Towards Incremental Binary Neural Image Compression

Stefan H. Klut
11331720

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Marco Federici MSc

Institute of Informatics
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 28th, 2019

Table of Contents

Abstract

Image compression is necessary in modern society to ensure fast and efficient data transfer. Sending information across a connection is done incrementally, and the compression method implemented in this paper aims to take advantage of this incremental nature of data transfer. This paper proposes a method to make binary neural image compression work incrementally. One of the most common methods for neural image compression is the autoencoder, which will be the basis of the incremental compression algorithm implementation in this paper. The goal of the incremental autoencoder is to reconstruct the image from the encoded data received, even if not all the encoded data has been sent yet. The incremental autoencoder is trained to deal with the missing data by corrupting part of the encoded data during the training process. The results of this paper show a strong indication that the incremental autoencoder is able to handle missing a part of the encoded data much better than a regular autoencoder. This research has shown great potential for reconstructing images using the proposed incremental autoencoder.

1 Introduction

1.1 Context

Even at a time when storage capacity is larger and cheaper than ever, compression still plays a vital role in the information industry. The need for fast data transfer has made compression one of the go-to solutions for faster connections. Almost all modern applications use some form of compression. The internet could not function without the compression that is used to send images, audio and videos over the internet. The amount of data that would need to be transferred otherwise is simply too big for the current network speeds. The storage capacity had been growing steadily at at rate of 44% per year for over 50 years (Wood (2009)). This growth has recently been slowing down, even with the introduction of newer technologies like heat-assisted magnetic recording (Gao (2018)). If humanity wants to continue its exponential growth in data creation (Mittal and Vetter (2016)), it needs to be able to save this data somewhere. Thus, it becomes apparent why there is a need for better data compression.

In data compression, there are two approaches: lossless compression and lossy compression (Sayood (2017)). Using lossless compression, no information is lost, and the uncompressed data is therefore identical to the original data. There are many instances in which no data can be lost. A loss of information in critical bank statements could result in a great financial loss. Of course, not all data types have information that can not be omitted. In lossy compression, some information is discarded to achieve an even higher compression rate. However, this yields uncompressed data that is not identical to the original data. For images this is often not a problem since there exists information stored in an image that the human eye can not perceive.

In the field of image compression there are many standards file formats, each with their own form of compression. The Portable Network Graphics (PNG) format supports lossless compression (Roelofs (1999), and is widely used on the internet for high quality images. The most commonly used image format on the internet is the JPEG format (Wallace (1992)), which uses lossy image compression to obtain even smaller file sizes than PNG. These standards have been around for years, with the PNG and the JPEG dating back to 1996 and 1992 respectively. There have been attempts to replace these standards (Skodras et al. (2001)), but the JPEG and PNG remain the standard for image compression.

In recent years the use of artificial neural networks (NN) has created a new wave of compression algorithms with even smaller file sizes. The increase in computing power has made training a NN for compression viable, with image quality comparable to or better than the traditional compression methods. One of the most common architectures for image compression using a NN is the autoencoder (AE). The goal of an AE is to learn a representation of the input data, which will then be reconstructed as similarly as possible to the original input. In the case of compression, the representation should be smaller than the input. Where normal AEs represent the encoded image using real numbers, the type of AE this paper proposes represents the encoded image as a binary string, as binary is the fundamental form of data representation on a computer.

When sending an image across a connection, the data encoding the image will be sent incrementally. To take full advantage of this method of sending data, the decoding of the image should also be done incrementally. For this to be possible, the compression needs to be incremental as well. This means that the first bits are the most important, and that subsequent data will improve the quality of the image. This paper proposes making an incremental AE, which is trained to encode and decode the image incrementally. A successful implementation of incremental compression allows the image to be reconstructed with only a small amount of the data from the image. Internet users with slower connections would be able to download part of the data to still display a low resolution version of the image.

1.2 Related Research

1.2.1 Compression with Autoencoders

AEs have traditionally been used as a way to reduce the dimensionality of data and for feature learning. The dimensionality reduction serves a similar purpose to principal component analysis (PCA). The reduced dimensionality representation can be used to improve many tasks (e.g. classification) by helping to avoid the curse of dimensionality (Goodfellow et al. (2016)). This reduced dimensionality is a form of compression.

In Hinton and Salakhutdinov (2006) an AE is compared to PCA, on the basis of the reconstruction error. The AE outperformed the PCA on handwritten digits from the MNIST data set (see section 3.1.1) after reducing the 784 pixel images to both 6 and 30 features. Hinton and Salakhutdinov (2006) demonstrated that deep AEs (AEs with multiple hidden layers) are very effective at non-linear dimension reduction if the initial weights are already close to the solution. If this is the case, the gradient descent is able to optimize the weights.

AEs have also been used to compress medical images. In the case of Tan and Eswaran (2011), the compressed images were mammograms. Due to the size of the images the authors of the paper chose to train on patches of the full image. These patches were randomly selected from the full image. In Tan and Eswaran (2011) the effect of different AE architectures and different patch sizes was also studied.

1.2.2 Binarization

There are not many AEs that encode the data in a categorical form in literature. The problem with trying to create a stochastic NN with discrete variables is that it is not possible to back propagate through the sample. Meaning that the layers before the categorical variable are not able to update their weights.

To remedy this issue, Jang et al. (2016) developed a gradient estimator which replaces the sample from a categorical distribution with a differentiable sample from the Gumbel-Softmax distribution. The Gumbel-Softmax distribution is able to be smoothly annealed to the categorical distribution. The similarity between the Gumbel-Softmax distribution and the categorical distribution is described by the Softmax temperature variable (τ). When τ approaches 0, the two distributions become identical. The Gumbel-Softmax distribution becomes useful if $\tau > 0$, as for temperatures higher than zero the Gumbel-Softmax distribution becomes smooth, and therefore has a gradient. The categorical samples that have been replaced with the Gumbel-Softmax samples will then be able to use back propagation to update the gradients.

The Gumbel-Softmax Estimator was shown to work with both Bernoulli and categorical distributions (Jang et al. (2016)). It outperformed other estimators for categorical variables when implemented in a variational autoencoder (VAE). VAEs are similar to AEs, instead of learning to represent the data, they learn the parameters of a probability distribution representing the data (Kingma and Welling (2013)). The Gumbel-Softmax Estimator also outperformed other estimators for Bernoulli variables in a VAE.

Sampling from the Gumbel-Softmax distribution works well if the samples do not need to be discrete. However, if the samples have to be discrete (like in binary), the normal Gumbel-Softmax

estimator does not work. The solution that Jang et al. (2016) provide for this problem is the Straight-Through Gumbel-Softmax Estimator. This estimator still uses its continuous estimation during back propagation, but in the forward pass the values of the estimation are discretized. In the case of binary information this results in the estimations only being ones and zeroes.

1.3 Contribution

The goal of this paper is to design an AE that is able to encode and decode incrementally. This incremental autoencoder (IAE) should be able to encode an image to a binary form, where the closer the bits are to the start of the encoded form the more important they are. What makes the IAE special is that it can decode the encoded form, even if it has only received part of the data. Using the bits it has received, the IAE reconstructs the image, without fully needing the bits it has not yet received.

The IAE combines a regular AE with the Straight-Through Gumbel-Softmax Estimator to create a binary AE. The Straight-Through Gumbel-Softmax Estimator is expected to work as a way to make the AE encode the image in a binary form, as it has been successfully used in an NN architecture similar to an AE (VAE).

After creating a binary AE, the paper will attempt to make the AE incremental. To the best of the author's knowledge, there have been no previous attempts to create such an AE. A new method must therefore be developed to make the AE incremental. The new method will train the AE by replacing part of the encoded data with random bits. By replacing some of the encoded data with random bits, the AE has to learn not to store important information in these bits, as the information might be replaced. By evaluating how well the new method works, the paper will try to answer the research question:

> "To what extent can image compression using autoencoders be made incremental?"

The IAE is hypothesized to learn to deal with the missing part of the encoded data better than the regular AE. The IAE will have been trained to deal with this type of data and should therefore be able to decode it with better results than the regular AE.

The paper will be constructed in the following way. At first, a summary of AEs will be given, followed by a method of making the encoded representation binary. Then, the attempted method of making the data compression incremental will be described. To determine if the AE was able to make the compression incremental, it will be compared to a regular AE and an AE that has been trained to deal with random corruption. The results of these comparisons will be analyzed and displayed. In these results, there is a strong indication that the IAE is better at dealing with incremental data than a regular AE. Finally, this paper ends with a discussion of improvements and suggestions for follow-up research.

1.4 Scientific Relevance

The IAE would be the first AE that has been trained to deal with missing bits in the encoded form of the data. The main purpose of this paper is a proof of concept of the method of making the AE incremental. If this paper were to find encouraging evidence that the AE is able to be made incremental, it would open up other research opportunities into making other types of compression methods incremental.

By studying how the AE deals with the missing data, it is possible to learn something about how to deal with corruption in general. AEs have been trained to fix corruption in images before (Xie et al. (2012)), however this corruption was added onto the original image. Dealing with corruption in the encoded version of the image is something that has not been done before. As replacing the missing data with random bits is a form of corruption, the IAE might be able to give insight into dealing with corruption in the encoded data.

By restricting the IAE to put the most important information in the first bits, the features in the front of the encoded data are artificially given a higher feature importance. This might also be

useful in other fields of research where features are already known to be of higher importance than others.

