Bachelor Datascience and Artificial Intelligence

Generating high resolution

CT images with GANs

Faruk Yüksel

Supervisors :
Dr. D.M. Pelt & M.C.R. Go

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

15/01/2026

1 Abstract

Generative Adversarial Networks (GANs) are powerful models for generating data that is statistically similar to a given dataset, making them attractive for domains where data acquisition may be costly or limited, such as tomography. In Computed Tomography (CT), scans are expensive and time-consuming to obtain, yet high-resolution data is essential for accurate interpretation. While GANs show promise in this field, they often struggle with reproducing fine-grained details, especially at higher resolutions.

This thesis investigates an alternative approach to high-resolution image generation by comparing two GAN-based pipelines: one that directly generates high-rsolution CT slices, and another that generates low-resolution slices which are subsequently upscaled. Both strategies employ WGAN-GP: a Wasserstein GAN variant that enforces the Lipschitz constraint through gradient penalty, resulting in more stable training than WGAN. For the upscaling, low-resolution GAN outputs are first upsampled using bicubic interpolation, and subsequently refined with SRCNN: a convolutional neural network trained to learn a mapping between low-resolution images and their high-resolution counterparts.

Model performance was evaluated using qualitative visual inspection, training dynamics and a quantitative metric in the Fréchet Inception Distance (FID). The results show that the low-resolution GAN pipeline consistently outperformed the high-resolution GAN in terms of visual realism and statistical similarity to real data. While SRCNN had limited impact on improving low-resolution GAN outputs, images upscaled using bicubic interpolation alone achieved comparable performance. These findings suggest that training GANs at low resolution followed by post-hoc upscaling can be an effective and computationally efficient alternative to direct high-resolution GAN training. Future work may investigate approaches with other GAN architectures and super-resolution techniques.

2 Contents

Contents

3 Introduction

Image generation is a popular topic in the field of Artificial Intelligence, with *Generative Adversarial Networks* (**GANs**) [GPAM+14] being among the most successful approaches. A GAN consists of two neural networks: a *generator* and a *discriminator*. The generator generates data as similar as possible to the training data, while the discriminator evaluates how similar generated data is to the real data. Similarity in this context refers to the synthetic data being statistically indistinguishable from the real data distribution. In this setup, the generator aims to "trick" the discriminator while the discriminator simultaneously improves at distinguishing synthetic from real data. Over time, this process incrementally improves both networks.
Early GANs typically produced low-resolution images. For example, the original paper by Goodfellow et al. [GPAM+14] used the MNIST dataset by LeCun et al. [LBBH98] and the Toronto faces dataset by Susskind et al. [SAH10], with resolutions of 28×28 and 48×48 pixels respectively. Those are very small numbers, especially for contexts where small details can be essential to correctly interpret an image. More modern architectures such as StyleGAN (Karras et al. [KLA19]) reach a maximum resolution of 1024×1024, which is much more capable of handling high-dimensional data, but at significantly higher computational cost.

One context where minor details are important is in *Tomography*, which is a term used for a type of imaging where some penetrating wave is used. By sending this wave through an object or organism, a varying amount of that wave is absorbed by the scanned subject based on internal density. Reading the waves that exit the opposite end allows for imaging the inside of the subject non-invasively, with relatively accurate 3-dimensional representations being possible if the wave is sent out from all around the object.
A well-known application of this is with *Computed Tomography* scans or CT-scans, which make use of X-rays [fda20]. The *Computed* part comes from an imaging system generating a number of cross-sectional slices, which, when put together in series, lead to a three-dimensional representation in a process referred to as *Tomographic Reconstruction*.
CT-scans can be quite expensive, considering that it requires specialized machinery, knowledgeable experts and software to calculate and ensure their realism. GANs then are seemingly natural fits for generating tomographic data, as they can assist other models that would make use of these scans by making it easier to construct a sufficient dataset. GANs historically struggle with generating accurate tomographic data however, due to it being high-resolution and detail-sensitive. *CT-SGAN* [PWH21], which, while still improving on prior models measurably, was not yet realistic enough to reliably generate CT data.

There are multiple ways this problem could be addressed, but the approach that is explored in this thesis is the use of upscaling on the output of a GAN. Upscaling in the context of GANs is generally performed by a GAN itself, such as with ESRGAN [WYW+18], which then lets the user either enlarge an existing image to higher resolution or view a smaller section in more detail.
Training low-resolution GANs followed by post-hoc upscaling on the other hand has limited research, which makes it an interesting candidate to investigate. It is possible that the advantages of GANs trained on lower resolution images can translate to higher resolution images once they have undergone upscaling. Additionally, it decouples image generation and resolution refinement, which means the two different methods can be tailored more specifically.

The main question to be answered is: **How effective is post-hoc upscaling at increasing the quality and resolution of low-resolution CT-scans generated by GANs?** This comes with a few related sub-questions:

- **How does upscaling affect the introduction of artifacts in synthetic CT-scans?** Some methods may indeed increase resolution but at the cost of introducing noisy or unrealistic patterns.
- **Does the combination of low-resolution GANs and upscaling provide a favorable trade-off between image quality and computational efficiency compared to training high-resolution GANs directly?** Even if upscaling leads to worse quality data, it might be more advantageous in contexts where producing data in larger quality is more important than perfect quality. Differences in resolution being quadratic can lead to a drastic difference in computation time.

4 Background

This section serves as an overview on all important topics related to the research.

4.1 Neural Networks

Neural Networks (NNs) are a widely used computational framework within machine learning. It is based on a loose model of neurons in the brain. NNs are composed of artificial neurons which are connected by *edges*, fulfilling the same purpose as synapses. These neurons are arranged in layers, with every NN containing an input- and an output layer with any amount of hidden layers in between them.

Each neuron receives some information from any connected nodes in prior layers, which then performs transformations on that input. It then computes the input with some non-linear function (referred to as the *Activation Function*) to produce the neuron's output. This produced value is a real number, and is sent to each connected subsequent neuron. The strength of this value is additionally influenced by the *weight* value of the connection between the current and the next neuron. Each hidden layer transforms values from its previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + ... + w_mx_m$, followed by activation function $f : R^m \to R^o$.
The main goal of a NN is, using a set of features $X = \{x_1, x_2, ..., x_m\}$, to learn an approximation of the input-output relationship that optimizes the weights between the neurons while outputting target y.

A common approach of optimizing this activation function is done by first estimating the loss of a function. By adjusting parameters of the model, such as the weights, the loss of the model often changes. The *gradient* is defined as the vector of partial derivatives of the loss function with respect to the model parameters, indicating how small changes in each parameter affect the loss.
This gradient value is used to optimize the model in a process referred to as gradient descent, as ideally during gradient descent the cost function is minimized by modifying the parameters until the lowest point of the curve is reached. Modification of the parameters is done by going backwards

through the layers of the model, as that makes computation more effective via the chain rule. Since it is calculated backwards, it is referred to as *backpropagation* [IBM].

Hidden layers can be composed of more neurons than are present in prior layers, but generally the output layer is composed of less neurons than the input layer, since that improves decision-making ability for the model. A NN used for a binary decision problem may only have a single neuron in its output layer, with its value being the key factor for which decision is made. With multiple neurons in the input layer, each may correspond to different labels in a labeling problem. It is widely applicable, as the architecture is flexible enough to support many different types of problems, including supervised and unsupervised learning [Har17]. It stands out from regular regression methods due to the multiple possible hidden layers allowing for non-linearly separable data to be distinguished.

A common application of NNs is with MultiLayer Perceptrons (MLP). MLPs are a variant of NN where every layer is fully connected, meaning that each neuron is connected to every neuron in its previous layer [lD25].

4.2 GANs

4.2.1 GAN

As was mentioned in section 3, a GAN is composed of two smaller neural networks: one generator that generates images, and one discriminator that discerns real data from synthetic data. Rather than explicitly optimizing image quality, GAN training is formulated as a minimax two-player game where the generator and discriminator have opposing goals [MW]. The model was implemented by making use of the MLPs mentioned in 4.1, as MLPs are expressive function approximators that align with the goals of discrimination and generation.
One agent's gain is the other's loss, as a more powerful discriminator requires the generator to create more realistic images, and a more powerful generator requires the discriminator to improve on its abilities. Goodfellow et al. [GPAM+14] describe it as analogous to a team of counterfeiters attempting to create fake currency without being detected, while the police tries to detect it. Eventually the competition would end when the counterfeit currency is functionally identical to real currency.

