Universiteit Leiden
The Netherlands

Bachelor Data Science & Artificial Intelligence

An analysis of Deep Learning algorithms for producing training data focused on chest X-ray images.

Sjouk Ketwaru

Supervisors:
Joost Batenburg & Serban Vadineanu

BACHELOR THESIS

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

22/02/2024

Abstract

The scarcity of data is a well-known problem in machine learning. This thesis will investigate various models to generate chest X-rays suitable for machine learning tasks. Four models are examined: Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Diffusion Models, and Autoregressive Models. The study focuses on their architectures, mechanisms, and effectiveness in generating realistic images.

The research involves implementing and comparing these models based on their performance in generating chest X-rays. Different settings and their performance are compared for each model. A discriminator and a classifier are used to compare the models. The discriminator is trained in a GAN, and the classifier is a convolutional neural network.

The results indicate that Diffusion Models come closest to the NIH chest X-ray dataset, followed by GANs, with VAEs in third place. The Autoregressive Model was not tuned well enough to provide competitive results. This study highlights the strengths and limitations of each model in creating data and their applications in medical imaging. It provides valuable insights into the effectiveness of various generative models for augmenting medical imaging datasets, potentially improving the training of machine learning algorithms in healthcare.

Contents

1 Introduction

X-rays allow humans to examine the inside of a human chest without cutting open a person entirely. An X-ray can help detect issues mainly with the lungs, heart outline, and bone structure. However, doctors can find detecting anomalies in this part of the human body challenging and time-consuming. Nowadays, many algorithms can try to detect these anomalies in the human body. These algorithms do so by trying to distinguish between a chest X-ray of a good-functioning human and a chest X-ray of a not-good-functioning human organ. However, identifying the X-rays belonging to non-healthy persons by solely observing the images can be difficult for the algorithm. Therefore, this thesis will utilize some newer techniques focused on machine and deep learning. These methods have been gaining popularity, including in detecting anomalies [PSCVDH21].

Machine learning can be seen as a type of algorithm that learns patterns from data and tries to make predictions. So Machine learning is rather a focus on learning on the given data rather than explicitly programming for a specific task. Deep learning on the other hand is a subset of machine learning. Deep learning is focused on using neural networks with many layers. The architecture is thus composed of layers of nodes which tend to model how a normal neuron in a human brain functions. Each deeper layer extracts more abstract features from the data.

One of the problems of these algorithms is that they need a lot of data to function properly. Sometimes this data is not available, say a rare disease. In those cases, producing synthetic training data using algorithms can help solve a lot of issues. In other words, these algorithms have the task of generating images. And that is exactly what this thesis focuses on. The idea is to have a dataset and try to produce images similar to that dataset so that another machine learning algorithm can learn from it. This thesis focuses on the following algorithms: Generative Adversarial Networks, Variational Autoencoders, Diffusion models and Autoregressive models. The intention is to also compare these models with the discriminator from the Generative Adversarial Network and subsequently train a discriminator as a normal classifier. Then the idea is to completely compare all of these results.

Different machine learning algorithms can be utilized to use the generated data. One such technique is an autoencoder network, which encodes unlabelled data to the latent space (a lower-dimensional space than the input data) and reconstructs the data based on its representation in the latent space. This algorithm, in particular, has been extensively used in different projects [CPP+20].

For the autoencoder network, the best way to train for anomalies is by simply training it solely on X-rays where the patient is healthy. Training in this way ensures that when the network encounters samples that have particular pathologies, the network will give a poor reconstruction, as the autoencoder did not learn an image that has a defect. Since the network will try to classify the difference between the input and output, the distance between the defective image and a non-defective image should be quite significant (on a per pixel bases as stated in) [CPP+20]. To automate this process of detecting the difference between a defective and non-defective image, a Convolutional neural network can be used to predict the probability of an illness or defect in the image given to the autoencoder [PLP+22].

There have also been instances where only convolutional neural networks are used and no in-

stance of an autoencoder [RIZ+17]. This was based on a model called CheXNet that consists of 121 layers trained on the CHestX-ray 14 dataset. The weights of this model are from a pre-trained model on ImageNet. The network utilized the optimizer Adam, a batch size of sixteen, and a learning rate of 1e-3. This model did perform better than a radiologist in recognizing Pneumonia on the X-ray images [RIZ+17].

Another method to detect defective samples (samples that have particular pathologies) is by using the confidence-aware anomaly detection model. This model is made up of an extractor of features, a detector for anomalies, and a module that predicts the confidence that an image is an anomaly. If the score from the detection module is significant enough or the module that predicts the confidence level returns a small enough score then the input will be seen as an anomaly case. The advantage of utilizing this approach and not the approach of binary classification is that there is no need to model individual viral pneumonia classes, instead the algorithm attempts to detect any case of viral pneumonia [unk21].

An additional viable approach would be to utilise a model called an autoencoder with pixel-wise uncertainty prediction. This model does not only reconstruct a distribution (like the variational autoencoder network above did). It also estimates how uncertain the reconstruction is for each pixel, and with that attempts to enhance the defect detection even further. As described in paper [MXW+20], the reconstruction uncertainty is lower in the region around the lung and because of that, these scores can be used to detect anomalies. Due to X-rays mainly being used in the region around the lung area.

An alternative to the above-mentioned ways to find defects in an X-ray image is using a Generative Adversarial One-Class Classifier. The following paper [TTH+19], describes how three networks, with input the chest X-ray images, compete and collaborate with one another to reconstruct the incoming image. (just like in the Generative Adversarial Network). If the reconstructed image does not look at least equivalent to the input image, thus the reconstruction was poor because the input image does not look like images used in the training set, then the chest X-ray image should have some defects.

In this thesis we use the NIH Chest X-ray dataset, This dataset is used because of its large number of images and the accuracy of the corresponding annotations. It boasts about 112,120 X-ray images sampled from 30,805 unique patients. These are automatically labelled with a 90 percent accuracy by an NLP algorithm created by the National Institute of Health in the United States of America. However, as can be seen from figure 1, the diseases are unevenly distributed throughout the dataset and there are not a lot of images for the diseases. For those reasons, focusing solely on generating an X-ray image and not on particular diseases seems better. Figure 2 shows the visual representation of the data. To address the research objectives, this thesis aims to answer the following main research question:
"Which algorithm (Generative Adversarial Networks, Variational Autoencoders, Diffusion models, or Autoregressive models) generates chest X-ray images with the highest fidelity for diagnostic purposes?" To further explore this main question, several sub-questions will be investigated:

1. How well is the Generative Adversarial Network discriminator at checking how well the other

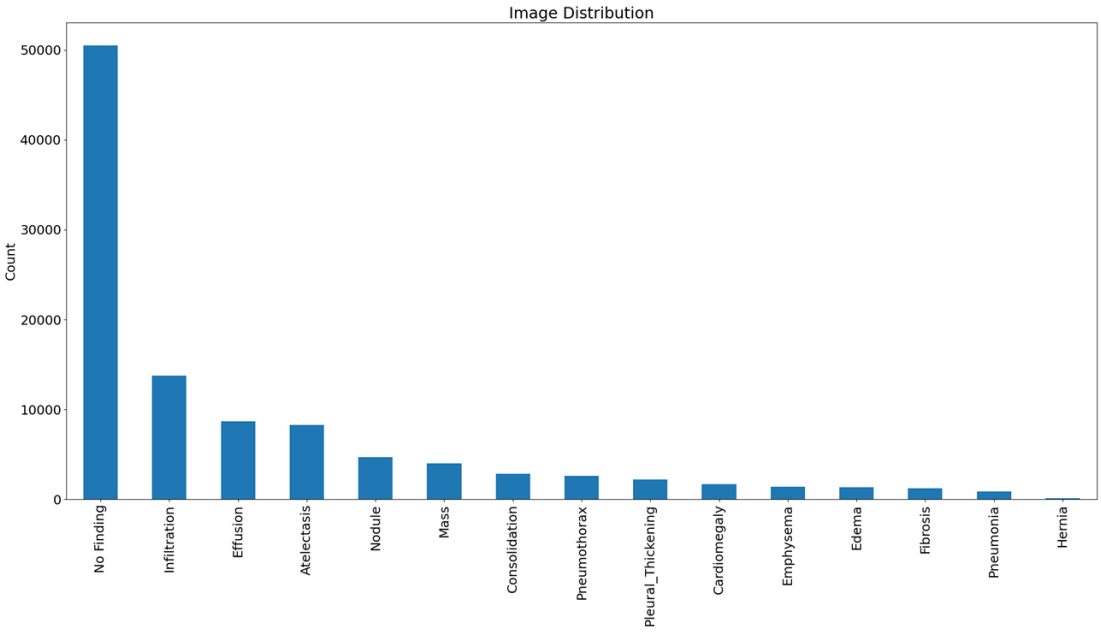

Figure 1: The distribution of the number of images in the NIH chest x-ray dataset. From both the train and test datasets together. No finding means the patient had no illnesses and is thus the control group. The other images are all illnesses.

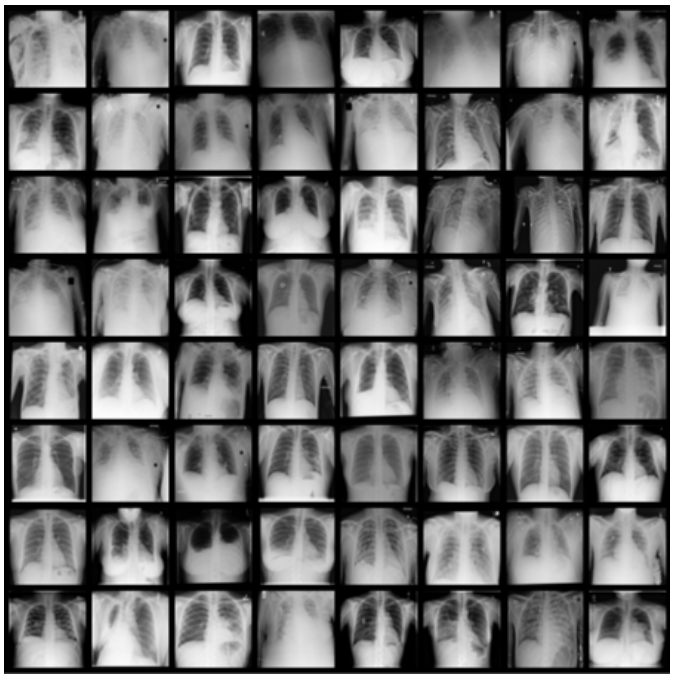

Figure 2: A sample of chest X-ray images from the NIH Chest X-ray dataset.

models are performing at creating a chest X-ray image?

2. How do the algorithms' generated images (VAE, autoregressive, GAN, diffusion model) compare to each other through the human eye when compared to the NIH chest X-ray dataset?

3. How do the algorithms' (VAE, autoregressive, GAN, diffusion model) compare to each other through the discriminator of the GAN?

4. How do the algorithms' (VAE, autoregressive, GAN, diffusion model) compare to each other through a classifier?

5. Which hyperparameter setting produces a result that is the most similar to the X-ray images on each algorithm?

6. Which of the algorithms with their optimal hyperparameter settings produces images that looks the most like X-ray images from the NIH chest X-ray dataset?

7. Which hyperparameter settings of a GAN (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?

8. How do the baseline GAN generated images compare to the NIH X-ray dataset images according to the human eye?

9. For which epoch are the GAN generated images closest to the NIH chest X-ray dataset?

10. For which learning rate is the GAN producing X-ray images that are closest to the NIH chest X-ray dataset?

11. For which image size are the GAN generated images closest to the NIH chest X-ray dataset?

12. For which hidden dimension size are the GAN generated images closest to the NIH chest X-ray dataset?

13. Which hyperparameter settings of a VAE (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?

14. How do the baseline VAE generated images compare to the NIH X-ray dataset images according to the human eye?

15. For which epoch are the VAE generated images closest to the NIH chest X-ray dataset?

16. For which learning rate is the VAE producing X-ray images that are closest to the NIH chest X-ray dataset?

17. For which image size are the VAE generated images closest to the NIH chest X-ray dataset?

18. For which hidden dimension size are the VAE generated images closest to the NIH chest X-ray dataset?

19. Which hyperparameter settings of an autoregressive model (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?

20. How do the baseline generated images by the autoregressive model compare to the NIH X-ray dataset images according to the human eye?

21. For which epoch are the generated images of the autoregressive model closest to the NIH chest X-ray dataset?

22. For which learning rate is the autoregressive model producing X-ray images that are closest to the NIH chest X-ray dataset?

23. For which image size are the images generated by the autoregressive model closest to the NIH chest X-ray dataset?

24. Which hyperparameter settings of a diffusion model (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?

25. How do the baseline generated images by the diffusion model compare to the NIH X-ray dataset images according to the human eye?

26. For which epoch are the generated images by the diffusion model closest to the NIH chest X-ray dataset?

27. For which learning rate is the diffusion model producing X-ray images that are closest to the NIH chest X-ray dataset?

28. For which image size are the images generated by the diffusion model closest to the NIH chest X-ray dataset?

By investigating these questions, this thesis seeks to provide a comprehensive comparison of different generative models in the context of medical image generation, thereby contributing valuable insights to the field of medical image analysis and synthetic data generation.
This thesis was completed as part of the Bachelor's program in Data Science and Artificial Intelligence at LIACS, under the supervision of Prof. dr. Joost Batenburg and Serban Vadineanu.

