $G : \mathbb{R}^d \rightarrow \mathbb{R}^{|x|}$. This method has the advantage that multiple inverse problems can be addressed by learning a single model, which acts as a strong prior restricting the solution to the domain of source images. Furthermore, there is no need for a set of corrupted data, as the model is solely trained on uncorrupted samples. A detailed description of the indirect approach can be found in section 6.

5.1 Methods using the direct approach

In [63], Izuka et al. proposed a GAN-based approach for image inpainting. In their work, the authors introduce a deep generative model-based method for image inpainting that produces locally and globally consistent images. The approach uses both local and global context discriminators, where the global discriminator evaluates if the entire image is coherent, while a local context discriminator ensures the local consistency of generated patches by looking at small areas centered at the completed region. In addition to the GAN loss, they use a weighted MSE loss considering the inpainting mask is used, in order to stabilize the training process. Yu et al. [64] further improved this method by additional usage of attention modeling and the WGAN loss with gradient penalty.

An image super resolution technique based on the direct approach was presented by Ledig et al. in [53]. In their work, they introduce a framework capable of inferring photo-realistic natural images for 4 times upscaling factors. The proposed GAN-based architecture (SRGAN) consists of multiple ResNet [65] blocks and is optimized by using a perceptual loss function, which is composed of an adversarial loss and a content loss. The adversarial loss encourages the model to produce images that reside in the space of natural images, whereas the content loss ensures similarity to the target image and is either used as the pixel-wise MSE loss or the VGG loss that describes the euclidean distance between hidden-layer activations of the target and the reconstructed image.

Kupyn et al. [66] proposed an end-to-end learning method for motion deblurring using the direct approach. Their aim is to recover a sharp image from a blurred version without any knowledge of the underlying blur kernel, which is referred to as blind image deblurring. Similarly to [53], the authors use a combination of adverserial loss and VGG-based content loss to train the generator. The introduced network architecture is based upon stacking multiple ResNet blocks with additional transposed convolution blocks.

GAN-based compressed sensing MRI reconstruction is addressed by Yang et al. in [67]. Their proposed DAGAN architecture consists of a U-Net [68] based generator with skip connections, which is trained in an end-to-end fashion to reduce aliasing artifacts. A novel objective function for training is presented, which is composed of a GAN-loss, a VGG loss, an MSE loss in image domain as well as an MSE loss in frequency domain. DAGAN achieves remarkable results in CS-MRI reconstruction, outperforming conventional reconstruction approaches and achieving results comparable to the state-of-the-art.

5.2 Methods using the indirect approach

The work of Bora et al. [69] deals with the general problem of compressed sensing, i.e. how to estimate the most likely solution of an under-determined system of linear equations $y = Ax$. The authors present an algorithm that uses generative models for compressed sensing, which simply uses gradient descent to optimize the latent vector z in order to obtain a solution $G(z)$ with a small measurement error $\|AG(z) - y\|^2$. ℓ_2-regularization on z is added to the objective function in order to encourage the latent vector to stay in regions of the Gaussian distribution from which z was sampled during training. This can be done due to the assumption that the generator part of the network learns a mapping from a low dimensional distribution to a high dimensional data distribution. Experiments in the form of a simple reconstruction (A is set to identity) and super-resolution have been carried out to validate the proposed assumptions. The authors state that as long as a good approximate solution to the objective can be found, the introduced method tends to yield good results, especially outperforming traditional methods such as LASSO [70] for a small amount of measurements y.

Tripathi et al. [71] utilize GANs in the indirect approach to reduce noise in fake images generated by the GAN as well as in real images. The proposed method is similar to that in [69], with the difference that only a single measurement y is observed. The authors demonstrate that their method is capable of denoising images generated by the trained GAN, obtaining significantly better results than with BM3D [72]. Furthermore, a consistent bias in latent vectors that represents the blur in reconstructions of noisy images was found. By subtracting this bias from the estimated latent vector, improved denoising results could be obtained. During the training process the latent vectors z have been sampled uniformly from the $[\text{-}1, 1]^{100}$ hypercube. The optimization of z was motivated by the work of [61], which suggest the use of stochastic clipping that replaces values larger than $[\text{-}1, 1]$ with random values in the allowed range. This leads to better results than simple clipping or not constraining z at all.

In [73], Yeh et al. address the task of semantic image inpainting with GANs used in the indirect approach. Their proposed method is able to perform photo realistic reconstruction of large missing regions in images. Instead of using the ℓ_2-norm as objective function to retrieve the optimal z, the authors found that the ℓ_1-norm performs slightly better. Additionally, the so called prior loss is added to the objective function, which is identical to the GAN loss for training the discriminator. Thus, it penalizes unrealistic images. Furthermore, the authors found that the reconstructed areas are not neccessarily in the same intensity range as the surrounding pixel. Hence, they further refine the solution by the use of Poisson blending [74].

6 Methods

Contents

This chapter describes the methods used to solve the inverse problems addressed in this thesis. Moreover, a novel approach, called auxiliary optimization, is presented, which allows to further improve the obtained reconstruction results. Additionally, this chapter describes the different datasets used to evaluate the performance of the proposed techniques.

6.1 Solving Inverse Problems by Latent Vector Recovery

As mentioned in chapter 5, two ways of solving inverse problems with GANs may be considered. Since the aim of this thesis is to address multiple inverse problems, the indirect approach is favored, and explained in the following.

As the deep generative model is trained in an unsupervised manner and solely on uncorrupted data, it oughts to well approximate the underlying data distribution $\mathbb{P}_r$. Therefore, under the assumption of an ideal approximation, all samples in the range of the generator are assumed to come from $\mathbb{P}_r$, which can be seen as an implicit prior on the solution of the inverse problem. This characteristic of the generator can be used to restrict the solution space of an inverse problem, similar to regularization.

Under the assumption that the measurement operator $\mathcal{F}$ is known, an inverse problem can be solved by optimizing a latent vector $z \in \mathbb{R}^d$ such that the observed image undergoing $\mathcal{F}$ represents a corrupted image $y \in \mathbb{R}^{M \times N \times C}$, which can be expressed as the following optimization problem:

$$z^* = \arg\min_z \|\mathcal{F}(G_\theta(z)) - y\|^2 + \lambda \|z\|^2 \tag{6.1}$$

The reconstructed image $x_{rec} = \mathbb{R}^{M \times N \times C}$ can be observed via inference of the generator, given the latent vector found in equation (6.1), i.e.

$$x_{rec} = G(z^*) \tag{6.2}$$

The InfoGAN approach is of particular interest in latent vector recovery, due to the disentangled representations in the latent code c, since the resulting objective function $\|G(c, z) - y\|^2$ has a disentangled landscape. This is beneficial for the optimization process and leads to better solutions for the inverse problems. The iterative algorithm used to solve the inverse problems can be seen in 4. Having a pre-trained generator with parameters θ, the optimal solution can be obtained by iteratively updating z using any variant of gradient descent until convergence. One update step corresponds to the gradient of (6.1) with respect to z, scaled with step size α.

Algorithm 4: Solving an inverse problem with known measurement function $\mathcal{F}(\cdot)$, by optimization of the latent vector.

```
Require: $\alpha$ - step size; $\theta$ - pre-trained $G$ parameter;
$\mathcal{F}(\cdot)$ - measurement function; $y$ - corrupted sample
Define: $E(\theta, z, y) = \|\mathcal{F}(G_\theta(z)) - y\|^2 + \lambda \|z\|^2$
begin
    Initialize: $z \longleftarrow \mathbf{0}$
    while not converged do
        $z^{k+1} \longleftarrow z^k - \alpha \cdot \nabla_z \, E(\theta, z^k, y)$
        $k \longleftarrow k + 1$
    $x_{rec} \longleftarrow G_\theta(z^k)$
```

In this approach, it is assumed that the generator is an exact model of the underlying data distribution. However, this theoretic assumption is not correct in practice. Therefore, often only insufficient reconstructions (even of uncorrupted images) are obtained after optimizing equation (6.1) to convergence, as can be seen in figure 7.1. The root cause for this behavior is unclear but might be found in the generator, which apparently is not able to sample from the entire data distribution. In other words, the generator learns an insufficient approximation of the true data distribution, which does not contain the desired data samples. A representative graphical interpretation is given in figure 6.1 that illustrates the mapping from the Gaussian hypersphere to the approximated data space.

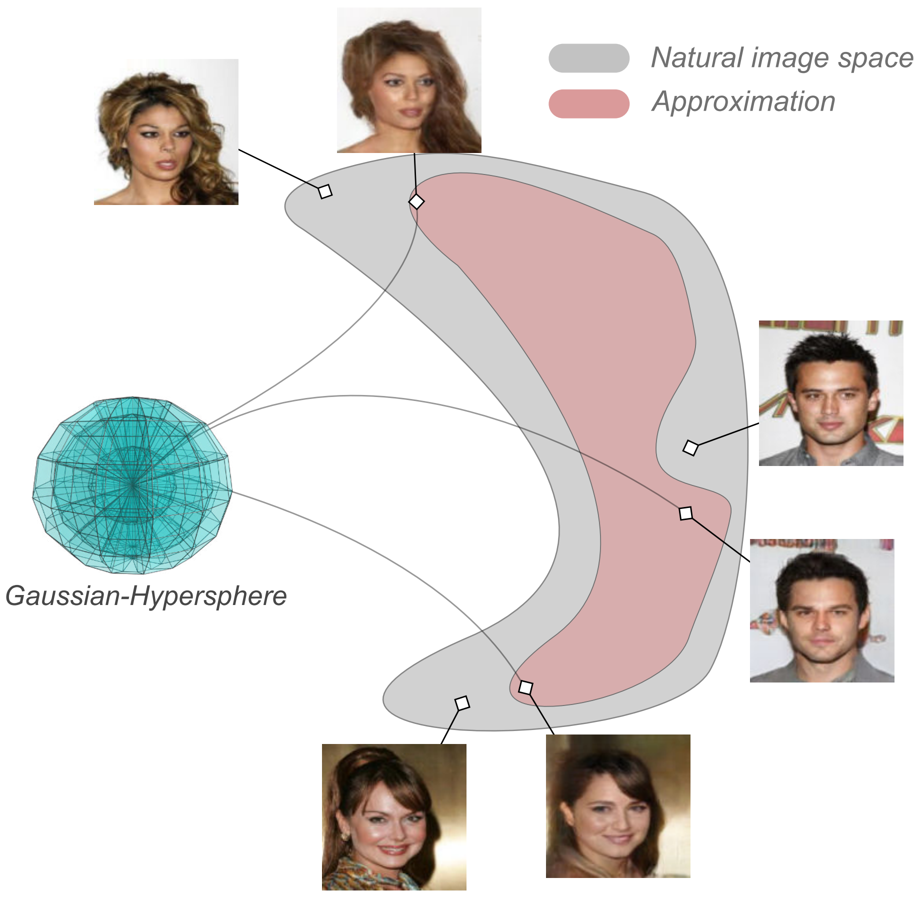

Figure 6.1: Illustration of original samples and generated samples on image space.

6.2 Auxiliary Optimization

To overcome the issue of insufficient approximation, an approach called auxiliary optimization is proposed in this work, which is capable of further refining the learned approximation with respect to specific samples. The suggested algorithm performs additional gradient update steps on the latent vector z as well as the generator parameter θ, which enforces the generator to push its range towards the data sample that should be reconstructed.

An illustration of the concept is given in figure 6.2. In fact, more than one data sample can be considered at once, which leads to an improved generator that is able to well represent multiple unseen data samples. At first glance, this property might seem unnecessary but it enables vector arithmetic operations on several unseen images.

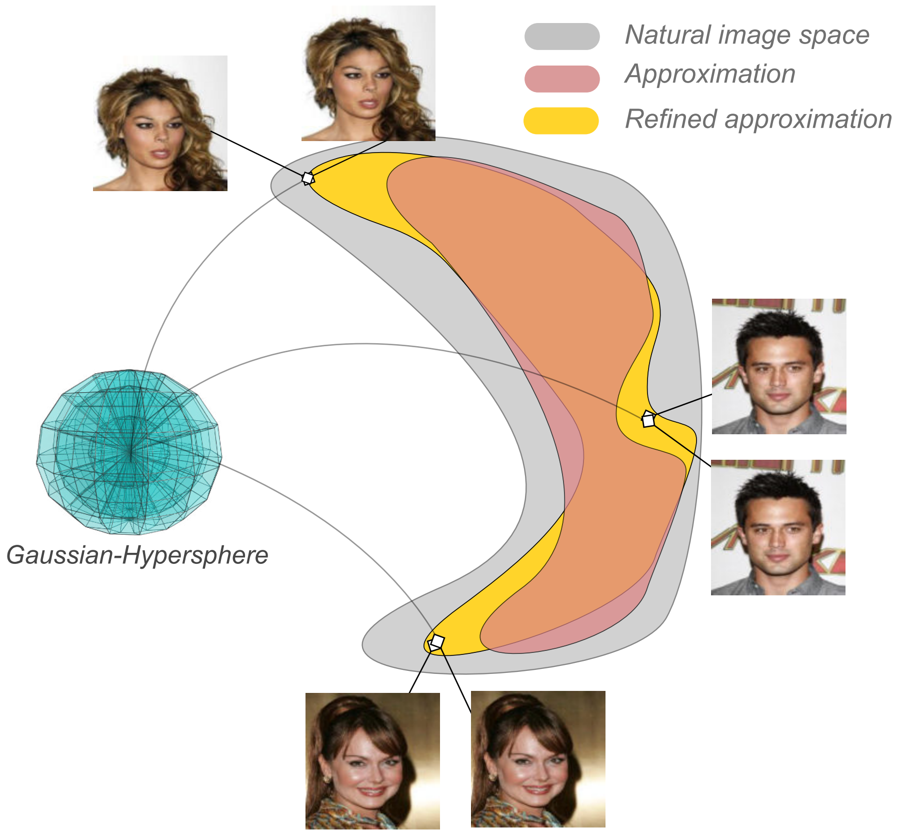

Figure 6.2: Illustration of original samples and generated samples on refined image space.

As stated in algorithm 5, auxiliary optimization is only performed after the optimization for z has converged. This is mandatory, as the implicit structure of the generator should be changed as little as possible, which is only the case if the image resulting from latent vector recovery is already close to the desired sample. Applying auxiliary optimization before the optimization for z has converged would cause the implicit prior to be destroyed, which makes the generator not usable for inverse problems any more. This is the case because learning a transformation from an arbitrary image that is not similar to the desired sample would cause the parameters θ of the generator to be substantially modified. Furthermore, a regularizer that penalizes large parameter changes during the auxiliary optimization can be utilized to keep the change in the generator structure small. The resulting objective for auxiliary optimization can be expressed as:

$$(z^*, \theta^*) = \arg\min_{z,\theta} \|\mathcal{F}(G_\theta(z)) - y\|^2 + \lambda\|z\|^2 + \beta\|\theta^0 - \theta\|^2, \tag{6.3}$$

with parameters λ and β that weight the regularization on z and the change of generator parameters θ with respect to the initial parameters θ^0. For the sake of simplicity, the ℓ^2-norm was employed to penalize large parameter changes. However, also different regularizers such as the ℓ^1-norm could be considered.

Algorithm 5 describes the proposed auxiliary optimization approach. The initial procedure for recovering the latent vectors z as well as the auxiliary optimization can be performed with any variant of gradient descent. Since the parameters of the generator are also optimized, auxiliary optimization is performed using Adam. After convergence of the objective, the reconstructed image x_{rec} can be observed via inference of the refined generator, given the optimized noise vector z^*:

$$x_{rec} = G_{\theta^*}(z^*) \tag{6.4}$$

Algorithm 5: Solving an inverse problem with known measurement function $\mathcal{F}(\cdot)$, by optimization of the latent vector, with auxiliary optimization.

```
Require: $\alpha$ - step size for latent vector recovery; $\theta^0$ - pre-trained $G$ parameter;
$\sigma$ - step size for auxiliary optimization; $\mathcal{F}(\cdot)$ - measurement function;
$y$ - corrupted sample;
Define: $E(\theta, z, y) = \|\mathcal{F}(G_\theta(z)) - y\|^2 + \lambda\|z\|^2 + \beta\|\theta^0 - \theta\|^2$
begin
    Initialize: $z \longleftarrow \mathbf{0}$
    while not converged do
        $z^{k+1} \longleftarrow z^k - \alpha \cdot \nabla_z \, E(\theta^0, z^k, y)$
        $k \longleftarrow k + 1$
    $\hat{z}^0 \longleftarrow z^{k+1}$
    while not converged do
        $\hat{z}^{n+1} \longleftarrow \hat{z}^n - \sigma \cdot \nabla_z \, E(\theta^n, \hat{z}^n, y)$
        $\theta^{n+1} \longleftarrow \theta^n - \sigma \cdot \nabla_\theta \, E(\theta^n, \hat{z}^n, y)$
        $n \longleftarrow n + 1$
    $x_{rec} \longleftarrow G_{\theta^n}(\hat{z}^n)$
```

The procedure of latent vector recovery along with auxiliary optimization for an inpainting problem can be seen in figure 6.3. At first, solely z is optimized until convergence, which already yields an image close to the original without any corruptions. After convergence at step 1520, the auxiliary optimization starts and pushes the solution even closer to the desired image. Immediately after starting this additional optimization procedure, the former converged loss drops until also the auxiliary optimization has converged. Since the parameters of the generator undergo only slight changes, the prior on the data distribution is preserved, which leads to a naturally looking estimate of the unknown regions.