Important terms for GANs are G and D that stand for the generator and the discriminator, and p_{data} and p_g which stand for the distribution that follows from all real datapoints and all generated datapoints respectively.
First, to learn the generator's distribution p_g over data x they defined a prior on input noise variables $p_z(z)$, and representing a mapping to the data space with $G(z;\theta_g)$ where G is a differentiable function, represented by a MLP with parameters θ_g. The second MLP for discriminator $D(x;\theta_d)$ outputs a single scalar $D(x)$, which represents the probability that data x is real data rather than p_g. D is trained to maximize the probability of correctly labeling training examples and samples from G. G is simultaneously trained to minimize $log(1 - D(G(z)))$. With p_{data}, this corresponds to the discriminator D and generator G playing a minimax game with the following value function $V(G, D)$:

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

Fully optimizing D in the inner loop of training can lead to vanishing generator gradients, or the discriminator growing too powerful too quickly, hence why they employed k steps of optimizing the D and one step of optimizing G per loop. This is an effective way to ensure D maintains its optimal solution, but G suffers early on in the training process, as D is easily able to discern the poor data from G, which saturates $log(1 - D(G(z)))$. Instead, G was trained to maximize $log(D(G(z)))$, which results in the same fixed point of the dynamics in G and D but allows for better gradients early in learning.

4.2.2 WGAN

Goodfellow et. al's original GAN framework [GPAM+14] proved effective, and has since inspired many variants. However, the base architecture suffers from a number of training issues, such as vanishing gradients. One variant designed to address some of these issues is *Wasserstein GAN*, or **WGAN** for short [ACB17].

The main topic of research by Arjovsky et al. concerns the choice of distance or divergence used to compare the probability distributions p_{data} and p_g, which quantifies how close p_g is to p_{data}, as an ideal GAN would have both distributions be identical.

The researched choices for distance and divergence were: *Total Variation* (TV) distance, *Kullback-Leibler* (KL) divergence, *Jensen-Shannon* (JS) divergence and *Earth-Mover* (EM) distance, also known as Wasserstein-1. TV, KL and JS become uninformative when p_g and p_{data} do not overlap, which can frequently be an issue when handling high-dimensional data such as images. Not only that, but it is also a common issue early on in most GAN methods. In these cases, divergences can fail to provide meaningful gradients, making learning unstable or ineffective as near-zero gradients may be provided. In contrast, Wasserstein distance remains stable under these conditions. It is calculated as follows:

$$W(p_{data}, p_g) = \inf_{\gamma \in \Pi(p_{data}, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{2}$$

where $\Pi(p_{data}, p_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are p_{data} and p_g. Each joint distribution specifies a 'transport plan' describing how much 'mass' is moved from samples $x \sim p_{data}$ to samples $y \sim p_g$. Intuitively, the Wasserstein distance measures the minimum cost required to transform one distribution into the other, with the cost being defined as the amount of mass transported multiplied by the moved distance.

Motivated by the Wasserstein distance having nice properties, Arjovsky et al. introduced the WGAN. In this framework, the discriminator is replaced by a *critic* which outputs real valued scores rather than probabilities. This critic approximates the Wasserstein-1 distance between p_{data} and p_g by maximizing the difference between expected outputs on real and generated samples, subject to a 1-Lipschitz constraint.

A function is 1-Lipschitz if its gradient norm is bounded by 1 almost everywhere. It is not necessary for every critic to satisfy this Lipschitz constraint, but it was found in the paper that critics that do satisfy it tend to produce better results.

The generator, in turn, minimizes this expected distance. In the original WGAN paper, the Lipschitz constraint is enforced by weight clipping, which crudely changes weights to be within the specified range. The authors note that this approach is imperfect, and invite future researchers to find better strategies. By reformulating GAN training as the minimization of Wasserstein distance, WGAN achieved a more stable performance than a standard GAN model.

The WGAN training objective is given by:

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim p_{data}}[D(x)] - \mathop{\mathbb{E}}_{\tilde{x} \sim p_g}[D(\tilde{x})] \tag{3}$$

where $\mathcal{D}$ denotes the set of all 1-Lipschitz functions and p_g is the model distribution implied by $\tilde{x} = G(z), z \sim p(z)$. Optimizing this objective with respect to the generator parameters minimizes the Wasserstein distance $W(p_{data}, p_g)$.

4.2.3 WGAN-GP

As Arjovsky et al. [ACB17] noted in their research, weight clipping is an ineffective strategy for enforcing the Lipschitz constraint required by WGANs, often leading to optimization difficulties and poor convergence. To address this issue, Gulrajani et al. proposed an alternative approach by making use of Gradient Penalty (WGAN-GP), which enforces the Lipschitz constraint more directly and does not suffer from the same unintended behavior as weight clipping [GAA+17].
As established in 4.2.2, WGAN requires the critic D to be 1-Lipschitz. A sufficient condition for this property is that the norm of the gradient of D, or the magnitude of that gradient vector, with respect to its input is bounded by 1 almost everywhere. WGAN-GP enforces this by directly penalizing deviations of the gradient norm, leading to the following critic loss function:

$$L = \mathop{E}_{\tilde{x} \sim p_g}[D(\tilde{x})] - \mathop{E}_{x \sim p_{data}}[D(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{4}$$

Here, $\mathbb{P}_{\hat{x}}$ is neither real data p_{data} nor generated data p_g, but is sampled uniformly on lines between sampled points from both data distributions. This choice is supported by the fact that an optimal critic for the Wasserstein distance exhibits linear behavior with the unit gradient norm along paths connecting coupled points from the two data distributions.
The gradient norm constraint is only on samples from $\mathbb{P}_{\hat{x}}$, as enforcing it everywhere is computationally difficult. In practice, Gulrajani et al. [GAA+17] noted that this restricted enforcement sufficed and led to stable results.
Equation 4 can be interpreted as follows. Samples are first drawn from p_{data} and p_g, which are then interpolated to get $\hat{x}$. With $\hat{x}$, the critic output $D(\hat{x})$ can be calculated, allowing the gradient with respect to the input $g = \nabla_{\hat{x}} D(\hat{x})$ to be computed. The gradient norm $||g||_2$ is then penalized according to $((||g||_2 - 1)^2)$, such that deviating gradient norms above and below 1 are penalized. This encourages the critic gradients to have a norm close to 1, matching the theoretical optimal critic.
Finally, λ, denotes the penalty coefficient which is set to 10, as this value was found empirically to work well across a wide range architectures and datasets [GAA+17].
WGAN-GP was shown to significantly outperform the original WGAN [ACB17], demonstrating that gradient penalty is a more effective measure of enforcing the Lipschitz constraint than weight clipping.

4.3 GANs in Tomography

Tomography is a field with a very wide spread of applications. A common use-case, in particular with computed tomography, is in the medical field as a way to non-invasively scan patients. This is especially useful for chest x-rays, as there is no non-invasive alternative to visualize internal organs [fda20]. Though that is not the only use case for computed tomography, as non-medical fields have a wide array of uses as well. Van Kaick and Delorme [vKD05] found several fields where tomography is useful, ranging from utilizing it to peer into sarcophagi and mummies without potentially damaging these delicate objects, to analyzing soil samples to evaluate soil fertility, to even uses in the industry and aviation on assembly lines and baggage checks respectively.
It was mentioned in the introduction (3) that GANs are natural fits for tomography, considering the fact that it can be quite difficult to gather a dataset that is of satisfactory quality to train a model. CT acquisition is expensive due to equipment, software and staffing requirements; hence why GANs in tomography are a prominent topic of research with many different implementations. One such implementation is the earlier mentioned CT-SGAN [PWH21], a type of GAN trained on chest scans. A goal of CT-SGAN is to generate scans by training on a limited set of training samples of 900 slices, while the GAN generates consecutive slices of an object. It is composed of four networks: Recurrent neural network R_v that captures volumetric information across slices, slice generator G_{slice}, and the slab discriminators D_{slice} and D_{slab} that discriminate at different scales. This architecture is quite different from the simpler GANs such as GAN [GPAM+14] or WGAN [ACB17], showing that it is a flexible architecture with many different possible implementations for assisting in expanding CT dataset size. CT-SGAN produced visually realistic slices and improved quantitative metrics compared to baseline methods.

4.4 Upscaling

Image upscaling, or super-resolution, has been studied long before machine learning methods were adapted for the task. Most methods of upscaling can be divided into two types of techniques: non-adaptive techniques, and adaptive techniques [PV15]. Non-adaptive techniques include the more rudimentary approaches that do not consider features or contexts of an image, but directly manipulate images. Some simple non-adaptive techniques include nearest neighbor, where each new pixel receives the same value as the nearest pixel in the original image source, and bilinear/bicubic interpolation, where new pixels take a weighted average of the values of either their neighboring pixels or near neighborhood respectively.
Adaptive techniques tend to be more powerful than the previous ones, but due to having to consider image features they also require much more computational power. A common and powerful approach is making use of *Convolutional Neural Networks* (CNNs) that make use of convolutions (or mathematical operations) on images. One such implementation is *Super Resolution CNN* (SRCNN), a CNN tailored for image super-resolution [CLHT15].

The goal of SRCNN is described as a real upscaled low-resolution image $\mathbf{Y}$, from which they tried to recover image $F(\mathbf{Y})$ which is as similar as possible to the ground truth high-resolution image $\mathbf{X}$. This involves learning the ideal mapping for F, which is done in three steps:

1. **Patch extraction and representation**: Extract overlapping patches from $\mathbf{Y}$ and represent each patch as a set of convolutional feature maps that encode local image structure.