2 Related Work

2.1 Machine learning algorithms for generating data

A problem in machine learning is that models sometimes do not get enough data to train on. This problem results in a worse machine learning model than if they had enough data. This mostly applies to more specific cases like specific illnesses, which do not have a lot of data on them yet.
This thesis aims to figure out which model generates chest X-rays as suitable for machine learning tasks as possible. First, this thesis will highlight the different algorithms for generating X-ray data. Then, the machine learning methods for recognizing illnesses will be highlighted. There are multiple

models suitable for the image generation this thesis aims for:

1. Generative Adversarial Networks (GAN)
   As described in [Bha18], the first generative adversarial networks were made using a fully connected neural network. A fully connected neural network is a network where each perceptron has a linear transformation to the input through the weights. In favour of more stability most of the GANs have eliminated the use of fully connected layers [ACB17] [GAA+17] [MKKY18] [RMC15]. In favour of convolutional GANs, this version relies on convolutional neural networks. Convolutional neural networks have Convolutional layers that take the dot product from a part of the input and the kernel. The kernel is like a filter that slides over the image, highlighting patterns by calculating the dot product. Other than the kernel taking the dot product, another difference is that not every node is connected. This construction creates more flexible learning. The weights in each layer are also much smaller, which helps in vision tasks and other high-dimensional inputs.

2. Variational Autoencoders
   The modern VAE (variational autoencoder) version is introduced in [Kin13]. It shows a way to efficiently perform posterior inference using neural networks. Posterior inference is the process of updating the probability distribution of a model's parameters based on observed data. In 2014, this paper [RMW14] also demonstrated the effectiveness of the methods used in a variational autoencoder for generative models (like the reparameterization trick shown in section 2.3.3). There has also been some research to improve the variational autoencoders in general. VampPrior [TW17] uses a mixture of posteriors to enable a more flexible prior distribution. Or Hierarchical VAEs [ZSE17] that use hierarchical latent representations to make more complex data distributions possible. VAE's have also been shown to be used for image generation but only on simple datasets like the MINST dataset (a simple standard dataset used for machine learning tasks, the dataset consists of handwritten numbers).

3. Diffusion
   The diffusion model [SD15] and variants of it have seen good results thus far [Ho20, Nic21, Son20, Rom21, Rua22, Din23a, Din23b, Hua23b], in terms of image [Ho20, Son20] and video [Ho22, Yin23] generation and much more [Rom21, Rua22, Met22]. The improvements which led to current version of Diffusion (as of 26-06-2024) started with DDPM [Ho20] which puts noise onto a picture, and learns to restore photos gradually. Then DDIM [Son20] came around, which mainly improved the pace of image generation by removing some steps. After that, the conditional latent diffusion [Rom21] representation came around. This image generation technique focuses on having multiple conditions like, images and texts and improving the inference swiftness of the model overall. After all those generations, there was finally the current day Diffusion [Rom21]. A model capable of generating lifelike photos. Given any text, its upgraded versions [Zha23, Hua23a, Mou23] have been utilised in many AI-generational products like Midjourney. [Du23]

4. Autoregressive Models

   Autoregressive models are likelihood models that model the statistical distribution of data by

estimating the substance. The models come close to the maximum likelihood:

$$"\theta^* = \arg\max_{\theta} E_{x \sim pdata(x)}[\log p_\theta(x)]"$$

[Dal19]. The models learn each probability for the conditions. This characteristic together with the maximum likelihood formula helps to get a negative log-likelihood (NLL) which is superior to other methods such as VAEs [Kin13] or flow models [Din16, Kin18]. PixelCNN [vdOKK16] has a NLL of 3.00 bits/dim on a dataset called CIFAR-10 and was the first to create a convolutional autoregressive model architecture. The CIFAR-10 dataset is a dataset commonly used for machine learning algorithms. The data has 10 classes: aeroplanes, automobiles, birds, kittens, deer, pups, frogs, stallions, ships and trucks. After a few years, further modifications [VDO16, Par18, Sal17, Che17] made the score lower to 2.85 bits/dim, which is the best NLL score to date on CIFAR-10.

2.2 Machine learning algorithms for utilizing data

As becomes clear from how these models work, there are significant differences between how they operate, making them strong candidates to be compared to one another.
The data generated by the algorithms used in this thesis could be used by many different machine learning algorithms. The most important ones will be highlighted here:

1. Convolutional autoencoder model
   Deep learning has been touted as a promising way to automatically classify and detect anomalies in chest X-ray images these have some good results thus far [WPL+17], [YYH18], [TWH+18]. The definition of anomaly detection is as follows: "Anomaly detection is the task of identifying unusual samples from the majority of the data" [unk21]. The inner workings of an autoencoder have been explained in the introduction. The algorithm works by encoding unlabelled data to the latent space (lower dimensional space than the input data). After this process the data gets reconstructed based on the representation in the latent space. If that representation is far from the input, then there must be some form of an anomaly. In this way the model detects anomalies.

2. Confidence aware anomaly detection
   Confidence aware anomaly detection has a feature extractor, a detector for anomalies and a module that predicts the confidence that an image is an anomaly. If the score from the detection module is significant enough or the module that predicts the confidence level returns a small enough score then the input will be seen as an anomaly case. Usually, defect detection focuses on a specific type "kernel-based one-class classification" [unk21], which includes other methods like the One Class SVM [SPS+01]. These methods attempt to utilize a hyperplane to separate the defects from the good samples. The above-described methods have some issues, like the curse of dimensionality. However, deep learning automatically learns the correct feature representation from the data. There have already been quite some efforts to get the advantages of deep learning to the field of anomalies. A deep SVDD model has for example, the deep neural network attempts to minimize a hypersphere. A deep SVDD model has been presented by [RVG+18]. Other research has been done that focuses on unsupervised detection, which works with generative adversarial networks to help detect defects that are unknown

from OCT images [SSW+19]. These methods are superior to the standard methods like the kernel based one.

3. Autoencoder with pixel-wise uncertainty prediction
   As explained above this model works by not only reconstructing a distribution (like the variational autoencoder network). But it also estimates how uncertain the reconstruction is for each pixel, and with that attempts to enhance the defect detection even further. As described earlier, the reconstruction uncertainty is lower in the region around the lung. There are often large errors around different regions (say going from the lung to the background), and this problem may result in more false positives (in other words, a not-ill patient can be detected as ill.). A solution to this issue is suppressing the reconstruction errors in this detection. In the following paper, [MXW+20], a probabilistic approach has been used to lower the importance of the regions with more reconstruction errors.

4. Generative adversarial one-class learning (specific for chest-x rays)
   This network in specific has been inspired by one class classification [MH96], [SKFA18]. and is quite similar to a generative adversarial network [GPAM+14]. These networks attempt to put the data in different categories from a set of images only containing that specific category. As explained earlier, there are three networks in this specific setup, with the chest X-ray images as input. The three different architectures are a CNN discriminator, an encoder and a U-Net autoencoder. The networks are competing and collaborating with one and another to reconstruct the incoming image. (just like in the autoencoder). If the reconstructed image is not equivalent to the input image, then the reconstruction was poor. The reconstruction is poor because the received input image does not look like the images used in the training set.

2.3 The architectures

This subsection will provide a more in-depth description of the different architectures used in this project.

2.3.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) can be characterised by training two neural networks in competition with each other. An easy way to look at how these compare is that one neural network is the image creator, and the other one is the image expert. The creator creates an image, and the expert judges how good the image is. The creator is the generator, $\mathcal{G}$; The expert is known as the discriminator $\mathcal{D}$, and aims to tell the image from the dataset and the generated image from the generator apart. Both $\mathcal{G}$ and $\mathcal{D}$ are trained simultaneously. The generator receives no images; The generator learns through the feedback of the discriminator. [Bha18] GANs have shown to be very capable in the field of image generation and are therefore, a must in this thesis. [HOT06]
In this study, the focus is mainly on the DCGAN (Deep Convolutional Generative Adversarial Network) (as described earlier in the related work section). DCGAN was first introduced in [RMC15]. The architecture is based on Convolutional GANs. A DCGAN goes one step further and also uses a strided convolutions. Normally, a convolution goes one step at a time. A strided convolution controls the step as it is moved over the pixels. Another feature of the DCGAN is the ReLU activation in

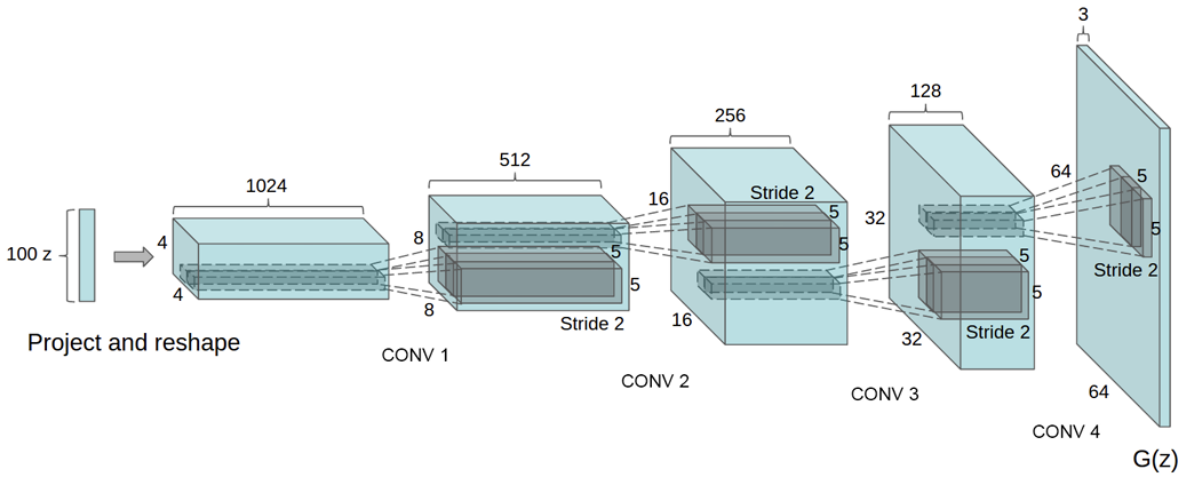

Figure 3: Explanation picture for the Generative Adversarial Network architecture. The boxes represent the Tensors.

the generator for all layers and the use of a LeakyReLU activation in the discriminator. The ReLU activation function follows the following function 1:

$$f(x) = \max(0, x) \tag{1}$$

where x is the input to the neuron, and f(x) the output of the ReLU function. The LeakyReLU has the following function 2:

$$f(x) = \max(0.01 * x, x) \tag{2}$$

where x is the input to the neuron, and f(x) the output of the ReLU function. A DCGAN also utilizes a batchnorm layer to normalize the output of the convolutional layers and with that attempts to stabilize learning. In the implementation used in this thesis, the LeakyReLU function was used in the generator to stabilise the GAN. The problem was that the GAN was quite unstable with the normal ReLU activation functions. The reason LeakyReLU was more stable is likely due to dead neurons, as when the input of a ReLU is negative, then the output of the ReLU is zero. If a neuron starts to give output zero, then it might continue to do so and with that stop the learning process. The final architecture looks quite similar to the one in [RMC15] as shown in figure 3. The generator ends with a tanh function (which is suitable for normalized image data), and the discriminator with a sigmoid function (which is suitable for checking if an input is real).
The tanh activation function is as follows:

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3}$$

where x is the input of the neuron, and $\sigma(x)$ is the output of the function. The sigmoid activation function is as follows:

$$\sigma(x) = \frac{e^x}{1 + e^x} \tag{4}$$

where x is the input of the neuron, and $\sigma(x)$ is the output of the function.

2.3.2 Diffusion models

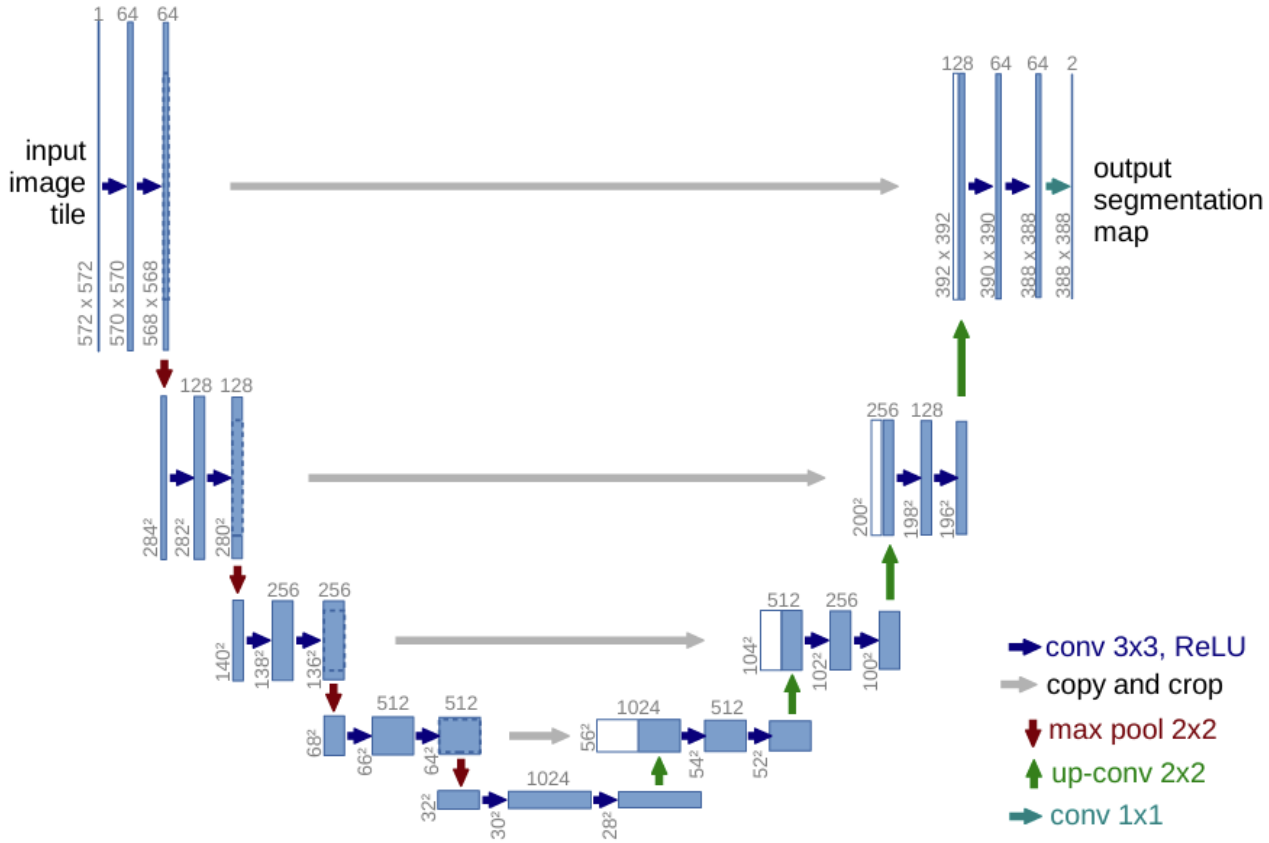

Figure 4: Explanation picture for the diffusion models. The blue lines represent the Tensors.