2 Method

2.1 Autoencoders

The basic compression method using NNs that this paper will extend upon is an AE. The general structure of an AE is described by Hinton and Salakhutdinov (2006). It states that an AE is a NN with traditionally a symmetrical structure. AEs consist of two parts: the encoder, which maps the original data to an encoding, and the decoder, which maps the encoded version of the data to a reconstruction of the original data. The goal of an AE is to do this reconstruction as good as it possibly can, according to some loss function which determines how good the reconstruction was.

Due to the fact that the objective of the AE is to reconstruct the original data, the output layer has the same number of nodes as the input layer. The middle layer is representative of the encoded data. All layers from the input layer to the middle layer are part of the encoder, and all the layers from the middle layer to the output layer are part of the decoder. When there are fewer nodes in the middle layer than there are in the input/output layer, the AE is said to be undercomplete. When the number of nodes in the middle layer is larger than the number of nodes in the input/output layer, the AE is said to be overcomplete (Charte et al. (2018)). In figure 1 a possible architecture for an AE can be seen.

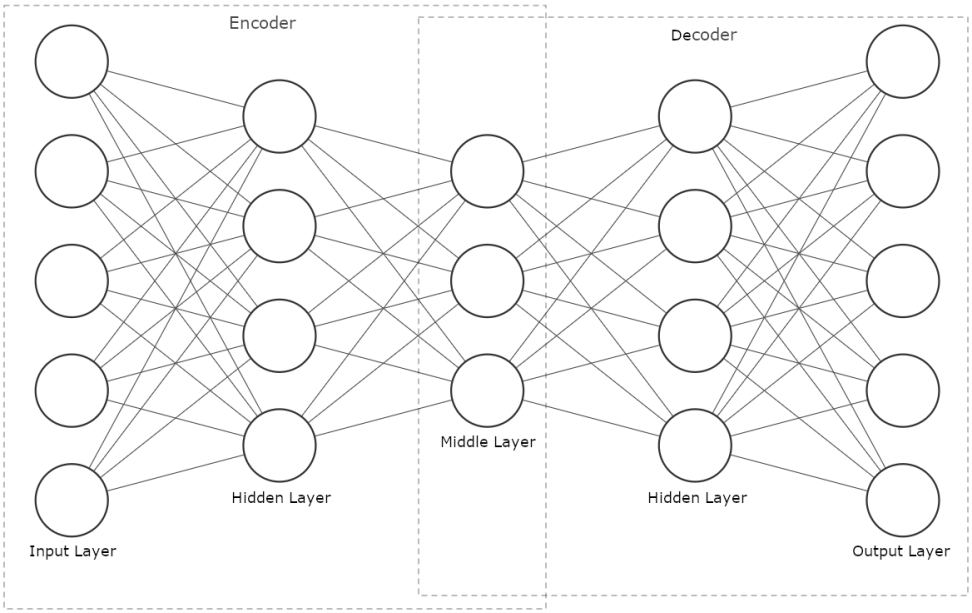


Figure 1: A possible architecture for an AE, with a middle layer with three variables. The AE is deep as it has hidden layers and is undercomplete as there are fewer nodes in the middle layer than in the input/output layer.

The number of layers in the encoder and decoder can vary depending on the application of the AE. When an AE consists of just the input, middle and output layer, the AE is called shallow. If there are more hidden layers in between the input, middle and output layer, the AE is called deep. The number of nodes in the layers can also vary depending on the application, especially the number of nodes in the middle layer. When using an AE for compression, as this paper is trying to achieve, the number of nodes in the middle layer needs to be smaller than the number of nodes in the input/output layer. Because the AE is a NN with a symmetrical structure, the encoder has the same number of layers as the decoder. The number of nodes within each layer is also the same, but in reversed order (Hinton and Salakhutdinov (2006)).

2.2 Binarization

When working with an AE, the middle layer is usually comprised of real numbers, however this is not the most compact form of storing the type of information that is being compressed. Several bits in the standard for floating point numbers are used for information like the sign and the exponent. While this information can be useful for other applications, when using an AE for compression all bits should be used in the most efficient way. To make sure that the bits are used in the most efficient way, the real numbers in the middle layer should be replaced by bits. That way, the AE can determine the most efficient way using these bits by itself. In order to get from real numbers to bits a Bernoulli distribution is used. By using bits instead of real numbers the information can be saved in the smallest space possible, as binary is the smallest form of data representation on a computer.

The Bernoulli distribution that is used needs to be able to deal with back propagation, to update the gradients of layers that came before the binarization. In order to achieve this, a method that generates samples that approximate the Bernoulli distribution is used. This estimator needs to return only zeroes and ones on the forward pass, and on the backwards pass the estimator needs to have a gradient so that the back propagation works. The real numbers that the encoder provides are used as the log probability of obtaining a one from the Bernoulli distribution, thus converting the real numbers from the encoder into either a one or a zero.

2.3 Corruption

An IAE is an AE that is able to deal with data being sent incrementally. If one want to take full advantage of the incrementally sent data the first bits sent should be the most important bits. Ideally, the AE is able to decode the data even when a portion of the data has not yet been sent.

The decoder of the AE can not have a smaller input, as this requires the structure of the AE to be changed. The solution to this problem is having the AE fill the missing part of the data with random bits. With these changes the AE can still maintain the same structure, but when receiving the real data it can replace the random bits with the real part of the data.

During training the AE should learn that during encoding and decoding the most important information about the images is stored in the first bits. The way this paper will approach this, is to use corruption on the encoded data during the training of the AE.

The encoder part of the AE returns the encoded form of the image that consist of bits. The corruption of these bits then works as follows. Before going to the decoder, a random number of bits is replaced with noise. The number of bits that will be replaced can be from zero to all of the encoded bits. The replaced bits will always be at the end of the encoded data. These bits are always located next to each other, meaning there are no gaps in between the replaced data. The steps to this method of corruption can be seen in figure 2.

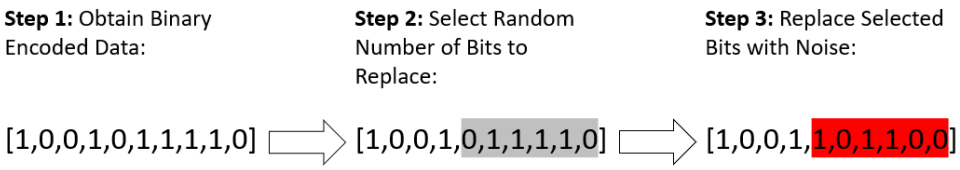


Figure 2: A schematic representation of the method of corruption.

This results in the first number of bits being replaced with noise less often than the final number of bits. Because the first bits are replaced with noise less often, the AE should learn to use these bits to store the most important information of the image. This in turn means that the AE learns that the bits closest to the end of the encoded image are the least important.

For the sake of comparison, this paper will also train an AE without any corruption and an AE with random corruption. The idea behind the AE with random corruption is similar to the IAE,

part of the encoded data is replaced with noise. However where the IAE replaces bits that are next to each other starting from the back, the random autoencoder (RAE) will replace random bits of the encoded data. This has the effect that every bit is equally likely to be replaced with noise. Therefore, the RAE will not be able to learn to prioritize any of the bits of the encoded data. Nevertheless, the RAE will have to try and learn to deal with the corruption.

3 Experiments

3.1 Data Sets

The AEs were trained three times on three different data sets. The most standard of the data sets is MNIST (Lecun et al. (1998)), which consist of images of handwritten digits. The other two data sets are variations of the MNIST data set. Whereas MNIST uses handwritten digits as the subject of the images, KMNIST (Clanuwat et al. (2018)) uses handwritten Japanese characters from the Kuzushiji alphabet. The final data set is Fashion-MNIST (Xiao et al. (2017)), which is a collection of thumbnail images of fashion items. All data sets share similar properties: the data sets contain 60000 training images and 10000 test images, the images are 28 by 28 pixels in size, they are saved in grayscale, the possible range of the pixels is between zero and one, and the data set contains labels to classify the images as one of ten types of objects. This makes it very easy to interchange between the different data sets.

3.1.1 MNIST

The MNIST data set was chosen because of its popularity in machine learning prototyping and its ease of use because of its availability in machine learning frameworks. The simplicity of the MNIST data set makes it a solid starting point for the experiments. If the experiments do not work with the MNIST data set, they very likely will not work on other more complicated data sets. Therefore the MNIST data set will be used as the initial data set when performing the experiments. The ten classes of the MNIST data set are the numbers ranging from zero to nine. In figure 3 a subset of the MNIST data set is displayed.

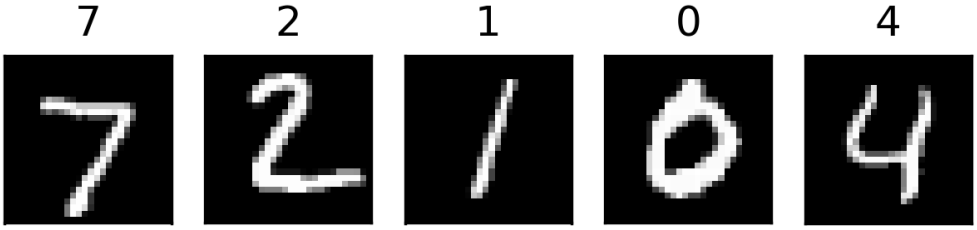


Figure 3: A subset of images from the MNIST data set with the class for the handwritten number displayed above the image.

