Lehrstuhl für Bildverarbeitung

RWTH AACHEN UNIVERSITY

Synthesizing Pseudo CTs with Generative Adversarial Networks and X-Ray based a priori Information

Synthese von Pseudo CTs mittels conditional Generative Adversarial Networks aus Röntgenbildern

Master Thesis

submitted by

Yu-Chia Lan

Mentor:

Justus Schock, M.Sc.

Institute of Imaging & Computer Vision
Prof. Dr.-Ing. Dorit Merhof
RWTH Aachen University
2020

I herewith declare that I have composed the present thesis myself and without use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The thesis in the same or similar form has not been submitted to any examination body and has not been published. This thesis was not yet, even in part, used in another examination or as a course performance.

Aachen, 29 July 2020

Yu-Chia Lan

Abstract

Computed Tomography scans have been established as one of the most important imaging diagnosis tools in medicine. Even though CT scans offer many advantages as an imaging tool, they also come at the cost of exposing the patient to unwanted radiation.

This work explores an alternative method of reconstructing CT images with the use of biplanar X-rays and generative adversarial networks with the goal of reducing the radiation exposure experienced by the patient. Additionally, with the augmentation of existing X-ray image datasets in mind, this work also develops a method to allow single image inference on a model that was trained using multiple inputs.

Several different architectures were evaluated and it was shown that a VAE/GAN based approach was more successful in generating detailed CT reconstructions. Different methods for single image inference were also proposed and evaluated with the KNN prediction method showing the best results.

Contents

List of Figures

List of Tables

1 Introduction

In the last decades, computed tomography (CT) has been established as one of the most important imaging diagnosis tools in medicine. CTs are constructed by capturing a large amount of X-ray sinusoids from a number of angles around the patient and reconstructing a 3D volume through geometric reconstruction. CT scans are generally fast, painless, noninvasive, accurate and produce three dimensional images in high resolution. The presentation of tissues in three dimensional space is invaluable for diagnosis and solves the overlay problem of conventional X-ray images, where projected tissues are indiscernable because of missing depth information.

However, due to the large amount of X-ray like images required for the reconstruction, the radiation doses experienced by the patient during the imaging process is not insignificant and can cause tissue damage in the long term [1, 2].

Conventional X-ray imaging on the other hand is less harmful to the patient, but only produces projection images in 2D space. X-ray scanners are also more widely accessible, especially in developing countries.

Recently, cross-domain synthesis methods between medical image modalities have gained traction due to the desired properties of different domains and the need to reduce unnecessary radiation dosage during the imaging process. Magnetic resonance imaging (MRI) for example does not expose the patient to ionizing radiation and thus synthesizing CT scans from MRIs [3, 4] for clinical applications that require them lets the patient avoid radiation exposure altogether.

In a similar vein, a method to synthesize full CT volumes from regular X-ray images would reduce the patients exposure to radiation considerably. Such a method would also simplify the process of generating CT scans and make them more accessible.

While the relationship of reconstructing X-rays from CTs has been explored in the context of digitally reconstructed radiographs (DRR) [5–7], research regarding the reconstruction of CTs from X-rays has remained sparse. The main challenge for this kind of reconstruction is the amount of ambiguity that cannot be determined explicitly when transforming from a two dimensional to a three dimensional domain.

Recent advances in deep neural networks and especially the development of generative adversarial networks (GAN) by Goodfellow et al. [8] and deep autoencoders have enabled more ambitious applications for generative models. Applications such as generating realistic photographs of human faces [9] or photo inpainting [10] demonstrate the ability of GANs to learn ambiguity and fill them according to the context that was presented, although most of them remained in the 2D domain.

The X2CT-GAN proposed by Ying et al. [11] is one of the few recent publications that approach the subject of reconstructing CTs from X-ray images. They

demonstrate that the synthesis is possible with biplanar X-rays as the basis for reconstruction, and that the ambiguity of missing information can be overcome with the use of GANs and autoencoders.

There are futher application cases which would benefit from the synthesis of CTs. Bone segmentation tasks on X-ray images for example can be very tricky in complicated regions because of overlapping bone tissue in the image [12, 13]. Extending the image into 3D space and segmenting from seperate slice views can simplify the procedure. Such applications suffer from another difficulty, which is that usually only X-rays from one view are available, which makes it impossible to reconstruct using the X2CT-GAN.

Therefore, in this work the following contributions are made:

- design of a deep learning based neural network architecture to synthesize pseudo CT images from regular X-ray images, building upon the results of X2CT-GAN.
- development of a method to allow single image inference, while using multiple images as inputs during the training stage.

2 State of the Art

The following chapter will provide an overview of the fundamentals and current state of the art this thesis is based on.

2.1 Medical Background

To understand the proposed methods and goals, it is essential to understand the physical properties of X-ray and computed tomographic imaging and their relationship with each other.

2.1.1 Projectional X-ray Imaging

Since Wilhelm Röntgen's discovery in 1895, X-ray imaging has been used extensively in the medical field for evaluation and diagnosis due to their speed and non invasive property.

X-rays are electromagnetic radiation that have enough energy to pass through solid and liquid substances, typically in the range of 100 electron Volts (eV) to 100 keV. While passing through the body, the energy of the X-ray is absorbed depending on the density or composition of the tissue. Bones for example absorb much more energy than water or fat. The remaining energy is captured on the other side of the patient by detectors. Assuming the X-rays originate from a single point and are directed towards an object, the process can be visualized as in Figure 2.1. The resulting energy on the other side can be described as the result of the X-ray being attenuated by all the different tissues from the source point to the detector. A single pass of an X-ray can therefore also be understood as a summation of all the information on the different attenuations of the different tissues it passes through. The image captured by the detector is consequently a two dimensional projection of the three dimensional object in the direction of the source to the detector, which is also why this is also called projectional radiography.

X-ray imaging in the medical field are used primarily for hard tissue diagnoses such as bone fractures, but also for some soft tissue diagnoses such as angiography or mammography. Real time X-ray imaging can also be used as a guidance tool for orthopedic surgery such as joint replacements.

Even though other advanced imaging methods such as computed tomography or magnetic resonance imaging exist, the plane X-ray image is still widely used due to being low-cost, fast to produce and being able to provide high-resolution images. An example of an X-ray image can be seen in Figure 2.2.

That being said, X-ray imaging have a couple of disadvantages. Given the nature of a projection, blocking between organs that are usually seperated by depths can occur as there is no way of discerning the depths of an organ inside the image. Furthermore, X-rays in general are ionizing radiation. Prolonged exposure can lead to tissue damage or even cancer. Therefore the amount of radiation dosis applied to a patient should be kept to a minimum. A common way to lower radiation exposure is to filter out 'soft' X-rays with lower energy that would otherwise be absorbed completely by soft tissue. However, radiation exposure is still an issue and the subject of ongoing research [14].

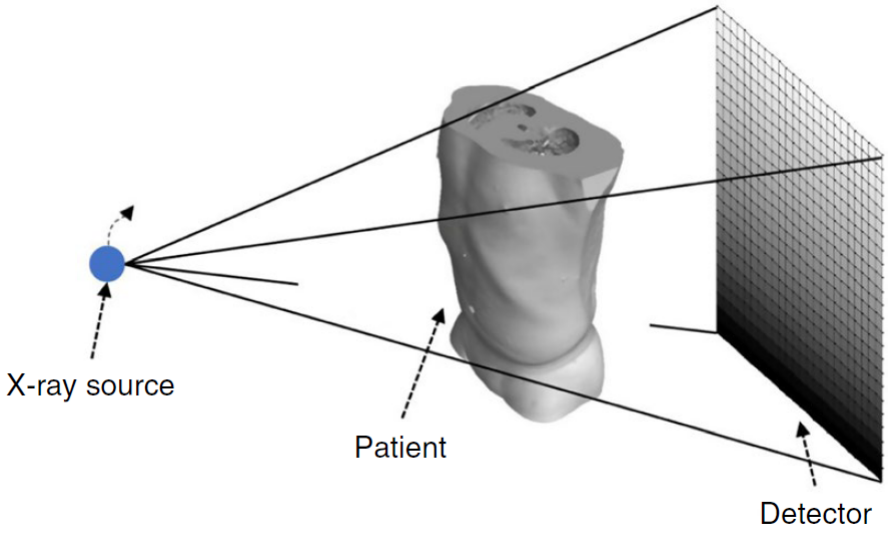


Figure 2.1: Geometric view of an X-ray source, patient and detector. Taken from [15]

2.1.2 X-ray Computed Tomography

X-ray computed tomography is one of the most important imaging methods in medicine. It is based on the same imaging principle as projectional X-ray imaging, but it solves the problem of organs and tissues overlapping in the image by introducing image slices.

Instead of only taking one X-ray image from one direction, modern CT scanners circle the X-ray source and receiver around the patient in a spiral or helical fashion while taking approximately 1000 sinograms (one dimensional X-ray images) per second from all angles. Spatial reconstruction algorithms, such as the filtered backpropagation, are then applied to combine these sinograms to image slice.

Stacking these images slices along the third dimension then reconstructs a three dimensional volumetric image [14].

Consequently, taking this large amount of X-ray images over a longer time also increases the radiation exposure dose experienced by the patient and increases the

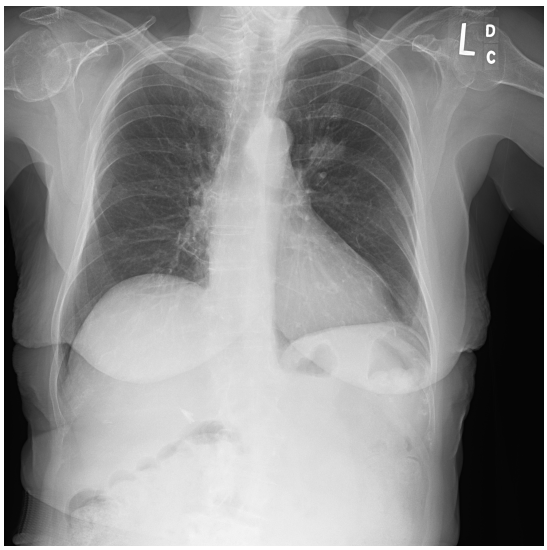


Figure 2.2: Example of a thorax X-ray image [16].

chance of ionizing damage [1, 2]. Much ongoing research is devoted to lowering the radiation dosage as much as possible [17, 18].

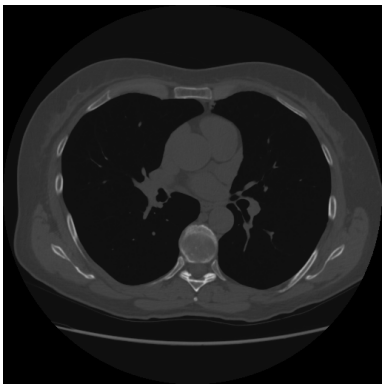
A reconstructed CT image provides more benefits as opposed to a plane X-ray image. First of all, the position of all tissue and organs can be accurately displayed in a three dimensional space without any blocking. As a three dimensional volumetric image, it can also provide different view options, i.e. axial, coronal, or sagittal planes (multiplanar reformatting). As described earlier, different tissues consist of different water and fat ratios and therefore attenuate differently. On the Hounsfield scale, which describes the linear attenuation coefficient of a substance in relation to the radiodensity of distilled water (defined as 0) and air (defined as -1000) at standard pressure and temperature, these tissues occupy different ranges of values. Thus, a CT image can also be windowed to a specific range of Hounsfield units to increase the contrast for explicit tissues [14].

CT scans are used to diagnose soft tissues such as blood vessels, organs, abnormal brain functions, but also for bones, joints and spinal conditions as well. Examples of a CT scan can be seen in Figure 2.3.

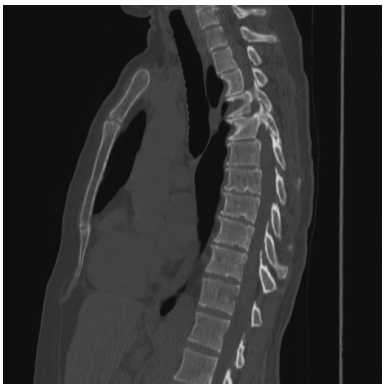
2.2 Deep Learning

Deep learning, also known as deep neural networks, is a subfield of machine learning based on multi-layered artificial neural networks. Recently, deep neural networks have received a lot of attention because of their ability to learn complex relationships, feature extracton and representation learning. The method has become popular across disciplines, such as computer vision [19, 20], autonomous driving [21], natural language processing [22, 23] and bio-medicine [24, 25].

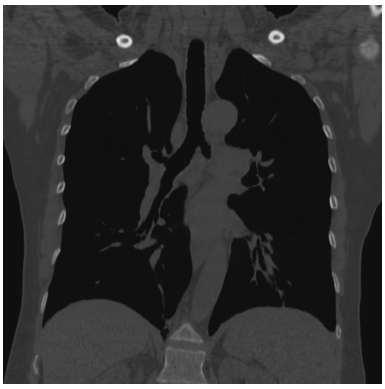
It can be divided into many subareas, but for the sake of briefness, only GANs



(a) Axial



(b) Sagittal



(c) Coronal

Figure 2.3: Volume slices of a CT scan from three different views [16].

and autoencoders will be covered.

This section aims to give the reader a general understanding of the deep learning network topologies used for the methodology.

2.2.1 Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a framework for generative models where two neural networks compete against each other in a zero-sum game. In general a generator G learns to model the data distribution while a discriminator D distinguishes between the real and generated data distributions. G and D are trained simultaneously in an adversarial training process, where overall objective of GANs is for the generator to capture the data distribution of $p_{data}(x)$.

2.2.1.1 Vanilla GAN

The Vanilla GAN, as proposed by Goodfellow et al., is the very first iteration of GANs.

The generator G is a function that maps a prior random noise vector z to data space p_z as $G(z)$. The randomness of the input vector is necessary to acquire varying samples from the network. G implicitly models the distribution p_g of the real distribution x by drawing samples from z.

The discriminator D sees the synthetic data p_g and the real data x in an alternating fashion. A single scalar is produced by D which represents the probability of the input belonging to the real data rather than p_g. D is trained to maximize a probability that the output of D matches the correct label, while G is trained to minimize it. The overall global optimum exists at $p_g = p_{data}$.

Thus, G and D play a two-player minimax game that can be expressed as

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{y \sim p_z(y)}[\log (1 - D(G(z)))] \tag{2.1}$$

which also describes the adversarial loss function $\mathcal{L}_{GAN}$:

$$\mathcal{L}_{GAN} = \log(D(x)) + \log(1 - D(G(z))) \tag{2.2}$$

To put this into an analogy, suppose there is a forger that makes fake paintings from Van Gogh and there is an art connoisseure that specializes in Van Gogh paintings and just started out in the counter counterfeit business. The forger creates a fake painting tries to sell it. The connoisseure looks at a genuine painting and then looks at the forgery and decides if it looks real or not. Based on this decision the forger improves his forging skills and tries again. The connoisseure, being new to the business, also learns more about distinguishing counterfeits from genuine paintings. If this goes on long enough, in theory the skills of the forger and the connoisseure will reach a point where they become equally competent.

It can be shown that the training criterion allows to recover the data distribution by the generator if G and D are given enough capacity.

During back propagation, the discriminator is first updated after the gradients D are calculated and after that, the gradients of the generator are calculated and G is updated. The procedure using minibatches is outlined in algorithm 1 and a more illustrative explanation is given in Figure 2.4 [8].

In practice, Equation 2.1 is not suited for the training of G since it does not provide sufficient gradients. The term $\log(1 - D(G(z)))$ saturates once D reaches higher accuracy and only delivers weak gradients. Instead, G should be trained to maximize $\log(D(G(z)))$. This provides stronger gradients at all times and does not change the general training behaviour.

In terms of game theory, one player tries to maximize its results while his opponent aims to minimize them. In this game of two players, each continously try to achieve a better outcome for themselves and, in order to do so, undermine their opponent. A stable equilibrium state in a non-cooperative game such as this is called Nash equilibrium [26]. Assuming both players are aware of the opposites optimization strategy, the Nash equilibrium is achieved when both players no longer have anything to gain by changing their strategy. In the game between the generator and the

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples from data distribution $p_{data}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)})))]$$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, ..., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}^{(i)})))$$

end for

discriminator, the Nash equilibrium is reached when the discriminator can no longer differentiate between real and fake inputs.

It can easily be spotted why a minimax game more complicated than just finding the minima. Training GANs is about finding a balance between the skill of generator and disriminator to eventually let them converge on the Nash equilibrium.

It is very easy to upset the balance, which is the reason why GANs are prone to instabilities, such as mode collapse, diminishing gradients or non-convergence while training.

Mode collapse happens when the generator produces an especially plausible output x^*. From the discriminators perspective, the generated output is so good, that it is classified as correct for all modes of the data. Going back to Equation 2.1, it can be seen that in the extreme case where $G(z)$ is always evaluated as real, the second term always becomes zero. What remains is $x^* = argmax_x D(x)$, which is independent from the input vector z. Consequently, gradients associated with z will approach zero over time. The generator gets stuck in a local minima. Usually though, this extreme case is less likely to occur, while a partial mode collapse happens more often, where the generator focuses on a few modes of data to fool the discriminator.

Diminishing gradients occur when the discriminator is much more skilled than the generator, which leads to the generators gradients vanishing over time. The generator is not able to learn anything.

Non-convergence happens when the gradients are too strong and the model pa-

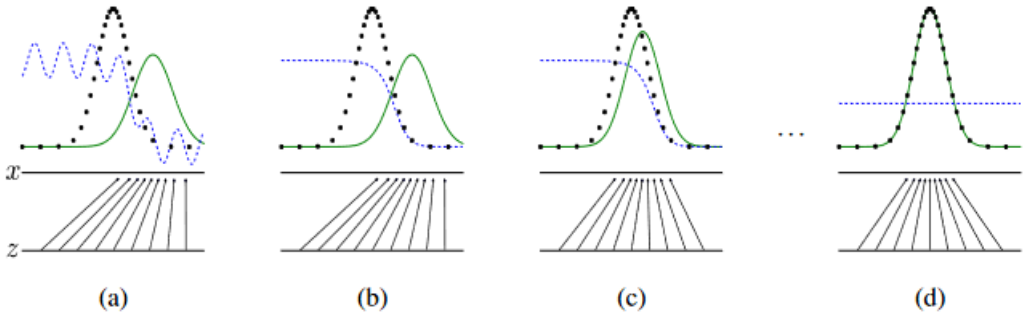


Figure 2.4: Generative adversarial networks are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the generative distribution $p_g(G)$ (green, solid line). The lower horizontal line is the domain from which z is sampled, in this case uniformly. The horizontal line above is part of the domain of x. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifer. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$. (c) After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$. Image and caption taken from [8]

rameters start to oscillate out of control and destabilize. The minimax game never converges.

Albeit sensitive, this adversarial training strategy is also what sets generative adversarial networks apart from other generative models that use direct comparison, such as an L1-Loss, of the generated output with the ground truth distribution. In a way it can be said that the discriminator is actually a neural network modeling a loss function for the generator to enforce the closeness of generated and true distributions. It is especially useful when learning distributions that can not easily be described exhaustively. Furthermore, when synthesizing new distributions, the model needs to generate similar distributions to the one it learned, not exactly the same.

Since its conception, the idea of competing networks spawned a large number of improved or alternative flavors of GANs for different purposes, such as human image synthesis (fake celebs [9]), image translation (pix2pix [27]), multi domain image translation (StarGan [28]), etc. Specifically in the field of medical imaging they have also demonstrated their effectiveness in cross-modality image transfer [3, 29–31] and

denoising [32, 33].

2.2.1.2 Deep Convolutional GAN

Deep convolutional GANs combine GANs with deep convolutional networks to extend GANs for unsupervised learning and image generation [34].

An example of a DCGAN network generator can be seen in Figure 2.5. The authors also make several proposals to improve training stability for GANs:

- Replace any pooling layers with strided convolutions in the discriminator and fractional-strided convolutions for deconvolutions in the generator.
- Avoid using fully connected hidden layers for deeper architectures.
- Use batch-normalization in the generator and discriminator to compensate for poor initialization and improve gradient flow through the network during backpropagation. This also prevents the generator from experiencing mode collapse. However, batch-normalization can lead to sample oscillation and model instability. This is avoided by not using batch-normalization for the generators output layer and the discriminators input layer.
- Use ReLU activations for all layers in the generator except for the output, where a Tanh activation is suggested.
- Use LeakyReLU activations in the discriminator [34].

To further stabilize training, the authors also suggest to use the Adam optimizer instead of SGD as proposed in [8].

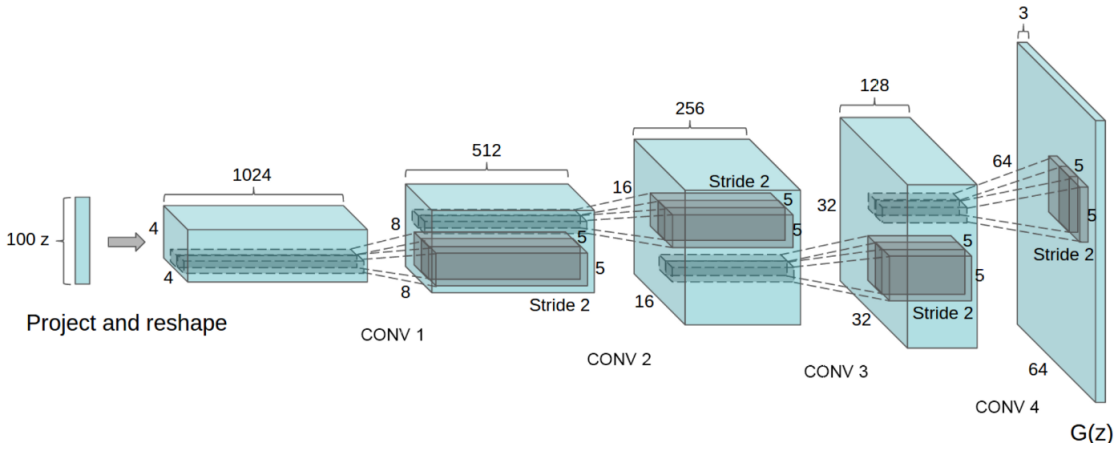


Figure 2.5: Generator structure of a DCGAN network. Image from [34].

2.2.1.3 Wasserstein GAN

Identifying that stability and mode collapse are a problem for GANs, efforts have been made to find a different loss function with better optimization characteristics. One of the propositions is the Wasserstein GAN which uses the Wasserstein distance as a loss function.

The Wasserstein distance (or Earth Mover distance) is a metric for the similarity of two distributions. Given the real distribution $\mathbb{P}_r$ and the generated distribution $\mathbb{P}_g$, the Wasserstein distance for both distributions is the minimum cost of mass transportation for transforming P_r into P_g. This can be expressed as the greatest lower bound for any transport plan as shown in Equation 2.3 [35].

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{2.3}$$

It can be shown that modifying Equation 2.1 to

$$\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p_{data}}[D(x)] + \mathbb{E}_{y\sim p_z(y)}[D(G(z)))] \tag{2.4}$$

is the equivalent of minimizing the Wasserstein distance in Equation 2.3, provided that the discriminator function f fulfills the Lipschtiz-constraint:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2| \tag{2.5}$$

The derivation is shown in [35]. The modified discriminator is now called a critic to differ with the previous discriminator.

To enforce the Lipschtiz-constraint, the authors initially propose to apply weight clipping on the weight of the critc's weights after updating the weights 2.3. Weight clipping is obviously a bad choice because it reduces the capacity of the critic and also weakens its ability to differentiate between real and fake images. It also introduces another hyperparameter c:

$$w \leftarrow clip(w, -c, c) \tag{2.6}$$

The model performance is very sensitive to c. Since the critic now gives a score with high values for real and low values for fake images, the output is not bound to the range [0,1] anymore. The critic is able to scale up its weights by a factor to improve the loss but not changing the actual gain of distinguising the images. Weight clipping limits the ability of the critic to scale its weights. The weights are only scaled for some time until saturation is reached. If c is too large, it can take a long time for the critic to be trained to optimality because the weights don't reach saturation. On the other hand, if the clipping parameter is too small, the critic does not learn properly and it can lead to vanishing gradients when the network is too deep.