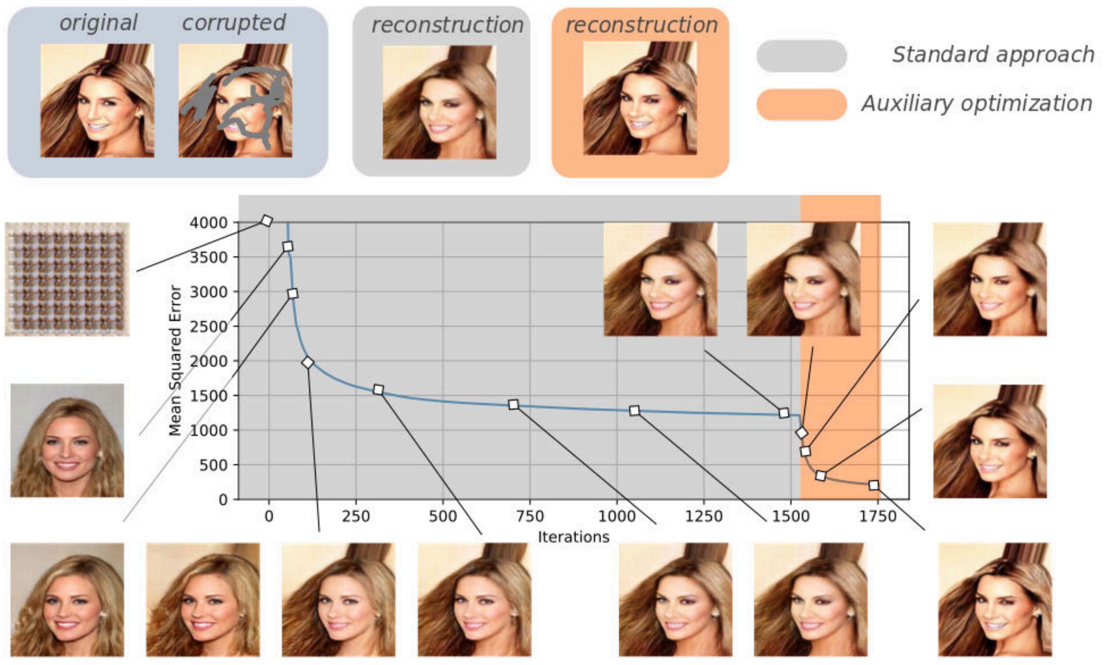

Figure 6.3: Reconstruction of a corrupted image using latent vector recovery with auxiliary optimization.

6.3 Mathematical formulation of Inverse Problems with GANs

The following sections contain the mathematical formulations for solving the inverse problems addressed in this thesis with auxiliary optimization. A detailed description of the inverse problems can be found in chapter 2.

6.3.1 Denoising

Following the assumption that the generator is only capable of generating noise free images, the following expression can be used to obtain a valid solution:

$$\min_{z \in \mathbb{R}^d} \|G_\theta(z) - y\|^2 + \lambda \|z\|^2 + \beta \|\theta^0 - \theta\|^2, \tag{6.5}$$

When using auxiliary optimization in denoising, it is important to note that the optimization of equation (6.5) must not be performed until convergence except with sufficient regularization on θ, as the reconstruction would otherwise overfit the noisy image.

6.3.2 Deblurring

As the blur kernel is assumed to be known for the experiments in this thesis, the following optimization problem can be used for deblurring:

$$\min_{z \in \mathbb{R}^d} \|h * G_\theta(z) - y\|^2 + \lambda \|z\|^2 + \beta \|\theta^0 - \theta\|^2, \tag{6.6}$$

where h is a blur kernel convolved with the generated image.

6.3.3 Super-Resolution

The generator in the framework of DCGANs cannot be used for arbitrary upsampling since is only able to generate images of a distinct resolution. Still, if an image with lower resolution is obtained, the generator of a GAN, trained on high resolution images, can be utilized to recover a detailed high resolution version of the sample.

The optimization problem used in super-resolution can be obtained as:

$$\min_{z \in \mathbb{R}^d} \|DB(G_\theta(z)) - y\|^2 + \lambda \|z\|^2 + \beta \|\theta^0 - \theta\|^2, \tag{6.7}$$

where DB is a down-sample blur operator that is applied to the generated sample. The resulting image has the same resolution as y.

6.3.4 Inpainting

Due to the strong implicit prior of a pre-trained generator, estimations of unknown image regions intuitively seem to be an inverse problem, which can be solved with the help of GANs. Since corrupted regions are not considered in the loss function, the latent vector corresponding to the source image can still be found if image regions are missing. Therefore, it is possible to accurately reconstruct the missing areas.

The inpainting problem can be formulated as:

$$\min_{z \in \mathbb{R}^d} \|M \odot G_\theta(z) - y\|^2 + \lambda \|z\|^2 + \beta \|\theta^0 - \theta\|^2, \tag{6.8}$$

where M represents the mask of image regions that are corrupted and are to be ignored in the optimization process. The reconstructed image consists of regions known in advance, as well as regions estimated in the optimization process.

6.3.5 MRI reconstruction

In this task, the difference between the reconstruction and the corrupted image is computed in Fourier space, as opposed to the already discussed inverse problems. However, the generator can still be used for image reconstruction since the prior constraining the reconstruction to the domain of source images still exists in Fourier domain.

Due to the spectral undersampling in accelerated MRI, the problem can be seen as an inpainting task in Fourier domain, which can be formulated as follows:

$$\min_{z \in \mathbb{R}^d} \| S \odot \mathscr{F}(G_\theta(z)) - y \|^2 + \lambda \|z\|^2 + \beta \|\theta^0 - \theta\|^2, \tag{6.9}$$

where $\mathscr{F}$ is the 2D-Fourier transform and S represents a matrix that blanks unsampled regions.

6.4 Training of Generative Adverserial Networks

In order to obtain generators with strong implicit priors constraining the reconstruction to the domain of source images, several GANs have been trained in advance. A detailed description of the architectures and the implementation are provided in the following sections.

6.4.1 Network Architectures

The GAN architectures used for approaching the inverse problems in this thesis are either based on the DCGAN architecture proposed in [56] or the DCGAN architecture with an extension to InfoGAN as proposed in [58]. As a minor change, 4×4 convolution kernels are used instead of 3×3 kernels, in order to increase the amount of pixels that contain information for the transposed convolution. An illustration of the generator used for the CelebaA dataset can be seen in figure 6.4. It shows building blocks that are necessary for the transformation of a latent vector $z \in \mathbb{R}^d$ into an image. A detailed overview of the generator architecture used for CelebA and MRT datasets is given in table 6.1.

Figure 6.5 shows the discriminator architecture along with the optional extension to InfoGAN. The discriminator takes an image and yields a scalar that helps to compute the Wasserstein distance. The InfoGAN extension network takes the flattened output from the last convolution layer of the discriminator and delivers an estimation of the categorical latent codes $c_{cat} \in \mathbb{R}^p$ as well as for the the continuous latent codes $c_{con} \in \mathbb{R}^q$. An overview of the GAN architecture used for the MNIST dataset can be found in the appendix in tables B.1 and B.2.

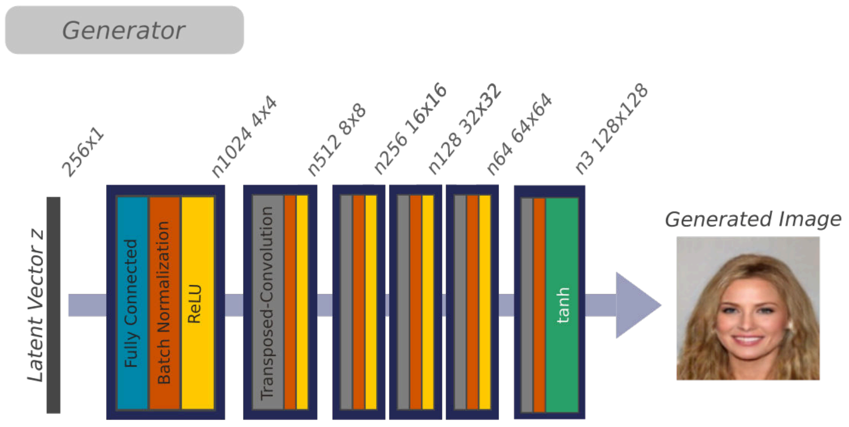

Figure 6.4: Architecture scheme of Generator used for CelebA Dataset.

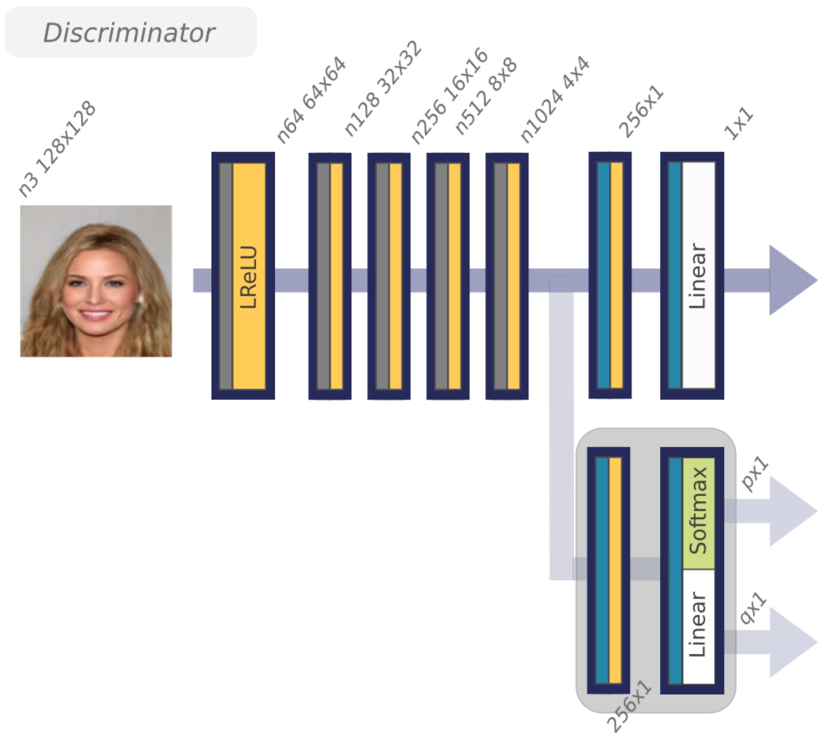

Figure 6.5: Architecture scheme of Discriminator used for CelebA Dataset.

Generator architecture used for CelebA/MRT dataset

Optimizer: $Adam(lr = 1e\text{-}4, \beta_1 = 0.5, \beta_1 = 0.9)$, **Batch size**: 128; **Parameter initialization**: $\mathcal{N}(\mu = 0, \sigma = 0.05)$

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
Type	Fully Connected	Transposed Convolution	Transposed Convolution	Transposed Convolution	Transposed Convolution	Transposed Convolution
Input Dimension	[1 × 256]	[4 × 4 × 1024]	[8 × 8 × 512]	[16 × 16 × 256]	[32 × 32 × 128]	[64 × 64 × 64]
Output Dimension	[4 × 4 × 1024]	[8 × 8 × 512]	[16 × 16 × 256]	[32 × 32 × 128]	[64 × 64 × 64]	[3 × 3 × 128]
Filter size	-	[4 × 4]	[4 × 4]	[4 × 4]	[4 × 4]	[4 × 4]
Stride	-	2	2	2	2	2
Padding	-	Same	Same	Same	Same	Same
Activation	ReLU	ReLU	ReLU	ReLU	ReLU	tanh
Batch Normalization	True	True	True	True	True	False
L2 Regularization	2.5*e*-5	2.5*e*-5	2.5*e*-5	2.5*e*-5	2.5*e*-5	2.5*e*-5

Table 6.1: Detailed architecture of generator used for CelebA/MRT dataset.

Discriminator architecture used for CelebA/MRT dataset

Optimizer: $Adam(lr = 1e\text{-}4, \beta_1 = 0.5, \beta_1 = 0.9)$, **Batch size**: 128; **Parameter initialization**: $\mathcal{N}(\mu = 0, \sigma = 0.05)$

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Type	Convolution	Convolution	Convolution	Convolution	Convolution	Fully Connected	Fully Connected
Input Dimension	[128 × 128× 3]	[64 × 64 × 64]	[32 × 32 × 128]	[16 × 16 × 256]	[8 × 8 × 512]	[16384 × 1]	[256 × 1]
Output Dimension	[64 × 64 × 64]	[32 × 32 × 128]	[16 × 16 × 256]	[8 × 8 × 512]	[4 × 4 × 1024]	[256 × 1]	[1 × 1]
Filter size	[4 × 4]	[4 × 4]	[4 × 4]	[4 × 4]	[4 × 4]	-	-
Stride	2	2	2	2	2	-	-
Padding	Same	Same	Same	Same	Same	-	-
Activation	LReLU	LReLU	LReLU	LReLU	LReLU	LReLU	Linear
Batch Normalization	False	False	False	False	False	False	False
L2 Regularization	-	-	-	-	-	-	-

Extension for discriminator to InfoGAN

Extension: The output of layer 5 is given to fully connected layers that map the features to the correct dimensionality needed for InfoGAN. The layers $\tilde{6}$, $\tilde{7}$ appear alongside the standard layers 6,7.

	Layer 1 - Layer 5	Layer $\tilde{6}$	Layer $\tilde{7}$
Type	Convolution	Fully Connected	Fully Connected
Input Dimension	[128 × 128× 3]	[16384 × 1]	[256 × $(p + q)$]
Output Dimension	[4 × 4 × 1024]	[256 × 1]	[$(p + q)$ × 1]
Activation	LReLU	LReLU	Softmax for p Linear for q
Batch Normalization	False	False	False
L2 Regularization	-	2.5e-5	2.5e-5

Table 6.2: Detailed architecture of discriminator used for CelebA/MRT dataset.

6.4.2 Training Procedure

Both GAN architectures (standard and InfoGAN extended), which are described in 6.4.1 have been trained as Wasserstein-GAN with gradient penalty (WGAN-GP), as it has significant advantages over the standard GAN-loss, see chapter 4.

The InfoGAN approach has been employed in this work under the hypothesis that latent vector recovery benefits from the disentangled representation enforced by the InfoGAN objective. The detailed training procedure is stated in algorithm 6.

Algorithm 6: Training of a Wasserstein GAN with gradient penalty and InfoGAN extension in mini-batch gradient descent manner.

```
Require: $\lambda$ - penalty coefficient; $\alpha$ - learning rate;
$\beta_1, \beta_2$ - Adam hyperparameter; $\theta_0$ - initial $G$ parameter;
$w_0$ - initial $D$ parameter with subest $w_{Q_0} \subset w_0$ - initial $Q$ parameter;
$m$ - batch size; $k$ - amount of $D$ updates; N - overall training steps;
$\lambda_{con}, \lambda_{cat}$ - InfoGAN hyperparameter

Define:
    $L_Q(w_Q, \tilde{x}, c) = \lambda_{con} \| D_{w,con}(\tilde{x}) - c_{con} \|^2 - \lambda_{cat} \sum_j c_{cat,j} \log(D_{w,cat}(\tilde{x})_j)$
    $L(w, x, \tilde{x}, \hat{x}, c) = D_w(x) - D_w(\tilde{x}) + \lambda(\| \nabla_{\hat{x}} D_w(\hat{x}) \| - 1)^2 + L_Q(w_Q, \tilde{x}, c)$

begin
    for $N$ steps do
        for $k$ steps do
            Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ batch of training samples
            Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
            Sample $\{c^{(i)}\}_{i=1}^m \sim \mathbb{P}_C$ batch of latent codes
            $\tilde{x} \longleftarrow G_\theta(z)$
            $\hat{x} \longleftarrow \epsilon x + (1 - \epsilon)\tilde{x}$
            $w \longleftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L(w, x^{(i)}, \tilde{x}^{(i)}, \hat{x}^{(i)}, c^{(i)}), \alpha, \beta_1, \beta_2)$

        Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}_Z$ batch of priors
        Sample $\{c^{(i)}\}_{i=1}^m \sim \mathbb{P}_C$ batch of latent codes
        $\tilde{x} \longleftarrow G_\theta(z)$
        $w_Q \longleftarrow \text{Adam}(\nabla_{w_Q} \frac{1}{m} \sum_{i=1}^m L_Q(w_Q, \tilde{x}^{(i)}, c^{(i)}), \alpha, \beta_1, \beta_2)$
        $\theta \longleftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m D_w(G_\theta(z^{(i)})), \alpha, \beta_1, \beta_2)$
```

6.4.3 Implementation Details

Hyper Parameter - Optimization of the generator as well as the discriminator have been performed by Adam with $\beta 1 = 0.9$, $\beta 1 = 0.999$ and a learning rate α of $1 \cdot 10^{-4}$. Network parameters for the discriminator and the generator have been initialized Gaussian randomly with a standard deviation of $\sigma = 0.02$. The overall amount of training steps N was set to 100.000 in all experiments, with the discriminator update factor k being set to 5. Contrary to [57], the weighting parameter for the gradient penalty term has been set to 100 instead of 10, which resulted in a slightly higher image quality.

Info GAN - When training was performed using the InfoGAN approach, the categorial parameter λ_{cat} was set to 2.0 while the continuous parameter λ_{con} was set to 0.01, where the parameter values have been chosen in a way that the additional losses are in the same range as the WGAN-GP objective. The categorial part of the latent codes $c_{cat} \in \{0,1\}^{10}$ was drawn from a uniform categorial distribution containing 10 categories. The continuous part of the latent codes c_{con} was drawn from a multivariate uniform distribution, where $c_{con} \in \mathbb{R}^{54}$ for MNIST dataset and $c_{con} \in \mathbb{R}^{118}$ for MRI and CelebA datasets.

Data Feeding - Since both the MRI and the CelebA datasets contain images in the range of several gigabyte, it is not possible to load all data simultaneously from the hard disk into the RAM. Therefore, a data feeding algorithm was implemented that uses multiprocessing to load and pre-process data batches in parallel, which significantly speeds up training.

Pre-processing - In order to stay in the valid range of the *tanh* nonlinearity, samples from the datasets had to be rescaled to the range of [-1, 1]. For the CelebA dataset, images have been randomly mirrored in order to observe a larger variety in the relevant information. No mirroring was performed for the MNIST and MRI datasets, since the underlying conditions that lead to the observed data are not symmetric, e.g. neuroanatomical differences between the left and right hemisphere of the brain or the digit 7, which is never written in a mirrored fashion.

Software - The implementation of the networks was done in python version 3.5, using the deep learning framework tensorflow [75] version 1.11.

Hardware - All models have been trained on a Nvidia RTX 2080Ti GPU that offers 11Gb of VRAM.

6.5 Datasets

The following sections describe three different datasets, which have been used to evaluate the performance of the GAN-based reconstruction methods addressed in this thesis.

6.5.1 MNIST

The MNIST (Modified National Institute of Standards and Technology) dataset consist of 70.000 labeled handwritten digits. The images are of size 28 $\times$ 28, gray-valued and centered [76]. As the DCGAN architecture uses transposed convolutions to up-sample each feature map by a factor of 2, samples have been resized to 32×32 in order to obtain an image size that can be expressed as $2^x, x \in \mathbb{N}$. Ten representative samples from the dataset can be seen in figure 6.6.

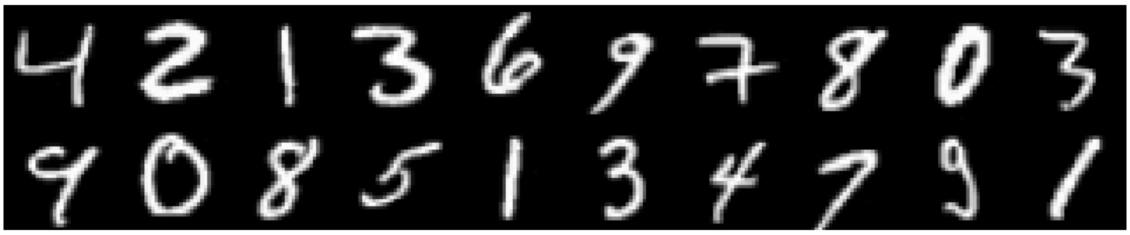

Figure 6.6: Samples from MNIST dataset.