3.1.2 Fashion-MNIST

The Fashion-MNIST data set was chosen to make sure that the AEs also functions with more complex images. While there are still ten classes, the variation within the classes is larger than in the MNIST data set. The classes in Fashion-MNIST represent a category of items, with a 1 now meaning that the image contains some form of pants. In the case of handwritten digits, the general shape of the digit does not vary too much. However, within the classes of the fashion item a lot of variation is possible. For example the class containing bags contains both handbags and carrier bags, which do not look that much alike. Another property of the Fashion-MNIST data set which potentially makes compression harder is the textures that are present on the fashion items. In figure 4 a subset of the Fashion-MNIST data set is displayed.

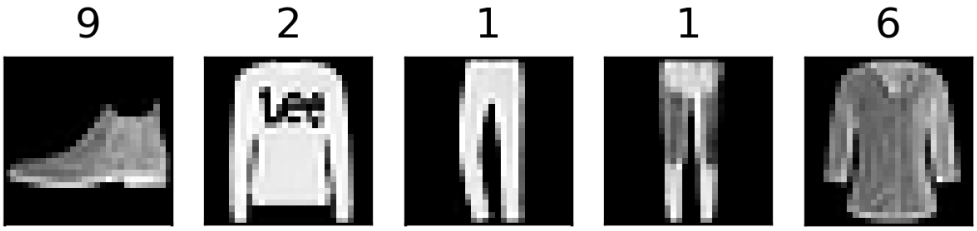


Figure 4: A subset of images from the Fashion-MNIST data set with the class for the fashion item displayed above the image.

3.1.3 KMNIST

The KMNIST data set serves a similar purpose as the Fashion-MNIST data set. It offers a set of images with more complex characters than the MNIST data. The images contains handwritten characters from the Kuzushiji alphabet, which is similar to a cursive alphabet in English. The simple Japanese alphabet (Hiragana) can be fully represented with 49 characters. To follow the same standard of having 10 classes in the data set multiple of the character types are mapped to the same class. For the purpose of compression there are still 49 types of characters. Due to the characters being cursive, there are even more possible ways to write these 49 characters. This combined with the complexity of Japanese characters provides a more challenging data set for compression. In figure 5 a subset of the KMNIST data set is displayed.

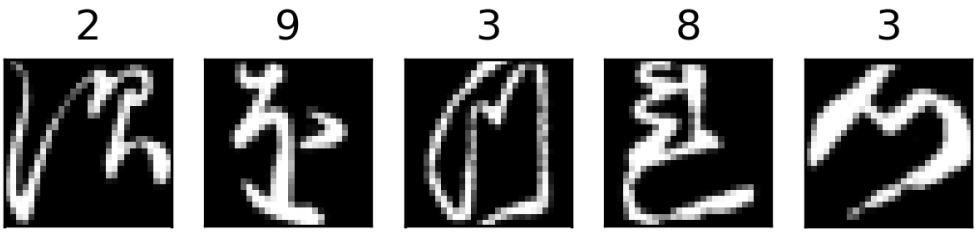


Figure 5: A subset of images from the KMNIST data set with the class for the 10 Hiragana characters displayed above the image.

3.2 Similarity Measures

When using lossy compression algorithms some quality is inevitability lost due to the way these algorithms work. In order to describe the quality of the reconstructed image, objective methods for assessing image quality are needed. The way of measuring how successful the compression was, is done by determining how similar the reconstructed image is to the original image. There are several similarity measures that can be used to evaluate the quality of the compression. This paper will use the following similarity measures: Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Normalized Root Mean Square Error (NRMSE) and Structural Similarity (SSIM).

3.2.1 Mean Squared Error

When computing the MSE of the reconstructed image, the measure takes the square of the difference found in each pixel (i.e. the squared errors) and averages (mean) these values (Wang and Bovik (2009)). The mathematical function describing the MSE can be found in equation 1. In the equation n is the number of pixels in the image, with x being the original image and y being the

reconstructed image.

$$\text{MSE}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{1}$$

The MSE is widely used in image processing. When comparing the effectiveness of new algorithms the MSE is most often the metric of choice. The popularity of the MSE is in large part due to its simplicity, the complexity for comparing a single pixel is only two summations and one multiplication.

3.2.2 Peak Signal-to-Noise Ratio

When working with images, the MSE is often replaced by the PSNR (Wang and Bovik (2009)). The PSNR is the ratio between the maximum (peak) strength of the signal and the error of the reconstructed image (noise). In the case of image compression, the peak strength is the highest possible value a pixel can have in the image. The mathematical function describing the PSNR can be found in equation 2

$$\begin{aligned}\text{PSNR}(x, y) &= 10 \cdot \log_{10}(\frac{p^2}{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}) \\ &= 10 \cdot \log_{10}(\frac{p^2}{\text{MSE}(x, y)})\end{aligned} \tag{2}$$

In this function the p is the peak strength of the signal, x is the original image and y is the reconstructed image. The error of the reconstructed image (i.e. the denominator) is equal to the MSE. Because the peak of the signal strength can be quite large, the PSNR is usually expressed on a logarithmic decibel scale, which is why the function has a logarithm around the ratio.

The PSNR is most useful if the images have different dynamic ranges. This is not the case for the data sets used. This means that the PSNR has the same properties as the MSE. So the main purpose of the PNSR is giving a recognizable metric for comparing the compression of images.

3.2.3 Normalized Root Mean Square Error

To compare different types of data the results from the MSE would not be valid, because the scale of the used data affects the outcome of the measure. To combat this problem, the data will have to be normalized, this is where the NRMSE comes in. The NRMSE takes the total squared error and divides it by the squared total value of the original image, then the root of this fraction is taken to obtain the NRMSE (Fienup (1997)).

$$\text{NRMSE(x,y)} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{\sum_{i=1}^{n} x_i^2}} \tag{3}$$

The NRMSE can be used to compare the error of the model for different data sets. Like the MSE, the NRMSE uses the total error in the calculation. Therefore the properties of the NRMSE are going to be similar to the MSE. However the normalization of the errors makes it so that the results obtained from different data sets can still be compared.

3.2.4 Structural Similarity

The MSE, the PSNR and the NRMSE all try to describe the error of the reconstruction by how much the pixels from the reconstruction differ from the pixels of the original image. Another vital part of human visual perception of image quality is the structural information of the image. The SSIM is a metric that attempts to capture this information in an objective measure (Wang et al.

(2004)). The SSIM is a combination of three comparison measurements: luminance, contrast and structure. These three components should be relative independent, meaning they do not affect each other.

The comparison measurements are constructed using the mean intensity (μ_z) and the standard deviation (σ_z) of an image. The formula for the mean intensity of an image z can be seen in equation 4 and the formula for the standard deviation of an image z is given in equation 5. In these formulas the n is the total number of pixels in the image.

$$\mu_z = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{4}$$

$$\sigma_z = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(z_i - \mu_z)^2} \tag{5}$$

To compute the SSIM the values of the mean intensity and the standard deviation are computed in a local 7 by 7 window which moves across the entire image one pixel at a time. For all these windows the three comparison metric are computed to get a local SSIM value. The mean of all these local SSIM values is the SSIM of the entire image. The formula for the luminance (l) of a window is described in equation 6, where x is the original window and y is the reconstructed window.

$$l(x,y) = \frac{\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{6}$$

In equation 6, C_1 is there to prevent instability when there are very small numbers in the denominator. To follow Wang et al. (2004) on the SSIM, the value for C_1 is defined as $C_1 = (K_1 \cdot p)^2$. In Wang et al. (2004) the value for $K_1 = 0.01$ and p is the maximum possible value of the image.

The formula for the contrast is very similar to the formula for the luminance. The mean intensity is replaced by the standard deviation. The resulting formula for the contrast (c) can be seen in equation 7.

$$c(x,y) = \frac{\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{7}$$

Like in equation 6 there needs to be a constant to prevent instability. In equation 7 this is C_2, which is defined as $C_2 = (K_2 \cdot p)^2$. In this case $K_2 = 0.03$, with p once again being the maximum possible value of the image.

The final comparison measurement structure also uses the covariance (σ_{zw}) of the original and the reconstructed image. The formula for the covariance of two images z and w can be seen in equation 8.

$$\sigma_{zw} = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \mu_z)(w_i - \mu_w) \tag{8}$$

The structure comparison formula can be seen in equation 9.

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{9}$$

The SSIM of a local window is a multiplication of the comparisons metrics, as can be seen in equation 10.

$$\text{SSIM}(x,y) = l(x,y) \cdot c(x,y) \cdot s(x,y) \tag{10}$$

If we define the constant in equation 9 to be $C_3 = \frac{C_2}{2}$, the equation for the SSIM can be simplified as seen in equation 11.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{11}$$

To obtain the SSIM of the original and the reconstructed images the local SSIM values will need to be averaged, this will be the total SSIM between the original and reconstructed images.

With the SSIM as an objective measure for the structural information, the evaluation of the compression quality will likely be closer to the human visual perception opinion. The SSIM is therefore a good measure to determine if the reconstruction is not only close to the original in euclidean space, but also visually close to the original image. However due to the increased amount of calculations necessary to calculate the SSIM it will not be as efficient as the previous similarity measures.