Intuitively, using the Wasserstein distance as the new loss function, the critic now gives a score describing the realness of the generated image and the original image, whereas the previous discriminator gave a probability of the generated image being real or fake. The authors suggest to use the RMSProp optimizer with a learning

rate of 0.00005. $c = 0.01$ is proposed for the clipping range and an update ratio of $n_{critic} = 5$. The training procedure is shown in Algorithm 2.

Algorithm 2 Wasserstein GAN training with RMSProp. Suggested settings: learning rate $\alpha = 0.00005$, $c = 0.01$, $n_{critic} = 5$

Require: α, the learning rate. c the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: $\theta_{d,0}$, initial critic parameters. $\theta_{g,0}$, initial generator's parameters.

while θ has not converged **do**

for $t = 0, \ldots, n_{critic}$ **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples from data distribution $p_{data}(\mathbf{x})$.

$g_{\theta_d} \leftarrow \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [D(\mathbf{x}^{(i)}) - D(G(\mathbf{z}^{(i)}))]$

$w \leftarrow w + \alpha \cdot RMSProp(w, w_{\theta_d})$

$w \leftarrow clip(w, -c, c)$

end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.

$g_\theta \leftarrow -\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} D(G(z^{(i)}))$

$\theta_g \leftarrow \theta_g - \alpha \cdot RMSProp(\theta_g, g_{\theta_g})$

end while

As stated by the authors, weight clipping is not a good solution.

In [36], Gulrajani et. al. propose an alternative method to satisfy the Lipschtiz-constraint, gradient penalty 2.7.

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2| \tag{2.7}$$

As proven in [35], a differentiable function f is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere. Points interpolated between the real and generated data should therefore have a gradient norm of 1 for f. Accordingly, the model is penalized if the gradient norm moves away from its target norm value of 1, instead of clipping the gradients. The modified loss function is shown in Equation 2.8, with λ as the gradient penalty coefficient.

$$\begin{aligned} \min_G \max_D V(G, D) =& \mathbb{E}_{x \sim p_{data}}[D(x)] + \mathbb{E}_{y \sim p_z(y)}[D(G(z)))] \\ &+ \underbrace{\lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\|\|\nabla \hat{x} D(\hat{x})\|\|_2 - 1)^2]}_{gradient\ penalty} \end{aligned} \tag{2.8}$$

Examples of convergence failure of the critic can be observed in Figure 2.6(a) and a more illustrative example on how the weights behave is depicted in 2.6(b).

It should also be noted, that the use of batch normalization in the critic is not compatible with gradient penalty because it creates correlations between samples in

the same batch. The gradient penalty is calculated for each individual input and not the entire batch. Batch normalization would therefore render the penalized training objective as invalid. The updated training procedure is shown in Algorithm 3

Algorithm 3 Wasserstein GAN training with gradient penalty and Adam optimizer. Suggested settings: $\lambda = 10$, $n_{critic} = 5$, learning rate $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters, $\theta_{d,0}$, initial generator's parameters $\theta_{g,0}$.
  while θ has not converged **do**
    for $t = 0, \ldots, n_{critic}$ **do**
      for $i = 0, \ldots, m$ **do**
        - Sample real data $\mathbf{x} \sim \mathbb{P}_{\supset \approx \supset}$, latent variable $\mathbf{x} \sim p_g(z)$, a random number $\epsilon \sim U[0, 1]$.

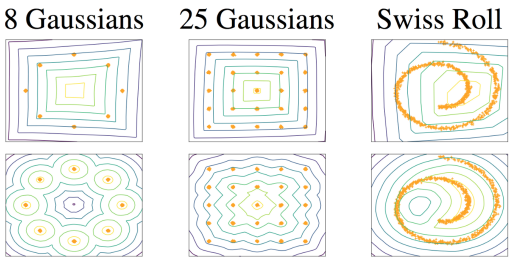
        $\tilde{\mathbf{x}} \leftarrow G_{\theta_g}(z)$
        $\hat{\mathbf{x}} \leftarrow \epsilon \cdot \mathbf{x} + (1 - \epsilon) \cdot \tilde{\mathbf{x}}$
        $L^{(i)} \leftarrow D_{\theta_d}(\mathbf{x}) - D_{\theta_d}(\tilde{\mathbf{x}}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_{\theta_d}(\hat{\mathbf{x}})\|_2 - 1)^2$
      end for
      $\theta_d \leftarrow Adam(\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, \theta_d, \alpha, \beta_1, \beta_2)$
    end for
    - Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.

    $\theta_g \leftarrow Adam(\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} -D_{\theta_d}(G_{\theta_g}(\mathbf{z})), \theta_g, \alpha, \beta_1, \beta_2)$
  end while

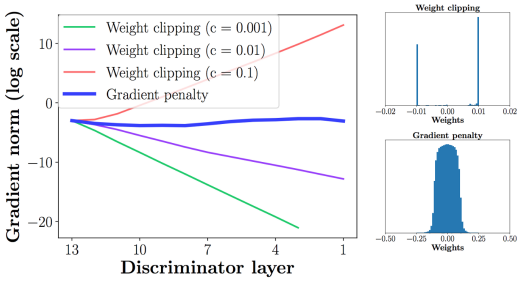
2.2.1.4 Conditional GAN

Vanilla GANs generate new images from a noise distribution without any additional information about the image itself, relying only on the judgement of the discriminator. This poses a problem in that there is no control over what kind of data the network generates. The generator might produce a distribution in the desired domain, but it is going to be completely random. The MNIST dataset for example contains images of hand written numbers ranging from zero to nine. A GAN trained to generate images in the domain of this dataset may be able to produce similar images of hand written numbers, but it is never specified which exact number it will generate because the discriminator only makes the distinction if the image does or does not belong in the domain of the dataset.

Conditional GANs solve this by introducing additional information, such as class labels into the training model. The GAN is now trained on the MNIST dataset and additionally, class labels for each number are provided. This way the GAN will be able to generate images of a specific number of the domain, for example, the number '9'. The additional information is not limited to labels, but can also be in the form of an image instead of the random latent vector (z).



(a) Value surfaces of WGAN critics trained to optimality on toy dataset using (top) weight clipping and (bottom) gradient penalty. Critics trained with weight clipping fail to capture higher moments of the data distribution. The 'generator' is held fixed at the real data plus Gaussian noise.



(b) (left) Gradient norms of deep WGAN critics during training on the Swiss Roll dataset either explode or vanish when using weight clipping, but not when using a gradient penalty. (right) Weight clipping (top) pushes weights towards two values (the extremes of the clipping range), unlike gradient penalty (bottom).

Figure 2.6: Images and captions from [36].

Introducing additional conditions can also improve training stability, speed and also quality of the generated images [37].

2.2.2 Cycle-GAN

Cycle-GANs also belong to the set of conditional GANs. A Cycle-GAN couples two GANs together in a cyclic manner for unpaired learning. As previously mentioned, GANs aim to learn to generate a given distribution. Cycle-GANs on the other hand, aim to learn the mapping functions between two domains. Given domain X and domain Y and using images as data, the first generator G transforms an image from domain X to domain Y, $G : X \rightarrow Y$ and the second generator F does the same from domain Y to domain X, $F : Y \rightarrow X$. On both sides, the discriminators D_X and D_Y are trained to distinguish whether an image is synthesized or not, i.e. if it belongs to their respective domain.

This way, two GANs are trained simultaneously and mapping functions in both directions can be learned. A simplified schematic can be seen in Figure 2.7.

After every training step, the adversarial loss from Equation 2.2 is calculated for both GANs as in Equations 2.9 and 2.10. Additionally, two cycle-consistency losses can be calculated. An image transformed from domain X to domain Y and then back to domain X again should be consistent with the original image. Comparing both images yields the forward cycle consistency loss that can be fed back to the network. This can also be done the other way round for domain Y, which is called a backward cycle consistency loss. The equation for both cycle consistency losses is shown in Equation 2.11.

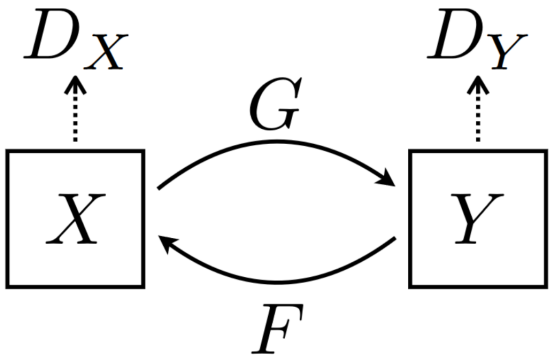


Figure 2.7: Cycle-Gan model with two mapping functions $G : X \longrightarrow Y$ and $F : Y \longrightarrow X$, and associated adversarial discriminators D_Y and D_X. Image and caption from [38].

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y, X, Y) = &\mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))]\end{aligned} \tag{2.9}$$

$$\begin{aligned}\mathcal{L}_{GAN}(F, D_X, Y, X) = &\mathbb{E}_{x \sim p_{data}(y)}[\log D_X(x)] \\ &+ \mathbb{E}_{y \sim p_{data}(y)}[\log(1 - D_X(F(y)))]\end{aligned} \tag{2.10}$$

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = &\mathbb{E}_{x \sim p_{data}(x)}[|F(G(x)) - x|_1] \\ &+ \mathbb{E}_{y \sim p_{data}(y)}[|G(F(y)) - y|_1]\end{aligned} \tag{2.11}$$

The total objective is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{GAN}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{GAN}(F, D_X, Y, X) \\ &+ \mathcal{L}_{cyc}(G, F)\end{aligned} \tag{2.12}$$

Cycle-GANs have found their application especially in image translation or style transfer [38]. A popular example for a Cycle-GAN application is the horse to zebra image translation, where the network learns a mapping function from the horse domain to the zebra domain and back again, as can be seen in Figure 2.8.

2.2.3 PatchGAN

The motivation for the PatchGAN comes from the nature of the L1 loss. The L1 loss is usually chosen as a reconstruction loss for image generation, but it is well understood that it causes blurry results because it only captures low frequencies of the image. The authors of [27] therefore suggest that the discriminator can be designed

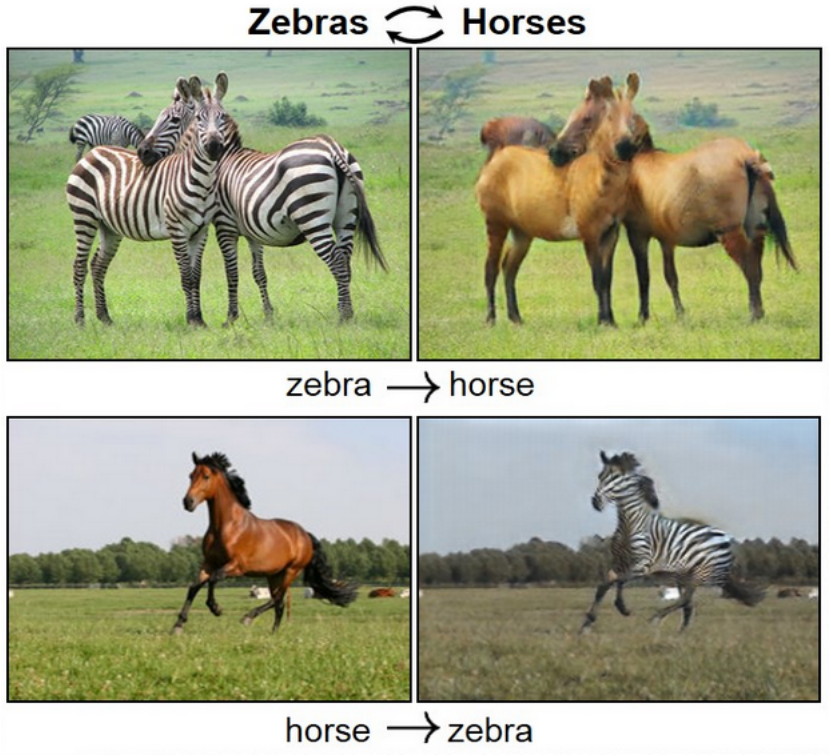


Figure 2.8: Domain transfer, transforming horses to zebras and back. Image taken from [38].

to model only high-frequency structure by restricting its attention to local image patches, instead of the entire image. This way the discriminator focuses on a kernel of NxN only and classifies the local structure as real or fake. The discriminator kernel is run convolutionally across the image and the scores are averaged as the overall output. It is demonstrated that the PatchGan discriminator can still produce high quality results even with N chosen to be much smaller than the original image size. A demonstration of the results using different patch sizes is shown in Figure 2.9. Therefore it can have fewer parameters, run faster and still be applied to arbitrarily large images. It can also be extended to image volumes. The PatchGAN discriminator can also be described as a Markovian discriminator because it effectively models the image as a Markov random field, assuming independence between pixels seperated by more than a patch diameter of N [39–41].

2.2.4 LSGAN

Sigmoid cross entropy loss functions in regular GANs have the tendency to result in vanishing gradients during the learning process. Motivated by this observation, the authors propose the adoption of a least squares loss function for the GAN discriminator instead.

Specifically, minimizing the objective function of regular GAN leads to vanishing

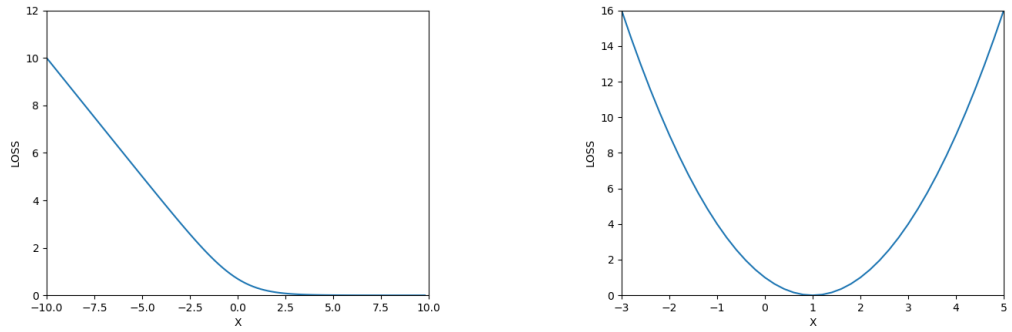
Figure 2.9: Patch size variations. The 1x1 PixelGAN encourages greater color diversity but has no effect on spatial statistics. The 16x16 PatchGAN creates locally sharp results, but also leads to tiling artifacts beyond the scale it can observe. The 70x70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (colorfulness) dimensions. The 286x286 ImageGAN produces results that are visually similar to the 70x70 PatchGAN, but somewhat lower quality according to metrics. Images and Caption taken from [27].

gradients and makes it hard to update the generator, which can be traced back to the use of the cross entropy loss.

The issue with the cross-entropy loss as is used in the original adversarial loss, is their behaviour concerning samples that are 'too correct'. A sample is 'too correct' if it is on the correct side of the decision boundary, but also very far away from it. For image generation purposes, these samples are not desired because the goal is actually to model the distribution of the target and not just classify the samples as right or wrong. Therefore these samples also need to be considered, but a cross-entropy loss does not penalize them enough. In fact, given the nature of a logarithmic function they are usually not penalized at all. As shown in Figure 2.10, positive values that are further away from the decision boundary are not penalized by the cross entropy loss, whereas the least squares loss penalizes both positive and negative values equaly based on their distance to the decision boundary. To deal with these 'too correct' points, the cross-entropy term is replaced by a least-squares term. The least-square loss is purely driven by the distance to the decision boundary. Any sample that is too far away on either side is penalized to encourage the samples to move closer to the decision boundary. It is shown, that the results generated by the LSGAN produces better results quality wise and that LSGANs perform more stable during the training process [42].

2.2.4.1 Progressive Growing of GANs

While GANs are know to produce highly detailed images, their fragile nature limits the image size and resolution it can generate. High resolution images makes it easier for the decoder to tell fake and true images apart, which leads to amplified gradients and makes training unstable. Therefore, to achieve high resolutions outputs from a GAN, a progressively grown GAN architecture is proposed in [43]. The network starts out small with low resolution inputs and as the training progresses, new layers are added to the generator and discriminator to introduce higher-resolution details. The paper shows that the proposed approach greatly improves speed and stability



(a) (b)

Figure 2.10: (a) shows the sigmoid cross entropy loss function and (b) shows the least squares loss function.

in high resolution training and that high resolution outputs at 1024x1024 pixels are possible.

2.2.5 Autoencoder

Autoencoders belong to the group of unsupervised learning techniques and are used for the task of representation learning. This subsection will give an overview of some of the relevent autoencoder varieties.

2.2.5.1 Vanilla Autoencoder

An autoencoder neural network is an unsupervised learning algorithm that learns to encode data into a latent space representation and then also learns to decompress the data to a representation as close to the original as possible [44]. Given an input x, the encoder part represents a function f that transforms the input into a latent space representation $h = f(x)$. The decoder then is a function g that transforms the latent space representation back to reconstruct the input $r = g(h)$. An autoencoder where the dimensions of the latent space are set to be smaller than that of the input data is called an undercomplete autoencoder. This will force the autoencoder to only learn a compressed representation of the original data. The autoencodert will work similar to a nonlinear unsorted principal component analysis.

An undercomplete autoencoder can be used as for denoising an image for example, since it only learns to encode the more important features in the image.

A sparse autoencoder is an undercomplete autoencoder with a sparsity penalty, which encourages fewer nodes in the hidden layer to activate during training. The intuition is that using a lot of nodes to learn the latent representation will result in each node only learning superficial or redundant features of the data. Reducing the number of activated hidden layer nodes while acomplishing the same reconstruction

enforces that these nodes learn more meaningful features from the data. Figure 2.11 shows an example of a convolutional autoencoder implementation.

The difficulty with training autencoders is that the latent space is often very irregular, which means that sampling from a random point in latent space might give very unexpected results. Two close points in latent space can also represent very different distributions and there is often empty space that cannot be decoded into any meaningful distribution. This means that a trained autoencoder cannot guarantee the correct decoding of an unknown input.

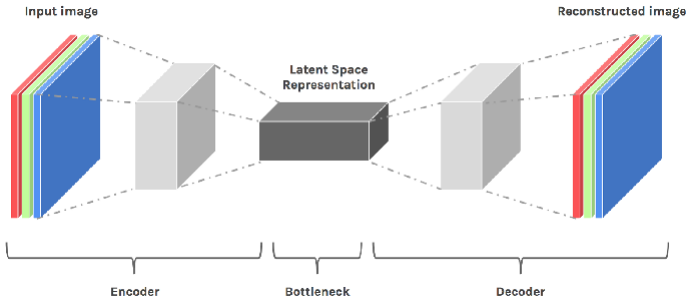


Figure 2.11: Example of an autoencoder network architecture.

2.2.5.2 Variational Autoencoder

Similar to the autoencoder, the variational autoencoder aims to learn a latent space representation of a distribution. To tackle the irregularities in latent space, a prior distribution is imposed upon the latent distribution. The latent space is usually regularized to approximate a standard normal distribution $z \sim \mathcal{N}(0, I)$ [45, 46].

Given an input x, the encoder network infers a parameterized latent representation $q_\phi(z|x)$ of x. The decoder network parameterizes $p_\theta(x|z)$ and generates x. Both networks are jointly trained by minimizing the Kullback-Leibler divergence between $q_\phi(z|x)$ and $p(z)$ and maximizing the expected pixel wise log-likelihood of $p_\theta(x|z)$. This can also be represented as maximizing the Evidence Lower Bound (ELBO) [47], which is also used as the loss function.

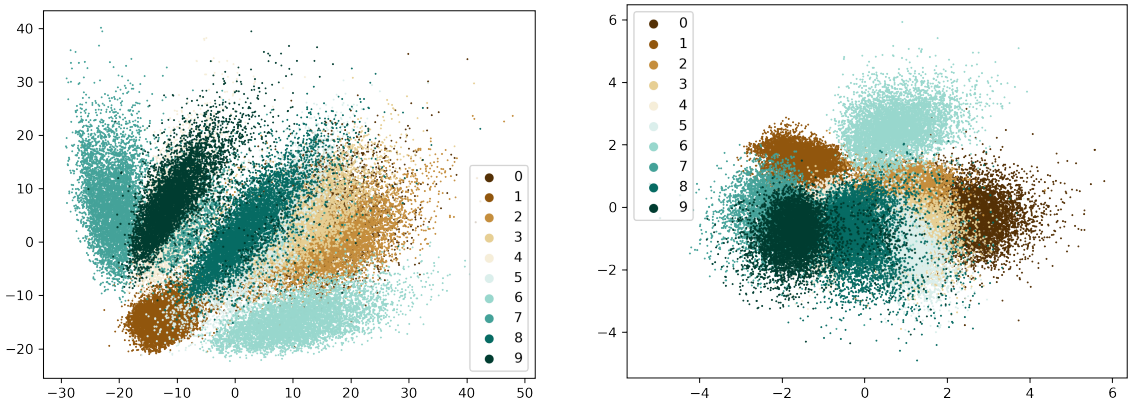
$$\begin{aligned} \mathcal{L}_{VAE} = ELBO &= \mathcal{L}_{llike}^{pixel} + \mathcal{L}_{prior} \\ &= \mathbb{E}[\log p_\theta(x|z)] - KL(q_\phi(z|x)||p(z)) \end{aligned} \tag{2.13}$$

, where $\mathcal{L}_{llike}^{pixel}$ is the pixel wise log-likelihood loss and $\mathcal{L}_{prior}$ is the Kullback-Leibler distance to a prior distribution. Regularizing the latent space at the bottleneck means that the autoencoder is forced to learn more disjunct features into the latent space. Therefore the latent space of a variational autoencoder is generally populated with more meaningful features.

A visualization for the latent distributions of an autoencoder and a variational autoencoder trained on the MNIST dataset can be seen in Figure 2.12. It can

be observed that the AE builds up large, mutually independent clusters over large ranges and that the clusters in the VAE are distributed around the origin (by design). They also build up smaller, spherical clusters that share a similar value range. As opposed to the AE, the distributions are also much more compact for the VAE. This means that the latent space is generally more interpretable since there are less blind spots that are not covered by a distribution and the distributions usually transition smoothly into each other.

However, due to the compact distribution, VAEs generally have a weaker compression ratio than AEs. This is because the network is penalized for learning latent representations which are unlikely to be drawn from a standard normal distribution and therefore the amount of information stored in the latent space is limited.



(a) AE **(b)** VAE

Figure 2.12: The distribution of latent values by label over the whole training data set both for AE (a) and VAE (b). In order to visualize the four-dimensional latent activations, a principal component analysis was applied to transform them into 2D space. Each color represents a different digit from the dataset. The diagram gives insight into the spatial distribution of the latent values. Images and caption taken from [48].

2.2.5.3 Adversarial Autoencoder

The adversarial autoencoder is an autoencoder which is regularized by matching the posterior latent space distribution $q(z)$ with an arbitrary prior $p(z)$ that is imposed on the distribution. The adversarial network is attached on top of the autoencoder as illustrated in Figure 2.13. There are two objectives to be optimized. First the reconstruction error, which the autoencoder tries to minimize. Second is the regularization of the latent space distribution $q(z)$. The encoder of the autencoder also doubles as the generator that tries to ensure the posterior $q(z)$ can trick the discriminator into believing that it comes from the true prior $p(z)$. Thereby guiding the latent space distribution $q(z)$ towards the prior distribution $p(z)$. The discriminator is specifically used to regularize the latent space distribution.

