Universiteit Leiden

Master Computer Science

Exploratory Analysis of Enforcing Class-Specific Features in Image Reconstruction through Network Inversion Methods.

Name:	Jan Żuromski
Student ID:	s3986403
Date:	25/07/2025
Specialisation:	Artificial Intelligence
1st supervisor:	Kees Joost Batenburg
2nd supervisor:	Daniël M. Pelt
3rd supervisor:	Jiayang Shi

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Chapter 1

Introduction

In the digital era, image reconstruction has emerged as a massive field of study that is crucial when one wants to perform a CT scan, obtain a high resolution image from a satellite or perform data restoration for forensics. Image reconstruction allows us to perform a seemingly miraculous act - recreate an high-quality image from degraded or incomplete data, often with incredible levels of accuracy. However, at the heart of image reconstruction lies a problem. If the data we have is incomplete, there are vast numbers of possible ways to impute the unknown contents we're looking for. Depending on how we choose to form said data, the outcomes of our reconstructions can vary significantly. Problems like this - ones with a multitude of possible solutions and a high sensitivity to changes in initial conditions - are referred to as *ill-posed*. We can combat this ill-posedness by introducing additional constraints on image reconstruction schemes. In fact, over time a lot of different solutions of constraining the problem have emerged with varying degrees of effectiveness. A vast majority of those methods focus on modelling the general characteristics of real-life images - they aim to filter out high-frequency noise or to obtain smooth transitions between neighbouring pixels or to retain crisp edges of objects. These methods, while admittedly very effective, generalize the issue of image reconstruction, neglecting the fact that different domains of images may be characterised by slightly different features that need to be accentuated in order to obtain realistic reconstructions. What if we know that the CT scan we try to obtain is an image of a brain and not of the chest? What if we know that the satellite took pictures of a dense city landscape and not of a village. What if the face we're trying to reconstruct is a phenotypical male? Could we include said information into the reconstruction process and would that in fact improve the quality of our resulting images? We feel that this area of image reconstruction has been so far relatively under-explored. In this thesis, we set out on an exploration of what it would exactly mean to input prior class into an image restoration method and what complexities await when we try to do so. Rather then presenting a single method that 'works' in a given context, we decided to focus on describing the problem in detail - we aim to offer understanding more than solutions. With

this in mind, we present below the flow of our thesis and the story that we're trying to tell.

1.1 Flow of the thesis

The overarching goal of this thesis is the exploration the complexity of using network inversion to enforce class-specific characteristic on image reconstructions. We formulated this report to reflect the natural problem-solving flow of the issue. We start with the most general implementation of network inversion and then progress into different aspects of its intricacies. Finally, we apply our findings to a specific use case scenario in order to provide a more tangible, quantitative evaluation of our most promising approaches. The bird's eye view of the flow of this thesis can be seen on Figure 1.1.

1.1.1 Part 1: Network Inversion

In the first part of this thesis, we focus on the crux of the issue - the network inversion. The goal of this operation, explained in detail in Chapter 2, is to extract dataset information gathered by a network during training. We explore the different basic building blocks that are required to successfully and reliably extract this information:

- Convolutional Neural Network (CNN): we limit our investigations to classification problems and use CNNs as our main model. We opted to use a lightweight network approach where we use the simplest architecture that obtains satisfactory performance on a given task.
- Image prior: obtaining realistic reconstructions from networks requires a strong prior. Based on literature review, we opted for a combination of total variation and ℓ_2-norm regularization.
- The choice of optimizer and hyperparameters: while in many common problems, the choice of optimizer has little effect on resulting performance of a network, this was not true in our case. We found that the type of optimizer is an important consideration when performing network inversion. This adds to the pool of at least 5 other hyperparameters that need to be optimized which presents a considerable challenge that needs to be addressed.

1.1.2 Part 2: Increasing Robustness

After performing the exploratory phase of Part 1, we focus on a big challenge of network inversion - how to force a network to learn a representation of the problem that not only corresponds to good classification performance but also to a greater susceptibility to inversion. More specifically, how do we train a classifier so that it's internal representation is aligned with visual qualities of

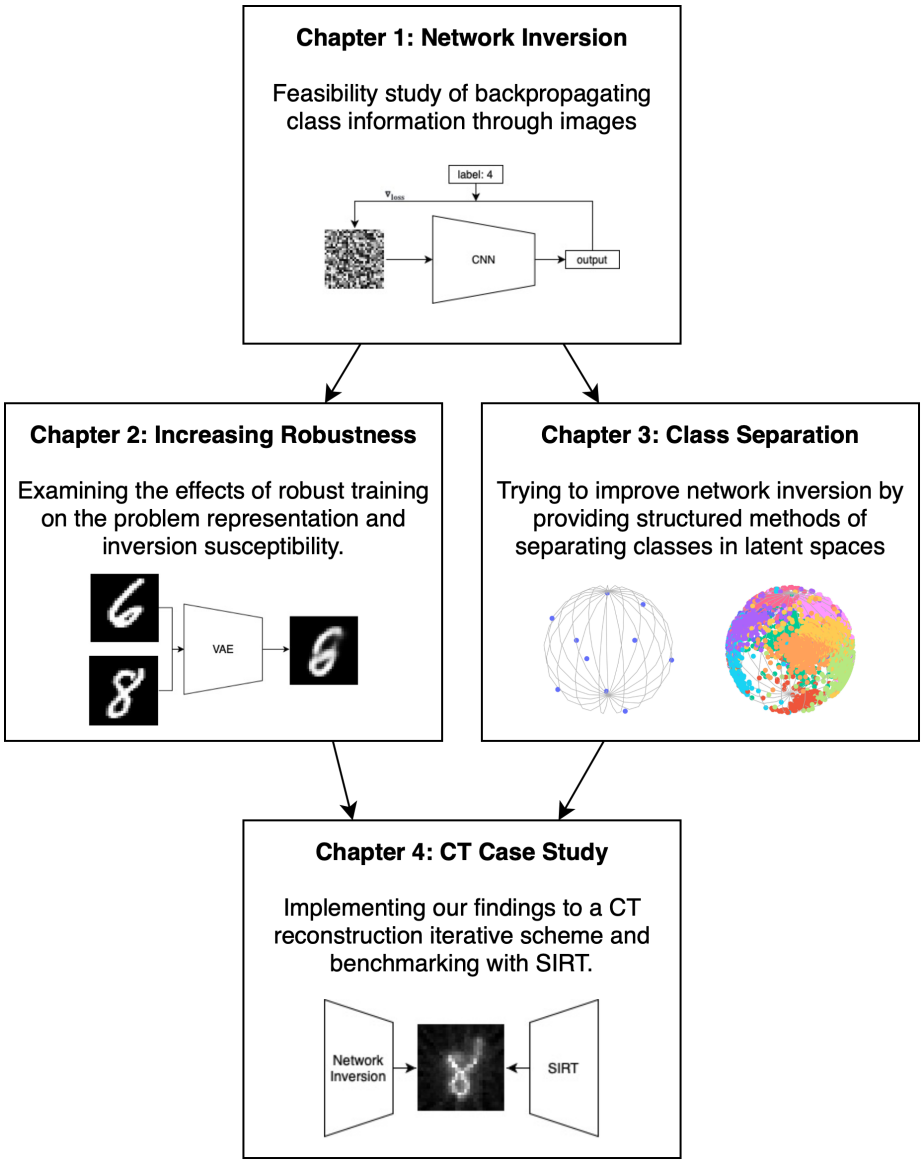


Figure 1.1: A bird's eye view on the flow of this thesis.

the images and classes. To that end, we turned our attention to robustness. In classic training contexts, robustness allows networks to better handle examples that are outside of the domain of the training dataset or examples that aim to fool the network into taking wrong decisions. In simplified terms, robustness means teaching the network to recognise what is and what isn't a valid class image and providing a smoother representation that's more stable to subtle changes. We decided to explore two approaches that aim to increase robustness: FGSM adversarial training and intermediate class-enriched datasets.

Fast Gradient Sign Method (FGSM) Following the observation on the relationship between robustness and susceptibility to inversion by Mejia et al. [21], we implemented the Fast Gradient Sign Method (FGSM) devised by Goodfellow et al. [10] to robustly train our network. The main idea behind using this approach is that presenting the network with examples that provide small perturbations in the network's resulting gradients makes the network more stable to noise and results in a model that is 'more confident' of the definitions of the classes. This should in turn result in more realistic reconstructions of our images.

Intermediate Class Examples Following the same idea of increasing robustness, we decided to test another approach - introducing intermediate class examples to our datasets. The aim here was to force network to learn a smoother representation by explicitly defining the traversal between classes. This was achieved by training a Variational Auto-Encoder (VAE) on the target training dataset and obtaining the intermediate class examples by means of weighted linear interpolation. As an externality, the labels became then distributions over the target classes rather than single numbers and the loss was calculated as the difference between the output and target distributions.

1.1.3 Part 3: Class Separation

One of the conclusions of the previous two parts of the thesis was the fact that class representations might be ill-separated in the latent space of networks. We hypothesised that improving this separation might prove beneficial in providing better reconstructions under network inversion. We investigated this claim by introducing the concept of prototyping under different geometries. We opted for two different approaches in this section: (1) hyperspherical prototypes and (2) hyperbolic prototypes with Busemann loss. These two approaches were chosen to explore how the classes can be separated under spaces with different curvatures and to understand whether forcing networks to learn mappings to fixed class prototypes rather than straightforward representations would improve their inversion capabilities.

1.2 The Evaluation Problem

Image reconstruction is a widely studied problem with a multitude of applications which leads to multiple evaluation schemes dependent on the use context. Ideally, our approaches should be evaluated in a quantitative manner and compare with state-of-the-art benchmarks on popular, challenging datasets. However, our interest in the problem makes the evaluation more elusive and best expressed in a qualitative manner. We are interested in the general ability of convolutional neural networks to store and transmit class-specific information. This means that we do not care about the variety of reconstructed images but rather in a static, singular representation of a whole class. Resulting from this is the following challenge: how to quantitatively evaluate an approach which results in a handful of images and what exactly should we compare the results to? To investigate this challenge, we present a brief review of the common evaluation methods in the general field of image reconstruction and discuss their utility in the context of our research. Before we do that, we start of this discussion with a brief introduction of three common quantitative metrics for image reconstruction as they are a recurring theme in the literature review part. Finally, we present our qualitative evaluation approach and then offer a specific use case in the field of Computer Tomography image reconstruction that will allow us to more comprehensively evaluate the method that resulted from our investigations.

1.2.1 Image reconstruction quantitative evaluation metrics

In order to properly discuss evaluation methods in the field of image reconstruction, one has to start with 3 popular pixel-wise quantitative metrics: mean squared error, peak signal-to-noise ratio and structural similarity index. All three metrics aim to describe differences between images but do so in different ways therefore commonly they are used at the same time by researchers to provide more comprehensible evaluations.

