Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Trygve Scheline Urdahl

Generative Adversarial Networks in X-ray Computed Tomography

Master's thesis in Physics and Mathematics
Supervisor: Basab Chattopadhyay
July 2021

Trygve Scheline Urdahl

Generative Adversarial Networks in X-ray Computed Tomography

Master’s thesis in Physics and Mathematics
Supervisor: Basab Chattopadhyay
July 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Abstract

X-ray computed tomography (CT) allows for non-destructive imaging of internal structures of materials. The process of creating CT images involves recording x-ray projections of a sample, and computationally reconstructing the projections into a 3D image of the sample. There is an increasing need to extend the methodology to imaging dynamical processes and also limit radiation induced damage on the studied materials. This requires that the projections are obtained with very little capture time and/or the number of projections are reduced. Such data collection strategies will result in noisy and artifact prone reconstructed images. In this thesis we utilize *generative adversarial networks* (GANs), a form of machine learning model, to denoise subsampled and noisy CT images.

A GAN has been trained to map noisy CT images to high-quality CT images, effectively denoising them. The GAN improves the structural similarity index measure (SSIM) of the noisy reconstruction from 0.233 to 0.789, and the mean squared error from 704.4 to 210.8, when denoising an undersampled CT dataset containing 46 uniformly sampled projections from a high-quality dataset which contains 1500 projections. The GAN has been tested for a range of undersampling levels as well as modifying the loss function. A log-cosh term has been introduced to the loss function used to train the GAN, yielding an improvement in the achieved SSIM from 0.788 to 0.789 for the aforementioned undersampled reconstruction, without introducing any discernible drawbacks.

The GAN denoising has been compared to a *prior image constrained compressed sensing* (PICCS) reconstruction of a dynamic CT dataset. The GAN denoising achieves comparable image quality to the PICCS reconstruction, with some sample details not distinguishable in the PICCS reconstruction being captured by the GAN denoised reconstruction.

Interaxial banding artifacts are introduced when denoising 2D slices of a 3D sample along an axial plane. These artifacts are reduced by using a depth parameter when training the GAN, allowing the denoising to utilize 3D spatial information from adjacent slices.

Sammendrag

Computertomografi (CT) med røntgenstråler åpner opp for ikke-destruktiv bildetaking av interne strukturer i materialer. Prosessen for å ta CT-bilder består av å måle røntgenprojeksjoner av en prøve, for deretter å beregningsmessig rekonstruere en 3D-modell av prøven fra projeksjonene. Det å måle projeksjonene tar tid, og det brukes en strålingskilde. Det er et økende behov for å forbedre teknikker for å ta bilde av dynamiske prosesser, samt å redusere strålingsskade på de avbildede materialene. Dette krever at projeksjonene blir tatt fort, eller at antallet projeksjoner reduseres. En slik endring i bildetakingsmetoden fører til økt støy og artefakter i de rekonstruerte bildene. I denne avhandlingen bruker vi *generative adversarielle nettverk* (GAN), en maskinlæringsmodell, til å redusere støy i undersamplete og støyfylte CT bilder.

Et GAN har blitt trent for å transformere støyfylte CT-bilder til høykvalitets CT-bilder, som i praksis vil si at det fjerner støy. Støyreduksjonen med GAN gir en økning i et strukturelt likhetsmål (SSIM) fra 0.223 til 0.789, og reduserer den midlere kvadratiske feilen fra 704.4 til 210.8, for et projeksjonsundersamplet CT-datasett med 46 projeksjoner uniformt undersamplet fra et høykvalitets datasett med 1500 projeksjoner. GAN-metoden har blitt testet med en rekke ulike grader av projeksjonsundersampling, samt med endringer i tapsfunksjonen. Et log-cosh-ledd har blitt lagt til i tapsfunksjonen brukt til å trene GANet. Det ga en forbedring i SSIM fra 0.778 til 0.789 for det forannevnte datasettet uten å introdusere noen nevneverdige ulemper.

Støyreduksjonen med GAN har blitt sammenlignet med en *tidligere-bilde-begrenset komprimert sensing* (PICCS)-rekonstruering av et dynamisk-CT-datasett. GAN-støyreduksjonen oppnår lignende resultater som PICCS-rekonstruksjonen, og noen detaljer som ikke er observerbare i PICCS-rekonstruksjonen kan sees i GAN-støyreduksjonen.

Artefakter mellom planene oppstår når metoden støyreduserer 2D bilder av et 3D objekt langs et plan. Disse artefaktene kan reduseres ved å bruke en dybde-parameter under treningen av GANet som lar metoden bruke andre nærliggende bilder for å bruke 3D informasjon til støyreduseringen.

Preface

This master's thesis is written to fulfill the requirements of the Master of Science (M.Sc.) program *Applied Physics and Mathematics* at the Norwegian University of Science and Technology(NTNU). It concludes a five year study program with a specialization in engineering physics.

The work of this thesis was performed during the spring semester of 2021. I have had the pleasure of working with the X-ray group at the Department of Physics at NTNU and PoreLab throughout the semester.

I would like to thank my supervisor Basab Chattopadhyay for providing this exciting project, as well as guiding me through it. His help and willingness to let me form the project after my interests have made this a very enjoyable semester.

Additionally, a big thank you goes to Fredrik K. Mürer for his input and feedback throughout the semester, especially during the final stretch. The comments he has given me have certainly improved the quality of this thesis. To PhD candidate Kim Robert Bjørk Tekseth, I would like to express my gratitude for providing an exciting dataset and fruitful discussions of my results.

Finally, I have to thank my friends who have listened to me blabber on about this project throughout the entirety of the semester. I have enjoyed annoying you all!

Trygve Scheline Urdahl
Trondheim, July 2021

Contents

Figures

Tables

Acronyms

ANN	artificial neural network.
ART	algebraic reconstruction technique.
CNN	convolutional neural network.
CT	computed tomography.
FBP	filtered back projection.
FDK	Feldkamp-Davis-Kress.
GAN	generative adversarial network.
HQ	high-quality.
IHHQ	in-house high-quality.
IHLQ	in-house low-quality.
LQ	low-quality.
MAE	mean absolute error.
ML	machine learning.
MSE	mean squared error.
PICCS	prior image constrained compressed sensing.
PSNR	peak signal-to-noise ratio.
ROI	region of interest.
SGD	stochastic gradient descent.
SIRT	simultaneous iterative reconstructive technique.
SSIM	structural similarity index measure.

Chapter 1

Introduction

X-ray computed tomography (CT) allows for non-destructive imaging of internal structures of materials in many disciplines including material science, medical imaging, and geological studies. The process involves recording x-ray projection images of an entire sample as it is rotated about a common axis, and then these images are reconstructed computationally to provide a 3D image of the sample [1]. Such experiments and reconstruction can be a lengthy process depending on the sample complexity.

In recent years, the use of machine learning (ML) has increased drastically. With an ever-increasing amount of data available, ML opens up opportunities of getting more from this data than what used to be possible. The field of ML is sufficiently young that it is still rapidly expanding, and new techniques are discovered regularly [2]. Among these recent discoveries is the generative adversarial network (GAN), first introduced in 2014 [3]. This class of neural networks has been used to synthesize images from specific categories (e.g. "image of bird", or "image of sunflower") [4], perform photo-realistic image super-resolution [5], and much more. In recent years it has also been used to reconstruct and denoise CT images [6, 7].

1.1 Motivation

The process of collecting high-quality CT images takes a long time and may cause unnecessarily large radiation doses. Some objects may need to be imaged quickly (e.g. dynamic CT where moving objects are studied). Unfortunately, capturing images quickly and reducing the radiation dose also leads to increased noise in the final images. The possibility of reducing both the capture time and radiation dose for CT imaging while keeping a sufficiently good image quality is therefore a topic of interest.

Furthermore, the possibility of quickly capturing several time frames of an object allows for time-resolved CT, also known as dynamic CT. Because the scans must be performed quickly to capture the time evolution of an object, they are highly prone to noise. Better denoising techniques may allow for improvements in dynamic CT.

1.2 Goal of Work

The goal of this thesis is to use GAN based image denoising to improve the image quality in CT, in cases where the experimental datasets are noisy and/or undersampled. The TomoGAN [7] denoising neural network will be tested, and its limitations will be explored. An analysis of when this denoising technique is suited will be given. This thesis will focus on undersampling artifacts, and not quantum noise (see Section 2.1.1).

Based on articles citing a log-cosh based loss function for training neural networks as a good candidate for image processing related tasks, the effect of changing the loss function used to train TomoGAN will be explored [8, 9].

The feasibility of using a GAN based denoising method for dynamic (i.e. time-resolved) CT scans will also be explored.

1.3 Thesis Structure

This thesis begins with Chapter 1 containing an introduction to the thesis, including its motivation and goals, and how the thesis is structured. It is then followed by two chapters of theory.

Chapter 2 will introduce the necessary theory to understand how CT images are captured, what the primary causes of noise in them are, and how the reconstruction process works. Next, Chapter 3 gives a basic introduction to ML, and more precisely neural networks and the theory needed to understand how GANs work.

Chapter 4 will present the structure of the specific GAN network used in this thesis, namely TomoGAN [7]. Following that, some image comparison metrics that will be used in the discussion are presented. The datasets that are used in this thesis will be presented in this chapter, and the method used for compiling a given dataset into a suitable format for the GAN to use is given.

In Chapter 5, the results of the denoising will be presented and discussed using different visualization methods and image comparison metrics defined in Section 4.2.

Finally, Chapter 6 will conclude this thesis and note some possibilities of further work.

Chapter 2

X-ray Computed Tomography

While this thesis is primarily focused on noise reduction using generative adversarial network (GAN)s, a brief overview of some of the basic principles behind computed tomography (CT) imaging will be given in this chapter. The focus will be on explaining the underlying theoretical foundation of CT imaging, including the main sources of noise. A brief description of a CT imaging setup is provided. An overview of common reconstruction methods, as well a more novel one, will also be presented.

2.1 X-ray Attenuation in a Sample

X-ray CT imaging is based on interaction between X-rays and matter. This interaction will attenuate X-rays that propagate through a sample according to the Beer-Lambert law, which is given as [10] [1]

$$I = I_0 \mathrm{e}^{-\int_{\mathrm{l}_0}^{\mathrm{l}} \mu(\mathrm{x,y,E})\mathrm{dl}}, \tag{2.1}$$

where I_0 and I are the incident and the attenuated X-ray beam intensities at positions l_0 and l respectively, and μ is the attenuation coefficient of the traversed matter. The integral in the exponent is the path of the photon beam through the sample [1]. The attenuation coefficient is dependent on energy (E), and typical X-ray CT systems span a range of wavelengths (i.e. energies). Because of this, Equation (2.1) must be modified to also account for the polychromatic nature of the X-ray source.

The total incident radiation can be determined by integrating over all photon energies,

$$I_0 = \int_{E_{min}}^{E_{max}} N(V,I) S(E) D(E) \, dE, \tag{2.2}$$

[1]Photons will not only attenuate, but also scatter as they propagate through the object. This effect is generally seen as a source of noise in CT imaging, and the effect of the x-ray scattering is often removed through modifying the setup or using correction algorithms [11].

with $N(V,I)$ being a variable introduced to account for photon flux depending on X-ray source tube voltage V and current I, $S(E)$ being the normalized X-ray source spectrum modulated by the absorption materials between the source and the detector (not including the sample), and $D(E)$ being the detector sensitivity modulated by protection materials on the detector. E_{min} and E_{max} bound the energy range of the radiation spectrum.

By combining Equations (2.1) and (2.2), we get the modified Beer-Lambert law accounting for the polychromatic X-rays [10],

$$I = \int_{E_{min}}^{E_{max}} N(V,I)S(E)D(E)\mathrm{e}^{-\int_{\mathrm{l_0}}^{\mathrm{l}} \mu(\mathrm{x,y,E})\mathrm{dl}}\mathrm{dE}. \tag{2.3}$$

This can be solved for the attenuation coefficient projection, giving [10]

$$\int_{l_0}^{l} \mu(x,y)\,dl = -\ln\left[\frac{\int S(E)D(E)\mathrm{e}^{-\int_{\mathrm{l_0}}^{\mathrm{l}} \mu(\mathrm{x,y,E})\mathrm{dl}}\mathrm{dE}}{\int S(E)D(E)\,dE}\right]. \tag{2.4}$$

The attenuated intensity I can be related to the attenuation coefficient projection μ of a path through the sample by use of this equation. When the attenuation coefficient projections are known, the 2D attenuation coefficient map can be reconstructed using a reconstruction algorithm if a sufficient number of projections are available. Strategies for numerical CT reconstructions will be described in Section 2.3.

An illustration of how a projection of an object relates to the Fourier transform of an object is provided in Figure 2.1. By collecting this projection for several angles, the entire Fourier transform of the object can be collected. This is known as the Fourier slice theorem [1, p. 57].

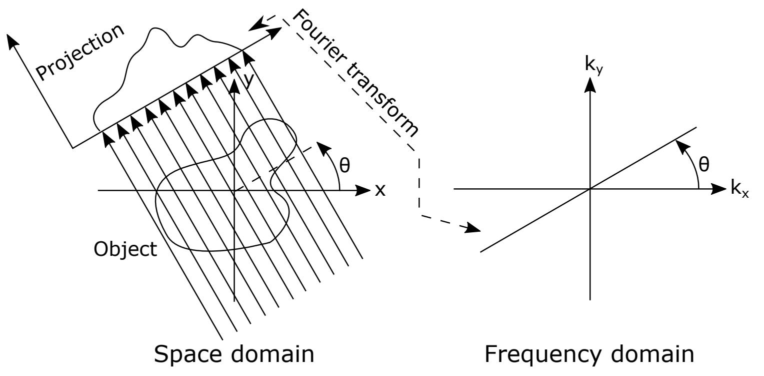


Figure 2.1: Illustration of how the projection of an object corresponding to an angle θ relates to the Fourier transform of the object. Adapted from [1].

2.1.1 Noise

Assuming a sufficient number of projections of the attenuation coefficient are available, the primary sources of noise in CT measurements are quantum noise and electronic noise [12].

The quantum noise, sometimes also known as shot noise or simply Poisson noise, is due to the statistical error of low photon counts. It can be modeled as a Poisson distribution [13],

$$P(X = x) = \frac{\mathrm{e}^{-\mathrm{xm}}\mathrm{m}^{\mathrm{x}}}{x!}, \tag{2.5}$$

with m being the mean signal value, $x \in \mathbb{N}^+$ being an integer representing the measured signal value, and X being a random variable denoting the number of photons generated by the X-ray source. Quantum noise can be reduced simply by increasing the incident X-ray beam intensity, however this is often not wanted as increasing the radiation dose has raised concerns about potential health risks [14, 15].

Electronic noise is related to the electronics of the X-ray detector, and it is modeled as additive white Gaussian (i.e. normal) noise [12],

$$\mathcal{N}(0,\sigma) = \frac{1}{2\pi\sigma}\mathrm{e}^{-\frac{\mathrm{x}^2}{2\sigma}}, \tag{2.6}$$

which corresponds to a normal distribution with mean signal value of 0 and standard deviation σ.

If an insufficient number of projections of the attenuation coefficient are available, it is known as a *missing wedge problem* or an *undersampling problem* [16]. These measurements are incomplete datasets with respect to standard requirements of established reconstruction algorithms. This leads to artifacts that appear as elongations of reconstructed details along the mean direction (i.e. the symmetry centre of the projections). An illustration of undersampling and missing wedge in Fourier space is provided in Figure 2.2. Several different reconstruction methods removing these artifacts have been attempted, also including ML based approaches [6, 7, 16].

A comparison of quantum noise and undersampling artifacting on a dataset imaging borosilicate glass beads is given in Figure 2.3. More details on the dataset are provided in Section 4.3.1.

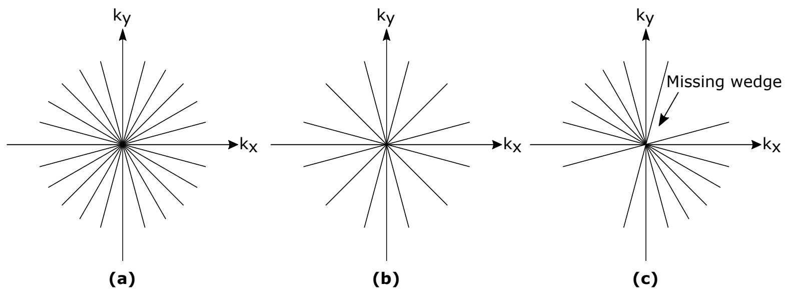


Figure 2.2: Illustration of undersampling and missing wedge in Fourier space. The figure contains: **(a)** sufficiently many projections, **(b)** undersampled projections, and **(c)** missing wedge projections.

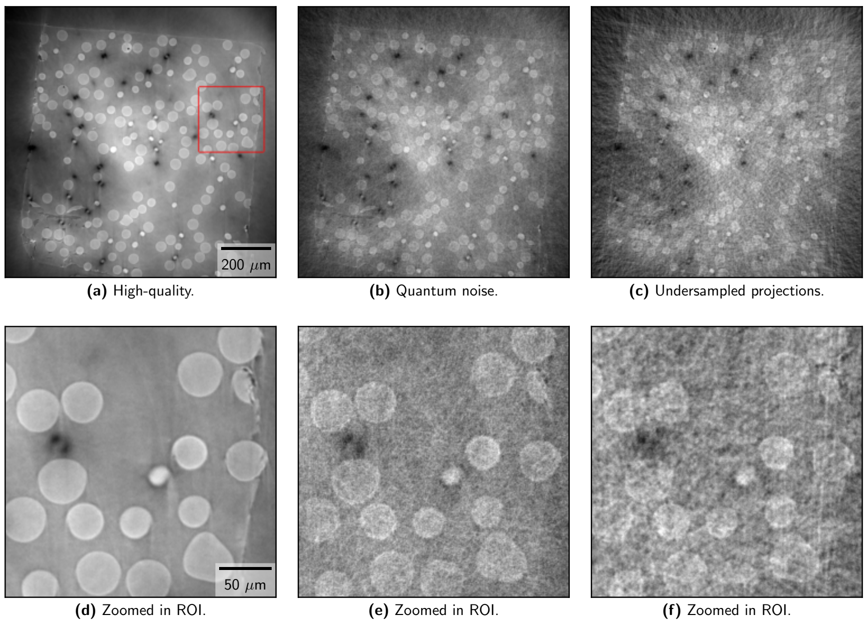


Figure 2.3: Comparison of a high-quality reconstruction, a quantum noise reconstruction, and an undersampled reconstruction. Images d), e), and f) are zoomed in regions of interest (ROIs) of images a), b), and c) respectively. The ROI is marked in a). The quantum noise is simulated by applying Poisson noise to the sinogram before reconstruction, and the undersampled reconstruction is simulated by selecting a subsampling of every 32nd projection from the high-quality reconstruction (i.e. uniformly subsampled projections). The images are of a central slice from the dataset tomo_00058 [17] reconstructed using filtered back projection (FBP) from TomoPy [18].

2.2 Imaging Method

In this section, the process of collecting x-ray CT images will be presented briefly.

2.2.1 Imaging Setup

There are different types of CT imaging systems. In Figure 2.4, a simple schematic of a *micro-CT* system is provided, where the object itself is the part of the system that is rotated. Other types of CT systems may instead rotate the x-ray source and/or detector around the object [1, pp. 126–129].