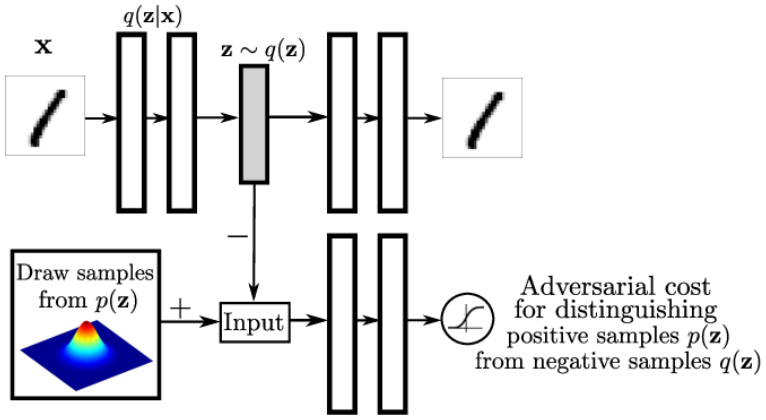


Figure 2.13: Architecture of an adversarial autoencoder. The top row is a standard autoencoder that reconstructs an image x from a latent code z. The bottom row diagrams a second network trained to discriminatively predict wheter a sample arises from the hidden code of the autoencoder or from a sampled distribution specified by the user. Image and caption taken from [49]

In contrast to the VAE, where a KL divergence loss is used to approximate the latent space distribution, an adversarial autoencoder uses the discriminator as a regularization function to impose a prior distribution on the latent space [49].

2.2.5.4 VAE/GAN

VAEs are specifically designed for latent modeling, but are limited by their usually element-wise similarity metric for generating more complex and context driven distributions such as images. Idealy there exists a higher level and sufficiently invariant similarity metric that can represent a given set of images. Rather than manually creating such a metric, a function could be learned to fit the task. This is where GANs come in because the discriminator implicitly has to learn how to describe the features it has to distinguish.

The generator of the GAN is replaced by the VAE as seen in Figure 2.14. The VAE generates the latent space from which the GAN generates the distribution. The discriminator produces a feature-wise error that replaces the element-wise error and helps the VAE learn and encode actual feature representation.

In practice, the authors propose to replace the VAE reconstruction loss with a Gaussian observation model for the discriminator. Let $Dis_l(x)$ represent the hidden representation of the lth layer of the discriminator. Then the observation model becomes

$$p(Dis_l(x)|z) = \mathcal{L}(Dis_l(x)|Dis_l(\tilde{x}), I) \tag{2.14}$$

where $\tilde{x} \sim Dec(z)$ is a sample from the decoder of x.
$\mathcal{L}_{llike}^{pixel}$ from Equation 2.13 thus becomes

$$\mathcal{L}_{llike}^{Dis_l} = -\mathbb{E}_{q(z|x)}[\log p(Dis_l(x)|z)] \tag{2.15}$$

Including the loss function from Equation 2.2, the total loss of the VAE-GAN is represented as:

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{Dis_l}^{pixel} + \mathcal{L}_{GAN} \tag{2.16}$$

It was shown that the combination of VAE and GANs can offer high fidelity image generations and also learning disentangled feature representation [50].

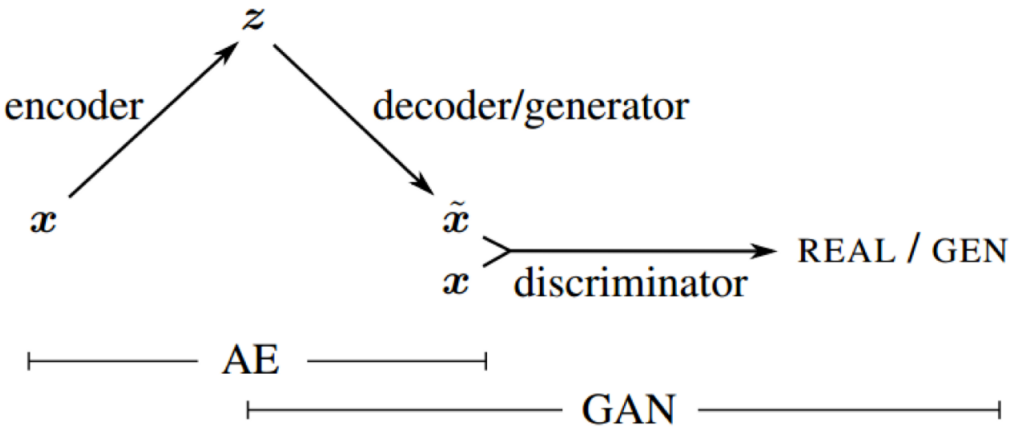


Figure 2.14: VAE-GAN schematic taken from [50].

2.2.6 Checkerboard Artifacts and Resized Convolution

Transposed convolution reconstructs a a higher spatial resolution by running a kernel over a padded input image, similar to a convolutional operation. It is also often called deconvolution. Although this is not strictly accurate mathematically, since a transposed convolution is not the inverse of a convolutional operation, it will still be referred to as such in the following for conciseness.

Checkerboard artifacts are often an unwanted byproduct of deconvolutional layers that appear in a grid-like pattern over a generated image. The reason for the appearence of checkerboard artifacts is often the overlapping of image patches during a deconvolutional operation where the stride does not match the kernel size, as illustrated in 2.15. The network requires additional effort to suppress this behaviour and with multiple deconvolutional layers in a network, pixel overlapping builds up through the network and often has a compounding effect for the artifacts. Resized convolutions are one way to avoid checkboard artifacts. The input image is resized and upsampled through neares-neighbor interpolation before is is passed through a one strided convolutional. The operation avoids pixel overlapping and prevents the occurence of checkerboard artifacts [51].

An example of an artistic style-transfer application with and without resized convolution can be seen in Figure

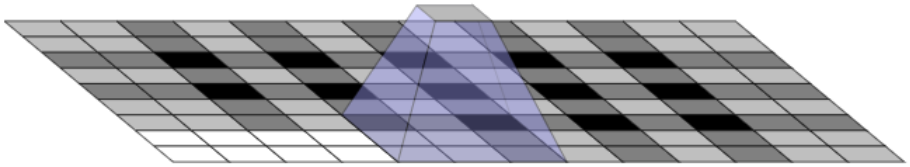
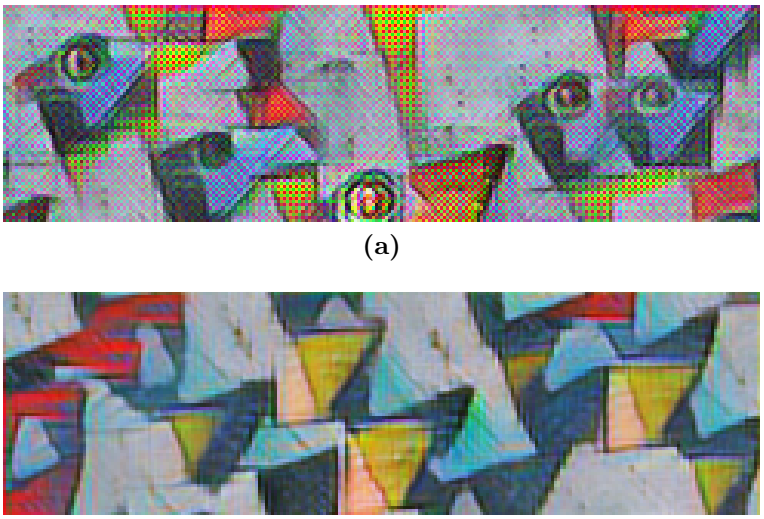


Figure 2.15: Standard deconvolution leads to overlapping with *stride* = 2, *kernel size* = 3, taken from [51].



(a)

(b)

Figure 2.16: (a) shows checkerboard artifacts when using standard deconvolution. (b) shows the same application with resized convolution and no checkerboard artifats.

2.3 Classical Data Analysis Methods

In this section some of the classical data analysis methods are explained to help understand the methods proposed.

2.3.1 K-Nearest Neighbor

K-nearest neightbors (KNN) is a simple algorithm used for a lot of supervised machine learning tasks, dating back to 1970. The main goal of KNN is to find a classification or regression for a queried case on the basis of its proximity to other training samples [52]. Proximity is usually expressed in form of a distance function, such as the Euclidean distance (square root distance between two points), Manhatten distance (distance between two points measured only at axes at right angles),

Minkowski distance (generalization of the two previous distance functions), etc. k represents the number of nearest neighboring samples that factor into the classification or regression. Usually the value for k is determined through examination of the data or retroactively through cross-validation. A large k value can result in a more accurate assignment because it reduces overall noise, but it is not guaranteed. Setting $k = 1$ simply assigns the class of the nearest sample to the queried case.

After initiating the value for k number of samples we're looking for, the distance of each training sample in our data to the case of interest is calculated recursively. A majority vote by class of the closest k samples then determines the class the queried case is assigned. In a regression case the average value of the k nearest samples is assigned to the query.

2.3.2 Principal Components Analysis

Principal component analysis (PCA) is used to reduce the dimensionality of large datasets by transforming the dataset into a vector space with less dimensions while still preserving the most variance possible. The basis vectors that form the new vector space of the dataset are called principle components. It is often used as an exploratory tool to visualize and make large datasets with many dimensions more interpretable.

The principal components of a datset are found by computing the covariance matrix of the dataset and finding the eigenvectors and the corresponding eigenvalues. For a datset of m dimensions, there exist m eigenvector and eigenvalue pairs. Eigenvectors are by definition uncorrelated and orthogonal from each other and therefore the eigenvectors of a covariance matrix describe a new set of orthogonal vectors, the principal components that fits the original dataset. The eigenvalues of each eigenvector are a measure of the amount of variance carried by each eigenvector, which means how much variance of the data can be covered by the single eigenvector. Dividing the eigenvalue of an eigenvector by the sum of all eigenvalues returns the percentage of explained variance of the eigenvector. It is common practice to use the explained variance value to determine how many principle components are retained. The eigenvectors with lesser explained variance are less significant and can be discarded. Finally, the original dataset is transformed into the space spanned by the remaining eigenvectors [53].

It is important to note that the principle components themselves don't have any significance or meaning regarding the original dataset. As described above, variance is the main criterion for principle components, therefore the covariance matrix changes with regards to the units of measurement of the data. Therefore it is common practice to standardize all variables before using the PCA.

2.3.3 Peak Signal-to-Noise Ratio

Peak signal-to-noise ratio (PSNR) is a widely used metric that is usually applied as a quality measurment between the original and a compressed image. It measures

the ratio of the maximum possible value and the amount of distorting noise present in the signal or image that affects the quality. Due to the high dynamic range of signals, it is usually calculated in logarithmic decibel scale. The higher the PSNR score, the better the quality. It is computed as

$$PSNR = 20 \log_{10}(\frac{MAX_f}{\sqrt{MSE}}) \tag{2.17}$$

where MAX_f is the maximum possible value (power) and MSE is the mean squared error over the entirety of the original and the compressed image:

$$MSE = \frac{1}{mn} \sum_{0}^{m-1} \sum_{0}^{n-1} \|f(i,j) - g(i,j)\|^2 \tag{2.18}$$

The main drawback of PSNR as a metric is that it is purely numerical. It does not account for local structures, consistencies, etc. that are usually taken into account by the human vision system, such as the structural similarity index.

2.3.4 Structural Similarity Index

Structural similarity index (SSIM) is a perceptual metric that is used to quantify image quality degradation. SSIM is a full reference metric that requires two image inputs. It cannot judge which image is quality wise superior, but only measures their differences. Therefore the SSIM is usually used to compare a processed image to the original image.

The SSIM represents a different philosophy than the numerical measurment techniques such as PSNR. It is based on the assumption that the human visual system is highly adapted to extract structural information from the viewing field. Image degradation is thus considered as the perceived change in structural information, which is based on the idea that pixels have strong inter-dependencies when spatially close. Therefore a measure of structural information change can provide a good approximation to image distortion perceived by the human eye. The structural similarity measurement is seperated into three factors: luminance, contrast and structure. Assuming that the structural information of objects in an image is independent of illumination and contrast, these two factors must be seperated and measured first before measuring the structural information. A diagram of the measurement system is shown in Figure 2.17.

The SSIM is calculated as follows:

$$SSIM(xy) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.19}$$

where μ_x and μ_y are the mean intensities of image signals x and y and σ_x and σ_y the standard deviations, respectively. $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$ are constants chosen for stability and mathematical properties, with L as the dynamic range of the pixel values and k_1 and k_2 set as 0.01 and 0.03 by default. The derivation is demonstrated in [54].

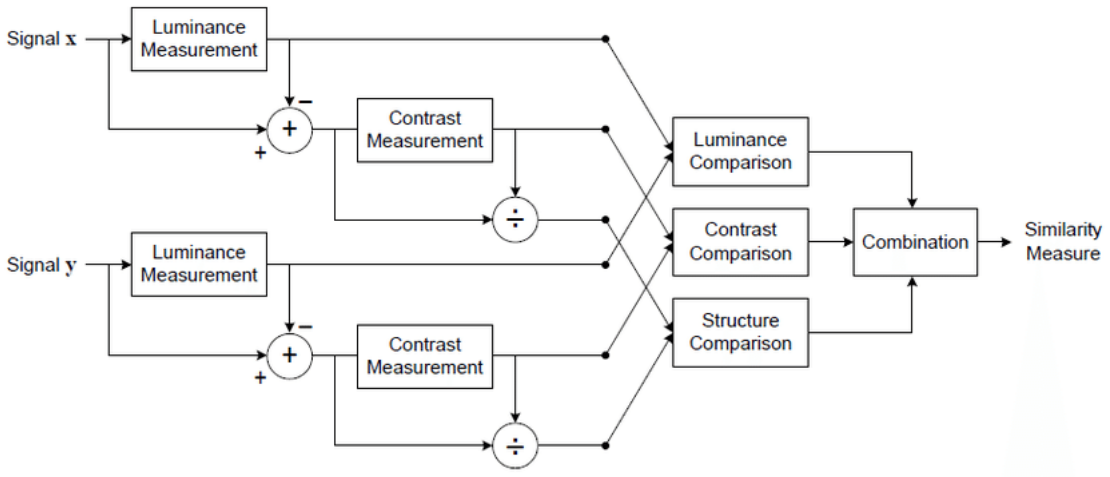


Figure 2.17: Diagram of Structural Similarity Index (SSIM) measurement system. Image and caption taken from [54].

2.4 Related Work

This section will give a brief review of related publications and how they helped in developing the idea for this thesis.

2.4.1 X2CT-GAN

The X2CT-GAN is a recently proposed method to reconstruct a 3D CT volume from two biplanar X-ray images using a generative adversarial network.

As mentioned, GANs are useful for finding a transform function from one domain to another and generate new images in another domain. In this case, the network attempts to learn how the features given by two X-ray images translate into the features from a corresponding CT volume [11].

The proposed design for the GAN generator includes two parallel identical encoder-decoder networks and a fusion network as seen in Figure 2.18. The encoder of one encoder-decoder pair takes one X-ray image as an input and learns the feature map representation through Densenet blocks while at the same time transfering low level structural information through particularly designed skip connections (Connection-B, Figure 3.2(b)) to the decoder. A special connection inbetween the encoder and decoder (Connection-A, Figure 3.2(a)) is also proposed where the two dimensional feature map representation is reshaped into a three dimensional structure to bridge the dimensional differences between encoder and decoder. An additional bridging connection (Connection-C, Figure 3.2(c)) is required to adjust the reconstructed volumes from both encoder-decoder networks and before feeding them into the fusion network. The decoder learns the stepwise deconvolution and reconstruction of the volume from the received representation and incorporates the low level strucural information given by the skip connection in every step. The other encoder-decoder

network works identically. For every step of the reconstruction, the output of both decoders is then given to the fusion network where the volumes are merged together.

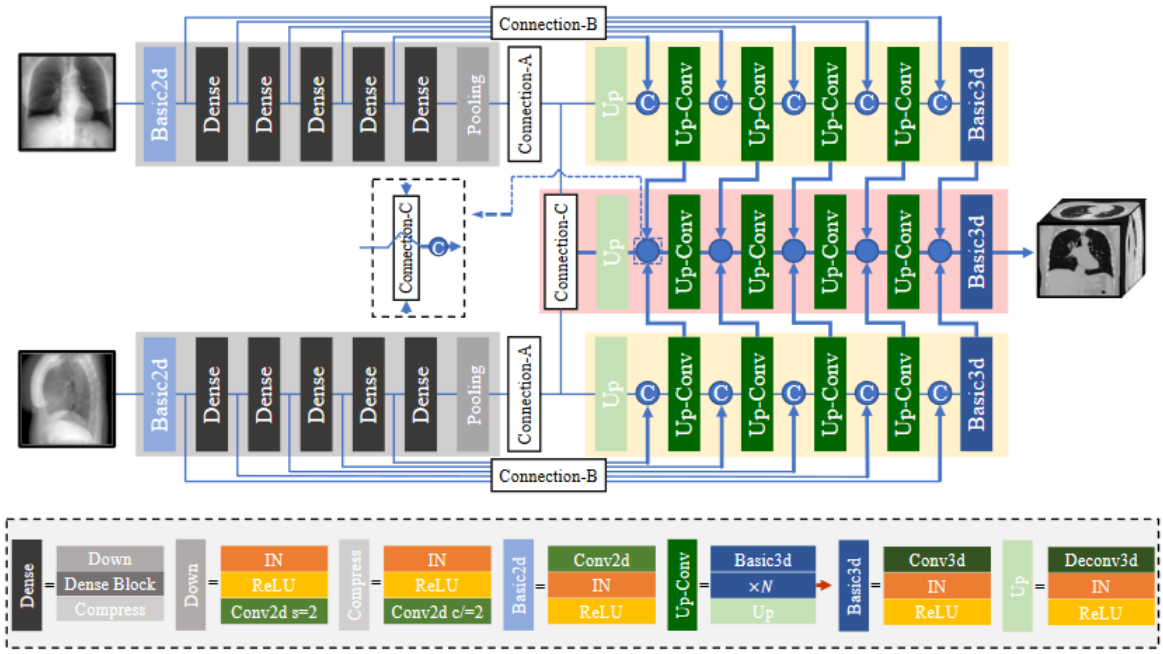


Figure 2.18: Figure taken from [11].

For the discriminator, a convolutional PatchGAN discriminator [38, 55] was used. The volume is divided into patches, and the discriminator classifies each patch individually as real or fake, which has the effect of restricting the classifier attention to the local structure. The PatchGAN's function can also be understood as a texture/style loss [38].

In order to guide the network to achieve the desired reconstruction, it is important to have suitable loss functions. First, a variation of the adversarial loss was used [42], where the logarithmic loss was replaced with a least-square loss. The LSGAN adversarial loss was found to be more useful to retain semantic information and more stabilizing because of the least-square loss. Secondly, the reconstruction loss $\mathcal{L}_{re}$ was introduced to enforce internal structural similarity. It is defined as an MSE loss. Thirdly, the projection loss $\mathcal{L}_{proj}$ inspired by [56] was used to compare the orthogonal projections from three projection planes between the reconstructed CT and the ground truth CT. While the reconstruction loss is a voxel-wise loss to enforce structural consistency, the projection loss focuses on the consistency of the general shape.

They have succesfully shown, that it is possible to reconstruct a three dimensional CT image from two dimensional X-rays with a deep learning GAN network. Ying et al.

2.4.2 PatRecon

An encoder-decoder framework with an additional transformation module at the bottleneck is proposed to generate three dimensional CT images from two dimensional X-ray images. The PatRecon network design is shown in Figure 2.19. A single projection image or a stack of projection images is used as the input for the representation network (encoder) to extract a feature vector from the input. A transformation module reshapes the vector into a three dimensional cube to make the transition from 2D to 3D. The feature cube is then used as the input for the generation network (decoder) to reconstruct the volume via deconvolution. The mean squared error is used as the loss function.

They show that it is possible to reconstruct a volume from a single projection image and that stacking projection images from the same view as input for the network does not necessary improve the results [15]. The network was trained and tested on lung and abdominal CT scans and also head-and-neck CTs. The dataset that was used is not publicly available.

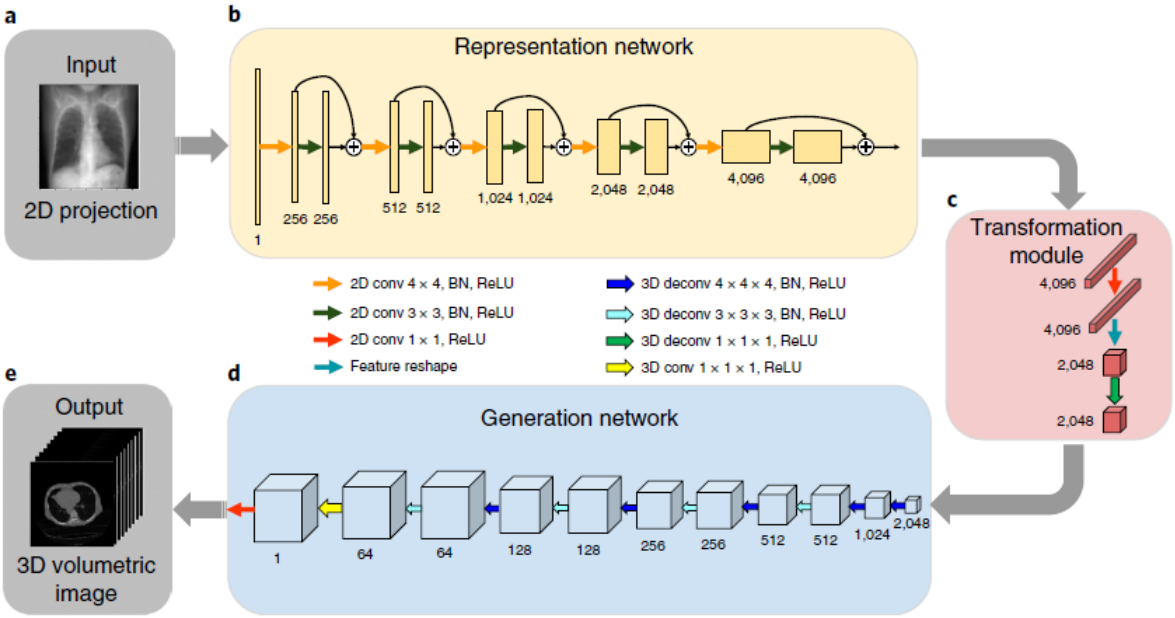


Figure 2.19: PatRecon network architecture. A single X-ray image is used as the input to the representation network with skip connections to extract a feature vector from the image. A transformation module Taken from [15].

3 Methods

The following chapter aims to present the methods and approaches used throughout this thesis. Having surveyed some of the recent publications that deal with the challenge of CT reconstruction, two papers were narrowed down to that come closest to our task while using different network designs, namely X2CT-GAN [11] and PatRecon [15]. It is possible to directly compare the results of both publications because they both worked on datasets of the same region, lung and abdominal CTs, and provided similar images of their results and metrics for quantization. Metric-wise the X2CT-GAN scored better in both SSIM and PSNR. This is also reflected in the resulting images which look sharper and richer in detail in general. One reason this can be attributed to is the use of only one X-ray view as input, as opposed to using two orthogonal views. In our opinion the information provided by two orthogonal views is significantly higher than a single view because they are able to represent the 3D volume in a more accurate way and offset the lack of depth information of a projected image. While two orthogonal views are more advantageous than a single view, I believe that adding a third view does not provide as much information gain as the second view did. Another reason for the lesser quality might be due to the overall simpler architecture, which may simply not be deep enough for the complex problem.

Admittedly, this kind of statement may be subjetive and the results by PatRecon were also not available in higher resolution for a more meticulous comparison. I nevertheless believe that the X2CT-GAN design offers a more solid foundation for our work.

In the first part, I will present the basic network architecture and the improvements that were made to achieve higher image quality for the outputs. The second part is dedicated to solving the problem of single image inference and the modifications that were made to implement it.

3.1 Basic Network Architecture

The goal is the generation of 3D CT images from 2D X-ray images. This means a conditional GAN is going to be trained that receives a priori information through the input images. The network can be seen as a variation of an autoencoder-GAN hybrid.

3.1.1 Generator

The generator is divided into two stages. The first stage contains two encoder-decoder networks in parallel with skip connections. The second stage is a fusion network that combines the generated volumes from both decoders in the first stage to reconstruct a CT Volume. The schematic for the generator is shown in Figure 3.1.