6.5.2 CelebA

Large-scale CelebFaces Attributes (CelebA) is a dataset that consits of more than 200K celebrity images with more than 40 attribute annotations per sample. The dataset covers various different poses and background clutter [77]. Furthermore, faces are aligned by eyes and mouth as well as cropped to obtain equal face sizes and position. Images are RGB and of size 218 $\times$ 178. In order to fit the requirements for DCGAN, samples have been reshaped to 128 $\times$ 128 . Figure 6.7 shows representative samples from the dataset.

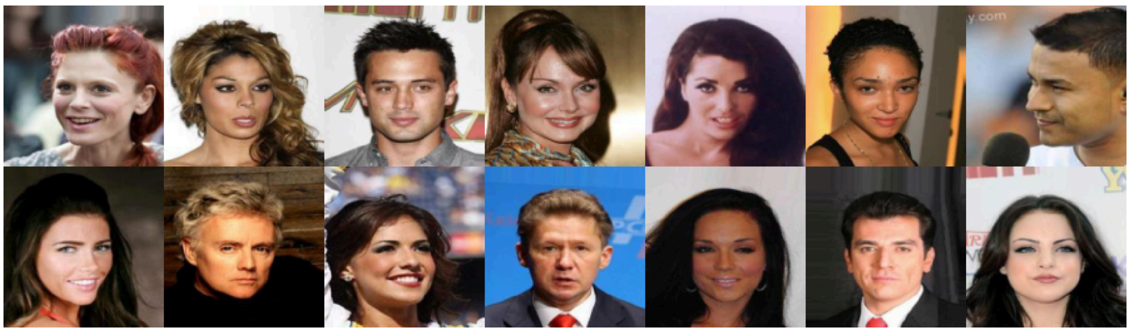

Figure 6.7: Samples from CelebA dataset.

6.5.3 MRI Data

The MRI data used in this thesis was provided by the Human Connectome Project. The dataset contains 1113 MRI scans of human heads, where each scan consists of 260 slices. Each slice is of size 260×311 and was resized to 128×128. Only the first 215 slices were used, since the last 45 slices contain almost no tissue. Samples from the dataset can be seen in figure 6.8.

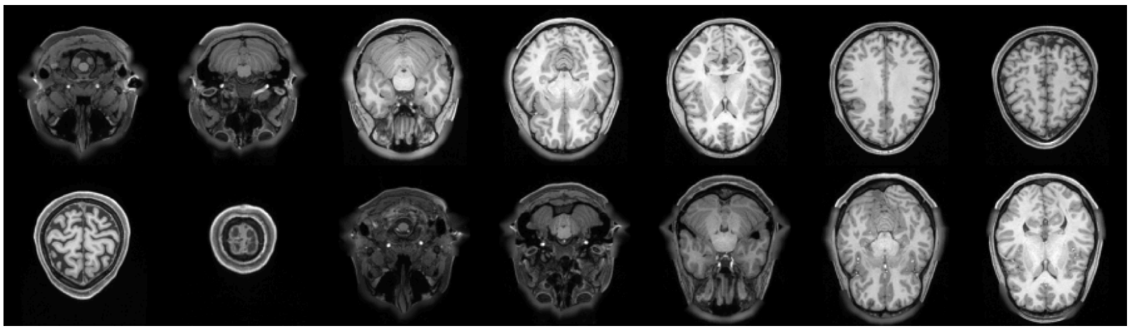

Figure 6.8: Samples from MRI dataset.

7 Results

Contents

The following chapter presents the qualitative and quantitative results, which were obtained by using the methods described in chapter 6. Every experiment has been performed with auxiliary optimization, using both the generator of the standard WGAN as well as the InfoGAN generator. All inverse problems have additionally been addressed with standard algorithms to provide a baseline for the reconstructions. In order to numerically compare reconstruction quality, the peak signal-to-noise ratio (PSNR) between 100 uncorrupted samples and the corresponding reconstructions was calculated. Furthermore, in order to obtain the degree of improvement, the PSNR between original and corrupted samples was evaluated and reported.

The PSNR of two images $a, b \in \mathbb{R}^{M \times N \times C}$ is computed by:

$$\mathrm{PSNR} = 20 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\frac{1}{MNC}\|a-b\|^2}\right), \tag{7.1}$$

where MAX_I denotes the range between lowest and highest possible pixel value. The 100 samples used for evaluation have not been used in the training process of the GAN.

Since, in general, the extended architecture delivered slightly better results, all images shown in this chapter are obtained from the Info-GAN generator. Additional qualitative results can be found in the appendix B.2.

7.1 Auxiliary optimization of θ

To demonstrate the advantage of auxiliary optimization, uncorrupted samples from the CelebA dataset have been reconstructed with (figure 7.1) and without (figure 7.2) auxiliary optimization. The PSNR of 100 uncorrupted samples was computed for both methods, where the native approach resulted in a PSNR of 17.6 ± 2.21 and the auxiliary optimization resulted in a PSNR of 28.6 ± 1.41.

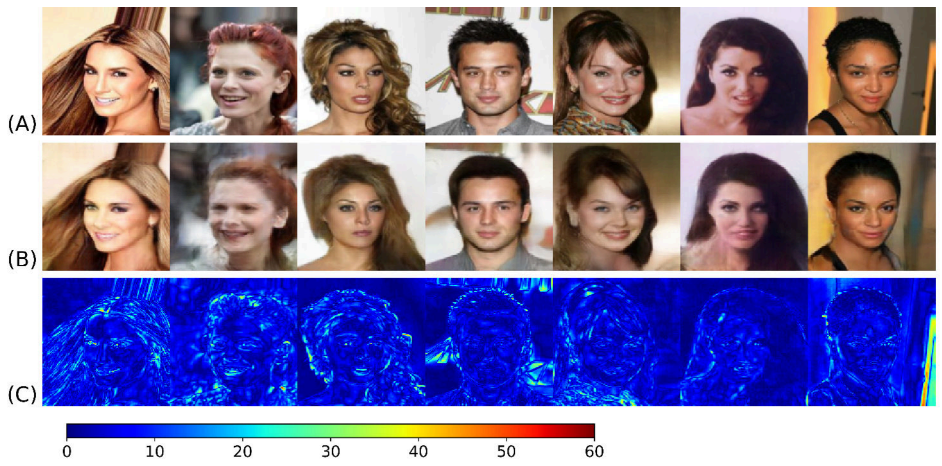

Figure 7.1: Reconstruction without auxiliary optimization. (A) shows original data. (B) shows reconstructed data. (C) shows the absolute difference between (A) and (B).

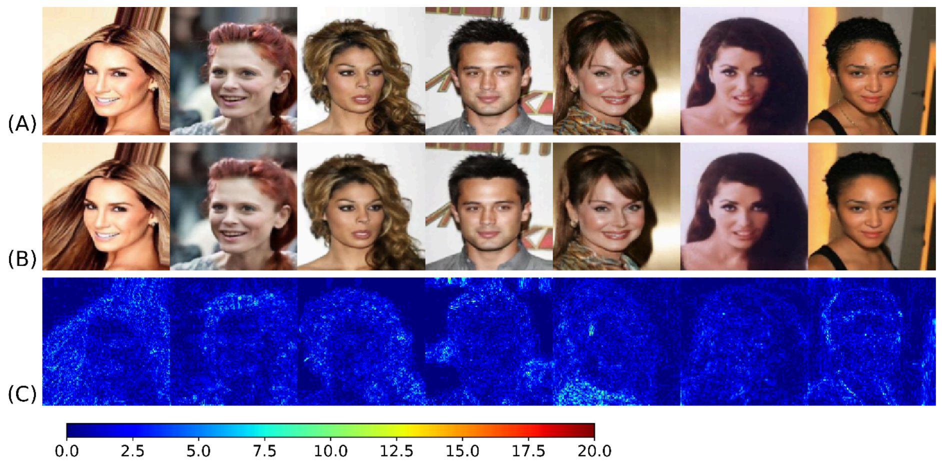

Figure 7.2: Reconstruction with auxiliary optimization. (A) shows original data. (B) shows reconstructed data. (C) shows the absolute difference between (A) and (B).

7.2 Denoising

Figures 7.3 to 7.5 illustrate the qualitative results for the task of denoising. The samples were corrupted by Gaussian noise with a standard deviatian of $\sigma = 12.5\%$ for the **MNIST** and the **CelebA** dataset and $\sigma = 5.0\%$ for the **MRI** dataset. The baseline was generated by using the BM3D denoising algorithm, proposed in [72]. The quantitative results in the form of PSNR can be observed in histograms 7.6 to 7.8, as well as in table 7.1.

7.2.1 Qualitative Results

In the following figures 7.3 to 7.5, row (A) shows the original data, row (B) shows the corrupted data, row (C) shows the reconstruction with the baseline approach, row (D) shows the absolute difference between (A) and (C), row (E) shows the GAN-based reconstruction and row (F) shows the absolute difference between (A) and (E).

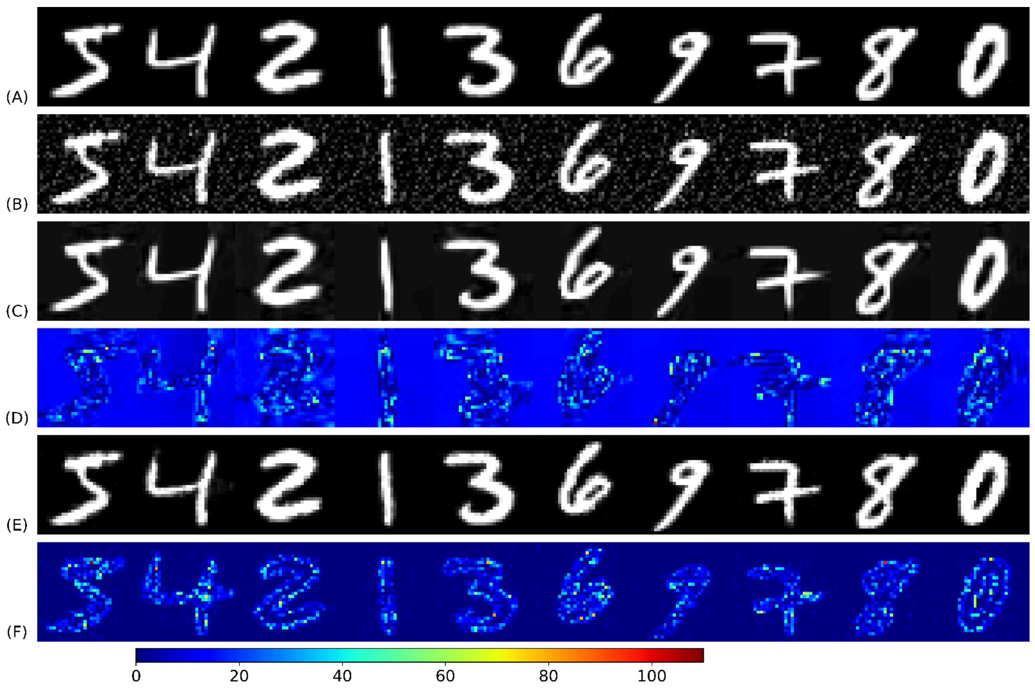

Figure 7.3: Denoising reconstruction of MNIST data.

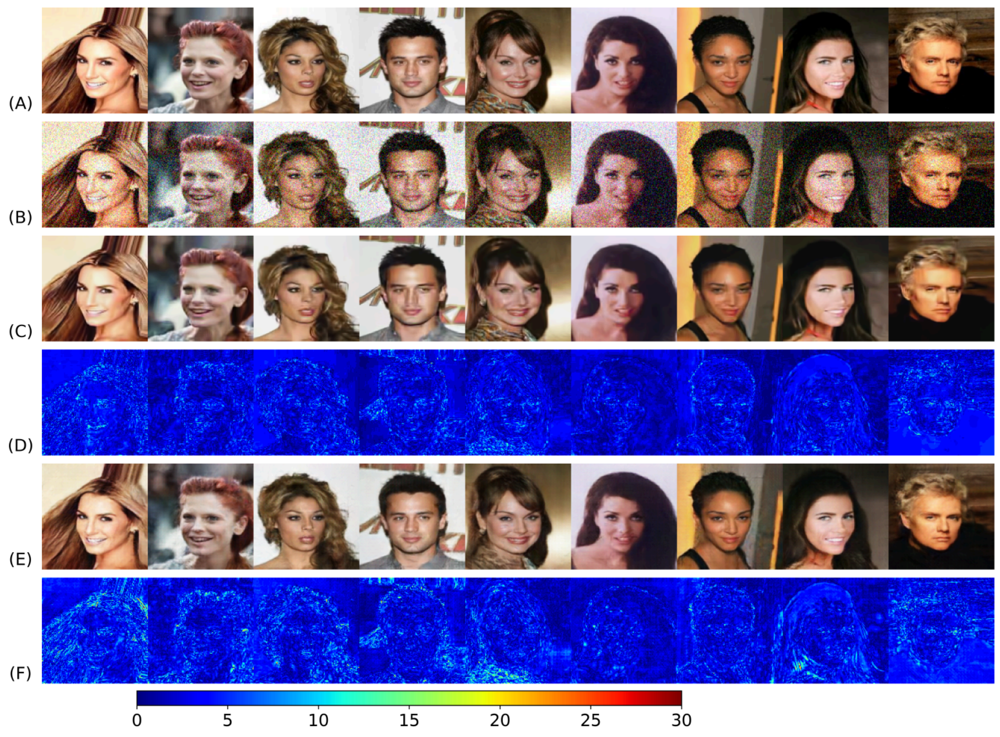

Figure 7.4: Denoising reconstruction of CelebA-data.

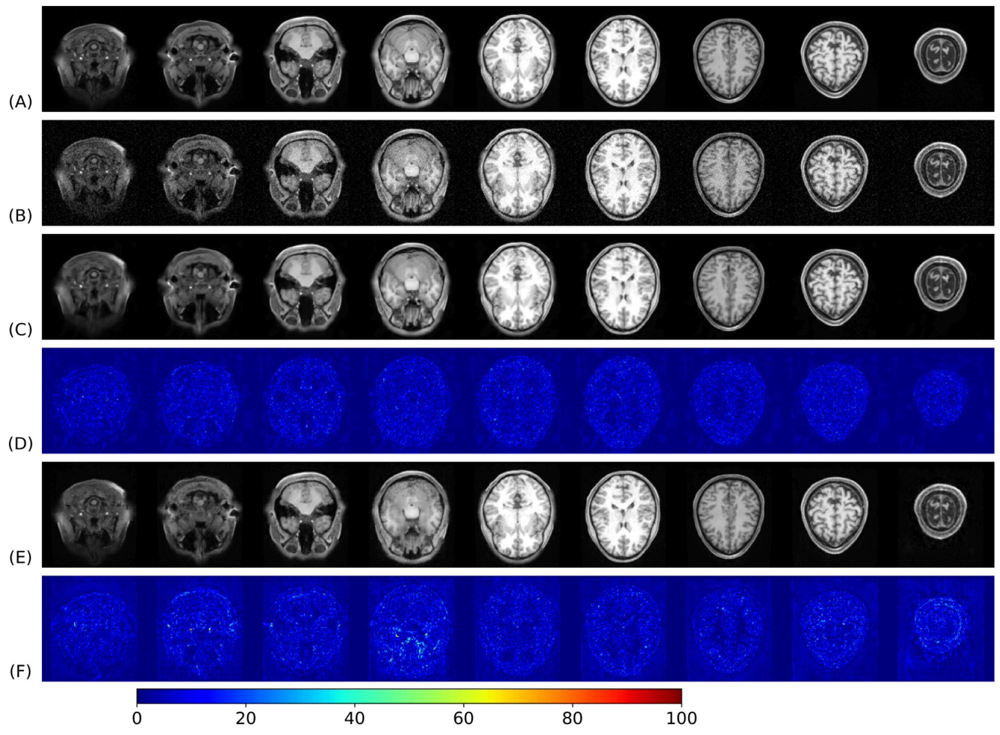

Figure 7.5: Denoising reconstruction of MRI data.

7.2.2 Quantitative Results

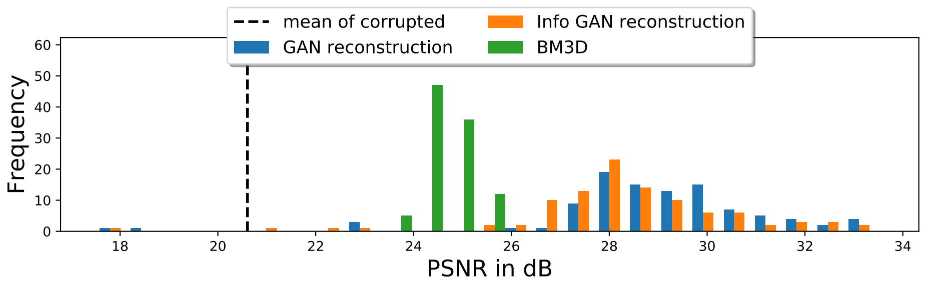

Figure 7.6: PSNR histogram of reconstruction results for MNIST dataset

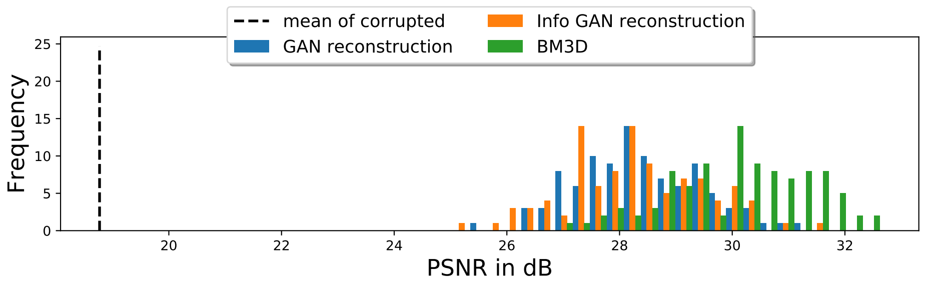

Figure 7.7: PSNR histogram of reconstruction results for CelebA dataset.