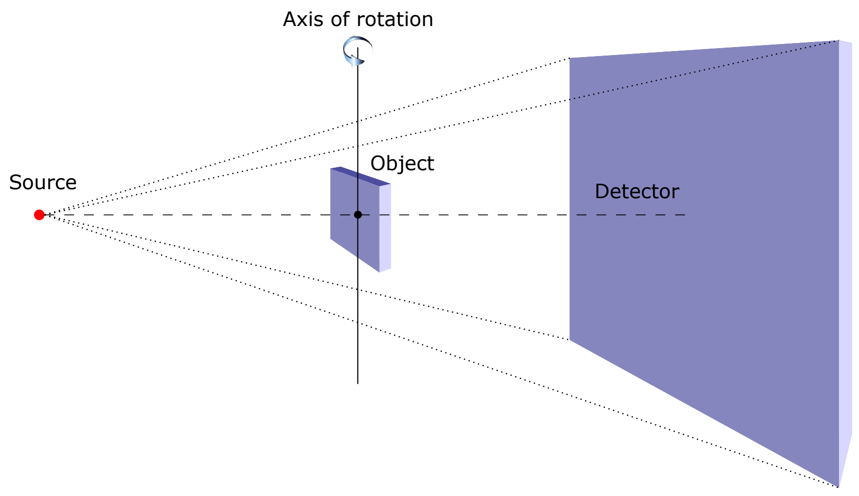


Figure 2.4: Illustration of a micro-CT imaging setup. The object is rotated around the axis of rotation to capture projections from several angles, and the x-ray source and detector are stationary.

There are typically three different detector types used. The detector output may be proportional to the total number of photons incident on it, it may be proportional to the total photon energy, or it may be dependent on energy deposition per unit mass. These three different measurement types typically correspond to counting-type detectors, scintillation-type detectors, and ionization detectors, respectively [1, p. 118].

The projections of a slice of the object correspond to one line on the detector for several angles of rotation. By collecting this projection along the same line on the detector while rotating the object, a *sinogram* of a slice of the object can be formed.

2.2.2 Sinograms

The captured projections form a *sinogram*. The sinogram of an image consists of several projections corresponding to the Radon transform of the image (see Section 2.3) from different angles, where each projection is one line in the sinogram. An example is provided in Figure 2.5.

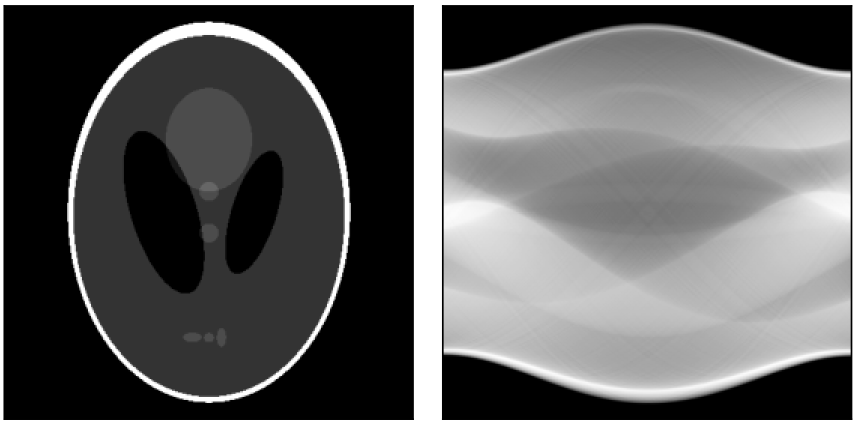

(a) Image. **(b)** Sinogram.

Figure 2.5: Illustration of an image and its sinogram. The image is the Shepp-Logan phantom [19]. The sinogram is created by plotting the Radon transform (see Section 2.3) of the image for 400 angles from 0 — 180 degrees.

2.3 CT Reconstruction

After acquiring the attenuation coefficient projections, or sinograms, this data must be reconstructed into an image (of either 2D or 3D) of the object. The collection of a sinogram P_θ for projection angle θ is given by the Radon transform [20, 21]

$$P_\theta(u) = \iint f(x,y)\,\delta\,(x\cos\theta + y\sin\theta - u)\,dx\,dy, \tag{2.7}$$

where u is the position on the detector and δ is the Dirac delta function. In practice, because of computational and instrumental limitations, the projection data are acquired only for a limited number of projections N_θ as well as N_d detector elements, and the imaged object is represented by a pixel grid of size $N \times N$. Thus, the acquired projection data is described by a vector $\boldsymbol{y} \in \mathbb{R}^{N_\theta \times N_d}$, the reconstructed image by a vector $\boldsymbol{x} \in \mathbb{R}^{N \times N}$, and the formation of the projection data can be stated as a linear system [21]

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}, \tag{2.8}$$

where element $a_{ij} \in \mathbb{R}$ of $\boldsymbol{A} \in \mathbb{R}^{N_\theta N_d \times N^2}$ is equal to the contribution of image pixel j to detector pixel i. This gives the tomographic reconstruction problem of recovering the unknown object $\boldsymbol{x}$ from the acquired projection data $\boldsymbol{y}$. It can be seen as performing the *inverse* Radon transform [6, 22].[2]

[2]The inverse Radon transform is an *inverse* problem. Inverse problems are often ill-posed. An ill-posed problem is a problem that does not meet any one or more of the three conditions suggested by

Conventional tomographic reconstruction algorithms are generally divided into two groups: direct reconstruction, and iterative reconstruction, however a third method using ML has also shown promise [6].

2.3.1 Direct Reconstruction

Direct tomographic reconstruction algorithms are based on finding an inversion formula of the continuous forward Radon transform, as given in Equation (2.7). The numerical implementation is done by using a discretized inverse Radon transform [21]. The most commonly used direct reconstruction algorithm is the filtered back projection (FBP) algorithm, which can be written as [21]

$$\boldsymbol{x}_{FBP} = \boldsymbol{A}^T \boldsymbol{C}_h \boldsymbol{y}, \tag{2.9}$$

where $\boldsymbol{C}_h$ is a 1D convolution operation that convolves each detector row in $\boldsymbol{y}$ with a filter $\boldsymbol{h} \in \mathbb{R}^{N_d}$. The filter is typically some standard filter that can be used for any reconstruction (e.g. the Ram-Lak filter), and may include low-pass filtering to reduce high frequency noise in the reconstructed image [23]. It has also been shown that this filter can be learned by use of ML (more specifically an artificial neural network (ANN), see Section 3.1) to further improve the performance of FBP [24].

Direct algorithms, such as FBP, have the advantage of most often being computationally efficient, as well as producing accurate results when enough projections are available [21]. It can be shown that the sufficient number of projections for FBP is roughly the same as the number of rays per projection (i.e. number of pixels across the object) [1, pp. 59, 183–186]. The issue with these techniques arises when only a limited number of projections are available, as they are generally highly prone to noise leading to insufficient image quality for further analysis [21]. This is where the use for iterative reconstruction algorithms arises.

For CT imaging systems where the X-ray beam is conical (as opposed to parallel), an alternative direct reconstruction method similar to FBP is the Feldkamp-Davis-Kress (FDK) reconstruction algorithm [25].

2.3.2 Iterative Reconstruction

Iterative tomographic reconstruction algorithms are based on iteratively solving the linear system given in Equation (2.8). Iterative CT algorithms can provide reconstructions with fewer artifacts and less noise than the FBP algorithm, in particular when using few measured projections. A common method used is to find images that minimize the l^2-norm of the residual error (i.e. the difference between the acquired sinograms, and the Radon transform of the reconstructed image[3])

Jacques Hadamard: existence, uniqueness, and stability [22]. The stability condition is most often violated. This means that its output is highly sensitive to small changes in the input (e.g. noise can drastically change the resulting output).

[3]Calculating the inverse Radon transform is an ill-posed problem and is challenging, however calculating the Radon transform itself is an easy task.

as well as an additional term g that penalizes images that do not confine to some prior knowledge or assumption of the imaged object. This process can be written as [21]

$$\boldsymbol{x}_{iter} = \underset{\boldsymbol{x}}{\operatorname{argmin}} ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2^2 + \lambda g(\boldsymbol{x}), \tag{2.10}$$

where $||\cdot||_2^2$ denotes the l^2-norm, and λ is the relative weight of the prior knowledge penalty compared to the residual error. If a prior knowledge penalty that fits a reconstruction well is chosen, iterative reconstruction algorithms can produce significantly more accurate reconstructions than direct methods when reconstructing from limited data [21]. However, if the chosen prior knowledge penalty does not fit well, or if the weighting parameter λ is poorly selected as it is a problem-dependent parameter, it may lead to poor quality reconstructions. Some of the commonly used iterative reconstruction algorithms are the algebraic reconstruction technique (ART) and the simultaneous iterative reconstructive technique (SIRT) [1, pp. 283–284].

A large drawback with iterative reconstruction is its (often) large computational cost. These types of reconstructions are slower, which may make it difficult to apply them to time-sensitive real-world tomographic data [21]. Newer and more powerful computers can to some extent offset this downside to iterative reconstructions [26]. Because of these limitations and drawbacks, direct reconstruction algorithms are still often preferred in many fields [27].

Prior image constrained compressed sensing (PICCS) is an iterative reconstruction algorithm [28]. It is used to better reconstruct datasets that are limited in the number of projections available when multiple imagings have been done for several time frames (known as dynamic CT imaging). By reconstructing a prior image from the union of interleaved datasets from several time frames (that are undersampled on their own), the PICCS reconstruction algorithm utilizes the spatial-temporal correlations in the imaging to make assumptions on the imaged object. The prior image is used as a constraint on the reconstruction of the undersampled reconstructions of each time frame.

2.3.3 Machine Learning Based Reconstruction

The terms used in this subsection will be properly introduced in Chapter 3. Furthermore, this is not the reconstruction technique used in this thesis, however it is worth mentioning as it has shown promise for reconstruction quality improvements.

In addition to direct and iterative reconstruction techniques, ML has been used to make an iterative-like reconstruction algorithm [6]. One method, termed GAN-rec, is based on a GAN (see Section 3.2.3) and is an ML method that does not require training of the network before reconstruction, instead using the training process as the reconstruction process.

It takes a given sinogram $\boldsymbol{y}$ and uses the generating network G to create a candidate reconstructed image $\boldsymbol{x} = G(\boldsymbol{y})$, then creates the corresponding candidate sinogram $\hat{\boldsymbol{y}} = P(\boldsymbol{x})$, where P is the Radon transform. The loss $L = ||y - \hat{y}||$ is then the basis of training the network (i.e. reconstructing the image).[4]

The sinogram-to-reconstruction transformation cannot be done by a conventional convolutional neural network (CNN)-style network, however it has been shown that a single fully connected layer can perform this transformation [29]. The accuracy of the transformation can be improved by increasing the number of layers and neurons (see Section 3.1), however this is dependent on the available computational power [6]. Because of this limitation, the generating network in GANrec is a modified version of U-net [30] (see Section 3.2.2), with three fully connected layers at the start to perform this transformation.

[4]The actual loss used for training GANrec also includes an adversarial loss, which will be introduced in Section 3.3.2, however it is omitted in this description for the sake of simplicity.

Chapter 3

Machine Learning

The term machine learning (ML) was coined by Arthur Samuel in 1959 [31]. An ML algorithm builds a model based on a dataset, intended to make predictions or classifications without being explicitly programmed how to do so.

Whereas some problems can easily be solved by programming an explicit algorithm (e.g. sorting a list, or FBP reconstruction), there are many cases where an exact algorithm fails to provide adequate solutions to the problem. A typical example is to filter out spam emails from an email inbox. The content and structure of the spam emails vary sufficiently to prevent filtering them with "hardcoded" rules, as in conventional algorithms. This is where ML comes in: an ML model can be trained to discern differences in a dataset without being explicitly told what to look for. So long as there is a sufficient amount of data to train the model with, it may be able to find a pattern in the data and thereby predict or classify new data, or augment or enhance the data [32, pp. 2–4].

There are many different ML algorithms, however in this thesis, only the class of *neural networks* will be discussed and the focus will be on *supervised learning*.

This chapter contains a brief introduction to neural networks and their basic components, an explanation of what a convolutional neural network (CNN) is, a description of encoder-decoder networks, and an introduction to GANs, before covering the basics of how a neural network is trained and giving an overview of some common loss functions used for this process.

3.1 Components of a Neural Network

Neural networks are computing systems that are designed to learn in ways similar to the human brain by being exposed to large amounts of data, and attempting to find some inherent pattern or system to the data.

Neural networks were initially designed to simulate the human brain and how it learns and adapts to new information [33]. Because of this, the basic building block of a neural network is called a neuron. Several neurons build up a layer, and several layers build up a neural network. Neurons in different layers have connections to each other (i.e. neurons in layer 1 are connected to neurons in layer 2), these connections have weights, and each neuron has a bias. A simple schematic of this is given in Figure 3.1. The value of a neuron is a real number and can be given as [34, p. 81]

$$Y_k = \sigma\left(\sum_{j=0}^{m} w_{kj}x_j + \lambda_k\right), \tag{3.1}$$

where k refers to which neuron it is, m is the number of inputs to the neuron, w_{kj} is the weight of connection j, x_j is the output value of neuron j into neuron k, λ_k is a bias term, and σ is the activation (or transfer) function, which will be introduced later. It is thus a weighted sum of the values of the neurons in the previous layer (or more precisely, of all the input neurons to a given neuron, which often is the previous layer) and the neuron's own bias, passed through an activation function. Note that this describes a simple fully connected feedforward ANN, more precisely a *multilayer perceptron*, and other types of neural networks may contain other types of layers [34].

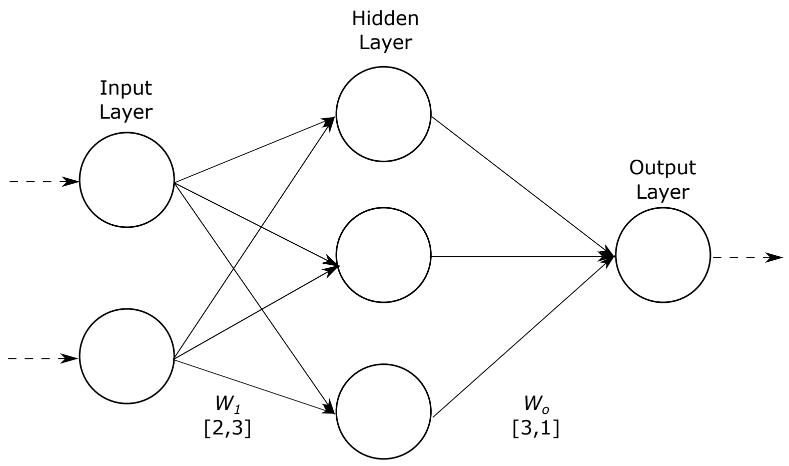


Figure 3.1: Illustration of a neural network. Each circle represents a neuron, the solid arrows represent connections between neurons, and the dotted arrows represent input and output channels. The dimensions of the network parameters are denoted as W_n, where n refers to the layer the parameters input into. This network specifically is a fully connected feedforward ANN, also known as a multi-layer perceptron, with one hidden layer.

The activation function, also known as the transfer function, is denoted as σ. Its purpose is to bound the value of a neuron so that the network does not diverge during training because of neurons with diverging values [34, p. 81]. Furthermore, the activation function is used to introduce nonlinearity to the network,[1] and it can be shown that a two-layer deep neural network with a nonlinear activation function is a *universal function approximator*, meaning it can approximate any

[1]For this reason, the identity activation function $f(x) = x$ generally performs poorly.

function mapping between two Euclidean spaces [35]. There are many different activation functions, and some examples are presented in Table 3.1 and plotted in Figure 3.2.

Table 3.1: Overview of some of the commonly used activation functions in neural networks.

Name	Function, $f(x)$
Identity	x
Rectified Linear Unit (ReLU)	$\max(0, x)$
Leaky Rectified Linear Unit (LReLU)	$\max(\alpha x, x), \alpha \in [0, 1]$
Logistic/soft step	$\frac{1}{1+e^{-x}}$
tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$
Softplus	$\ln(1 + e^x)$

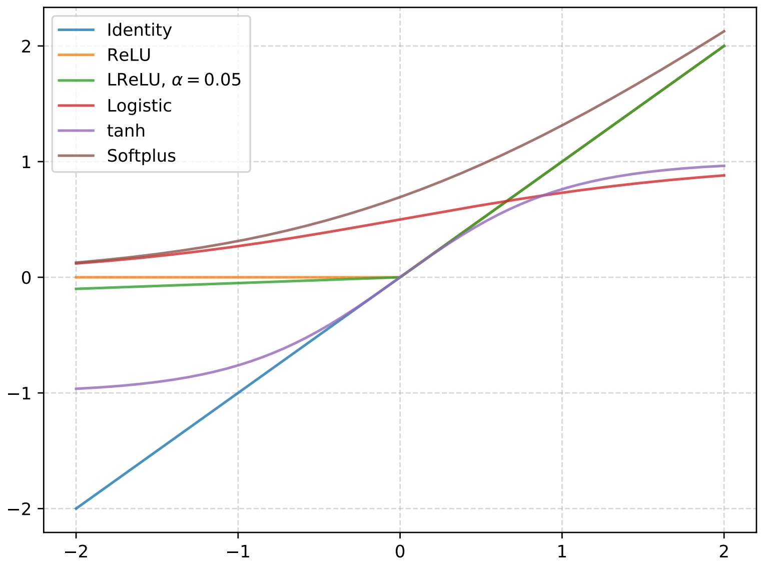


Figure 3.2: Plot showing a selection of activation functions for $x \in [-2, 2]$. Note that identity, ReLU, and LReLU are overlapping for $x \in [0, 2]$. See Table 3.1 for definitions of the functions.

The output of a neural network can be defined to be any shape (e.g. a vector, or matrix). In Figure 3.1 the output is a single value, however it could just as well have been defined as a vector. If the output is a single value it can for instance be interpreted as a probability, however if it is a vector of length n it can be seen as n probabilities of different events or features. The output of a neural network is often called a feature map, because it can be seen as a mapping of the features of the input data.

For example, if a neural network is trained with a dataset containing images of handwritten digits 0−9, an output with a size of 10 could contain probabilities of a given image containing a specific digit where each output value is the probability of one digit. One well-known dataset that is often used for this exact problem is the MNIST dataset [36].

3.2 Neural Network Types

There are many different types of neural networks that are suited for different problems. Here, a selection of types that lead up to the GAN structure used in this thesis will be introduced.

3.2.1 Convolutional Neural Network

A convolutional neural network (CNN) builds upon the structure of the ANN, however it adds a new type of layer: the convolutional layer. Instead of containing a set of neurons, this layer contains one (or more) convolutional kernel(s), and performs a convolution of the input to the layer, with the kernel(s). This type of network was first introduced in 1999,[2] and has shown to perform well for many different image related tasks [37, 38]. The convolution operation allows the network to utilize 2D information by performing 2D convolutions.[3]

The discrete convolution operator is defined as [40, pp. 899–901]

$$g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x+dx, y+dy), \tag{3.2}$$

where $g(x,y)$ is the convoluted matrix, $f(x,y)$ is the original matrix, and ω is a convolution kernel of dimension $(2a+1, 2b+1)$.[4] For the sake of simplicity, kernel dimensions will be referred to as $a \times b$ where a and b represent the kernel dimensions, and not the half-dimension as would correspond to Equation (3.2).