3.3 Autoencoder Architecture

The AEs in this paper all have the same architecture. As a result of the images in the data sets being 28 by 28, the input and output layer have to be 784 nodes so that all pixels are used as input. The AE is deep and undercomplete, meaning it has multiple hidden layers and a middle layer which is smaller than the input and output layers. Between the input layer and the middle layer (the encoder) are two hidden layers, one with 512 nodes and another with 256 nodes. The middle layer has 256 nodes as well. This means that after binarization the information containing the image is only 256 bits. Since an AE is symmetrical, the decoder also has two hidden layers with the same nodes. All the layers are fully connected, meaning that all nodes from the previous layer affect all the nodes in the next layer. The implementation of the AE was done using the PyTorch machine learning library for python (Paszke et al. (2017)). The general structure of the AE used in this paper can be seen in figure 6.

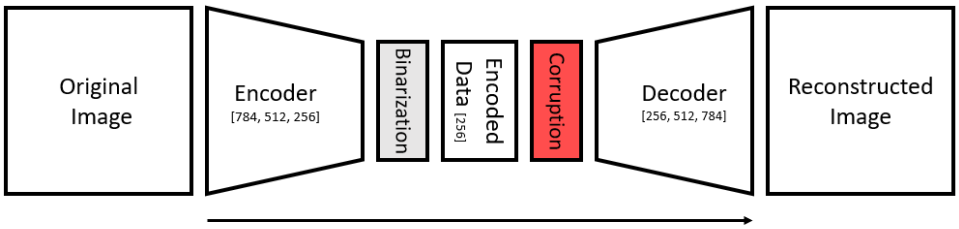


Figure 6: The general structure of the AE. The values between brackets denote the number of nodes in a layer.

To ensure that the AE is more than just linear operations an activation function is needed (Nwankpa et al. (2018)). The activation function used by this paper is the hyperbolic tangent (tanh). The activation function for tanh can be seen in equation 12.

$$\tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{12}$$

This activation function is used in the nodes of every layer except the final and middle layer. In the final layer the output should be between one and zero to produce an image that can be viewed. In order to achieve this the sigmoid function is used. The sigmoid activation function can be seen in equation 13.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

This activation function assures that the values of the output are between zero and one. This makes it possible to compare the reconstructed images with the original images, because the pixel values of the original images are also between zero and one.

3.4 Training

All models discussed in this paper were trained using batches of 64 images. The individual images are 28 by 28 in size. The algorithm used for stochastic gradient-based optimization during back propagation is *Adam* (Kingma and Ba (2014)). The models were trained for 50 epochs, meaning all training images were used 50 times during training. The binarization of the middle layer is done with a Bernoulli distribution which is able to deal with the back propagation. The Straight-Through Gumbel-Softmax Estimator from Jang et al. (2016) makes this possible. With this estimator the binary data has a gradient during back propagation, allowing the optimizer to also optimize the layers before the binarization. The temperature used for the estimator was $\tau = 1$. The outputs from the layer before the middle layer are taken as logarithmic inputs for the Straight-Through Gumbel-Softmax Estimator (Maddison et al. (2016), Jang et al. (2016)). The implementation of the Straight-Through Gumbel-Softmax Estimator used in this paper is part of the universal probabilistic programming language Pyro (Bingham et al. (2018)), which is build on the PyTorch machine learning library.

During training the AE will try to learn to reconstruct the original image. The loss function is there to determine how well this reconstruction is going during the training. The loss function used by this paper is the Mean Squared Error, which is a similarity measure described in section 3.2.1. This similarity measure was chosen to be the loss function because the MSE is standard in literature. The loss function can also be used to determine if the AE is still improving after each epoch. By plotting the loss against the epoch, the improvement over time becomes visible. With this plot it is possible to determine if the AE can still improve or if extra training will not yield any better results. The plots that are created during training can be seen in appendix A.

There are three different types of training for the models. During the training of the first AE no additions were made to the AE, resulting in an regular AE. The other AEs are trained with corruption of the encoded data. One of the AE is trained with the incremental corruption and the other with random corruption. With 256 bits for the encoded data, the number of bits replaced for each image can vary between 0 and 256. The noise that is used to replace the encoded bits is created using a Bernoulli distribution with a probability of 0.5 of getting a zero and subsequently also a probability of 0.5 of getting a one. This should result in noise with an equal distribution of zeros and ones.

3.5 Testing

After training the IAE should have learned to put the most important information about the image in the first bits. To verify if this is actually the case the AEs will be tasked with reconstructing the image with an increasingly smaller number of bits.

To simulate the missing data part of the data will be replaced with noise. One byte (8 bits) at a time the encoded data will be replaced with noise. The noise is created with a Bernoulli distribution and has an equal distribution of ones and zeros. After replacing part of the encoded data, the reconstruction is evaluated with the similarity measures that were discussed in section 3.2. The outcome of these similarity measures will be saved in a data frame for further use.

The replacement of the encoded data with noise is done in three ways. The first way is incremental corruption, starting from the back. This method of corruption is the same type of corruption that the IAE was trained to deal with. Thus the IAE should be able to deal with this corruption much better than the regular AE. The second type of corruption is the random corruption, which replaces random bits from the encoded data. This is the same type of corruption as the RAE is trained to deal with, therefore it should perform the best out of the three AEs. The final type of corruption is also incremental, but unlike the other type of incremental corruption this one starts from the front. This should serve as a excellent way to show that the IAE is saving the most important data in the first bits.

With the image quality metrics for different percentages of corruption, a plot can be made showing the curve of quality degradation. The curves for the different AEs can than be compared to show how they deal with the corruption.

To give more than just a mathematical interpretation of how the AEs deal with corruption the reconstructed images will also be shown for the different corruption percentages. This allows for human assessment of the image quality.

4 Results

4.1 Objective Similarity Measures

In order to compare different compression methods it is important to have an objective measure for image quality, in this papers case the similarity measures described by section 3.2. By comparing the three different AEs based on an objective measure it is possible to make mathematical conclusions about the image quality after reconstruction. In the following section image quality is compared to the corruption percentage based on the MSE and the SSIM. The results of the PSNR and the NRMSE look very similar to the MSE and SSIM. The plots of the PSNR and the NRMSE have therefore been omitted here, but can still be seen in appendix B. The data set that was used for all figures in section 4.1 was the MNIST data set, which was the most basic of the three data sets. In this section the results of all types of corruption will be shown. The results for the Fashion-MNIST and KMNIST data sets can be seen in appendix B.

4.1.1 Incremental Corruption

In figure 7 the MSE error is plotted against the corruption percentage for all three types of AEs. As the metric is the MSE, the lower the score the better the quality. The regular AE starts off with the lowest MSE of the three models of AEs, but when the corruption increases the regular AE has the steepest decline in image quality. While the IAE starts as the second best, the increase of corruption has the least effect on the IAE. The RAE has the highest MSE with no corruption, however the increasing corruption has less of an effect than it did on the regular AE.

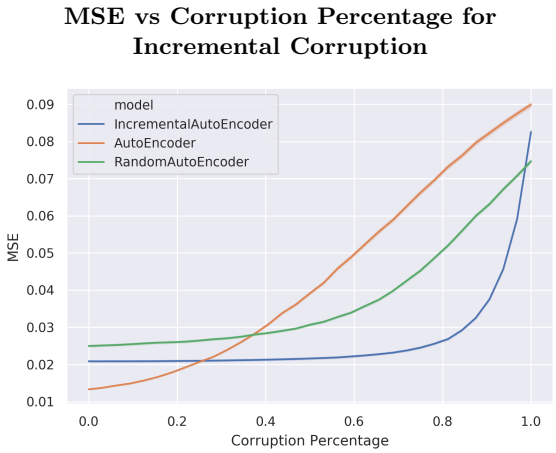


Figure 7: The image quality as described by the MSE plotted against the corruption percentage, with incremental corruption on the MNIST data set.

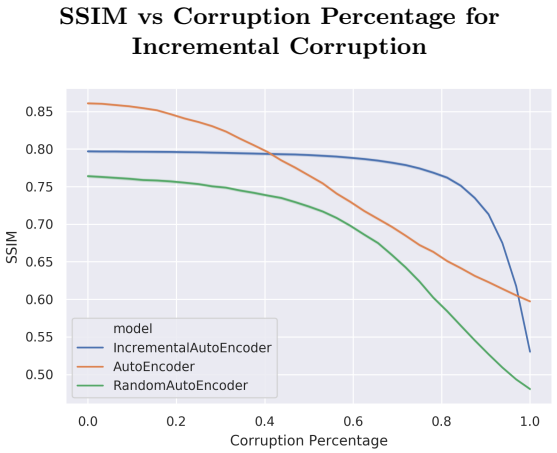


Figure 8: The image quality as described by the SSIM plotted against the corruption percentage, with incremental corruption on the MNIST data set.

In figure 8 the same is done but instead of the MSE, the similarity metric used is the SSIM. With the SSIM a higher score means better image quality. The regular AE again starts as the best performing model, but is overtaken by the IAE as the corruption increases. While the RAE

obtained a lower MSE at the higher corruptions, this is not the case with the SSIM. For all corruption percentages the RAE obtained the lowest SSIM.

4.1.2 Reverse Incremental Corruption

In figure 9 the effect of changing the corruption type from incremental to reverse incremental can be seen. In this figure the similarity metric used is the MSE. This change has minimal effect on the regular and RAE, however the effect on the IAE is apparent. While the begin and end point for it are the same, the curve that the IAE follows is completely different. With only minimal corruption the IAE becomes the worst scoring model.