Mean Squared Error (MSE) This metric, commonly used as a loss function, is the most straightforward of the metrics chosen in this thesis. It measures the difference between the obtained image and the ground truth through pixel-wise comparison, putting more weight to substantial differences. For two images of equal dimensions $m \times n$, MSE is defined as follows:

$$\mathrm{MSE}(X, Y) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [X_{ij} - Y_{ij}]^2 \tag{1.1}$$

MSE is a handy tool for describing pixel-wise differences between two images and is commonly used as a loss function in teaching models. However, it can be often misleading as it does not capture perceived differences/similarities. Com-

monly, comparing visually similar images can yield substantial Mean Squared Error values - e.g., if their luminance values differ.

Peak Signal-to-Noise Ratio (PSNR) Peak Signal-to-Noise Ratio is a metric commonly used in assessing the quality of reconstructed images or videos, e.g., when using lossy compression algorithms. It is defined as a ratio than the strongest possible signal in an image and the amount of noise that is introduced by the reconstruction process; the amount of noise is expressed as Mean Squared Error (MSE).

$$\mathrm{PSNR}(X,Y) = 10\log_{10}\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}(X,Y)} \tag{1.2}$$

In the above equation, MAX_I refers to the maximum pixel intensity value (equal to 2^n-1 given n bits per pixel). PSNR is expressed in decibels (dB) with higher values corresponding to better reconstruction quality. PSNR is bounded by 0 as minimum and its maximum bound is determined by the maximum intensity value (for 8 bits of data per pixel the maximum PSNR is 48.13 dB). While PSNR is a widely used metric for benchmarking reconstruction algorithms and offers satisfactory evaluation quality in most cases, it does not correspond well to human perceived quality.

Structural Similarity Index Measure (SSIM) Introduced in 2004, SSIM has gained tremendous popularity for its ability to quantify the differences in visual qualities of two images in a way that corresponds to human perception. SSIM's philosophy is based on comparing perceived changes in structural information [32]. This is achieved by comparing three distinct aspects of the images:

- Local luminance (μ_X), defined as the mean of the pixel intensities across an image:

$$\mu_X = \frac{1}{mn}\sum_{i=0}^{m}\sum_{j=0}^{n} X_{ij} \tag{1.3}$$

  The luminances of the two images are then compared as follows:

$$l(X,Y) = \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1} \tag{1.4}$$

  Here, C_1 is a small constant that is introduced for numerical stability (i.e., to avoid division by 0) and is defined as the squared product of the pixel intensity range for the two images and a small positive constant $k_1 \ll 1$ ($C_1 = (k_1 L)^2$). As a result luminance is a metric in the range $(0;1\rangle$, where higher values correspond to more similar luminances.

- Local contrast (σ_X), defined as the standard deviation of the intensities across the image:

$$\sigma_X = \frac{1}{mn}\sum_{i=0}^{m}\sum_{j=0}^{n}(X_{ij} - \mu(X)) \tag{1.5}$$

Contrasts of the two images are then compared in a manner analogous to luminances:

$$c(X,Y) = \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2} \tag{1.6}$$

Again, a constant C_2 is applied for numerical stability, formulated as $C_2 = k_2 L$ with k_2 being a positive constant $k_2 \ll 1$ and L being the pixel intensity range for the two images.

- Structural similarity of the two images, which is expressed using the correlation coefficient of the pixel intensities:

$$s(X,Y) = \frac{\sigma_{XY} + C_3}{\sigma_X\sigma_Y + C_3} \tag{1.7}$$

Where σ_{XY} is calculated as:

$$\sigma_{XY} = \frac{1}{mn-1}\sum_{i=0}^{m}\sum_{j=0}^{n}[(X_{ij} - \mu_X)(Y_{ij} - \mu_Y)] \tag{1.8}$$

The constant C_3 is analogous to C_2 and C_1.

Finally, the Structural Similarity index measure combines the three qualities into a single measure:

$$\text{SSIM}(X,Y) = l(X,Y)^\alpha \cdot c(X,Y)^\beta \cdot s(X,Y)^\gamma \tag{1.9}$$

Hyperparameters α, β and γ allow to adapt the relative importance of the different measures. In this thesis, all three measures were weighted equally ($\alpha = \beta = \gamma = 1$).

1.2.2 Image Reconstruction Evaluation Methods in Literature

We can consider our approaches within two different areas of study - image reconstruction and network inversion. The two fields differ significantly in their evaluation schemes with image reconstruction having more established and unified methods of evaluation. In most cases of image reconstruction, the overall goal is to restore some distorted or incomplete input into a high quality ground truth image. This can be exemplified by areas like image de-noising [5, 9], image in-painting [11, 33] and super-resolution [31, 34] that all fall into this category. Evaluating such approaches is quite straightforward: given a batch of ground truth images, distort them accordingly to the task at hand and try to restore them so that they resemble the ground truth image as closely as possible. In order to properly capture differences that are important for the given tasks, appropriate metrics have to be leveraged. Popularly used metrics include MSE, PSNR and SSIM, all of which are described in Subsection 1.2.1. Some of the insights from these approaches are relevant to our investigations: we want to

promote realistic reconstructions that preserve edges well and provide pixel-wise correlation. Having said that, we want to examine a more general capability of the network inversion to reconstruct class-specific images. In other words we do not care if an individual image is identical to its ground truth but rather whether the network can transmit information that makes the image 'look more like' a given class. In the field of network inversion on other hand, the evaluation tasks tend to have a larger variety and depend on the downstream task. Some of those evaluations can be quantitative, like counting the number of the training dataset images that can be recovered from a network [13] or reconstructing an image based on the activations said image provokes in a network [35]. Given however that we are not interested in specific instances of images, neither of these approaches would be adequate in our case. More aligned with our work are qualitative methods in the field of network inversion, such as the work of Mordvintsev et al. [24] or Mahendran et al. [20] who recover images based on gradients in different layers of networks. We adapt their approaches and present our evaluation scheme described in the next section.

1.2.3 Our Evaluation Approach

In order to maintain integrity of the project and establish means of comparison between different approaches, we decided to establish an evaluation pipeline that would compare implementations in a fair way that was relevant to the research question. The initial, exploratory stage involved testing the feasibility of the approach and the potential strength of the signal. This step, conducted on the MNIST dataset, consisted of two parts:

1. Transforming noise to meaningful images through label-guided optimization - a batch of 10x 28x28 px images of gaussian noise were passed to a tested model along with a batch of 10 labels (0 through 9). The resulting images were evaluated based on the visual properties.

2. Transforming an image of a digit into an image of a different label - images were provided along with labels that corresponding to different classes than the original images. The models were then tasked with transforming the image towards the new labels. The resulting images were evaluated based on the visual properties.

This method of evaluation allows us to compare efficacy of inverted models in transferring class knowledge in different contexts. As we will see in later chapters, this method allowed us to discriminate between approaches in a satisfactory manner. In order to comprehensively compare models that fared similarly well, we developed a second evaluation framework which was aimed at evaluating models in the context of the task laid out in the research question: improving reconstruction quality through the combination of a model-based and data-based approach. The framework was constructed:

1. Forward projections of test images were obtained under specified conditions (projection geometry type, number of angles, number and spacing

of detector pixels, etc.),

2. The resulting sinograms were passed through 50 iterations of SIRT in order to obtain initial reconstructions.

3. The initial reconstructions subsequently underwent 50 additional iterations of alternating transformations - SIRT and the tested model step - in a ratio of 3:1.

4. The final reconstructions were compared with the ground truth images using MSE, PSNR, and SSIM. The results were benchmarked against a reconstruction provided by 100 of SIRT only.

Chapter 2

Network Inversion

2.1 Image Reconstruction

2.1.1 The Inverse Problem

There are many reasons why one might want to reconstruct images. Obtaining a readable image of a CT scan, trying to understand the inner workings of a Convolutional Neural Networks or reducing the amount of noise in a compressed image might seem like very different problems but they all fundamentally ask the same question: given the result of applying a forward process, how can we faithfully reconstruct the given input? Mathematically, the forward process can be represented as follows:

$$\hat{y} = Ax_{\text{input}} + \eta \tag{2.1}$$

In many real-world applications recovering x_{input} doesn't always have a closed-form solution. Therefore, the problem of inverting the forward process is often defined as a least-squared solution:

$$x_{\text{input}} = \arg\min_{x} \|Ax_{\text{input}} - \hat{y}\|_2^2 \tag{2.2}$$

Reformulating the inverse problem as a least-squared problem allows us to approach it using minimisation methods but in its naïve form the obtained solutions would often be unstable and, in many cases, would lead to overfitting to training data. The reconstructions would not be stable to small changes in data, violating the Hadamard's third criterion [12] for a well-posed problem. In order to combat this, we will add a regularization term $\mathcal{R}(x)$ to our solution:

$$\hat{x}_{\text{input}} = \arg\min_{x}(\|Ax - \hat{y}\|_2^2 + \gamma\mathcal{R}(x)) \tag{2.3}$$

2.1.2 Image Prior

When dealing with image reconstruction, a consequential hurdle emerges - the space of potential images is vast and without additional information, many

algorithms struggle to provide a satisfactory result. However, only a small fraction of all possible images are representative of what can be encountered in real-life problems. These realistic images tend to have some very general characteristics. The regularization term can be understood as an image prior - it helps us determine what we expect from real-life images (in the general sense or in the context of specific tasks). By defining the prior, we're limiting the space of possible image reconstructions and accentuate their expected characteristics. There are numerous ways of defining the image prior, with the most popular being ℓ_2-norm regularization and Total Variation Regularization.

ℓ_2-norm regularization The least squares approach to the inverse problem tends to over-amplify noise, leading to unrealistic reconstructions. ℓ_2-norm regularization is a special case of Tikhonov regularization (with an identity matrix chosen as the Tikhonov matrix) which penalises large pixel intensity values across the whole image. This achieves two things: (1) it helps to maintain the pixel intensity values within reasonable ranges that are more likely to correspond to ground truth images and (2) prevents oversaturation of the reconstruction. The regularization term is simply defined as the Euclidean norm of the pixel intensities:

$$\|x\|_2^2 = \sum_{i=0}^{n} |x_i|^2 \tag{2.4}$$

Total Variation (TV) regularization TV regularization handles another characteristic of real-life images: neighbouring pixels tend to be correlated. By penalising differences in intensities between neighbouring pixels, TV regularization enforces smoother images with lower entropy. We define it as follows:

$$TV(x) = \sum_{i=0}^{m} \sum_{j=0}^{n} ((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2) \tag{2.5}$$

2.2 Neural Networks for Visual Tasks

While the inverse problem can apply to many different forward operators, we will be focusing on one specific case: the Convolutional Neural Networks (CNNs). In order to interpret images, CNNs leverage two operations: convolutional filtering and pooling [17]. Discrete convolutions are used to extract features from images. By using trainable filters, the network learns to extract generalizable patterns from data - starting with low-level features like lines or basic shapes to more advanced characteristics as we progress deeper into the architecture. The features are then pooled after each convolutional layer, e.g., using a max operation across a small neighbourhood of pixels. This introduces invariance towards minute differences between patterns and decreases the dimensionality

of the image, allowing for better generalisability and faster processing. An example of a CNN is presented in Figure 2.1. An image is passed through several blocks of convolutions, non-linearities (like ReLU) and pooling layers, with the dimensionality of the features being decreased as the image progresses through the network. Finally, the resulting feature vector is passed through a series of fully connected layers which perform the end-point task, e.g., classification or object detection. This powerful technique has found use in a variety of complex tasks like facial expressions' analysis, automated driving, feature extraction or improving computer tomography reconstructions.