A visualization of the convolution of a matrix (which could represent an image) with a given kernel is provided in Figure 3.3. Here, the kernel dimensions are 3×3. The output matrix has reduced dimensions corresponding to the kernel dimensions. This reduction can be given as

$$(x_o, y_o) = (x_i - (a-1), x_i - (b-1)), \tag{3.3}$$

where (x_o, y_o) are the output dimensions, (x_i, y_i) are the input dimensions, and a and b are the kernel dimensions. In some situations it may not be wanted to reduce the dimensions of the input, and padding the input with zeroes on all sides can be used to combat this. This technique is called *zero-padding* [41].

[2]There is some disagreement around whether the paper by LeCun in 1999 [37] was truly the introduction of CNNs, however it is often seen as it.

[3]Likewise higher-dimensional information may be used by performing higher-dimensional convolutions [39].

[4]The dimensions of the kernel are typically square and odd, such as 3×3 or 5×5, giving $dx, dy \in [-1,1]$ or $dx, dy \in [-2,2]$.

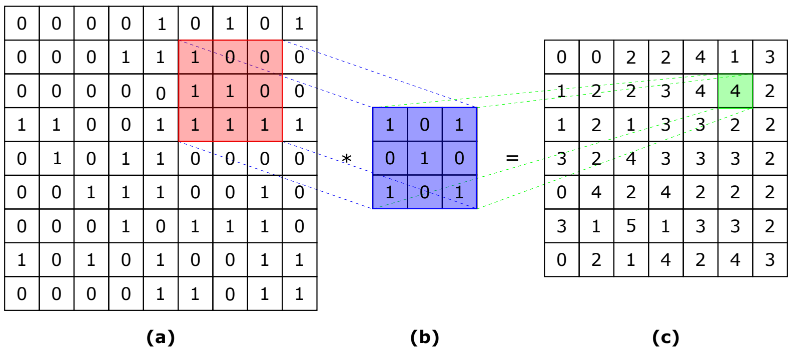


Figure 3.3: Illustration of a 2D discrete convolution operation: a) an input matrix of dimension (9, 9), where the numbers can e.g. represent intensities in a gray-scale image, b) a 3×3 convolutional kernel, and c) the resulting convolution of dimension (7, 7). This convolution has a stride of 1. Note that the output dimension is smaller than the input dimension.

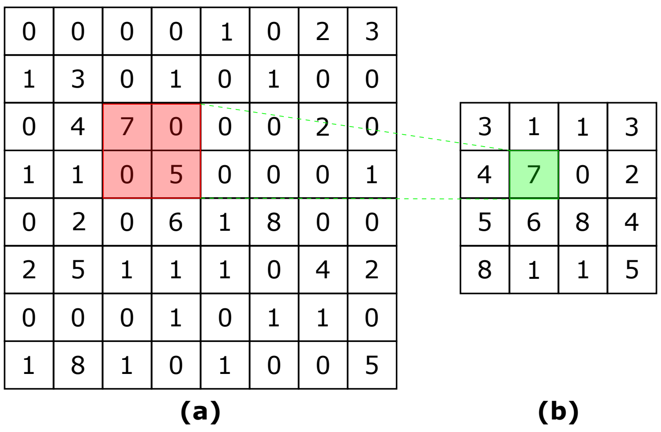
Multiple convolutional kernels can be used in parallel in each convolutional layer. The number of kernels is then referred to as the *number of channels*.

The *stride* of a convolution is how far the kernel shifts [41]. In the example given in Figure 3.3, the stride is 1. If the stride were set to 2, the kernel would shift two units in the matrix for each output. This would mean less overlap between each value in the output, but also further reduction of the output dimensions.

Often, *pooling layers* are included in CNNs. These layers are used to down sample the feature maps after a convolutional layer by applying some pooling function (e.g. max, average, sum) to an area of the feature map, reducing the dimension. This can for instance be a 2×2 max pooling layer that looks at a 2×2 section of a feature map and replaces it with a single value corresponding to the maximum value of the original section. Pooling layers reduce the dimensions of the feature map corresponding to the size of the pooling (e.g. a 2×2 pooling layer reduces both dimensions of a feature map by a factor of 2). An example is provided in Figure 3.4.

The part of the convolution that a CNN learns is the values in the convolutional kernel. Each layer of the CNN may have several kernels that are applied in parallel (e.g. 32 kernels applied to the same input). Each kernel is often called a filter. One of the advantages of using convolution in neural networks is the reduction in the number of trainable parameters: a typical convolutional kernel contains $9 - 49$ parameters (for kernels of dimensions 3×3 to 7×7), however a fully connected feedforward network may have several thousand parameters for each layer.[5]

[5]Consider an image of dimension 100×100. In a fully connected ANN there would have to be $100 \cdot 100 = 10^4$ connections from each neuron in one layer to the next layer for a total of 10^8 connections, where each connection has a trainable parameter (not counting the biases). In comparison, if using a CNN with 32 channels (or kernels) of dimension 3×3 there are only a total of $32 \cdot 3 \cdot 3 = 288$ trainable parameters.



0	0	0	0	1	0	2	3
1	3	0	1	0	1	0	0
0	4	7	0	0	0	2	0
1	1	0	5	0	0	0	1
0	2	0	6	1	8	0	0
2	5	1	1	1	0	4	2
0	0	0	1	0	1	1	0
1	8	1	0	1	0	0	5

3	1	1	3
4	7	0	2
5	6	8	4
8	1	1	5

Figure 3.4: Illustration of a 2D 2×2 max pooling with a stride of 2: a) an input matrix of dimension $(8,8)$, where the numbers can e.g. represent intensities in a grayscale image, and b) the resulting max pooled matrix of dimension $(4,4)$. Every element in b) corresponds to the maximum of four elements in a).

3.2.2 Encoder-Decoder Network

An encoder-decoder network is a type of ANN that learns to copy its input to its output [42]. It consists of two parts (as the name suggests): an encoder, and a decoder. The task of the encoder is to take the input and encode it into a feature map. The decoder then takes the resulting feature map and decodes it into an output similar to the original input. An illustration of this structure is given in Figure 3.5. The encoder's goal is to extract the relevant information from the input, ignoring any signal noise or unwanted data. The decoder then recreates something similar to the original data from the "denoised" feature map. It is common to use CNNs as both encoder and decoder, where consecutive layers in the encoder reduce the dimensions of the feature maps and consecutive layers in the decoder increase the dimensions of the feature maps. Encoder-decoder networks have been shown to perform well in many different tasks, such as image segmentation [43] and PET image reconstruction [44].

In an encoder-decoder network, the encoder and decoder are two separate networks that can work independently of each other. Another similar network structure that builds upon the encoder-decoder is the *U-net* convolutional network, originally proposed for biomedical image segmentation [30]. It also contains an encoder and a decoder part, however the two networks are not separable as there are skip-connections between layers in the encoder and decoder. In a normal encoder-decoder network, there is first one mapping from the input X to the feature map L, $E\colon X \mapsto L$, and then a mapping from the feature map L to the output Y, $D\colon L \mapsto Y$. These two mappings are not dependent on each other. In the U-net architecture however, the mapping in the decoder also depends on the input X, making it $D\colon [X + L] \mapsto Y$.

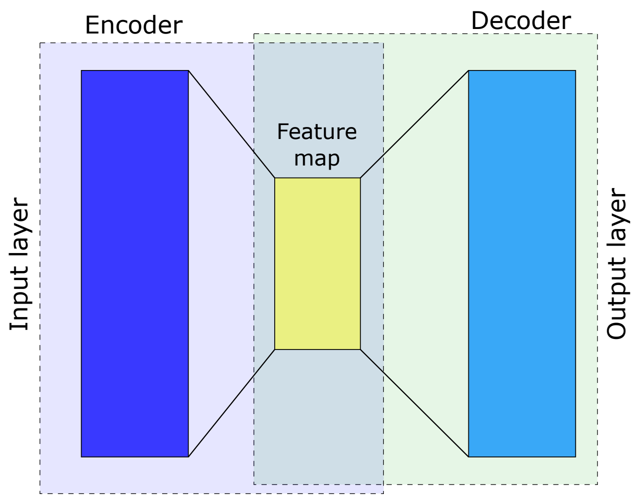


Figure 3.5: Illustration of the general structure of an encoder-decoder network. It consists of two separable networks, the encoder and the decoder, working together. The input and output layers are of the same dimensions.

3.2.3 Generative Adversarial Network

Generative adversarial networks (GANs) were introduced in 2014 by Goodfellow *et al.* as a novel method of estimating generating models via an adversarial process [3]. This type of neural network consists of two separate networks: a generator and a discriminator. The generator, called G, captures the distribution of the training data and generates new samples from that distribution, while the discriminator D estimates the probability that a given sample came from the training data (i.e. is a real sample) rather than being a generated sample from G. An illustration of the GAN structure is given in Figure 3.6.

The two networks play a game where they try to minimize their own cost, or error rates, while at the same time maximizing the other network's cost [45].[6] As opposed to normal neural networks that are based on optimization to reduce their error rates, GANs are based on game theory [45].

To generate random samples from the distribution of the training data, a fully trained GAN is given random noise as input and then maps that to a random sample, such as was done by Zhang *et al.* [46]. This allows the network to generate new samples that are similar, but not equal, to the training data. Another common use case for GANs is, instead of feeding the network random noise as input, feeding it some data that needs augmentation. This has been used to denoise images and has been shown to be a viable method for image super-resolution [5, 47].

GANs are generally seen as unsupervised learning algorithms with a super-

[6]GANs are designed to reach a Nash equilibrium at which neither of the two networks can reduce its costs without changing the other network's parameters [7].

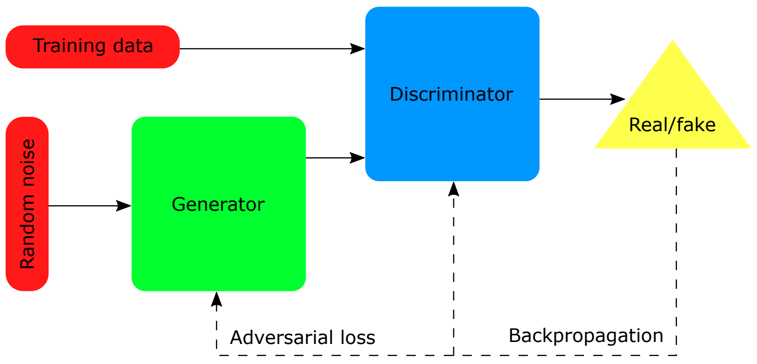


Figure 3.6: Illustration of a GAN structure. The generator takes a random input and attempts to generate a sample similar to the training data. The discriminator attempts to distinguish a training data sample (i.e. real sample) from a generated sample. The output from the discriminator is used to train the generator to better generate samples similar to the training data, as well as the discriminator to better distinguish between real and generated samples.

vised loss as part of the training. For a general GAN, the training data is an unlabeled dataset and the GAN tries to model a probability distribution of the dataset in order to randomly generate new samples. By generating new samples, it is trivial to apply labels to the original (i.e. real) and generated samples and then use these labels to perform supervised learning to train the discriminator.

When the input dataset to the GAN no longer only contains a set to learn the distribution of, but instead the GAN training process is used to train a generator that takes a non-random input and outputs an augmented version of the input (such as was done by Liu *et al.* [7]), it can be seen as a supervised learning algorithm.

3.3 Training a Neural Network

The process of tuning all the parameters (i.e. weights and biases) of a neural network is called training. During training, input data from a training dataset is forwards propagated through the network, and the resulting feature map is compared to an expected feature map (e.g. manually labeled data).[7] The difference in these feature maps is calculated using a loss function, and the loss is then backward propagated through the network to update each and every parameter to reduce the loss.

Generally, the entire training dataset is repeatedly passed through the network multiple times. Each full runthrough of the training dataset is called an *epoch* of

[7]This is what is called supervised learning, as opposed to unsupervised learning where there is no ground truth answer to compare to, instead the network is trying to learn some inherent structure of the data without explicit labels (e.g. clustering).

training. This however can often introduce a problem: the training dataset can typically not fully fit in the computer memory at once. Therefore it is divided into mini batches, and after each mini batch the weights are adjusted. The propagation of one mini batch is often called one iteration, and thus one epoch consists of several iterations. The size of a mini batch is a tunable parameter, however typically it is in the range of $32-512$ (e.g. 128 in the well-known article by A. Krizhevsky *et al.* [38]).[8] The size of a mini batch can sometimes also be referred to as the batch size.

3.3.1 Hyperparameters

During training, the parameters of the neural network are automatically changed, however there are some parameters that are set manually beforehand. These are called hyperparameters [49]. Some typical hyperparameters are:

- Number of layers (i.e. depth of network).
- Size (or dimensions) of layers.
- Learning rate.
- Number of iterations to train the network (or number of epochs).
- Mini batch size.

The choice of hyperparameters varies with the problem at hand, and is often determined by trial and error. Hyperparameter optimization, allowing automatically tuning hyperparameters for a wide range of scientific problems, is an important topic in machine learning research [50].

3.3.2 Loss Functions

Loss functions are mathematical descriptions that quantify the difference between the feature maps generated by the network, and the expected features (i.e. labels) of the training data. Depending on the problem type, different loss functions may perform better than others, however there are some standard loss functions often used. Some of these, as well as some specific ones used in this thesis, will be presented here. The losses are calculated on a per-pixel basis and summed unless otherwise stated.

Perhaps the most commonly used loss function is the mean squared error (MSE). It is closely related to the L2-norm,[9] and it can be defined as

$$L_{\mathrm{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2, \tag{3.4}$$

[8]There is ongoing research into techniques to increase the batch size by several orders of magnitude as larger batches allow for easier parallelization, however large batch sizes have been shown to cause instability during training [48].

[9]Sometimes the MSE loss is improperly called the L2-norm, however that is incorrect. The L2-norm can be defined as the square root of MSE.

where Y is the correct (labeled) value, $\hat{Y}$ is the predicted value, and N is the number of samples. This often performs well, however in cases such as image processing or image super-resolution it has been shown to cause blurring [9].

Another similar loss function is the mean absolute error (MAE), which is closely related to the L1-norm. It can be defined as

$$L_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N} |Y_i - \hat{Y}_i|, \tag{3.5}$$

with Y and $\hat{Y}_i$ being the same as previously defined. This loss function does not over-penalize larger errors, and therefore may have different convergence properties than MSE [9]. It has been shown to perform better than MSE in some image processing cases [9, 51].

A more recently introduced loss function is the log-cosh loss function, defined as [8]

$$L_{\text{Log-cosh}} = \frac{1}{a}\sum_{i=1}^{N} \log(\cosh(a(Y_i - \hat{Y}_i))), \tag{3.6}$$

where Y and $\hat{Y}$ are as previously defined, log is the logarithm, cosh is the hyperbolic cosine function, and a is some positive hyperparameter $a \in \mathbb{R}^+$. It behaves similar to MSE around the origin, and similar to MAE at other points. It has been shown to perform well in image processing-related tasks [9].

All the aforementioned loss functions rely on pixel-wise losses. Another type of loss function that has been shown to perform well in image processing-related tasks is the use of a *feature space-based loss* [52]. In the case of image processing, it means that the loss is based on measuring the difference in the feature space of the *inference* of a pre-trained network.[10] Here, the pre-trained VGG network is used to measure a visual loss [53]. This specific loss function is termed visual loss, or VGG loss, and is defined as [7, 52]

$$L_{\text{VGG}} = \sum_{i=1}^{N}\sum_{j=1}^{W_f}\sum_{k=1}^{H_f} \left(V_{\theta_{\text{VGG}}}(Y_i)_{j,k} - V_{\theta_{\text{VGG}}}(\hat{Y}_i)_{j,k}\right)^2, \tag{3.7}$$

where Y and $\hat{Y}$ are as previously defined, $V_{\theta_{\text{VGG}}}(Y)$ is the VGG feature map representation of image Y, and W_f and H_f are the dimensions of the feature maps extracted by the pre-trained VGG network. The VGG network is trained with natural images, specifically the ImageNet dataset [54], however it has been shown to work well as a feature extractor for CT images [55].

Specific to GANs is the adversarial loss. It is a measure of how well the generator network is able to produce samples that the discriminator network is unable to distinguish from real samples. It can be written as [7]

$$L_{\text{Adv}} = -\frac{1}{N}\sum_{i=1}^{N} D\left(\hat{Y}_i\right), \tag{3.8}$$

[10]Inference refers to using an already trained network on new data.

where $\hat{Y}_i$ is the generated guess from the generator network, and D is the discriminator network giving a binary classification $D\left(\hat{Y}_i\right) \in [0, 1]$ depending on whether it believes the given image is a real or generated one. Minimizing this loss ensures that the generator network produces samples that have a similar feature map (when extracted by the discriminator network) to real samples, and this process is the basis of GANs.

Weighted loss

In practice it is common to use a weighted sum of different loss functions. An example containing MSE, log-cosh, and VGG loss can be given as

$$L_{\text{Total}} = \lambda_{\text{MSE}} L_{\text{MSE}} + \lambda_{\text{Log-cosh}} L_{\text{Log-cosh}} + \lambda_{\text{VGG}} L_{\text{VGG}}, \tag{3.9}$$

where λ_N is a hyperparameter controlling the weight of L_N.

3.3.3 Backpropagation

Backpropagation is the name given to the process of calculating the needed updates to the parameters of a neural network to reduce the error rate, or loss, of the network. It consists of calculating the partial derivatives of the loss function for each parameter of the network, and then updating them accordingly [56].

The process of passing an input through a network to get some result (i.e. inference) can be seen as forwards propagating through the network. When updating the parameters of the network, a backpropagation algorithm begins by calculating the error of the neurons in the final layer of the network, and then working its way backward layer by layer. For each parameter, its contribution to the total loss of the network is calculated, and the gradient of this contribution is calculated. The backpropagation scheme itself does not update the parameters, but rather it finds what part of the loss corresponds to what parameter. An optimizer is then applied to update the parameters.

3.3.4 Optimizers

To calculate the updates to all parameters, some optimizing method must be used. Two of the most common ones will be briefly introduced here.

Stochastic Gradient Descent

The simplest type of optimizer that is often used in training neural networks is the stochastic gradient descent (SGD). It is an iterative method for optimizing an objective function that has suitable smoothness properties (e.g. differentiability) [57]. It looks at the error in the feature map of the training network (when compared to the labeled ground truth), and calculates an approximation of the gradient needed to update all the weights in the network to reduce the error. Because of the use of mini batches during training of neural networks, the SGD method only looks at a randomly selected subset of the whole training data and it is therefore called a stochastic method. The learning rate of SGD is the step size used when updating the weights based on the calculated gradient.

ADAM

ADAM is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [58]. It can be seen as an extension to SGD. While SGD has one single learning rate, ADAM has one learning rate for each different parameter based on estimates of first and second moments of the gradients.

Chapter 4

Method and Datasets

In this chapter, the specific neural network used to denoise CT images will be introduced, some image comparison metrics will be given alongside a brief discussion of them, the used datasets will be described, and the method used to compile a given dataset into a suitable format for the neural network will be explained.