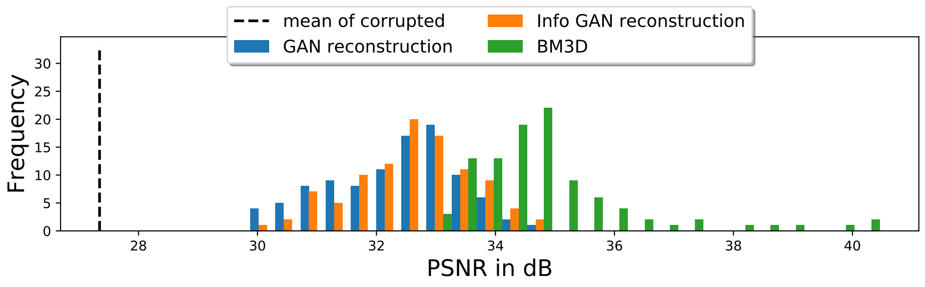

Figure 7.8: PSNR histogram of reconstruction results for MRI dataset.

Denoising Results

	No Rec	BM3D Rec	GAN Rec	Info GAN Rec	Optimizer	α	λ	β
MNIST	20.60 ± 0.12	24.65 ± 0.44	28.99 ± 2.38	28.43 ± 2.24	NAG/NAG	1e-3/1e-6	1	50
CelebA	18.77 ± 0.34	30.13 ± 1.24	28.38 ± 1.10	28.30 ± 1.23	Adam/Adam	1e-5/1e-6	1	150
MRI	27.35 ± 0.33	34.86 ± 1.41	32.36 ± 1.06	32.62 ± 0.98	Adam/Adam	1e-5/1e-6	1	20

Table 7.1: PSNR results for all datasets (mean ± std.) along with the optimizers and parameter settings.

7.3 Inpainting

The qualitative results for the inpainting task can be seen in figures 7.9 to 7.11. All samples were corrupted with an identical hand-drawn inpainting mask, which was resized to fit the image dimensions. The baseline was generated by using an inpainting algorithm based on the biharmonic equation. The quantitative results in the form of PSNR can be seen in histograms 7.12 to 7.14, as well as in table 7.2.

7.3.1 Qualitative Results

In the following figures 7.9 to 7.11, row (A) shows the original data, row (B) shows the corrupted data, row (C) shows the reconstruction with the baseline approach, row (D) shows the absolute difference between (A) and (C), row (E) shows the GAN-based reconstruction and row (F) shows the absolute difference between (A) and (E).

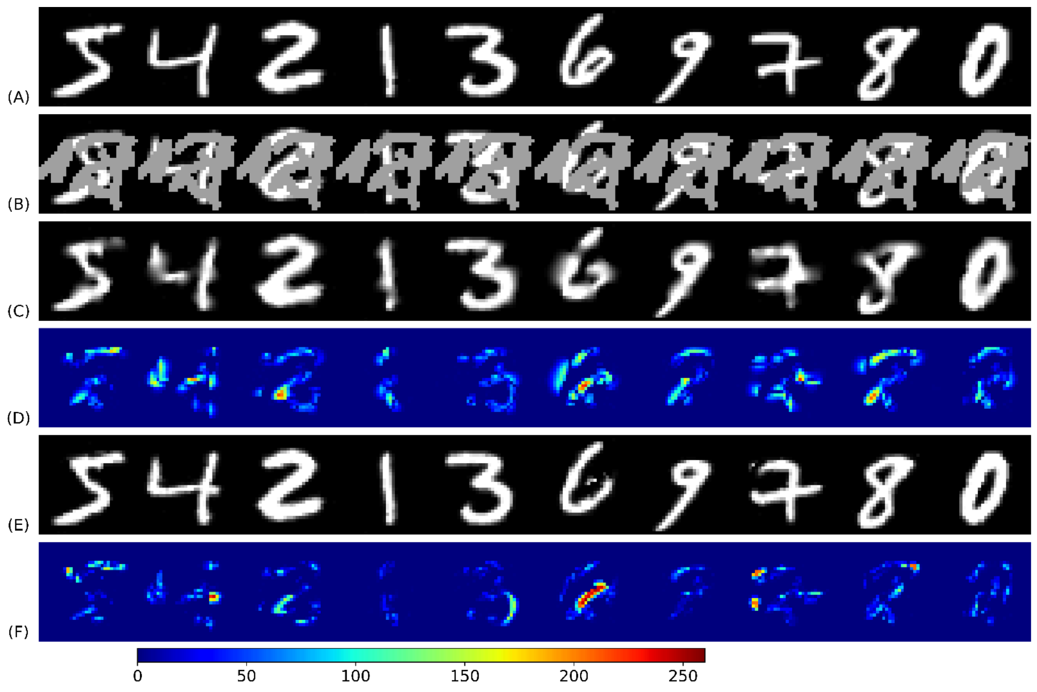

Figure 7.9: Inpainting reconstruction of MNIST data.

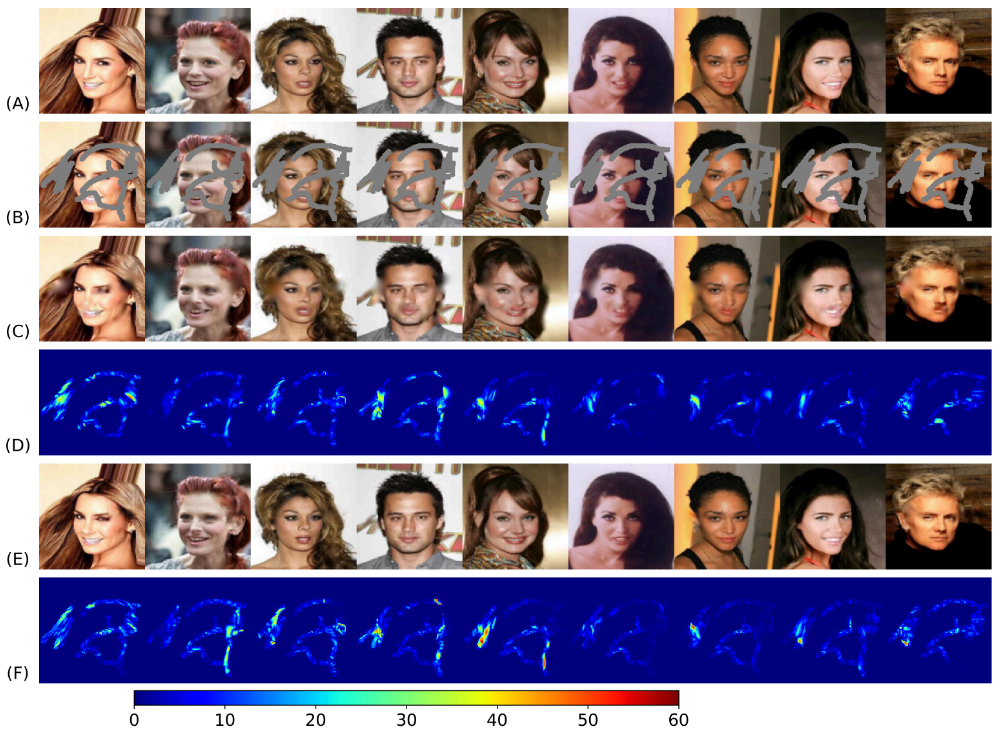

Figure 7.10: Inpainting reconstruction of CelebA data.

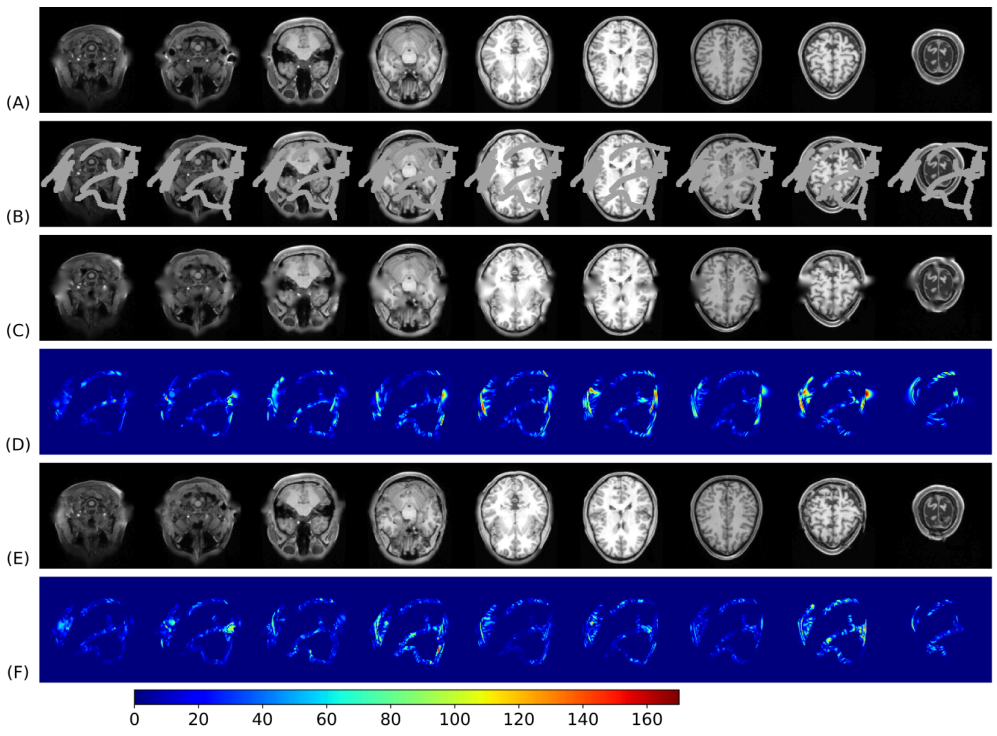

Figure 7.11: Inpainting reconstruction of MRI data.

7.3.2 Quantitative Results

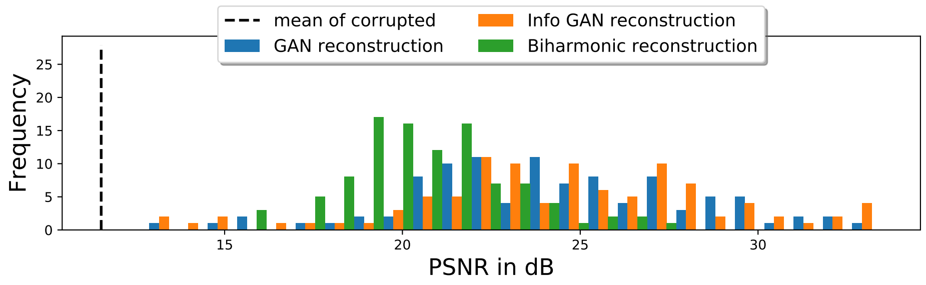

Figure 7.12: PSNR histogram of reconstruction results for MNIST dataset.

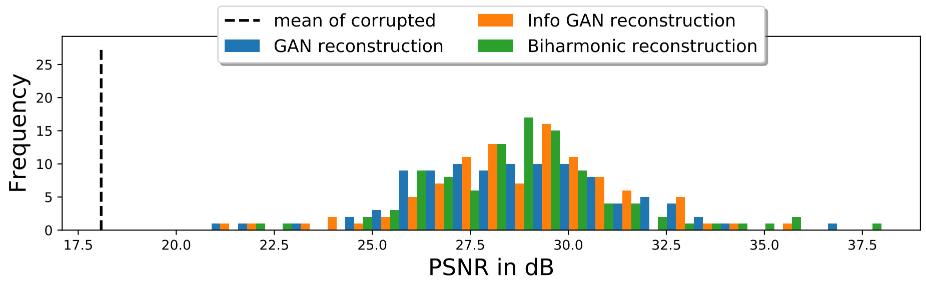

Figure 7.13: PSNR histogram of reconstruction results for CelebA dataset.

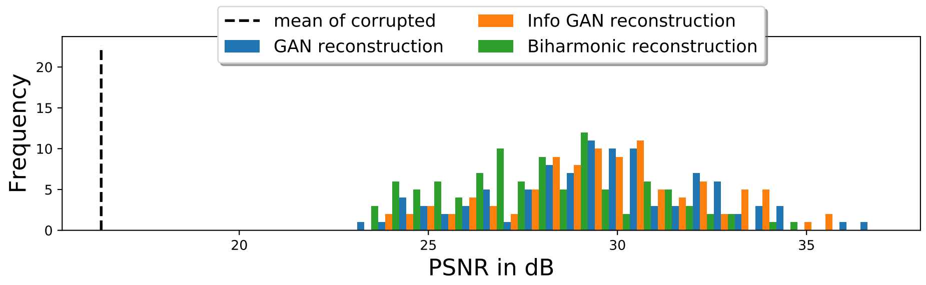

Figure 7.14: PSNR histogram of reconstruction results for MRI dataset.

Inpainting Results

	No Rec	Biharmonic Rec	GAN Rec	Info GAN Rec	Optimizer	α	λ	β
MNIST	11.52 ± 0.22	20.66 ± 2.29	24.27 ± 3.89	24.57 ± 4.26	NAG/NAG	1e-3/1e-6	1	20
CelebA	18.09 ± 1.57	28.74 ± 2.52	28.81 ± 2.63	28.81 ± 2.49	NAG/NAG	1e-5/1e-6	1	20
MRI	16.35 ± 1.18	27.82 ± 2.57	29.66 ± 2.78	29.77 ± 2.64	Adam/Adam	1e-5/1e-6	0	0

Table 7.2: PSNR results for all datasets (mean ± std.) along with the optimizers and parameter settings.

7.4 Super-Resolution

The qualitative results for the task of super-resolution can be seen in figures 7.15 to 7.17. Samples for all datasets were upscaled by a factor of 2. The corrupted images were generated by blurring with a Gaussian blur kernel of size 10×10 and a standard deviation of $\sigma = 1.0$ before down-sampling. The baseline was generated by bicubic upsampling of down-sized samples. Quantitative results in the form of PSNR can be observed in histograms 7.18 to 7.20, as well as in table 7.3.

7.4.1 Qualitative Results

In the following figures 7.15 to 7.17, row (A) shows the original data, row (B) shows the corrupted data, row (C) shows the reconstruction with the baseline approach, row (D) shows the absolute difference between (A) and (C), row (E) shows the GAN-based reconstruction and row (F) shows the absolute difference between (A) and (E).

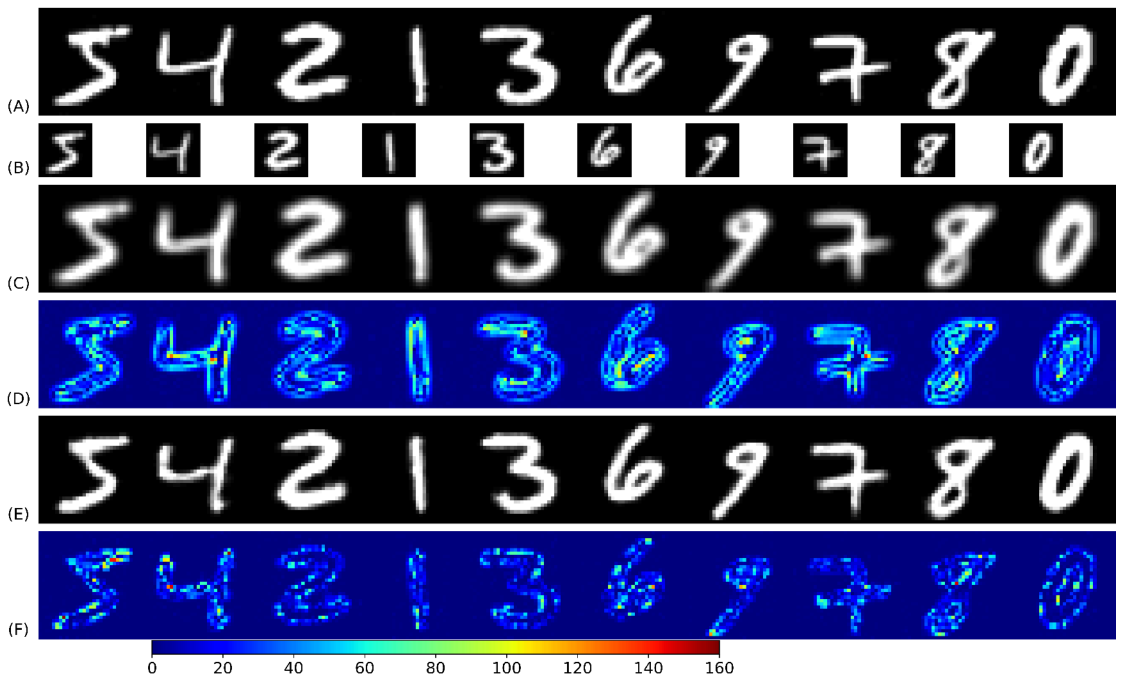

Figure 7.15: Super-resolution of MNIST data.

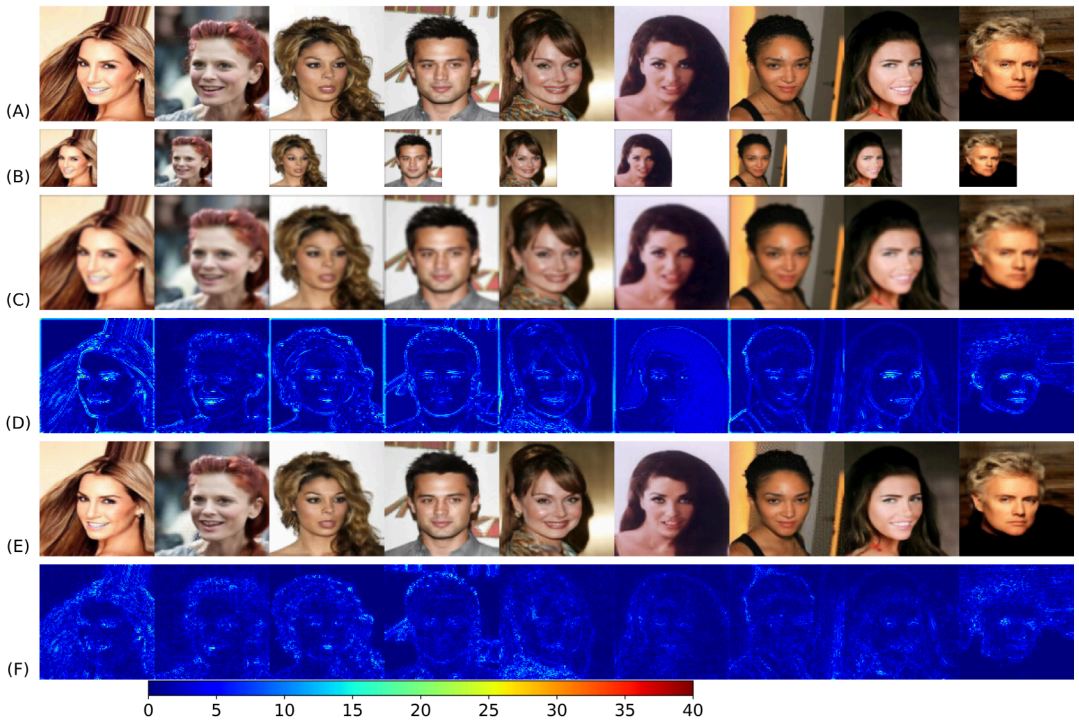

Figure 7.16: Super-resolution of CelebA data.