MSE vs Corruption Percentage for Reverse Incremental Corruption

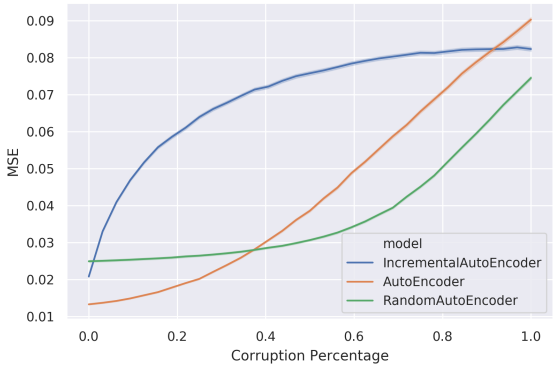


Figure 9: The image quality as described by the MSE plotted against the corruption percentage, with reverse incremental corruption on the MNIST data set.

SSIM vs Corruption Percentage for Reverse Incremental Corruption

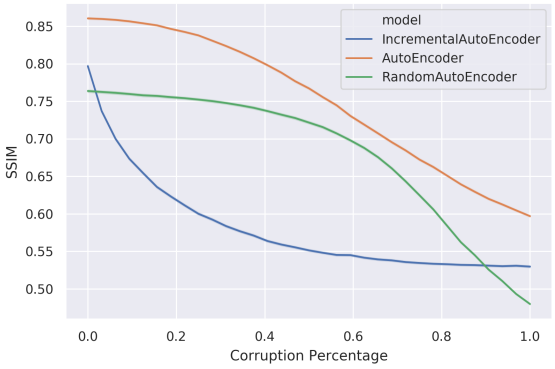


Figure 10: The image quality as described by the SSIM plotted against the corruption percentage, with reverse incremental corruption on the MNIST data set.

With the SSIM the effect of changing from incremental corruption to reverse incremental corruption is the same. This can be observed in figure 10. The regular AE and RAE are largely unaffected, but the IAE has a different curve. With a small amount of corruption the IAE obtains the lowest SSIM. The regular AE performs the best out of the three models for every corruption percentage.

4.1.3 Random Corruption

Finally in figure 11 the MSE of the three AE models is plotted against the corruption percentage with random corruption. As with the other types of corruption, if there no corruption the regular AE performs best, the IAE second best and the RAE performs worst. With higher corruption this order flips, the RAE performs best, the IAE remains second best and the regular AE performs the worst.

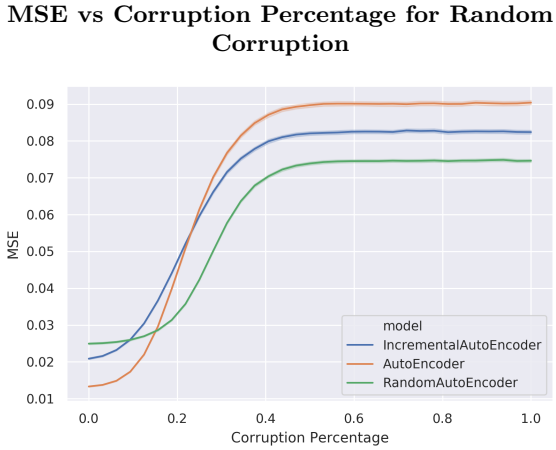


Figure 11: The image quality as described by the MSE plotted against the corruption percentage, with random corruption on the MNIST data set.

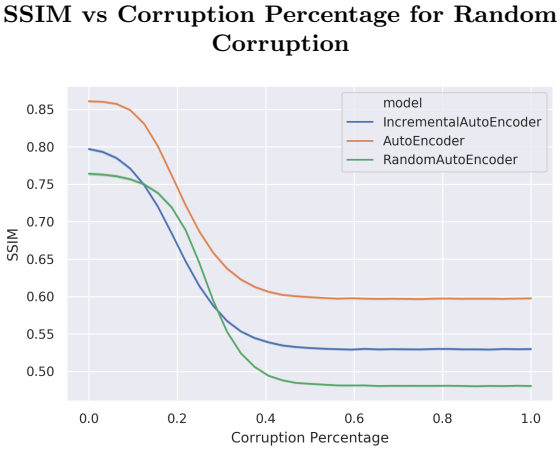


Figure 12: The image quality as described by the SSIM plotted against the corruption percentage, with random corruption on the MNIST data set.

In figure 12 the SSIM is plotted against the corruption percentage with random corruption. The order of the AEs based on image quality starts the same as for the MSE, but for the SSIM the order does not changes as it does for the MSE. The RAE performs slightly better than the IAE for corruption percentages between 0.1 and 0.3, but still ends as the worst performing model for the higher corruption percentages.

To demonstrate that some data sets are harder than others for the AE, the MSE is plotted against the corruption percentage for the reconstruction by the IAE for all three data sets in figure 13. The SSIM after reconstruction by the IAE for all three data sets is also plotted against the corruption percentage for all three data sets. This can be seen in figure 14.

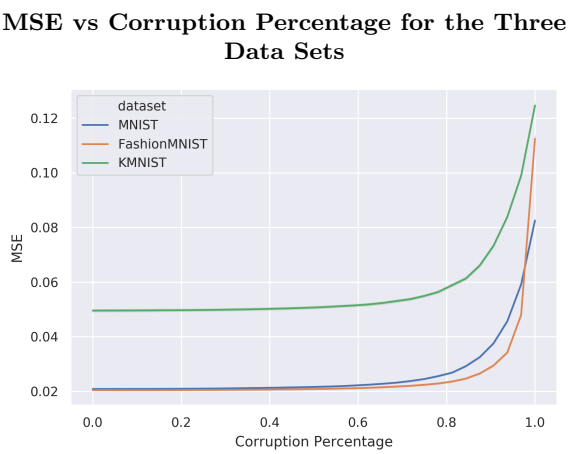


Figure 13: The image quality as described by the MSE plotted against the corruption percentage, comparing the reconstruction by the IAE for all three data sets.

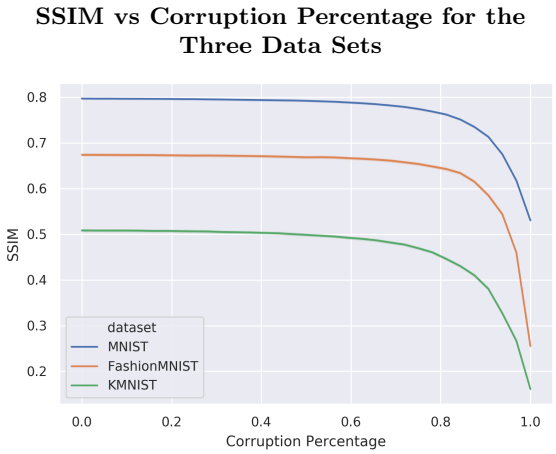


Figure 14: The image quality as described by the SSIM plotted against the corruption percentage, comparing the reconstruction by the IAE for all three data sets.

When comparing the reconstruction errors of the different data sets, the MSE values for the KMNIST data set are much larger than the other two data sets. The Fashion-MNIST and the MNIST data sets perform comparable in term of the MSE. When comparing the data sets based on the SSIM, the MNIST data set obtains the best results, followed by the Fashion-MNIST data set, and the KMNIST data set was the hardest to reconstruct.

4.2 Visualization

With the visualization of the images during the corruption it is possible to get a sense of what the drop in image quality described by the objective quality measure actually represent in changes to the image. This allows for human perception to evaluate the image quality for the different corruption percentages. In this section only the comparison between the different AEs for the incremental corruption is shown. For the full visualization of all the AEs for all the corruption types see appendix C.

4.2.1 MNIST Corruption

Figure 15 shows the reconstruction of the images from the MNIST data set after incremental corruption. In this figure the different AEs can be compared when trying to reconstruct images of the handwritten digits seven, two and one for various corruption percentages.