2. **Non-linear mapping**: Transform the extracted feature maps through non-linear convolution to produce high-resolution feature maps that each correspond to a patch.

3. **Reconstruction**: Aggregate the high-resolution patch-wise representations to generate the final high-resolution image $F(\mathbf{Y})$.

Training is done with a dataset of images, and for each high-resolution image, a correspdoning low-resolution image is generated by bicubic downsampling. This downsampled set is then upscaled back to its original resolution with bicubic interpolation. With those two, SRCNN attempts to find a mapping to transform the upsampled low-resolution images as accurately as possible into their real high-resolution form. SRCNN does not upsample images at any point during the process, but rather is a model that can accurately refine images that have been upsampled with simple bicubic interpolation.

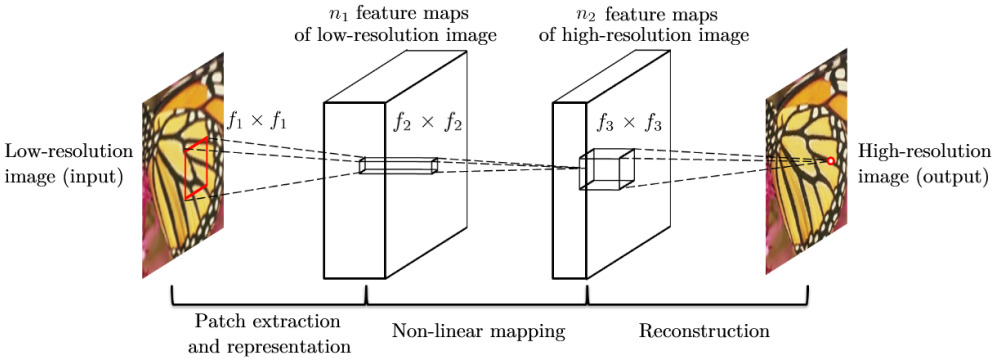


Figure 1: A visual representation of how SRCNN works from the paper by Dong et al. [CLHT15]. It shows the three steps SRCNN takes to upscale a low-resolution image $\mathbf{Y}$ with a mapping F such that $F(\mathbf{Y})$ is as close as possible to the ground truth high-resolution image $\mathbf{X}$. It first extracts patches from $\mathbf{Y}$, which it represents as vectors that are then mapped onto each other to create patches. These patches are then aggregated to generate the final image $F(\mathbf{Y})$.

5 Methodology

5.1 Dataset

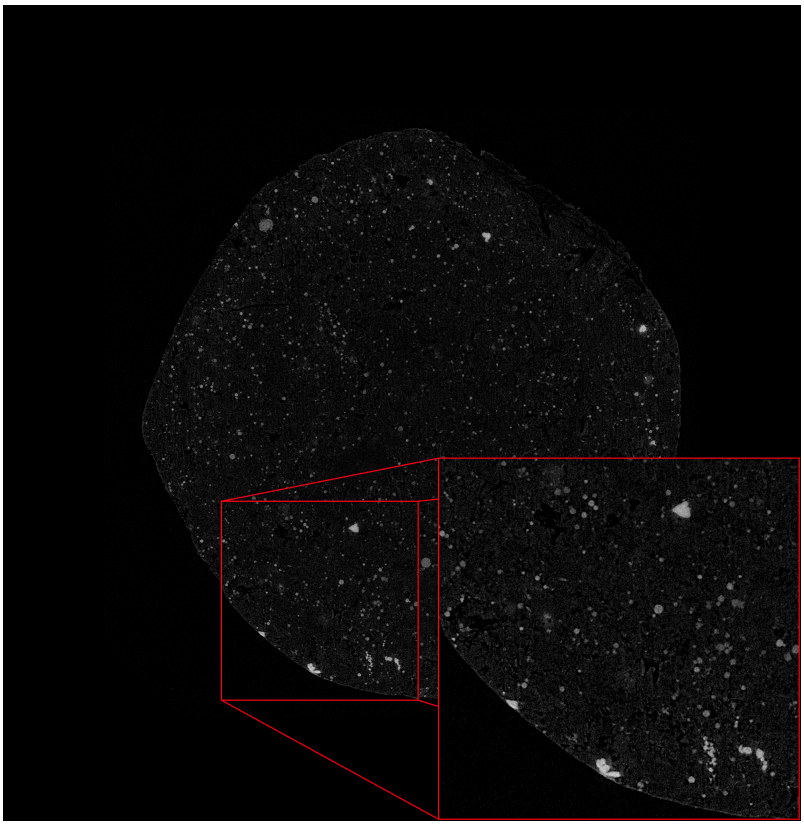


Figure 2: One example slice from the *Round_Robin* dataset from Tomobank, reconstructed using their *tomopy_rec.py* file for reconstructing datasets from the .h5 format mentioned in subsection 5.1. The original data is in the 32-bit .tiff format which is not accepted by the chosen GAN implementation (and cannot be shown in LaTeX), hence why the dataset has been converted into the 8-bit .png format. For visibility in the figure it has also been edited in image-editing software to be brighter and have higher contrast, but this is not performed for the actual dataset outside of the methods described in 5.1. A smaller section highlighted with a red border is enlarged for a more detailed view.

This experiment makes use of *Tomobank*, a data bank consisting of qualitative tomographic datasets [GCXJ14]. The *Round_Robin* subset includes two categories: North Sea shale samples and sand samples. For this study, the North Sea shale samples were selected due to their high-resolution data, small grain size and strong shape- and lattice-preferred orientation. Small grain size is particularly important to evaluate whether the upscaled GAN can reproduce fine details as effectively as the high-resolution GAN.

The data is stored in *.h5* files, which is a hierarchical data format that can store large amounts of data in multidimensional arrays. The dataset used from the shale sample datasets is *tomo_00001*,

as it consists of 1792 slices, which is large enough to be able to train a GAN.
First, the data must be reconstructed from the .h5 format to be usable, as downloading the uncompressed data is impractical. A step-by-step instruction is available on the *Round_Robin* page on the Tomobank website, where the offer the python script '*tomopy_rec.py*' that can be ran to reconstruct any dataset in their database. This reconstructs the data into the *.tiff* file format, which can be done for either a single slice from the specified file, or for the entire dataset. For the script to work however, an environment has to be set up with Conda. A full guide on how to do so can be found here on their website.
This data must be converted into a different file format, as .tiff files are 32-bit — a format that is not accepted by the chosen GAN implementations (detailed in 5.2). Additionally, the slices are 2048×2048 pixels, which is much too detailed for most GAN use. Therefore, a custom script was necessary to both reformat the data from .tiff to .png — a type of format accepted by the GAN due to being 8-bit — and to downscale the images from 2048×2048 pixels to 256×256 and 64×64 for the high- and low-resolution GANs respectively. 64×64 is sufficient to preserve the global shale structure and finer details, and 256×256 was chosen as it is increased with a consistent scaling factor of 4, is able to capture the details well and provides a challenge for the upscaling method at a computationally feasible level. Input/output directories, target resolution, and logging options can be adjusted. Predetermined intensity bounds are also provided for the test slice generated with *tomopy_rec.py*.
To reduce inter-scan brightness variability, intensity clipping was applied to all images. The minimum and maximum intensity values of every image were collected into separate distributions, with the global intensity clipping limits being set to the lower and upper 1st percentile values of these distributions. The same bounds were also used for intensity normalization, which enhanced image contrast and visibility while preserving the relative proportions between intensity values.
Lastly, the first 59 slices corresponding to the sample edges were excluded from the dataset, as they had minimal features.

Figure 3 shows a slice of the shale sample dataset at the two resolution sizes used in this study, displayed at their true proportions. The image exhibits low contrast, with most regions appearing in similar shades of gray, and a circular ring surrounding the shale sample. This ring is present in all slices and is unlikely to affect image quality.
Within the shale sample itself, dark gray regions correspond to likely air pockets or empty space near the edges of the sample. The lighter areas, consisting of small-grain structures dispersed throughout the slice, likely represent mineral deposits that absorb more radiation and therefore appear brighter in the scan. Their varying size, distribution and density of these features present the primary challenge for the GANs to replicate accurately.