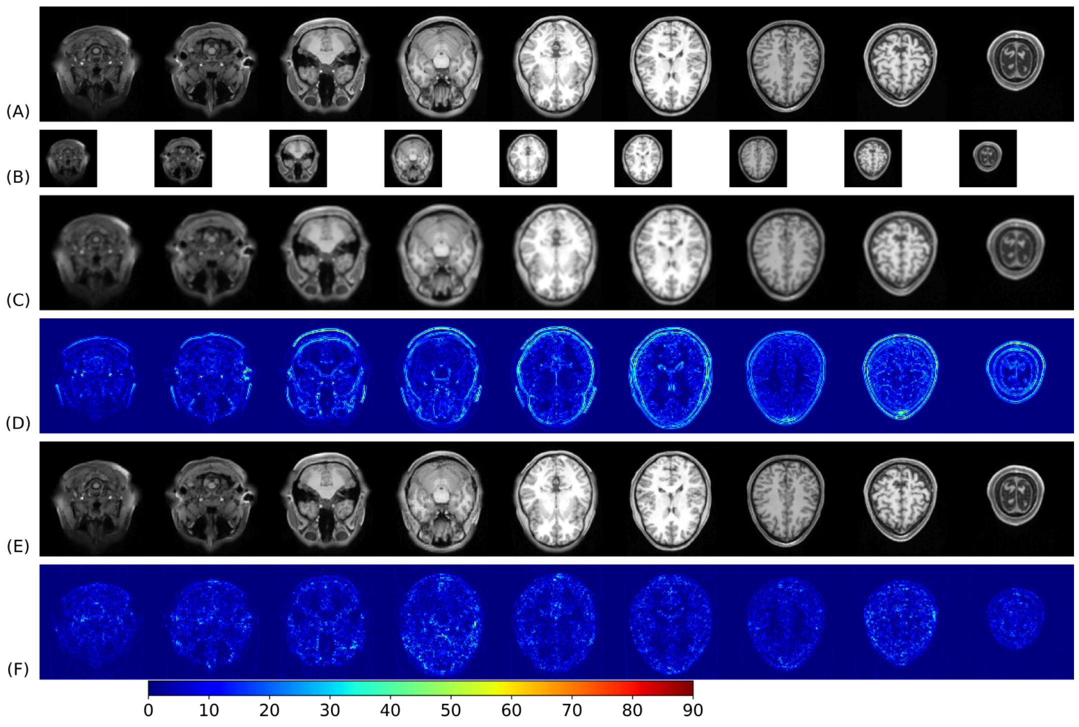

Figure 7.17: Super-resolution of MRI data.

7.4.2 Quantitative Results

Note: As the image dimensions regarding the corrputed and the source image are not equal in super-resolution, no mean PSNR of corrupted samples was calculated.

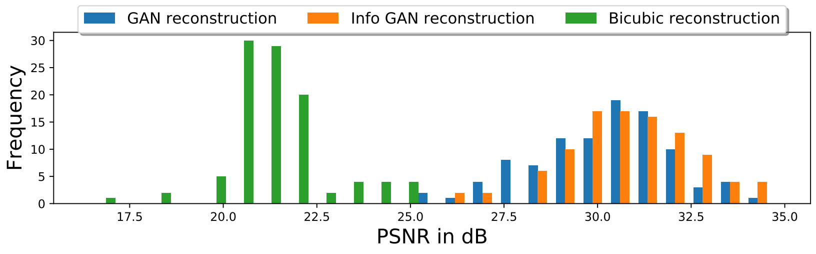

Figure 7.18: PSNR histogram of reconstruction results for MNIST dataset.

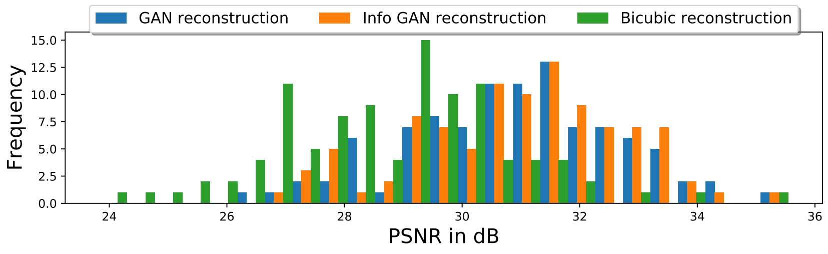

Figure 7.19: PSNR histogram of reconstruction results for CelebA dataset.

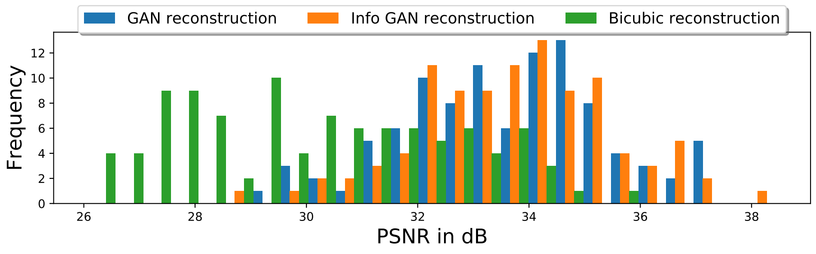

Figure 7.20: PSNR histogram of reconstruction results for MRI dataset.

Super-resolution Results							
	Bicubic Rec	GAN Rec	Info GAN Rec	Optimizer	α	λ	β
MNIST	21.53 ± 1.28	30.15 ± 1.90	30.74 ± 1.71	Adam/Adam	1e-5/1e-6	0	0
CelebA	28.82 ± 2.61	30.92 ± 1.86	31.05 ± 1.86	Adam/Adam	1e-5/1e-6	0	0
MRI	30.19 ± 2.39	33.66 ± 1.80	33.79 ± 1.77	Adam/Adam	1e-5/1e-6	0	0

Table 7.3: PSNR results for all datasets (mean ± std.) along with the optimizers and parameter settings.

7.5 Deblurring

The qualitative results for image deblurring are shown in figures 7.21 to 7.23. All samples were blurred by a Gaussian blur kernel of size 15×15 with standard deviation set to $\sigma = 2.0$ for the **MNIST** dataset and $\sigma = 1.0$ for the **CelebA** and **MRI** dataset. The baseline was generated by deblurring with a Wiener-filter. Quantitative results in the form of PSNR can be seen in histograms 7.24 to 7.26, as well as in table 7.4.

7.5.1 Qualitative Results

In the following figures 7.21 to 7.23, row (A) shows the original data, row (B) shows the corrupted data, row (C) shows the reconstruction with the baseline approach, row (D) shows the absolute difference between (A) and (C), row (E) shows the GAN-based reconstruction and row (F) shows the absolute difference between (A) and (E).

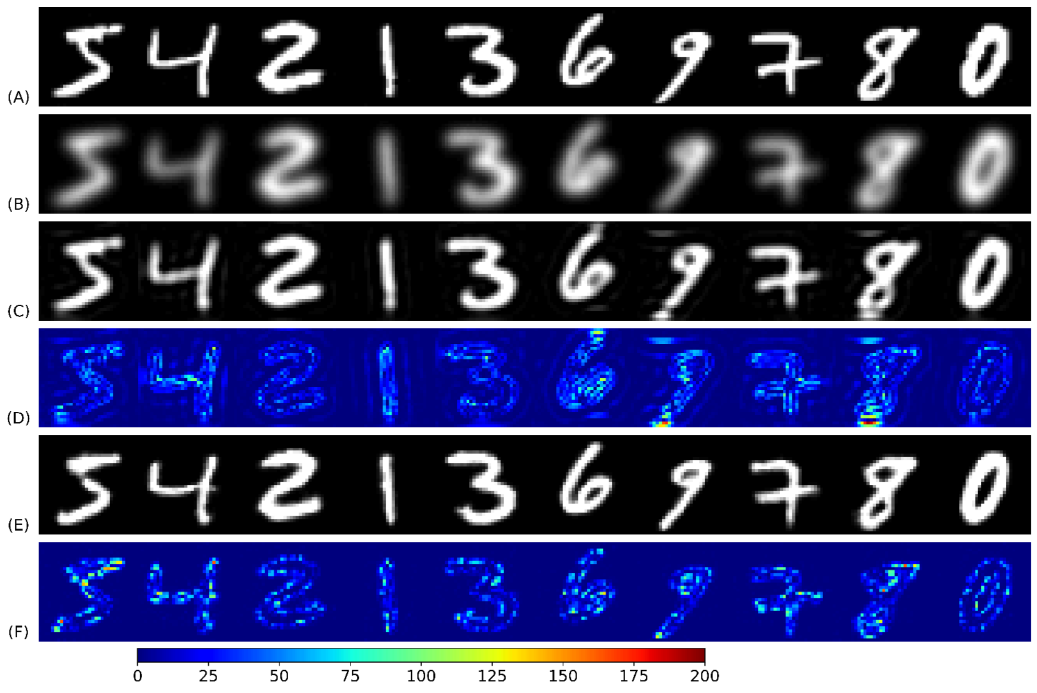

Figure 7.21: Deblurring reconstruction of MNIST data.

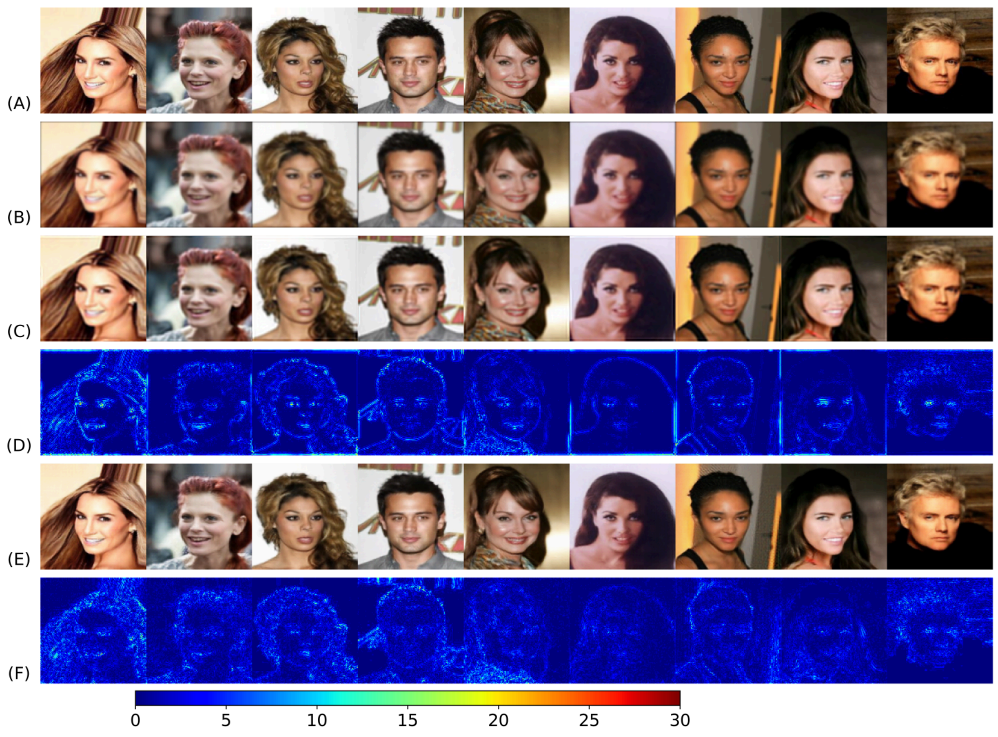

Figure 7.22: Deblurring reconstruction of CelebA data.

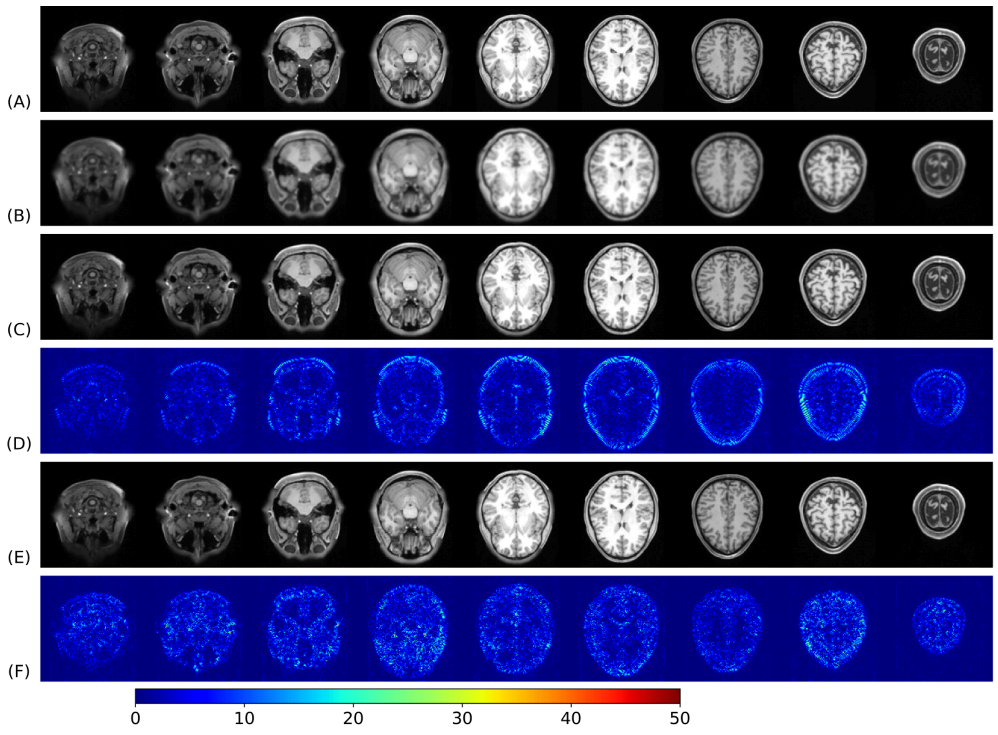

Figure 7.23: Deblurring reconstruction of MRI data.

7.5.2 Quantitative Results

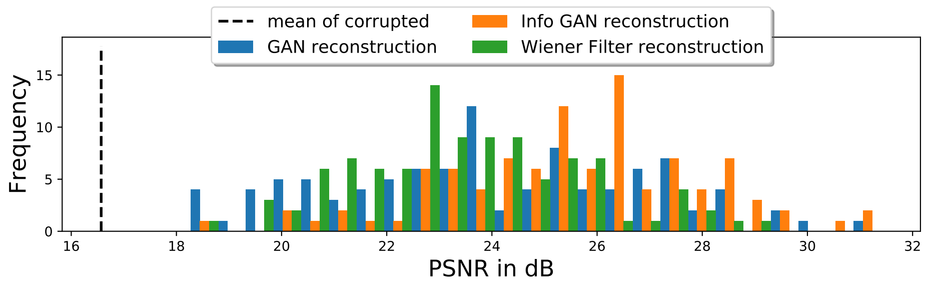

Figure 7.24: PSNR histogram of reconstruction results for MNIST dataset.

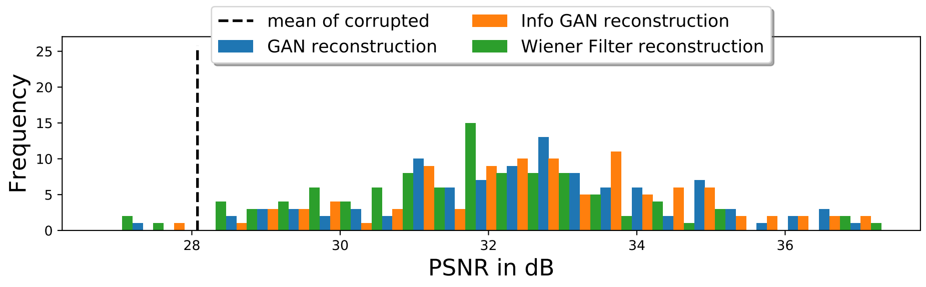

Figure 7.25: PSNR histogram of reconstruction results for CelebA dataset.

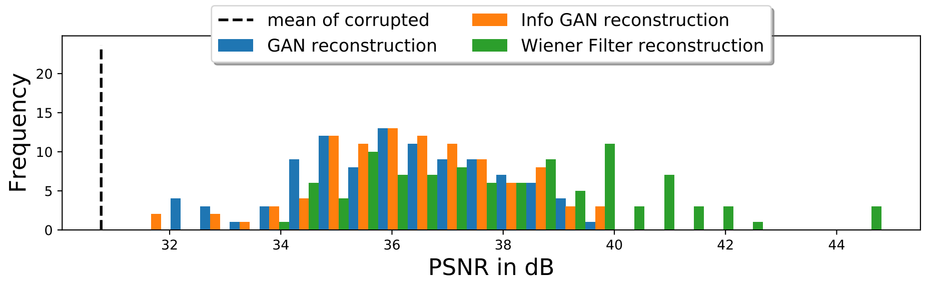

Figure 7.26: PSNR histogram of reconstruction results for MRI dataset.

Deblurring Results

	No Rec	Wiener Filter	GAN Rec	Info GAN Rec	Optimizer	α	λ	β
MNIST	16.57 ± 1.03	23.50 ± 2.16	24.03 ± 2.98	25.62 ± 2.40	Adam/Adam	1e-3/1e-6	0	0
CelebA	28.07 ± 2.15	31.51 ± 1.99	32.66 ± 1.97	32.80 ± 1.96	Adam/Adam	1e-5/1e-6	0	0
MRI	30.77 ± 2.38	38.10/ ± 2.42	36.13 ± 1.74	36.41 ± 1.66	Adam/Adam	1e-5/1e-6	0	0

Table 7.4: PSNR results for all datasets (mean ± std.) along with the optimizers and parameter settings.

7.6 MRI Reconstruction

The qualitative results of compressed sensing reconstruction for the **MNIST** and the **MRI** dataset are shown in figures 7.27 and 7.28. Quantitative results in the form of PSNR can be observed in histograms 7.29 and 7.30, as well as in table 7.5. Undersampling was performed by setting every second line in Fourier space to zero, which corresponds to an acceleration factor of 2 in fast magnetic resonance imaging. No baseline algorithm was implemented for this task.

7.6.1 Qualitative Results

In the following figures 7.21 to 7.23, row (A) shows the original data, (B) shows the corrupted data, (C) shows the reconstructed data and (D) shows the absolute difference between (A) and (C).