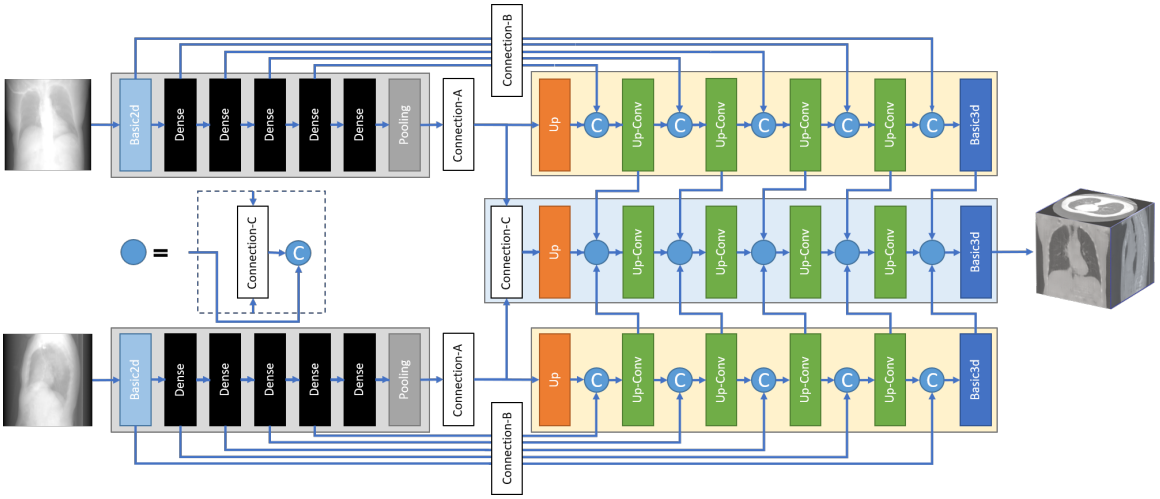


Figure 3.1: Schematic of the basic generator structure.

The sub-networks are linked via the special modules Connection-A, Connection-B and Connection-C, each designed to improve the transition from 2D to 3D. The modules are explained in more detail in 3.1.1.3. The details of the different layers composing each higher level block and lower level blocks are shown in Tables 3.1 and 3.2, respectively.

3.1.1.1 Encoder-Decoder

The encoder-decoder network can be split into two parts, the encoder and the decoder. The encoder contains one Basic2d block, five dense blocks, and a max pooling layer, where the dense blocks are inspired by the DenseNet architecture in [57]. The dense connectivity in the DenseNet blocks provides a compact and efficient way of extracting important features from the X-ray images and cascading them enables the encoder to learn features of different information levels. This is also useful, as that low level information can be shuttled into the volume reconstruction process via the Connection-B skip modules.

The decoder has the task of reconstructing the volume from the learned feature latent space provided through Connection-A with cascading deconvolutional Up and Up-Conv blocks 3.1, mirroring the convolutional blocks of the encoder. Inbetween each deconvolutional stage, low level information from the skip connections are integrated into the partially reconstructed volume before going through the next

deconvolutional block. The last stage in the decoder is a Basic3d block and the result is passed to the last stage of the fusion network.

3.1.1.2 Fusion Network

The fusion network brings the reconstructed three dimensional volumes from both encoder-decoder networks together.

The initial input of the fusion network is the combination of the latent space representations taken from both encoders through Connection-A and combined through Connection-C.

It is assumed that the biplanar X-rays were captured within a very small time frame so that a potential data shift is negligable. Therefore Connection-C can simply average both volumes after transforming them into the same coordinate space.

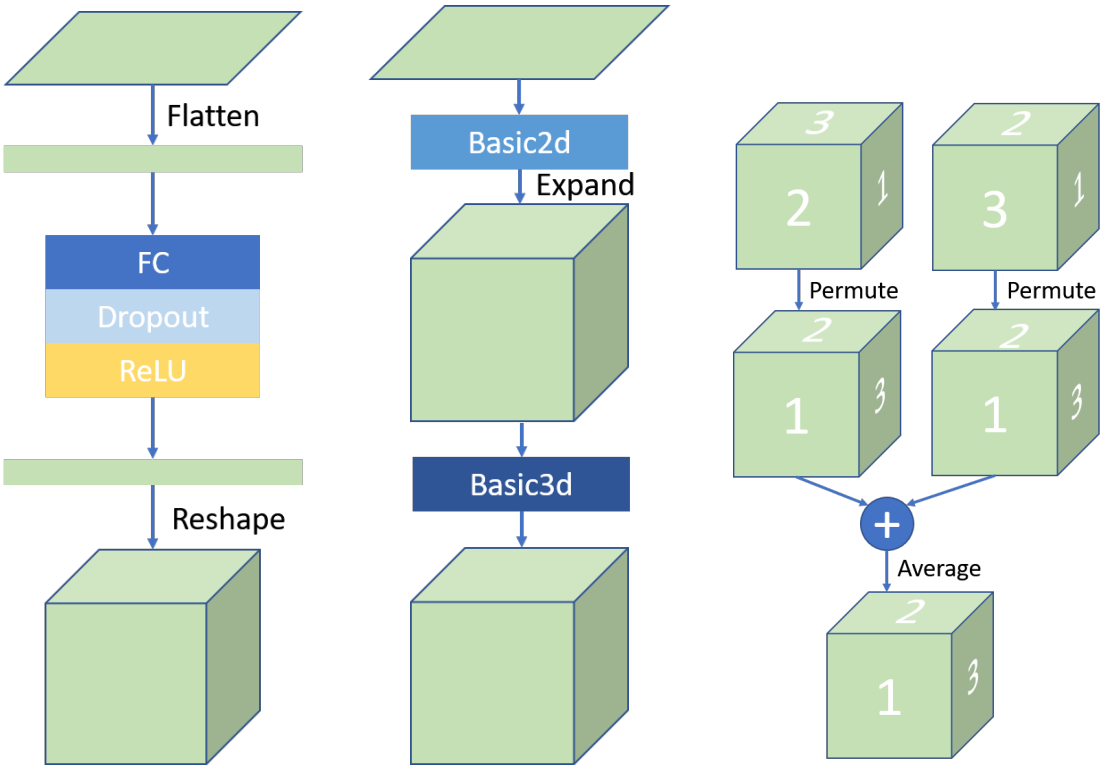
Similar to the decoder, the fusion network is built by cascading deconvolutional Up and Up-Conv blocks, ending with one Basic3d block. Inbetween every deconvolutional stage, the fusion network receives additional partially reconstruced volumes from both decoders that are integrated with the reconstructed volume itself produced. The last reconstructed volume is passed through the Basic3d block and outputs a reconstructed CT.

The fusion network is able to capture any structural inconsistencies from the decoded volumes and backpropagates them to the seperate networks. Therefore the network is vital for the reconstruction process.

3.1.1.3 Bridge Connections

Connection-A and Connection-B are both 2D to 3D bridge connections that link the bottleneck of the autoencoder to the decoder. Both connection modules are designed to convert as much useful information to the decoder as possible. Connection-A (Figure 3.2(a)) is used exclusively for the bottleneck between encoder and decoder. It flattens the input image before converting it to a 3D volume through a fully connected layer and a reshaping operation [11], [58]. Although the features can be succesfully converted from 2D to 3D, the spatial information is lost. It can be argued, that the features encoded into the latent space before going through Connection-A are already disentangled and not spatially bound, which makes it unnecessary to preserve the spatial information. Connection-B (Figure 3.2(b)) on the other hand expands the incoming image artificially by stacking its duplicates along the z-axis. The Basic2d module is used to enfore channel numbers to match the channels at the same stage at the encoder and the Basic3d module encodes the pseudo-3D feature map. Unlike Connection-A, Connection-B shuttles low-level information to impose strong correlations on the shape and appearence between input and ouput. This is the main reason that a stacking operation is used to create the volume. Connection-B is used exclusively for the skip connections. Connection-C (Figure 3.2(c)) is used to fuse the 3D volumes that were generated by the two decoders. Since the 3D volumes were generated from different views, their orientations are different.

Therefore, they have to be rotated the same way before they are given to the fusion network for the next deconvolutional stages. It is also used to generate the initial input for the fusion network from the reshaped volumes generated by Connection-A.



(a) Connection-A **(b)** Connection-B **(c)** Connection-C

Figure 3.2: Illustration of the three different bridge connection modules.

3.1.2 Discriminator

PatchGAN discriminators have had a lot of success with volumetric images. I employ a similar implementation as in [27] [39] for our discriminator.

The PatchGAN discriminator divides the volume into smaller patch volumes that each receive a score. Penalties on the structure are thus only given on the scale of the volume patch, instead of the entire image. A precursor of the PatchGAN with similar design was originally proposed to capture local style statistics [39]. As such, the PatchGAN can also be understood as a form of texture or style loss and has been proven to be effective in capturing localized characteristics of the image while also improving the discriminative capacity of the network.

3.1.3 Loss Functions

3.1.3.1 Adversarial Loss

The performance of the generator and discriminator are judged by the output of the discriminator, as described in 2.1. The generator is rewarded if the discriminator categorized the fake image as true and penalized otherwise. Similarly, the discriminator is penalized if the decision was correct and otherwise if it was incorrect. In order to promote consistency between the semantic information taken from the input X-rays and the generated CT volume, a different formulation of the adversarial loss derived from LSGAN [42] was used. It was shown that a least-squares adversarial loss achieves more realistic image details and improves training stability.

$$\mathcal{L}_{LSGAN}(D) = \frac{1}{2}[\mathbb{E}_{y\sim p(CT)}(D(y|x)-1)^2 + \mathbb{E}_{x\sim p(Xray)}(D(G(x)|x)-0)^2] \tag{3.1}$$

$$\mathcal{L}_{LSGAN}(G) = \frac{1}{2}[\mathbb{E}_{y\sim p(Xray)}(D(G(x)|x)-1)^2] \tag{3.2}$$

3.1.3.2 Reconstruction Loss

A combination of adversarial loss and reconstruction loss has proven to be helpful for image generation [10]. While the adversarial loss tries to construct a realistic prediction, it is not guaranteed to learn the correct internal structure of the input. To improve upon the structural consistency of the generated CT volume, a voxelwise reconstruction loss is included, which is implemented as a MSE loss:

$$\mathcal{L}_{re} = \mathbb{E}_{x,y}||y - G(x)||_2^2 \tag{3.3}$$

3.1.3.3 Projection Loss

To enforce the three dimensional structure of the reconstructed CT, shadow projections in three directions are compared with an L1 loss. The shadow projections are inspired by [56] where a similar technique was used for single view 3D object reconstruction. For each volume, all voxel values are summed up in the direction of one of the axis. For the resulting two dimensional image, all pixels over a certain threshold are set to 1 and those below are set to 0. The intention is to help the network learn an overall consistant shape and enforce sharper image boundaries through the L1 loss. The shadow projection loss is formulated as:

$$\begin{aligned}\mathcal{L}_{pl,sp} = \frac{1}{3}[&\mathbb{E}_{x,y}||P_{sp,ax}(y) - P_{sp,ax}(G(x))||_1 + \\ &\mathbb{E}_{x,y}||P_{sp,sa}(y) - P_{sp,sa}(G(x))||_1 + \\ &\mathbb{E}_{x,y}||P_{sp,co}(y) - P_{sp,co}(G(x))||_1]\end{aligned} \tag{3.4}$$

where $P_{sp,ax}$, $P_{sp,sa}$ and $P_{sp,co}$ represent the shadow projections in the axial, sagittal and coronal view, respectively.

3.1.3.4 Total Loss

With the individual loss functions defined, the final objective function for generator and discriminator are described as:

$$D_{total} = arg\min_{D} \lambda_1 \mathcal{L}_{LSGAN}(D) \tag{3.5}$$

$$G_{total} = arg\min_{G} [\lambda_1 \mathcal{L}_{LSGAN}(G) + \lambda_2 \mathcal{L}_{re} + \lambda_3 \mathcal{L}_{pl,sp}] \tag{3.6}$$

with λ_1, λ_2 and λ_3 as the weights for each loss, respectively.

3.1.4 Improved Projection Loss

It was mentioned in [11] that the projection loss did not make a very significant impact on the overall performance of the network in comparison to the reconstruction loss. I believe that the reason is because the projection used did not provide enough useful information for training.

Intuitively, I believe that a full DRR projection can offer a more considerable impact on the training performance. A full DRR projection provides similar contour characteristics as well as more detailed information on the inner structure of the CT image.

Since the DRR projection is also used for X-ray synthesis, the concept is similar to the cycle-consistency loss from [38]. An X-ray transformed into the CT domain and projected back again should result in the same X-ray image as the original input.

The new loss function is calculated same as above using the L1 distance, which is then averaged over all views:

$$\begin{aligned} \mathcal{L}_{pl,drr} = \frac{1}{3}[&\mathbb{E}_{x,y}||P_{drr,ax}(y) - P_{drr,ax}(G(x))||_1 + \\ &\mathbb{E}_{x,y}||P_{drr,sa}(y) - P_{drr,sa}(G(x))||_1 + \\ &\mathbb{E}_{x,y}||P_{drr,co}(y) - P_{drr,co}(G(x))||_1] \end{aligned} \tag{3.7}$$

where $P_{drr,ax}$, $P_{drr,sa}$ and $P_{drr,co}$ represent the DRR projections in the axial, sagittal and coronal view, respectively.

3.1.5 Variational Adversarial Autoencoder

Variational autoencoders have been shown to successfully create a very interpretable and regulated latent space. By reparameterizing the latent space and enforcing a prior distribution as regularization term the variational autoencoder is able to learn disentangled feature representation in latent space. Since the encoder-decoder

architecture is very similar to an autoencoder, this can leverage and a parameterized and regularized latent space can be introduced for the network.

Suppose the input of the network is x and the encoder preceding the module is represented as $g_1(x)$. First the input has to be encoded into a gaussian distribution over the latent space. This is realized through two convolutional layers, each representing the functions $h_2(x)$ and $g_2(x)$ that map x to the two parameters of a gaussian distribution μ_x and σ_x:

$$\begin{aligned} g_1(x) &= h_1(x) \\ \mu_x = g(x) &= g_2(g_1(x)) \\ \sigma_x = h(x) &= h_2(h_1(x)) \end{aligned} \tag{3.8}$$

Subsequently, one point from the distribution is sampled as the latent representation z and passed on as the output. However, the sampling step is not straight forward. Using μ_x and σ_x directly as parameters for a Gaussian distribution such that $z \sim \mathcal{N}(\mu_x, \sigma_x)$ prevents backpropagation during training because it is not differentiable. In this case the reparametrization trick can be used and the expression can be changed to $z = \sigma_x \zeta + \mu_x$ with $\zeta = \mathcal{N}(0, I)$. This makes gradient descent possible because it is not necessary to perform backpropagation on $\mathcal{N}(0, I)$ and μ_x and σ_x are used in a linear combination.

To complete the variational autoencoder two loss functions are needed, namely a reconstruction term and a regularization term. The reconstruction term is the same as the already existing reconstruction loss. The regularization term is implemented as a Kullback-Leibler divergence loss $\mathcal{L}_{KL}$.

The Kullback-Leibler divergence is a measure of similarity between two distributions over a certain region. It is also called the earth-mover distance because it measures the cost to transform one distribution to the other if both distributions were represented as mounds of dirt.

Typically, the standard Gaussian distribution is chosen as a prior because it is desireable for the latent distribution to be complete and continuous throughout the latent space.

To implement the variational autoencode, the Multivariate Gaussian module (Figure 3.3) is added and position right before Connection-A. It contains a parameterizing block that contains two convolutional layers and a sampling function for reparametrization. The modified design is shown in Figure 3.4.

The Kullback-Leibler divergence loss in Equation 3.9 as suggested by [46].

$$\begin{aligned} \mathcal{L}_{KL} &= KL(\mathcal{N}(\mu_x, \sigma_x), \mathcal{N}(0, I)) \\ &= -\frac{1}{2} \sum (1 + \log \sigma_x^2 - \mu_x^2 - \sigma_x^2) \end{aligned} \tag{3.9}$$

For the network, additional considerations for the skip connections have to be made. Initially I did not want to include a latent space regularisation term for the

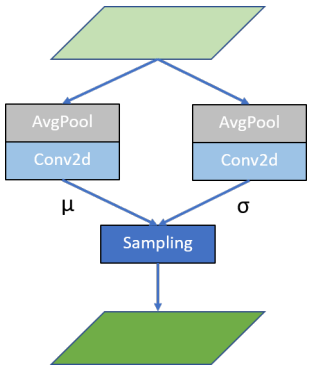


Figure 3.3: Multivariate Gaussian module.

skip connections because they are supposed to transfer low level information to the decoder. This means that they are not fully encoded and regularized by design. Adding a KL-regularization term would contradict this notion. There is also no guarantee that the spaces can converge to the prior because the spaces are not encoded or partially encoded, especially for the skip connections in the early stages of the network. In conclusion, this means it would increase the complexity of our training because of five additional loss functions and also introduce more instability by trying to regularize unencoded spaces, while going against the initial design of the skip connections. However, this particular latent space regularization is still going to be implemented in Section 3.2 for single image inference.

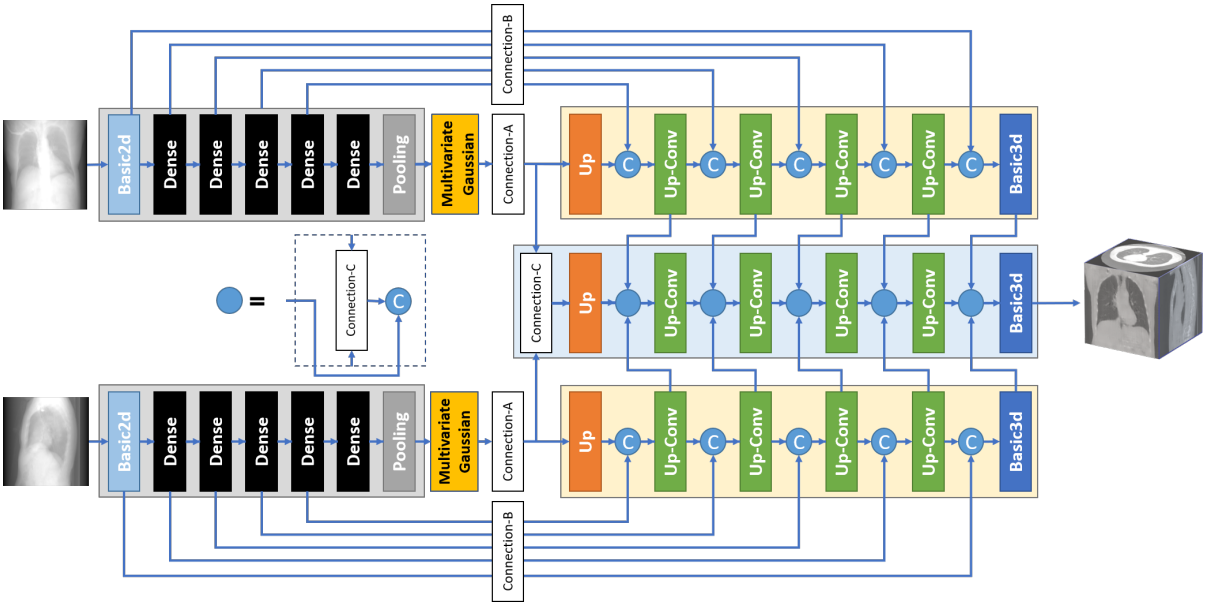


Figure 3.4: Generator architecture with multivariate Gaussian latent spaces.

3.2 Single Image Inference

One major goal of this work is to enable single image inference for our model. The reasoning behind this is that while one X-ray image is easy to obtain, the threshold of acquiring a second one for a perpendicular view is significantly higher for some use cases. Projection images from the axial view contain more useful information than the coronal view, but is also an X-ray image that can not be obtained through conventional X-ray imaging due to physical limitations. To make the application of the model simple and convenient, permitting inference from only one input image is an important step.

However, the architecture as described in Section 3.1.5, Figure 3.4 was designed for training with two input images, images x and y. Dropping one input image would mean that half of the network will miss important feature information for reconstruction.

Implementing a seperate network was considered to achieve the task, such as adding a seperate Cycle-GAN that generates the second view y from the main input x, or an additional variational autoencoder to setup a teacher-student system. I ultimately decided against it because Cycle-GANs have already been proven to perform poorly on geomerical transfer tasks and implementing a teacher-student architecture would be beyond the scope of this work.

Therefore the efforts were concentrated on the existing design and attempt to achieve single image inference in one training cycle.

3.2.1 KNN Prediction

The KNN prediction method is based on the k-nearest neightbors algorithm and requires a look-up table for reference.

First, the generation of a latent space look-up table for the latent space representations $p(z|x)$ of the first view is implemented. In general, the encoder learns the features of the image and encodes it into a latent representation in feature space. After learning through sufficiently enough epochs, it can be assumed that the feature space provided by both encoders has reached a stable state so that similar input images are encoded in a similar region, forming clusters. $Encoder_A$ encodes input x into the latent representation $p_A(z|x)$.

During training, each input image pair generated from the same CT image is tagged with an index i. The encoded latent representations from $Encoder_A$ is extracted during training. This constructs a lookup table that associates the latent space representations of all training images $\Sigma p_A(z|x_{train,n})$ with the corresponding training images of the second view $\Sigma y_{train,n}$.

During inference, the latent representation $p_A(z|x_{in})$ of the unknown input image x_{in} is first generated with a decoupled $Encoder_A$. $p_A(z|x_{in})$ should fall within the cluster of the latent representations $\Sigma p_A(z|x_{train,n})$. Now the k-nearest-neighbor algorithm can be used to find the latent representation $p(z|x_{train,i})$ that lies nearest to $p(z|x_{in})$.

The k-nearest neighbor algorithm is usually used to find the classification or regression of a queried sample by taking the majority vote of the k nearest training samples.

In this case, the KNN algorithm is simply used to find the closest sample in latent space, which is equivalent to setting $k = 1$.

Because of the close proximity in latent space of $p(z|x_{train,i})$ and $p(z|x_{in})$, it can be expected that $x_{train,i}$ is feature-wise the most similar to x_{in}. Therefore, the complementary $y_{train,i}$ should, feature-wise, also be very similar to y_{in}. The look up table can hence be used to find $y_{train,i}$ and use it as a supporting input image with x_{in} to make a full CT reconstruction.

However, this method heavily depends on the completeness of the training dataset and it has to be guaranteed, that there exists an image pair within the training dataset that is feature-wise very close to the testing image. The network is also not trained to compensate for the offset between y_{in} and $y_{train,i}$. Additionally, the training dataset has to be provided with the model to make inference possible, which makes it substantially less convenient.

3.2.2 PCA Prediction

The PCA prediction method also builds upon the latent space representations of the X-ray images. This time an accurate representation of the second input image $p(z|y_{in})$ is computed through a principal component analysis [53] and a linear combination in latent space. This approach entails, that $Encoder_B$ can be completely left out during inference time. All the inputs for $Decoder_B$ are predicted from $p(z|x_{in})$.

To make a linear combination in latent space plausible, the representations for each input image first need to be encoded in the same latent space. This is achieved by sharing the weights of $Encoder_A$ and $Encoder_B$. During training, this means that both encoders learn to encode their respective inputs into the same feature space. In a fully disentangled feature space, each dimension represents a different distinct characteristic of the image. With weight shared encoders, it follows that the feature space finds characteristics that can be used to describe both X-ray views X and Y. As a crude example, the thickness of a certain vertebra could represent an important feature for both views that is encoded into features space. A different characteristic such as the width of the collar bone may only be an important feature for X, but not for Y and vice versa. In feature space therefore, all representations of view X and all representations of view Y will occupy seperate subspaces with common feature dimensions and dimensions that important for one, but irrelevant for the other.