Diffusion models are generative models that were studied for their ability to generate high-quality images. This model functions by putting noise over a picture and afterwards learning to remove the noise step by step so that the original data gets recovered.

The first step in the diffusion process is the forward diffusion process. In this process, the model gradually adds noise until the image only consists of noise. The forward process usually only uses the Gaussian noise addition with the formula:

$$\varphi(Z) = \frac{1}{\sigma * \sqrt{2 * \pi}} e^{\frac{-(Z-\mu)^2}{(2\sigma^2)}} \tag{5}$$

Now the reverse diffusion process is considered. When a noisy image enters, the process will denoise it step by step until the original image is recovered. Denoising is often achieved by a model that tries to predict and remove noise. The U-Net model is usually used for this step. The distribution that is left when everything is denoised and the original image is back is the eventual distribution that is used in the image generation process.

The U-Net model was first proposed in [RFB15]. Here, the U-Net model is described in figure 4. First the image starts on the left side of the diagram (the encoder). In this example, the image has size 572 x 572 x 1. Following the blue arrow, the image goes through convolutional layers. Each step has two convolutions. Each convolution uses a 3 x 3 kernel, whereafter the ReLU activation follows. This process doubles the channels in the feature maps. The first convolution makes them 64, the next convolution makes them 128 and henceforth. The process of convolutional layers is needed because, it allows networks to extract important features out of input images and reduces complexity. The kernel is like a filter that slides over the image, highlighting patterns by calculating the dot product. The ReLU activation function makes the model non-linear and helps learn more complex functions.

Then there are two paths, the data gets transferred to both paths. One corresponds to the grey arrow, this path is also known as the skip connections. The second path is the contracting path, the red arrow. First the gray arrow path, the skip connections try to copy the feature map from the current contracting path and concatenate them with the upsampled version in the expansive path (the right of the diagram). This should ensure that the spatial information does not get lost, as every max pooling makes the image's resolution a tad lower.

The other option is the red arrow, this is the contracting path. Here, there are max-pooling layers. These layers reduce the spatial dimensions of the feature maps by half. This action is performed to capture more high-level features and helps reduce the complexity.

Now, onto the expansive path (also known as the decoder). This path reconstructs the lost spatial dimensions and produces a segmentation map. A segmentation map is a map where a part of the image is labelled. The first calculation on the decoder is the up-convolution layers. Every step up samples the feature maps by using a 2 x 2 transposed convolution. This upsampling technique will ensure that the spatial dimensions get doubled, 28 x 28 to 56 x 56 and henceforth. Like the convolutional layers described earlier in the contracting path, these layers involve two convolutions with a 3 x 3 kernel followed by ReLU activation.

2.3.3 Variational Autoencoders

Before explaining what a Variational Autoencoder is, the definition for an autoencoder is needed: "An autoencoder is a type of algorithm with the primary purpose of learning an "informative" representation of the data that can be used for different applications by learning to reconstruct a set of input observations well enough" [Mic22]. Based on the paper [Kin13], the variational autoencoder consists of an encoder and a decoder. First, the algorithm starts with an encoder that accepts the input and creates a latent feature representation (A lower-dimensional version of the original input). Finally, it ends in a decoder that tries to reconstruct the data back to the original input. The idea is that what comes out of the decoder should be as close as possible to the dataset given in the beginning (the input).

The variational Autoencoder (VAE) is an improved version of the autoencoder. Instead of creating a vector representation for the latent feature representation, the VAE puts the input in a distribution, and the decoder takes samples from that distribution and constructs an output. KL-divergence is

used to compare the two distributions.

The variational autoencoder also uses a reparameterization trick to get samples from the latent dimensions during training to help improve the model. First of all, the goal of the encoder is to compress the input to the latent space, generating a mean and log variance. It first has a fully connected layer that puts the input dimensions in the latent space. Other connections in the hidden space are also fully connected. The encoder also uses LeakyReLu layers between the hidden dimensions.

Next, a bit about the reparameterization trick. The idea behind it is instead of sampling from the distribution, the sample is taken from the normal distribution and then shifted towards the one from the encoder. The shifting is performed by the mean and variance that the encoder outputs. The formula used here is formula 6:

$$z = \mu + \sigma * \epsilon \tag{6}$$

with the μ meaning the mean of the latent space distribution, the σ the standard deviation and the ϵ is random noise sampled from a normal distribution.

Now, we will consider the decoder. The decoder aims to reconstruct the input from the latent space to the output. The decoder consists of a fully connected layer that expands the latent dimensions to the hidden ones. After those hidden dimensions, the output layer returns an image. Between all of these layers LeakyReLu is used. The output layer has used a sigmoid activation function to help normalize the image between 0 and 1.

2.3.4 Autoregressive models

Autoregressive models are likelihood models that model the statistical distribution of data by estimating the substance. The models try to come as close as possible to the maximum likelihood function:

$$"\theta^* = \arg\max_{\theta} E_{x \sim pdata(x)}[\log p_\theta(x)]"$$

[Dal19]. The models learn each probability for the conditions.

Autoregressive models were first introduced in [vdOKK16]. In this paper, the pixel CNN framework is described. The pixelCNN model tries to generate images pixel by pixel. The model tries to model the probability of all pixels sequentially, conditioning every pixel on the previously generated ones. In the implementation, typically Pixelconvlayers instead of the normal convolutional layers are used here. This custom layer adds a mask to ensure that each pixel is dependent only on the previous pixel.

The heart of this model is the residual block. The residual block first lets the input go into a 1x1 convolution with a ReLU activation function. The purpose of this convolution is to lower the depth of the channels not the spatial dimensions. Then the input goes through a pixel convolution layer. This layer does the same as earlier described, it makes sure that each pixel depends solely on the previous ones. Then, that input goes through another 1x1 convolutional layer with a ReLU activation function to restore the input back to its original dimensions. In between connections there are also once again skip connections. The idea behind these connections is to connect the

input tensor to the output tensor after the convolutional layers have been performed. These skip connections intend to lower the chance of a vanishing gradient. The vanishing gradient problem is a problem where the weights of a network get so small that it stops functioning properly. The weights of the network also can not get updated anymore due to them being so small.

3 Experiments

The purpose of the experiments is to answer the following research questions from earlier:

1. For each algorithm, which hyperparameter setting produces a result that is the most similar to the X-ray images in the dataset?

2. Which algorithm, with its optimal hyperparameter settings, produces images that most closely resemble the X-ray images?

To answer question one, the algorithms and their hyperparameter changes are compared (see section 3.2 to section 3.5). To answer question two the algorithms' optimal hyperparameters were selected. This information was necessary to answer the final question: "Which algorithm (Generative Adversarial Networks, Variational Autoencoders, Diffusion models, or Autoregressive models) generates chest X-ray images with the highest fidelity for diagnostic purposes?".

3.1 The testing systems

In the experiments two systems having the following specifications were used.
System 1:

1. CPU: Ryzen 7 7700X @ 4.5GHz (16 threads)

2. GPU: RTX 3070 TUF Gaming 8GB of VRAM

3. RAM: 32 GB

4. Motherboard: ROG STRIX B650E-F GAMING WIFI

5. PSU: Corsair RM650

System 2:

1. CPU: 24 Intel Xeon Silver 4214 cores @ 2.20GHz (48 threads)

2. GPU: RTX 3090 24GB of VRAM

3. RAM: 256 GB

The appropriate packages are available in the GitHub repository under requirements.txt and can be installed through the conda environment. In this particular case, miniconda was used to create the environments. Due to both systems being relatively low in GPU VRAM, the images were scaled down to 128, 64, and 32.

3.2 Generative Adversarial Networks

3.2.1 How the experiment is structured for Generative Adversarial Networks

The question which was attempted to be answered in this section was: "Which hyperparameter settings of a GAN (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?". To answer this question the hyperparameters have been tuned until an X-ray came out of it. First this model was tested with the original implementation highlighted in [RMC15]. This implementation was first tested on the MINST dataset. However, after testing this particular setup on the NIH-X-rays dataset it became clear that the model becomes unstable after a few epochs (1-5) and does not recover. The final baseline version used had a few tweaks to battle this instability. Firstly, the activation function has been switched from a ReLU function to a LeakyReLU with a negative slope of 0.2. LeakyReLU's may help because of their ability to avoid dead neurons also called the dying ReLU problem. The dying ReLU problem occurs when a large number of neurons output zero for all inputs, and with that, they become unusable for the model. Second of all, the data was normalized (between 0 and 1) and converted to greyscale (most other algorithms also had this change). This normalization ensures that the model only learns based on the data that it should utilize and makes the code more efficient. Lastly, the diversity loss has been halved to improve stability. Some other settings, like the initial weights and parameters have also been finetuned to get better performance. There were also attempts at: changing the learning rate of the discriminator and the generator, making sure that the discriminator learned slower than the generator and making the real and fake labels floats. Nevertheless, these changes did not improve the algorithms performance. For the specific implementation used in this thesis, the baseline had the following parameters:

1. Batch size: 128
2. Image channels: 1 (greyscale)
3. Image size: 64 x 64
4. Learning rate: 1e-4
5. CUDA: true
6. Hidden dimensions: 64
7. Kernel size: 4
8. Padding: 0
9. Seed: 0

During the production of the plots the model was in evaluation mode and the gradients were not calculated because torch.no_grad() was used. The backpropagation was also disabled during testing. These settings led to the most stable version of a GAN. Which is why this was the baseline option and thus answers the question:"Which hyperparameter settings of a GAN (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?"

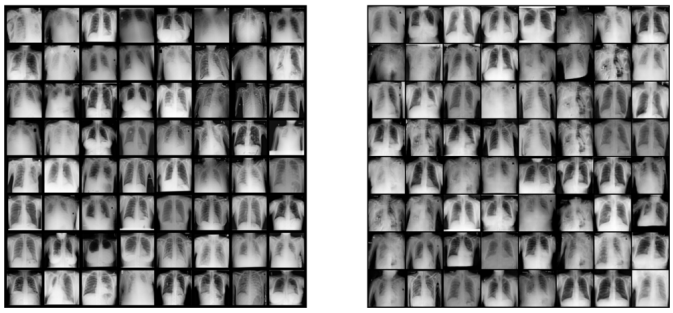

Figure 5: (left) pictures from the dataset itself to compare to the pictures on the right, (right) pictures generated by the Generative Adversarial Network after 50 epochs and a learning rate of 1e-4.

3.2.2 Visual comparison baseline

The question that was attempted to be answered in this subsection is: "How do the baseline GAN generated images compare to the NIH X-ray dataset images according to the human eye?". This can be achieved by generating the images and comparing them to the images in the dataset itself. Figure 5 shows on the left a picture from the dataset and on the right a generated picture by the GAN. The GAN was trained with 50 epochs, a learning rate of 1e-4, a hidden dimension size of 400 and an image size of 64 x 64, as described in the previous section. Looking at the pictures from figure 5, it seems that the GAN still has some small issues. This might mean that the program is overfitting, trying to find a way to overgeneralize what an X-ray of a human chest looks like (this could be true for either the discriminator or the generator or both). Another explanation could be that the GAN has not yet learned enough and is underfitting, as the images with problems seem to have scribbles. Most pictures do seem clear. Fifty epochs is taken as baseline, as fifty epochs was a common setting in most models and a recommended starting point. The algorithm seems quite stable with the current settings, as described in section 2.1.1. Regarding computational efficiency, the algorithm took about a day to train on system 1 while utilizing the GPU.
In conclusion, The images produced are already of quite high quality, however there were still some small issues likely due to overfitting or underfitting.

3.2.3 Epoch comparison

The question that was attempted to be answered in this subsection is: "For which epoch are the GAN generated images closest to the NIH chest X-ray dataset?". To answer the question the baseline algorithm is ran for 50 epochs. In figure 6 is shown how the discriminator loss changes at every epoch of the baseline GAN model described earlier. From the figure, it becomes clear that the model does improve over the epochs. However there are some limitations here as the discriminator is trained on samples generated by the GAN model. Because of this limitation, the discriminator might be biased towards the samples the GAN draws even if those do not represent the dataset. Figure 6 also shows that the discriminator is usually quite good at detecting the fake generated samples from the GAN. This might also be because the discriminator was trained with the generator and has already evaluated these samples. It also shows a high point and the 49th epoch, likely because of the balance between the discriminator and the generator.
In conclusion, 50 would be the amount of epochs for which the GAN generates images closest to

the NIH chest X-ray dataset.

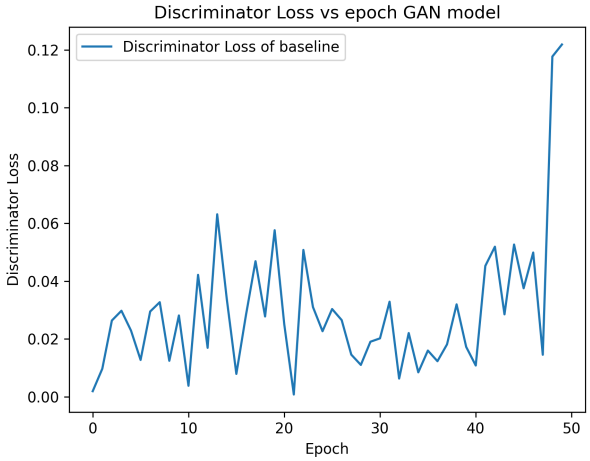

Figure 6: Here is a Generative Adversarial Network shown with the discriminator loss per epoch the network has a learning rate of 1e-4, after 50 epochs, and 64 x 64 as image size.