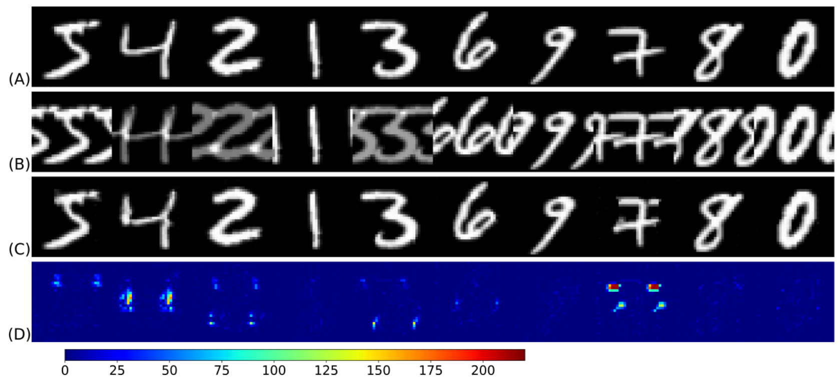

Figure 7.27: Compressed Sensing reconstruction of MNIST data.

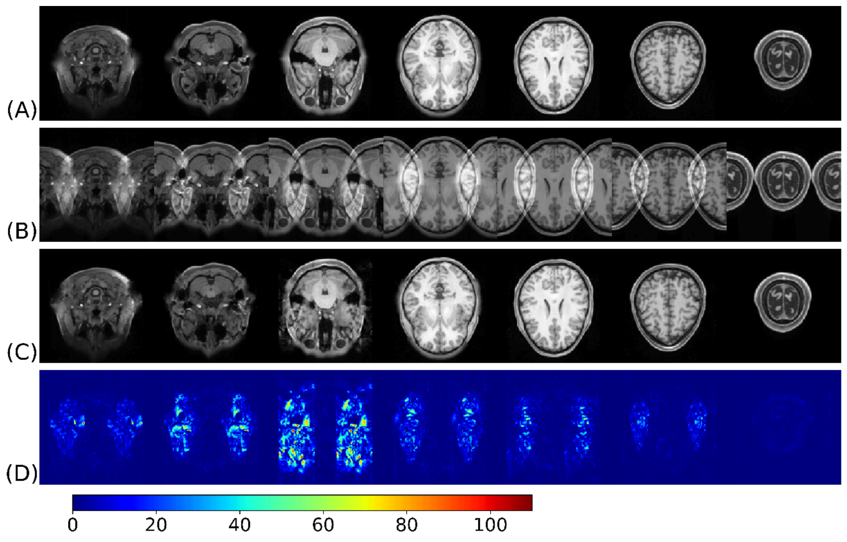

Figure 7.28: Compressed Sensing reconstruction of MRI data.

7.6.2 Quantitative Results

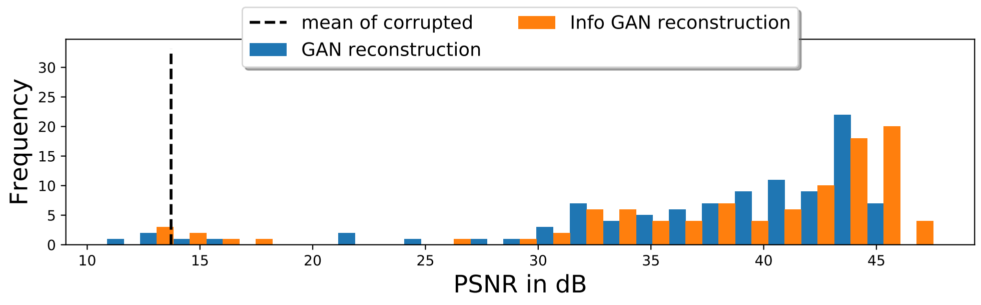

Figure 7.29: PSNR histogram of reconstruction results for MNIST dataset.

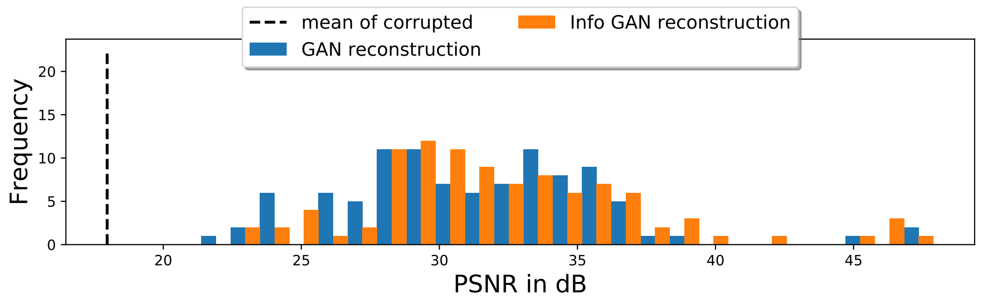

Figure 7.30: PSNR histogram of reconstruction results for MRI dataset.

MRI reconstruction Results							
	No Rec.	GAN Rec.	Info GAN Rec.	Optimizer	α	λ	β
MNIST	13.71 ± 1.63	37.69 ± 7.68	38.74 ± 8.27	Adam/Adam	1e-3/1e-5	0	10
MRI	17.97 ± 3.33	31.39 ± 4.73	32.33 ± 5.02	Adam/Adam	1e-2/1e-6	0	10

Table 7.5: PSNR results for all datasets (mean ± std.) along with the optimizers and parameter settings.

7.7 Interpolation

Simultaneous auxiliary optimization of two uncorrupted samples allows performing vector arithmetical operations. Figures 7.31 to 7.34 show the resulting intermediate images when linear interpolation is performed between the obtained latent vectors z_1 and z_2.

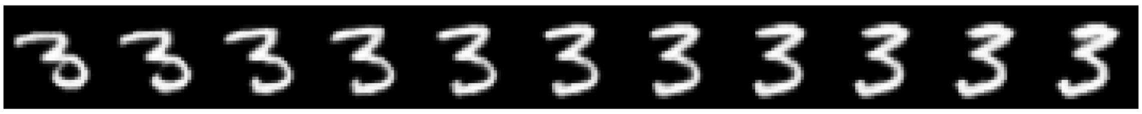

Figure 7.31: Interpolation of two MNIST digits from the same digit class with different rotation.

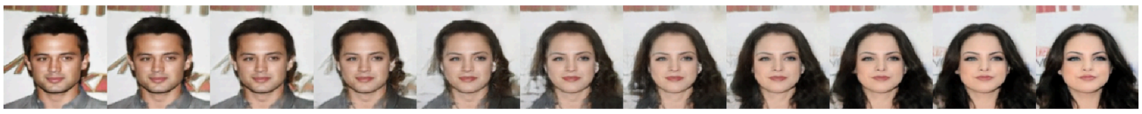

Figure 7.32: Interpolation of a short haired male and long haired female sample from the CelebA dataset.

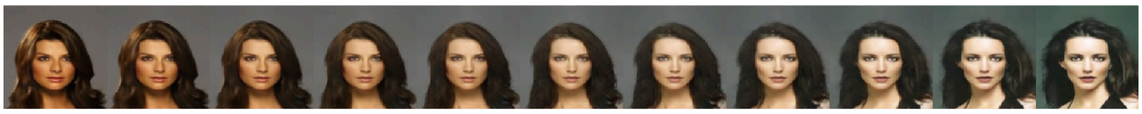

Figure 7.33: Interpolation of two long haired female samples from the CelebA dataset.

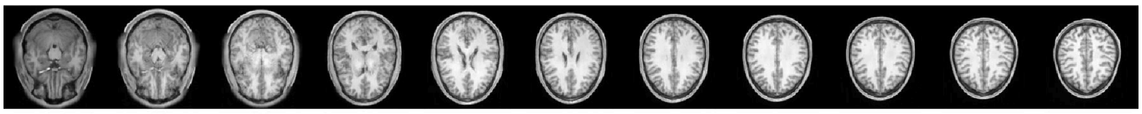

Figure 7.34: Interpolation of two brain MRI slices obtained at different vertical positions.

7.8 Generated Data

As a pre-trained generator is necessary to provide a good prior on the data distribution, a GAN needs to be trained for a specific dataset in advance. Figures 7.35 to 7.37 show samples that were produced by the generators of GANs trained in this thesis. All samples are generated randomly (no cherry picking).

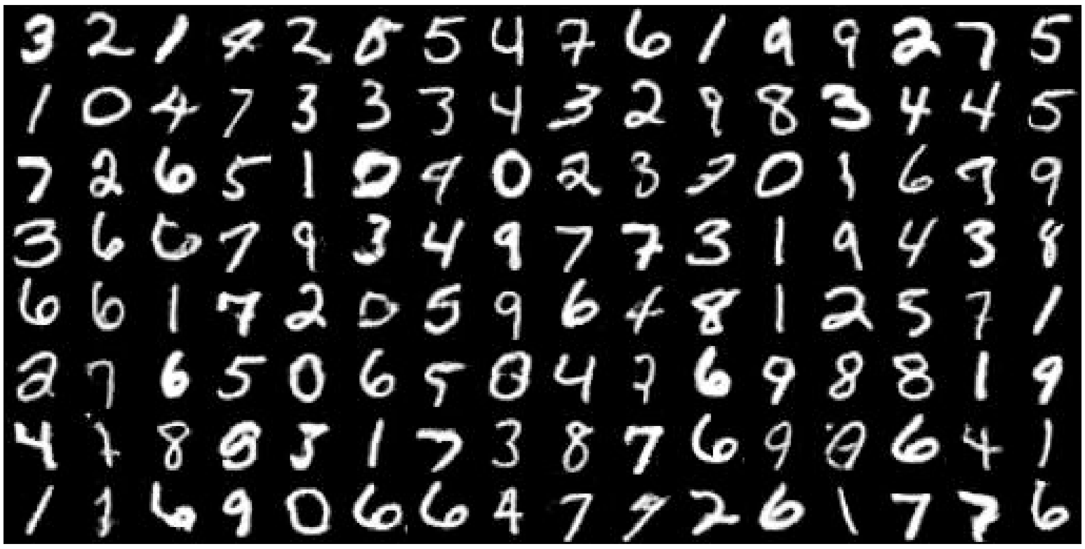

Figure 7.35: Data generated with GAN trained on MNIST dataset.

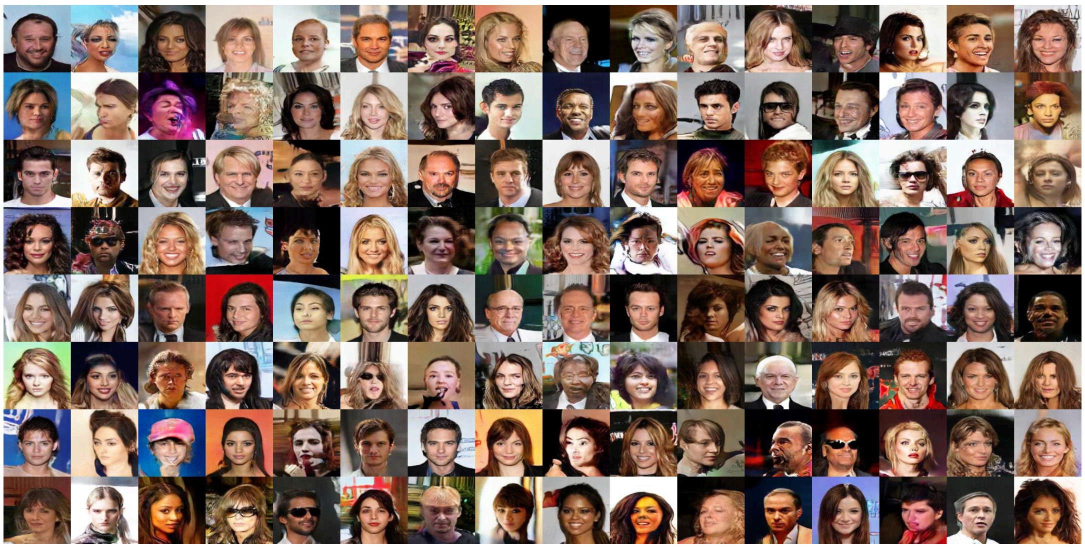

Figure 7.36: Data generated with GAN trained on CelebA dataset.

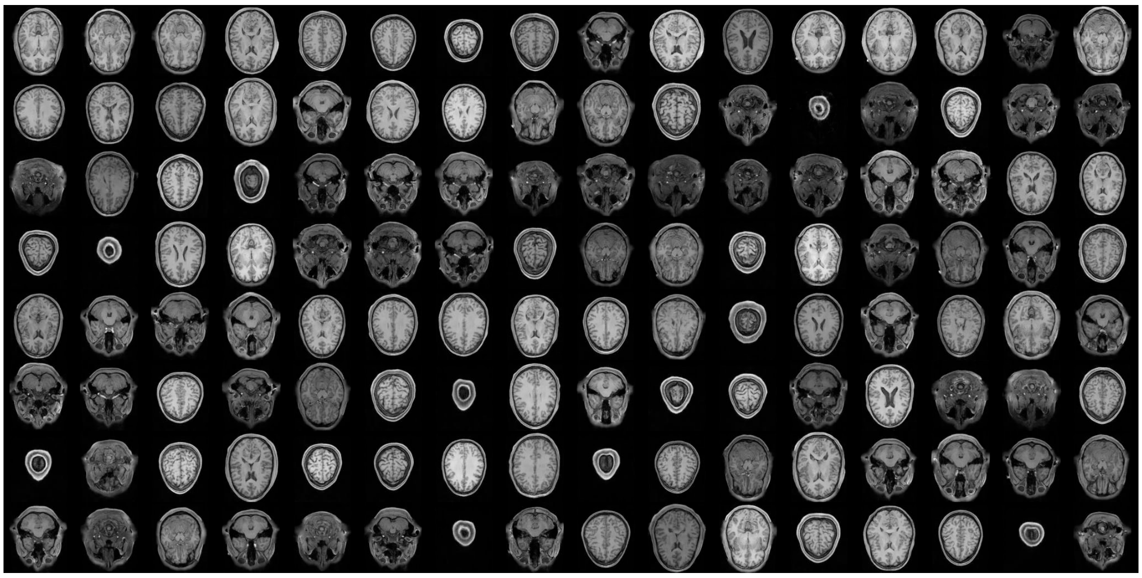

Figure 7.37: Data generated with GAN trained on MRI dataset.

7.9 Discussion

Importance of auxiliary optimization: The experiments conducted in section 7.1 showed that it is crucial to further refine the approximated distribution of the generator using auxiliary optimization in order to obtain satisfying representations. Images produced by latent vectors recovered without auxiliary optimization look relatively similar. However, they are still clearly different from the source image, which is in accordance with the findings in other work, such as [69].

Inverse Problems:
The experimental results presented in sections 7.2 to 7.6 give evidence that the proposed methods are able to solve commonly encountered inverse problems. Auxiliary optimization was again performed for all tasks in order to obtain satisfying results.

- ***Denoising:*** The qualitative results for denoising the MNIST dataset (figure 7.3) show that the proposed method (Info-GAN with auxiliary optimization) slightly improves the reconstruction in comparison to the baseline approach (BM3D). This finding is also reinforced by comparing the provided plots of the absolute difference between the source image and reconstruction (difference plots). In contrast, the qualitative results for denoising the CelebA dataset (figure 7.4) and the MRI dataset (figure 7.5), respectively, show a slightly better reconstruction when using BM3D, although the differences to the proposed method are subtle. This is also visible when comparing the corresponding difference plots. However, from a purely visual point of view, the GAN-based reconstructions appear more natural.

The quantitative results illustrated in histograms (figures 7.6 to 7.8) and summarized in table 7.1 show that for the MNIST dataset, both the reconstruction with a classical GAN and an InfoGAN clearly outperform the baseline method. The classical GAN approach yields even slightly better results than the InfoGAN approach. However, there is also a very small fraction of samples for which the GAN-based reconstructions fail, which is reflected in a PSNR that is even lower than the mean PSNR of the corrupted samples. When it comes to the CelebA as well as the MRI dataset, BM3D performs significantly better than both GAN-based methods.

- ***Inpainting:*** When inspecting the qualitative results (figures 7.9 to 7.11) for the inpainting task, it can be seen that the reconstructed images delivered by the proposed approach are visually much more appealing than the reconstructions of the baseline method (biharmonic inpainting). When only considering and comparing the difference plots, however, there is no clear evidence which method performs better.

  The quantitative results of the inpainting task are presented in histograms (figures 7.12 to 7.14) and table 7.2. For all datasets, the InfoGAN-based reconstruction approach yields the best results in terms of PSNR. However,when it comes to the CelebA dataset, all methods performed almost equally well, which is in clear contrast to the mere visual impression of the reconstructed samples. This suggests that the PSNR is not necessarily an ideal performance measure when it comes to evaluating the quality of a reconstructed image.

- ***Super-resolution:*** The qualitative results for inpainting, seen in figures 7.15 to 7.17, show that the GAN-based method gives considerably better reconstruction results for all datasets, in comparison to the baseline method (bicubic interpolation), which is also clearly recognizable in the provided difference plots.

  The quantitative results presented in histograms (figures 7.18 to 7.20) and summarized in table 7.3 show a significantly better performance in terms of PSNR for the proposed approach in comparison to the baseline method for the MNIST and MRI dataset. Also for the CelebA dataset better reconstruction results could be achieved with the GAN-based approach, however the difference to the baseline is less pronounced.

- ***Deblurring:*** The qualitative results for the task of deblurring can be seen in figures 7.21 to 7.23. When it comes to the MNIST dataset, it can be observed that the GAN-based reconstruction yields qualitatively better results, as the baseline approach (Wiener filter) introduces artifacts and produces less sharp images in general. When only considering the difference plots of the CelebA and MRI dataset, no clear statement about the performance of the reconstruction algorithms can be made. As, for the CelebA dataset, the reconstructed images obtained by the GAN-based approach, look sharper and therefore visually more appealing.

  The histograms in figures 7.24 to 7.26 and table 7.4 show the quantitative results for deblurring. It can be observed that for MNIST and CelebA, the GAN-based reconstruction yields the best results, whereas for MRI data the Wiener filter performs better.

- ***MRI Reconstruction:*** The qualitative reconstruction results for the MNIST dataset and the MRI dataset presented in figures 7.27 and 7.28 show that images undersampled in Fourier domain can be successfully reconstructed by the proposed GAN-based approach. However, for some samples, small artifacts can be observed after reconstruction, which is especially highlighted in the difference plots.

  The quantitative results presented in histograms 7.29 and 7.30 as well as in table 7.5, illustrate that for both datasets the majority of samples are reconstructed well. Nevertheless, for the MNIST dataset, both GAN-based approaches fail to reconstruct a small set of samples.