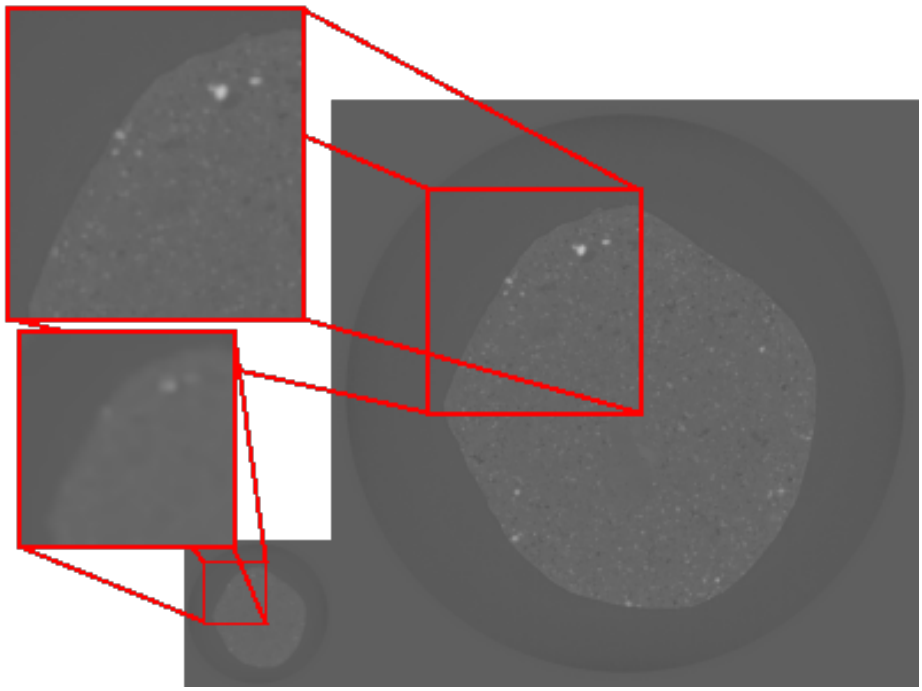


Figure 3: One slice of the Round_Robin shale sample dataset that is portrayed at a resolution of both 64×64 and of 256×256 in their real size proportions. These are examples of the data that is used to train both the low- and high-resolution GANS, with no changes made for visibility like in figure 2. The same section in both slices has been highlighted in red and enlarged on the left-hand side. The difference in image quality between the two resolutions is evident, with the finer details being lost in the low-resolution counterpart.

5.2 GAN Implementations

Simple GAN architectures were preferred to ensure that differences in performance between the GANs could be attributed to the upscaling method, rather than model complexity.
With that in mind, the Wasserstein GAN (WGAN) [ACB17] was first selected. The WGAN is quite simple, and can easily be modified to handle CT scan data instead of the MNIST dataset [LBBH98] as is it does in the template. However, the WGAN on its own performed quite poorly on the high-resolution dataset, which is why a slightly more powerful variant was necessary.
While the reduced performance of the high-resolution WGAN might reflect an inherent characteristic of the underlying problem, and modifying the implementation could potentially affect the realism of the process, we adopted WGAN-GP to mitigate optimization stability. This allowed both low- and high-resolution GANs to achieve comparable performance levels, which facilitates a more meaningful comparative analysis. In addition, WGAN-GP can easily be implemented with small changes to how the Lipschitz constraint is satisfied, as it is nearly identical in other aspects.

A GitHub repository consisting of GAN implementations making use of PyTorch was used for WGAN-GP [LN]. This repository links to many papers researching 32 GAN strategies, along with a PyTorch implementation for each of them. Some modifications were necessary to make ensure it

would work for the Tomobank dataset, however.
Minimal changes were made aside from the number of training epochs. For this experiment, the time budget was set to 4 hours for both algorithms rather than an iteration budget for a total training time of 8 hours since GANs can theoretically continue improving indefinitely. Therefore, the number of epochs was set to an impossibly high number to reach, with time being the more relevant factor for a GAN rather than a set number of epochs. The time limit was measured at the end of every epoch, which prevented it from interfering with progress images being saved.
The latent dimensions value, a variable that stands for the dimensionality of the latent space, was modified for the high-resolution GAN. At first we attempted a trial run with the latent dimensions being equal for both models so that there would be as little outside influence on model performance as possible, but that greatly worsened the results of the high-resolution GAN. Therefore it was increased to 200, which yielded much more competitive results that allow more informative comparisons.
Loss values for both the generator and **critic** (as WGAN-GP has a critic, not a discriminator) are stored in a separate log file, since otherwise they did not end up saved. These give insight into the performance of the two sides of the GANs, forming a useful tool to evaluate both models.
The original implementation made use of sequential layers, which puts all pixels in one long "list" of sorts. Training can certainly be done sequentially, but that lessens the ability of the GAN to learn proper spatial structure and will heavily impact results. Turning it into a convolutional network does allow it to learn that structure. In the initial layer for the generator, the image first starts at a resolution of 16×16, which is just enough for it to be able to encode global structure. The subsequent layers were separated into "blocks" of layers, with each "block" being separated a number of steps:

1. Upsample the image to double spacial resolution
2. Apply convolution to refine features
3. Apply ReLU activation
4. Repeat convolution and ReLU to further refine features

Since the initial image is 16×16, doubled spacial resolution for each block allows for clean upsampling into 64×64 and 256×256 by adding blocks to the model until the desired resolution is reached at the end. Convolution and activation are performed twice, as the first time it adapts the features to the new higher resolution, and the second time lets it refine the features a little more. Each step starts with a number of channels equal to twice the final desired resolution size, with the number of channels halving from the second block onward. As the size of the image shrinks, focus is put on the finer details rather than the larger structure, which requires less channels. The generator outputs images with a final $Tanh()$ activation, which normalizes the output range from $[-1, 1]$, stabilizing the GAN by preventing exploding coefficients.
The critic mirrors the generator's architecture, excluding upsampling. Channel dimensions increase across blocks to capture finer details progressively. LeakyReLU is used here over regular ReLU, as neurons never activating is a more likely possibility in the critic. LeakyReLU activations were used to prevent neurons from remaining inactive, ensuring stable gradient flow. Finally, the critic maps the learned feature representations to a single scalar, which reflects the realism of the input image.

Progress images by the generator were initially saved every couple of batches, but this interfered with the number of epochs, as the batch interval would not always align with the number of epochs. Now, progress images are saved every n epochs instead, with n being set to 5 for the low-resolution GAN, and 2 for the high-resolution GAN. This is done because the low-resolution GAN has a much shorter training loop than the high-resolution GAN, which resulted in a much larger amount of epochs in the set time. A smaller n for the low-resolution GAN results in an excessively large number of progress images being saved, while a larger n for the high-resolution GAN provides less insight into the improvement of the GAN over time.

5.3 Upscaling Method

SRCNN (introduced in subsection 4.4) was used to upscale the low-resolution images, with the implementation sourced from this git repository [Lor]. The model was trained on a variant of the low-resolution dataset that was upscaled with simple bicubic interpolation (mentioned in 4.4), which was done prior with a simple script making use of OpenCV [Smo]. From this, it learns a mapping on how to refine these images so that they match the true high-resolution images as closely as possible.
During training, a subset of data is reserved as a validation set. At the end of each epoch, the SRCNN output for the validation images is compared to the corresponding ground-truth high-resolution images, which is used to compute the *Peak Signal-to-Noise Ratio* (PSNR) and the *Structural Similarity Index Measure* (SSIM). Model checkpoints are saved throughout training, with the checkpoint that achieves the highest PSNR and SSIM on the validation set being selected to refine the upscaled low-resolution outputs.
PSNR and SSIM are both standard image quality metrics to gauge reconstruction performance. PSNR quantifies the pixel-wise reconstruction fidelity between a generated image and its real data counterpart, with higher values indicating a lower reconstruction error. SSIM measures visual similarity based on structural information, luminance and contrast of the image.

The SRCNN model required minimal modifications to accommodate grayscale images. Specifically, single-channel images were internally converted to three-channel inputs, as the original model was trained on color images. The images themselves are not altered, which does not affect the final output or the learning process.
The original variant of the SRCNN implementation used 58000 training epochs, which is excessive for this dataset and may lead to overfitting or redundant calculations. To address this, early stopping based on the *Mean Squared Error* (MSE) was implemented. The average validation MSE is calculated at the end of every training epoch alongside PSNR and SSIM, with training being terminated early if no improvement greater than 1×10^{-5} (rounded to six decimals) is observed over 20 consecutive epochs. Since the model saves the best-performing checkpoint at each epoch, early stopping does not risk discarding the optimal model.

5.4 Evaluation

Evaluation is a simpler compared to model training, but remains essential for assessing GAN performance. Since the GANs produce image slices as output, preliminary qualitative evaluation can be performed based on visual inspection by comparing real slices, high-resolution GAN-generated slices and SRCNN-refined low-resolution GAN slices.

However, manual inspection alone is insufficient, as it is subjective and may fail to capture subtle statistical differences between the image distributions. While critic and generator loss provides insight into the training dynamics, low loss does not necessarily imply perceptual or statistical similarity to the real data.

To complement qualitative assessment, a quantitative evaluation metric is required. The Fréchet Inception Distance (FID) [HRUN18] is employed to measure similarity between the distribution of real images, p_{data}, and the distribution of generated images, p_g. FID computes the mean and covariance of deep feature representations extracted from an Inception network, with a lower value indicating closer alignment between the two distributions. Examples visualising the relationship between various type of image disturbances and FID can be observed in figure 4.

In this study, FID is computed using the same dataset that was used for training. While using a held-out dataset is often preferred, comparing generated images directly against their corresponding real data can be informative in this context, as it allows for a direct assessment of how closely each model reproduces the characteristics of their training data. The resulting scores are used to compare the performance of the high-resolution GAN and the SRCNN-enhanced low-resolution GAN.