3.2.4 Learning rate comparison

The question that was attempted to be answered in this subsection is: "For which learning rate is the GAN producing X-ray images that are closest to the NIH chest X-ray dataset?". To answer the question the baseline GAN, a GAN with a higher learning rate (1e-3) and a lower learning rate (1e-5) are compared to one and another. Due to the algorithm's instability, the Generative

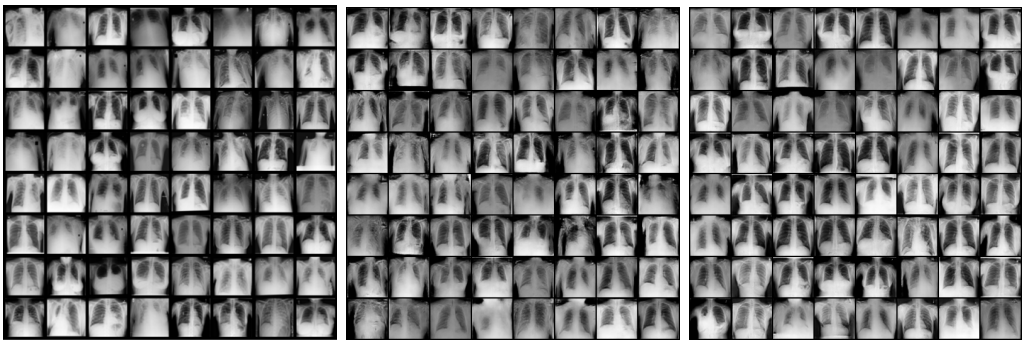

Figure 7: (left) pictures from the dataset itself to compare to the pictures on the right and middle, (middle) pictures generated by the Generative Adversarial Network after 50 epochs and a learning rate of 1e-5. (right) pictures generated by the Generative Adversarial Network after 50 epochs and a learning rate of 1e-3.(higher is better as the loss is higher if the discriminator has more trouble detecting the differences).

Adversarial Network has been tested for 50 epochs for 1e-5 as learning rate the result can be seen in figure 7. Sometimes, the model falls apart, meaning that the entire set of pictures becomes blurry, unclear, or has artefacts. An artefact is a picture where there are clear blocks of distortion or

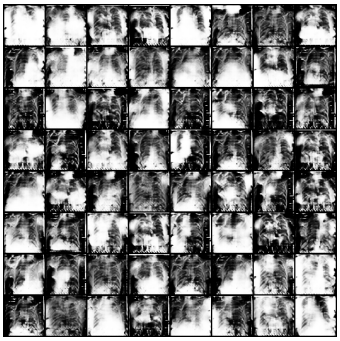

Figure 8: Picture of a Generative Adversarial Network that has fallen apart in epoch 50 with learning rate 1e-3 and has some artefacts.

off-color pixels (for example, figure 8). Sometimes the model recovers from these issues, In this case under the 50 epochs no model was having artefacts. Previous iterations did have some artefacts, the GAN with a learning rate of 1e-3 for an example. Initially, this model ran with the same hyperparameters as described in the baseline section, however the model became unstable and gave the result of figure 8. To counteract these issues, the number of hidden layers was increased for both the generator and the discriminator to 128 instead of 64 for the 1e-3 variant. This change may have increased performance due to a regularizing effect, which makes the model less prone to overfitting as the model is generalizing less. Some other settings were attempted to be changed like: loss function (Least squares loss and wasserstein), the betas of the optimizer, the optimizer (adamW instead of adam), changing the kernel size and increasing the diversity loss. However, these changes did not achieve a better result. The other learning rates still had the same settings as set in the baseline. In figure 9 the comparison between different settings of the GAN can be seen. It becomes clear that the baseline presents the highest scores out of all of them. The GAN model with a learning rate of 1e-5 is performing the least well, the reason for this might be that the lower learning rate made the model learn the data at a slower rate. The lower the learning rate, the lower the influence on the model's parameters, which, in this case, is the model's weight. The higher learning rate of 1e-3 might be performing better, however the model might be overshooting the more optimal solution that the baseline model finds. For those reasons, the baseline could be optimal as is visible because of its highest peak score in the graph. This result could be tainted as the discriminator has come from the baseline GAN.
In conclusion, the baseline outperforms all of the other learning rate settings. Likely due to the 1e-5 learning slower and the 1e-3 overshooting the optimal weights.

3.2.5 Image size comparison

The question that was attempted to be answered in this subsection is: "For which image size are the GAN generated images closest to the NIH chest X-ray dataset?". To answer the question the baseline GAN (64 x 64), a GAN with a image size (128 x 128) and a lower image size (32 x 32) are compared to one and another. Figure 10 shows 32 x 32 images generated by a GAN with a learning rate of 1e-4 after 50 epochs with 64 hidden layers. The images are a tad more blurry than the ones in the baseline, other then that there are not any major flaws visually. The same can not be said for figure 11. In this figure, a GAN generated 128x128 images with 64 hidden layers, a

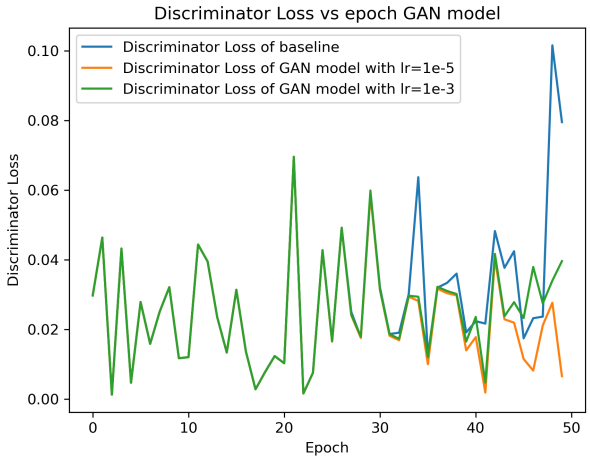

Figure 9: Discriminator loss after attempting to compare: the baseline, a learning rate of 1e-5 and a learning rate of 1e-3. These tests are on a Generative Adversarial Network after 50 epochs and an image size of 64 x 64.

learning rate of 1e-4 and after 50 epochs. Visually these images do not look like X-ray images. This model was very unstable and did not recover from its issues. In order to utilize a 128 x 128 image the GAN and discriminator needed an extra layer, this extra layer caused the model to be very unstable. To stabilize this GAN the following has been tried: changing the kernel size, changing the learning rate to 1e-5, changing the learning rate to 1e-6, giving the generator and the discriminator a different learning rate (1e-4 and 1e-5 respectively), making the gap between the two learning rates bigger (5e-4 and 1e-6 respectively), changing the input noise vector (both lower and higher has been tried), changing the number of hidden layers (128, 64, 32 have all been tried). However in almost all cases the model collapsed in the first two - three epochs. Yielding the result of figure 11. Figure 12 contains a graph which compares the discriminator loss per epoch per different image size setting. For each image size the discriminator is also trained on that image size. Looking at figure 12, it becomes clear that the GAN model with image size 128 had some spikes. During those spikes the image was somewhat visible (but also very noisy) which may have lead to higher scores from the discriminator. The discriminator might not be able to recognize the X-rays due to the noise in the generated images. Both the baseline and image size of 32 come out on top. For the comparison of the image size of 32 a discriminator for the image size of 32 is used which has one layer less than the 64 version. This layer change may have improved the model's performance, as there are fewer variables to tune.
In conclusion, the baseline and the 32 x 32 variants both created images close the dataset. However, in the exact loss numbers it became evident that the baseline outperformed the 32 x 32 variant.

3.2.6 Hidden dimension size comparison

The question that was attempted to be answered in this subsection is: "For which hidden dimension size are the GAN generated images closest to the NIH chest X-ray dataset?". To answer the question the baseline GAN (hidden dimension size 64), a GAN with a hidden dimension size 128 and a lower image size hidden dimension size 32 are compared to one and another. Figure 13 shows 64 x 64

Figure 10: (left) pictures from the dataset to compare to those on the right, (right) pictures generated by the Generative Adversarial Network after 50 epochs and an image size of 32 x 32.

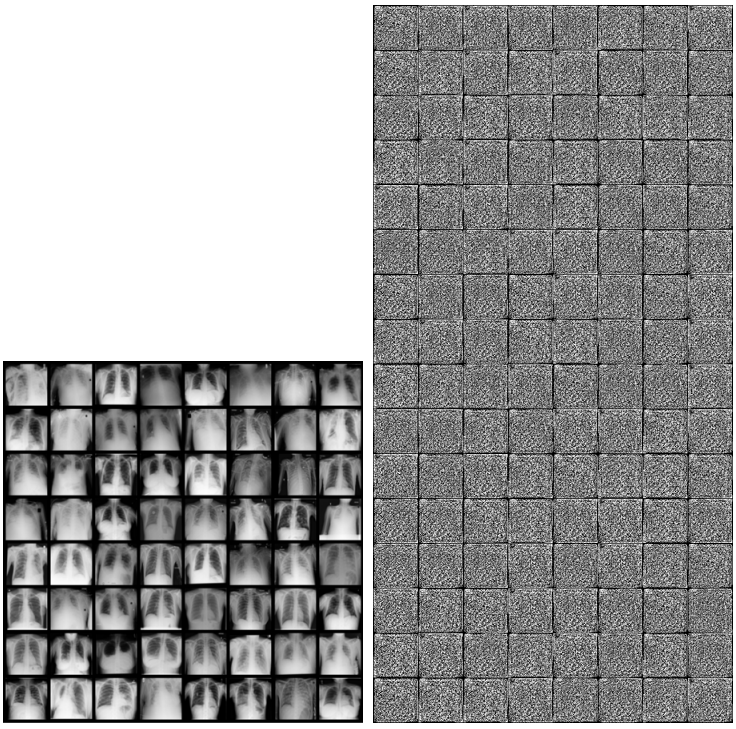

Figure 11: (left) pictures from the dataset to compare to those on the right, (right) pictures generated by the Generative Adversarial Network after 50 epochs and an image size of 128 x 128.

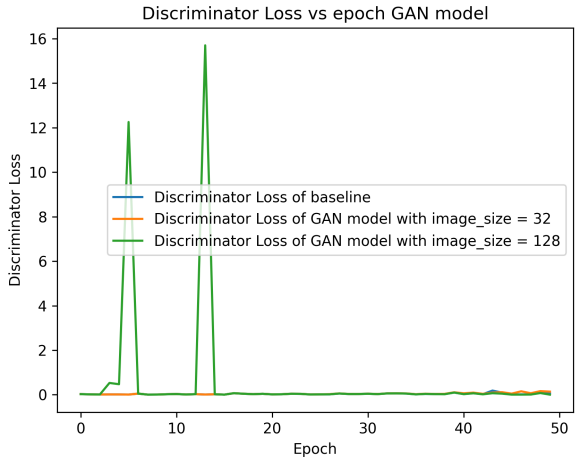

Figure 12: (left) pictures from the dataset to compare to those on the right, (right) pictures generated by the Generative Adversarial Network after 50 epochs and an image size of 32 x 32.

images generated by a GAN with a learning rate of 1e-4 after 50 epochs with 32 channels, one with 128 channels and an image from the dataset. Figure 14 shows the graph comparing discriminator loss of GAN model between the different channel sizes (the baseline performs as well as the channels = 32). In the visual comparison, it becomes clear that the model might overgeneralize in the hidden dimension is 32 version, as there are fewer channels, the model might not be able to extract the right amount of features due to there being less depth in the model. From the graph and the visual representation it becomes clear that the 128 channels might be a too complicated model to stabilize properly, giving the artefacts shown in the figure and the high fluctuations in the graph. The baseline performs the same as the channels of 128. In this implementation no changes were made to the baseline other than that the amount of channels changes.
In conclusion, the baseline model generates images that are closest to the dataset the model with 32 hidden dimensions might be too simple and the 128 hidden dimension version too complicated.

3.3 Variational Autoencoder

3.3.1 How the experiment is structured for the variational autoencoder

The question which was attempted to be answered in this section was: "Which hyperparameter settings of a VAE (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?". To answer this question the hyperparameters have been tuned until an X-ray came out of it. The variational autoencoder (VAE) was also hard to stabilize like the GANs compared to the first tests with the MINST dataset (an easier dataset to learn, about handwritten numbers). In the encoder/decoder itself, the VAE initially had only two hidden layers. However, the VAE could not capture any complex structures with that. That is why the VAE now has four hidden layers, it also has a leakyReLU function instead of the normal ReLU activation. The LeakyReLU gives more stability as it does not have the dying ReLU problem. The number of hidden dimensions was also too low initially, when testing for the MINST dataset 200 was enough. However, for the X-ray dataset 400 was adequate. Due to

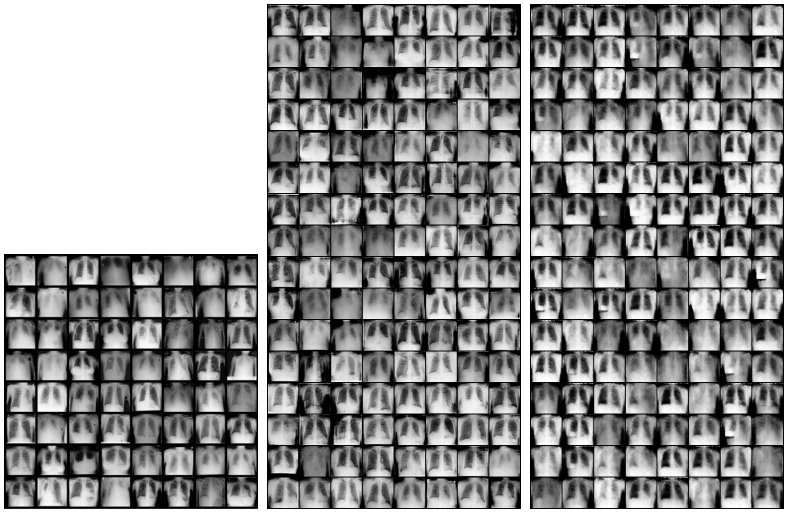

Figure 13: (left) pictures from the dataset itself to compare to the pictures on the right, (middle) pictures generated by the Generative Adversarial Network after 50 epochs and a channel size of 128. (right) pictures generated by the Generative Adversarial Network after 50 epochs and a channel size of 32.