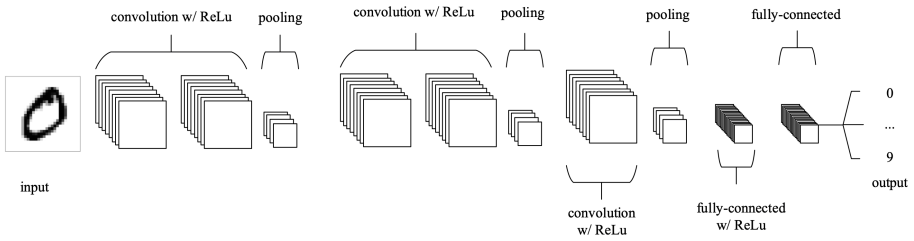


Figure 2.1: CNNs parse images by applying a series of convolutions before processing the resulting features through fully connected layers. An example of a CNN architecture from O'Shea et al. [25].

2.3 Inverting the Neural Network

In this thesis, we focus on the following task: what are the important complexities one has to take into account when trying to apply the inversion problem to convolutional neural networks. This brings us to the crux of the project: inverting the neural network. Mathematically, we can formulate network inversion by rephrasing the least squares solution to the inversion problem, presented in Equation 2.2. We now formulate our forward operator as a CNN $\mathcal{F}$ parametrised by its weights ϕ. We also substitute the least squares for a more general loss function L. This results in the following equation:

$$x^* = \arg\min_x [L(\mathcal{F}_\phi(x), \hat{y}) + \mathcal{R}(x)] \tag{2.6}$$

2.3.1 Network Inversion in Related Works

Network inversion emerges in several domains of AI research, including adversarial attacks, explainable AI [19] and knowledge distillation [35]. In the context of this thesis, we are focusing on its applications to Convolutional Neural Networks. In a 2015 'Inceptionism: Going Deeper into Neural Networks' blogpost, Mordvintsev et al. inversed a Convolutional Neural Network to reconstruct images from noise based on specific classification labels [24]. They underscored the

importance of a strong prior, opting themselves for a combination of ℓ_2-norm and total variation regularization. Their results showed promise in terms of enforcing some characteristics of labels onto images. However, the resulting images reflected rather the general 'essence' of classes and would not be suitable for our task. We take the takeaways of the researchers' work in the context of emphasising a strong prior and apply it in our algorithm to obtain better image reconstructions but focus on realistic reconstructions rather than artistic quality.

Focusing on the field of knowledge distillation without accessing the training datasets, Yin et al. inversed a teacher network to generate examples for a smaller student network [35]. They achieved reconstructions of high fidelity by extending the prior by feature distribution regularization. In order to avoid accessing the datasets, the researchers leveraged the batch normalization layers which in many commonly used libraries store running average feature statistics. In this thesis, we applied the prior used by aforementioned researchers to obtain reconstructions in the context of CT image reconstruction.

Mahendran et al. [19] leveraged network inversion to compare its efficacy in the field of AI explainability with representations like Histogram of Oriented Gradients (HOG) and Scale-Invariant Feature Transform (SIFT). Additionally, they examined the relationship between the layer level at which the reconstruction process started and the quality of resulting reconstructions. They especially focused on obtaining series of reconstructions from each representation, with the aim of getting more information about how the networks interpret images. We applied similar method of optimisation as the researchers and analysed their work on obtaining reconstructions from each layer of the network to inform our selection of network architectures but focused on obtaining single realistic reconstructions rather than series of snapshots that represent the networks' 'thinking'.

2.4 Approach

We started our investigations with a simple proof of concept. We implemented a relatively small convolutional neural network and trained it to classify digits from the MNIST dataset. We used two version of the training scheme: in the first scheme, The network was designed to return logits for each of the 10 possible digits. The resulting network was then used in our experimental pipeline that consisted of two tasks.

Task 1: Optimizing noise into images of digits In this task we tested the ability of feasibility of optimizing noise into an image of a digit using the just the information contained within the network. A batch of ten tiles of size 28x28 each (the size of an image in the MNIST dataset) was classified by the network. Then a loss comprising of cross entropy and our regularization term (a weighted sum of total variation and ℓ_2-norm) was backpropagated through

the noise tiles using an optimizer. The images were then investigated visually to assess their resemblance to the appropriate digits.

Task 2: Transforming an image of a digit into an image of a different digit In this task we verified the ability of our approach to transform an existing image of a digit into an image of a different digit. This experiment was similar in its execution to Task 1 with the difference of the input being an existing image rather than pure noise. The resulting image was then visually assessed for its similarity to the target label (digit).

2.5 Results

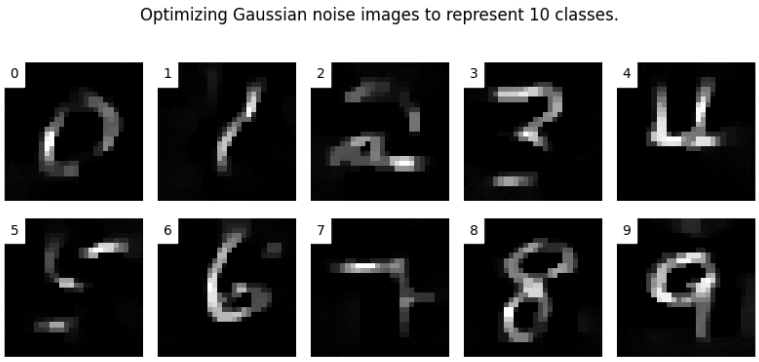
The results of Tasks 1 and 2 are shown in Figure 2.2 for SGD and in Figure 2.3 for Adam. Figure 2.2a shows digits that are clearly distinguishable from an uniform background (with the exception of digits 4 and 9, which lack readability in comparison to other digits). Similarly, transformation of an image of digit 9 into an 8 was successful for the SGD-trained classifier. However, the results are wildly different in the case of an Adam-trained classifier. None of the digits were successfully optimized during Task 1 and, during Task 2, the image was transformed into what seems like pure noise.

2.6 Conclusions

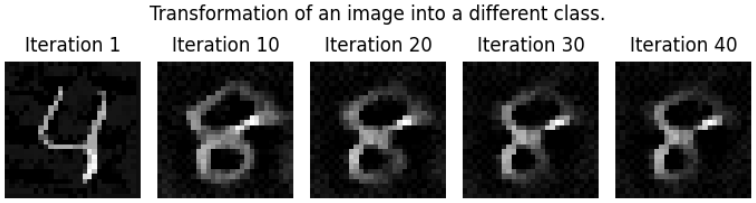
The results obtained using a classifier trained with an SGD optimizer serve as a proof of concept for the feasibility of the approach. Both tasks were largely successful and the obtained reconstructions were clearly distinguishable from each other. Having said that, an interesting problem emerged. The whole approach fails when the SGD optimizer used to train the classifier is substituted for Adam. As seen on Figure 2.3, both tasks failed in that case. It is worth noting that the varying the optimizers in the image optimization phase had no influence on the results in both cases.

2.6.1 Analysing the optimizer discrepancy

Pinning down directly what is the reason for such a discrepancy is a non-trivial task considering there are two distinct phases with different goals that may be the source of the resulting differences - the classifier training and the image optimization. First, let's consider the training phase. Both models achieved comparable classification accuracy of around 92% when assessed on the test set of the MNIST dataset. Additionally, literature suggests that both optimizers should achieve comparable end performance [8, 26]. Considering this fact, it seems that the discrepancy is more likely to be due to the way that the two modalities of the approach are adapted to the second task - optimizing the images. Our hypothesis is that Adam tends to locate a different loss minimum



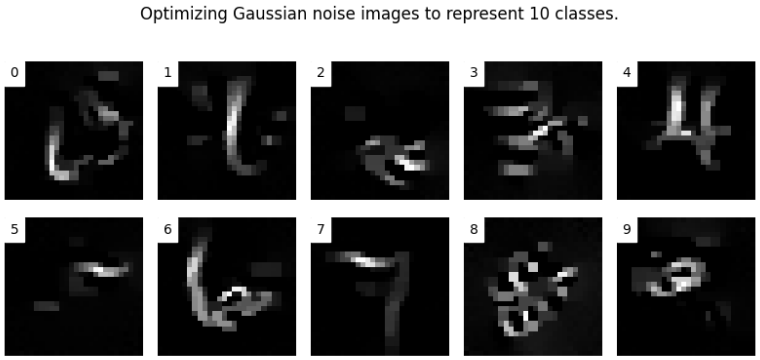
(a) Optimizing Gaussian noise images into images of digits through the usage of classifiers yielded images of low quality but, in most cases, clearly representing the assigned number.



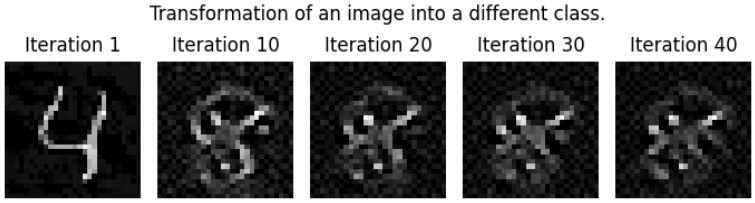
(b) Transforming an image to represent a different digit than originally yielded unsatisfactory results

Figure 2.2: Best results obtained through initial exploration of direct image optimization.

than SGD when learning the classification task. That minimum provides comparable or better performance in that specific task but generalizes worse to the image optimization tasks than its SGD counterpart. With that hypothesis in mind, we proceed to the next chapters where we explore the intricacies of the latent representions.



(a) Using an Adam-trained classifier for the task of obtaining 10 digits from noise provided curious, subpar results in comparison to SGD.



(b) Transforming an image to represent a different digit than originally yielded unsatisfactory results

Figure 2.3: Best results obtained through initial exploration of direct image optimization using an Adam-trained classifier. No tenable performance was registered, especially in comparison with the SGD counterpart.

Chapter 3

Increasing Network Robustness

3.1 When networks are confidently wrong

The previous section of this thesis has shown the potential of inverting networks to reconstruct images based just on the final output signal. Using a relatively simple convolutional neural network, we have managed to recreate convincingly looking images of digits by simply determining which digits we wanted to obtain (see Figure 2.2). However, at the same time, we managed to find obstacles that prevented us from achieving satisfactory performance in a wider array of settings. Specifically, just by changing the optimizer that was used to train the classifier from SGD to Adam, seemingly all ability to reconstruct images was lost (see Figure 2.3). Since the drop-off was surprising - the change of optimisers does not normally result in that big changes of performance - we decided to investigate the matter further. We hypothesised that during the training process, Adam tended to find minima that overfitted to the classification task, focusing on patterns in data that were not aligned with what humans focus on when deciphering hand-written digits. The idea of the misalignment between the inputs and the outputs, and specifically fooling networks into making wrong classifications is an actively studied area of adversarial attacks [7, 27]. Specifically, adversarial examples generation focuses on creating examples that elicit specific responses from models against their original design [10].