First, the clusters in feature space are extracted. After training is finished, the set of latent representation for x_{train} and y_{train} can be obtained as

$$z_A = \{p_A(z|x_{train,0}), p_A(z|x_{train,1}), \ldots, p_A(z|x_{train,n})\} \text{ and}$$
$$z_B = \{p_B(z|y_{train,0}), p_B(z|y_{train,1}), \ldots, p_B(z|y_{train,n})\},$$

respectively.

The assumtion is made that some of the representations can be used to describe almost all the features that are used to describe one individual cluster. To find these representations a principle component analysis is applied on the clusters of X and Y.

Principal component analysis (PCA) is used to find the most uncorrelated dimensions to the dataset while still preserving the most variance possible. The basis vectors that form the new basis of the dataset are callend principle components. It is often used as an exploratory tool to visualize and make large datasets with many dimensions more interpretable.

The PCA is applied on the cluster of z_A to find the components PC_A that can explain the set of representations with high variance. The components of PC_A span the PCA space of z_A. The same is done for the set z_B to yield the same number of principle components PC_B. The principle components represent the X-ray images in feature space that share the most features with other samples of their individual views and can cover the entire space of features through linear combinations.

During inference, the latent representation $p_A(z|x_{in})$ of an unknown test image xin is generated first. The representation is then projected into the PCA space of z_A and a weight matrix of the principle components is calculated such that $PCA(p_A(z|x_{in})) = W_A \cdot PC_A$.

It is further assumed, that corresponding representations carry equal weight when describing their clusters. Therefore, the same weighting matrix W_A is used to linearly combine the principle components and calculate $PCA(p_{pred}(z|y_{in}))$, such that $PCA(p_{pred}(z|y_{in})) = W_A \cdot PC_B$.

Transforming $PCA(p_{pred}(z|y_{in}))$ back into latent space, $p_{pred}(z|y_{in})$ as the input can now be used for the $Decoder_B$, completely forgoing the encoding step.

The same steps need to be repeated for all inputs of $Decoder_B$, which also includes the skip connections. As mentioned in Section 3.1.5, here the latent space regularization for the skip connections are included to satisfy the conditions for the linear combination. The generator with shared weights and multivariate gaussian skip connections can be seen in Figure 3.5.

There are a couple weaknesses with this method. First, even if the representations for X and Y occupy the same space with weight shared encoders, it does not necessary guarantee that the first k principal components of both clusters characterize the clusters in the same way. Secondly, the principal componentes can not necessarily be weighted the same way because the variance distribution might be completely different, which calls into question whether the weight matrix W_A of PC_A can be applied to PC_B. It is also uncertain if the relation between the cluster of z_A and $p(z|x_{in})$ can be applied to the second cluster of z_B and $p_{pred}(z|y_{in})$ at all.

3.2.3 PCA Loss

To counter these weaknesses, a new loss function is envisioned. It needs to be asserted that the predicion of $p_{pred}(z|y_{in})$ is the same as $p_B(z|y_{in})$ and thus the model has to be trained to learn this relationship. During the training step, the

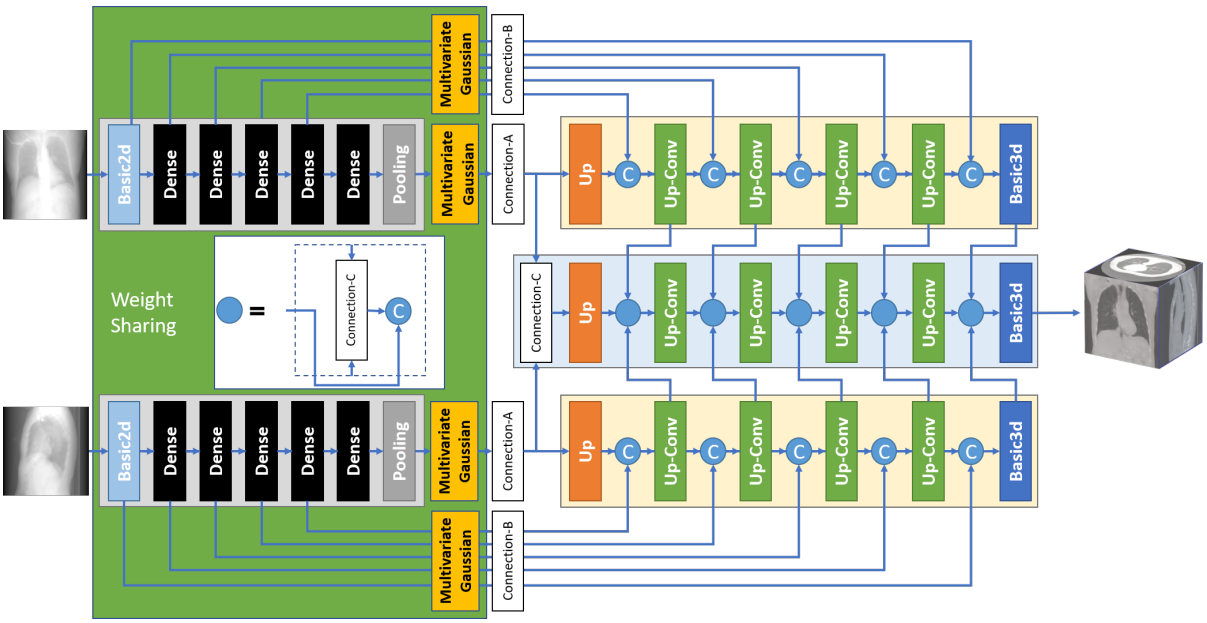


Figure 3.5: Modified generator design with multivariate Gaussian latent spaces and weight sharing.

latent space representations of m training image sets for the current encoder weights are calculated to populate the latent space such that

$$z_A = \{p_A(z|x_{train,0}), p_A(z|x_{train,1}), \ldots, p_A(z|x_{train,m})\}.$$

Ideally m is equal to the total number of training data n. For a random input image set x_{in} and y_{in}, taken from the training set, the same procedure as described in the section above is then initiated to make a prediction $p_{pred}(z|y_{in})$ from x_{in}. The actual encoded representation $p_B(z|y_{in})$ is calculated by passing y_{in} through $Encoder_B$. An L1 distance loss is then measured for $p_B(z|y_{in})$ and $p_{pred}(z|y_{in})$. This is done for all encoded latent spaces including the skip connections. In a broad sense, the network is trained to accomodate our method of prediction and encourage the encoders to produce feature spaces in which the prediction works.

Unfortunately, an implementation like this also brings new shortcomings. A complicated new loss function is certain to impact stability in a negative way, as do the latent space regularizations for the skip connections. Furthermore, m passes through $Encoder_A$ and m predictions are required during one training step. Therefore the choice of m will have a significant impact on training speed. m needs to be big enough to span a meaningful PCA space, but not too big as to make training impossibly slow. However, m can also not be smaller than the number of components k that are used for the PCA or else the space cannot be accurately explained.

Nonetheless, the new PCA loss function as shown in Equation 3.10 would enable the prediction of all latent representations of y_{in} during inference.

$$\mathcal{L}_{PCA} = \mathbb{E}(|p_{pred}(z|y_{in}) - p_B(z|y_{in})|_1) \tag{3.10}$$

3.2.4 Asymmetrical Training

The prediction for the inputs of the bridge connection Connection-B presents a huge challenge. Using the PCA prediction as a loss function means that during every training step, latent space representations have to be predicted for the entire training set, six times. Additionally the KL-Loss is used to regulate each latent space. This introduces a lot of possibilities for instability and increases the chance of the network not converging.

Identifying the problems for single image inference caused by the skip connections, it is proposed that the network is modified towards an asymmetrical design.

A simple way of solving the problem may come with dropping the skip connections for the second encoder-decoder and going back to variational autoencoders without regularized skip-connections. The benefits are two-fold. Firstly, the need to regularize the latent spaces of the skip connections is eliminated, which simplifies and stabilizes the network. Secondly, it also results in a less complicated inference step that now only involves the prediction of the bottle neck latent representation of the second encoder-decoder. The asymmetrical network is shown in Figure 3.6. An obvious drawback is the sacrifice of low level information from the second image that helps the reconstruction in the decoder.

Carrying over the idea of a latent space representation loss introduced in the section above, a distance based loss between the sampled latent space representations of the input images $p(z|x)$ and $p(z|y)$ is added. This is based on the notion that the loss of information during inference without a second input needs to be compensated for. Instead the second input image can be used to influence and guide the feature space construction by $Encoder_A$ during training. This can be interpreted as somewhat of transfer learning, except both networks learn at the same time. The feature latent space produced by the first encoder will be influenced by the features produced by the second encoder. After training is finished the hope is that the encoder can produce all the features from both images by only seeing the first input image. And from this feature space, the decoders can still work seperately and reconstruct 3D volumes from different views.

Table 3.1: Higher level building blocks for generator architecture are combined from Level 1 Blocks for simplicity. Note, for the Up-Conv block the Basic3d sub-block can be repeated N times, but the overall scaling factor should remain 2. The result of the Basic3d sub-blocks is used as an output for the fusion network before the Up sub-block is called.

High Level Blocks	Components
Dense	Down DenseNetBlock Compress
Up-Conv	Basic3d xN Up

Table 3.2: Low level building blocks are combinations of basic layers. To keep it simple, $kernel = 3$, $stride = 1$ and $padding = 1$ are used for all convolutional layers. The scaling factor the resized convolution is 2.

Low Level Blocks	Components
Down	Instance Norm ReLU Conv2d
Compress	Instance Norm ReLU Conv2d
Basic2d	Conv2d Instance Norm ReLU
Basic3d	Conv3d Instance Norm ReLU
Up	Resized Conv3d Instance Norm ReLU

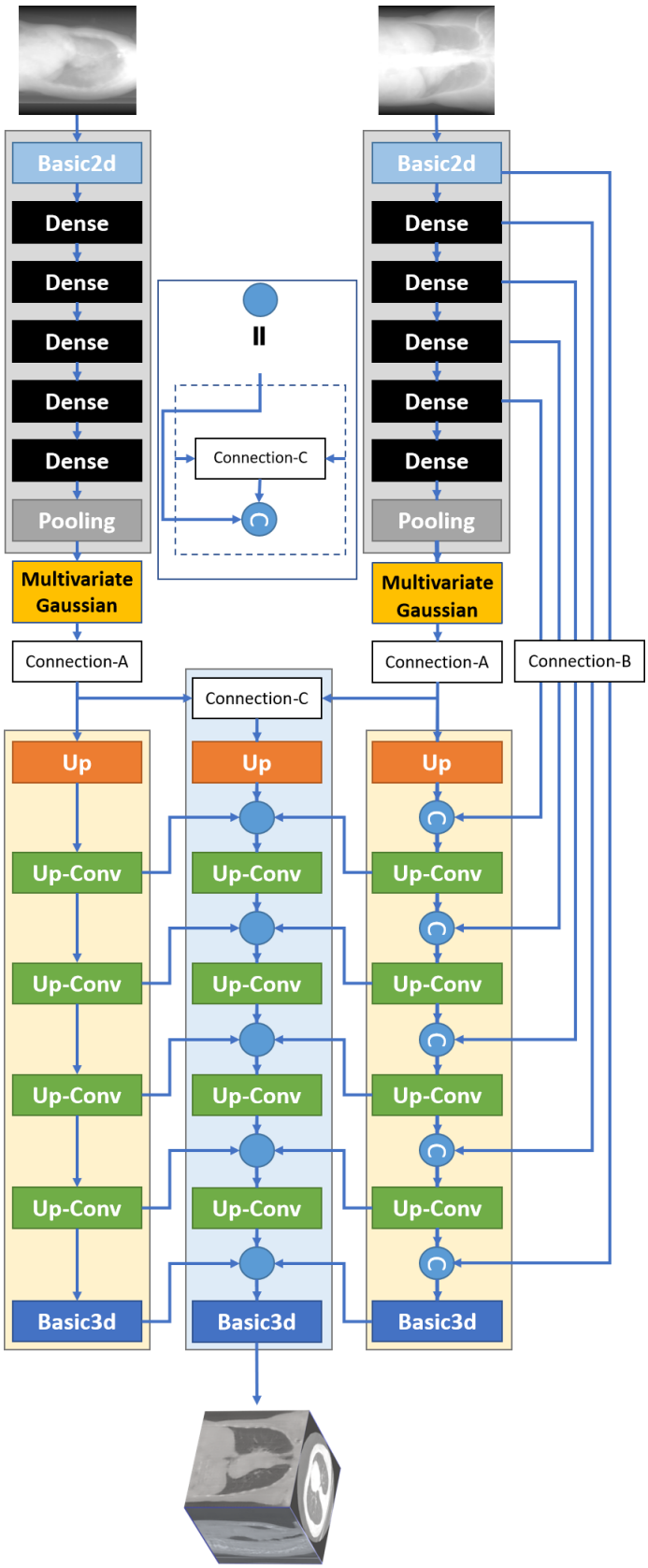


Figure 3.6: Asymmetrical generator architecture.

4 Dataset

4.1 LIDC

The Lung Image Database Consortium image collection (LIDC-IDRI) is a dataset provided by The Cancer Imaging Archive for public use [16]. It was created by a collaboration of seven academic centers and eight medical imaging companies with the goal development, training and evaluation of computer-assisted diagnostic methods for lung cancer detection and diagnosis.

The dataset contains 1080 cases of diagnostic and lung cancer screening thoracic computed tomography scans with manually marked up annotations, each. For our use case, only the thoracic computed tomography scans are relevant.

Additionally, X-ray images are required to accompany the CT scans for training. As to our knowledge, there is no dataset publicly available that contains thoracic X-Rays from different views that are perpendicular to each other, they were created artificially through the method of digitally reconstructed radiographs from the computed tomography scans of the LIDC dataset. Per scan, three projections in total were made, one for each view (axial, sagittal and coronal).

4.2 Digitally Reconstructed Radiographs (DRR)

A digitally reconstructed radiograph is an X-ray image that is reconstructed digitally from a computed tomography scan. A ray-tracing algorithm used for the reconstruction enables an approximation of the image that would result from photoelectric or Compton interactions [5].

The algorithm is divided into two parts. First the Hounsfield values of the CT screen must be converted to voxel wise linear attenuation:

$$\mu_x = [\frac{HU_x}{1000} + 1] * \mu_{H2O} \tag{4.1}$$

with HU_x as the voxel intensity at voxel x in Hounsfield units and μ_{H2O} as the linear attenuation coefficient of water.

Secondly, the resulting intensity of a single ray passing through the volume can be simulated as

$$I = I_0 \sum_{i=0}^{n} e^{-\mu_i d_i} \tag{4.2}$$

with I_0 as the initial intensity before entering the volume, n being the number of voxels the ray passes through, μ_i as the attenuation of the i th voxel and d_i as the individual depth of the ith voxel.

The reconstructed X-ray images were reviewed by an experienced radiologist and have been found to be accurate for all intents and purposes.

4.3 Pre-processing

The dataset is pre-processed before being fed into the network.

The heterogeneous imaging protocols of the CT scans lead to differences in value ranges and resolutions for different scans. The number of slices captured also vary from scan to scan. To make the dataset more homogenous, all CT scans are re-sampled into a cubic form of 512 x 512 x 512 pixels before the DRR projections are applied. Afterwards, all images are resized to 128 x 128 and 128 x 128 x 128, respectively.

All images are then normalized to a range of 0 to 1, following standard image pre-processing procedure.

The network thus receives a tuple containing one CT and the corresponding X-ray images from the coronal and sagittal view from the dataset.

5 Experiments and Results

In the following chapter, training details and metrics are explained and the results from the experiments are shown.

5.1 Training Details

The generator and discriminator are trained alternatively following the stardard process for training GANs, with the difference that the generator network is updated five times more often than the discriminator. For the PatchGAN discriminator, a patch size of 70x70 was chosen, as it was demonstrated to work well in [11,38]. Both networks are trained with the Adam optimizer. The learning rate is set to $1e-4$ and the momentum parameters are set to $\beta_1 = 0.5$ and $\beta_2 = 0.999$. After training 50 epochs, a linear learning rate decay policy is employed to decrease the learning rate to zero. The model is trained for 100 epochs. Concerning the weighting parameters for the loss functions, the adversarial loss is important for generating realistic details, but the correct structural reconstruction should be prioritized. Therefore, for the experiments the weighting parameters are set as $\lambda_1 = 0.1$ and $\lambda_2 = \lambda_3 = 10$.

An annealing scheme for both the Kullback-Leibler loss and the projection loss is also applied. The KL-loss starts out at 0.001 and is increased linearly over 50 epochs until it reaches 0.1. The two projection losses are used in combination as described in Equation 5.1. The value of α increases linearly from 0 to 1 over 20 epochs, starting from epoch 20.

The network was trained using one Nvidia Titan X graphics card. Due to GPU memory constraints, the batch size is limited to 1 for all experiments. Furthermore, instance normalization is used for all normalization layers.

5.2 Metrics

The quality of the reconstructed images are evaluated with peak signal-to-noise ratio (PSNR) and structured similarity (SSIM) index.

PSNR is a global image quality index and is a popular metric for measuring digital signals [59]. SSIM measures the similarity of two images by incorporating brightness, contrast and local structures. Therefore SSIM is a closer approximation of the human vision system [54].

5.3 Experiments

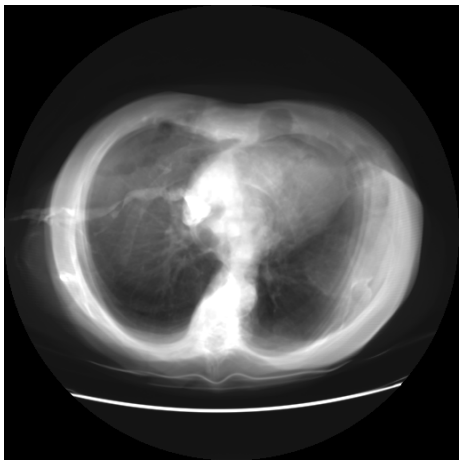
Based on the basic network architecture, I first set out to modify the network and improve the image and reconstruction quality. All evaluations are conducted on images that were predicted by the trained network, which is weight frozen and set to evaluation mode before prediction. For all predictions that do not include single image inference, the images are predicted by feeding biplanar X-ray images to the network, same as during training. For visualization purposes the same test image was used for all predictions.

5.3.1 DRR

Generating realistic X-rays is very important for the training of the network and forms the basis of our experiments. Without realistic X-ray images available as input images, it can be very difficult to show that the network is indeed effective in reconstructing CT images. Figure 5.1 shows the generated X-ray projections from our ground truth CT data in original resolution.

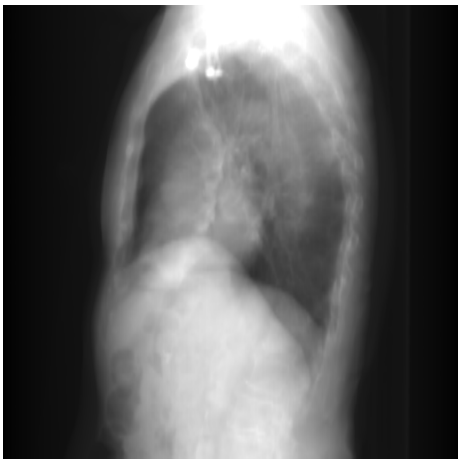
5.3.2 Preliminary Experiments

In this subsection, the results for some preliminary experiments are shown, which establish the importance of some particular components of the network. Omitting the fusion network potentially simplifies the network and enables faster training. Therefore the network is tested without the fusion network and having the average calculated between the results from both decoders and using a Basic3d block as the final layer. Figure 5.2(a) shows the reconstructions without using a fusion network. The difference in quality in comparison to the results from the vanilla network indeed justify the need for a fusion network that brings the results from both decoders together in different stages. Motivated by [15], where the encoder-decoder network did not include skip connections, Connection-B skip connections were dropped from the vanilla network to test if they provided a positive impact on the training result. Figure 5.2(b) shows the reconstructed images from the vanilla network without skip connections. In earlier stages this setup managed to produce images with some details, including bone structure and lung details. But the overall results were decidedly poorer than the results from the vanilla network as it converged on a very blurry representation. Finally, a prediction with the same input image for both encoder-decoder pairs was carried out Figure 5.2(c) shows the result with the coronal X-ray as both inputs and (d) shows the result with the axial view as both inputs. This shows that both branches are important in the reconstruction of the CT image. It can be concluded that the fusion network is essential for the networks performance. On the other hand it may be possible that the network without skip connections is able to reconstruct detailed images, although of worse quality. This is important for the asymmetrical setup tested in later sections. It is also confirmed,



(a) Axial

(b) Sagittal



(c) Coronal

Figure 5.1: X-ray images reconstructed by DRR from three different views. Views (a) and (b) are used to train the network.

that both branches contribute to the reconstruction and both learn different features during training.

Table 5.1: Average PSNR and SSIM scores for basic model with deconvolutional operations.

	PSNR	**SSIM**
Standard Deconvolution	20.592	0.713
Resized Convolution	20.981	0.723

5.3.3 Resized Convolution

The basic network applied deconvolution layers for the decoders when reconstructing the volumes and the results show obvious checkerboard artifacts. These were espcecially noticable because the entire volume is reconstructed through deconvolution and with multiple layers staggered on top of each other the effect of checkerboard artifacts is amplified. A comparison of the results is shown in Figure 5.3. The sagittal view (Figure 5.3(b)(left)) clearly shows the checkerboard artifacts in the area of the spine resulting from deconvolution. The right image shows the results of the basic network with resized convolution layers. The images show that it is able to siginificantly reduce checkerboard artifacts during reconstruction. A slight increase in the metric scores could also be observed as seen in Table 5.1. All following experiments use resized convolutions instead of standard deconvolutional operations.

5.3.4 Projection Loss

Using DRR as projection loss should help the network achieve even better structural consistency for the reconstructions.

Although the information content of $\mathcal{L}_{pl,drr}$ is high, it might not be suited to initiate training with when the GAN is unstable to begin with. In comparison, $\mathcal{L}_{pl,sp}$ is simpler and could be more helpful in the early stages. However, I believe that $\mathcal{L}_{pl,drr}$ can be very helpful in later stages of the training and I therefore propose a combination of both projection losses in a transition scheme, gradually transitioning the network from $\mathcal{L}_{pl,sp}$ to $\mathcal{L}_{pl,drr}$ over several epochs. An additional hyperparameter weight l_4 is also introduced for $\mathcal{L}_{pl,drr}$:

$$\mathcal{L}_{pl,c} = l_3 \cdot (1 - \alpha) \cdot \mathcal{L}_{pl,sp} + l_4 \cdot \alpha \cdot \mathcal{L}_{pl,drr} \tag{5.1}$$

where α starts out from zero and is gradually incremented to 1 to avoid a hard transition between the two projection losses. As the numerical values of $\mathcal{L}_{pl,drr}$ are higher than $\mathcal{L}_{pl,sp}$, l_4 is chosen to be 0.5 to stabilize training.

In the following experiments the vanilla projection loss $\mathcal{L}_{pl,sp}$ is compared against the improved DRR projection loss $\mathcal{L}_{pl,drr}$ and also a combination of both $\mathcal{L}_{pl,c}$. In order for the experiments with DRR projection loss to train properly, the dataset had to be windowed uniformely beforhand, while the vanilla network was able to adjust to the widely different intensity distributions of the original dataset. The results are shown in Figure 5.4 and the corresponding metric scores in Table 5.2.

Table 5.2: PSNR and SSIM scores for basic model with different projection losses.