As an additional baseline, the real dataset is randomly split into two equally sized subsets. The FID score is then calculated between them, providing an estimate of the inherent variability within the dataset. A generated dataset achieving an FID score close to this baseline indicates that it successfully replicates the statistical spread of the real dataset.

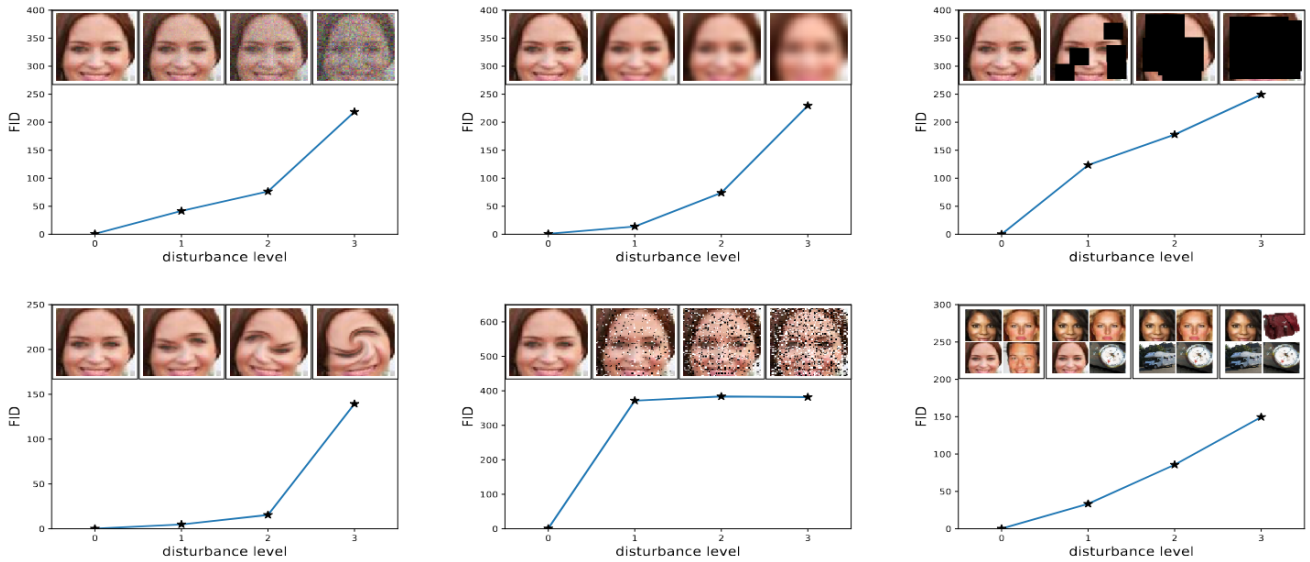
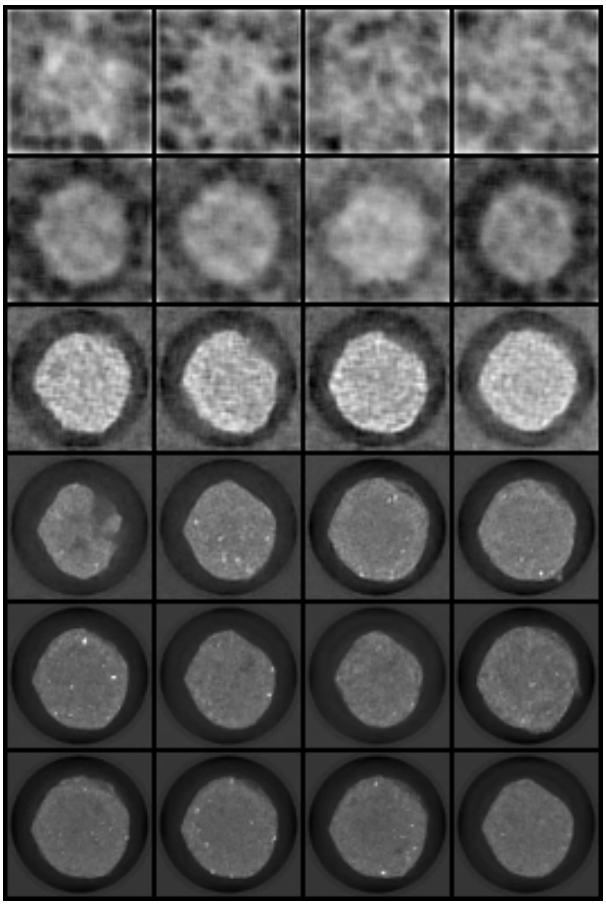
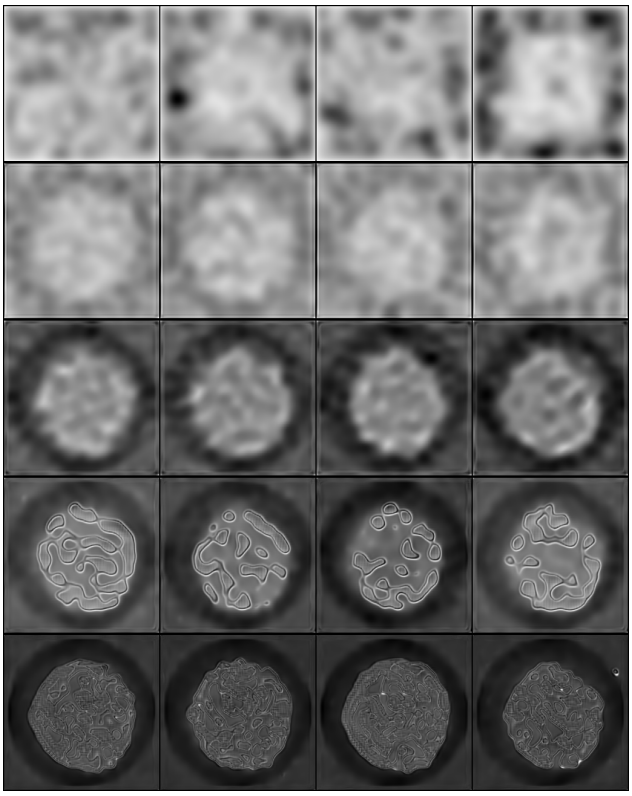


Figure 4: An example of a number of disturbances and their effect on the Fréchet Inception Distance (FID), sourced from the paper by Heusel et al.[HRUN18]. The FID score is on the y-axis, with the level of disturbance on the image on the x-axis. Every disturbance level is visualised Types of disturbances are in order: Gaussian noise, Gaussian blur, implanted black rectangles, a swirl effect, salt and pepper noise, and the used CelebA dataset [LLWT15] interjected with imagenet images [RDS+15].

6 Results



(a) Low-resolution GAN



(b) High-resolution GAN

Figure 5: Two figures containing several generated slices by both the low- and high-resolution gan over time, separated by thin black lines. Each row contains four generated slices for a specific epoch. Images were generated every n epochs, with $n = 5$ for the low-resolution GAN and $n = 2$ for the high-resolution GAN. Images for this figure were hand-picked based on epochs where significant change in the quality of images was visible. The low-resolution GAN shows recorded images at epochs 20, 60, 360, 1500, 3600 and 7500. The high-resolution GAN shows recorded images at epochs 20, 60, 140, 240, 464

Figure 5 shows generated images illustrating the progress for the low- and high-resolution GANs over time. Each row corresponds to a set of four images generated at the same epoch, selected at points where notable improvements in image quality were visible. Due to the smaller resolution of its training data, the low-resolution GAN completed 7630 epochs, whereas the high-resolution GAN completed only 466.

By the end of training, both GANs had successfully learned the overall structure of the shale sample slices. The low-resolution GAN produced slices that closely resemble the training data shown in figure 3, with minor noise being visible within the shale structure. The high-resolution GAN also captured the global structure and some finer details, such as the mineral deposits; however, it

introduced irregular, blotchy patterns not present in the original dataset. Additionally, regions outside these patterns appear noticeably blurrier compared to the outputs from the low-resolution GAN.