3.2 Adversarial examples and how can we use them

While adversarial attacks in general are outside the scope of this thesis, we can leverage some of the findings of this fields to help in our experimentations. Notably, Mejia et al. [21] found in their research that networks trained to be

robust against adversarial examples attacks are more susceptible to another type of attacks - networks inversions. While the authors of said paper present it as a drawback of robust training, susceptibility to network inversions could actually be seen as desirable in the context of our research. Another important finding that we will leverage in our work, stems from Goodfellow's et al. 2015 paper [10]; the researchers speculated that neural networks are prone to adversarial attacks because of their *linearity*. They elaborated that despite deliberate non-linearities in the architectures of neural networks, they spend most time in a linear regime. Basing on those observations, they offered the **Fast gradient sign method (FGSM)** for generating adversarial samples. This surprisingly simple and very efficient method leverages the gradient of loss to introduce small perturbation to inputs that fool the network:

$$X_{\text{new}} = X + \epsilon \cdot \text{sign}(\nabla_x J(X, y_{\text{true}})) \tag{3.1}$$

Here, 'sign' is a simple function that returns 1, 0 or -1 depending whether the input is positive, equal to zero or negative, respectively. $J(\theta, x, y)$ is a loss function (depending on the task at hand, e.g., cross entropy for classification) and ϵ is a hyperparameter that dictates the size of the perturbation. While the method itself is interesting, using it directly would be of no use to our deliberations. In order to benefit from it, we need to employ it in a training regime of our networks to increase their robustness and, hopefully, their susceptibility to network inversions as suggested by Mejia et al. [21] which in turn could potentially improve our reconstructions and overcome the Adam training issues. We follow the works of Goodfellow et al. and Madry et al. [10, 18] to train a robust neural network using the following loss function:

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)), y) \tag{3.2}$$

The above criterion combines values of a standard loss function for two cases: the regular training example and the example obtained through perturbation of the regular training example. This regime should force the network to become more robust against adversarial attacks and provide a 'smoother' optimisation space for our image reconstruction task. The hyperparameter $\alpha \in (0, 1)$ weighs the importance of both terms, however Goodfellow et al. present $\alpha = 0.5$ as a satisfactory value for most cases and we follow their advice here.

3.3 Intermediate classes

In Section 3.2, we laid out the argument for increasing robustness as a method for improving the reconstruction capabilities of our network inversions. Aside from using gradient-perturbed examples, we decided to investigate another approach for increasing robustness - generating intermediate class examples through the use of Variational Auto-Encoders.
Our approach to this task necessitates an introduction of a new group of models - Auto-Encoders.

3.3.1 Auto-Encoders

This group of models aims to learn a dense representation of a given domain of inputs by means of a bottle-neck network [3]. The architecture of Auto-Encoders can be divided into two parts: the encoder and the decoder. The decoder takes the input an passes it through its layers, condensing the input to a representation whose dimensionality is no larger the dimensionality of the input. This latent representation is then passed to the decoder which aims to recreate the original input with as much veracity as possible. The resulting network can be used for a variety of tasks, including de-noising, data compression, etc.

Variational Auto-Encoders (VAEs) This group of Auto-Encoders deserves additional attention thanks to their special characteristics [15]. VAEs differ from standard Auto-Encoders in their latent space is represented - through a Gaussian distribution. The encoder condenses the inputs into an N-dimensional latent space where each dimension is represented by a mean and a standard deviation of a Gaussian distribution. The decoder then samples from these distributions and subsequently tries to recreate the input, similarly to standard Auto-Encoders. The sampling step grants VAEs a unique characteristics - the outputs of the model become stochastic which means that we can use VAEs to generate completely new data points.
Having these two approaches in mind, we devised the following research question:

> Can the performance of networks inversions in relation to image reconstruction be improved by prioritising network robustness during training?

3.4 Approach

In order to generate intermediate class examples we decided to leverage the latent spaces of Variational Auto-Encoders. We trained a VAE on the MNIST dataset, then sampled a portion of the training dataset and obtained codings of the sample. We then picked pairs of codings at random and computed their weighted sum according to the following equation:

$$z_{\text{new}} = \alpha z_1 + (1 - \alpha) z_2 \tag{3.3}$$

Here, z_i stands for a coding of an image provided by the VAE's encoder and α is a term weighting parameter that was chosen at random for each pair. The resulting codings were then decoded to form new examples of intermediate images. In order to appropriately represent the newly formed images, we transformed the labels from a single digit label to a vector that represents the probability distribution of the mixed digits. For example, if we mixed an image of a 4 and an image of an 8, the label would be represented as a vector $[0, 0, 0, 0, \alpha, 0, 0, 0, 1 - \alpha, 0]$.

3.5 Results

3.5.1 Adversarial Training

Qualitative analysis of the influence of adversarial training shows an enormous improvement in reconstruction quality. This was especially true for the Adam training context where initially no results could be observed - the reconstructions were meaningless and noisy. After introducing FGSM examples to our training loss, the situation changed immensely - in both tasks, the reconstructions were legible and realistic (see Figure 3.1).

3.5.2 VAE Intermediate Classes

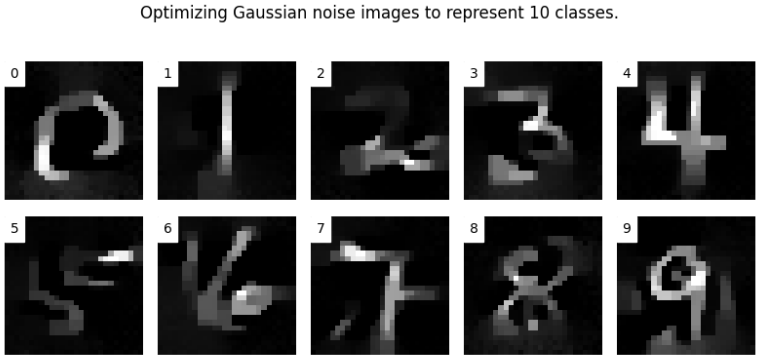
Examples of the new digits can be observed in Figure 3.2. While the model offered satisfactory performance in many cases, sometimes a linear interpolation led not to a visual mix of two numbers but rather to a different digit altogether (e.g., image in the bottom right corner of Figure 3.2). When comparing the reconstruction results, we can observe a convincing increase in performance in the case of a classifier optimized with Adam. Conversely, the classifier trained with SGD showed a drop-off in the reconstruction quality in comparison to experiments run without intermediate classes (see Figures 2.2 and 3.3).

3.5.3 Class separation analysis

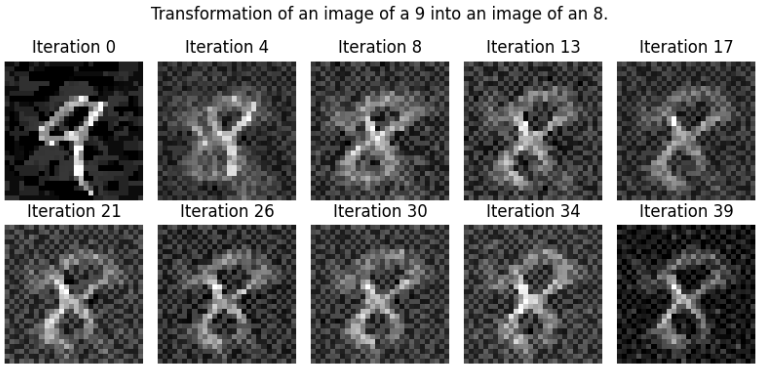
During the process of creating intermediate classes using a VAE, we observed that performing linear interpolations between codings of digits in many cases yielded unexpected results. Examples of such situations can be observed in Figure 3.2 - interpolating between a 0 and a 4 here resembles a 9 or an 8 (image in row 3, column 4). This suggested poor class separation and could potentially impede the quality of a classifier trained on such data (a significant drop-off could be seen in the case of a classifier trained with SGD on 'merged' data). In order to further investigate this finding, a t-SNE analysis of the latent space of the VAE was performed. Its results, which can be seen in Figure 3.5, confirm the pitfall of linear interpolation in this case.

3.6 Conclusions

Initial explorations of using convolutional neural networks for image reconstruction resulted in promising albeit unstable results. We managed to obtain realistic examples from SGD-trained contexts however classifiers trained using Adam proved unable to provide any results. In this section of the thesis we explored improving the reconstruction ability of our networks by introducing robust training. Our hypothesis was that increasing the variety of the dataset would allow the networks to create a more linear representation of our problems which would then correspond to a smoother direct optimisation of images. We focused on two distinct approaches: (1) training with FGSM adversarial



(a) The ten digit optimization task yielded well-defined, clear results for most cases after training the classifier robustly.



(b) In comparison to the naïve training scheme, the transformation results were very satisfactory. A checker-board effect can be observed on the final image that curiously could not be removed through stronger regularization without negatively influencing the digit representation.

Figure 3.1: Best results obtained through direct image optimization for a classifier trained using Adam and a robust training scheme with adversarial examples. Overall, an immense improvement was observed in comparison with the naïve training scheme.

examples and (2) training with intermediate class examples generated using a VAE.