4.1 TomoGAN

The denoising neural network used in this thesis is called TomoGAN [7]. It is a GAN where the generator is based on the U-net model (see Section 3.2.2) [30], and the input to the GAN is not random noise, but rather noisy images. There are a number of key differences in the generator from the U-net model, namely [7]:

- There are three (instead of four) down and up sampling layers.
- All convolutions have zero-padding to keep image dimensions unchanged during convolutions.
- The generator input is a stack of d adjacent slices where eight 1×1 convolutions are applied.

In the down sampling part of the generator, three sets of two 3×3 convolutional layers with ReLU activation functions extract feature maps. Between each set of convolutional layers there is a 2×2 max pooling layer, for a total down sampling of 8 leading to a feature map of size 1/8 of the original image in either dimension (e.g. images of dimensions 1024×1024 will result in a feature map of dimensions 128×128). The number of channels in a convolutional layer changes throughout the network, beginning with 8 channels, and the feature map after the down sampling process has 128 channels.

The up sampling part of the generator is symmetric to the down sampling part, containing three sets of two 3×3 convolutional layers. The max pooling layers are replaced by bi-linear interpolation layers that instead of reducing the size of the feature map, increase it. Again, all the convolutional kernels have kernels of size 3×3 and use ReLU activation functions. At each of the three sets of convolutional layers, the feature maps from the corresponding set in the down sampling section are concatenated to the feature maps outputted from the previous set of convolutional layers.[1] Finally, a 1×1 convolutional layer with a ReLU activation function followed by a 1×1 convolutional layer with a linear activation function is used to combine all the channels into a final output image.

Because of the three 2×2 max pooling layers, the dimensions of the input images must all be divisible by $2^3 = 8$.

A schematic of the generator of TomoGAN is provided in Figure 4.1.

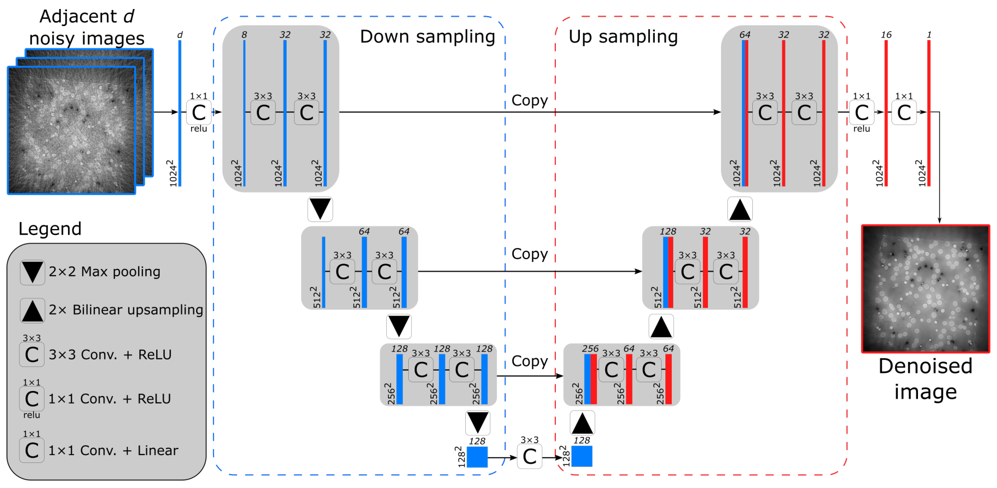


Figure 4.1: A visualization of the structure of TomoGAN. Here, an image of size 1024×1024 has been used as an example. Every bar corresponds to a multi-channel feature map with the number of channels written above and the dimension of the feature map written on the bottom left. All convolutions use zero-padding. The type of activation function used with a given convolution can be seen in the legend. The copy operations transfer the feature map from a down sampling layer and concatenates it to the corresponding up sampling layer. Figure adapted from [7].

The discriminator network used to help train TomoGAN consists of six 2D 3×3 convolutional layers with leaky ReLU activation functions, as well as two fully connected layers.

No changes have been made to the structure of the TomoGAN network in this thesis, however some changes have been made to the loss function used to train the network. The original TomoGAN network, as given by Liu *et al.* [7], was trained using a loss function comprising of an MSE loss, a feature based (VGG) loss, and an adversarial loss, given as

$$L_{\text{Total}} = \lambda_{\text{MSE}} L_{\text{MSE}} + \lambda_{\text{VGG}} L_{\text{VGG}} + \lambda_{\text{Adv}} L_{\text{Adv}}. \tag{4.1}$$

Based on articles citing the log-cosh loss function as a good candidate for image processing [8, 9], a log-cosh loss function (see Section 3.3.2) was added as an

[1]These skip-connections, as they are often called, are what make this structure a U-net style network and not simply an encoder-decoder network [30].

additional term to the total loss, giving the new loss function

$$L_{\text{Total}} = \lambda_{\text{MSE}} L_{\text{MSE}} + \lambda_{\text{Log-cosh}} L_{\text{Log-cosh}} + \lambda_{\text{VGG}} L_{\text{VGG}} + \lambda_{\text{Adv}} L_{\text{Adv}}. \tag{4.2}$$

The weights for the different loss functions were mostly unchanged, with only λ_{MSE} being slightly reduced to account for the fact that $L_{\text{Log-cosh}}$ covers a similar role (as they have similar activations).

The TomoGAN network performs what is known as *image-to-image translation*, meaning it translates one possible representation of an image into another [59, 60].

4.2 Image Comparison Metrics

To quantify the performance of the denoising method, some metrics must be defined. While there are some standards to doing this, not all methods perform equally well.

4.2.1 Mean Squared Error

The most commonly used *full-reference* image quality metric is the mean squared error (MSE), as defined (as a loss function) in Equation (3.4). While it provides some sense of similarity and is easy to calculate and physically understand, a low (i.e. good) MSE does not necessarily correspond to a high degree of visual similarity, and it is therefore not a good metric to compare visual similarities in images [61, 62]. Likewise, the peak signal-to-noise ratio (PSNR), which is another very commonly used metric, does not correspond well to visual similarity [62].

4.2.2 Structural Similarity Index Measure

The structural similarity index measure (SSIM) is a metric to measure similarity between images [63]. It is a full-reference image quality assessment, meaning it compares a reference image to a low-quality image.[2]

The SSIM takes into account three metrics of the image, namely: luminance, contrast, and structure. They are combined, giving the definition of SSIM as [63]

$$\text{SSIM}(x,y) = \frac{\left(2\mu_x\mu_y + C_1\right)\left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}, \tag{4.3}$$

where x and y are the two images to compare, $\mu_{\{x,y\}}$ is the mean pixel value of an image, $\sigma_{\{x,y\}}$ is the standard deviation of the pixel values of an image, σ_{xy} is the covariance of two images, and $C_{\{1,2\}}$ are regularization constants. It is symmetric

[2]Other types of image quality assessments are either *no-reference* where there is no reference image to compare to, or *reduced-reference* where there is some partial information from a reference image to compare to (e.g. a set of extracted features)[63].

(i.e. $\mathrm{SSIM}(x,y) = \mathrm{SSIM}(y,x)$), bounded (i.e. $\mathrm{SSIM}(x,y) \leq 1$), and has a unique maximum (i.e. $\mathrm{SSIM}(x,y) = 1$ if and only if $x = y$) [63]. A more robust analysis of the mathematical properties of the SSIM was performed in [64].

4.3 Datasets

A selection of different datasets have been used to train and test the TomoGAN network. Some of these have been collected from TomoBank [18], one is collected in-house at NTNU, and one is from the ID16B beamline at the European Synchrotron Radiation Facility (ESRF). This section will give a description of each used dataset.

The chosen datasets have been selected to represent several different CT imaging challenges and situations. The borosilicate glass spheres dataset, obtained from the TomoBank database [18], was chosen as an ideal beamline dataset where there is access to a high-quality dataset to properly train TomoGAN. The soda lime glass spheres dataset, collected in-house at NTNU, was chosen to examine the feasibility of using TomoGAN for dynamic CT datasets, and to compare TomoGAN to a reconstruction based denoising technique, namely PICCS. Finally, the Pierre shale dataset, collected at the ID16B beamline of ESRF, was chosen because the heterogeneity of the shale makes it a challenging sample to study, and it represents a real sample in contrast to the idealized cases in the previous two samples. Furthermore, the Pierre shale dataset does not have a corresponding high-quality dataset to train TomoGAN, instead requiring other datasets to be used to train TomoGAN to attempt denoising.

4.3.1 Borosilicate Glass Spheres and TomoBank

TomoBank is an X-ray tomography data bank providing experimental and simulated datasets with the aim to foster collaboration among computational scientists, beamline scientists, and experimentalists [18]. It provides several types of datasets imaging different samples as well as simulated phantoms. Some of these datasets have been used in this thesis.

The tomo_00058 dataset is a dataset imaging borosilicate glass spheres encased in a polypropylene matrix [17]. It contains a 20% concentration of glass spheres. The technical information of the dataset is given in Table 4.1, and an overview of how many projections are used in different subsamplings to simulate artifacting due to undersampling is given in Table 4.2. The subsampling is done with evenly angular spacing.

This dataset will be referred to as the borosilicate glass spheres dataset. An arbitrary slice of this dataset, with and without simulated undersampling artifacts, is provided in Figure 4.2.

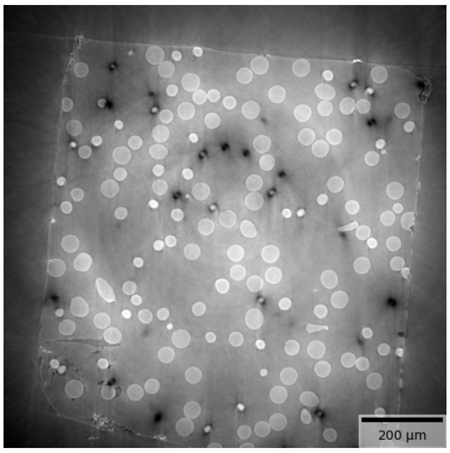
Two datasets of shale from TomoBank were also used, namely tomo_00001 and tomo_00002 [65]. These datasets are two shale samples collected from the North Sea and from the Upper Barnett Formation in Texas. They are both imaged

Table 4.1: Technical information of the borosilicate glass spheres dataset. The dataset was scanned at the Advanced Photon Source (APS) at Argonne National Laboratory, USA. Further details are available from [17].

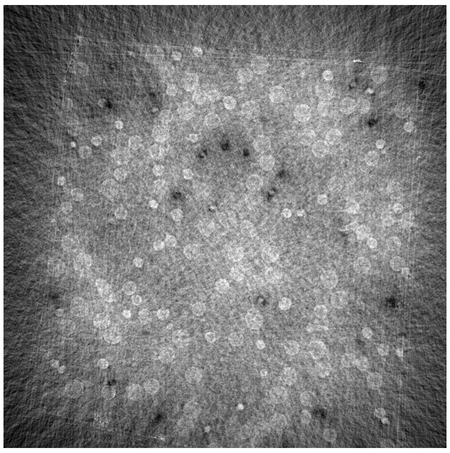
Beamline	APS 2-BM-A
Sample rotation ranges	180 degrees
Number of projections	1500
Energy	27.4 keV
Exposure time	1 ms
Pixel size	0.65 µm
Detector dimension x	2560
Detector dimension y	2160

Table 4.2: Overview of the number of projections for different subsampling factors of the borosilicate glass spheres dataset for simulating undersampling artifacts. The subsampling is uniformly distributed over the entire range of captured projections (i.e. undersampling and not missing wedge, see Section 2.1.1).

Subsampling factor	Projections
1	1500
8	187
16	93
32	46
48	31



(a) High-quality reconstruction (1500 projections)



(b) Undersampled projections (46 projections)

Figure 4.2: Different reconstructions of the borosilicate glass spheres dataset. Both reconstructions are made using FBP reconstruction. The two images are of the same arbitrary slice from the datasets.

at the Advanced Photon Source (APS) of Argonne National Laboratory, USA. These datasets are only used to train a network for denoising of another shale sample, namely the Pierre shale sample (see Section 4.3.3).

All datasets used from TomoBank were reconstructed using FBP from the TomoPy library [66].

4.3.2 Soda Lime Glass Spheres

An in-house captured dataset imaging a sample of soda lime glass spheres in a capillary tube containing a potassium iodide (KI) doped water solution has been used. The soda lime glass spheres have diameters ranging 1250 µm to 1650 µm, the capillary tube has an inner diameter of 2.5 mm and an outer diameter of 4.0 mm, and the potassium iodide doped water has a concentration of 0.5 M KI. The sample was imaged twice: one slow high-quality imaging, and one fast low-quality imaging. The technical information of the two imagings of the in-house dataset is given in Tables 4.3 and 4.4.

Table 4.3: Technical information of the high-quality imaging of the soda lime glass spheres in-house dataset.

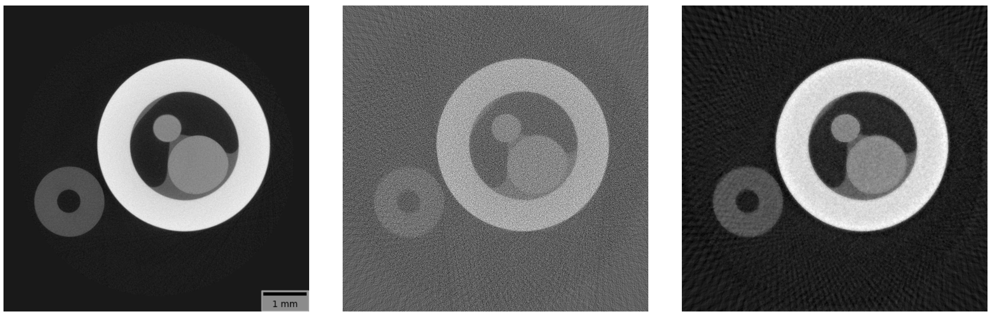
Instrument	Nikon XT H 225 ST
Number of projections	1570
Voltage	170 kV
Current	45 µA
Gain	17 dB
Exposure time	1000 ms
Pixel size	7.06 µm

Table 4.4: Technical information of the fast scan of the soda lime glass spheres in-house dataset.

Instrument	Nikon XT H 225 ST
Number of projections	150
Voltage	170 kV
Current	65 µA
Gain	24 dB
Exposure time	134 ms
Pixel size	7.06 µm

For the sake of simplicity, the two datasets will also be referred to as the in-house high-quality (IHHQ) and the in-house low-quality (IHLQ) datasets.

Because the used CT instrument has a conical beam, the FDK reconstruction algorithm was used to reconstruct these datasets. A PICCS reconstruction of the IHLQ dataset is also included as an alternative denoising method for comparison.



(a) IHHQ. **(b)** IHLQ FDK. **(c)** IHLQ PICCS.

Figure 4.3: Different reconstructions of the IHHQ and IHLQ soda lime glass spheres datasets. The three images are of the same arbitrary slice from the datasets.

A figure showing an arbitrary slice from the IHHQ, IHLQ FDK, and IHLQ PICCS datasets is provided in Figure 4.3.

4.3.3 Pierre Shale

This sample is a piece of Pierre shale from the east of the Rocky Mountains in North America, measured with phase-contrast CT at the beamline ID16B at the European Synchrotron Radiation Facility (ESRF). Technical details of the dataset are provided in Table 4.5, and a slice of the sample is provided in Figure 4.4.

Table 4.5: Technical information of the Pierre shale dataset. The dataset was scanned at the ID16B beamline at the European Synchrotron Radiation Facility (ESRF).

Beamline	ESRF ID16B
Sample rotation ranges	180 degrees
Number of projections	2160
Energy	30 keV
Pixel size	60 nm

4.4 Compiling a Dataset for Training

In this section, an explanation of how to compile a set of images into a suitable dataset for training the TomoGAN network, or to denoise with an already trained network, will be presented.

The process of preparing a dataset for either training or inference (i.e. denoising using an already trained network) consists of a few steps. A flowchart visualizing this process is provided in Figure 4.5. The detailed steps are as follows:

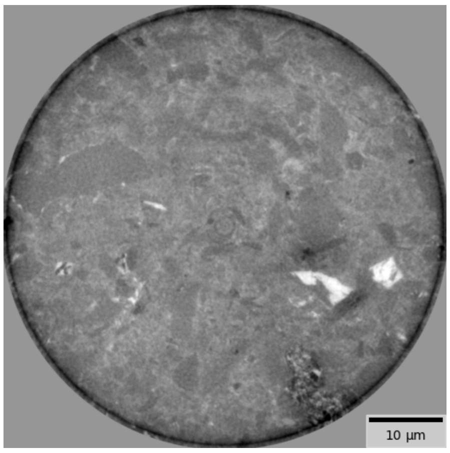


Figure 4.4: View of the Pierre shale dataset. No high-quality reconstruction is available.

1. Begin with a full stack of reconstructed images. If the dataset is meant to be used to train the network, both a set of high-quality (HQ) reconstructions as well as a corresponding set (of the same dataset) of low-quality (LQ) reconstructions are needed. They must be sorted the same way so that image 0 of the HQ dataset corresponds to image 0 of the LQ dataset. If it is a dataset that is to be denoised, only the LQ dataset is needed.
2. The images must then be cropped to remove unnecessary empty space and to help the network pick up on the important parts of the image. For a dataset that has not been properly cropped, the network may struggle to converge to a good solution, yielding poor results. This cropping step is done by visually inspecting the dataset and cropping an appropriate part of the images. If the dimensions of the images are very large, it may be too large to properly fit in memory during training or denoising. The images must then be resized to an appropriate smaller size. The dimensions must also be divisible by 8 (see Section 4.1).
3. Once the images are cropped, they must be normalized and converted to the datatype uint8, as that is best suited for use in the network.[3] This process consists of several steps. For each image in the dataset do the following:
    a. First, find the minimum and maximum pixel values of an image. This can be the absolute minimum and maximum, but ideally it should be a lower and upper percentile of the pixel values to avoid single extreme pixel values causing a loss of dynamic range in the image, as the pixel values will be limited to 256 distinct values.
    b. Once the minimum and maximum values are determined, the image

[3]Using uint8 instead of e.g. float32 reduces the memory footprint drastically, and the hardware used to perform the calculations needed to train a neural network is often optimized for this type of calculations using uint8.