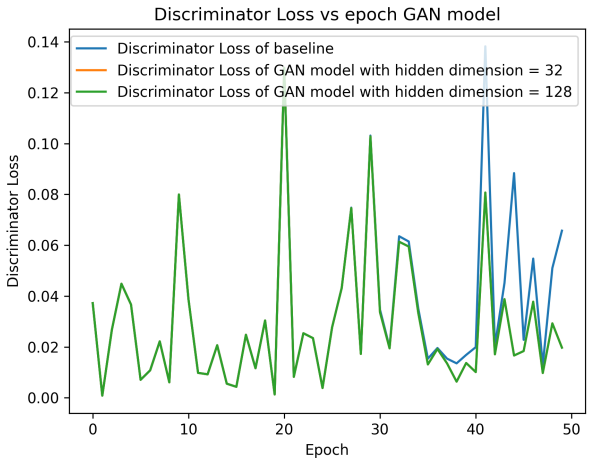

Figure 14: Graph containing the GANs where the hidden dimension size is tested (32, 64 and 128). The models ran with a learning rate of 1e-4, and 50 epochs exactly as the baseline.

instability, the learning rate was also a tad too high at 1e-3 initially and now at 1e-5. The same can be said for the amount of epochs which was first 20 now 50. The biggest problem was that even with all of these changes, the model would not stabilize. The model worked only after giving more data by providing higher-resolution images (The image size 128 instead of 64 was used). The images were also normalized between 0 and 1 and set to greyscale before feeding them to the model. Cudnn.benchmark was also enabled to make faster learning possible. For the specific implementation used in this thesis the baseline had the following parameters:

1. Batch size: 128

2. Image channels: 1 (greyscale)
3. Image size: 128 x 128
4. Learning rate: 1e-4
5. CUDA: true
6. Hidden dimensions: 400
7. Seed: 0

During the production of graphs the model was on evaluation mode and the gradients were not calculated because torch.no_grad() was used. The backpropagation was also disabled during testing. To conclude, the parameters stated above are the correct ones to have a stable algorithm that generates X-ray images.

3.3.2 Visual comparison baseline

The question that was attempted to be answered in this subsection is: "How do the baseline VAE generated images compare to the NIH X-ray dataset images according to the human eye?". This can be achieved by generating the images and comparing them to the images in the dataset itself. Figure 15 shows images from the dataset next to images generated by the baseline version of a Variational autoencoder. As is visible from figure 15, the Variational autoencoder might try to generalize the images too much. The pictures also seem not as detailed as the real images are (and as well as the other algorithms are recreating the X-rays).

To conclude the VAE does have some issues with generating images, it might try to overgeneralize.

3.3.3 Epoch size comparison

The question that was attempted to be answered in this subsection is: "For which epoch are the VAE generated images closest to the NIH chest X-ray dataset?". To answer the question the baseline algorithm is ran for 50 epochs. Figure 16 shows the discriminator loss over the baseline of the VAE. The figure shows that the discriminator cannot tell the difference well between the different epochs. This might be because the discriminator is solely trained on the images the generator from a GAN would reproduce. Those images are already quite detailed, and the VAE creates images that are not detailed but do improve.
In conclusion, the amount of loss per epoch does not change much which is why the baseline was kept.

3.3.4 Learning rate comparison

The question that was attempted to be answered in this subsection is: "For which learning rate is the VAE producing X-ray images that are closest to the NIH chest X-ray dataset?". To answer the question the baseline VAE, a VAE a lower learning rate (1e-5) and one with a learning rate of (1e-6) are compared to one and another. A higher learning rate caused instability likely due to its too aggressive way of updating and is for that reason not added to the model.

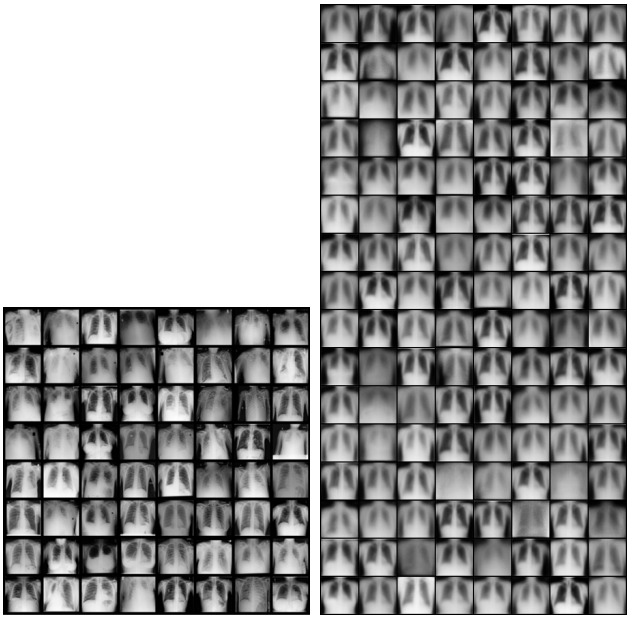

Figure 15: (left) pictures from the dataset itself to compare to the pictures on the right, (right) pictures generated by the Variational Autoencoder and a learning rate of 1e-4, after 50 epochs, 128 x 128 as image size and a hidden dimension size of 400.

Figure 17 shows the discriminator loss against the different learning rate settings in a VAE. In figure 17, it becomes visible that all the learning rates produced similar results. The discriminator might not be able to pickup the difference between them. This comparison was also difficult with the naked eye as the X-rays looked alike no matter the learning rate. A learning rate higher than the baseline created artifacts which is why only learning rates below the baseline were chosen. The models were not changed from their baseline other than the learning rate to perform these graphs. In conclusion, as the changes were not visible the baseline version is kept as the most appropriate one.

3.3.5 Image size comparison

The question that was attempted to be answered in this subsection is: "For which image size are the VAE generated images closest to the NIH chest X-ray dataset?". To answer the question the baseline VAE (128 x 128), a VAE with a image size (64 x 64) and a lower image size (32 x 32) are compared to one and another. The 256 x 256 version was too unstable. The 64 x 64 version was stabilized by using a much lower learning rate (1e-6). In figure 18 the different image sizes have been compared (128, 32 and baseline) against the discriminator loss. In figure 19 the images generated by img size 32, 64, 128, and a picture of the dataset can be found. From the graph, the 64 and 128 image sizes had similar performance. However, the image size = 32 made the biggest difference. This difference might be because the discriminator might have some trouble detecting the 32 x 32 variants because their resolution is lower and they are a bit less sharp than their GANs counterparts. One hidden layer for the 32 x 32 counterparts has been removed to make the model more stable. The rest of the parameters stayed the same. From the visual representation the 32 and 128 variants are also close. However, because of the big difference in the discriminator scores

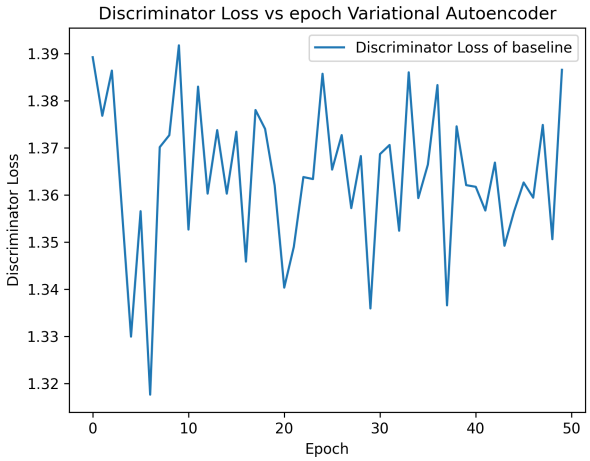

Figure 16: This graph compares the discriminator loss of the baseline per epoch. The VAE is trained on a 128 x 128 image size, with learning rate 1e-4 and hidden dimension size 400.

the image size of 32 was performing the best.
In conclusion, the image size of 32 was closest to the NIH chest X-ray dataset. Perhaps due to its lower resolution that the discriminator was less accurate.

3.3.6 Hidden dimension size comparison

The question that was attempted to be answered in this subsection is: "For which hidden dimension size are the VAE generated images closest to the NIH chest X-ray dataset?". To answer the question the baseline VAE (hidden dimension size 400), a VAE with a hidden dimension size 500 and a lower image size hidden dimension size 300 are compared to one and another. In figure 20 is shown how the different hidden dimension sizes compare through the discriminator loss. In figure 21 the hidden dimension sizes generated images can be compared, left is an image from the dataset, left middle is the baseline, right middle is the hidden dimension size of 300 and right is the hidden dimension size of 500. All the VAE's had a learning rate of 1e-4, an image size of 128 x 128 and was ran for 50 epochs. From the figure no notable differences emerge, between the different hidden dimension settings. From the graph it becomes clear that the hidden dimension of 500 had a better final outcome but the learning curve was a bit more unstable then the others. This might be due to the fact that more dimensions allow the algorithm to learn more complex representations of the data. The instability might be because there are more parameters if the amount of hidden dimensions increases. The baseline and the VAE with 300 hidden dimensions had about the same result.
In conclusion, due to the instability of the hidden dimension size of 500 and no visible changes or difference with the 300 version, the baseline was favourable.

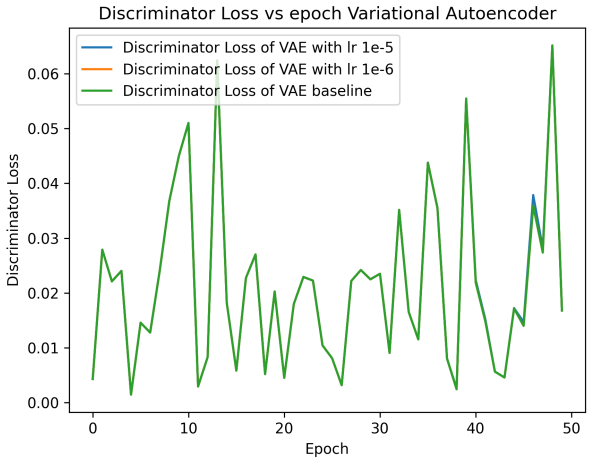

Figure 17: Variational Autoencoder and a learning rate of 1e-4 (baseline), 1e-5 and 1e-6 after 50 epochs, 128 x 128 as image size and a hidden dimension size of 400.

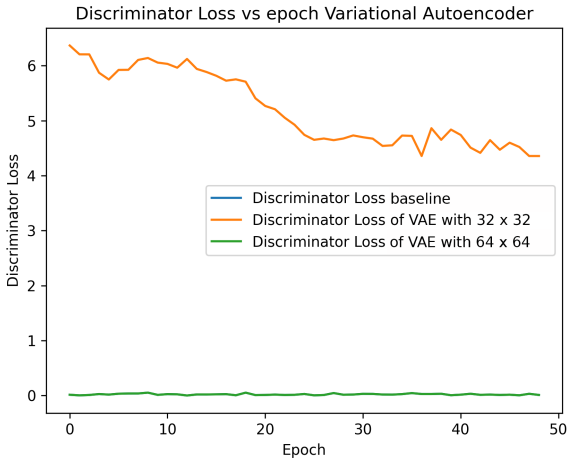

Figure 18: This graph compares the different image sizes of a variational autoencoder. The VAE is tested with an image size of 32, an image size of 128 and the baseline algorithm.

3.4 Autoregressive model

3.4.1 How the experiment is structured for diffusion models

The question which was attempted to be answered in this section was: "Which hyperparameter settings of an autoregressive model (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?". To answer this question the hyperparameters have been tuned until an X-ray came out of it. This model was also quite stubborn in stabilizing. The baseline is therefore not stable. Here are the modifications that were attempted to make the algorithm function: Changing kernel size (both higher and lower), changing amount of layers (both higher and lower), changing number of channels, changing the image size, adding more epochs, changing the probability function (softmax to logistic mixture),

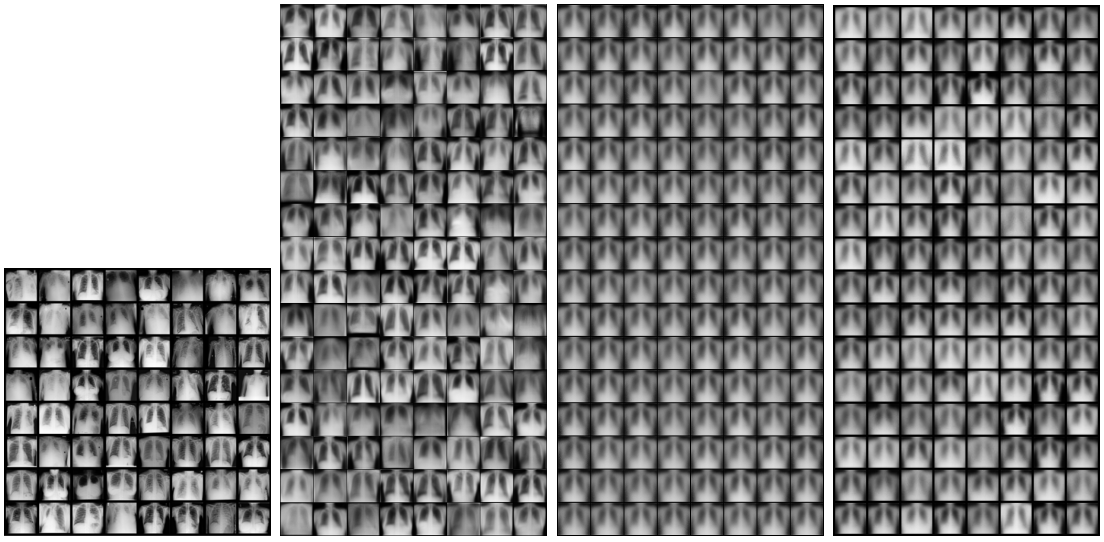

Figure 19: This figure is a visualization of the image sizes, baseline (128 x 128) (middle left), 64 (middle right) and 32 (right) compared to the sample from the dataset (left)