Adversarial Training Introducing adversarial training presented by Goodfellow et al. [10] offered a significant boost in qualitative reconstruction quality. The results can be observed in Figure 3.1 - the obtained reconstructions are of high quality and more robust to hyperparameter changes. The effect of intro-

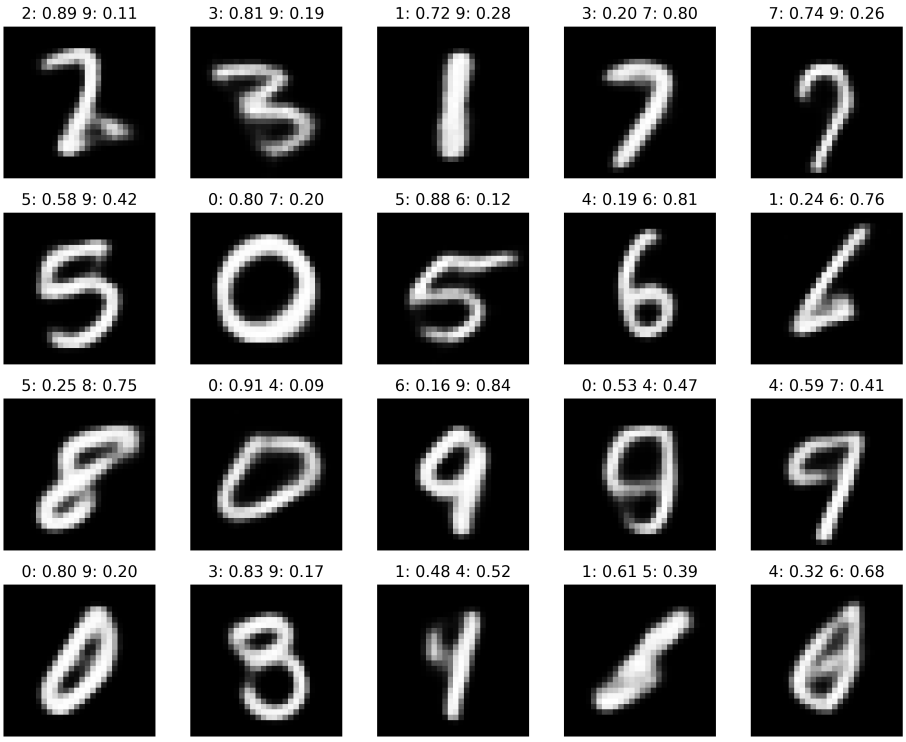
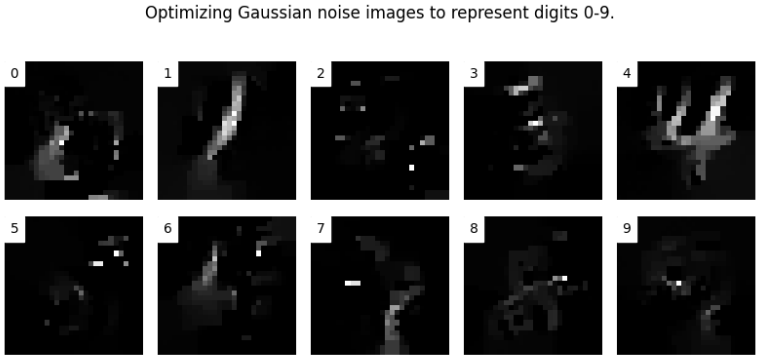


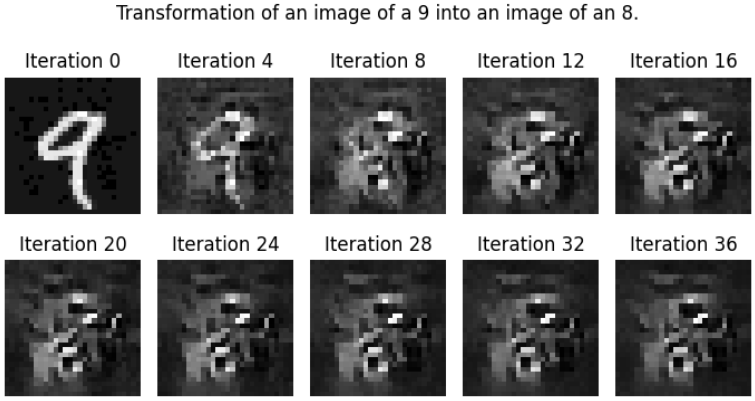
Figure 3.2: Examples of merged digits produced by the Variational Auto-Encoder. The numbers were generated by decoding a weighted sum of encodings of examples from the original MNIST dataset. The titles of the images correspond to weights assigned to encodings of corresponding labels.

ducing adversity is especially visible when compared with original results when the naïve training scheme was utilised.

Intermediate Classes Our second approach based on a similar idea of increasing the linearity of the problem representation. However this time, instead of introducing adversity, we took the approach of explicitly defining intermediate classes. To that end, we leveraged a VAE trained on the MNIST dataset. We generated samples that combined the latent representations of different digits and used them to test our assumption. The results were promising however with some caveats. Introducing the new intermediate examples did improve relative performance of the Adam-trained classifier, confirming the suspicion that a smoother optimization space would make it easier for Adam to find a more generalisable optimum. However, the relative performance of an SGD-trained classifier actually dropped of. We posit that this was likely due one specific drawback of our VAE-guided approach: in some cases, the linear interpolations



(a) Optimizing Gaussian noise images into images of digits through the usage of classifiers yielded images of low quality but, in most cases, clearly representing the assigned number.

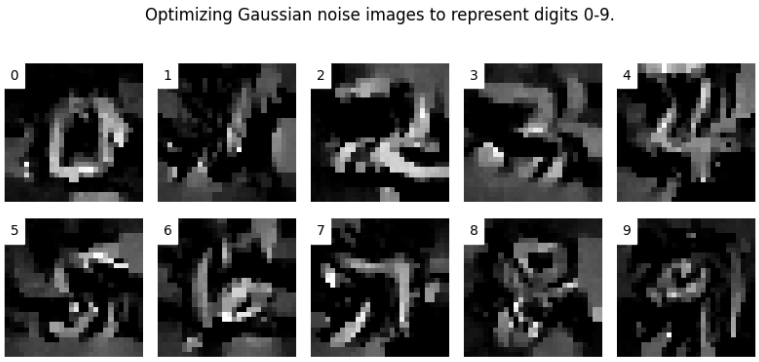


(b) Transforming an image to represent a different digit than originally yielded unsatisfactory results

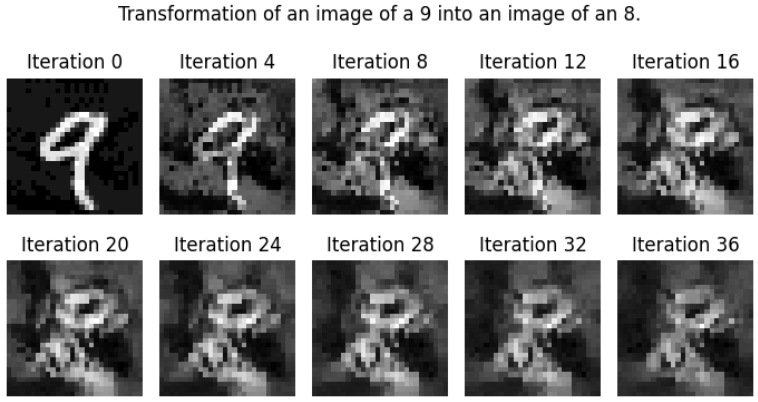
Figure 3.3: Best results obtained through initial exploration of direct image optimization.

between the codings of two different digits could inadvertently result in a whole new digit whatsoever. In order to confirm this hypothesis, we conducted an additional t-SNE analysis of the latent space of our VAE. The analysis confirmed our suspicions; VAE did not separate the classes sufficiently for us to interpolate between them in a straightforward fashion.

Overall, increasing network robustness proved to be the correct way of increasing our reconstruction quality. This confirmed the conclusions of Mejia et al. [21] that robustness of the network is positively correlated with its sus-



(a) Using an Adam-trained classifier for the task of obtaining 10 digits from noise provided curious, subpar results in comparison to SGD.



(b) Transforming an image to represent a different digit than originally yielded unsatisfactory results

Figure 3.4: Best results obtained through direct image optimization for the classifier trained with a dataset enriched by VAE-generated examples.

ceptibility to inversion. While at this point, we started obtaining realistic and satisfactory reconstructions that were robust to a wider range of hyperparameters, there was another aspect of the representation issue that we decided to tackle - class separation.

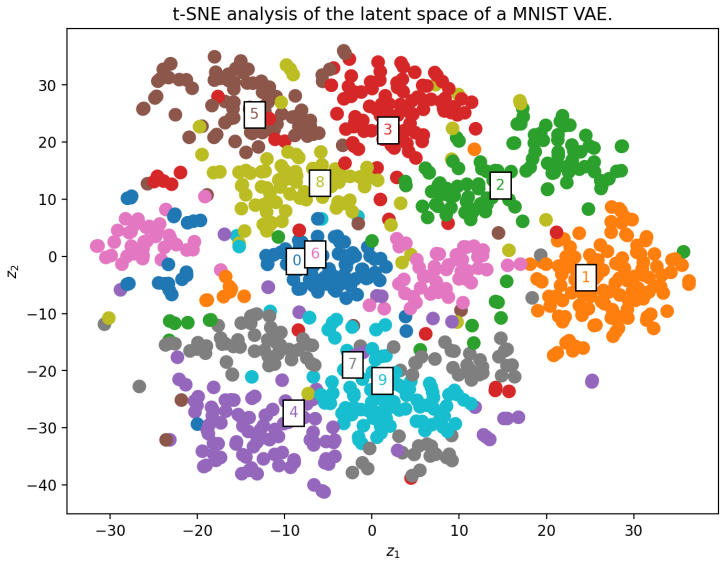


Figure 3.5: Decomposition of the latent space of a Variational Auto-Encoder (VAE) trained on the MNIST dataset. Interpolating in this space can lead to unintended consequences - e.g., when taking the interpolation between a '0' and a '5', the output would probably be an '8'.

Chapter 4

Class Separation

In Chapter 3 we discovered how lack of class separation can impede our overall goal of image reconstruction. Given how generating additional, intermediate training samples yielded a promising boost in performance, we decided that the best train of action would be to try and alleviate this issue. We decided on using two techniques that can help separate classes in latent spaces: prototyping and using non-euclidean geometries.

4.1 Latent Space Geometries

When training any Computer Vision Neural Network, the input instances are transformed into a latent space. The geometry of this latent space can be instrumental to the behaviour and performance of the resulting network. In this thesis, we have examined three distinct geometries and their influence on the quality image reconstructions under network inversion.

Euclidean Geometry

Euclidean geometry is a mathematical system that has been the paradigm for most artificial intelligence applications. The underlying assumption of Euclidean geometry is that it is defined on a flat space which is reflected in Euclid's fifth postulate:

> That, if a straight line falling on two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if produced indefinitely, meet on that side on which the angles are less than two right angles.

Euclidean geometry is well suited for a wide range of problems and has been historically adopted in most of the state-of-the-art Machine Learning systems. However, in some contexts, other geometries that are characterised by curved spaces have been proven to be better choices.

Hyperspherical Geometry

A handy intuition for hyperspherical geometry can be obtained in its 3D case - a sphere. Defined by constant positive curvature, hyperspherical geometry differs from Euclidean theoretically in only one aspect - the Fifth Postulate. In hyperspherical geometries, parallel lines, in contrast to Euclidean, do at some point meet. In the context of this thesis, two aspects of hyperspherical geometry are important:

- Points are defined on a unit n-sphere:

$$S = \{x \in \mathcal{R}^{n+1} : \|x\| = 1\} \tag{4.1}$$

- The shortest distance between two points is no longer a straight line. Instead, we have to follow a geodesic along the curvature of the hyperplane. In this thesis, we will use the negative cosine similarity as a proxy for distance [23]:

$$\text{dist}(x, y) = 1 - \cos\theta_{x,y} \tag{4.2}$$

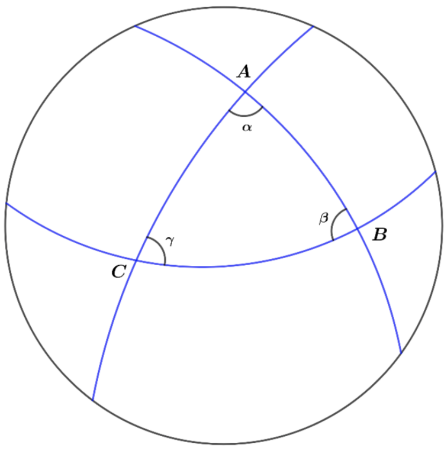


Figure 4.1: Under a hyperspherical geometry, the shortest lines between points - geodesics - become arcs rather than straight lines. Image from John Kogut's lecture [16].