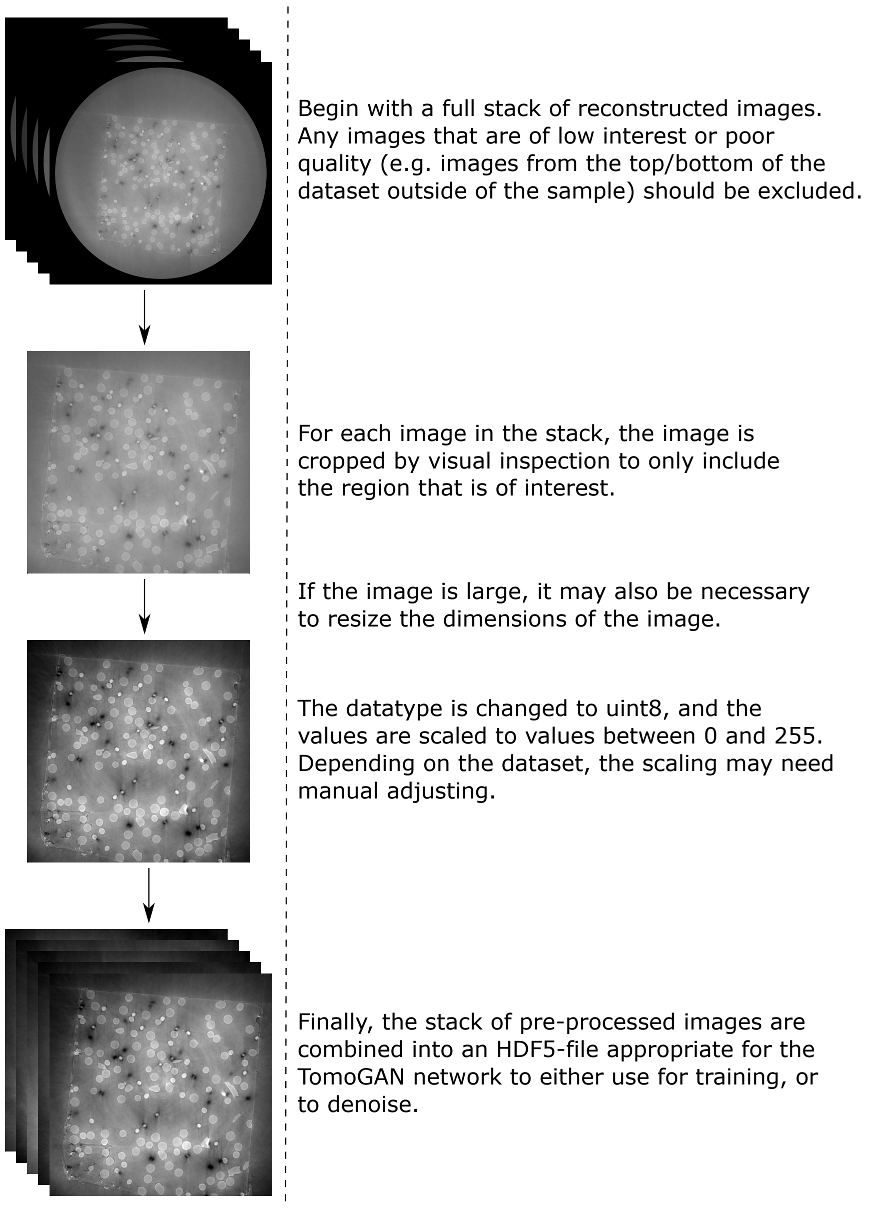
pixel values should be clipped to the previously determined minimum and maximum values. This is necessary when the minimum and maximum values are percentiles to ensure all pixel values are within the new value range set by the percentiles.

c. After clipping the pixel values, all pixel values can be scaled linearly between 0 and 255 according to the formula

$$\hat{x} = 255 \cdot \frac{x - I_{min}}{I_{max} - I_{min}}, \tag{4.4}$$

where $\hat{x}$ is the updated scaled pixel value, x is the old pixel value, and I is the whole image.

4. Once all images are scaled and converted to uint8, they can be filled into an HDF5-file containing one dataset for the HQ images, and if the dataset is to be used to train the network, one dataset for the LQ images. The dimensions of the datasets should be (N, H, W), where N is the number of images, and (H, W) are the dimensions of the images.
5. When the dataset is to be used to train the network, both the HQ and the LQ datasets must be split into two datasets: one for training, and one for testing. About 15% of the total dataset should be set aside for testing. If the depth parameter of TomoGAN is to be used, both the training and test datasets need to be sorted (i.e. adjacent slices are adjacent in the dataset).



Begin with a full stack of reconstructed images. Any images that are of low interest or poor quality (e.g. images from the top/bottom of the dataset outside of the sample) should be excluded.

For each image in the stack, the image is cropped by visual inspection to only include the region that is of interest.

If the image is large, it may also be necessary to resize the dimensions of the image.

The datatype is changed to uint8, and the values are scaled to values between 0 and 255. Depending on the dataset, the scaling may need manual adjusting.

Finally, the stack of pre-processed images are combined into an HDF5-file appropriate for the TomoGAN network to either use for training, or to denoise.

Figure 4.5: Flowchart showing the process of compiling a dataset for training the TomoGAN denoising network. See Section 4.4 for a thorough explanation.

Chapter 5

Results and Discussion

This chapter will present the results of denoising the different datasets, and discuss the quality of the results. For the three different datasets introduced in Section 4.3, different parts of the denoising will be analysed.

For the borosilicate glass spheres dataset, the focus has been on how much noise can be removed, what effect changing the loss function has on the denoising, and what effect the pre-processing steps have on the denoising.

The soda lime glass spheres dataset denoising has focused on comparing TomoGAN denoising to PICCS reconstruction denoising, and how well the denoising captures 3D information.

Finally, the Pierre shale dataset has been used to explore the possibilities of using TomoGAN to denoise when a corresponding high-quality dataset is unavailable.

5.1 Borosilicate Glass Spheres

In this section, denoising of the borosilicate glass spheres dataset (see Section 4.3.1) from TomoBank has been explored. On this dataset, the effect of the depth parameter of TomoGAN has not been explored (see Section 4.1), instead focus has been on how much noise can be removed, and the effects of pre-processing and changing the loss function for training.

5.1.1 Denoising with Different Amounts of Noise

To explore the efficiency of the denoising method, the borosilicate glass spheres dataset was reconstructed with four different levels of projection undersampling. By using subsampling factors of 8, 16, 32, and 48 (see Table 4.2), four datasets with different levels of artifacting were simulated. They are given alongside the HQ reconstruction in Figure 5.1.

The TomoGAN network was trained with a loss function containing MSE, log-cosh, VGG, and adversarial components, a depth of 1 was used, and the network was trained for 100000 iterations with a mini batch size of 16. The resulting denoised images are given in Figure 5.2

It is evident that a subsampling factor of 48, corresponding to merely 31 projections, leads to too much noise to recover any usable denoised image. This is further confirmed by looking at plots of the pixel values along two different lines on the denoised images and comparing them with the noisy and the HQ image, as is provided in Figures 5.3 and 5.4. For the other three levels of undersampling the denoising seems to result in images sufficiently similar to the HQ image, especially for subsampling factors 8 and 16.

Table 5.1 contains SSIM and MSE values for the noisy and denoised images shown in Figures 5.1 and 5.2. We see here that for subsampling factors 8, 16, and 32, the MSE is improved by denoising, however for subsampling factor 48 it is drastically worsened. All denoisings achieve, to varying degrees, SSIM improvements. The fact that the poor denoising of the subsampling factor of 48 achieves an improvement in SSIM from 0.193 to 0.657 indicates that SSIM is not able to sufficiently gauge the image quality in this case.

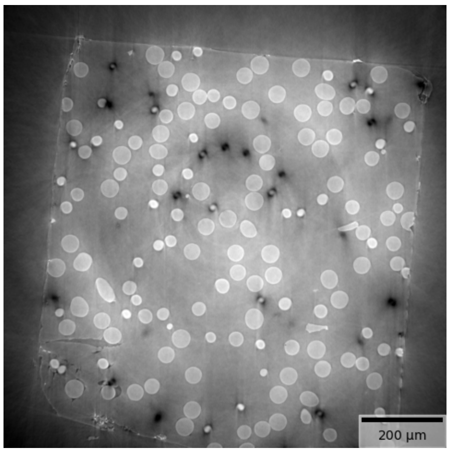
It is worth noting that a subsampling factor of 32 corresponds to 46 projections, which is similar to the 64 projections used in the noisiest simulated dataset in the original article by Liu *et al.* [7]. The achieved SSIM score on a real dataset of 0.789 in this thesis is similar to their results on a simulated phantom.

Table 5.1: Overview of SSIM for different levels of simulated projection undersampling on the borosilicate glass spheres dataset. The projection undersampling was simulated by subsampling the number of projections by a factor as given in the subsampling factor column. All denoising was done with TomoGAN using a loss function containing MSE, log-cosh, VGG, and adversarial loss components, a depth of 1, and the network was trained for 100000 iterations with a mini batch size of 16.

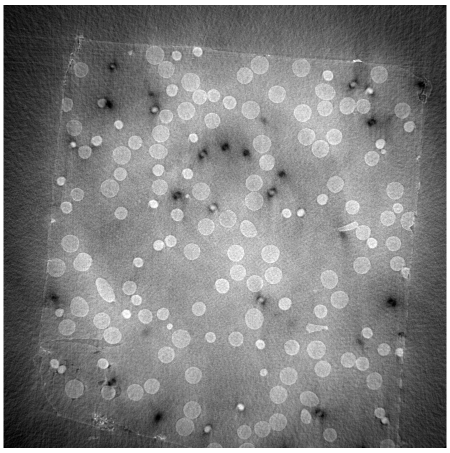
Subsampling factor	Projections	SSIM		MSE	
		Noisy	Denoised	Noisy	Denoised
1	1500	1.0	–	0.0	–
8	187	0.492	0.842	148.3	33.3
16	93	0.335	0.816	348.5	74.9
32	46	0.233	0.789	704.4	210.8
48	31	0.193	0.657	976.6	2362.6

5.1.2 Effect of Pre-processing Images

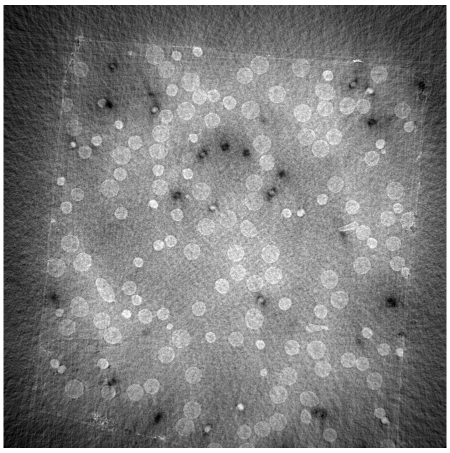
Attempting to denoise the borosilicate glass spheres dataset without cropping the reconstructed images yielded poor results, as can be seen in Figure 5.5. The network does not seem to have managed to converge properly, and the results look to simply be blurring the images. The use of an MSE based loss function may be part of the cause of the blurring, as this loss has been shown to cause blurring in image processing related tasks such as this (see Section 3.3.2).



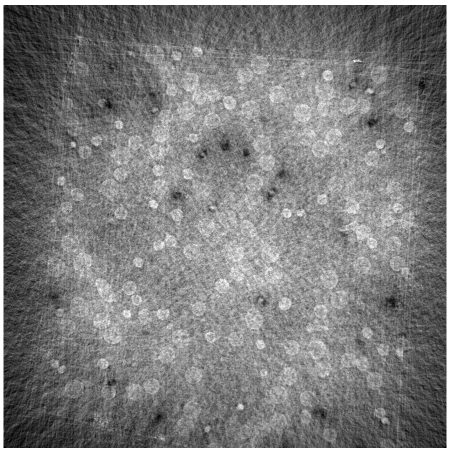
(a) HQ (1500 projections).



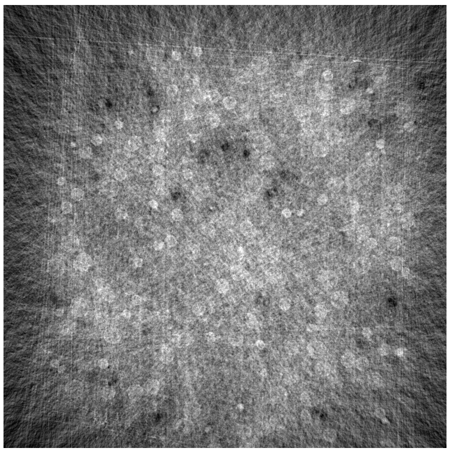
(b) Subsampling factor 8 (187 projections).



(c) Subsampling factor 16 (93 projections).

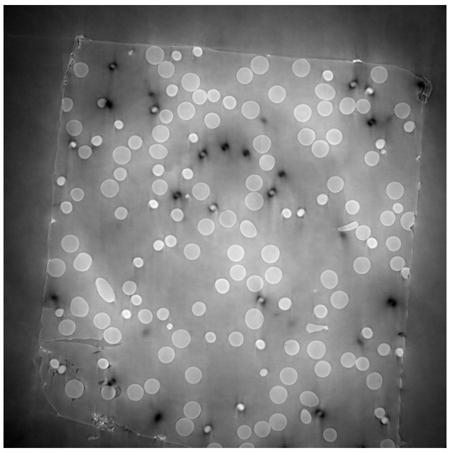


(d) Subsampling factor 32 (46 projections).

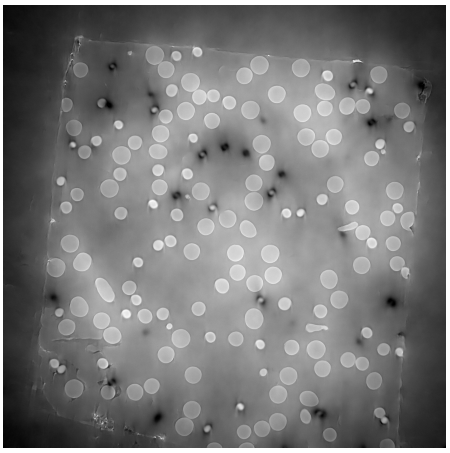


(e) Subsampling factor 48 (31 projections).

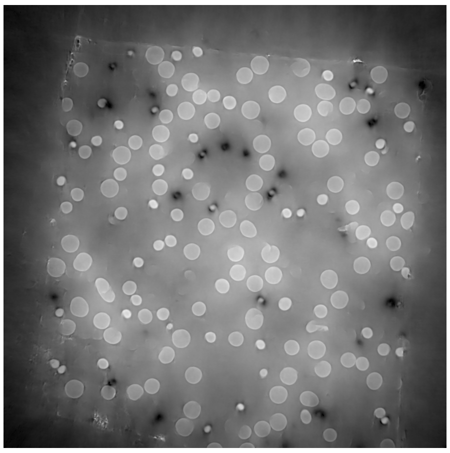
Figure 5.1: Comparison of different levels of projection undersampling on the borosilicate glass spheres dataset by reconstructing the images with the FBP algorithm. The same cross-section is displayed for different reconstructions made by varying the number of projections used.



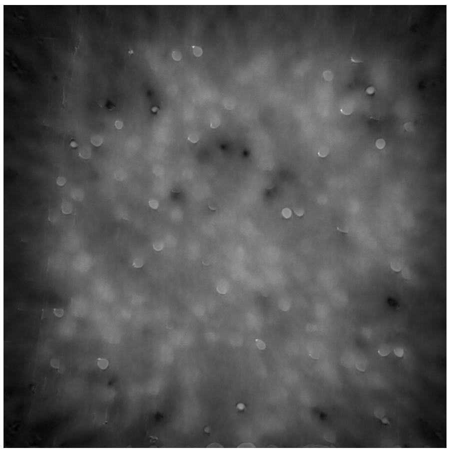
(a) Subsampling factor 8 (187 projections).



(b) Subsampling factor 16 (93 projections).



(c) Subsampling factor 32 (46 projections).



(d) Subsampling factor 48 (31 projections).

Figure 5.2: Comparison of denoising of different levels of projection undersampling on the borosilicate glass spheres dataset, where the corresponding noisy images are given in Figure 5.1. The denoising was done with TomoGAN using a loss function containing MSE, log-cosh, VGG, and adversarial loss components, a depth of 1, and the network was trained for 100000 iterations with a mini batch size of 16.

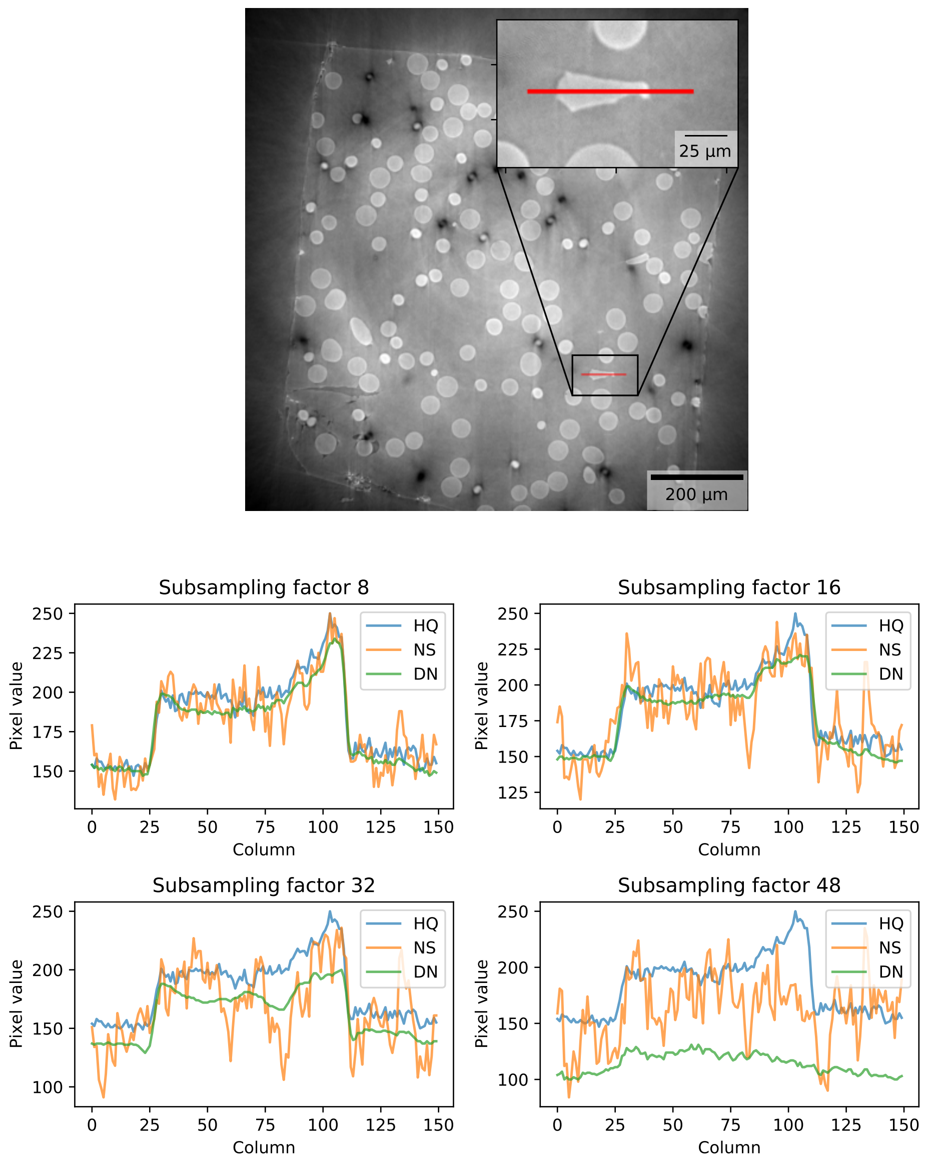
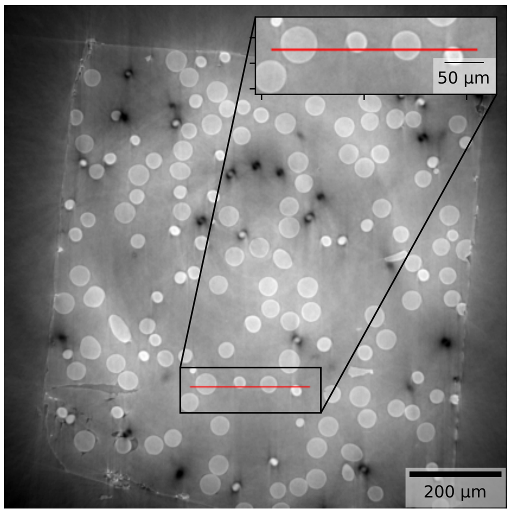


Figure 5.3: The plots show pixel values for 150 pixels on a horizontal line on the borosilicate glass spheres dataset, as shown by the red line on the HQ image above, for denoising of four different levels of projection undersampling. HQ corresponds to the HQ image, NS to the noisy/low-quality image, and DN to the denoised image. The denoising was done with TomoGAN using a loss function containing MSE, log-cosh, VGG, and adversarial loss components, a depth of 1, and the network was trained for 100000 iterations with a mini batch size of 16.