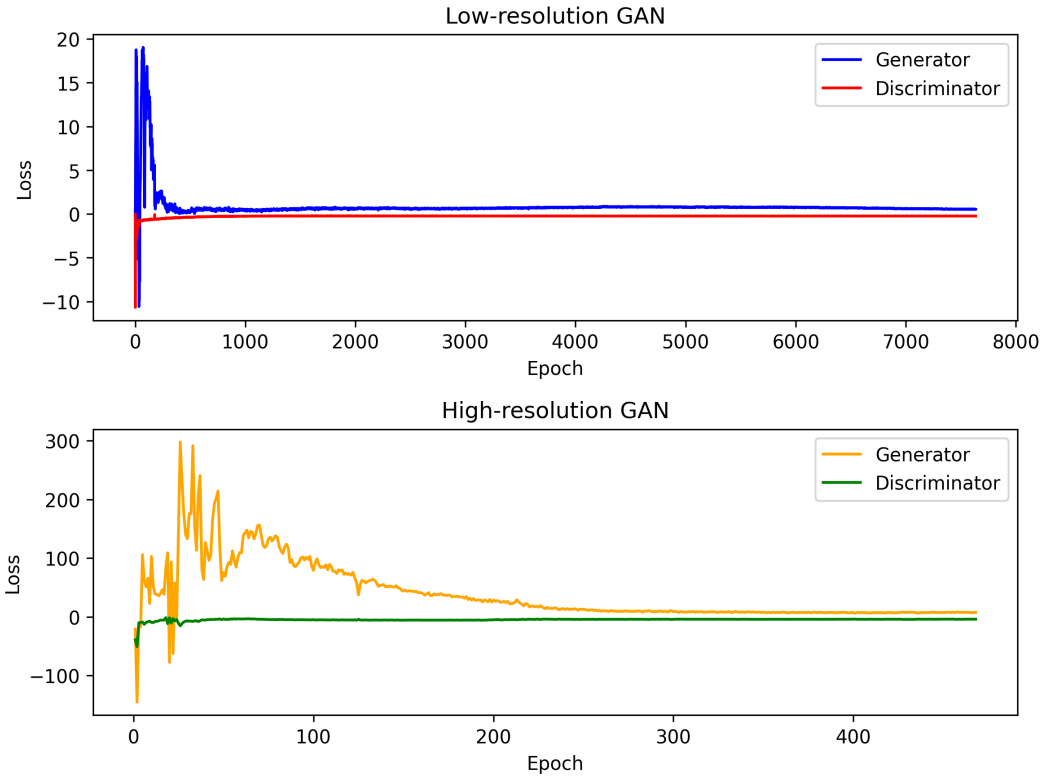


Figure 6: Two plots that show the loss of the generator and critic for both the low- and high-resolution GAN implementations. The x-axis and y-axis for both plots stand for the number of epochs and loss, although the values on the axes are different. This is both due to the low-resolution GAN running for many more epochs, and the generator of the high-resolution having significantly higher loss than the other generator and critics, especially during the first 200 epochs. While the low-resolution GAN stabilizes in 400 epochs and the high-resolution GAN in 250, it is much earlier relative to the amount of real time that has passed during learning.

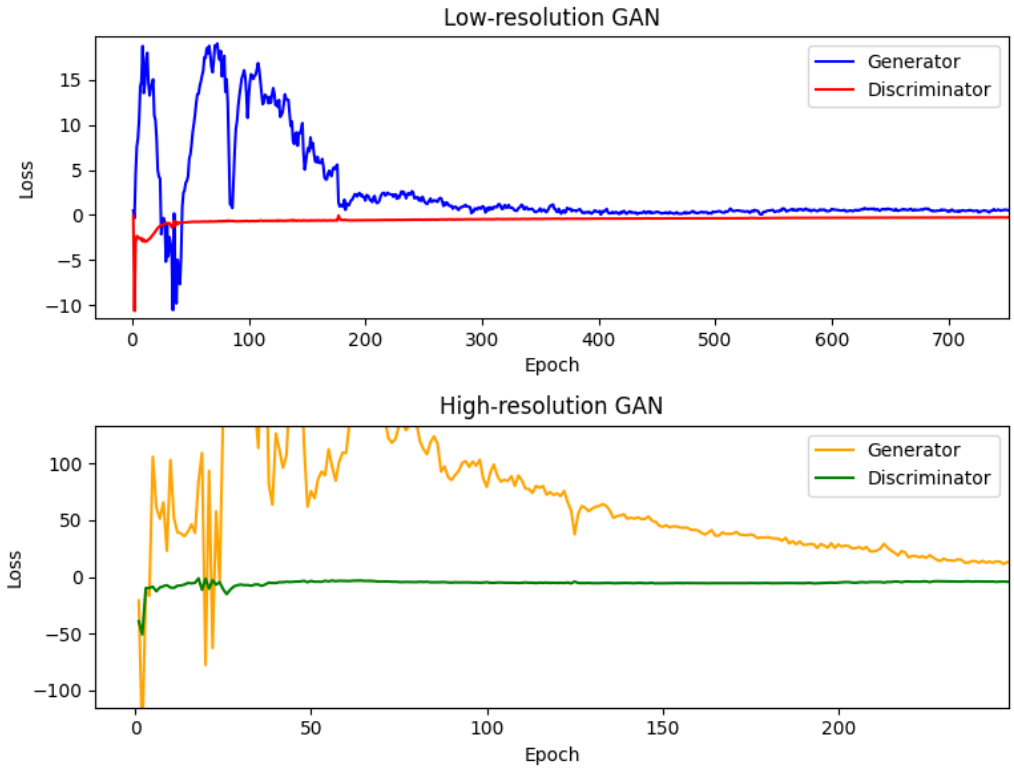


Figure 7: The same plot as in figure 6, but moved to the left-hand side of the plot where the initial training and critic losses are more visible, as in figure 6 a significant portion is dedicated to the stabilized loss deeper into the training process. The x-axis now extends to 750 epochs for the low-resolution GAN and 250 for the high-resolution GAN

Figure 6 and 7 show the loss of both GAN implementations per epoch, with the y-axis representing the loss, and the x-axis representing the epoch count. The critics for both models converge relatively quickly compared to the generators. The low-resolution GAN's generator also converges early, whereas the high-resolution GAN's generator exhibits higher loss values and only begins to stabilize after approximately half of the training time has elapsed.

Convergence of both networks during training is generally a positive sign, indicating that the generator and critic are improving in a balanced, adversarial manner. The quality of the generated images, however, ultimately determines whether the convergence is truly positive or negative. As observed in figure 5, both models consistenly improve their outputs, suggesting that the behavior visible in the loss corresponds to effective learning. Loss values stagnating without corresponding improvement in image quality is a classic sign of training issues.

The difference in stabilization timing is notable. The low-resolution GAN's generator stabilizes within roughly 400 epochs, while the high resolution GAN's generator does so after approximately 250 epochs. Despite the larger number of epochs for the low-resolution GAN, its training completes much faster in real elapsed time due to the smaller input resolution leading to less time required per epoch.

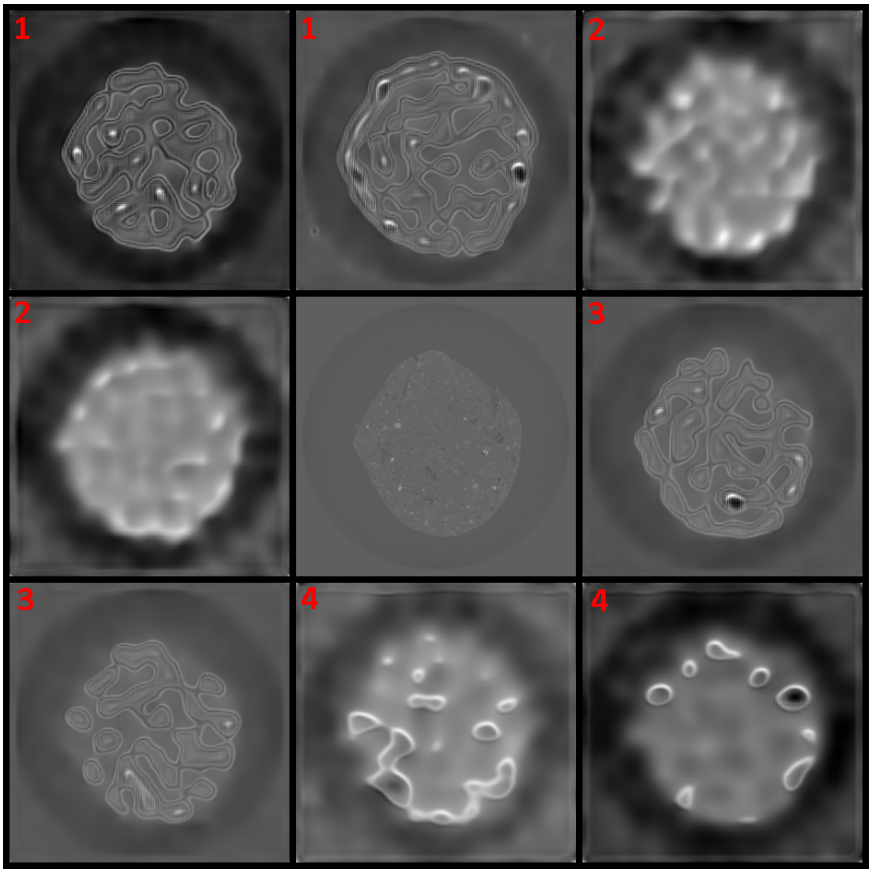


Figure 8: A set of images generated by a number of different iterations of the high-resolution GAN, with the central image being a real slice as a point of comparison. Slices with the same number in the top left correspond to the same iteration of high-resolution GAN. The high-resolution GAN was quite volatile, as the only difference between these variants and the final model that can be seen in figure 3b are a slightly modified number of latent dimensions set in the hyperparameters, or simply restarting the GAN when it stagnated or entered mode collapse (a mode where output diversity is extremely low).