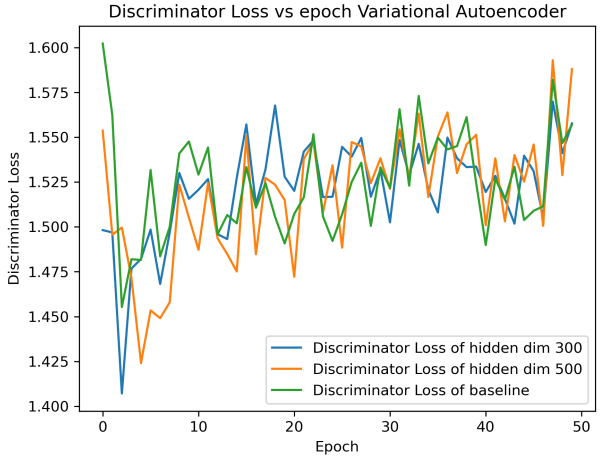

Figure 20: This graph compares the parameter hidden dimensions of a variational autoencoder. The VAE is tested with a hidden dimension of 300, 500 and 400 (baseline), with a learning rate of 1e-5 and 128 x 128 images.

changing the ReLU's to LeakyReLU's, changing the size of the masks, changing the amount of A layers so that those layers do not have access to the future information. However, all of these changes were to no avail. The baseline model was still unstable. Regarding computational efficiency, the algorithm took about a day to train on system 1 while utilizing the GPU. It took about two days for the image size 128 on system 2, while utilizing the GPU. For the specific implementation used in this thesis the baseline had the following parameters:

1. Batch size: 128
2. Image channels: 1 (greyscale)

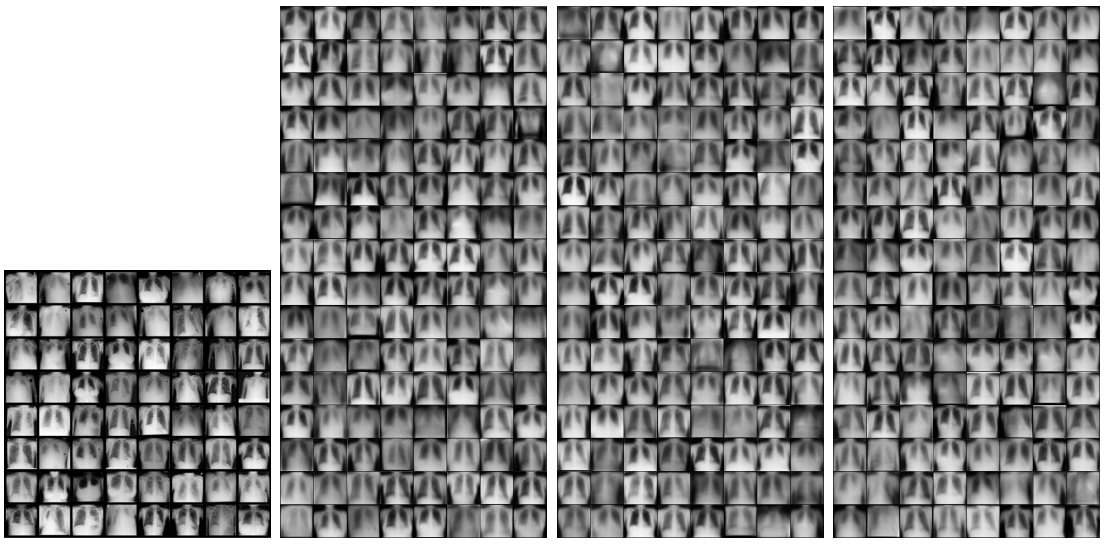

Figure 21: This figure is a visualization of the VAE generated images for the hidden dimensions, baseline (400) (middle left), a hidden dimension size of 300 (middle right) and a hidden dimension size of 500 (right) compared to the sample from the dataset (left)

3. Image size: 64 x 64
4. Learning rate: 1e-4
5. Kernel size: 7
6. layers: 10
7. CUDA: true
8. channels: 400
9. Seed: 0

In conclusion, the above mentioned parameters are the baseline parameters for the autoregressive model.

3.4.2 Visual comparison baseline

The question that was attempted to be answered in this subsection is: "How do the baseline generated images by the autoregressive model compare to the NIH X-ray dataset images according to the human eye?". This can be achieved by generating the images and comparing them to the images in the dataset itself. In figure 22 there is a comparison between a picture from the dataset and an image created by the baseline of the model. From these, it becomes clear that the model only got a part of an X-ray correctly. This might be due to either overfitting, making the model give artefacts, or underfitting, meaning the model has not been able to adapt to the format of an X-ray. Either way, the model is not functioning well.
In conclusion, the autoregressive model is not functioning well against the NIH chest X-ray dataset.

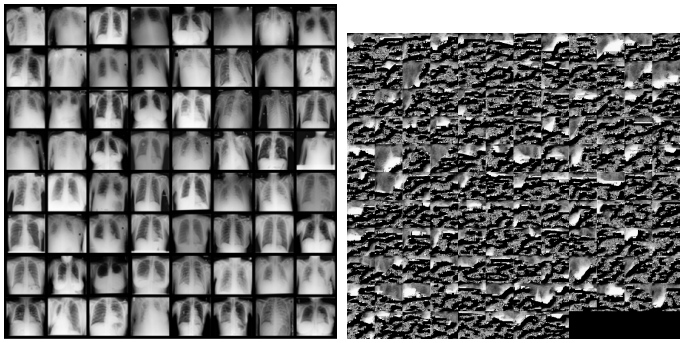

Figure 22: This figure contains (left) Image from the dataset, (right) image generated by the baseline autoregressive model.

3.4.3 Epoch size comparison

The question that was attempted to be answered in this subsection is: "For which epoch are the generated images of the autoregressive model closest to the NIH chest X-ray dataset?". To answer the question the baseline algorithm is ran for 25 epochs. Figure 23 contains the baseline autoregressive model against the discriminator. From the graph it becomes clear that the model does not improve much, as the lower the loss of the discriminator the more correct the discriminator is with recognizing the generated images.
In conclusion, due to the model not being as fine tuned as a whole the model is kept at 25 epochs.

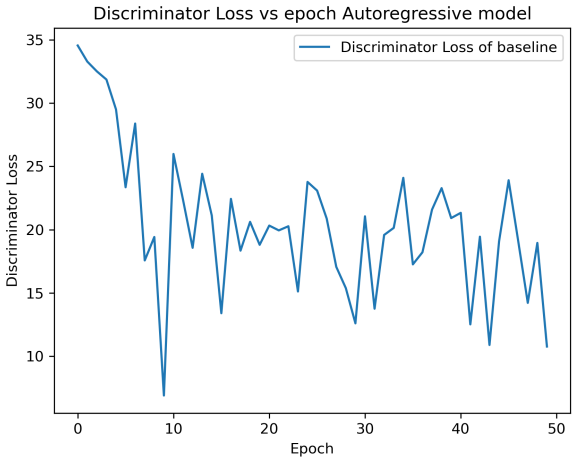

Figure 23: This graph contains the baseline autoregressive model against a discriminator.

3.4.4 Learning rate comparison

The question that was attempted to be answered in this subsection is: "For which learning rate is the autoregressive model producing X-ray images that are closest to the NIH chest X-ray dataset?". To answer the question the baseline autoregressive model, a autoregressive model with a higher learning rate (5e-4) and a lower learning rate (1e-5) are compared to one and another. Figure

24 shows a graph from the autoregressive model with different learning rates, 1e-5, 5e-4 and the baseline. These models did not have other variables that were different from the baseline. Only the learning rate was changed. From figure 24 it becomes clear that the discriminator can not tell at all if the fake samples are fake or real in this particular case. The only reason the 5e-4 scored much lower is because most of the images are darker. Because the images are darker they look a bit less like X-rays and the discriminator can tell the difference easier. The baseline and the lr = 1e-5 versions have lighter images which makes the discriminator even less capable of noting the difference, as it uses the same greyscale as an X-ray. The images of lr 1e-5 and lr 5e-4 can also be compared in figure 25. figure 25 compares the generated images from the different learning rate settings to an image from the dataset.
In conclusion, due to the algorithm not being fine tuned well enough the baseline of 1e-4 is kept.

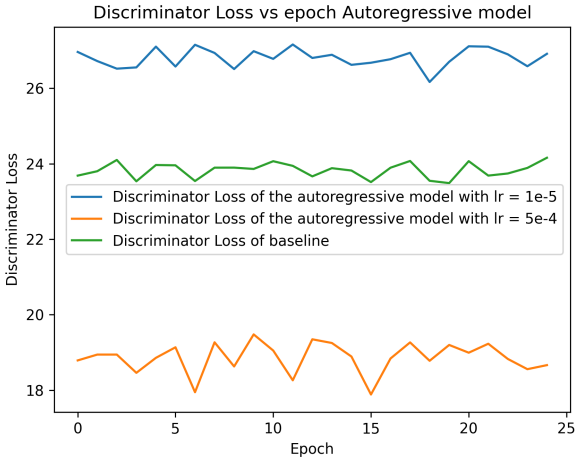

Figure 24: This figure contains a graph from the autoregressive model with different learning rates, 1e-5, 5e-4 and the baseline of 1e-4.

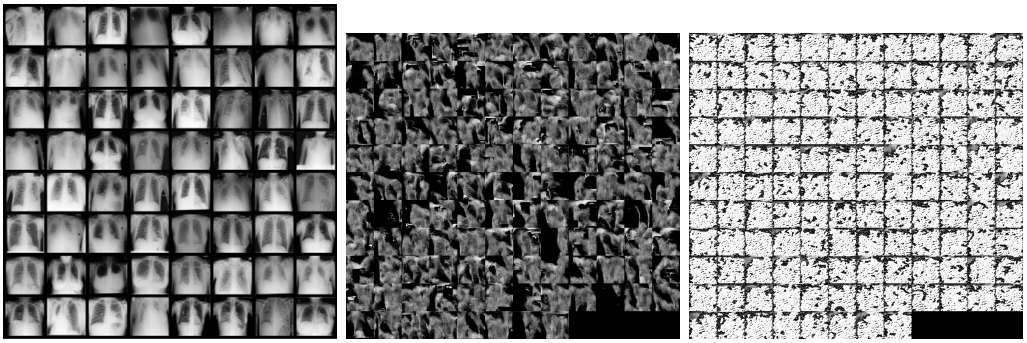

Figure 25: This figure contains: (left) a picture from the dataset (middle) a diffusion generated picture with a learning rate of 5e-4, (right) with a learning rate of 1e-5. Here a visual comparison between the two learning rates can be made.

3.4.5 Image size comparison

The question that was attempted to be answered in this subsection is: "For which image size are the images generated by the autoregressive model closest to the NIH chest X-ray dataset?". To answer the question the baseline autoregressive model (64 x 64), an autoregressive model with a image size (128 x 128) and a lower image size (32 x 32) are compared to one and another. From figure 26 becomes clear that the model with image size 128 has the highest score, the image size 32 had the lowest, and the baseline was in between. Image size 32 had images closest to an X-ray image but were still quite noisy. However, because the image size 32 had the images closest to a real X-ray is the reason why it has such a low score as the discriminator can identify the X-rays and add them to the fake labels. Figure 27 has a visualization of the generated images in the different image size settings. In that figure the problem of image size 32 is visible. The other image sizes however did not resemble an X-ray image, image size 128 was mostly white which is why the discriminator could not recognize it at all as a fake image and said it was real. Most likely because of the greyscale of the images. The same can be applied to the baseline.
In conclusion, the image size of 32 delivered the most promising results even if the discriminator might be less accurate on lower resolution images. The visual comparison still made it clear that the 32 image version of the autoregressive model was closest to the NIH chest X-ray dataset.

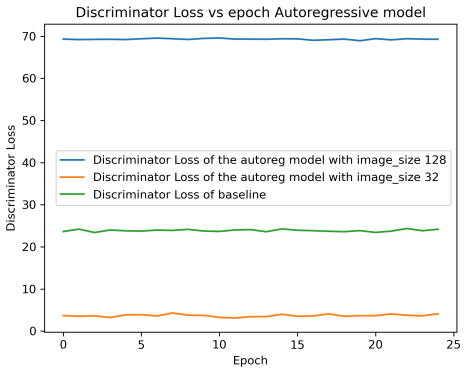

Figure 26: This figure contains a graph from the autoregressive model with different image sizes, 128, 64 (baseline) and 32.

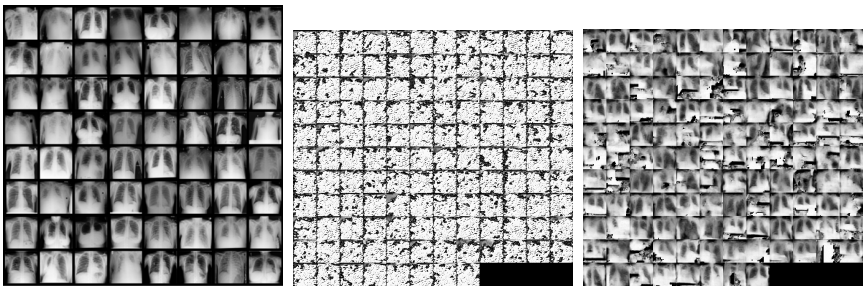

Figure 27: (left) pictures from the dataset itself to compare to the pictures on the right, (middle) pictures generated by the autoregressive models after 25 epochs and an image size of 128.(right) pictures generated by the autoregressive model after 25 epochs and an image size of 32.