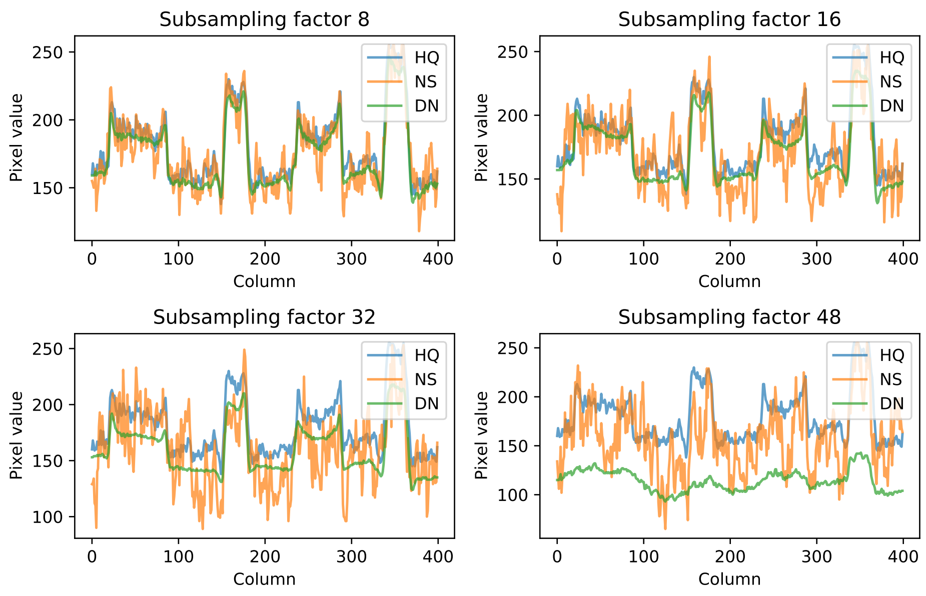


Figure 5.4: The plots show pixel values for 400 pixels on a horizontal line on the borosilicate glass spheres dataset, as shown by the red line on the HQ image above, for denoising of four different levels of projection undersampling. Figure elements are as in Figure 5.3.

When the images are all cropped, with no other changes being made, the network performs better. This is given in Figure 5.6. While there is artifacting from the denoising (especially when looking at shapes that differ from the common shapes in the dataset, such as non-circular shapes), when comparing the HQ and denoised images it is evident that a majority of the features have been preserved with the noise being drastically reduced.

Comparing the non-cropped denoising with a denoising only run with a few iterations, not reaching full convergence, as seen in Figure 5.7, the results are similar, indicating that the non-cropped denoising may have struggled to converge properly.

Note that the images in Figures 5.5 and 5.6 have different intensity scales because of the scaling step of the preprocessing (see Section 4.4).

This denoising was performed with TomoGAN using a loss function containing MSE, log-cosh, VGG, and adversarial loss components, a depth of 1, and the network was trained for 100000 iterations with a mini batch size of 16.

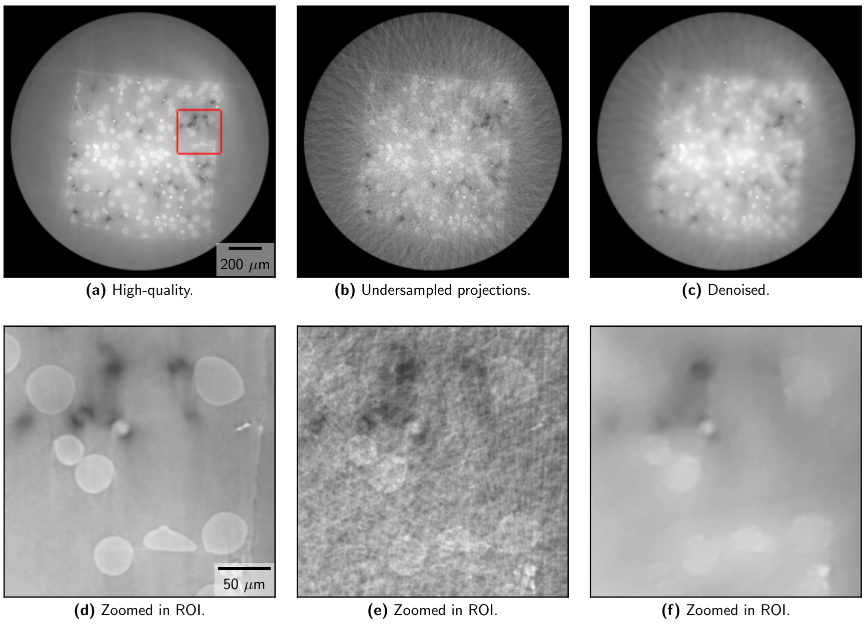
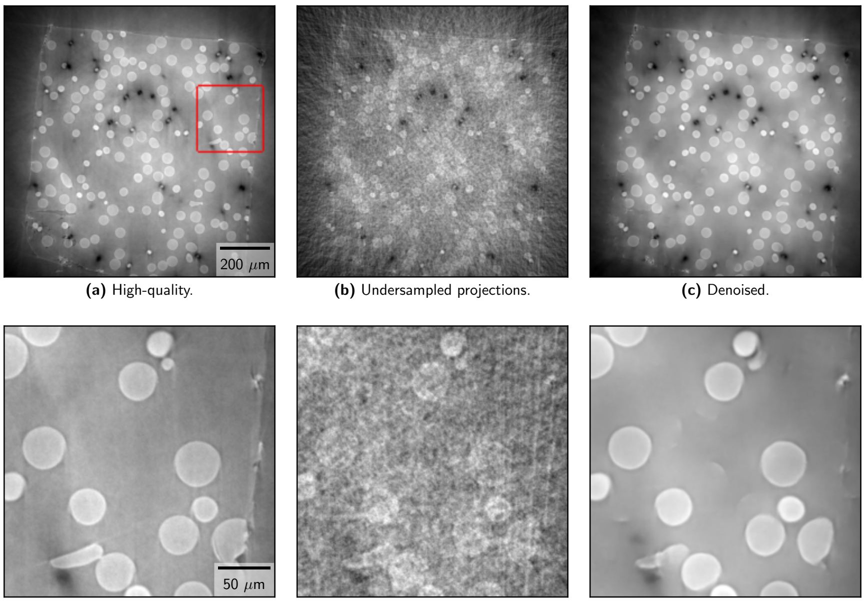


Figure 5.5: Denoising of non-cropped borosilicate glass spheres dataset. Images d), e), and f) are zoomed in ROIs of images a), b), and c) respectively. The ROI is marked in a).

5.1.3 Hyperparameter and Loss Function Changes

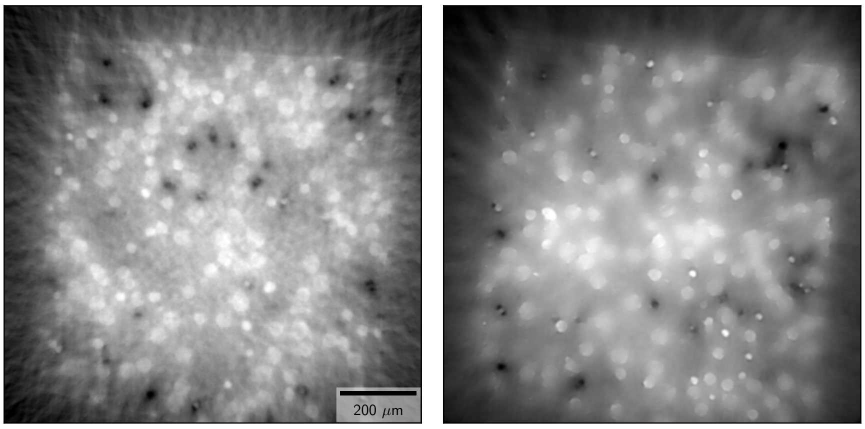
How the change in the loss function affected the denoising operation was explored for the borosilicate glass spheres dataset. A log-cosh term (see Section 3.3.2) was added to the loss function, and the resulting denoised images obtained using different loss functions were compared. All other hyperparameters were kept con-



(a) High-quality. **(b)** Undersampled projections. **(c)** Denoised.

(d) Zoomed in ROI. **(e)** Zoomed in ROI. **(f)** Zoomed in ROI.

Figure 5.6: Denoising of cropped borosilicate glass spheres dataset. Images d), e), and f) are zoomed in ROIs of images a), b), and c) respectively. The ROI is marked in a).



(a) Cropped denoising. **(b)** Non-cropped denoising.

Figure 5.7: Non-cropped image denoising compared to non-converged cropped image denoising on the borosilicate glass spheres dataset. The cropped denoising is after merely 500 iterations of the same denoising as in Figure 5.6. The non-cropped denoising has been cropped after denoising for easier comparison. The two images are of different slices of the same dataset.

stant, only the composition of the loss function was changed (i.e. adding log-cosh loss and reducing λ_{MSE}). The resulting two denoisings for an arbitrary slice is given in Figure 5.8.

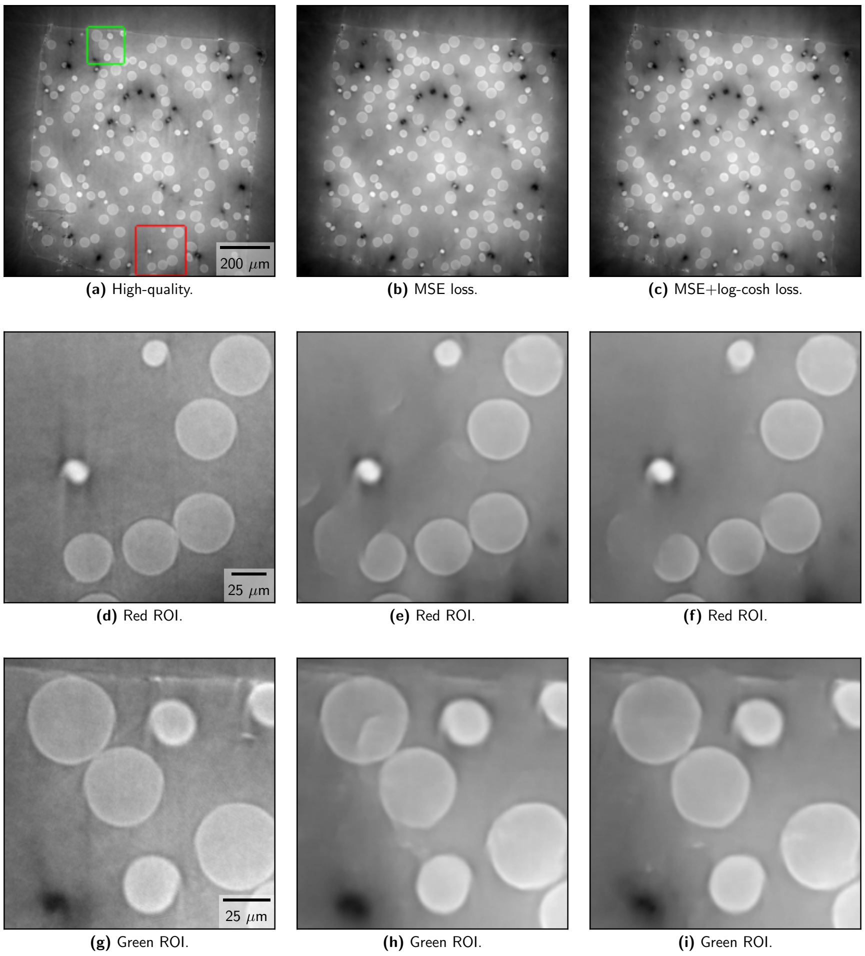


Figure 5.8: Effect on denoising of changing loss function. Both denoisings are of the borosilicate glass spheres dataset with a subsampling factor of 32. Two ROIs have been marked in **(a)**, and can be seen in **(d)**-**(i)**.

While there is little visual difference in the two denoisings, the denoising containing a log-cosh based loss achieves a slight improvement in the SSIM score compared to the one without (0.789 vs. 0.788). There is also no measurable difference in the training time of the network when incorporating another loss function. Because of this minor improvement with no discernible drawback, the loss function containing a log-cosh component is chosen as the default loss function for denoising in this thesis. It is possible that a larger improvement may be achievable by further tuning the weights of the different loss functions, as this has not been thoroughly explored.

5.1.4 Loss Function Evolution

To try and understand how the TomoGAN network is performing during training, the evolution of the loss function and its components has been plotted. This is given in Figure 5.9. We see that the main reduction in the loss of the network happens during early stages of training, especially the first few iterations. In the first 500 iterations the loss changes from ≈ 7000 to ≈ 100, however a denoising after 500 iterations does not perform well (as is given in Figure 5.7a). At 100000 iterations, the loss is ≈ 70. In the early iterations the denoising solves for low spatial-frequency features which result in large decreases in the loss function. In later iterations high frequency information is captured, which has a smaller impact on the loss as these are finer details.

It is worth noting that the exact values of the loss are of no importance, as the scale can be altered by changing the weighting parameters of the loss function. It is the relative contributions of the different losses as well as how they change that is relevant.

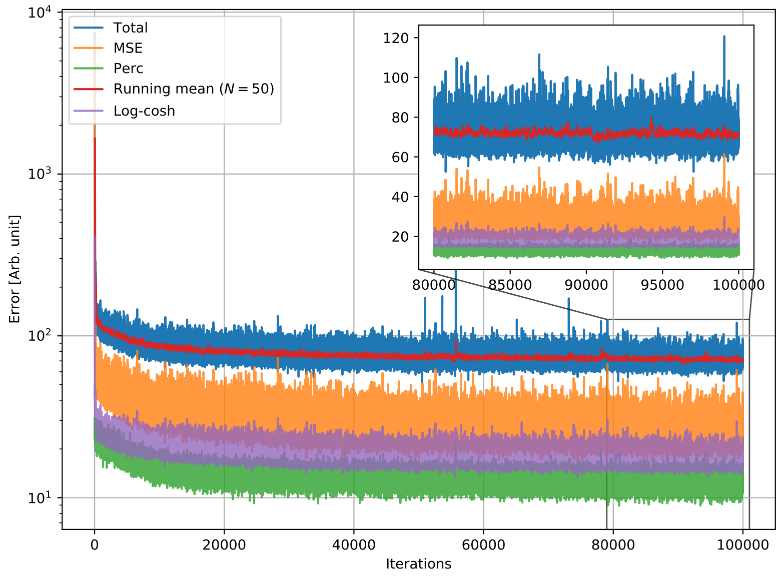
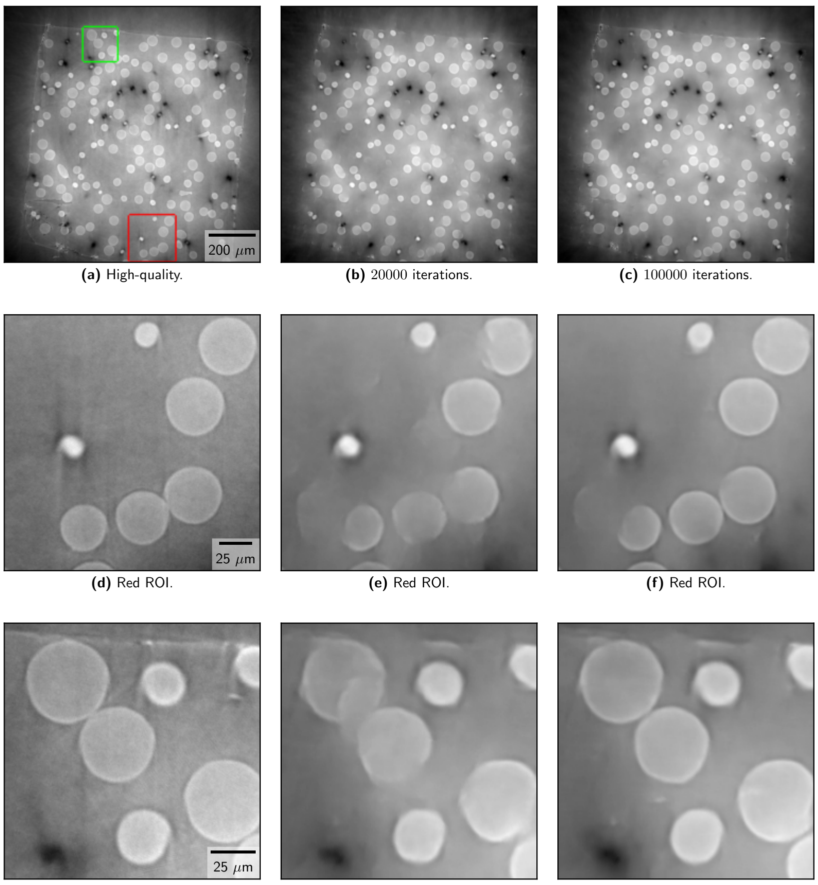


Figure 5.9: Plot showing how the loss function evolves during 100000 iterations of training on the borosilicate glass spheres dataset with a subsampling factor of 32. The adversarial loss has been omitted from the plot. The inset axis shows the final 20000 iterations. Note that the main axis is logarithmic, with the inset axis being linear. The unit of the loss is arbitrary. The running mean is the mean of the previous 50 values.

If denoising performance could be gauged based solely on Figure 5.9, the denoised image after 20000 iterations should be very similar to after 100000 iterations. Looking at Figure 5.10 it is evident that this is not correct. The denoising after 20000 iterations achieves an SSIM of 0.778 compared to 0.789 after 100000 iterations. After 20000 iterations, the spheres in the borosilicate glass spheres dataset are less sharp and more artifacts have been introduced. This is consistent with high frequency information being captured at later iterations. How the SSIM and MSE of an arbitrary slice of the dataset evolve through training is given in Figure 5.11.



(a) High-quality. **(b)** 20000 iterations. **(c)** 100000 iterations.

(d) Red ROI. **(e)** Red ROI. **(f)** Red ROI.

(g) Green ROI. **(h)** Green ROI. **(i)** Green ROI.

Figure 5.10: Effect on denoising of number of iterations. Both denoisings are of the borosilicate glass spheres dataset with a subsampling factor of 32. Two ROIs have been marked in **(a)**, and can be seen in **(d)**-**(i)**. The scale bar for **(a)** applies to **(b)** and **(c)**, the scale bar for **(d)** applies to **(e)** and **(f)**, and the scale bar for **(g)** applies to **(h)** and **(i)**.