While not the final iteration of the model used for comparison, the high-resolution GAN proved unstable during implementation and experimentation. Figure 8 shows some examples of unused variants of the high-resolution GAN, with a real slice placed in the center for comparison. Although the overall shale sample structure is discernible, these iterations exhibited a number of common issues that are still present, to a lesser extent, in the final model (figure 5b). Iterations 2 and 4, in particular, are excessively blurry, making them unsuitable as comparison points against the low-resolution GAN. Iterations 1, 3 and, to a lesser extent, 4 feature the irregular, blotchy patterns also visible in the final model.
It should be noted that these issues may have been mitigated with additional training time, as GANs can continue to improve if the loss is stable and image quality steadily increases. Both properties were observed in figures 5 and 6/7, but but the experiment was limited by the prior allocated training time. Furthermore, the low-resolution GAN was improving simultaneously at a much faster pace, which highlights the practical challenge of training high-resolution GANs.

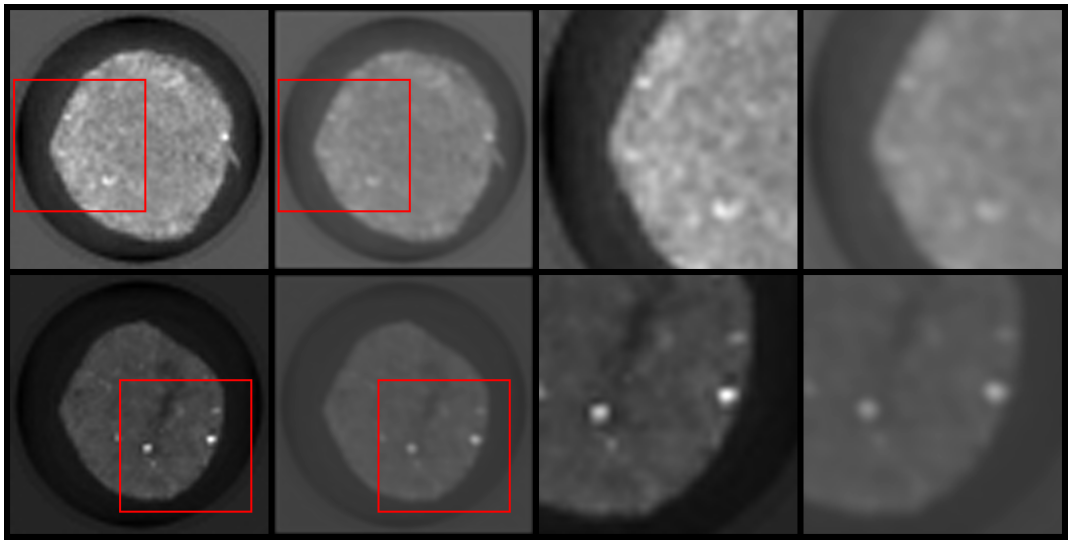
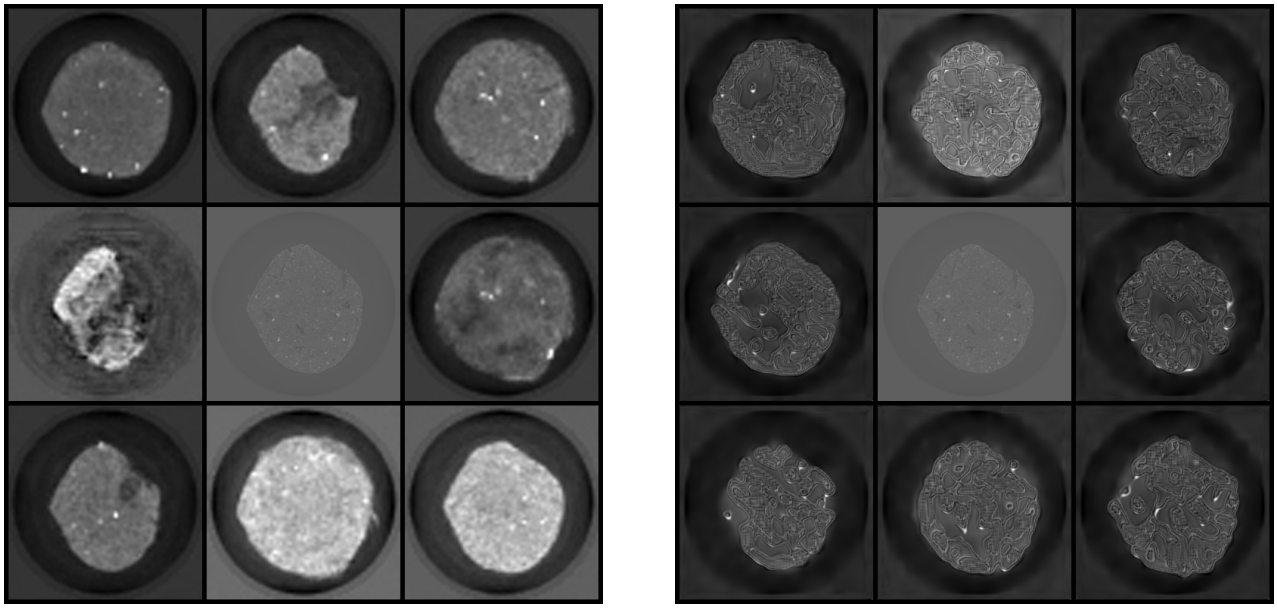


Figure 9: A figure showing the effects of SRCNN on the low-resolution GAN-generated images. Both rows correspond to the same generated sample, with the first and third columns corresponding to slices that were only upscaled with bicubic interpolation, and the second and fourth columns corresponding to the same slices after they were refined by SRCNN. The first two columns contain the entire slice, with the sections highlighted in red being enlarged in the latter two columns.

SRCNN was trained on the low- and high-resolution shale slices for 355 epochs, with the best performance achieved at epoch 212. While this may appear inconsistent with the early stopping criterion of 20 epochs, it is important to note that the best model selection was based on PSNR and SSIM, whereas early stopping was triggered by the MSE.

Visual inspection of figure 9 indicates a noticeable effect by SRCNN on the images. The low-resolution GAN produced slices with exaggerated contrast, seen in the strong distinction between lighter and darker regions, which deviates from the real dataset (see 3. SRCNN partially corrects this by restoring a weaker contrast, more closely resembling that of the original shale slices.
In addition to contrast correction, SRCNN reduces a portion of the noise present in the GAN-generated samples. However, this improvement is accompanied with increased blurring, diminishing the visibility of the fine-grained structures. As a result, while SRCNN improves global intensity characteristics and visual consistency with the real data, it simultaneously suppresses the small-grain details. Overall, the positive and negative effects balance each other, resulting in a relatively neutral impact on the perceived image quality.



(a) Low-resolution GAN

(b) High-resolution GAN

Figure 10: Two figures showing a number of slices generated by the low- and high-resolution GANs. Both have 8 generated slices surrounding the same real slice used as comparison in figure 8. The low-resolution slices have been upscaled with simple bicubic interpolation, and were then refined with SRCNN trained on the real dataset. The high-resolution slices were obtained directly from the high-resolution GAN. All images were hand-picked to show the diversity of slices present in the

Figure 10 presents the final outputs of both GAN pipelines. Eight representative slices were selected from the 1000 generated samples at the end of training for each model, chosen to illustrate the diversity of the generated images. The low-resolution GAN outputs (left) were first upscaled to 256×256 with bicubic interpolation and subsequently refined using SRCNN, whereas the high-resolution outputs (right) are shown directly at native resolution. A real slice from the dataset is included in the middle for reference.

As obvserved earlier in figure 5, the high-resolution GAN produces images that are predominantly blurry, with sharp details largely confined to prominent inkblot-like structures. These structures appear consistently across generated samples, but are not present in the original dataset. This represents a significant deviation from the true shale morphology.

After upsampling and SRCNN refinement, the low-resolution GAN outputs retain global structure of the shale samples across most slices and generally contain less blur than the high-resolution GAN outputs. While occasional failures occur, such as the noticeably lacking middle-left slice, these are relatively rare. The refined images are successful in reproducing mineral-dense sections, seen as larger bright specks, and preserve structural variations such as the feature absence around air pockets and edges of the shale sample, present in the real data. However, finer structural detail remains poorly reconstructed, with the shale appearing noisy outside of the earlier mentioned features, contrary to the smoother texture of the original slices.

Overall, both models succeeded in capturing the large-scale structure of the shale samples, but each also introduced artifacts. The high-resolution GAN exhibits more severe and systematic artifacts, whereas the RCNN-enhanced low-resolution GAN produces fewer, though still noticeable,

degradations, primarily in the form of noise leading to loss of some finer details.