Hyperbolic Geometries

Hyperbolic geometries are an alternative to Euclidean geometry that are characterised by constant negative curvatures (i.e., the space is constantly curved in opposing directions, resembling somewhat a horse saddle). Several hyperbolic geometry models exists, differing between each other by how points and geodesics are defined on them. The three most popular models are the Poincaré

disk, the Hyperboloid (the Lorentz model) and the Klein model [22]. In the context of this thesis, I used the Poincaré disk as a representative of this group of geometries due to its relative simplicity of use and analogies to the hyperspherical geometry model.

The Poincaré disk The Poincaré disk model is a representation of the hyperbolic space by means of a unit ball defined in the Euclidean space as:

$$\mathbb{D}_d = \{p \in \mathbb{R}^d : p_1^2 + \ldots + p_d^2 < 1\} \tag{4.3}$$

The shortest distance, also called a geodesic, between two points in the Poincaré disk is defined as the arc of a Euclidean circle that passes through both points and intersects with the boundary $\mathbb{D}_d$ at a right angle. The Poincaré disk is popular for a lot of applications because of its attractiveness in visualizations in its 2D case but also because it is conformal, i.e. hyperbolic angles are measured as in Euclidean geometry [22].

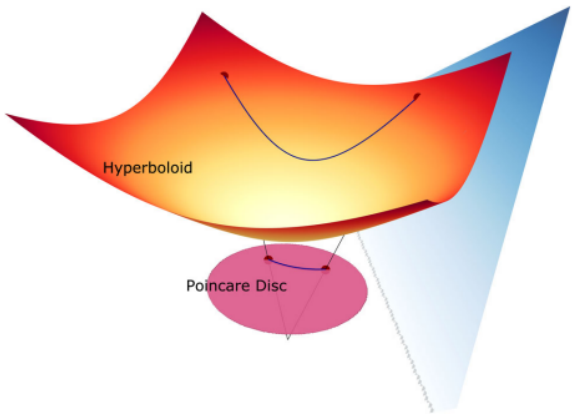


Figure 4.2: Visualization of the Poincaré disk as a projection of a 3D hyperboloid.

4.1.1 Prototype Learning

In classical machine learning approaches, a fresh network starts training with no knowledge about the data at hand. During training, the network fits to the data, creating a latent representation that (at least theoretically) explains the underlying patterns. While this approach works well in most cases, sometimes the characteristics of the problem or the data make it difficult to represent them in such a way. Prototype learning allows to steer the learning process by either enforcing large separation or introducing semantic information before the start of training [4].

4.2 Prototype Learning

Prototype learning is a useful method for decreasing dimensionality of models while retaining performance, ensuring appropriate class separation and incorporating information about data into models before the start of actual training [2, 4, 23]. In the context of hyperbolic learning, it can also provide help in structuring hierarchical data before the start of training [2, 22]. Mettes et al. in their 2019 paper [23] focus on hyperspherical prototypes. They present a learned method for positioning the prototypes based on desired dimensionality, the type of problem (classification/regression) and the number of classes (in the case of classification). In their approach, the network learns a mapping from inputs to corresponding prototypes by maximising the cosine similarity between them. In our work we took their general approach and applied it to our problem

4.3 Prototype Learning

4.3.1 Hyperspherical Prototypes

One approach of achieving good separation of classes is through placing prototypes on a hypersphere and then learn a mapping from inputs to appropriate prototypes. This presents two problems: (1) how to find a placement of well-separated prototypes and (2) how to describe the mapping in an efficient way in a hyperspherical geometry? In this thesis we based our approach on the work of Mettes et al. [23]. The crux of their work lies in leveraging cosine (dis)similarity for both placing the prototypes and learning the mapping.

Placing the prototypes Obtaining optimal spacing on a 2D hypersphere (i.e., a circle) is straightforward - given K classes, we can divide the circle into K equal slices by placing a point every $360°/K$. However, performing this task in higher dimensions becomes way more challenging and straightforward solutions exist only for some edge cases [28]. Mettes and al. offer a solution based on the assumption that the set of optimally spaced prototypes minimises the maximal cosine similarity between any two prototypes. They therefore propose a learned approach with the following loss function:

$$\mathcal{L} = \frac{1}{|K|} \sum_{i}^{K} \max_{j \in K} M_{ij}, \quad M = \hat{P}\hat{P}^T - 2I, \; s.t. \forall_i \left\| \hat{P}_i \right\| = 1 \tag{4.4}$$

The matrix M in the equation above is the dot product of the set of prototypes with itself (which equates to the cosine similarity between each pair of prototypes). A doubled identity matrix is subtracted to avoid self-selection.

Learning the mapping Once we obtain the prototypes, we can learn to map the inputs into appropriate prototypes by once again using the cosine similarity

- this time we aim to maximise it between the mapping and its corresponding prototypes. The loss function is thus formulated as follows:

$$\mathcal{L} = \sum_{i=0}^{N} (1 - \cos \theta_{z_i, p_{y_i}})^2 = \sum_{i=0}^{N} (1 - \frac{|z_i \cdot p_{y_i}|}{\|z_i\| \, \|p_{y_i}\|})^2 \tag{4.5}$$

4.3.2 Hyperbolic Prototypes

The main motivation behind using the hyperbolic prototypes is the exponential expansion of the hyperbolic space [22]. This characteristic becomes useful when dealing with hierarchical, or tree-like structures. Moreover, since the space becomes effectively bigger, we can potentially fit more data into the same space. This same characteristic introduces however an additional difficulty in modelling the prototyping learning problem in a hyperbolic geometry. If we consider a Poincaré disk model, placing the prototypes on the boundary (as we did in the hyperspherical case) is no longer feasible. The Poincaré disk model is bounded by an imaginary circle, meaning that the boundary lies at an infinite distance from the centre and is therefore not reachable. We overcome this problem by following the approach of Atigh et al [2]. The researchers modelled our problem using a mathematical limit as training loss - the Busemann loss [6]:

$$b_p(z) = \lim_{t \to \infty} (d_{\mathbb{B}}(\gamma_p(t), z) - t) \tag{4.6}$$

In the Poincaré model that we leverage in this thesis, this Busemann limit can be expressed in close form as:

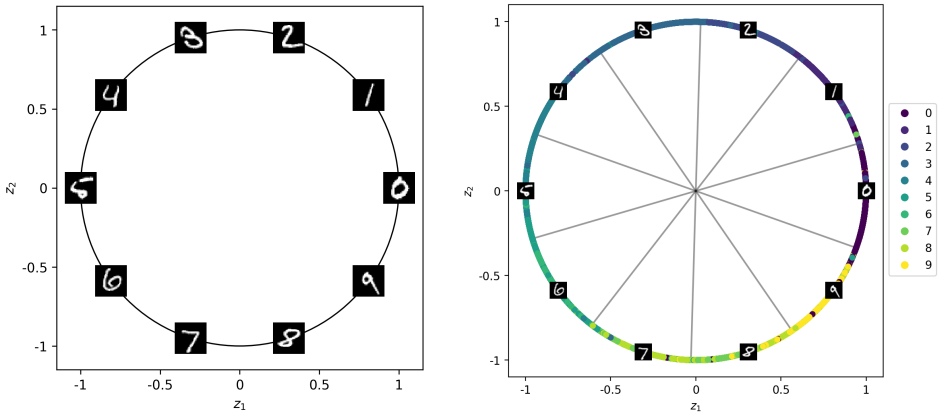
$$b_p(z) = \log \frac{\|p - z\|^2}{1 - \|z\|^2} \tag{4.7}$$

4.4 Results

4.4.1 Hyperspherical Prototypes

Manifold	Precision [%]	Recall [%]	F1-score [%]
2D (circle)	78.4	77.2	77.1
3D (sphere)	90.9	90.4	90.4
5D (hypersphere)	**97.7**	**97.7**	**97.7**

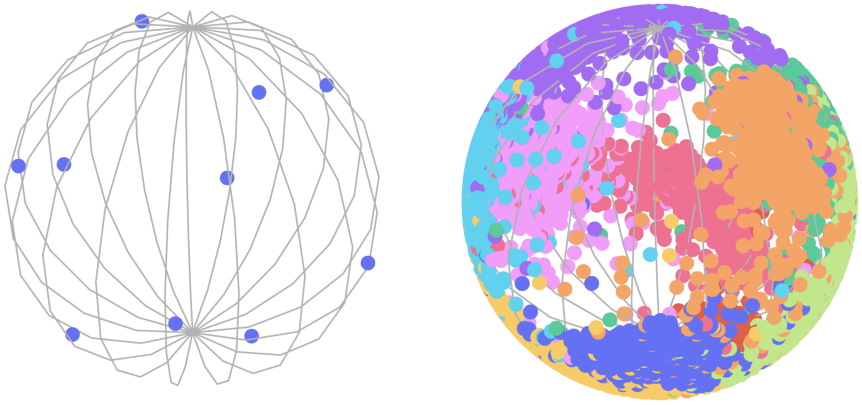
Table 4.1: Comparison of classification evaluation metrics on the test set of MNIST for different dimensionalities of the manifold. 2D manifold was to small to fit the 10-class problem of MNIST. Optimal results were obtained for a 5-dimensional manifold.



(a) Before the start of training, the prototypes were placed on a unit circle every $\frac{\pi}{10}$ rad.

(b) 2-dimensional mapping space was not enough to accommodate the 10 classes of the MNIST dataset.

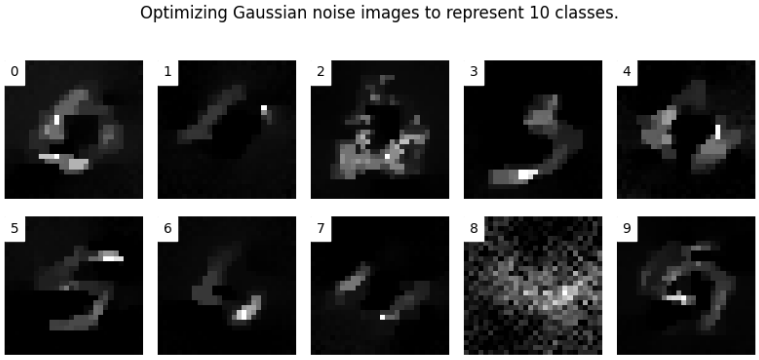
Figure 4.3: When examinating the feasibility of the 2D hyperspherical prototype approach to MNIST classification, 10 prototypes were placed on a unit circle and a mapping input $\rightarrow$ prototype was learned.