3.5 Diffusion model

3.5.1 How the experiment is structured for diffusion models

The question which was attempted to be answered in this section was: "Which hyperparameter settings of a diffusion model (batch size, image channels, image size, learning rate, hidden dimensions, kernel size, padding) give a stable algorithm that generates x-ray images?". To answer this question the hyperparameters have been tuned until an X-ray came out of it. For the specific implementation used in this thesis the baseline had the following parameters:

1. Batch size: 128
2. Image channels: 1 (greyscale)
3. Image size: 64 x 64
4. Learning rate: 1e-4
5. CUDA: true
6. precision: fp16
7. Seed: 0
8. Kernel size: 4
9. Padding: 0
10. Output channels for the UNet blocks: 128, 256, 512
11. Gradient accumulation steps: 1 (Off)
12. Warmup steps: 500

During the production of graphs the model was on evaluation mode and the gradients were not calculated because torch.no_grad() was used. The backpropagation was also disabled during testing. In conclusion, the parameters above were chosen as the baseline parameters because of its stability and the images it generated where close to the NIH chest X-ray dataset.

3.5.2 Visual comparison baseline

The question that was attempted to be answered in this subsection is: "How do the baseline generated images by the diffusion model compare to the NIH X-ray dataset images according to the human eye?". This can be achieved by generating the images and comparing them to the images in the dataset itself. Figure 28 shows on the right an image from the dataset and on the left an image generated by the diffusion model. From the figure becomes clear that the model did overall fairly well, sometimes the algorithm still has some pictures that are not that well identifiable. Regarding computational efficiency, the algorithm took about 3.5 days to train on system 1 while utilizing the GPU and about 1,5 days to train on system 2. Due to the lower VRAM on system 1 the algorithm could never save the model since it kept crashing.
In conclusion, the diffusion model comes very close to the NIH chest X-ray dataset, visually even as close as the GAN.

Figure 28: (left) pictures from the dataset itself to compare to the pictures on the right, (right) pictures generated by the Diffusion model and a learning rate of 1e-4.

3.5.3 Epoch size comparison

The question that was attempted to be answered in this subsection is: "For which epoch are the generated images by the diffusion model closest to the NIH chest X-ray dataset?". To answer the question the baseline algorithm is ran for 50 epochs. Figure 29 shows the baseline diffusion model against the discriminator loss per epoch. The discriminator seems quite capable of detecting the difference between a real sample or a sample generated by the diffusion model. This might be due to its training on the GANs, which were already quite accurate, therefore training the discriminator on high-quality samples. It might also be due to the way the generator generates its images that creates the same noise as the diffusion model does. To conclude the baseline amount of 50 is kept due to the low amount of difference.

3.5.4 Learning rate comparison

The question that was attempted to be answered in this subsection is: "For which learning rate is the diffusion model producing X-ray images that are closest to the NIH chest X-ray dataset?". To answer the question the baseline diffusion model, a diffusion model with a higher learning rate (5e-4) and a lower learning rate (1e-5) are compared to one and another. Figure 30 shows on the left, an image from the database in the middle images generated by the diffusion model with a learning rate of 5e-4, and on the right generated images with a learning rate of 1e-5. Figure 31 shows a graph comparing the discriminator loss against different learning rate settings for the diffusion model. As becomes clear from the figures (figure 30 and figure 31), a higher learning rate would perform about the same as the baseline. The results overall do look quite similar. The 1e-5 might be a bit unclear as a lower learning rate makes sure that the model updates its weights slower. Other than

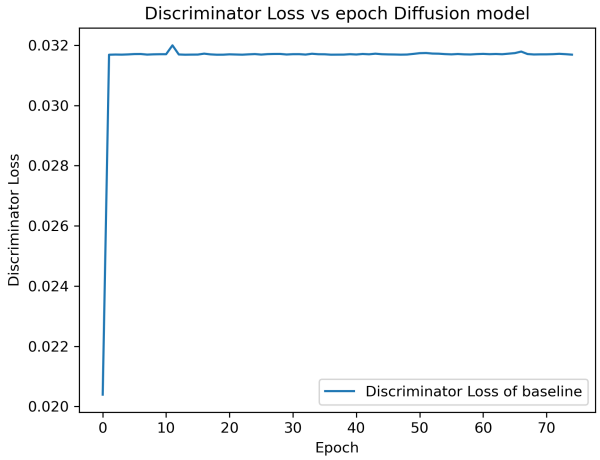

Figure 29: The discriminator loss for detecting the difference between the fake sample and a real sample (higher is better as the higher the discriminator loss the harder it must have been for the discriminator to detect the fake samples generated by the diffusion model).

the learning rate changes, the hyperparameters stayed the same as in the baseline model.
In conclusion, the baseline performed as well as the higher learning rate, for that reason the higher learning rate is kept.

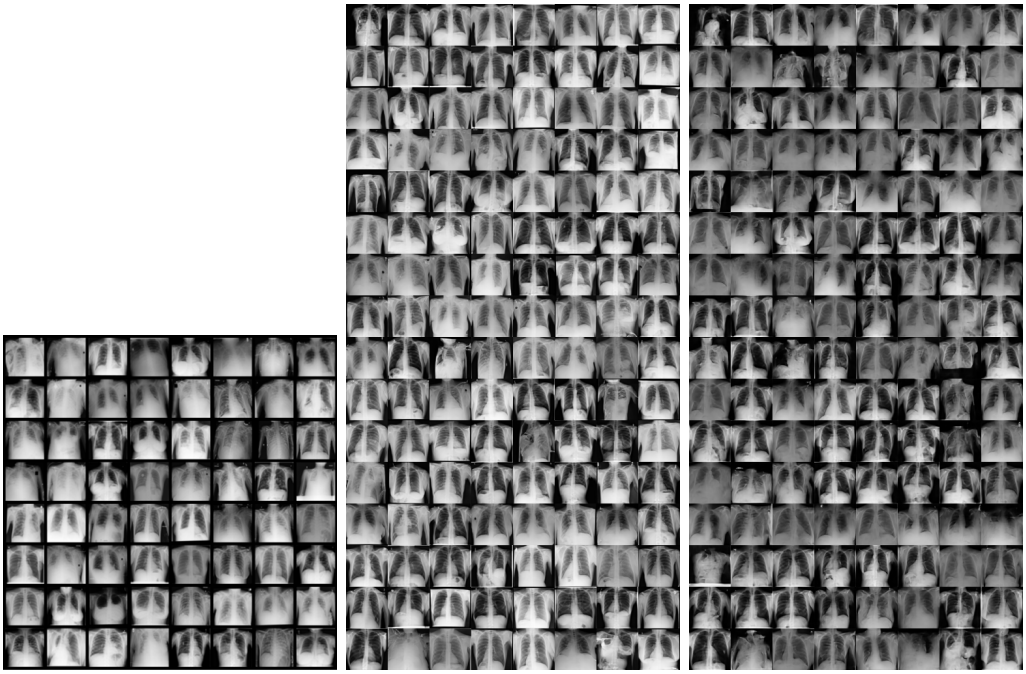

Figure 30: This figure contains: (left) a picture from the dataset (middle) a diffusion generated picture with a learning rate of 5e-4, (right) with a learning rate of 1e-5. Here a visual comparison between the two learning rates can be made.

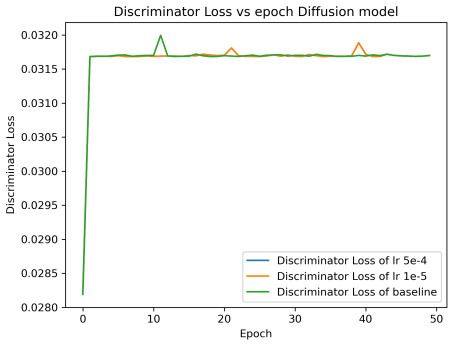

Figure 31: This graph contains a comparison between the different learning rate settings of the diffusion model and the discriminator loss per epoch

3.5.5 Image size comparison

The question that was attempted to be answered in this subsection is: "For which image size are the images generated by the diffusion model closest to the NIH chest X-ray dataset?". To answer the question the baseline diffusion model (64 x 64), and a lower image size (32 x 32) are compared to one and another. A higher resolution image was not taken as the amount of GPU VRAM the algorithm needed was more then there was available on both system 1 and 2. A too complicated model for a 32 x 32 image size created a black image like in figure 32. To combat this the model for 32 x 32 has been heavily simplified, the model has only three layers for downsampling: 2D ResNet Downsampling Block, 2D ResNet Downsampling Block with Spatial Self-Attention and then another 2D ResNet Downsampling Block. For up sampling it has a similar configuration but then with upsampling blocks instead of down sampling blocks. The model is as follows: 2D ResNet Upsampling Block, 2D ResNet Upsampling Block with Spatial Self-Attention, and a 2D ResNet Upsampling Block again. Other attempts at stabilizing the diffusion model by lowering the learning rate, changing the optimizer, changing the precision, changing the loss function did not help as much. Other than the model changes the hyperparameters stayed the same as the baseline model. Due to limitations of both systems the 128 x 128 diffusion model could not be tested, the code required more VRAM then available on these machines. However, after simplifying the model, the diffusion model could produce proper images. In figure 34 The comparison between the dataset itself (left), and the diffusion model after 50 epochs and an image size of 32 x 32 (right) is visualized. In figure 33, the difference between the baseline and the image size = 32 for the discriminator loss is visualized. Image size 32 has a much higher discriminator loss, this might be due to there being less pixels and the discriminator might not be able to quantify as well if there is less data.
In conclusion, the image size of 32 has created images closer to the NIH chest X-ray dataset both through the human eye and the discriminator.

3.6 Comparison based on the human eye

The question attempted to answer in this subsection is: "How do the algorithms' generated images (VAE, autoregressive, GAN, diffusion model) compare to each other through the human eye when compared to the NIH chest X-ray dataset?". This question will be answered by the writer comparing

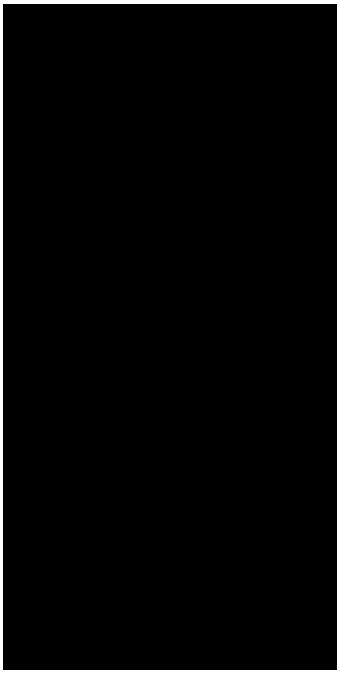

Figure 32: Picture of a diffusion model when there are too many layers when the resolution of the pictures is 32 x 32.

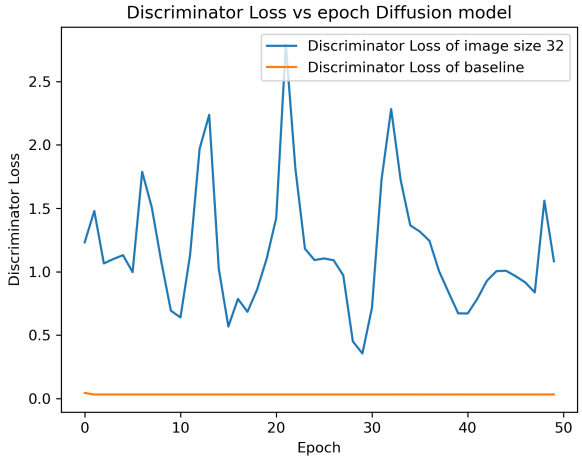

Figure 33: The discriminator loss for detecting the difference between the fake sample and a real sample. Diffusion algorithm with inputs 32 x 32 and inputs 64 x 64 (baseline).

the differences between the images. Figure 35 displays an image from the dataset, the GAN baseline generated image, a VAE generated image, an autoregressive model generated image, and an image generated by the diffusion model. The autoregressive model does not seem to be stable. Therefore, this model is the least close to the NIH chest X-ray dataset. The VAE model, does not seem to generate very detailed images. Because of that, the VAE model is also not close to the NIH chest X-ray dataset. The GAN while close to the dataset still has some small artefacts and somewhat unclear images. That is why the diffusion model is closest to the dataset, it also has very little

Figure 34: (left) pictures from the dataset itself to compare to the pictures on the right, (right) pictures generated by the Diffusion model after 50 epochs and an image size of 32 x 32.

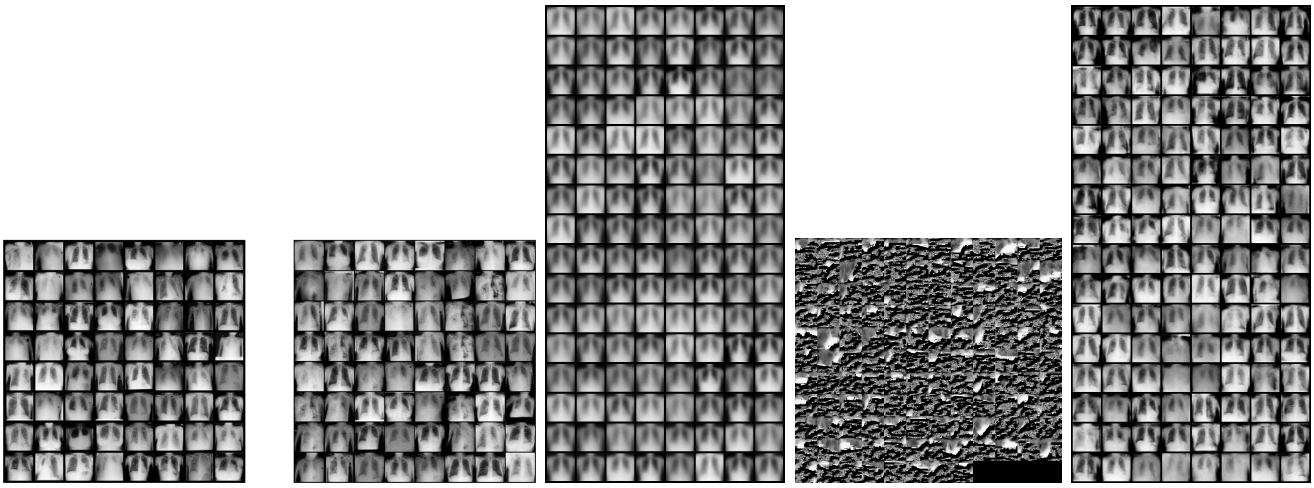

Figure 35: (left) pictures from the dataset itself to compare to the pictures on the right, (middle left) pictures generated by a VAE after 50 epochs and an image size of 32 x 32. (Middle right) images generated by the baseline autoregressive model. (right) pictures generated by the Diffusion model after 50 epochs and an image size of 32 x 32.

artefacts in its generated images. In conclusion, the results suggest that the diffusion model is the closest to the dataset, then the GAN, then the VAE and lastly the autoregressive model.