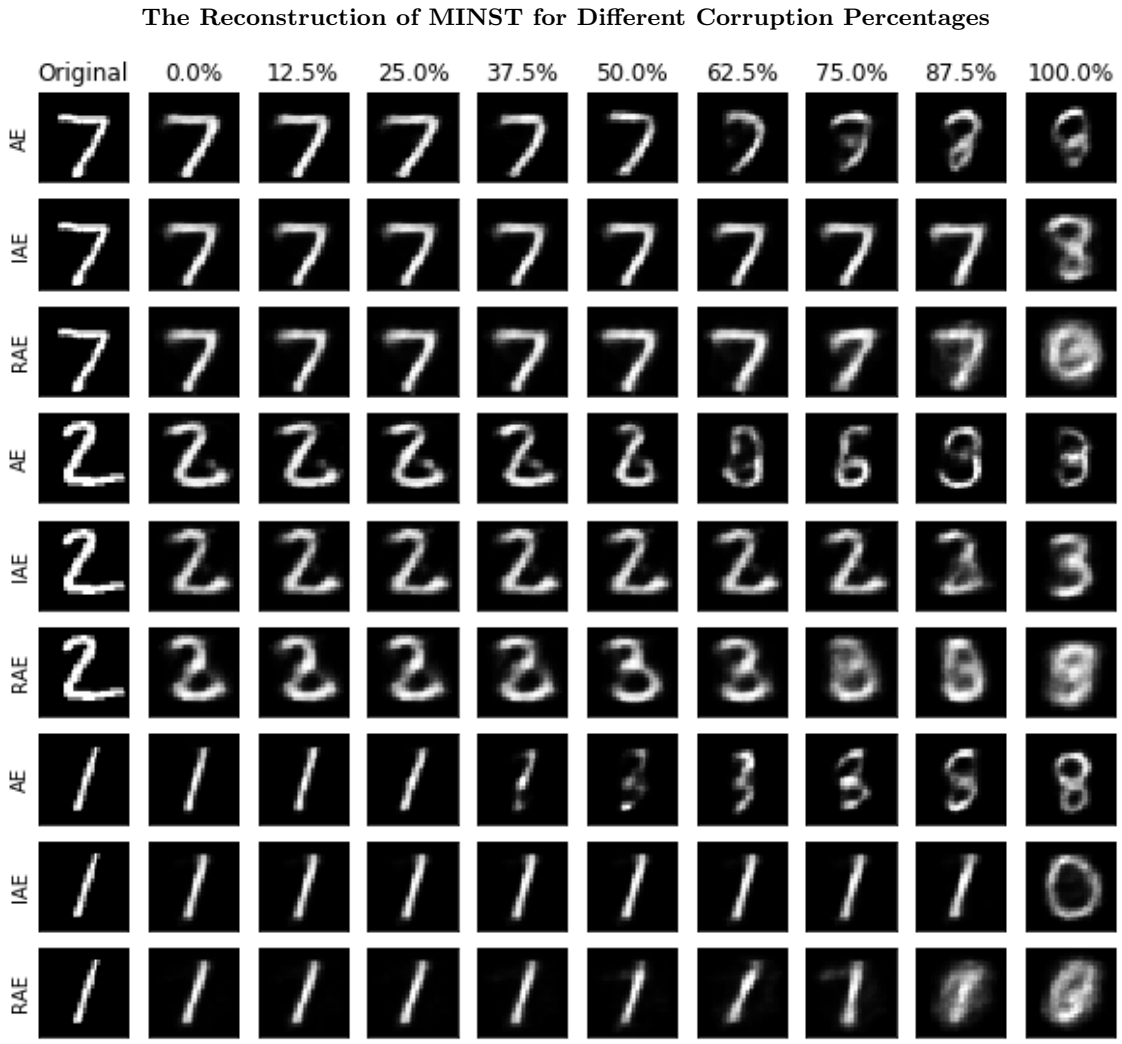


Figure 15: Visualization of the reconstructed images from the MNIST data set after incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

4.2.2 Fashion-MNIST Corruption

In figure 16 the effect of the incremental corruption on images from the Fashion-MNIST data set can be seen. In this figure the different AEs can be compared when trying to reconstruct images of a shoe, a shirt and pants for various corruption percentages.

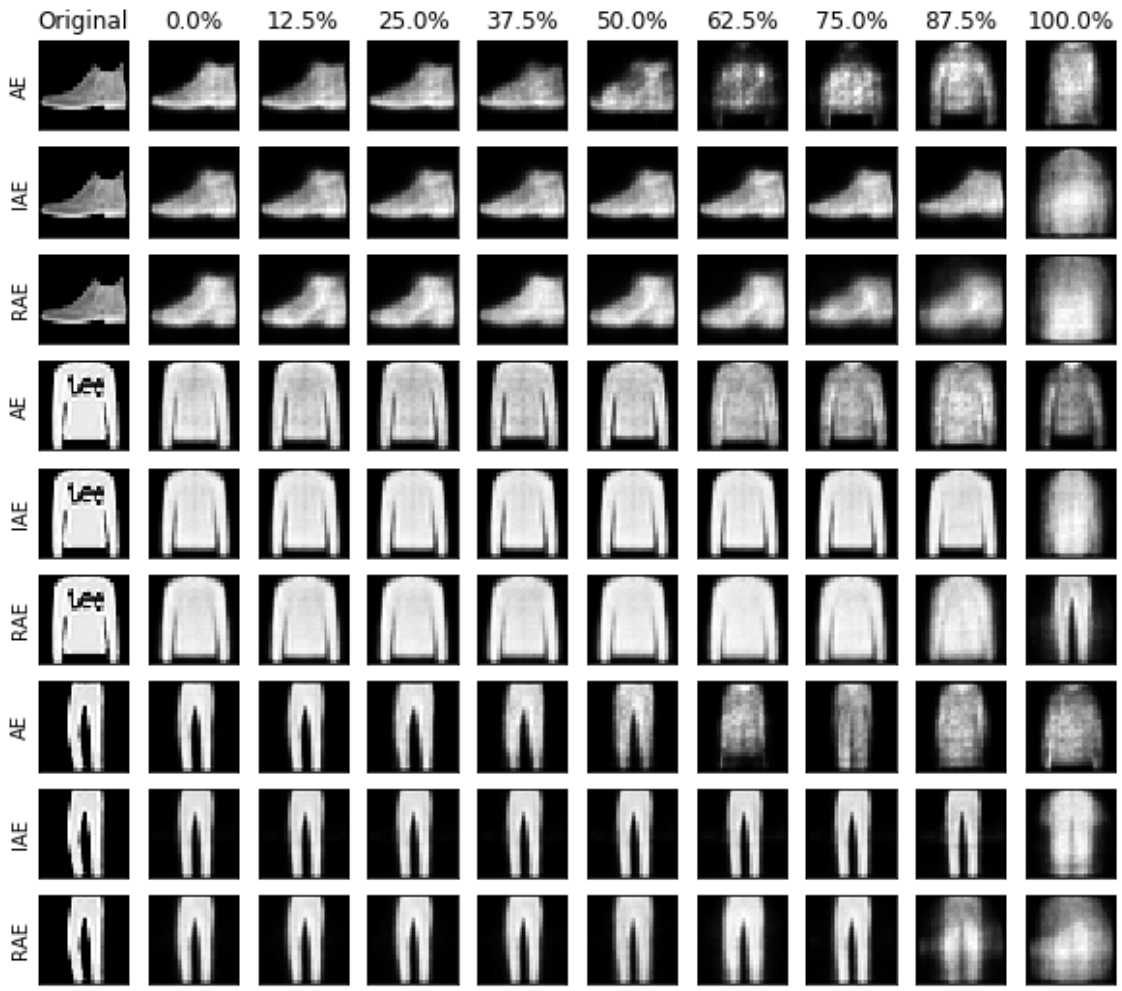


Figure 16: Visualization of the reconstructed images from the Fashion-MNIST data set after incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

4.2.3 KMNIST Corruption

Finally the visualization of the reconstructions by three different AEs for the images from the KMNIST data set can be seen in figure 17. In this figure the reconstruction of three Japanese characters is displayed.

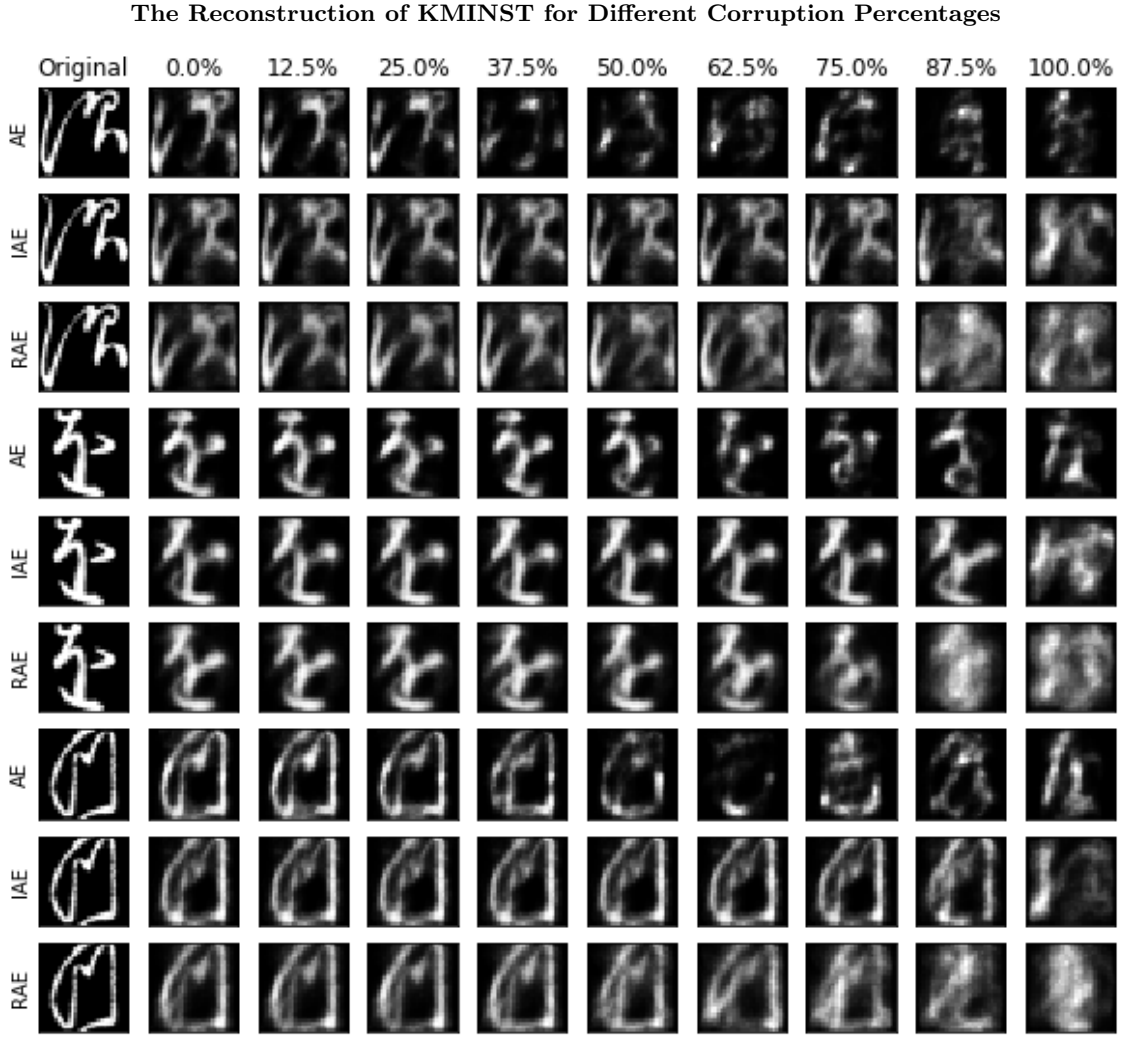


Figure 17: Visualization of the reconstructed images from the KMNIST data set after incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