5.2 Soda Lime Glass Spheres

The in-house captured dataset IHLQ of soda lime glass spheres (see Section 4.3.2) has been denoised using TomoGAN. The network has been trained to try and map the IHLQ dataset to the IHHQ dataset. As a comparison, two different reconstruction methods for the IHLQ dataset have been included: the FDK direct reconstruction method and the PICCS iterative reconstruction method.

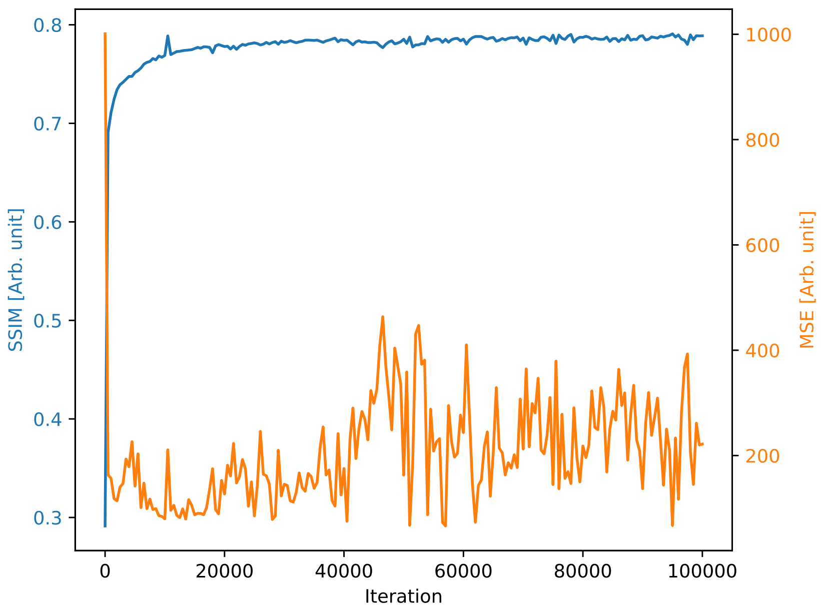


Figure 5.11: Plot showing how the SSIM and MSE of an arbitrary slice of the borosilicate glass spheres dataset evolve during 100000 iterations of training with a subsampling factor of 32.

This section focuses on comparing the TomoGAN denoising to using PICCS for denoising during reconstruction, specifically for dynamic CT datasets, and also looks at how well 3D information is denoised.

5.2.1 TomoGAN Compared to PICCS

To compare the TomoGAN denoising to the IHLQ PICCS reconstruction, the pixel values along a line on an arbitrary slice of the soda lime glass spheres dataset have been examined. A plot is provided in Figure 5.12. We see here that the IHHQ and IHLQ PICCS based reconstructions, as well as the denoised IHLQ dataset, are similar. The IHLQ FDK reconstruction is noisy, and it is hard to clearly discern boundaries.

This is further confirmed by looking at the histograms of the four different reconstructions and denoisings, provided in Figure 5.13. Whereas the IHLQ PICCS reconstruction is closely related to the IHHQ reconstruction, it is evident that the IHLQ FDK reconstruction is a gaussian curve, indicating it is primarily noise. No pixel value peaks can be discerned. The histogram of the denoised IHLQ dataset has peaks similarly located as the IHHQ dataset, however the peaks are narrower and the dynamic range of the denoised image is increased.[1] The number of peaks in the pixel values for IHHQ and IHLQ PICCS is the same as for the denoised IHLQ.

The TomoGAN denoising of the IHLQ FDK reconstruction has captured the

[1]Dynamic range refers the ratio between the brightest and darkest parts of the image.

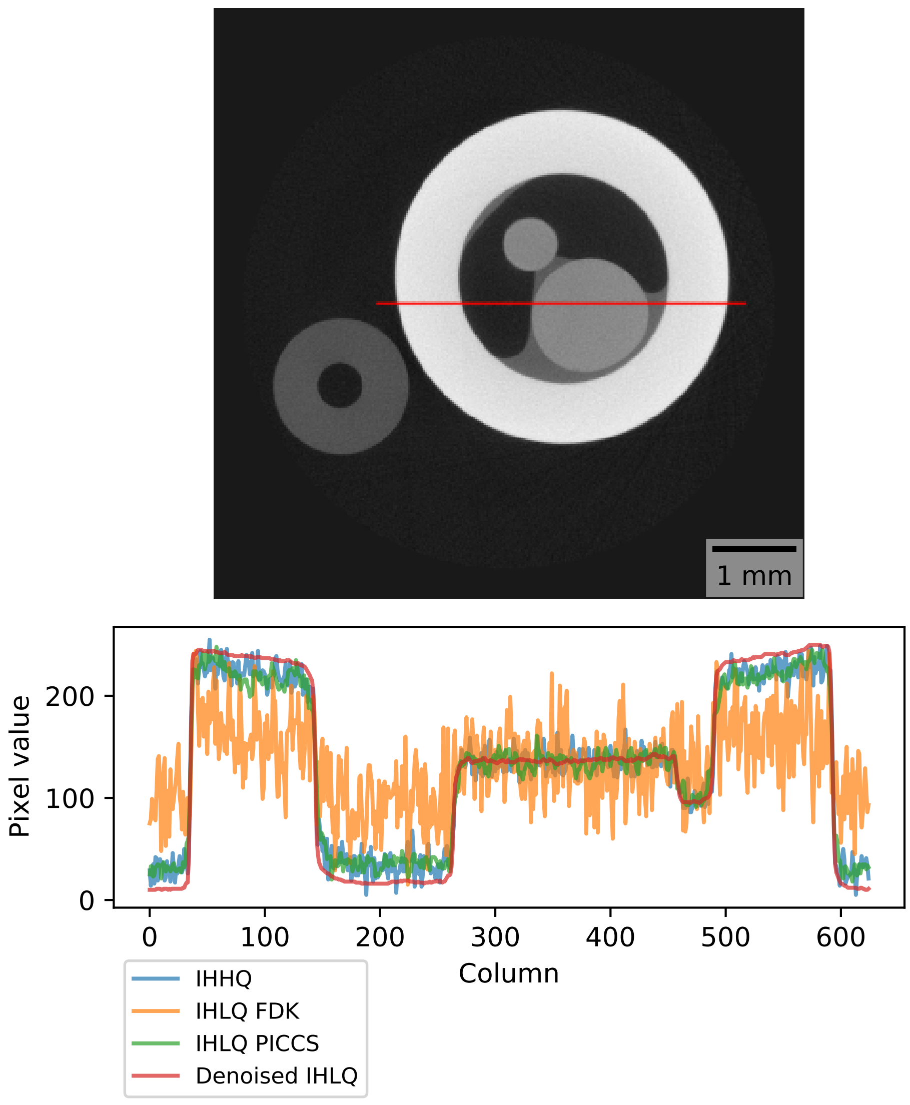


Figure 5.12: The plot shows pixel values for 625 pixels on a horizontal line, as indicated by the red line on the HQ image above, on the IHHQ and IHLQ dataset, both noisy and denoised. The denoised values are from denoising the IHLQ FDK reconstruction with TomoGAN.

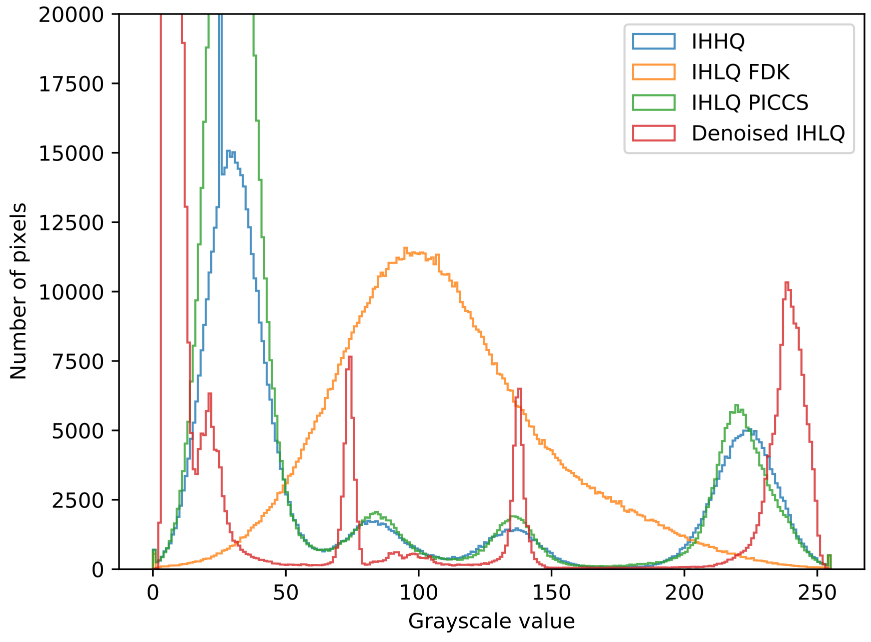


Figure 5.13: Histograms of an arbitrary slice from the IHHQ and IHLQ FDK data-set, both noisy and denoised. The denoised values are from denoising the IHLQ FDK reconstruction with TomoGAN. Note that the ordinate has been cropped to a max value of 20000.

boundaries between different materials in the dataset, as can be seen in Figure 5.12. All boundaries correspond well to the IHHQ reconstruction. The pixel values within a material are more consistent in the denoised reconstruction than the IHHQ reconstruction, and we see that the pixel values utilize the available range of values more (i.e. the lowest pixel values are lower, and the highest pixel values are higher). The dynamic range is increased.

5.2.2 Effect of Depth Parameter on 3D Denoising

For this dataset, the 3D stability of the denoising method has been explored. Because the TomoGAN network uses 2D convolutions (see Section 4.1), it is primarily a 2D denoising method. The network allows for 3D information to be included in the denoising by adjusting a depth parameter, which looks at adjacent slices and performs a 1×1 convolution across them, however by inspecting the structure of the network (see Figure 4.1) it is evident that this 3D information is only used at early stages of the denoising. Only the initial 1×1 convolutional layer, before the down sampling part of the network, looks at the depth.

The result of the limited use of 3D information is that the network has a plane that is the primary focus of the denoising. By inspecting Figure 5.14 it is evident that the denoising is done on the axial plane. For a denoising with a depth of 1, it is very clear that there is a lack of consistency between axial planes. Looking at the yellow ROI there are banding artifacts introduced between the axial layers, interaxial artifacting has been introduced. The same denoising with a depth of 7 still contains some of these interaxial banding artifacts, however the introduction of some 3D knowledge reduces the artifacting. When instead looking at the blue ROI, which is on the axial plane, there are no banding artifacts. By visual inspection of Figure 5.14, we see that the interaxial banding artifacting is reduced when the depth is increased. The specific depth that yields the best results is dependent on dataset characteristics, such as feature resolution [7].

In the green ROI in Figure 5.14, we see details that were indistinguishable in the IHLQ FDK reconstruction, and not properly visible in the IHLQ PICCS reconstruction, are being captured by TomoGAN. These details are clearly visible when only a depth of 1 is used, however they are highly prone to interaxial banding artifacts between layers, and increasing the depth helps reduce this artifacting.

5.3 Pierre Shale

The Pierre shale dataset (see Section 4.3.3) does not have a corresponding high-quality dataset available. It was therefore denoised using TomoGAN trained on different datasets.

This section looks at the performance of TomoGAN when a corresponding high-quality dataset is not available.

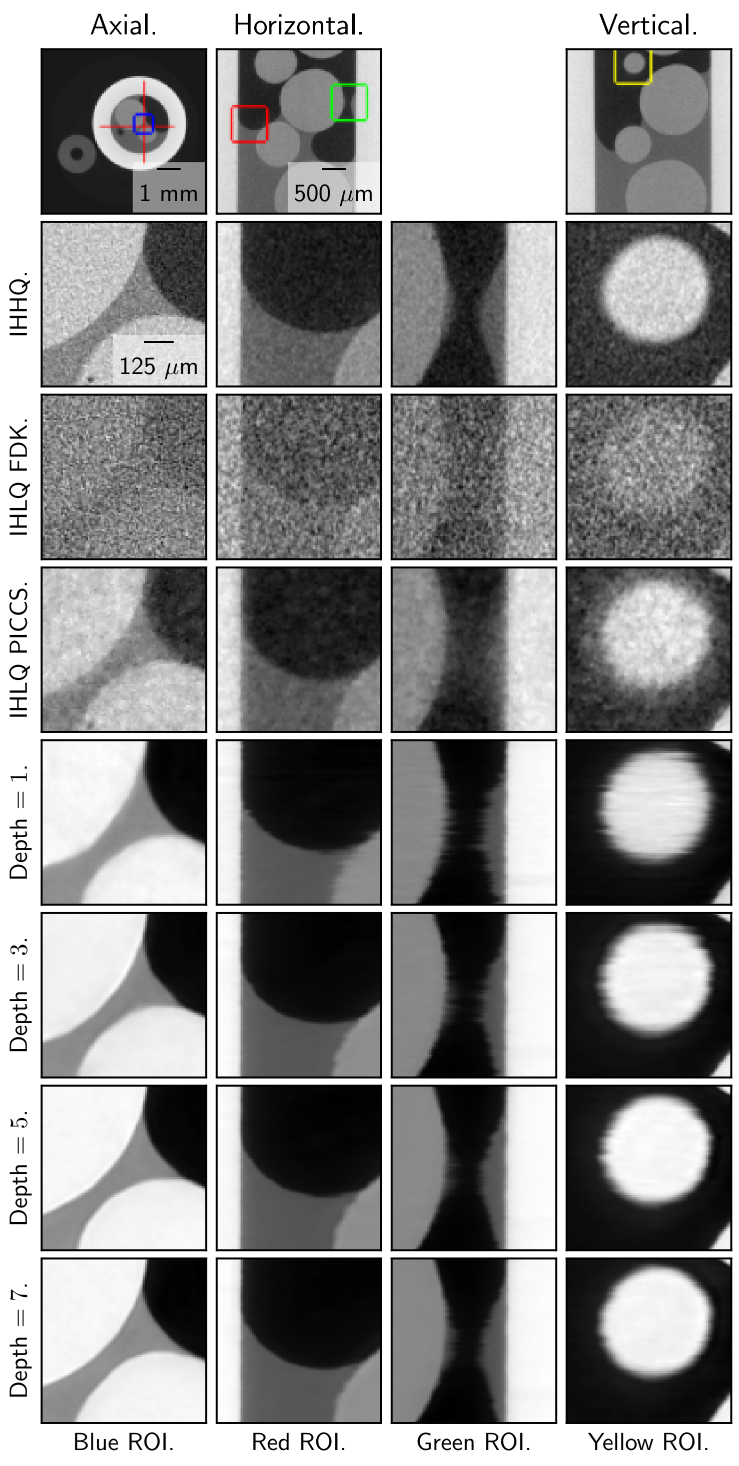


Figure 5.14: Different reconstructions and denoisings of the IHHQ and IHLQ datasets. The top three images are full views of the IHHQ dataset. The two horizontal and vertical images correspond to the horizontal and veritcal red lines in the axial image, and give a view of 3D slices perpendicular to the axial slice in the sample. Four ROIs have been marked in the top figures and have been zoomed in for the different reconstructions and denoisings. All zoomed in figures have the same scale bar.

5.3.1 Denoising Without High-quality Dataset

Because a high-quality imaging of the same dataset is unavailable, four different datasets were used to train TomoGAN to attempt to denoise the Pierre shale dataset. An overview of the datasets used to train for denoising is provided in Table 5.2, and the corresponding denoisings for an arbitrary slice of the dataset are provided in Figure 5.15.

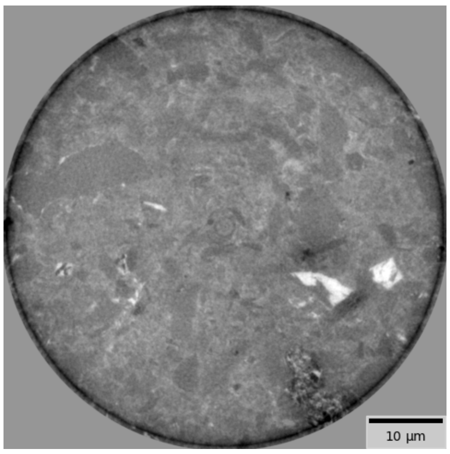
Denoising A is performed with a network not trained on a shale dataset, and denoisings B-D are performed with a network trained on shale denoising, however for different samples than the Pierre shale sample.

Table 5.2: Pierre shale denoising details. tomo_00058 is the borosilicate glass spheres dataset, and tomo_00001 and tomo_00002 are two shale datasets from TomoBank (see Section 4.3.1). The altered scaling in the pre-processing (see Section 4.4) of denoising D corresponds to changing the maximum and minimum pixel values used to crop the pixel values in an image.

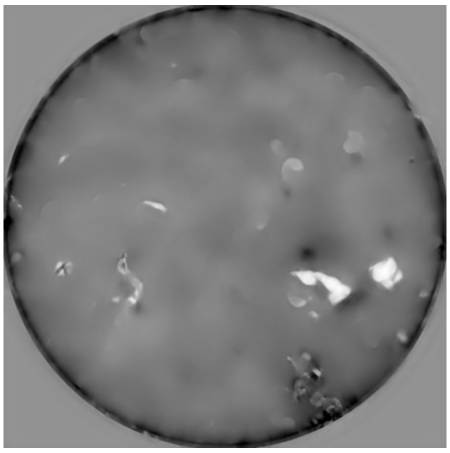
Denoising name	Training dataset	Pre-processing
Denoising A	tomo_00058	Standard
Denoising B	tomo_00001	Standard
Denoising C	tomo_00002	Standard
Denoising D	tomo_00002	Altered scaling

From Figure 5.15 we see that none of the four different denoisings perform well, with denoising A (corresponding to the network trained for the borosilicate glass spheres dataset) performing the worst. All denoisings cause blurring of the images. This is further confirmed by looking at the histograms of the images, which is provided in Figure 5.16. All histograms contain a large spike corresponding to the gray area around the sample, as well as a gaussian curve indicating noise or poor segmentation. No clear spikes in the pixel values can be distinguished for any of the denoisings, nor the original image.

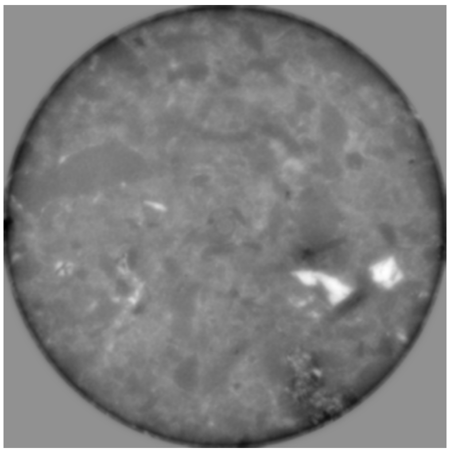
Based on these observations, it seems likely that denoising requires finding a similar training dataset. Based on visual inspection of Figure 5.15, the denoising using the network trained on the borosilicate glass spheres dataset performs poorly, and the other three denoisings using shale datasets all achieve better results, though still not of sufficient quality for further analysis. If a training dataset more similar to the Pierre shale dataset were available, the denoising results may be improved significantly. Changing the pre-processing steps may also further improve the denoising results, as indicated by visual inspection of Figures 5.15d and 5.15e, where the latter causes less blurring even though both are trained on the same dataset with only a change in the pre-processing. The TomoGAN network has been reported to work for denoising shale datasets by Liu *et al.* [7].