3.7 Comparison based on the discriminator

3.7.1 Results from the discriminator that is from the Generative Adversarial Network

The question attempted to be answered in this subsection is: "How do the algorithms' (VAE, autoregressive, GAN, diffusion model) compare to each other through the discriminator of the GAN?". To answer this question all the hyperparameters that the previous results suggest were optimal were chosen for the algorithms. Then they were all compared using a discriminator from the GAN in its 50th epoch. Figure 36 compares the discriminator loss of all the best versions of the models together. Generally, higher is better however if a model does not produce images that look like X-rays then the discriminator will also give it a high score. The models that utilize a 32 x 32 image size also use a 32 x 32 discriminator as that discriminator model is slightly different. This may impact the results as the 32 x 32 discriminator model has one layer less and also has less data as 32 x 32 is less data than 64 x 64. This graph shows that the VAE and autoregressive models have the most challenging images for the Discriminator to detect. Simply because they look less like an X-ray image from the dataset. The best-performing model with this graph combined with the visual representation is the diffusion model. As that model creates an accurate X-ray image. In conclusion, the diffusion model is the algorithm that has a combination of the highest score of the discriminator while still creating images close to the NIH chest X-ray dataset.

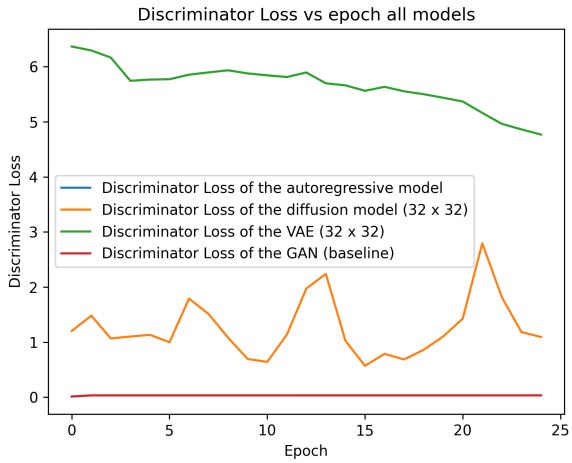

Figure 36: The discriminator loss against all models per epoch (the VAE has the same values as the autoregressive model).

3.7.2 Results from the discriminator that is trained as a classifier

The question attempted to be answered in this section is: "How do the algorithms' (VAE, autoregressive, GAN, diffusion model) compare to each other through a classifier?". To answer this question all the hyperparameters that the previous results suggest were optimal were chosen for the algorithms. Then they were all compared using a classifier algorithm. In figure 37 is shown classifier scores of a trained classifier on all the best versions of the models. The trained classifier is a trained CNN model with as training data the NIH X-ray images and the same amount of images per model.

The trained model consists of two convolutional layers, a max pooling layer and two linear layers. The activation functions are sigmoid to help get the output between 0 and 1. The value of 0,5 means that the classifier thought these images were generated. The VAE and the diffusion model are at the 0,5 mark. In conclusion, the VAE and the diffusion model were both easily detectable for the classifier algorithm. The GAN and the autoregressive model however proved to be tougher. These might also be harder to detect as the model does not recognize the autoregressive model images well enough and the GAN because it was already accurate. Because the GAN was the only one of the two that also produced images that are visually close to the algorithm it performed the best for this subsection.

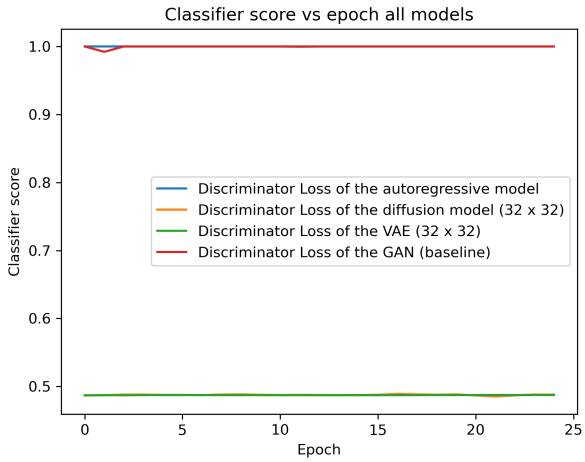

Figure 37: The classifier loss against all models per epoch

4 Discussion

The diffusion model generated the most suitable X-ray images to be used by other machine learning algorithms. This conclusion was based on evaluations through visualisations and comparison graphs of both the discriminator and the classifier. The GAN was in the second place, followed by the VAE. The autoregressive model demonstrated instability and could not be tested adequately, indicating a need for improvement through future research.

During this study, problems arose. Finetuning the models was sometimes challenging as certain models proved unstable when adjusted to various configurations. In additon, it took a long time to test all the conditions. One run of 50 epochs of the diffusion model alone took about two or three days, as training takes one day and generating a graph another day. The models could be more accurately finetuned if there was more time, which would be a good starting point for a future study. More time spent on the autoregressive model would also be beneficial for future research. It would also be a good idea to add the classifier during testing of the hyperparameters, to obtain a more balanced view of how the models perform overall.

The most surprising results were that the 32 x 32 models performed the best overall, in almost every model. Usually, less data results in a poorer performance than having more data. The better performance was likely due to the discriminator and the classifier having less data to recognise the generated images and not because they were higher-quality images. In addition, the GAN also likely had higher scores for the discriminator because the generator and the discriminator were in balance with one another. During the previous epochs (before the 50th), the discriminator had already utilised the generated images, and perhaps it recognised them and thus give them a lower score.

No statistical testing was performed in this study. The reason was that it would have been challenging to compare the visual manual inspection of the generated images with the discriminator and classifier scores.

For future work, it would be beneficial to develop a more reliable autoregressive model to enable comprehensive testing and comparison. Additionally, utilising more powerful systems that are capable of generating higher-resolution images could reveal more detailed flaws in the various methods. The current systems' limitations in VRAM prevented the diffusion model from running at 128 x 128 dimensions. Hence, future studies should employ systems with higher VRAM capacity.

Further research could also be conducted using a different comparison method, as the discriminator from the GAN had already been trained on the GAN. This point could have lowered the results against the GAN. The discriminator might be less able to find the difference in images generated in other ways than using a GAN.

Overall, enhancing the stability of the autoregressive model and increasing the system's capabilities, as well as using unbiased evaluation methods, are crucial steps for advancing the field and achieving accurate image generation. The ultimate aim is to test what model for medical X-ray images generates images the best images for use by other machine learning algorithms. The trained algorithms can be used to generate 10,000 images each. The images can then be given to an algorithm that is

trained on the NIH chest X-ray dataset (see section 2.2). The levels of recognition can be tested by examening how the machine learning algorithms for utilising data classify images.

5 Conclusion

To conclude, different models were tested and each models hyperparameters that comes closest to the NIH chest X-ray dataset were chosen. This should help in answering the final question: "Which algorithm (Generative Adversarial Networks, Variational Autoencoders, Diffusion models, or Autoregressive models) generates chest X-ray images with the highest fidelity for diagnostic purposes?". For the diffusion model was that the image size of 32, for the Variational Autoencoder (VAE) too and for the Generative Adversarial Network (GAN) and autoregressive model the baseline was closest to the NIH chest X-ray dataset. After that the models were compared by a discriminator of a GAN and by a classifier. Combining those results with the human eye comparison, it was clear that the diffusion model was closest to the NIH chest X-ray dataset.

A The code

The code can be found here:
https://github.com/DMR-max/gen_x_ray_images

References

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, January 2017.

[Bha18] Antonia Creswell; Tom White; Vincent Dumoulin; Kai Arulkumaran; Biswa Sengupta; Anil A. Bharath. Generative Adversarial Networks: An Overview, 1 2018.

[Che17] Xi Chen. Pixelsnail: an improved autoregressive generative model, December 2017.

[CPP+20] Kelton A. P. Costa, João Paulo Papa, Leandro A. Passos, Danilo Colombo, Javier Del Ser, Khan Muhammad, and Victor Hugo C. De Albuquerque. A critical literature survey and prospects on tampering and anomaly detection in image data. *Applied soft computing*, 97:106727, 12 2020.

[Dal19] Murtaza Dalal. Autoregressive models: What are they good for?, October 2019.

[Din16] Laurent Dinh. Density estimation using real nvp, May 2016.

[Din23a] Anh-Dung Dinh. PixelAsParam: A Gradient View on Diffusion Sampling with Guidance, 7 2023.

[Din23b] Anh-Dung Dinh. Rethinking Conditional Diffusion Sampling with Progressive Guidance, 11 2023.

[Du23] Chengbin Du. Stable Diffusion is Unstable, 6 2023.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, March 2017.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial Nets, 2014.

[Ho20] Jonathan Ho. Denoising Diffusion Probabilistic Models, 6 2020.

[Ho22] Jonathan Ho. Video Diffusion Models, 4 2022.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006.

[Hua23a] Lianghua Huang. Composer: Creative and Controllable Image Synthesis with Composable Conditions, 2 2023.

[Hua23b] Tao Huang. Knowledge Diffusion for Distillation, 5 2023.

[Kin13] Diederik P Kingma. Auto-encoding variational bayes, December 2013.

[Kin18] Diederik P. Kingma. Glow: Generative flow with invertible 1x1 convolutions, July 2018.

[Met22] Gal Metzer. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures, 11 2022.

[MH96] Mary M. Moya and Don Hush. Network constraints and multi-objective optimization for one-class classification. *Neural networks*, 9(3):463–474, 4 1996.

[Mic22] Umberto Michelucci. An Introduction to Autoencoders, 1 2022.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, February 2018.

[Mou23] Chong Mou. T2I-Adapter: Learning Adapters to Dig out More Controllable Ability for Text-to-Image Diffusion Models, 2 2023.

[MXW+20] Yifan Mao, Fei-Fei Xue, Ruixuan Wang, Jianguo Zhang, Wei-Shi Zheng, and Hongmei Li. *Abnormality detection in chest X-Ray images using uncertainty prediction autoencoders*. 1 2020.

[Nic21] Alex Nichol. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models, 12 2021.

[Par18] Niki Parmar. Image transformer, February 2018.

[PLP+22] Alice Presenti, Z. Liang, Luis F. Alves Pereira, Jan Sijbers, and Jan De Beenhouwer. Automatic anomaly detection from X-ray images based on autoencoders. *Nondestructive testing and evaluation*, 37(5):552–565, 6 2022.

[PSCVDH21] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection. *ACM computing surveys*, 54(2):1–38, 3 2021.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-NET: Convolutional Networks for Biomedical Image Segmentation, 5 2015.

[RIZ+17] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning, 11 2017.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 11 2015.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models, 1 2014.

[Rom21] Robin Rombach. High-Resolution Image Synthesis with Latent Diffusion Models, 12 2021.

[Rua22] Ludan Ruan. MM-Diffusion: Learning Multi-Modal Diffusion Models for Joint Audio and Video Generation, 12 2022.

[RVG+18] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep One-Class Classification, 7 2018.

[Sal17] Tim Salimans. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, January 2017.

[SD15] Jascha Sohl-Dickstein. Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 3 2015.

[SKFA18] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned One-Class classifier for novelty detection, 2 2018.

[Son20] Jiaming Song. Denoising Diffusion Implicit Models, 10 2020.

[SPS+01] Bernhard Schölkopf, John Platt, John Shawe-Taylor, Alex Smola, and Robert C. Williamson. Estimating the support of a High-Dimensional distribution. *Neural computation*, 13(7):1443–1471, 7 2001.

[SSW+19] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, 54:30–44, 5 2019.

[TTH+19] Yuxing Tang, Youbao Tang, Mei Han, Jing Xiao, and Ronald M. Summers. Abnormal chest x-ray identification with generative adversarial One-Class classifier, 3 2019.

[TW17] Jakub M. Tomczak and Max Welling. VAE with a VampPrior, 5 2017.

[TWH+18] Yuxing Tang, Xiaosong Wang, Adam P. Harrison, Le Lu, Jing Xiao, and Ronald M. Summers. Attention-Guided Curriculum learning for weakly supervised classification and localization of thoracic diseases on chest radiographs, 7 2018.

[unk21] Viral pneumonia screening on chest X-Rays using Confidence-Aware anomaly Detection, 3 2021.

[VDO16] Aaron Van Den Oord. Conditional image generation with pixelcnn decoders, June 2016.

[vdOKK16] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR.

[WPL+17] Xiaosong Wang, Yifan Peng, Le Lü, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. ChestX-Ray8: Hospital-Scale chest X-Ray database and Benchmarks on Weakly-Supervised Classification and Localization of Common thorax Diseases. *IEEE CVPR 2017*, 7 2017.

[Yin23] Shengming Yin. NUWA-XL: Diffusion over Diffusion for eXtremely Long Video Generation, 3 2023.

[YYH18] Elliot Yates, Louise C. Yates, and Hugh Harvey. Machine learning "red dot": open-source, cloud, deep convolutional neural networks in chest radiograph binary normality classification. *Clinical radiology*, 73(9):827–831, 9 2018.

[Zha23] Lvmin Zhang. Adding Conditional Control to Text-to-Image Diffusion Models, 2 2023.

[ZSE17] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models, 2 2017.