	PSNR	**SSIM**
Shadow Projection	20.592	0.723
DRR Projection	18.787	0.549
Projection Combination	18.800	0.552

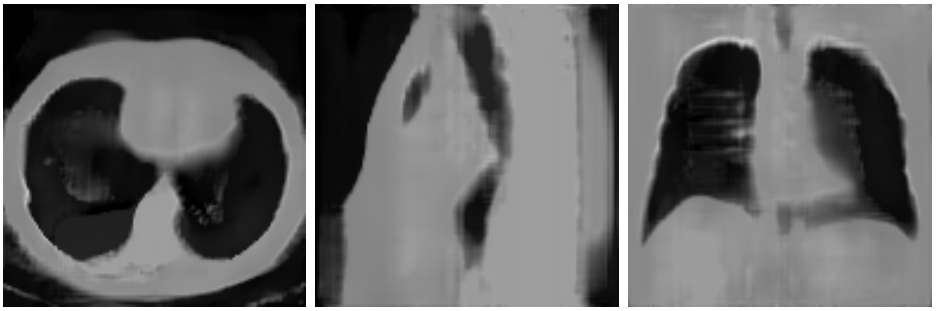
Table 5.3: Average PSNR and SSIM scores of the different VAE/GAN variations.

	PSNR	**SSIM**
VAE + AE	19.623	0.610
VAE + VAE	17.895	0.511
VAE + VAE Shared Encoder	17.680	0.496
VAE + VAE Full Weight Sharing	20.358	0.649
VAE-b + VAE-b	18.486	0.506

5.3.5 VAE/GAN

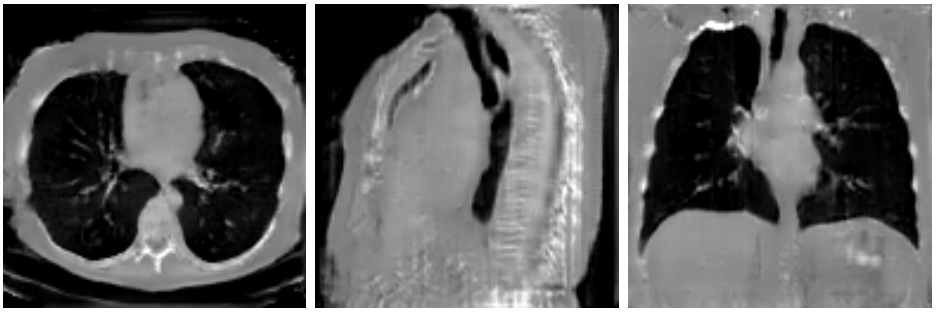
Introducing a disentangled latent space is expected to contribute greatly to the image quality of the output. In the following experiments, the encoder-decoder pairs, $A + B$ are tested with the following combinations: VAE + AE, VAE + VAE and VAE-b + VAE-b. AE is the autoencoder from the vanilla version, VAE is the autoencoder including a multivariate Gaussian module at the bottleneck and VAE-b is an VAE with multivariate Gaussian modules implemented for all the skip connections also.

A comparison of the results of VAE + AE, VAE + VAE and VAE-b + VAE-b can be seen in Figure 5.6. Additionally, two weight sharing combinations for VAE + VAE were tested, namely sharing the encoder only and full weight sharing. The results are shown in Figure 5.7.

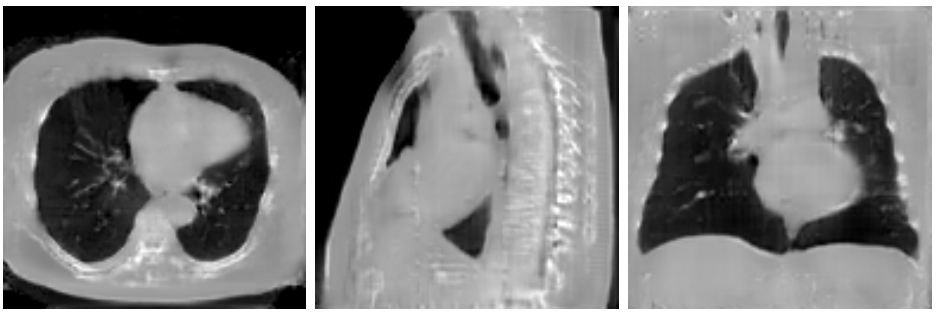


(a) Without Fusion

(b) Without Connection B

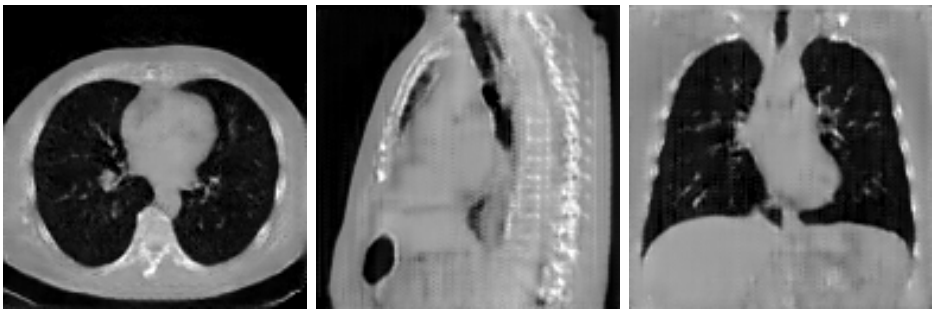


(c) Coronal X-ray as Prediction Input

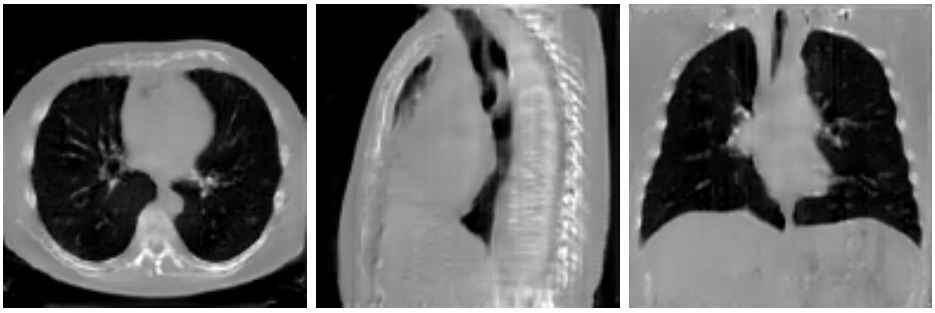


(d) Axial X-ray as Prediction Input

Figure 5.2: Reconstruction results (a) without using the fusion network, (b) without Connection B and (c) using the same image as inputs.

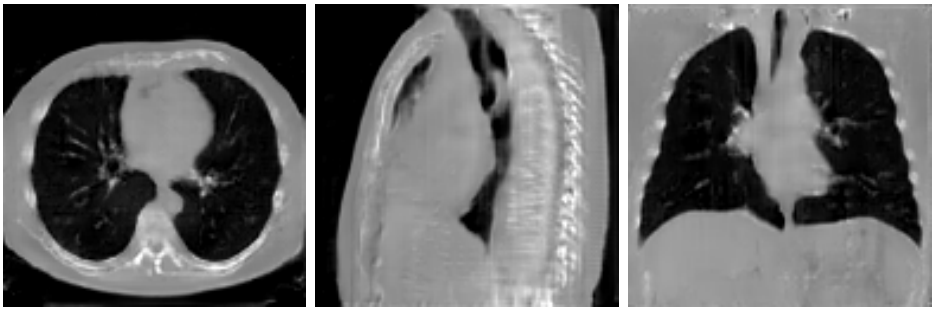


(a) Standard Deconvolution

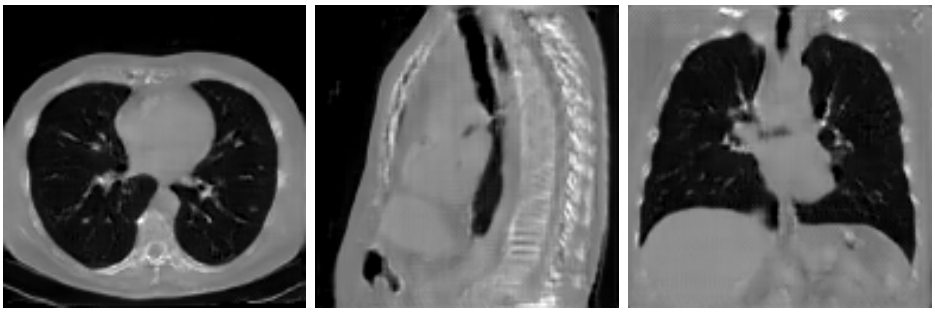


(b) Resized Convolution

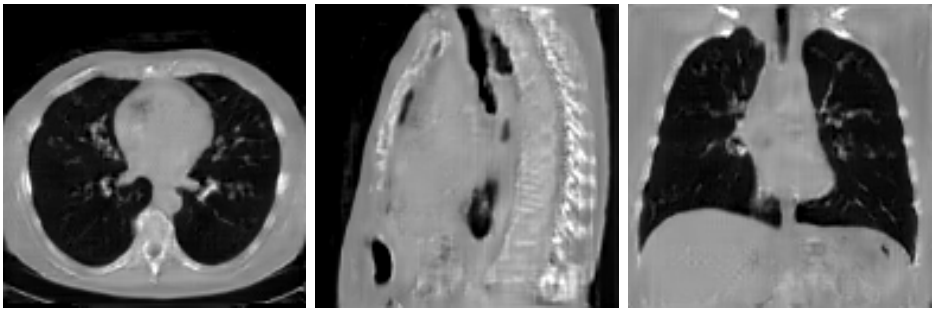
Figure 5.3: A comparison between the results from standard deconvolution (a) and resized convolution (b).



(a) Shadow Projection



(b) DRR Projection

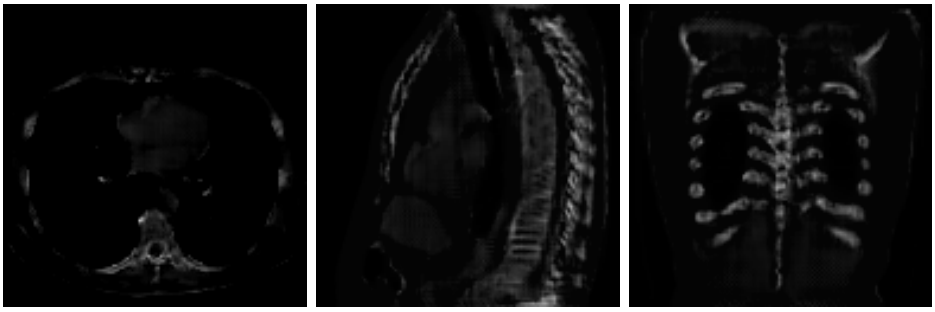


(c) Combination

(d) Ground Truth

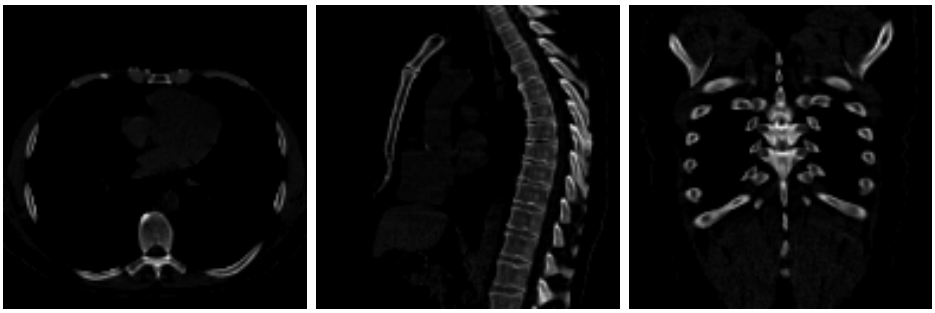
Figure 5.4: Images predicted from the vanilla model utilizing different projection losses. (a) shows the results with just shadow projection as projection loss, (b) shows the results using DRR projection and (c) shows the results of a combination of both. The ground truth image is shown in (c).

(a) Shadow Projection



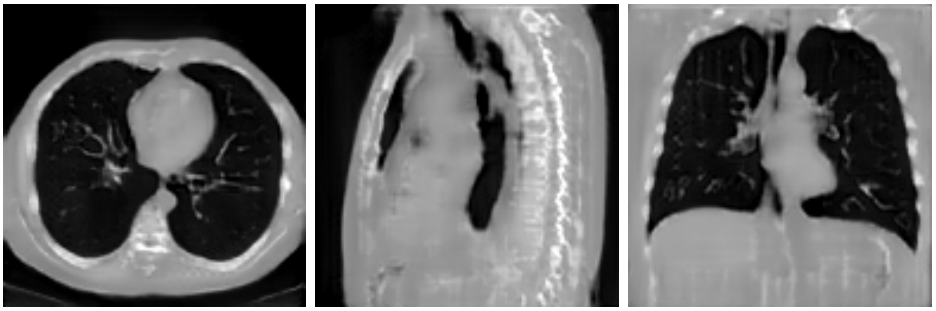
(b) DRR Projection

(c) Combination

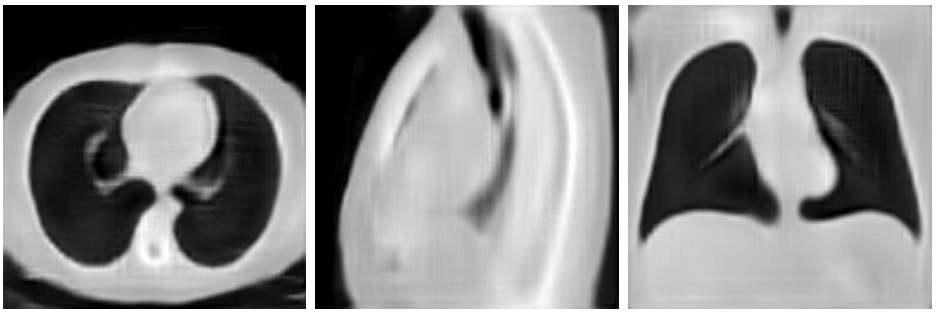


(d) Ground Truth

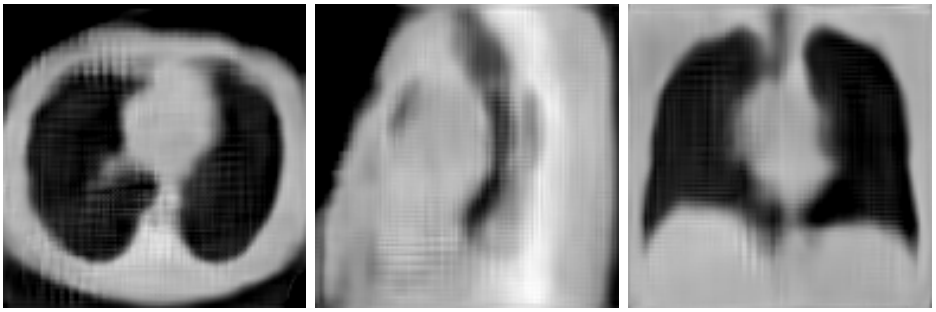
Figure 5.5: Images predicted by the vanilla model utilizing different projection losses and additional bone window applied. The windowing was applied after the images were predicted. (a) shows the results with just shadow projection as projection loss, (b) shows the results using DRR projection and (c) shows the results of a combination of both. The ground truth image is shown in (c).



(a) VAE + AE



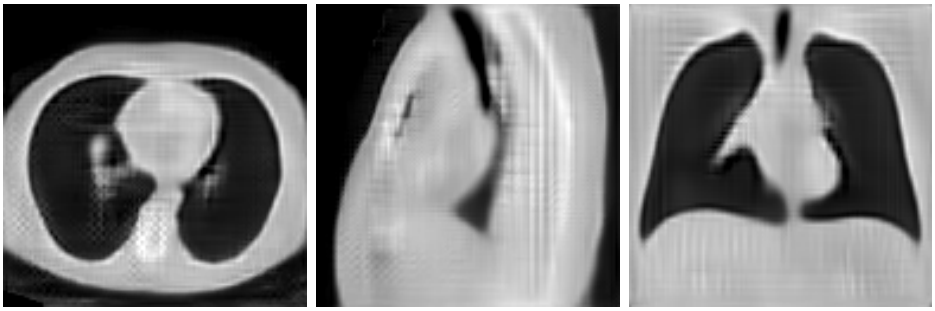
(b) VAE + VAE



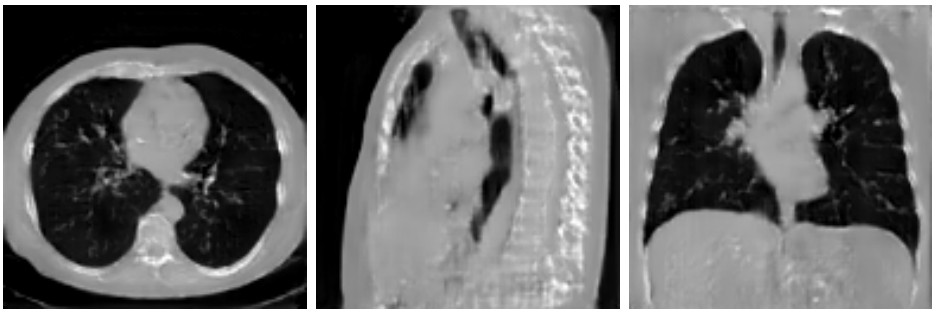
(c) VAE-b + VAE-b

(d) Ground Truth

Figure 5.6: The results from different VAE/GAN configurations. VAE + AE is shown in (a), VAE + VAE in (b), VAE-b + VAE-b in (c) and the ground truth is shown in (d).



(a) VAE + VAE Shared Encoder



(b) VAE + VAE Full Weight Sharing

(c) Ground Truth

Figure 5.7: Results from different weight sharing configurations of the VAE + VAE model. (a) shows the VAE + VAE network with shared encoders. (b) shows the VAE + VAE network with full weight sharing. And (c) shows the ground truht image.

5.3.5.1 Asymmetrical VAE/GAN

To make single image inference easier, the asymmetrical model drops the Connection B skip connections for the second encoder-decoder pair.

The hypothesis for the asymmetrical architecture is that during training the features learned by $Encoder_B$ will influence the features learned by $Encoder_A$. And since the goal of $\mathcal{L}_{z_space}$ is to pull the encoded latent representations together, $Encoder_A$ will hypothetically learn to find a feature representation of input x_in that can also represent input y_in. But in the same way the latent space representation encoded by $Encoder_A$ is pulled towards the latent space representation encoded by $Encoder_A$, it also happens the other way around. To prevent both encoders from learning a mixture of both representations, the gradient for $Encoder_B$ is detached just for the calculation of this loss function.

Regrettably, I was not able to test out this hypothesis in a successful experiment and the reason will be covered in the analysis chapter.

However, two models were trained on the asymmetrical model without the the latent space L1 loss. The first model had independent encoder-decoder pairs and the second one had encoders with shared weights. The single image inference was carried out with the PCA prediction method for both models.

5.3.6 Single Image Inference

This section shows the results of several approaches to achieve single image inference.

5.3.6.1 KNN prediction

KNN prediction can be utilized by all model implementations. The latent space representations are saved with the checkpoints so that they match the current model weights and can be loaded during prediction. Figure 5.9 shows the prediction results from a VAE + VAE model with full weight sharing in comparison with regular prediction and ground truth.

5.3.6.2 PCA prediction

The PCA prediction is performed after the training step is completed and requires the latent space representations of all training images at the time the training is finished. First the saved latent space representation of all training image pairs are loaded and a PCA is performed on the representations of the first view. The best k representations are used that can together explain the variance of the distribution to about 98.0%. Afterwards, the image to be tested is passed through $Encoder_A$ only to yield its representation in the latent space. This representation is transformed into PCA space using the previous top_k representations and a weight vector defines the transformation. A PCA is also conducted on the representations of the second view in latent space to generate the same number of principal components for that space. These components are multiplied by the weight vector produced in the previous step to create a pseudo representation of a pseudo second view input. This pseudo representation is inverse transformed back to the latent space and used as the input for $Decoder_B$. The same steps are repeated for the latent representations of the skip connections.

The PCA prediction method was first applied to the asymmetrical model and the asymmetrical model with weight shared encoder. The results of the predictions using this method are shown in Figure 5.10 and the average metrics are shown in Table 5.4.

The results of a PCA prediction method applied to the VAE-b + VAE-b network is schown in Figure 5.11.

Table 5.4: Average PSNR and SSIM scores of (a) KNN prediction, (b) PCA prediction on asymmetrical VAE/GAN, (c) PCA prediction on asymmetrical VAE/GAN with shared encoder and (d) PCA prediction on VAE-b + VAE-b model.

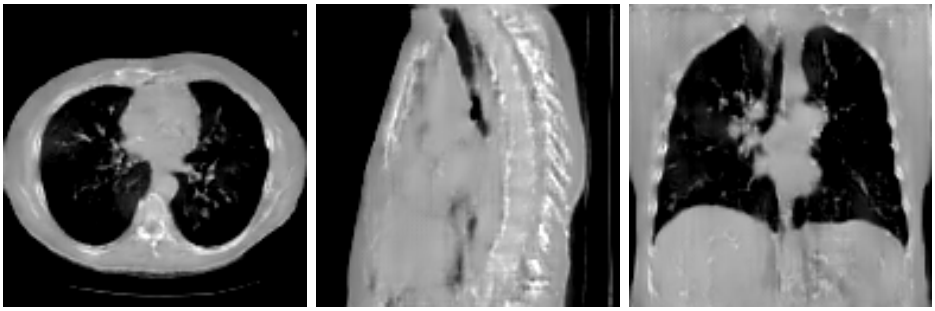
	PSNR	**SSIM**
KNN	16.687	0.543
PCA Asymmetrical	17.226	0.484
PCA Asymmetrical Shared Encoder	13.295	0.443
PCA VAE-b + VAE-b	16.000	0.443

(a) Asymmetrical

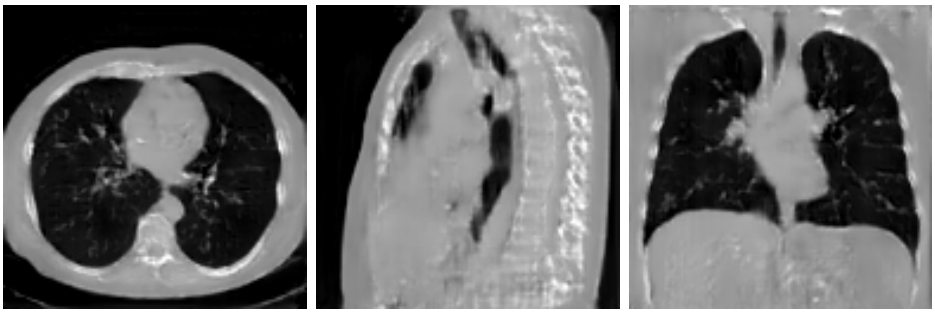
(b) Asymmetrical Shared Encoder

(c) Ground Truth

Figure 5.8: Results from different configurations of the asymmetrical model. (a) shows the asymmetrical model as described in Figure 3.6. (b) shows the asymmetrical model with encoder sharing.



(a) KNN



(b) Regular Prediction

(c) Ground Truth

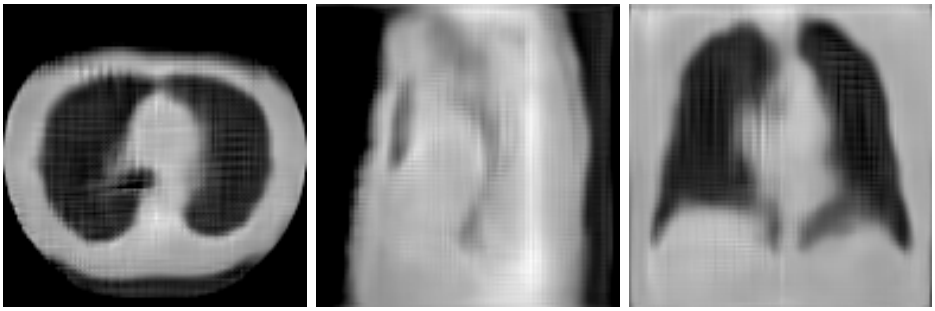
Figure 5.9: Results from single image inference with the KNN prediction method. (a) shows the KNN prediction result of the VAE + VAE model, (b) shows the result from a regular prediction and (c) shows the ground truth.