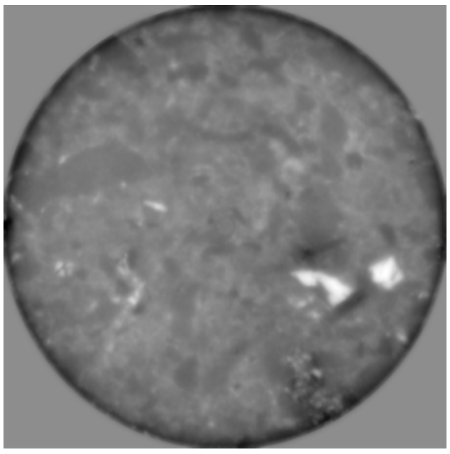
(a) Original image.



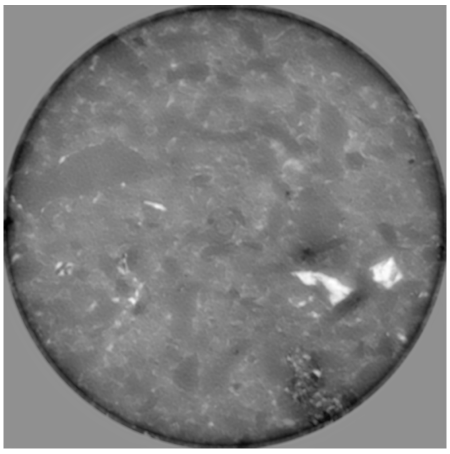
(b) Denoising A.



(c) Denoising B.



(d) Denoising C.



(e) Denoising D.

Figure 5.15: Attempted Pierre shale denoising without a corresponding high-quality dataset. The subfigure captions refer to different trainings of TomoGAN (see Table 5.2).

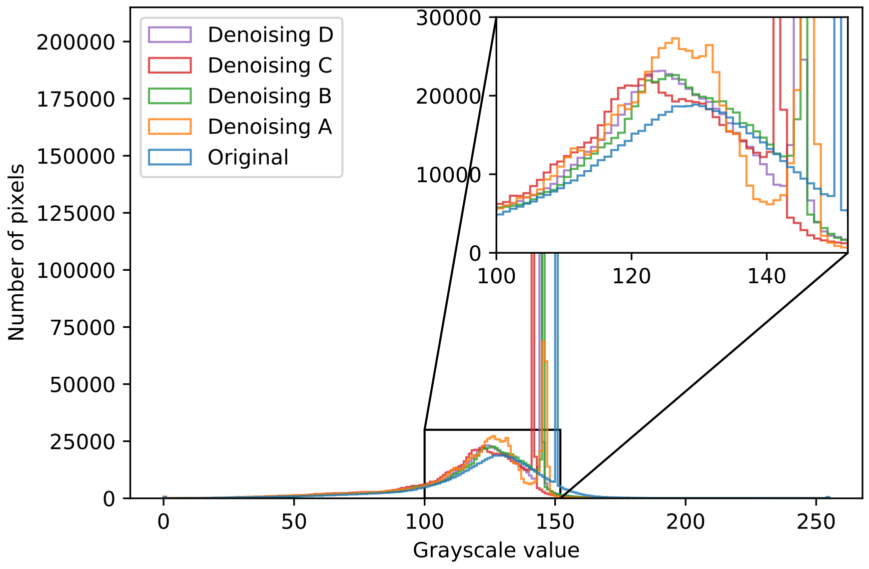


Figure 5.16: Histogram of Pierre shale denoisings. The inset axis contains a zoomed in plot with the ordinate cropped to 30000. See Table 5.2 for details on the labels in the legend.

Chapter 6

Conclusion

In this thesis, CT images have been denoised using the TomoGAN denoising neural network. TomoGAN is able to achieve vast improvements in image quality for cases where a corresponding high-quality dataset is available to train the network on.

A change to the loss function used to train TomoGAN, namely including a log-cosh loss term, was proposed and tested. It yielded minor improvements to the achieved SSIM score for denoising without introducing any discernible drawbacks.

The 3D denoising capabilities of TomoGAN have been explored. The use of the depth parameter of TomoGAN allows the denoising to utilize 3D spatial information when denoising, reducing the interaxial artifacting introduced by denoising single axial slices of a 3D object. Denoising of a dynamic CT dataset imaging soda lime glass spheres achieves comparable image quality to PICCS based reconstruction.

The method is highly dependent on having access to good training data. For denoising where there is no available dataset to train the network, the method is unable to produce usable denoisings. This method is therefore not suited for general-purpose denoising of arbitrary datasets.

Furthermore, the network has been shown to require the training data to be pre-processed to a suitable format in order to achieve usable results.

6.1 Further Work

When 3D datasets are denoised, the TomoGAN denoising neural network is only able to capture 3D information through the use of the depth parameter. The transformation from noisy to noiseless images itself is a 2D transformation (through the use of 2D convolutions). Implementing the same network with 3D convolutions to truly be a 3D denoising method may yield vast improvements to the results [39]. CT imaging is a 3D technique, and the denoising method should utilize that fact.

Furthermore, altering the structure of the network has not been explored in this thesis. The field of GANs is in rapid development [45], and altering the structure of the generator in TomoGAN to utilize new discoveries that arise may further improve results.

Image augmentation (e.g. rotation, zoom, flips) may be used to increase the size of the training dataset in situations where a limited training dataset is available, however this will still require access to a suitable training dataset for a given noisy dataset. Unfortunately, this is a limitation of this method.

There is no inherent feature of TomoGAN currently specializing it to be used for CT images. Some other denoising techniques use the Radon transform to include the reconstruction process itself in the denoising [6, 44]. Altering the TomoGAN method to utilize the full extent of the information available from CT imaging (e.g. a sinogram-based loss) may yield further improvements.

Bibliography

[1] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. IEEE Press, 1998, ISBN: O-87942-1 98-3.

[2] M. I. Jordan and T. M. Mitchell, 'Machine learning: Trends, perspectives, and prospects,' *Science*, vol. 349, no. 6245, pp. 255–260, 2015. DOI: `10.1126/science.aaa8415`.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, 'Generative adversarial networks,' 2014. arXiv: `1406.2661`.

[4] J. Bao, D. Chen, F. Wen, H. Li and G. Hua, 'CVAE-GAN: Fine-grained image generation through asymmetric training,' in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[5] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, 'Photo-realistic single image super-resolution using a generative adversarial network,' in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. arXiv: `1609.04802`.

[6] X. Yang, M. Kahnt, D. Brückner, A. Schropp, Y. Fam, J. Becher, J.-D. Grunwaldt, T. L. Sheppard and C. G. Schroer, 'Tomographic reconstruction with a generative adversarial network,' *Journal of Synchrotron Radiation*, vol. 27, no. 2, pp. 486–493, Mar. 2020. DOI: `10.1107/S1600577520000831`.

[7] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo and I. Foster, 'TomoGAN: Low-dose synchrotron x-ray tomography with generative adversarial networks: Discussion,' *Journal of the Optical Society of America A*, vol. 37, no. 3, pp. 422–434, 2020. DOI: `10.1364/JOSAA.375595`.

[8] P. Chen, G. Chen and S. Zhang, 'Log hyperbolic cosine loss improves variational auto-encoder,' 2018. [Online]. Available: `https://openreview.net/forum?id=rkglvsC9Ym`.

[9] H. Zhao, O. Gallo, I. Frosio and J. Kautz, 'Loss functions for image restoration with neural networks,' *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017. DOI: `10.1109/TCI.2016.2644865`.

[10] M. Paziresh, A. M. Kingston, S. J. Latham, W. K. Fullagar and G. M. Myers, 'Tomography of atomic number and density of materials using dual-energy imaging and the Alvarez and Macovski attenuation model,' *Journal of Applied Physics*, vol. 119, no. 21, p. 214 901, 2016. DOI: 10.1063/1.4950807.

[11] R. Ning, X. Tang and D. Conover, 'X-ray scatter correction algorithm for cone beam CT imaging,' *Medical Physics*, vol. 31, no. 5, pp. 1195–1202, 2004. DOI: 10.1118/1.1711475.

[12] F. E. Boas, D. Fleischmann *et al.*, 'CT artifacts: Causes and reduction techniques,' *Imaging Med*, vol. 4, no. 2, pp. 229–240, 2012.

[13] B. R. Whiting, P. Massoumzadeh, O. A. Earl, J. A. O'Sullivan, D. L. Snyder and J. F. Williamson, 'Properties of preprocessed sinogram data in x-ray computed tomography,' *Medical Physics*, vol. 33, no. 9, pp. 3290–3303, Aug. 2006. DOI: 10.1118/1.2230762.

[14] D. J. Brenner and E. J. Hall, 'Computed tomography — an increasing source of radiation exposure,' *New England Journal of Medicine*, vol. 357, no. 22, pp. 2277–2284, 2007. DOI: 10.1056/NEJMra072149.

[15] 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, L. Parker and A. Berrington de González, '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. DOI: 10.1016/S0140-6736(12)60815-0.

[16] A. Kupsch, A. Lange, M. P. Hentschel, S. Lück, V. Schmidt, R. Grothausmann, A. Hilger and I. Manke, 'Missing wedge computed tomography by iterative algorithm DIRECTT,' *Journal of Microscopy*, vol. 261, no. 1, pp. 36–45, 2016. DOI: 10.1111/jmi.12313.

[17] S. Singh, T. Stannard, S. Singh, A. Singaravelu, X. Xiao and N. Chawla, *Varied volume fractions of borosilicate glass spheres with diameter gaussian distributed from 38-45 micronsen cased in a polypropylene matrix*, 2017. DOI: 10.17038/XSD/1373576.

[18] F. D. Carlo, D. Gürsoy, D. J. Ching, K. J. Batenburg, W. Ludwig, L. Mancini, F. Marone, R. Mokso, D. M. Pelt, J. Sijbers and M. Rivers, 'TomoBank: A tomographic data repository for computational x-ray science,' *Measurement Science and Technology*, vol. 29, no. 3, p. 034 004, Feb. 2018. DOI: 10.1088/1361-6501/aa9c19.

[19] L. A. Shepp and B. F. Logan, 'The Fourier reconstruction of a head section,' *IEEE Transactions on Nuclear Science*, vol. 21, no. 3, pp. 21–43, 1974. DOI: 10.1109/TNS.1974.6499235.

[20] J. Radon, 'On the determination of functions from their integral values along certain manifolds,' *IEEE Transactions on Medical Imaging*, vol. 5, no. 4, pp. 170–176, 1986. DOI: 10.1109/TMI.1986.4307775.

[21] D. M. Pelt, K. J. Batenburg and J. A. Sethian, 'Improving tomographic reconstruction from limited data using mixed-scale dense convolutional neural networks,' *Journal of Imaging*, vol. 4, no. 11, 2018, ISSN: 2313-433X. DOI: `10.3390/jimaging4110128`.

[22] S. I. Kabanikhin, 'Definitions and examples of inverse and ill-posed problems,' vol. 16, no. 4, pp. 317–357, 2008. DOI: `10.1515/JIIP.2008.019`.

[23] T. Farquhar, A. Chatziioannou, G. Chinn, M. Dahlbom and E. Hoffman, 'An investigation of filter choice for filtered back-projection reconstruction in pet,' *IEEE Transactions on Nuclear Science*, vol. 45, no. 3, pp. 1133–1137, 1998. DOI: `10.1109/23.681991`.

[24] D. M. Pelt and K. J. Batenburg, 'Fast tomographic reconstruction from limited data using artificial neural networks,' *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5238–5251, 2013. DOI: `10.1109/TIP.2013.2283142`.

[25] L. A. Feldkamp, L. C. Davis and J. W. Kress, 'Practical cone-beam algorithm,' *J. Opt. Soc. Am. A*, vol. 1, no. 6, pp. 612–619, Jun. 1984. DOI: `10.1364/JOSAA.1.000612`.

[26] M. J. Willemink, A. M. Schilham, T. Leiner, P. T. M. Willem, P. A. de Jong and R. P. Budde, 'Iterative reconstruction does not substantially delay CT imaging in an emergency setting,' *Insights into imaging*, vol. 4, no. 3, pp. 391–397, 2013.

[27] X. Pan, E. Y. Sidky and M. Vannier, 'Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction?' *Inverse Problems*, vol. 25, no. 12, p. 123 009, Dec. 2009. DOI: `10.1088/0266-5611/25/12/123009`.

[28] G.-H. Chen, J. Tang and S. Leng, 'Prior image constrained compressed sensing (PICCS): A method to accurately reconstruct dynamic CT images from highly undersampled projection data sets,' *Medical Physics*, vol. 35, no. 2, pp. 660–663, 2008. DOI: `10.1118/1.2836423`.

[29] P. Paschalis, N. Giokaris, A. Karabarbounis, G. Loudos, D. Maintas, C. Papanicolas, V. Spanoudaki, C. Tsoumpas and E. Stiliaris, 'Tomographic image reconstruction using artificial neural networks,' *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 527, no. 1, pp. 211–215, 2004. DOI: `10.1016/j.nima.2004.03.122`.

[30] O. Ronneberger, P. Fischer and T. Brox, 'U-Net: Convolutional networks for biomedical image segmentation,' in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[31] A. L. Samuel, 'Some studies in machine learning using the game of checkers,' *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959. DOI: 10.1147/rd.33.0210.

[32] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. Cambridge, MA: MIT Press, 2010, ISBN: 978-0262043793.

[33] W. S. McCulloch and W. Pitts, 'A logical calculus of the ideas immanent in nervous activity,' *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. DOI: 10.1007/BF02478259.

[34] S.-C. Wang, 'Artificial neural network,' in *Interdisciplinary Computing in Java Programming*. Boston, MA: Springer US, 2003, pp. 81–100. DOI: 10.1007/978-1-4615-0377-4_5.

[35] G. Cybenko, 'Approximation by superpositions of a sigmoidal function,' *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989. DOI: 10.1007/BF02551274.

[36] Y. LeCun and C. Cortes, 'MNIST handwritten digit database,' 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[37] Y. LeCun, P. Haffner, L. Bottou and Y. Bengio, 'Object recognition with gradient-based learning,' in *Shape, contour and grouping in computer vision*, Springer, 1999, pp. 319–345.

[38] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'ImageNet classification with deep convolutional neural networks,' in *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012, pp. 1097–1105. DOI: 10.1145/3065386.

[39] H. Shan, Y. Zhang, Q. Yang, U. Kruger, M. K. Kalra, L. Sun, W. Cong and G. Wang, '3-D convolutional encoder-decoder network for low-dose CT via transfer learning from a 2-D trained network,' *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1522–1534, 2018. DOI: 10.1109/TMI.2018.2832217.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd. The MIT Press, 2009, ISBN: 0262033844.

[41] K. O'Shea and R. Nash, 'An introduction to convolutional neural networks,' 2015. arXiv: 1511.08458.

[42] M. A. Kramer, 'Nonlinear principal component analysis using autoassociative neural networks,' *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991. DOI: 10.1002/aic.690370209.

[43] V. Badrinarayanan, A. Kendall and R. Cipolla, 'SegNet: A deep convolutional encoder-decoder architecture for image segmentation,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. DOI: 10.1109/TPAMI.2016.2644615.

[44] I. Häggström, C. R. Schmidtlein, G. Campanella and T. J. Fuchs, 'DeepPET: A deep encoder–decoder network for directly solving the PET image reconstruction inverse problem,' *Medical Image Analysis*, vol. 54, pp. 253–262, 2019. DOI: `10.1016/j.media.2019.03.013`.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, 'Generative adversarial networks,' *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020. DOI: `10.1145/3422622`.

[46] H. Zhang, I. Goodfellow, D. Metaxas and A. Odena, 'Self-attention generative adversarial networks,' in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, Sep. 2019, pp. 7354–7363.

[47] A. Alsaiari, R. Rustagi, A. Alhakamy, M. M. Thomas and A. G. Forbes, 'Image denoising using a generative adversarial network,' in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, 2019, pp. 126–132. DOI: `10.1109/INFOCT.2019.8710893`.

[48] Y. You, I. Gitman and B. Ginsburg, 'Large batch training of convolutional networks,' 2017. arXiv: `1708.03888`.

[49] M. Claesen and B. D. Moor, 'Hyperparameter search in machine learning,' 2015. arXiv: `1502.02127`.

[50] M. Feurer and F. Hutter, 'Hyperparameter optimization,' in *Automated Machine Learning*, Springer, Cham, 2019, pp. 3–33.

[51] B. Kim, M. Han, H. Shim and J. Baek, 'A performance comparison of convolutional neural network-based image denoising methods: The effect of loss functions on low-dose CT images,' *Medical Physics*, vol. 46, no. 9, pp. 3906–3923, 2019. DOI: `10.1002/mp.13713`.

[52] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, 'Photo-realistic single image super-resolution using a generative adversarial network,' 2016. arXiv: `1609.04802`.

[53] K. Simonyan and A. Zisserman, 'Very deep convolutional networks for large-scale image recognition,' 2015. arXiv: `1409.1556`.

[54] 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*, 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[55] 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. DOI: `10.1109/TMI.2018.2827462`.

[56] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors,' *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. DOI: 10.1038/323533a0.

[57] H. Robbins and S. Monro, 'A stochastic approximation method,' *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951. DOI: 10.1214/aoms/1177729586.

[58] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization,' 2015. arXiv: 1412.6980.

[59] Y. Pang, J. Lin, T. Qin and Z. Chen, 'Image-to-image translation: Methods and applications,' 2021. arXiv: 2101.08629.

[60] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, 'Image-to-image translation with conditional adversarial networks,' 2018. arXiv: 1611.07004.

[61] P. Teo and D. Heeger, 'Perceptual image distortion,' in *Proceedings of 1st International Conference on Image Processing*, vol. 2, 1994, 982–986 vol.2. DOI: 10.1109/ICIP.1994.413502.

[62] A. Eskicioglu and P. Fisher, 'Image quality measures and their performance,' *IEEE Transactions on Communications*, vol. 43, no. 12, pp. 2959–2965, 1995. DOI: 10.1109/26.477498.

[63] Z. Wang, A. Bovik, H. Sheikh and E. Simoncelli, 'Image quality assessment: From error visibility to structural similarity,' *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[64] D. Brunet, E. R. Vrscay and Z. Wang, 'On the mathematical properties of the structural similarity index,' *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1488–1499, 2012. DOI: 10.1109/TIP.2011.2173206.

[65] W. Kanitpanyacharoen, D. Parkinson, F. De Carlo, F. Marone, M. Stampanoni, R. Mokso, A. MacDowell and H.-R. Wenk, *The tomography round-robin datasets*, 2016. DOI: 10.17038/XSD/1344306.

[66] D. Gürsoy, F. De Carlo, X. Xiao and C. Jacobsen, 'TomoPy: A framework for the analysis of synchrotron tomographic data,' *Journal of synchrotron radiation*, vol. 21, no. Pt 5, pp. 1188–1193, Sep. 2014. DOI: 10.1107/S1600577514013939.