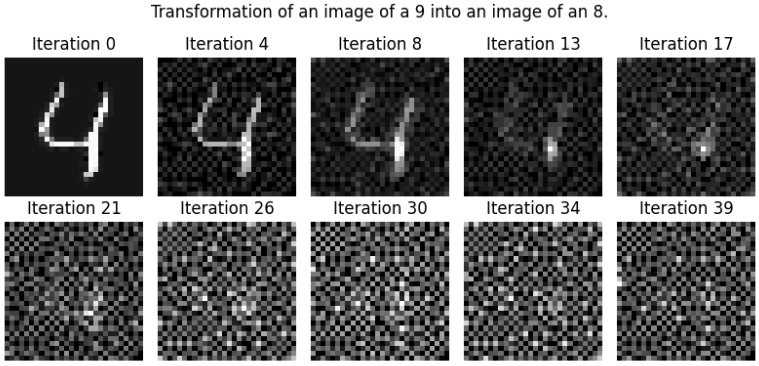
(a) Prototypes (blue dots) placed on a unit sphere. In order to appropriately space out the prototypes, a learned approach was used.

(b) Mapping of the MNIST test set onto the unit sphere by a trained network.

Figure 4.4: Visualizations of (a) the prototypes on a unit sphere and (b) the mappings of the MNIST test set onto said sphere. A 3D sphere provided ample space for the 10 prototypes and the corresponding dataset to be fit.



(a) Using an Adam-trained classifier for the task of obtaining 10 digits from noise provided curious, subpar results in comparison to SGD.



(b) Transforming an image to represent a different digit than originally yielded unsatisfactory results

Figure 4.5: Best results obtained through direct image optimization for the classifier trained with a dataset enriched by VAE-generated examples.

4.4.2 Hyperbolic Prototypes

In order to evaluate the hyperbolic approach, a Busemann prototype network was trained on the MNIST dataset. We compared prototypes of three different dimensionalities - 2, 3 and 5 dimensions - to select one that provides the best performance while remaining lightweight. All three dimensionalities performed very well, especially to low-dimensionality versions of the hyperspherical approach. Considering the highest generalisation performance of the 5D network (see Figure 4.2) this dimensionality was chosen for further studies. Performing network inversion on said network yielded mixed results. The reconstructions were subpar in comparison to Euclidean, non-prototypical approaches, however

the reconstructed digits were somewhat visible. Especially interesting was the 'connect the dots' pattern, visible in Figure 4.7.

Manifold	Precision [%]	Recall [%]	F1-Score [%]
2D	92.4	92.4	92.4
3D	95.5	95.5	95.5
5D	**96.2**	**96.2**	**96.2**

Table 4.2: Classification results on the MIST test dataset for the Busemann Hyperbolic Prototypes.

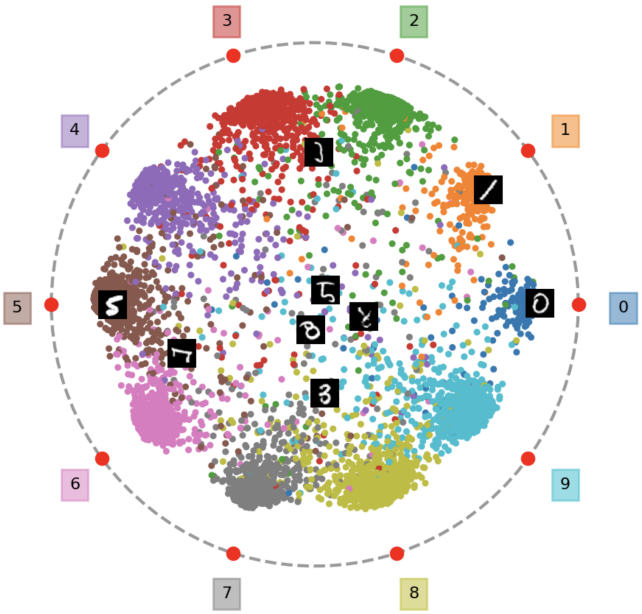
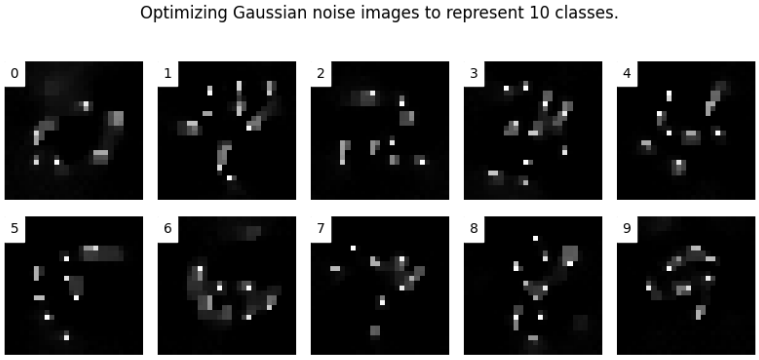


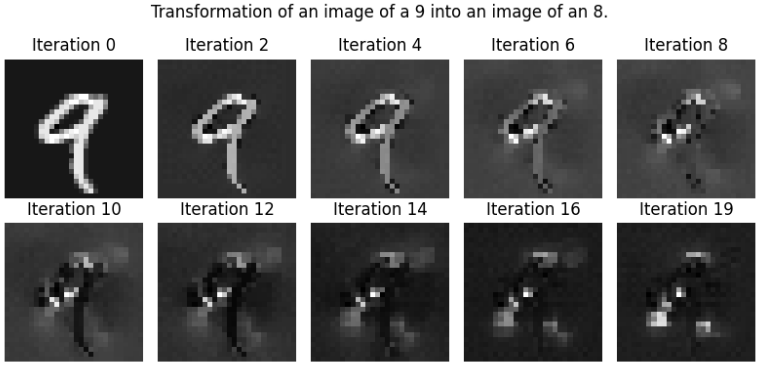
Figure 4.6: Mapping of the MNIST test set onto the Busemann prototypes. The closer a point is to a prototype (points on the outline of the circle), the more confident model was in its mapping. Uncertain mappings are located towards the centre of the circle. Colours of the points correspond to the class their labels in the test set.

4.5 Conclusions

Inverting a network to perform image reconstruction requires an accurate representation of the features that determine the class of an input image but also an alignment between how the network internally represents and recognized the images and how humans perceive them. One way to do it, we hypothesised,



(a) Inverting a hyperbolic prototypes network yields curious results - the reconstructed digits have a 'connect the dots effect'.



(b) While the signal from the inverted network in this case provided substantial change toward the assigned digit, the results were subpar in comparison to other methods.

Figure 4.7: Qualitative results obtained through a 3D hyperbolic prototypes network.

is to provide a better separation of the classes or a different geometry of the representation. To that end, we evaluated two distinct prototyping approaches - hyperbolic and hyperspherical prototypes. The results showed a subpar performance in comparison to non-prototypical approaches conducted in Euclidean geometry. This, as we understand can be traced back to the core idea of prototypical learning - instead of a representation of the images, we learn a mapping from inputs to fixed prototypes. While such approach leads to well separated classes (given large enough space, i.e. the dimensionality of the prototypes), it does not correspond well to our network inversion tasks. Having said that, an interesting phenomenon occurred when using the hyperbolic prototype approach. The digits reconstructed in task 1 were legible, however a 'connect the

dots' effect appeared. The exact reasons for this occurrence and solutions to increase fidelity of the reconstructions are a matter for future work. It is however our opinion that prioritising class separation is not ultimately worth the loss of the smoothness of the representation.

Chapter 5

Case Study

Our investigations so far explored the general feasibility of inverting network and extracting the information they store. We have shown that, in certain contexts, we can reconstruct classes of datasets using the latent information stored in their weights. Our exploration of the issue was however purely qualitative and inherently subjective - the results were examined and compared visually by the authors. Such approach gave us information on how different learning contexts adapted to the task of inversion, it lacked however fair comparisons between approaches that fared similarly under the qualitative regime. In Section 1.2.2 we have elaborated why we have chosen to evaluate our models in that way. Having said that, a quantitative comparison of methods and their usability in more real-life scenarios would be beneficial. With that in mind, we have decided to conduct a case study. Image reconstruction in Computed Tomography is famously an ill-posed problem. While some data-based approaches were developed, model-based, iterative methods like SIRT still remain state-of-the-art. We wanted to answer the following question: "Does introducing a network inversion scheme into the iterative model-based methods improve reconstruction quality?"

5.1 Background

Computer Tomography (CT) offers a relatively non-invasive insight into the inner structure of objects, animals, plants or humans. Using Computer Tomography is not trivial - mapping the detected attenuations of the γ-rays is an *ill-posed problem*. Obtaining high-quality CT reconstructions that are robust to noise has been an active area of research for decades.

5.1.1 Conventional methods of CT reconstruction

Historically, CT image reconstruction has been performed using a model-driven approach. The reconstruction algorithms were design to mimic the inverse of the forward CT operator. Initially, such methods did not yield satisfying results

due to the instability of the problem; the reconstructions were very dependent on small changes to input. As a solution to this problem, iterative methods were developed. These methods commonly were based on the idea of decreasing the variance of the reconstruction by a small fraction while at the same time respecting the similarity to the inverse model.

FBP The Filtered Back-Projection (FBP) [14] is an algorithm that aims to reconstruct images obtained in parallel geometries in a two-step fashion. First, the 1D projections are transformed into the frequency domain where a variation of a low-pass filter is applied. This step aims to discard noisy portion of the image, resulting in a reconstruction that more closely resembles the input image. During the second step, the filtered projections are transferred back into the image domain and back-projected onto the image space along their respective angles. FBP constitutes a light-weight method of obtaining moderate-quality reconstructions. The use of the low-pass filter results in reconstructions that tend to be blurry due to the overemphasis of low-frequency data.

SIRT The Simultaneous Iterative Reconstruction Technique (SIRT) algorithm is an algebraic iterative method that provides better reconstruction results than analytic methods like FBP [14]. SIRT provides reconstructions by iteratively improving an initial guess based on the forward operator data. During each step, the guess is projected onto ith projection hyperplane under the assumption that if a unique solution exists, the algorithm will converge to it, with every step providing a better reconstruction than the previous step. SIRT provides satisfactory reconstructions, is robust to noise and allows to incorporate constraints (e.g., non-negativity, regularization terms). However, it is also computationally heavy. especially in comparison to analytical methods like FBP.

5.1.2 Data-driven approaches to CT reconstruction

With the advent of artificial intelligence and its promising achievements in the field of computer vision, numerous researchers tried to implement such methods in the field of CT reconstructions. Such methods can be divided into three distinct types:

Learned post-processing (learned de-noising) - where a data-driven approach is employed to improve a reconstruction obtained through conventional reconstruction methods.

Learned regularization - a regularization term is learned and then combined with traditional iterative algebraic methods.