(a) PCA

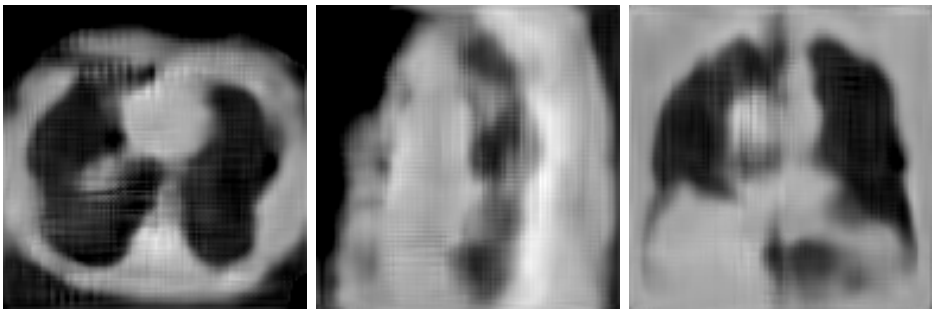
(b) Regular Prediction

(c) Ground Truth

Figure 5.10: Results from single image inference with the PCA prediction method and asymmetrical model. (a) shows the result from PCA prediction, (b) the result from regular prediction and (c) the ground truth.



(a) Successful



(b) Failed

(c) Regular Prediction

(d) Ground Truth

Figure 5.11: Results from Single Image Inference PCA prediction on asymmetrical model. (a) shows a more succesful prediction sample and (b) shows a failed sample.

6 Discussion and Analysis

In the following chapter, the results of the experiments are analyzed.

6.1 Checkerboard artifacts

The use of resized convolution successfully removed most of the checkerboard artifacts present in the image. As an unexpected side effect, resized convolutions also increased the speed of training.

6.2 Preliminary Experiments

It was expected that omitting the fusion network would result in poor quality images that were not able to produce many details, as seen in Figure 5.2(a). The results suggest, that the network only managed to learn the differences of air and tissue, but did not learn the finer differences in contrast between bone tissue and soft tissue. This further suggests, that the fusion network is essential in combining the features.

Dropping Connection B from the network achieved similarly blurry results, as shown in Figure 5.2(b). Although it should be noted, that in earlier epochs the network managed to produce more detailed reconstructions and converged on blurry results in the end. This suggests, that with additional effort it is possible for the network to be tuned to produce good results, which eliminates the need for skip connections altogether. On the other hand, the more detailed reconstructions were still inferior to results from the vanilla network in the same stages of training, so the end result might still be inferior.

The results for using the same X-ray image as input image for both encoder-decoder pairs are interesting. There are obvious errors in comparison with outputs of the normal training procedure, for example there are several small gaps and dents in the contours of the body and some of the inner structures are distorted. Since both 5.2(c) and (d) have similar issue with reconstruction but demonstrated in different sections of the image, it is safe to say that both encoder-decoder networks are equally important of the reconstruction and the joint training allows them to focus on different aspects of reconstruction from different images.

6.3 Projection Loss

In the following figure the effect of different projection losses are compared. First the original shadow projection loss, secondly the projection loss using DRR projections, and lastly the results using a combination of both in transition.

Under visual inspection, it is at first difficult to find differences that can be directly attributed to the different projection methods.

Focusing on soft tissue first in Figure 5.4, the axial view (left) is very similar for all results. 5.4(b) managed to reconstruct the bronchi, which are the fine white lines in the lungs, more accurately than 5.4(a) and 5.4(c), but all iterations reconstructed the shape of the heart slightly wrong. In the sagittal view (middle), more contrast can be observed for the soft tissue in 5.4(b). The shape of the air pockets (lungs and trachea) are also more similar to the ground truth. In 5.4(c) the contrast for the soft tissue is not as visible and there are slight errors in the positions of the air pockets. For 5.4(a), all soft tissue have the same intensity and can not be differentiated. In the coronal view (right) all implementations reconstruct the shape of the heart reasonably well. Concentrating on the bronchi again, the contrast for 5.4(b) and 5.4(c) are generally higher, although none of them manage to reconstruct them accurately in this view.

Even more insight can be gained by focusing on the bone structure only. Figure 5.5 shows the windowed results so that only the bones in the thorax are visible. In the axial view it can be observed that the shape of the rib cage is generally very similar to the ground truth for all reconstructions. Looking at the vertebra specifically, it can be seen that 5.5(a) shows cleaner reconstruction of the contours. From the sagittal view it can clearly be seen that the width and shape of the spine for 5.5(b) and 5.5(c) is much closer to the ground truth 5.5(d) than 5.5(a), where the base is too wide. The individual vertebras are also not as differentiated than in 5.5(b) and 5.5(c). From the coronal view it is also obvious that 5.5(b) and 5.5(c) are more similar to 5.5(d). The coronal view of 5.5(a) shows more ribs than visible in the ground truth image which indicates that the shape of the rib cage is very different. Structuraly 5.5(c) is most similar to the ground truth although the constrast in 5.5(b) within the same window is better.

Overall it can be concluded that the DRR projection is indeed very helpful in providing more structural information during the reconstruction process. It also shows, that metrics such as PSNR and SSIM are not suitable for reconstruction tasks like these. According to Table 5.2, using the shadow projection loss yields higher scores than just the DRR projection or a combination of both, even though in the detailed comparison above the results were generally not as accurate.

Interestingly, while the bone window was applied to all images, for the reconstructed samples there are still traces of soft tissues to be seen in almost all three views. This indicates that the network could not completely seperate the intensity values of soft tissue and bone tissue.

6.4 VAE/GAN

A major difficulty that was encountered when introducing a regularized latent space was the instability of a generative adversarial network. Small changes to the network often resulted in completely different outcomes with exploding gradients. This often requires extensive tuning of hyperparameters such as the learning rate, weighting parameters or optimizer step intervals, as well as finer parameters such as the number of epochs over which the weight of $\mathcal{L}_{kl}$ grows. Combined with the hardware limitations requiring a batch size of 1, the training period of a single run often exceeded 3 days to conclude.

Oftentimes, running the same configurations multiple times will also yield different results, even with maintaining constant seeding for initialization. This also testifies to the instability a GAN setup.

Figure 5.6(a) shows the results from the VAE + AE combination. The first thing to observe from the axial and sagittal view is that the shape of the body was not reconstructed accurately, which is an issue that perpetuates through out all the following experiments. In comparison to the ground truth image in 5.6(e), the slight dent on the top side of the axial view is not present in 5.6(a). The body is also wider in the sagittal view. The different body shape is probably due to the influence of other images in the training set. In comparison with the results from Figure 5.4(a) and (b), it can also be seen that the DRR projection loss did a better job in this regard and that the shadow projection loss did not enforce the shape properly. That being said, the shape of 5.6(a) is much closer to the ground truth than the shape of 5.4(a), especially noticable in the sagittal view. Even though the shape error still persists, the results from 5.6(a) in general seem to be an improvement to 5.4(a). Especially the reconstruction of the spinal area seems to be more succesfull as the structure and bend of the spine is more similar to the ground truth. In the sagittal view, the shapes of the 'air pockets', which are residuals of the lung and air way, also have clearer boundaries and contours. This suggests, that the variational autoencoder is more capable of seperating different mediums while reconstructing the image.

The results from both VAE + VAE, Figure 5.6(b), and the VAE + VAE with shared encoders, Figure 5.7(a) are poorer in comparison to VAE + AE and VAE + VAE with full weight sharing. The overall contours of the shapes and the inner structure is largely present and consistent. The models however could not pick up the finer details of the structure. In some parts of the images, especially present in (c), even checkerboard artifacts can be observed. In many ways the results for VAE + VAE and VAE + VAE with shared encoders were unsatisfying. The network design in itself does not, in my opinion, warrant such drastic differences in comparison with the other results that produced good and detailed images. Therefore the main suspicion lies with improperly set hyperparameters. The introduction of a Gaussian latent space distribution may have upset the weighting balance of the loss functions, which are also tied to the learning rate in general. Due to the instability of GANs, the original set of hyperparameters that worked on the previous network models

might becomes unsuitable when changing parts of the network. And given the long training times, it was difficult to conduct a comprehensive hyperparameter search. Given the right set of hyperparameters and loss balancing, I anticipate that the model can produce similar results as VAE + AE or VAE + VAE with full weight sharing. It was however not possible within the time frame of this work for this to be completed.

The results from VAE + VAE with full weight sharing (Figure 5.7(b)) were the most surprising. From a network design standpoint, it was expected that the implementation with encoders and decoders both sharing their weights would yield the worst results. A single encoder-decoder pair essentially had to learn how to extract the features and reconstruct the volume from two different projection images. The experiments, however, show that the reconstructed images from this model don't differ much from the previous results in 5.6(a) or 5.6(b). In some regions the reconstruction was even more succesful than the other results. Specifically looking at the sagittal view of 5.7(b), the spine vertebra are more individually defined and the width of the spine is also closer to the ground truth. The reconstruction errors present in 5.7(b) are also mostly present in the previous results, which makes it even more impressive.

This suggests, that a single encoder-decoder pair is deep enough to accomodate all the features from both views. This would also mean that a simpler model architecture could be used to achieve similar results.

6.4.1 Asymmetrical Model

The results from the asymmetrical model are shown in Figure 5.8. In comparison with previous results, the asymmetrical model was not able to achieve reconstructions of comparable quality. Although the shape of the body is correct in a broad sense, both reconstructions of (a) and (b) are very blurry with poorly defined structures and details. In the axial view of (a) it can be seen that the shape of the reconstruction has a slight gap in the upper left section. In the sagittal view of (a), the edges on the left side of the image are not clearly defined and transition between grey and black. Interestingly, both (a) and (b) also show checkerboard artifacts that should have been reduced by the use of resized convolution previously. The gaps in reconstruction suggest that the skip connections transferring low level structural information to the decoder had a bigger impact than previously expected. Although it is unclear why the implementation with shared encoders does not have the same issue. The general lack of detail suggests that the model is not able to converge and learn the details. This can have several explanations, but as mentioned in the earlier section, the main suspect is the imbalance of hyperparameters that cause the network to be unable to converge.

However, the results from (b) look promising because it does show an indication that the asymmetrical model is capable of learning the detailed features of the CT image.

6.5 Prediction Methods

This section will analyse the results of the different approaches taken for single image inference.

6.5.1 KNN Prediction

The KNN predictions were able to run on all network models, provided the latent space at the bottleneck of encoder and decoder was saved during the training run.

The results of such a prediction can be seen in Figure 5.9(a). Looking at the metrics in Table 5.4(a) first, it can be noticed that there is a noticable drop in both metrics. This drop is reflected in the irregular reconstruction by the network. Comparing the volume slices in (a) with the ground truth and the regular prediction, it is very noticable that there is a difference in the shapes of the reconstructed body. This is likely due to the fact, that the KNN prediction is looking for the training image that is most similar to the test image to use as the missing input. This implies that the comprehensiveness of the training dataset can be directly correlated with quality of results of the KNN prediction.

A training dataset that does not cover enough data variance cannot guarantee that a similar image to the test image is found. Since the second input image has equal influence on the reconstruction as the test image, it inevitably results in a skewed reconstruction prone to erros. This resonates with the results shown in Figure 5.2(c),(d).

Despite the structural dissimilarities, it is worth pointing out that the model is robust enough to generate an output that preserves the general features of a thorax CT despite having two input X-rays from different cases. This means that the model has learned where and how general attributes of the thorax are located and shaped. The spine for example is at the right place and also has the shape of a spine, but not exactly the same shape as the one shown in the ground truth. The same goes for other organs such as the lungs or heart.

Overall it can be said that the KNN prediction is a succesful method. Further improvements can be made by adjusting the distribution of the training dataset to better approach the variance of all real world cases.

6.5.2 PCA Prediction

The PCA prediction method was applied to the asymmetrical model and the VAE-b + VAE-b model.

For the asymmetrical model, this was only possible because the second encoder-decoder pair did not include any skip connections. Therefore only the latent space at the bottleneck of the second encoder-decoder needs to be predicted. This would not have been viable for the VAE + VAE setup since a prediction for every skip connection has to be made and the skip connections are not regularized to the same distribution between the first and second encoder-decoder pairs, which makes a

PCA prediction ineffective. The result of the PCA prediction with the asymmetrical model can be seen in Figure 5.11. From visual inspection, the results are surprisingly similar. In fact, calculating the average PSNR and average SSIM between the regular predictions and the PCA predictions yields a score of $PSNR_{avg} = 22.128$ and $SSIM_{avg} = 0.7005$, respectively. At first glance this is a very positive result because it means that the PCA prediction can generate very similar reconstructions as a regular prediction for most of the test dataset. Another possibility for the high scoring is that the modified second encoder-decoder did not actually contribute that much to the reconstruction. To verify this, the same model is used to reconstruct CTs with the same X-ray image for both inputs and the metrics are calculated against the original results from the PCA prediction. The results for the metrics are $PSNR_{avg} = 45.089$ and $SSIM_{avg} = 0.997$. The SSIM score indicates that there is almost not discernable difference between both images. As shown in Figure 5.2(c) and (d), a reconstruction using wrong input images results in an objectively worse reconstruction. Thus it can be argued that the scores are so high because the second encoder-decoder does not influence the prediction of the model. A likely explanation for this finding is that the second encoder-decoder actually did not learn any helpful features for the network and does not effectively contribute to the outcome of the reconstruction. The first encoder-decoder pair and the fusion network did most of the heavy lifting for the asymmetrical model. This new observation leads to several new options for the future. Firstly, it is now understood that this may be the reason for the blurry and not converging results. This presents a new starting point in improving the asymmetrical model. Secondly, it may indicate that the model structure is overall too complex and that a lot of the layers don't learn anything substantial at all. Simplifying the model could potentially reduce the memory footprint of the model and improve training speed. The results for the PCA prediction method using the VAE-b + VAE-b model was less succesful. As shown in Figure 5.11(a), it is possible for the inference to produce similar images to the results of a regular prediction. However, most of the results look more similar to 5.11(b). The are several possible explanations. Firstly, as mentioned before in Section 3.2, the encoded latent spaces for both encoders may not align well enough to permit the PCA prediction method. That is, the latent space dimensions sorted out by principle component analysis may represent different features for both latent spaces. Secondly, since the results from the regular prediction is also not satisfying, it is difficult to tell if the results are attributed to the model itself or the prediction method. Table 5.4, shows the metric scores for the examined PCA prediction methods.

6.6 Metrics

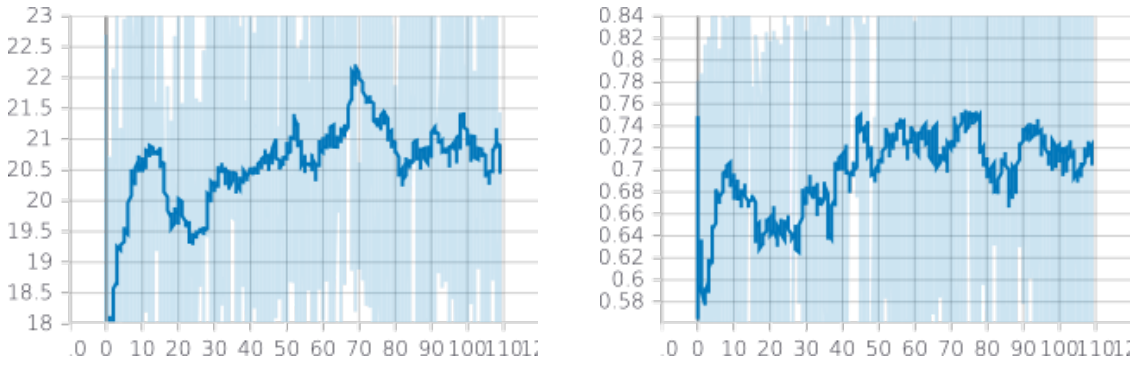
It was observed, that a high score in SSIM or PSNR did not necesssary equal a good reconstruction as opposed to judging the image by visual inspection.

To demonstrate this point further, Figure 6.1 shows the graphs of both PSNR

and SSIM during the training stage.

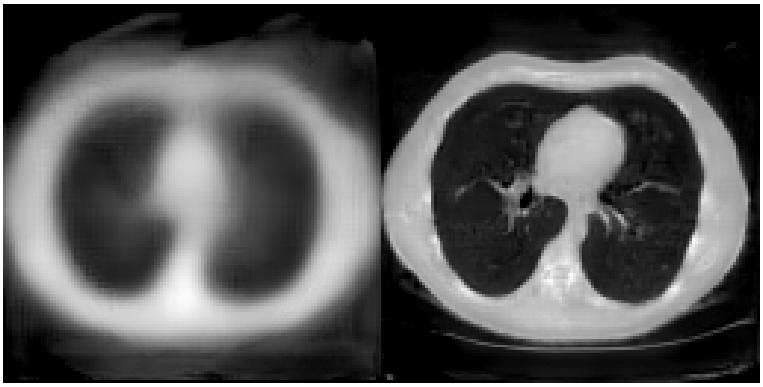
A model would often hit relatively high scores in the early stages where the network had just learned the general structure, but still produced very blurry images. In Figure 6.1(a) and (b), both curves have their early peaks around the tenth epoch. The corresponding output images to the epochs are shown in Figure 6.2. Once more details and higher contrasts are learned, it can be seen that the scores proceed to drop. This makes sense because the reconstructed details were often wrong in the beginning, which result in poorer PSNR or SSIM scores. Blurry images on the other hand score better because they are less wrong than images with sharp details that are in the wrong position. An example of low scores but detailed output can be seen in epoch 30. Near epoch 70, there is another peak in the PSNR graph which does not show up for SSIM. Although the corresponding output image (c) for that epoch does show an improvement in terms of sharpness, the image is visually not much different than the output image from epoch 100 (d), even though the score goes down again. At the end of training, at 100 epochs, both PSNR and SSIM scores settle approximately around the same values as the first peak. The output images however are visually much more detailed than in the beginning.

Another example for the disconnect between image quality and metrics can be shown in Figures 5.4 and 5.5, where an increase in correct reconstruction or sharper details also did not necessarily correlate with an increase of SSIM or PSNR scores. The results in 5.4(b) are visually and structurally closer to the ground truth but score worse on the metrics than 5.4(a).

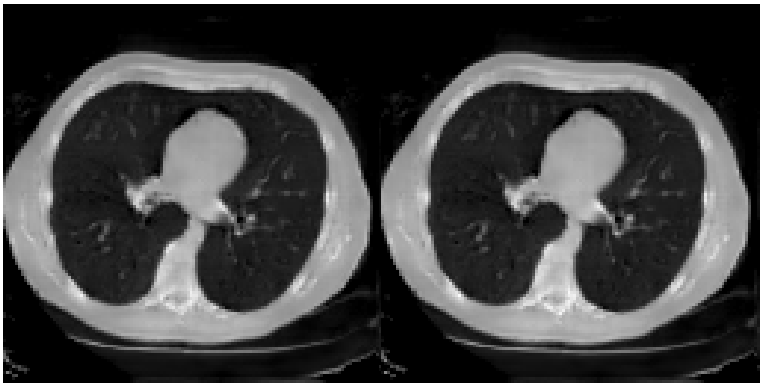


(a) PSNR **(b)** SSIM

Figure 6.1: The graph for the PSNR score during training is shown in (a) and the curve for SSIM is shown in (b). The y-axis represent the scores and the x-axis the epochs.



(a) Epoch 10 **(b)** Epoch 30



(c) Epoch 70 **(d)** Epoch 100

Figure 6.2: Image slices from different epochs of the training stage.

7 Conclusion and Future Work

It was shown that the synthesis of pseudo CT images with a high degree of detailed reconstruction is possible from a set of biplanar X-rays. Furthermore, this work shows that single image inference with comparable results is possible despite training the network with two input images.

Changes including resized convolutions and a more advanced projection loss were proven to increase the quality of the reconstruction. The VAE/GAN approach did offer some improvements towards the vanilla network regarding bone tissue details and contrasts. However, the current results also offer more room for improvement. Given the sensitive nature of GANs, further effort will be devoted to adjust and tune the network in future works to maximize the potential of VAE/GANs.

It was also shown that a single image inference was possible for a network that initially uses two input images during the training stage. Two methods were evaluated, the KNN prediction method and the PCA prediction method. The KNN prediction method was shown to recreate similar reconstructions regarding the representation of details and contrast. While there were differences in the shape of the reconstructed CT, the results suggest that improvements can be made by further tuning the dataset. As such the KNN prediction method is a straightforward and uncomplicated implementation and will be improved upon in future works. The PCA prediction method has the potential to be a more precise and robust prediction method than the KNN prediction method because it does not rely on the completeness of the training dataset in the same way the KNN prediction method does. The results of the experiments for the prediction are largely attributed to the trained model itself, which also affect the outcome of the prediction. Therefore, considering the results, it can be said that it is successful under the circumstances and that further evaluation is needed in the future.

In future works, different methods of achieving single image inference involving additional networks will be inspected and evaluated. A teacher student model with the pretrained model as teacher and a simpler VAE model with only one image input as student could condense the learned information and enabling the network to achieve the same results with only one image input. Another method that is possible would be to find a mapping function between the latent space representations with the use of another neural network.

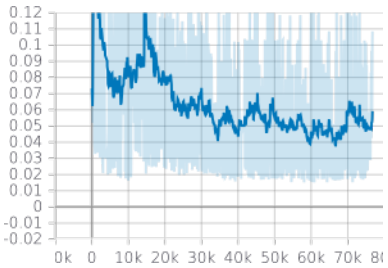
The visual quality of the reconstructed image will remain a focal point in future propects. One particular barrier to tackle is that of the resolution, which was limited by the capacity of the available hardware. Achieving high resolution may not be as straight forward as training with the images in original resolution (512x512x512) though. It was shown, that GANs have difficulties directly training with high reso-

lutions because higher resolutions make it easier to tell generated images and ground truth images apart. In other words, the job of the discriminator is made easier with the increase of resolution [60]. The solution may present itself in the progressive growing method proposed in [43]. Once the network is stable on lower resolutions, input resolutions are increased and layers added steadily until the desired resolution is achieved. Architecture wise, the design of the model is certainly expandable, however the limits imposed by the available hardware still remain.

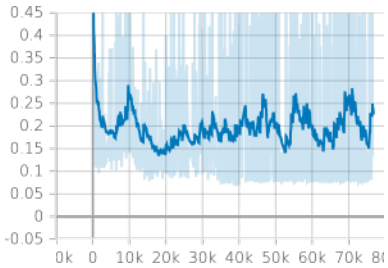
Finally, inspired by recent advances in [61, 62], future efforts will also be made towards the interpretability and disentanglement of the latent space. Ideally this will enable precise adjustments of independent features in the latent space and the structured synthesis of CT images.