5 Discussion

5.1 Interpretation

Although the IAE started off preforming worse than the regular AE, it was able to deal with the incremental corruption in a much better way (see section 4.1). The MSE and SSIM almost did not change until almost half of the bits in the encoded data were replaced with noise. This indicates that the bits that had been replaced held very little important information about the original image. This would mean that the IAE had learned to store the most important information about the image in the first bits.

This indication is confirmed by how the IAE dealt with the reverse incremental corruption. As this meant replacing the first, and thus most important, bits with noise. With these bits replaced the IAE had a much harder time reconstructing the original images, resulting in a very low score for both the SSIM and MSE. This also points to the IAE learning to store more important information in bits that are closer to the start.

The visualization of the reconstructed images (see section 4.2) tells a similar tale, with the regular

AE deviating from the original image at a lower percentage of corruption. The images reconstructed by the IAE at higher corruption percentages all resembled the original reconstruction, meaning that the replaced bits had very little to do with the reconstruction of the image.

This does seem to suggest however that the final bits are not used at all by the IAE in the reconstruction of the images, which is not what the intended result was. All additional bits given to the IAE should improve the quality of the reconstructed image. If the final bits add no extra quality to the image this is no longer the case. This means that the final bits do not even have to be sent to recreate the image, indicating that the image could have been saved in fewer bits then currently used.

An interesting observation is that the RAE had a lower MSE score than the regular AE at the higher corruption percentages, but it did not have a higher SSIM score. As the SSIM and the MSE calculate the similarity in a different way this discrepancy is not totally unexpected. It is however a great example for why it is important to have multiple measures for the quality. The fact that the IAE out preforms the regular AE in all measures is confirmation that it is able to deal with corruption better than the regular AE.

5.2 Evaluation

While this method of training an AE to be able to deal with incrementally sent data seems promising there are still some issues. For starters, the image quality of the reconstructed images with no data missing is lower for the IAE than for the regular AE. Evidently the ability to deal with incremental data comes at the price of loss in image quality. The loss in image quality might be something that deters people from using the IAE.

Another improvement that can easily be made is to train the models for more epochs, allowing the AEs to learn to reconstruct the images even better. After 50 epochs the improvements in the loss function were minimal, however this does not mean that the image quality could not still be improving. A small change in the objective quality measure may result in a big difference in human visual perception.

Although the AEs work well for the data set on which they were trained, they do not work for general compression. This is a problem that most AEs have, but still one to consider when comparing the AEs to general compression algorithms. For the AEs in this paper, their specificity becomes apparent when the entire encoded data had been replaced with random bits. Even though the AE would produce an image that looked nothing like the original image, the image it create could still be part of the data set on which the AE was trained (see section 4.2). It may be able to avoid this issue by training on a larger data set with more general images.

The structure of the AEs is also something that could be improved upon. The number of hidden layers and the number of nodes in these layers have not been tested to be the optimal value. This means that changing the number of layers and the number of nodes might produce better results. Especially changing the middle layer is something that will have a large impact on the outcome of the AE. A smaller middle layer means more compression, but there is no certainty that an AE with a lot smaller middle is able to learn how to deal with incremental data. The smaller number of bits in the middle layer might mean that the AE can not store the most important information in the first bits. All bits could be necessary to accurately describe the image, not allowing any bits to be more important than the others.

5.3 Follow-up Research

As this paper was mainly a proof of concept, there are still a lot of research topics that can be explored as a follow-up to this research. The images that were used in this research were all in grayscale. As most images are no longer just black and white, it would be interesting to investigate if this method for making AEs incremental also works for images with color.

Further research should also be conducted to find if this method also works for larger images. This could mean just having a larger input layer. However this results in a much larger AE to train. A possible way to avoid this is, is to split the image into several smaller images which can then be

reconstructed by the smaller AE. For an larger image one might also consider using an AE that uses convolution. Figuring out which of these methods of dealing with larger images works with the method of making an AE incremental could be a potential further research topic. And if one of these methods does work, determining which of the methods yields the best results should also be explored.

It would also be interesting to see if a similar method of making an AE learn how to deal with incremental data could also work for a non-binary representation of the image. Instead of using ones and zeroes as a representation for the image the AE would use real numbers. This means that the noise should also be comprised of real numbers.

As stated in section 5.2 the structure of the AE should also still be tested. Mainly, with what size of middle layer would the method to make AEs incremental still work. The smaller the size the more important the individual bits become, making it harder to treat some bits as less important then others. This could potentially make this method unsuitable for higher levels of compression.

5.4 Conclusion

The goal of this paper was to evaluate to what extent the compression using an autoencoder can be made incremental. Initially the regular autoencoder achieved higher image quality according to all measures. However, as hypothesized, the incremental autoencoder obtained better scores on the measures for image quality when the last bit of the encoded data was missing. When, instead, the first bit of encoded data is missing the incremental autoencoder performed much worse than the regular autoencoder. This is a strong indication that the incremental autoencoder learned to put the most important information about an image in the first bits. The method for creating an incremental autoencoder proposed in this paper shows great potential for incremental compression in the future.

Bibliography

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. (2018). Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*.

Charte, D., Charte, F., García, S., del Jesus, M. J., and Herrera, F. (2018). A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion*, 44:78–96. doi:10.1016/j.inffus.2017.12.007.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical japanese literature. arXiv:1812.01718.

Fienup, J. R. (1997). Invariant error metrics for image reconstruction. *Appl. Opt.*, 36(32):8352–8357. doi:10.1364/AO.36.008352.

Gao, K. (2018). Architecture for hard disk drives. *IEEE Magnetics Letters*, 9:1–5. doi:10.1109/LMAG.2018.2789888.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507. doi:10.1126/science.1127647.

Jang, E., Gu, S., and Poole, B. (2016). Categorical Reparameterization with Gumbel-Softmax. *arXiv e-prints*. arXiv:1611.01144.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*. arXiv:1412.6980.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv e-prints*. arXiv:1312.6114.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. doi:10.1109/5.726791.

Maddison, C. J., Mnih, A., and Whye Teh, Y. (2016). The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv e-prints*. arXiv:1611.00712.

Mittal, S. and Vetter, J. S. (2016). A survey of architectural approaches for data compression in cache and main memory systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1524–1536. doi:10.1109/TPDS.2015.2435788.

Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv e-prints*. arXiv:1811.03378.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Roelofs, G. (1999). *PNG: The Definitive Guide*. Creating and programming portable networks graphics. O'Reilly.

Sayood, K. (2017). *Introduction to Data Compression*. The Morgan Kaufmann Series in Multimedia Information and Systems. Elsevier Science.

Skodras, A., Christopoulos, C., and Ebrahimi, T. (2001). The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58. doi:10.1109/79.952804.

Tan, C. C. and Eswaran, C. (2011). Using autoencoders for mammogram compression. *Journal of Medical Systems*, 35(1):49–58. doi:10.1007/s10916-009-9340-3.

Wallace, G. K. (1992). The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv. doi:10.1109/30.125072.

Wang, Z. and Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117. doi:10.1109/MSP.2008.930649.