End-to-end optimization In such methods, a data-driven approach is used to model the whole inverse operator, mapping input data to complete reconstructions. Such approaches are constly to perform in a single step and do not

scale well to data sizes seen in real world. Instead, some authors propose iterative data-driven methods that move raw data towards complete iterations in small steps. [1]

5.2 Experiments

In order to comprehensively compare models that fared similarly well, we developed a second evaluation framework which was aimed at evaluating models in the context of the task laid out in the research question: improving reconstruction quality through the combination of a conventional and data-based approaches. The experiments in this section were performed using the ASTRA toolbox for electron tomography research [29, 30]. The framework was constructed:

1. Forward projections of test images were obtained under specified conditions (projection geometry type, number of angles, number and spacing of detector pixels, etc.),
2. The resulting sinograms were passed through 50 iterations of SIRT in order to obtain initial reconstructions.
3. The initial reconstructions subsequently underwent 50 additional iterations of alternating transformations - SIRT and the tested model step - in a ratio of 3:1.
4. The final reconstructions were compared with the ground truth images using MSE, PSNR, and SSIM. The results were benchmarked against a reconstruction provided by 100 of SIRT only.

5.3 Results

5.4 Conclusions

In this case study we have explored the effects of integrating a network inversion approach into the well-established, conventional, iterative CT reconstruction pipelines. Our experiments show that network inversion can bring limited performance boost to angle-constraint CT reconstruction set-ups. This effect is however, in our opinion, at least partially brought on by the regularisation techniques rather than the inversion itself. We theorise that consequential performance improvements in this area using network inversions are possible, however the technique should be applied on problems more complex than MNIST. In order to achieve that, additional work needs to be performed to improve the technique and make it more stable to hyperparameter changes.

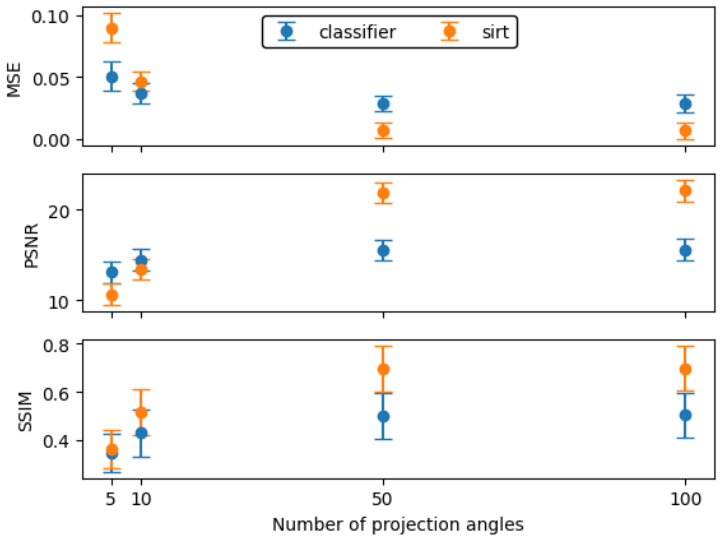


Figure 5.1: Comparison of average evaluation metrics for different numbers of projection angles between pure SIRT and our mixed approach. While a standard SIRT approach was comprehensively better given sufficient number of angles, in very angle-constrained contexts our approach provided moderately better results.

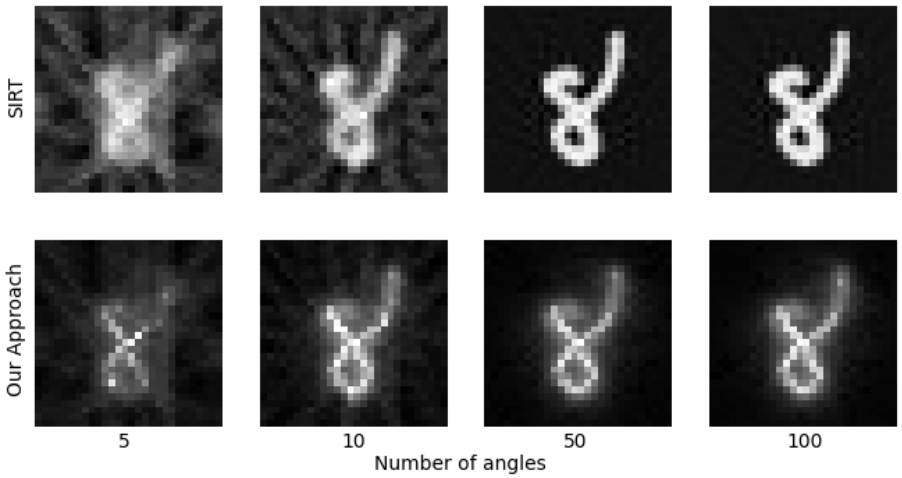


Figure 5.2: Comparison of reconstructions for different numbers of projection angles between SIRT and our approach.

Chapter 6

General Conclusions

In this thesis, we have explored the idea of introducing prior class information into the image reconstruction process via the information encoded in the weights of pre-trained convolutional neural networks. We based our approach on the network inversion mechanism which so far has been explored in several types of adversarial attacks [7, 10, 18] and in AI explainability [20, 24, 35] but has not yet found usage in the area of image reconstruction. Considering that the general task we wanted to perform - retrieving visual information for whole classes - was under-represented in literature, we decided to formulate this thesis as an exploration of this concept. We started with a proof of concept study and subsequently investigated two areas of the problem that we deemed interesting - robustness and class separation. Finally, we concluded with a case study in the area of CT image reconstruction. Our main findings are as follows:

1. Reconstructing class-specific images using neural network inversion in a black-box set-up is feasible and highly satisfactory results can be obtained for simple synthetic datasets.

2. Robustness plays an important role in susceptibility to inversion as robustly trained networks offer a smoother problem representation with better handling of 'garbage' examples.

3. Class separation may play a role in quality of reconstructions, however prototypical methods tend to harm the overall performance due to their focus on classification rather then representation.

4. Using network inversion to improve image reconstructions can bring small boosts in performance in highly constrained settings (e.g., CT reconstructions with very limited number of angles).

5. Using network inversion for image reconstructions presents a difficult problem that requires thoughtful hyperparameter optimisation.

Previous works that could be considered aligned with our research ideas, i.e., aimed to recover high-fidelity images from networks, have in most cases opted

to use a white-box to networks. We decided to focus on a black-box approach where we could perform predictions using a network but we did not use access to the network's weights. In previous literature, such approaches reached subpar results, however this could be also attributed to the fact that said research focused on other tasks where a black-box approach is less feasible. It should be noted that performing network inversion yields a consequential computational overhead. This is due to the fact that for each reconstruction, we need to perform many steps back-propagation (in the case of our experiments, 1000 steps per image). In most learned approaches, this overhead is contained to the training phase. Overall, this exploration thesis has shown the feasibility of the concept of introducing class-specific information into the image reconstruction process. We explored some caveats of this complex problem and offered insights into their purposefulness. For future work, increasing stability of the reconstruction process should be studied and methods for decreasing the number of hyperparameters should be explored. Additionally, this approaches should be validated for more complex, real-life datasets.

Bibliography

[1] Jonas Adler and Ozan Öktem. Learned primal-dual reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1322–1332, 2018.

[2] Mina Ghadimi Atigh, Martin Keller-Ressel, and Pascal Mettes. Hyperbolic busemann learning with ideal prototypes, 2021.

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, April 2021. arXiv:2003.05991 [cs].

[4] Michael Biehl, Barbara Hammer, and Thomas Villmann. Prototype-based models in machine learning. *WIREs Cognitive Science*, 7(2):92–111, 2016.

[5] A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[6] Herbert Busemann. The Geometry Of Geodesics. *Academic Press Inc.*, 1955.

[7] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45, 2021.

[8] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On empirical comparisons of optimizers for deep learning, 2020.

[9] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniques. *Vis. Comput. Ind. Biomed. Art*, 2(1):7, July 2019.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, March 2015. arXiv:1412.6572 [stat].

[11] Christine Guillemot and Olivier Le Meur. Image Inpainting : Overview and Recent Advances. *IEEE Signal Processing Magazine*, 31(1):127–144, January 2014.

[12] Jacques Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, page 49–52, 1902.

[13] Niv Haim, Gal Vardi, Gilad Yehudai, Ohad Shamir, and Michal Irani. Reconstructing training data from trained neural networks, 2022.

[14] Avinash C. Kak and Malcolm Slaney. *Principles of Computerized Tomographic Imaging*. Society for Industrial and Applied Mathematics, 2001.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[16] John Kogut. Planes, spheres and surfaces: From straight lines, triangles and differentiation to geodesics, covariant differentiation, curvature and holonomy, 04 2019.

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks, September 2019. arXiv:1706.06083 [stat].

[19] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them, 2014.

[20] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[21] Felipe A. Mejia, Paul Gamble, Zigfried Hampel-Arias, Michael Lomnitz, Nina Lopatina, Lucas Tindall, and Maria Alejandra Barrios. Robust or Private? Adversarial Training Makes Models More Vulnerable to Privacy Attacks, June 2019. arXiv:1906.06449 [cs].

[22] Pascal Mettes, Mina Ghadimi Atigh, Martin Keller-Ressel, Jeffrey Gu, and Serena Yeung. Hyperbolic deep learning in computer vision: A survey. *International Journal of Computer Vision*, 132(9):3484–3508, Mar 2024.

[23] Pascal Mettes, Elise van der Pol, and Cees G. M. Snoek. Hyperspherical prototype networks, 2019.

[24] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, Jun 2015.

[25] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[26] Frank Schneider, Lukas Balles, and Philipp Hennig. Deepobs: A deep learning optimizer benchmark suite, 2019.

[27] Alex Serban, Erik Poll, and Joost Visser. Adversarial examples on object recognition: A comprehensive survey, 2020.

[28] Pieter Merkus Lambertus Tammes. *On the origin of number and arrangement of the places of exit on the surface of pollen-grains*. PhD thesis, University of Groningen, 1930. Relation: http://www.rug.nl/ Rights: De Bussy.

[29] Wim van Aarle, Willem Jan Palenstijn, Jeroen Cant, Eline Janssens, Folkert Bleichrodt, Andrei Dabravolski, Jan De Beenhouwer, K. Joost Batenburg, and Jan Sijbers. Fast and flexible x-ray tomography using the astra toolbox. *Opt. Express*, 24(22):25129–25147, Oct 2016.

[30] Wim van Aarle, Willem Jan Palenstijn, Jan De Beenhouwer, Thomas Altantzis, Sara Bals, K. Joost Batenburg, and Jan Sijbers. The astra toolbox: A platform for advanced algorithm development in electron tomography. *Ultramicroscopy*, 157:35–47, 2015.

[31] Zhihao Wang, Jian Chen, and Steven C. H. Hoi. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3365–3387, 2021.

[32] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[33] Hanyu Xiang, Qin Zou, Muhammad Ali Nawaz, Xianfeng Huang, Fan Zhang, and Hongkai Yu. Deep learning for image inpainting: A survey. *Pattern Recognition*, 134:109046, 2023.

[34] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019.

[35] Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion, 2020.