Overall, the qualitative results showed that the majority of the reconstructions obtained by the proposed GAN-based approach are naturally looking and of high visual quality. However, as the problems are ill-posed, recovering the exact original images is not possible Therefore, reconstructions show deviations from the source images, which is, however, acceptable to a certain degree. Also, the quantitative results show a major improvement to the corrupted samples for almost all samples in the experiments. In comparison to the baseline reconstruction algorithms, the proposed method at least shows equivalent performance, mostly outperforming the standard methods, except from denoising and deblurring the MRI dataset as well as denoising the CelebA dataset.

Furthermore, the experiments have shown that the hypothesis of better optimization conditions due to disentangled representations using InfoGAN is correct. In almost all inverse problems addressed in this thesis, reconstructions performed with the help of InfoGAN generators outperformed the reconstructions of the regular trained GAN, with up to 3 dB PSNR. Still, in a few cases reconstruction quality was low or even below the mean PSNR of corrupted samples, which means that the reconstruction has failed.

Sources of Error: A major reason for the occurrence of failure cases is the fact that a sufficiently good representation has to be found in the latent vector recovery, prior to auxiliary optimization. If this is not possible due to problems in the optimization, or because the desired sample is far beyond the region of the generator, reconstructions appear to be unnatural and corrupted. Nevertheless, the histograms in sections 7.2 to 7.6 show that the vast majority of corrupted samples can be reconstructed well.

Choice of optimization algorithm: While performing the experiments, it was found that different optimization algorithms need to be chosen for individual datasets and inverse problems in order to achieve the best results. In particular, the inverse problems of denoising and inpainting can be satisfyingly solved for the MNIST and CelebA dataset using gradient descent with Nesterov momentum, whereas for the MRI dataset, Adam had to be used in order to obtain good results. As for the tasks of super-resolution and deblurring, CelebA and MNIST needed to be optimized using Adam as well.

Choice of Hyperparameter: It was found, that regularization of the latent vector $\|z\|^2$ and the change in network parameters $\|\theta_0 - \theta\|^2$ is both task and dataset dependent. For denoising, regularization was performed for all applied datasets, as otherwise the auxiliary optimization step would overfit the noise in the image. For the inpainting task, no regularization was used in case of the MRI dataset. When it comes to the remaining inverse problems, regularization only delivered better results for the CelebA dataset.

Drawbacks: Even tough the presented results seem promising, several issues appear when using a GAN to solve inverse problems.

- Lack of generalization: In the proposed approach, a separate GAN needs to be trained for each individual data set. As each individual generator represents an approximation of a distinct distribution, generalization to data from a different distribution is not possible. This is a significant disadvantage to conventional methods, which can usually be used independently from the data distribution.

- Translation, Rotation variance: If unseen data does not show the same alignment as the training data, again the algorithm is prone to fail, since no sufficiently well representation in latent space can be found. Therefore, it is necessary to perform image registration on unseen data in order to achieve the required alignment.

- Limitation of resolution: Since samples generated by a GAN are limited to a certain image resolution, the reconstructions following from solving inverse problems are limited to that resolution as well. Again, this statement does in general not hold for conventional reconstruction methods.

- Failure Cases: No reasonable reconstruction can be obtained for certain samples in the datasets. As already mentioned, the main cause for this failure cases is the lack of finding a sufficiently good representation in the latent vector recovery step. Some failed reconstructions are shown in the appendix B.3.

- Computation Cost: Deep learning-based algorithms that directly model the inverse of the measurement process in the inference path are usually able to perform reconstructions almost in real-time. In contrast to this, a high number of iteration steps involving the computation of back-propagation need to be performed for the proposed method. If not implemented on GPU, reconstruction of a single sample of even low resolution can take up to several minutes.

Image Interpolation: The experiments shown in section 7.7 illustrate that samples generated by interpolation of latent vectors that were obtained by auxiliary optimization lead to meaningful results, as long as the initial samples do not deviate too far from each other. Interestingly, when interpolating two MRI slices of different depth, the interpolation results show anatomically correct structures as they are seen in real intermediate MRI slices, which provides further evidence that auxiliary optimization does not destroy the internal structure of the generator.

Generated Data: The data generated by the GANs trained in this thesis are of satisfying quality and at least comparable to results in other work [55], [56], [58], that use the same architectures and training methods.

8 Conclusion and Outlook

Conclusion

The aim of this master‘s thesis was to solve commonly encountered inverse problems in imaging using generative adverserial networks (GANs). In particular, the addressed problems included denoising, inpainting, super-resolution, deblurring and MRI reconstruction. In general, when solving inverse problems an analytical regularizer is required to enforce stability by incorporating a priori information. However, finding such an analytical regularizer is a tedious and complex task. In order to avoid the requirement of explicit regularization, GANs, which offer a strong implicit prior that constrains the reconstruction to the domain of source images, were used in this work. In contrast to the theoretic assumption, this implicit prior is only an approximation of the true source distribution, and does therefore not exactly cover the entire domain of source images. Thus, a method (auxiliary optimization) was proposed that manipulates the approximated distribution of the GAN by introducing an additional optimization task, which results in an improved representation space. This additional domain modification enables close to exact representations of unseen samples and thereby also improves reconstruction results of inverse problems. In order to evaluate the proposed methods, three different dataset (MNIST, CelebA and MRI) were used. First, the GANs (classical GAN and InfoGAN) were trained for each dataset separately. Then, unseen samples were corrupted and subsequently reconstructed using the according trained GAN in an inverse manner. All experiments in this thesis were conducted with both a classical GAN-based reconstruction and the corresponding InfoGAN extension using the proposed auxiliary optimization technique. Additionally, all inverse problems (except MRI reconstruction) were solved with standard methods in order to obtain a baseline. The majority of reconstructions obtained by the GAN-based approaches are visually highly appealing, satisfying with regard to PSNR and in general show a major improvement to the corrupted images. Moreover, the proposed method outperforms the baseline algorithms for almost all inverse problems, except from denoising and deblurring the MRI dataset as well as denoising the CelebA dataset. Additionally, samples from the learned data distribution of the three datasets were generated by the trained GANs. Furthermore, to reinforce the statement that auxiliary optimization does not destroy the internal structure of the generator, vector arithmetical operations in form of linear interpolation between latent vectors were successfully performed.

Outlook

Even tough the results of this thesis show that GANs can be beneficial for solving inverse problems, many interesting question still remain open. Since a well trained generator is an essential component in this framework, more complex architectures (such as ResNet[55]) and more sophisticated training methods (e.g. attention mechanisms [78]) that yield better generative results could be considered in future work. Additionally, further changes in the objective function of inverse problems such as content loss [79] or different regularizers (e.g. ℓ^1-regularization) for the generator parameters during auxiliary optimization could be considered. Further work could also elaborate on the effects of modifying only distinct layers of the generator during auxiliary optimization. Since the proposed approach allows the representation of unseen data samples on a low dimensional latent space, interesting opportunities in vector arithmetic image manipulation arise and should be studied in further work. Moreover, other inverse problems such as computed tomography reconstruction, re-coloring and blind image deconvolution could be addressed with the proposed method.

B Appendix

B.1 MNIST GAN Architecture

Generator architecture used for MNIST dataset

Optimizer: $Adam(lr = 1e\text{-}4, \beta_1 = 0.5, \beta_1 = 0.9)$, **Batch size**: 128; **Parameter initialization**: $\mathcal{N}(\mu = 0, \sigma = 0.05)$

	Layer 1	Layer 2	Layer 3	Layer 4
Type	Fully Connected	Transposed Convolution	Transposed Convolution	Transposed Convolution
Input Dimension	[1 × 128]	[4 × 4 × 256]	[8 × 8 × 128]	[16 × 16 × 64]
Output Dimension	[4 × 4 × 256]	[8 × 8 × 128]	[16 × 16 × 64]	[32 × 32 × 1]
Filter size	-	[4 × 4]	[4 × 4]	[4 × 4]
Stride	-	2	2	2
Padding	-	Same	Same	Same
Activation	ReLU	ReLU	ReLU	tanh
Batch Normalization	True	True	True	False
L2 Regularization	2.5*e*-5	2.5*e*-5	2.5*e*-5	2.5*e*-5

Table B.1: Detailed architecture of generator used for MNIST dataset.

Discriminator architecture used for MNIST dataset

Optimizer: $Adam(lr = 1e\text{-}4, \beta_1 = 0.5, \beta_1 = 0.9)$, **Batch size**: 128; **Parameter initialization**: $\mathcal{N}(\mu = 0, \sigma = 0.05)$

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
Type	Convolution	Convolution	Convolution	Fully Connected	Fully Connected
Input Dimension	[32 × 32× 3]	[16 × 16 × 64]	[8 × 8 × 128]	[4096 × 1]	[256 × 1]
Output Dimension	[16 × 16 × 64]	[8 × 8 × 128]	[4 × 4 × 256]	[256 × 1]	[1 × 1]
Filter size	[4 × 4]	[4 × 4]	[4 × 4]	-	-
Stride	2	2	2	-	-
Padding	Same	Same	Same	-	-
Activation	LReLU	LReLU	LReLU	LReLU	Linear
Batch Normalization	False	False	False	False	False
L2 Regularization	-	-	-	-	-

Extension for discriminator to InfoGAN

Extension: The output of layer 3 is given to fully connected layers that map the features to the correct dimensionality needed for InfoGAN. The layers $\tilde{4}$, $\tilde{5}$ appear alongside the standard layers 4,5.

	Layer 1 - Layer 3	Layer $\tilde{4}$	Layer $\tilde{5}$
Type	Convolution	Fully Connected	Fully Connected
Input Dimension	[32 × 32× 3]	[4096 × 1]	[256 × $(p + q)$]
Output Dimension	[4 × 4 × 256]	[256 × 1]	[$(p + q)$ × 1]
Activation	LReLU	LReLU	Softmax for p Linear for q
Batch Normalization	False	False	False
L2 Regularization	-	2.5e-5	2.5e-5

Table B.2: Detailed architecture of discriminator used for MNIST dataset.

B.2 Additional Results

Sparse-inpainting CelebA (10% of pixels given)

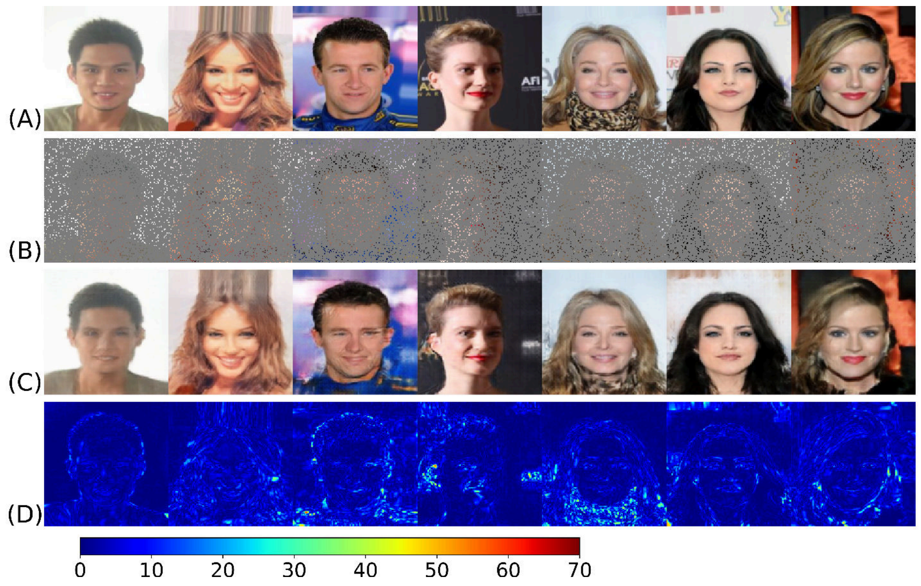

Figure B.1: Sparse-inpainting of CelebA data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

Eye region inpainting CelebA

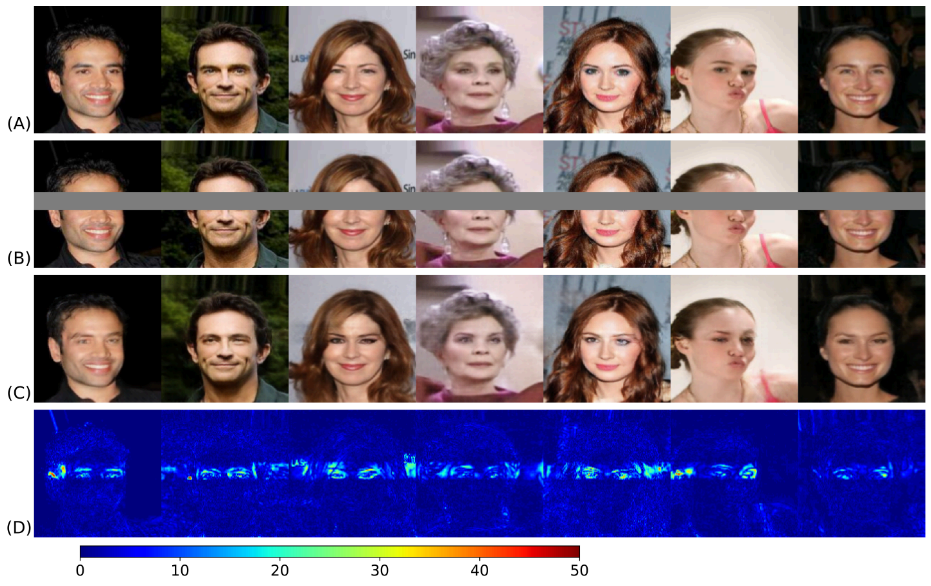

Figure B.2: Inpainting reconstruction of CelebA data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

Face-inpainting CelebA

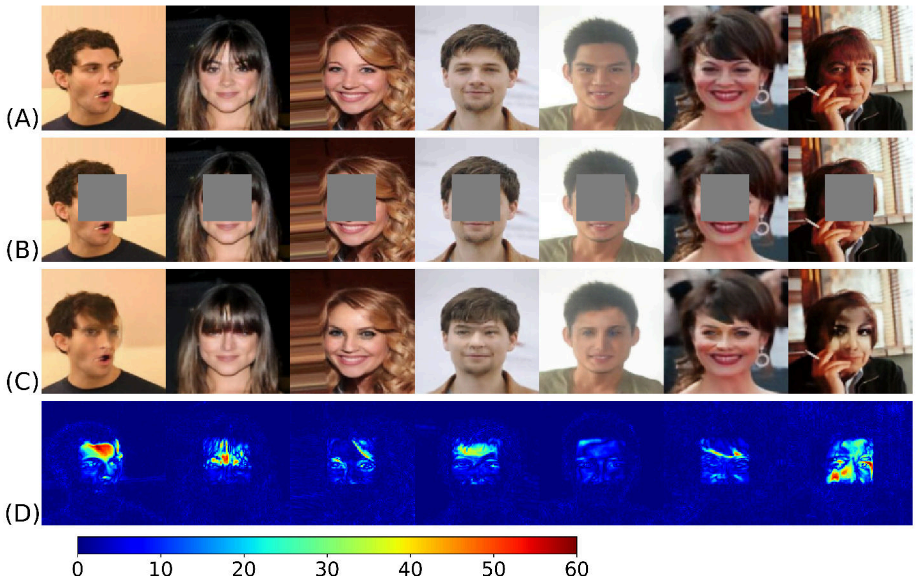

Figure B.3: Inpainting reconstruction of CelebA data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

Half image inpainting CelebA

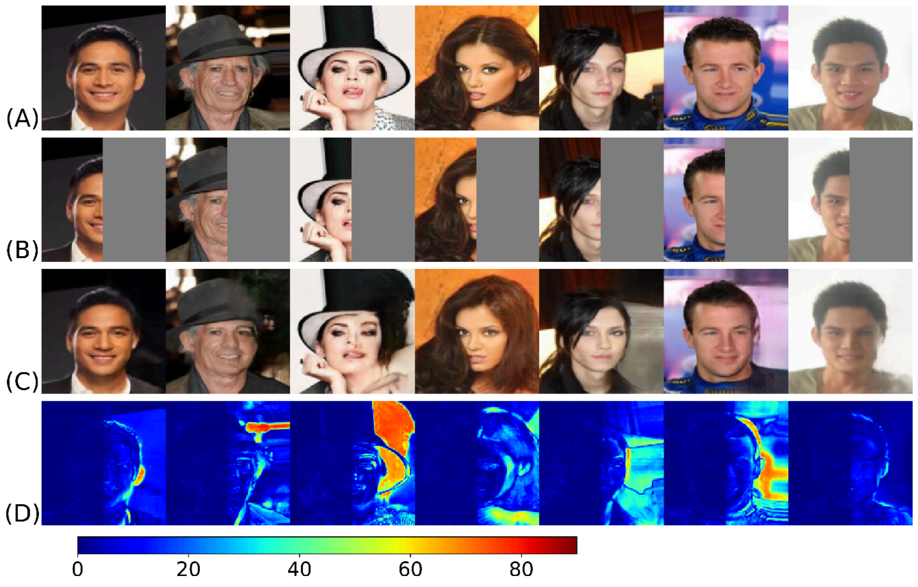

Figure B.4: Inpainting reconstruction of CelebA data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

4× Super-resolution CelebA

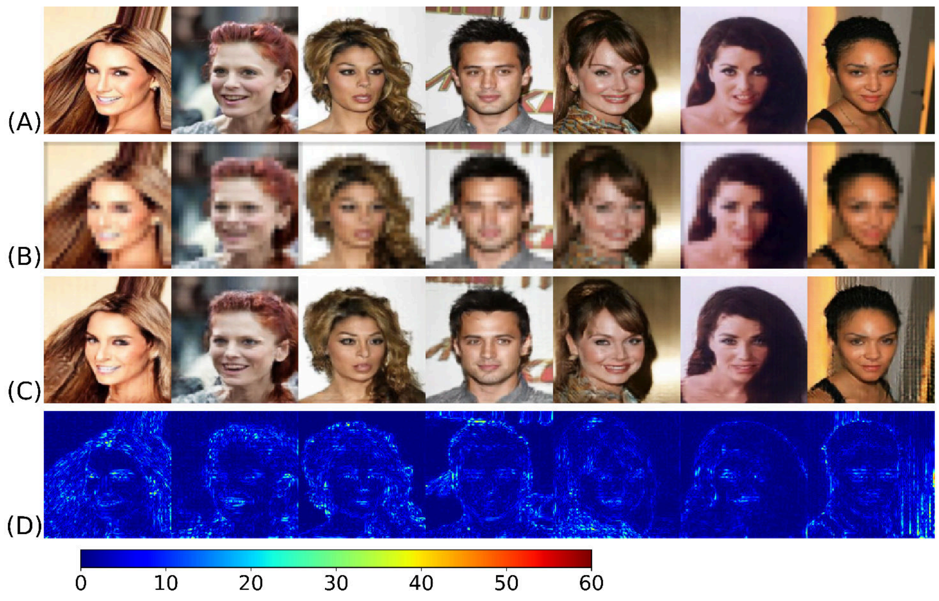

Figure B.5: Super-resolution of CelebA data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

Deblurring CelebA (Blur kernel: 15×15, $\sigma = 2$)

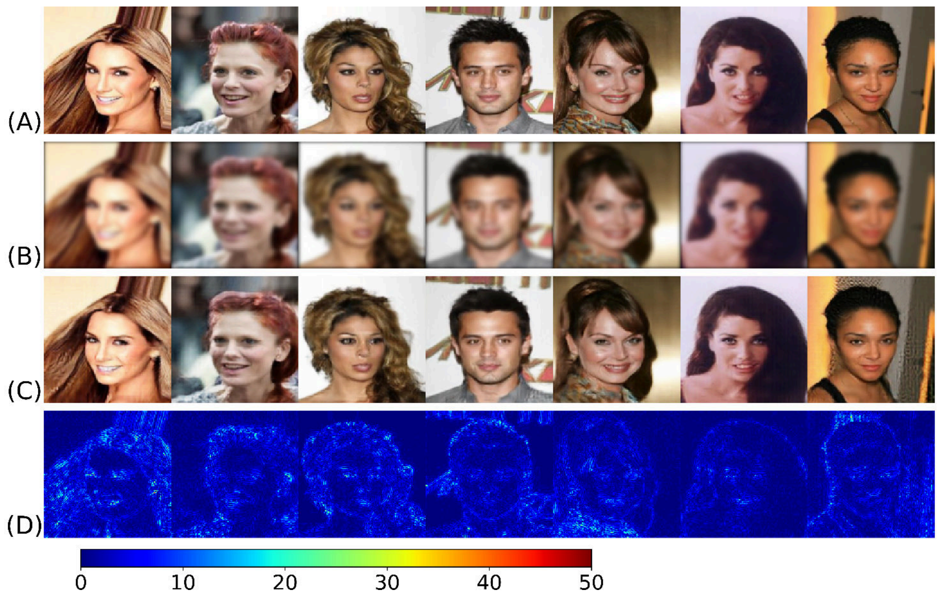

Figure B.6: Deblurring of CelebA data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

B.3 Failure Cases

Fourier space undersampling. Failed reconstructions for MNIST data.