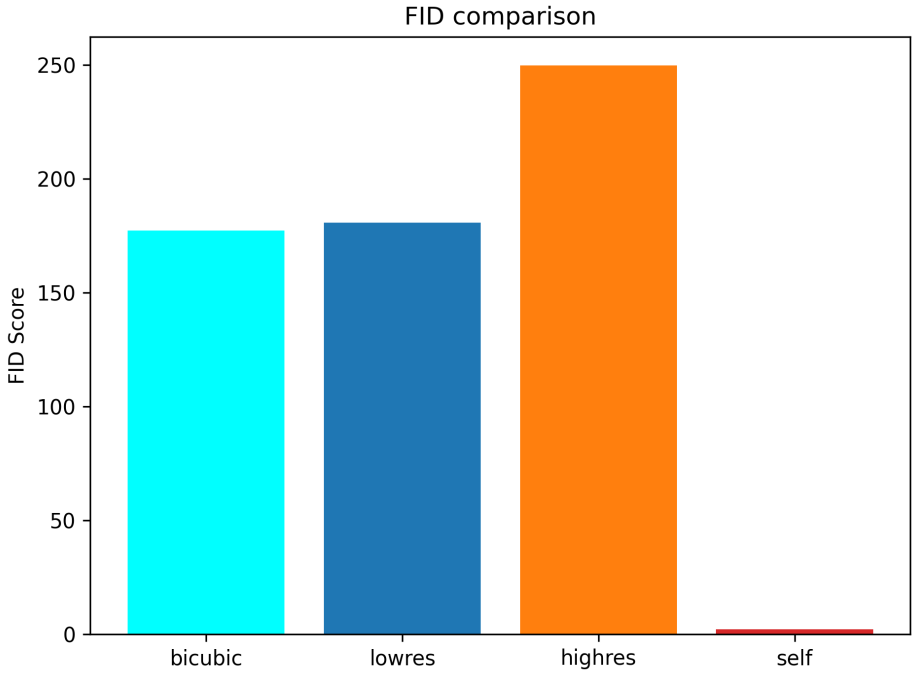


Figure 11: A bar plot showing the FID [HRUN18] for the low-resolution GAN-generated images that have only been upscaled with bicubic interpolation, the SRCNN-refined low-resolution GAN-generated images, the high-resolution GAN-generated images and two equal splits of the real data. The lower the FID is, the more accurate the compared dataset is to the real dataset.

The FID computed between two random splits of the real dataset resulted in a score of 2.1400. This low value is expected given the relatively limited variation between slices in the shale dataset, indicating the low spread of the real data distribution.
Both GANs had substantially higher FID scores, signifying that their p_g remains far from p_{data}. When comparing the two strategies, the low-resolution GAN achieved a noticeable lower FID of 180.8249, while the high-resolution GAN had an FID of 249.9273. This suggests that the low-resolution pipeline better captures the properties of the real dataset.
Interestingly, refining the low-resolution GAN outputs with SRCNN did not improve the SRCNN score. After applying only bicubic interpolation, the low-resolution GAN had a FID of 177.2358, which increased to 180.8249 after SRCNN refinement: an increase of approximately 2.03%. This indicates the SRCNN had little positive effect on the statistical similarity of the low-resolution GAN-generated images, and may even have slightly degraded it.
While this increase is small, it aligns with the qualitative observations discussed earlier with figure 9, where SRCNN appeared to correct contrast inconsistencies but did not recover fine structural detail. These results suggest that SRCNN primarily altered low-level image statistics rather than improving on retrieving the small-grain details.

While SRCNN only marginally improved the low-resolution GAN pipeline, the results nevertheless indicate that low-resolution GANs, even when upscaled with simple bicubic interpolation, can outperform high-resolution GANs in certain contexts. Across both qualitative and quantitave evaluation, the low-resolution GAN produced more consistent and realistic outputs than the high-resolution GAN. These findings suggest that increasing the output resolution alone does not necessarily lead to increased performance of a GAN.
Future work could explore alternative combinations of GAN architectures and upscaling techniques, as this thesis only investigated WGAN-GP with and without SRCNN. Other GAN variants or super-resolution methods, whether adaptive or non-adaptive, may prove more effective. Additionally, further refinement is possible through systemic hyperparameter optimization, which was outside the scope of this study.

7 Conclusion

In this thesis, we examined Generative Adversarial Networks (GANs) as a powerful approach of image generation and their applications in tomography. Since GANs often struggle to accurately reproduce small-grain details, particularly at higher resolutions, we explored an alternative strategy: generating images at low resolution and subsequently making use of post-hoc upscaling, rather than training a GAN to directly generate high-resolution images.
To investigate this approach, two implementations of WGAN-GP were developed. One model was trained to generate images at a resolution of 256×256, while the other was trained on low-resolution images of 64×64. Both models were based on existing implementations from a publicly available GitHub repository [LN]. The low-resolution GAN-generated images were first upscaled using bicubic interpolation, and then refined with SRCNN; a convolutional neural network that learns a mapping from real low-resolution data to their real high-resolution counterparts. SRCNN was likewise sourced from an existing implementation [Lor].
All models were trained on the *Round_Robin* dataset from Tomobank [GCXJ14], which consists of CT scans of shale samples from the North Sea. This dataset was selected because the fine-grained mineral structures present in shale samples provide a suitable test case for evaluating the ability of the GANs to preserve detail in synthetic CT scans.

The primary research question adressed in this thesis was:**How effective is post-hoc upscaling at increasing the quality and resolution of low-resolution CT scans generated by GANs?** The results indicate that SRCNN primarily improved low-level image characteristics, such as contrast, rather than producing substantial improvements in statistical similarity or visual realism. In fact, low-resolution GAN outputs upscaled using bicubic interpolation alone performed comparably to those that were additionally refined using SRCNN. In both cases, however, the low-resolution GAN pipeline outperformed the high-resolution GAN in this experimental setting.
A secondary question concerned the introduction of artifacts: **How does upscaling affect the introduction of artifacts in synthetic CT scans?** Both approaches introduced artifacts into the generated data, but the low-resolution GAN exhibited fewer and less severe artifacts than the high-resolution GAN. Artifacts in the low-resolution pipeline were generally confined to noise within the shale sample itself, whereas the high-resolution GAN introduced more pronounced and widespread artifacts across all generated images.

Finally, this thesis posed a final question: **Does the combination of low-resolution GANs and upscaling provide a favorable trade-off between image quality and computational efficiency compared to training high-resolution GANs directly?** The results suggest that such a trade-off is indeed favorable under constrained training conditions. Within a training time of 4 hours per model, the low-resolution GAN consistently outperformed the high-resolution GAN, indicating its potential suitability for similar applications.

Nevertheless, a key limitation of this study is the relatively short training duration. It is plausible that the high-resolution GAN would benefit from longer training times. Future work could therefore investigate extending the duration of training, alternative GAN architectures, or different super-resolution techniques.

References

[ACB17] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *Courant Institute of Mathematical Sciences, Facebook AI Research*, 2017.

[CLHT15] Dong C, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks, 2015.

[fda20] What is computed tomography? *U.S. Food and Drug Administration*, 2020.

[GAA+17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *Montreal Institute for Learning Algorithms*, 2017. https://arxiv.org/abs/1704.00028.

[GCXJ14] D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen. Tomopy: a framework for the analysis of synchrotron tomographic data, 2014. https://tomobank.readthedocs.io/en/latest/index.html.

[GPAM+14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Cornell University*, 2014.

[Har17] L. Hardesty. Explained: Neural networks. *MIT News Office*, 2017. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[HRUN18] M. Heusel, H. Ramsauer, T. Unterthiner, and B. Nessler. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *LIT AI Lab Institute of Bioinformatics*, 2018.

[IBM] IBM. What is gradient descent? *https://www.ibm.com/think/topics/gradient-descent*.

[KLA19] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[lD25] Scikit learn Developers. 1.17.1. multi-layer perceptron. *https://scikit-learn.org/stable/modules/neural_networks_supervised.html*, 2007-2025.

[LLWT15] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[LN] E. Linder-Norén. Pytorch-gan: Pytorch implementations of generative adversarial networks. https://github.com/eriklindernoren/PyTorch-GAN.

[Lor] Lornatang. Srcnn-pytorch. https://github.com/Lornatang/SRCNN-PyTorch.

[MW] Merriam-Webster. minimax. https://www.merriam-webster.com/dictionary/minimax.

[PV15] P. Parsania and P. Virparia. A review: Image interpolation techniques for image scaling. *International Journal of Innovative Research in Computer and Communication Engineering*, 02:7409–7414, 2015.

[PWH21] A. Pesaranghader, Y. Wang, and M. Havaei. Ct-sgan: Computed tomography synthesis gan. *Lecture Notes in Computer Science*, 13003, 2021.

[RDS+15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[SAH10] J. M. Susskind, A. K. Anderson, and G. E. Hinton. The toronto face dataset, 2010.

[Smo] A. Smorkalov. opencv-python. https://github.com/opencv/opencv-python.

[vKD05] G. van Kaick and S. Delorme. Computed tomography in various fields outside medicine. *Springer*, 15:74–81, 2005.

[WYW+18] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. Change Loy, Y. Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.