A Appendices

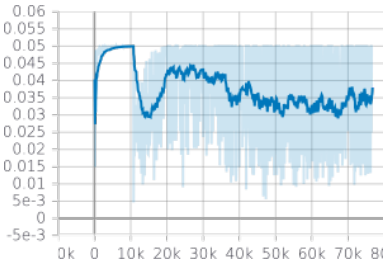
A.1 Loss Curves



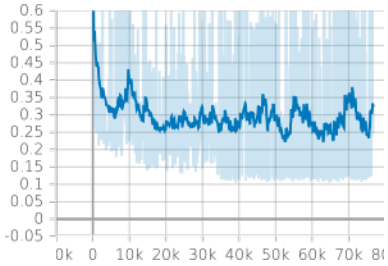
(a) Reconstruction Loss



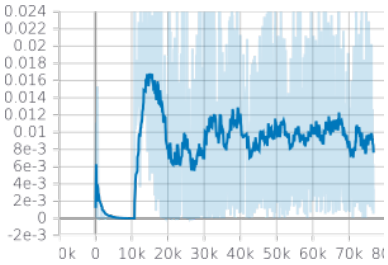
(b) Projection Loss



(c) Adversarial Generator Loss

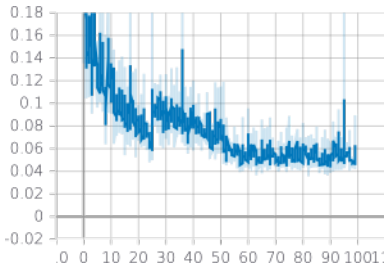


(d) Generator Loss

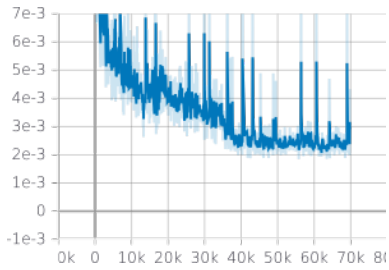


(e) Adversarial Discriminator Loss

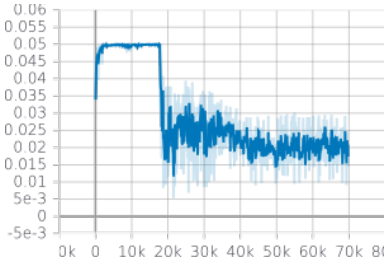
Figure A.1: The graphs show the loss curves for the vanilla network with resized convolution and shadow projections loss



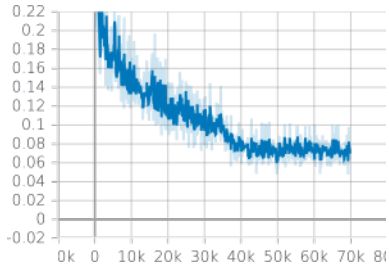
(a) Reconstruction Loss



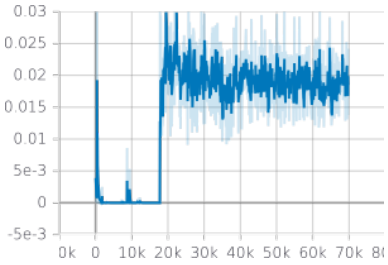
(b) Projection Loss



(c) Adversarial Generator Loss



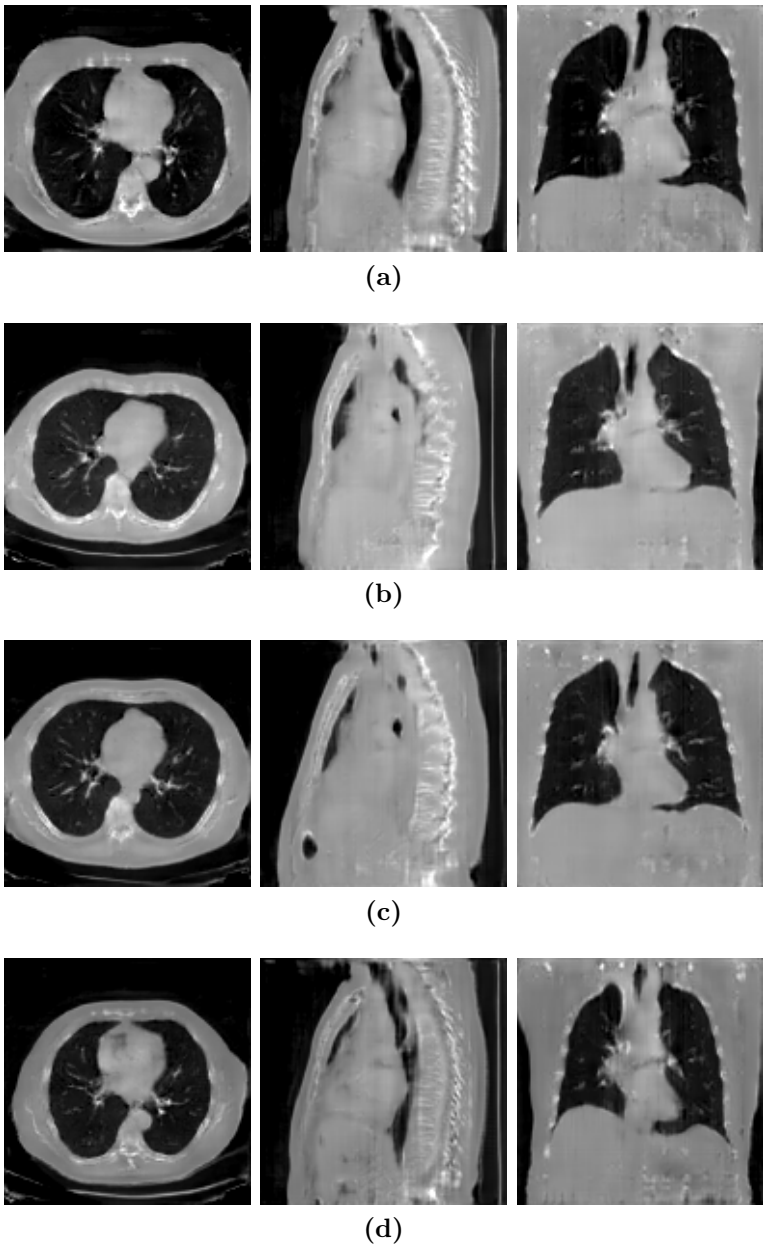
(d) Generator Loss



(e) Adversarial Discriminator Loss

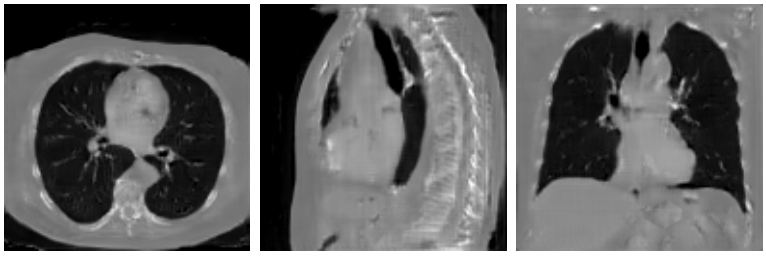
Figure A.2: The graphs show the loss curves for the vanilla network with DRR projections loss.

A.2 Additional Results

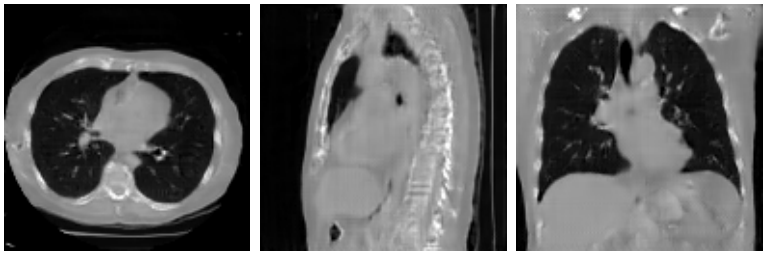


(a) **(b)** **(c)** **(d)**

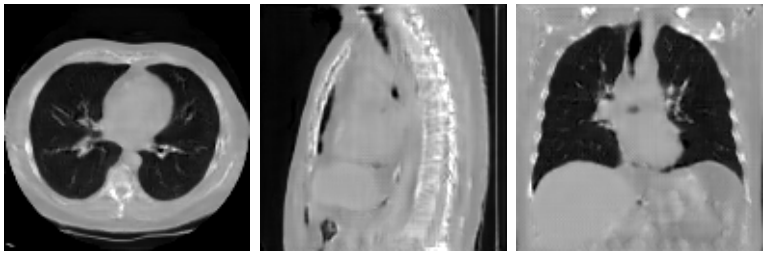
Figure A.3: Reconstruction samples from vanilla network with resized convolution.



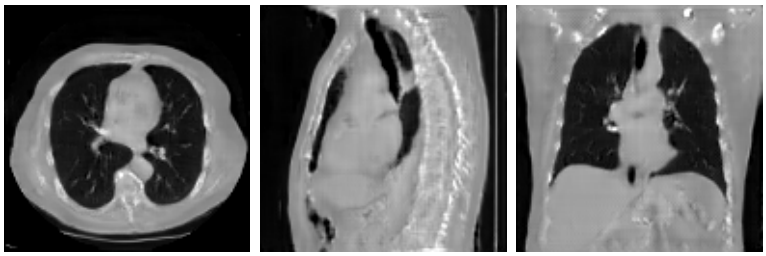
(a)



(b)

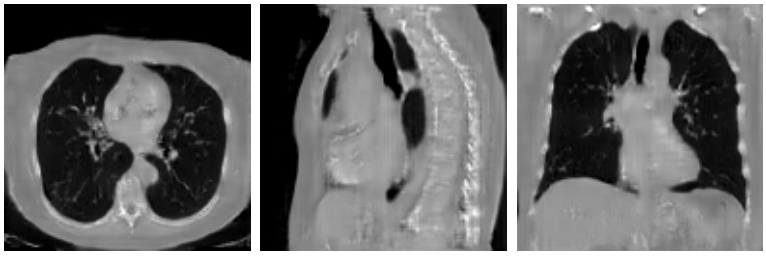


(c)

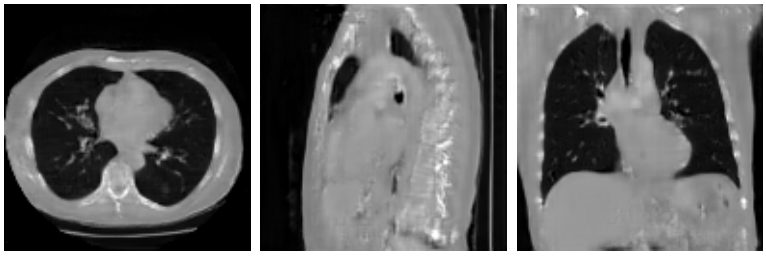


(d)

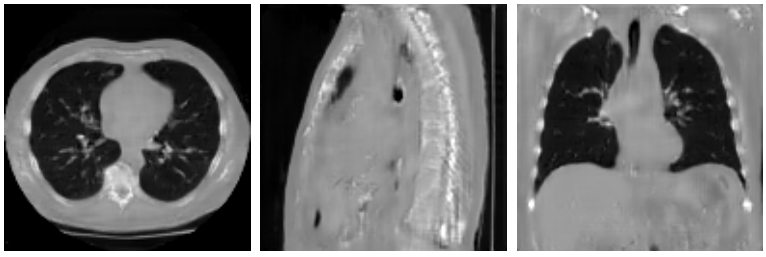
Figure A.4: Reconstruction samples from vanilla network with resized convolution and DRR projection loss.



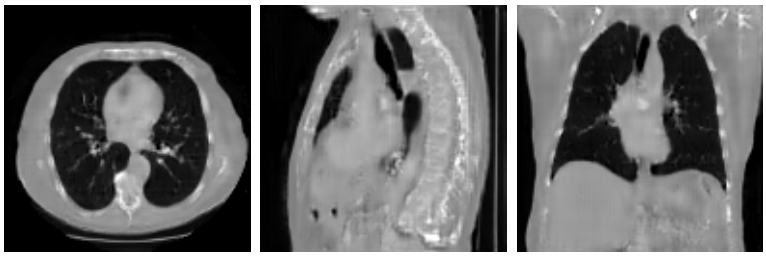
(a)



(b)

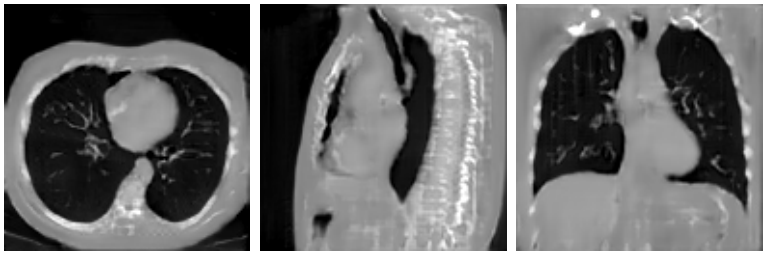


(c)

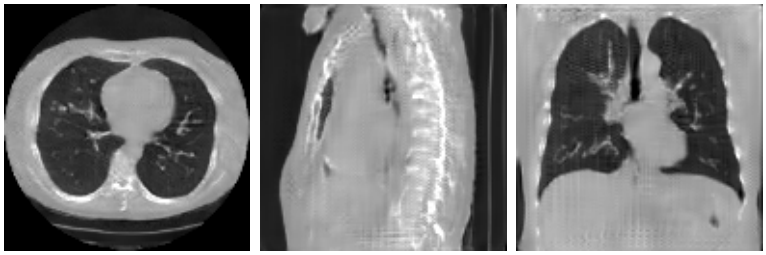


(d)

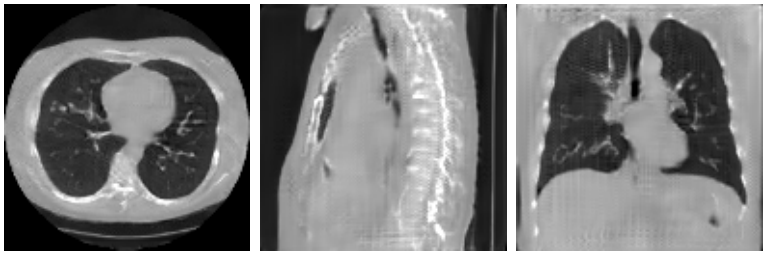
Figure A.5: Reconstruction samples from vanilla network with resized convolution and combined projection loss.



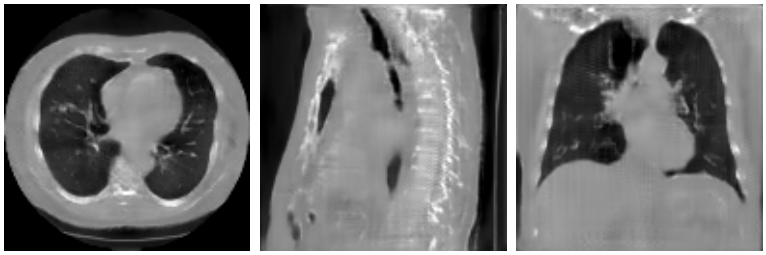
(a)



(b)

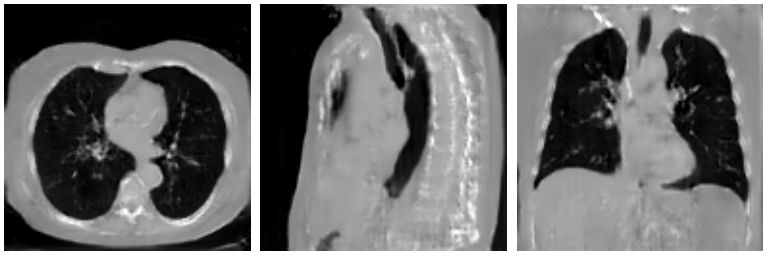


(c)

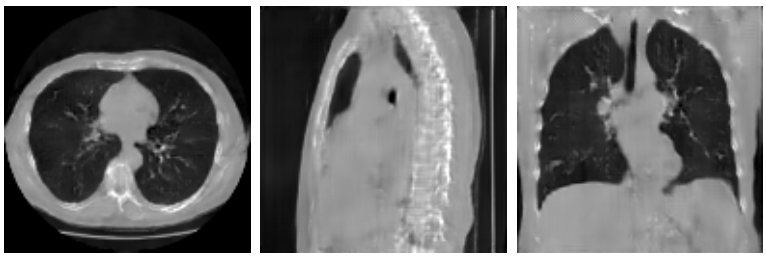


(d)

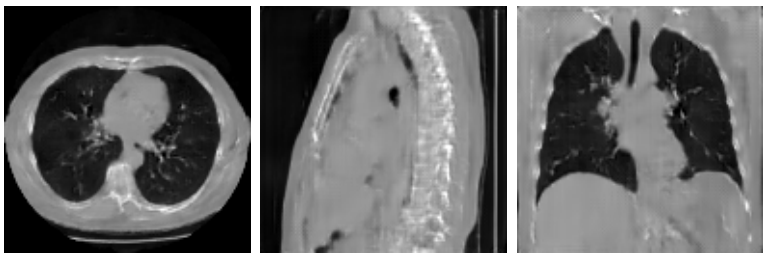
Figure A.6: Reconstruction samples from VAE + AE model.



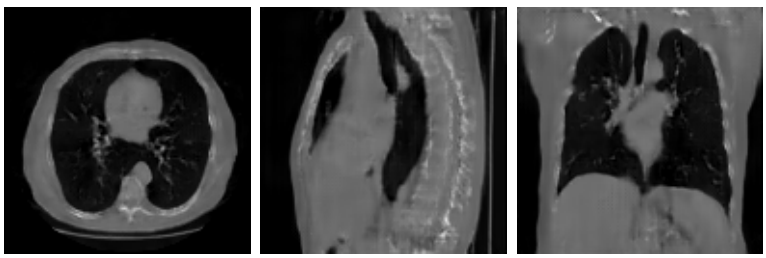
(a)



(b)



(c)



(d)

Figure A.7: Reconstruction samples from VAE + VAE model with full weight sharing.

Bibliography

[1] D. J. Brenner, C. D. Elliston, E. J. Hall, and W. E. Berdon, "Estimated risks of radiation-induced fatal cancer from pediatric ct," *American journal of roentgenology*, vol. 176, no. 2, pp. 289–296, 2001.

[2] M. S. Pearce, J. A. Salotti, M. P. Little, K. McHugh, C. Lee, K. P. Kim, N. L. Howe, C. M. Ronckers, P. Rajaraman, A. W. Craft *et al.*, "Radiation exposure from ct scans in childhood and subsequent risk of leukaemia and brain tumours: a retrospective cohort study," *The Lancet*, vol. 380, no. 9840, pp. 499–505, 2012.

[3] Y. Lei, J. Harms, T. Wang, Y. Liu, H.-K. Shu, A. B. Jani, W. J. Curran, H. Mao, T. Liu, and X. Yang, "Mri-only based synthetic ct generation using dense cycle consistent generative adversarial networks," *Medical physics*, vol. 46, no. 8, pp. 3565–3581, 2019.

[4] D. Nie, X. Cao, Y. Gao, L. Wang, and D. Shen, "Estimating ct image from mri data using 3d fully convolutional networks," in *Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 170–178.

[5] N. Milickovic, D. Baltas, S. Giannouli, M. Lahanas, and N. Zamboglou, "Ct imaging based digitally reconstructed radiographs and their application in brachytherapy," *Physics in Medicine & Biology*, vol. 45, no. 10, p. 2787, 2000.

[6] G. W. Sherouse, K. Novins, and E. L. Chaney, "Computation of digitally reconstructed radiographs for use in radiotherapy treatment design," *International Journal of Radiation Oncology* Biology* Physics*, vol. 18, no. 3, pp. 651–658, 1990.

[7] D. B. Russakoff, T. Rohlfing, K. Mori, D. Rueckert, A. Ho, J. R. Adler, and C. R. Maurer, "Fast generation of digitally reconstructed radiographs using attenuation fields with application to 2d-3d image registration," *IEEE transactions on medical imaging*, vol. 24, no. 11, pp. 1441–1454, 2005.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[9] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[10] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[11] X. Ying, H. Guo, K. Ma, J. Wu, Z. Weng, and Y. Zheng, "X2ct-gan: reconstructing ct from biplanar x-rays with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 10 619–10 628.

[12] J. Wu and M. R. Mahfouz, "Robust x-ray image segmentation by spectral clustering and active shape model," *Journal of Medical Imaging*, vol. 3, no. 3, p. 034005, 2016.

[13] Y. Chen, X. Ee, W. K. Leow, and T. S. Howe, "Automatic extraction of femur contours from hip x-ray images," in *International Workshop on Computer Vision for Biomedical Image Applications*. Springer, 2005, pp. 200–209.

[14] O. Dössler, *Bildgebende Verfahren in der Medizin*. Springer, 2000.

[15] L. Shen, W. Zhao, and L. Xing, "Patient-specific reconstruction of volumetric computed tomography images from a single projection view via deep learning," *Nature biomedical engineering*, vol. 3, no. 11, pp. 880–888, 2019.

[16] S. G. Armato III, G. McLennan, L. Bidaut, M. F. McNitt-Gray, C. R. Meyer, A. P. Reeves, B. Zhao, D. R. Aberle, C. I. Henschke, E. A. Hoffman *et al.*, "The lung image database consortium (lidc) and image database resource initiative (idri): a completed reference database of lung nodules on ct scans," *Medical physics*, vol. 38, no. 2, pp. 915–931, 2011.

[17] E. Kang, J. Min, and J. C. Ye, "A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction," *Medical physics*, vol. 44, no. 10, pp. e360–e375, 2017.

[18] T. Kubo, P.-J. P. Lin, W. Stiller, M. Takahashi, H.-U. Kauczor, Y. Ohno, and H. Hatabu, "Radiation dose reduction in chest ct: a review," *American journal of roentgenology*, vol. 190, no. 2, pp. 335–343, 2008.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[20] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[21] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[23] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[24] L. Li, L. Qin, Z. Xu, Y. Yin, X. Wang, B. Kong, J. Bai, Y. Lu, Z. Fang, Q. Song *et al.*, "Artificial intelligence distinguishes covid-19 from community acquired pneumonia on chest ct," *Radiology*, p. 200905, 2020.

[25] B. Alipanahi, A. Delong, M. T. Weirauch, and B. J. Frey, "Predicting the sequence specificities of dna-and rna-binding proteins by deep learning," *Nature biotechnology*, vol. 33, no. 8, pp. 831–838, 2015.

[26] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951. [Online]. Available: http://www.jstor.org/stable/1969529

[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[28] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.

[29] D. Nie, R. Trullo, J. Lian, C. Petitjean, S. Ruan, Q. Wang, and D. Shen, "Medical image synthesis with context-aware generative adversarial networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2017, pp. 417–425.

[30] Z. Zhang, L. Yang, and Y. Zheng, "Translating and segmenting multimodal medical volumes with cycle-and shape-consistency generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern Recognition*, 2018, pp. 9242–9251.

[31] J. M. Wolterink, A. M. Dinkla, M. H. Savenije, P. R. Seevinck, C. A. van den Berg, and I. Išgum, "Deep mr to ct synthesis using unpaired data," in *International workshop on simulation and synthesis in medical imaging*. Springer, 2017, pp. 14–23.

[32] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun, and G. Wang, "Low-dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss," *IEEE transactions on medical imaging*, vol. 37, no. 6, pp. 1348–1357, 2018.

[33] J. M. Wolterink, T. Leiner, M. A. Viergever, and I. Išgum, "Generative adversarial networks for noise reduction in low-dose ct," *IEEE transactions on medical imaging*, vol. 36, no. 12, pp. 2536–2545, 2017.

[34] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[35] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[37] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[38] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[39] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European conference on computer vision*. Springer, 2016, pp. 702–716.

[40] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. IEEE, 1999, pp. 1033–1038.

[41] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in neural information processing systems*, 2015, pp. 262–270.

[42] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2794–2802.

[43] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[44] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[45] D. Jimenez Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," 01 2014.

[46] D. Kingma, D. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," *Advances in Neural Information Processing Systems*, vol. 4, 06 2014.

[47] V. Prokhorov, E. Shareghi, Y. Li, M. T. Pilehvar, and N. Collier, "On the importance of the kullback-leibler divergence term in variational autoencoders for text generation," *arXiv preprint arXiv:1909.13668*, 2019.

[48] T. Spinner, J. Körner, J. Görtler, and O. Deussen, "Towards an interpretable latent space : an intuitive comparison of autoencoders with variational autoencoders," in *Proceedings of the Workshop on Visualization for AI Explainability 2018 (VISxAI)*, 2018. [Online]. Available: https://thilospinner.com/towards-an-interpretable-latent-space/

[49] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[50] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[51] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[52] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879

[53] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[54] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[55] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," *CoRR*, vol. abs/1604.04382, 2016. [Online]. Available: http://arxiv.org/abs/1604.04382

[56] L. Jiang, S. Shi, X. Qi, and J. Jia, "Gal: Geometric adversarial loss for single-view 3d-object reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 802–816.

[57] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[58] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen, "Image reconstruction by domain-transform manifold learning," *Nature*, vol. 555, no. 7697, pp. 487–492, 2018.

[59] I. Bankman, *Handbook of medical image processing and analysis*. Elsevier, 2008.

[60] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2642–2651.

[61] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[62] S. Pidhorskyi, D. A. Adjeroh, and G. Doretto, "Adversarial latent autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 104–14 113.