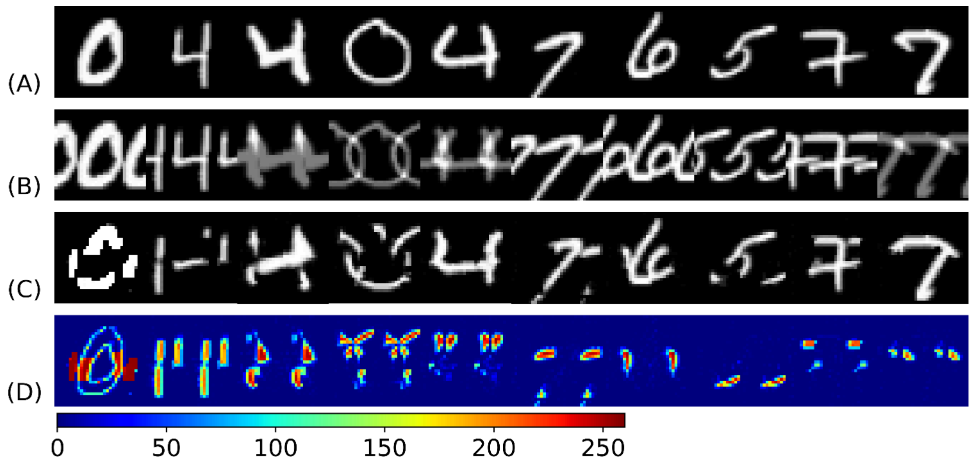

Figure B.7: Failed reconstructions for MNIST data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

Inpainting. Failed reconstructions for MRI data.

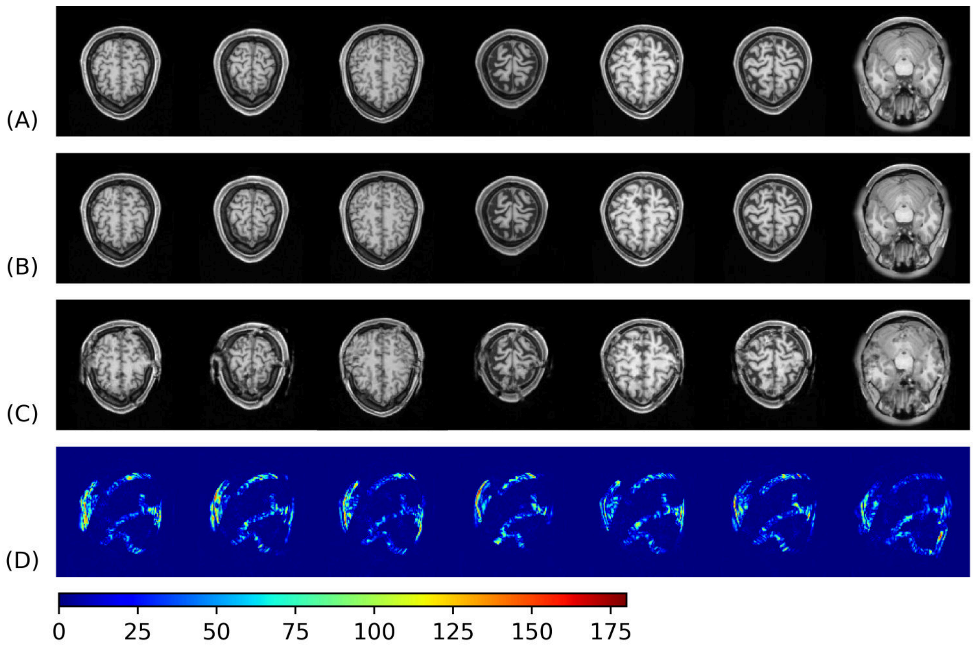

Figure B.8: Failed reconstructions for MRI data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

Denoising. Failed reconstructions for MRI data (5% Gaussian Noise).

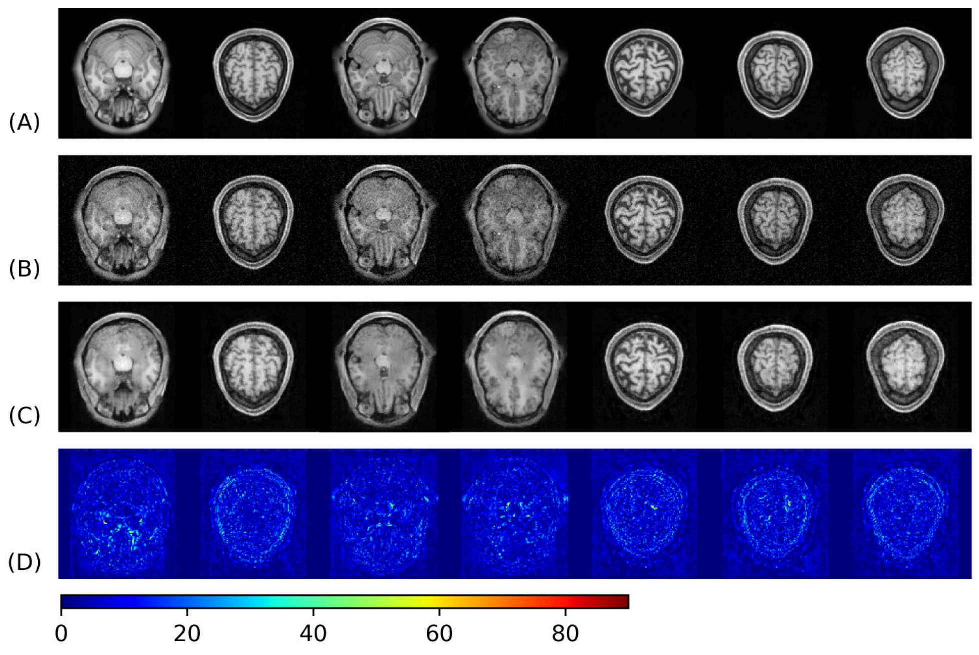

Figure B.9: Failed reconstructions for MRI data. (A) original data. (B) corrupted data. (C) reconstructed data. (D) absolute difference between (A) and (C).

Bibliography

[1] Rudin, L. I., Osher, S. & Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60, 259-268. (page 2)

[2] Anirudh, R., Thiagarajan, J. J., Kailkhura, B. & Bremer, T. (2018). An Unsupervised Approach to Solving Inverse Problems using Generative Adversarial Networks. *1805.07281.* (page 2)

[3] Keller, J. B. (1995). *Inverse Problems.* The American Mathematical Monthly, 83(2), 107-118. (page 5)

[4] Bertero M. & Boccacci P. (1998) *Introduction to Inverse Problems in Imaging.* U.K., Bristol:IOP. (page 6)

[5] Kabanikhin, S. (2008). Definitions and examples of inverse and ill-posed problems. *Journal of Inverse and Ill-posed Problems*, 16(4), 317-357. (page 6)

[6] Benning, M., & Burger, M. (2018). Modern regularization methods for inverse problems. *Acta Numerica*, 27, 1-111. (page 7)

[7] Tikhonov, A. N. (1963). Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4, 1035-1038. (page 7)

[8] Gravel, P., Beaudoin, G. & de Guise, J. A. (2004). A method for modeling noise in medical images. *IEEE Trans. Med. Imaging*, 23, 1221-1232. (page 8)

[9] Wang, R. & Tao, D. (2014). Recent Progress in Image Deblurring.*1409.6838.* (page 9)

[10] Yang, J., & Huang, T. S. (2017). Image super-resolution: Historical overview and future challenges. *Super-Resolution Imaging* (pp. 1-33). CRC Press (page 10)

[11] Vlaardingerbroek, M. T. & Boer, J. A. (1996) *Magnetic Resonance Imaging. Theory and Practice.* Springer Verlag: Germany. (page 11)

[12] Jacobs, M., Ibrahim, T. S. & Ouwerkerk, R. (2007). MR imaging: brief overview and emerging applications. *RadioGraphics*, 27(4), 1213-1229. (page 11)

[13] Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5), 717-727. (page 13)

[14] Lai, M. (2015). Deep Learning for Medical Image Segmentation. *arXiv preprint arXiv:1505.02000.* (page 13, 17, 25)

[15] Kandel, E., Schwartz, J. & Jessel, T. (2000) *Principles of Neural Science.* New York: McGraw-Hill. (page 13)

[16] Forstmann, B., Keuken M. & Alkemade, A. (2015). *An Introduction to Human Brain Anatomy. In: Forstmann B., Wagenmakers EJ. (eds) An Introduction to Model-Based Cognitive Neuroscience.* New York, NY: Springer. (page 13)

[17] Krenker, A., Beŝter, J. & Kos, A. (2011). *Introduction to the Artificial Neural Networks. In: Suzuki, K. (eds) Artificial neural Networks* INTECH Open Access Publisher. (page 13)

[18] Akgün, E., & Demir, M. (2018). Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks. *International Journal of Assessment Tools in Education*, 5(3), 491-509. (page 13)

[19] Marblestone, A. H., Wayne, G. & Körding, K. P. (2016). Toward an Integration of Deep Learning and Neuroscience. *Front. Comput. Neurosci. 10(94).* (page 14)

[20] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65, 386-408. (page 14)

[21] Minski, M.L. & Papert, S. A. (1969) *Perceptrons: an introduction to computational geometry.* Cambridge: MIT Press. (page 15)

[22] Zilouchian, A.; (2001). *Fundamentals of neural networks. In: Jamshidi M, editor. Intelligent control systems using soft computing methodologies.* CRC Press. (page 15)

[23] Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning.* MIT Press. (page 15, 23, 26, 30, 32)

[24] Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2, 356-366. (page 15)

[25] Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6, 861-867. (page 15)

[26] Bishop, Christopher M. (2006). *Pattern recognition and machine learning.* New York: Springer. (page 18, 23, 30)

[27] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533. (page 19)

[28] In Zhou, S. K., In Greenspan, H., & In Shen, D. (2017). *Deep learning for medical image analysis.* Academic Press. (page 20, 25)

[29] Polyak, B.T. (1964) Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4, 1-17 (page 21)

[30] Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence O(1/k2). *Doklady ANSSSR (translated as Soviet.Math.Docl.)*, 269, 543-547 (page 22)

[31] Sutskever, I., Martens, J., Dahl, G. E. & Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. *ICML*, 3, 1139-1147 (page 22)

[32] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arxiv:1412.6980.* (page 22)

[33] Duchi, J. C., Hazan, E. & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12, 2121-2159. (page 22)

[34] Tieleman, T. & Hinton, G. (2012). Lecture 6.5 - RMSProp,*COURSERA: Neural Networks for Machine Learning.* Technical report. (page 22)

[35] O'Shea K.T. & Nash R. (2015). An Introduction to Convolutional Neural Networks. *ArXiv e-prints 1511.08458.* (page 23)

[36] Lecun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks MIT Press.* (page 23)

[37] LeCun, Y., Bengio, Y. & Hinton, G. E. (2015). Deep learning. *Nature*, 521, 436-444. (page 24, 25)

[38] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1, 541-551. (page 25)

[39] Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* (p./pp. 1097-1105) (page 25)

[40] Long, J., Shelhamer, E. & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* (page 25)

[41] Maas, A.L., Hannun, A.Y. & Ng, A.Y.(2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 30, 13-14. (page 26, 28)

[42] Bertsekas, D. (1999). *Nonlinear Programming.* Athena Scientific. (page 26)

[43] Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5, 157-166. (page 27)

[44] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, Proceedings of the 32nd International Conference on Machine Learning (ICML), volume 37 of *Proceedings of Machine Learning Research*, pages 448-456. PMLR. (page 29, 30)

[45] Bjorck, J., Gomes, C. P. & Selman, B. (2018). *Understanding Batch Normalization.* CoRR, abs/1806.02375. (page 29)

[46] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press. (page 31)

[47] Kukacka, J., Golkov, V. & Cremers, D. (2017). Regularization for Deep Learning: A Taxonomy. *CoRR, abs/1710.10686.* (page 31)

[48] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. & Bharath, A. (2017). Generative Adversarial Networks: An Overview. *arxiv:1710.07035.* (page 34, 38)

[49] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, 27, 2672-2680 (page 34, 35, 36, 37, 40, 41)

[50] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. & Lee, H. (2016). Generative adversarial text to image synthesis. *arxiv:1605.05396.* (page 34)

[51] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J. & Catanzaro, B. (2017). High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *1711.11585.* (page 34)

[52] Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *1703.10593.* (page 34)

[53] Ledig, C., Theis, L., Huszar, F., Caballero, J., Aitken, A. P., Tejani, A., Totz, J., Wang, Z. & Shi, W. (2016). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.*arxiv:1609.04802.* (page 34, 48, 49)

[54] Klenke, A. (2014). *Probability theory: a comprehensive course.* Springer-Verlag, London (page 34)

[55] Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein GAN. *arxiv:1701.07875.* (page 37, 39, 40, 41, 89, 92)

[56] Radford, A., Metz, L. & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434.* (page 38, 39, 58, 89)

[57] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. (2017). Improved Training of Wasserstein GANs. *arXiv:1704.00028.* (page 41, 42, 63)

[58] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. & Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *arxiv:1606.03657.* (page 43, 58, 89)

[59] Barber, D. & Agakov, F. V. (2003). The IM Algorithm: A Variational Approach to Information Maximization. *NIPS* (p./pp. 201-208) (page)

[60] Spurr, A., Aksan, E. & Hilliges, O. (2017). Guiding InfoGAN with Semi-Supervision.*arxiv:1707.04487.* (page 44)

[61] Lipton, Z. C. & Tripathi, S. (2017). Precise Recovery of Latent Vectors from Generative Adversarial Networks.*arxiv:1702.04782.* (page 44, 50)

[62] Creswell, A. & Bharath, A. A. (2018). Inverting The Generator Of A Generative Adversarial Network (II). *arxiv:1802.05701.* (page 45)

[63] Iizuka, S., Simo-Serra, E. & Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Trans. Graph., 36, 107:1-107:14.* (page 48)

[64] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X. & Huang, T. S. (2018). Generative Image Inpainting with Contextual Attention.*arxiv:1801.07892.* (page 48)

[65] He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arxiv:1512.03385.* (page 48)

[66] Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D. & Matas, J. (2017). DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks.*arxiv:1711.07064.* (page 49)

[67] Yang, G., Yu, S., Dong, H., Slabaugh, G. G., Dragotti, P. L., Ye, X., Liu, F., Arridge, S. R., Keegan, J., Guo, Y. & Firmin, D. N. (2018). DAGAN: Deep De-Aliasing Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction. *IEEE Trans. Med. Imaging*, 37, 1310-1321. (page 49)

[68] Ronneberger, O., Fischer, P. & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation.*arxiv:1505.04597.* (page 49)

[69] Bora, A., Jalal, A., Price, E. & Dimakis, A. G. (2017). Compressed Sensing using Generative Models.*arxiv:1703.03208.* (page 49, 50, 85)

[70] Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective: Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73, 273-282. (page 49)

[71] Tripathi, S., Lipton, Z. C. & Nguyen, T. Q. (2018). Correction by Projection: Denoising Images with Generative Adversarial Networks. *arxiv:1803.04477.* (page 50)

[72] Dabov, K., Foi, A., Katkovnik, V. & Egiazarian, K. O. (2007). Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Trans. Image Processing*, 16, 2080-2095. (page 50, 69)

[73] Yeh, R. A., Chen, C., Lim, T.-Y., Schwing, A. G., Hasegawa-Johnson, M. & Do, M. N. (2017). Semantic Image Inpainting with Deep Generative Models. *CVPR* (p./pp. 6882-6890) (page 50)

[74] Pérez, P., Gangnet, M. & Blake, A. (2003). Poisson image editing. *ACM Trans. Graph., 22, 313-318.* (page 50)

[75] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. & others (2016). TensorFlow: A System for Large-Scale Machine Learning.*OSDI (p./pp. 265-283).* (page 63)

[76] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* (p./pp. 2278-2324). (page 64)

[77] Yang, S., Luo, P., Loy, C. C. & Tang, X. (2015). From Facial Parts Responses to Face Detection: A Deep Learning Approach. *ICCV* (p./pp. 3676-3684). (page 64)

[78] Zhang, H., Goodfellow, I. J., Metaxas, D. N. & Odena, A. (2018). Self-Attention Generative Adversarial Networks. *arxiv:1805.08318.* (page 92)

[79] Johnson, J., Alahi, A. & Li, F.-F. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. textitarxiv:1603.08155. (page 92)