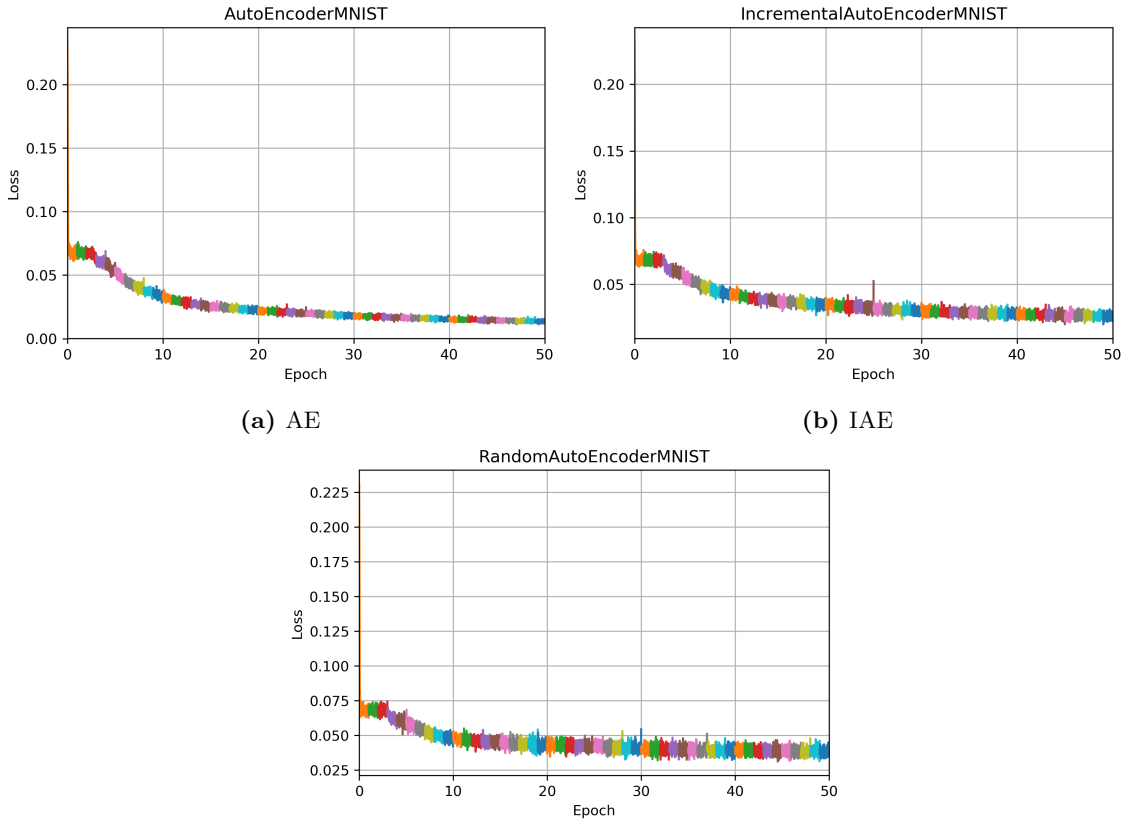
Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612. doi:10.1109/TIP.2003.819861.

Wood, R. (2009). Future hard disk drive systems. *Journal of Magnetism and Magnetic Materials*, 321(6):555 – 561. doi:10.1016/j.jmmm.2008.07.027.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747.

Xie, J., Xu, L., and Chen, E. (2012). Image denoising and inpainting with deep neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 341–349. Curran Associates, Inc.

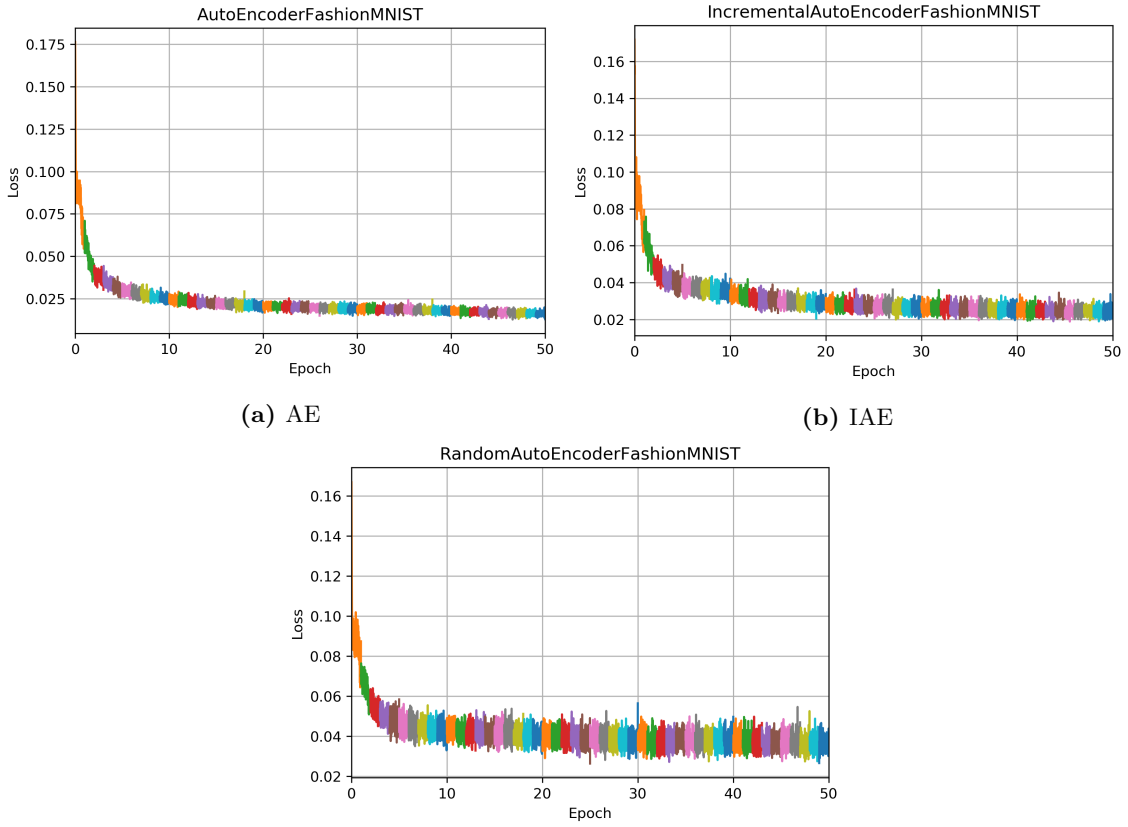
A Training plots



(a) AE **(b)** IAE

(c) RAE

Figure 18: The training curves plotting the loss against the epochs for the different autoencoders for the MNIST data set. Every color represents an epoch of training.

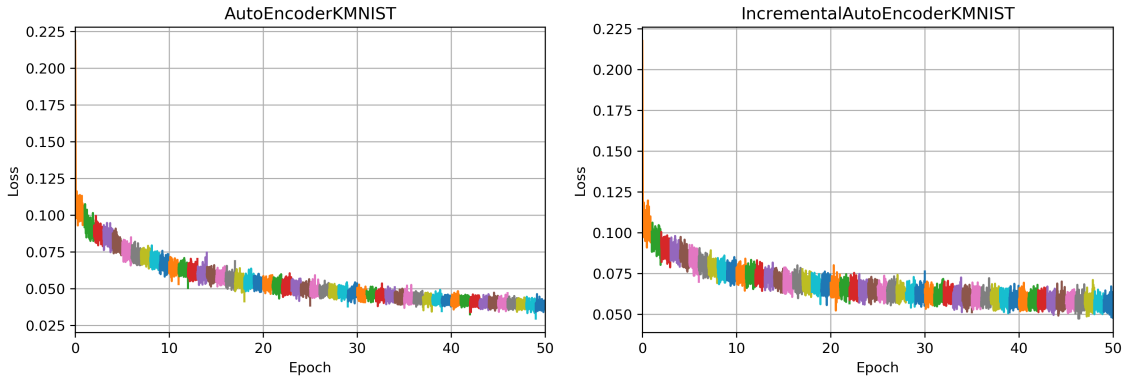


(a) AE

(b) IAE

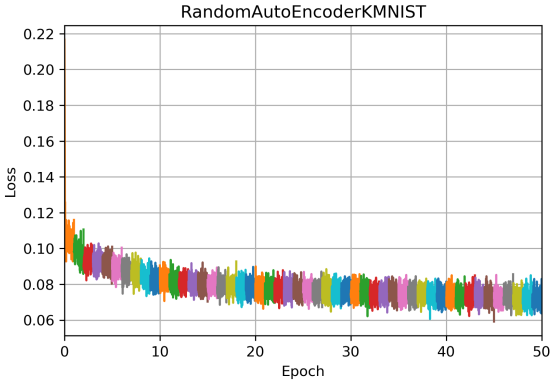
(c) RAE

Figure 19: The training curves plotting the loss against the epochs for the different autoencoders on the Fashion-MNIST data set. Every color represents an epoch of training.



(a) AE

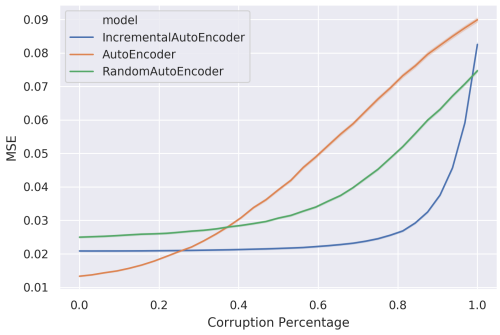
(b) IAE



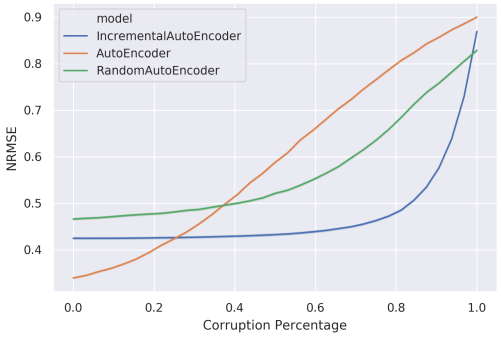
(c) RAE

Figure 20: The training curves plotting the loss against the epochs for the different autoencoders for the KMNIST data set. Every color represents an epoch of training.

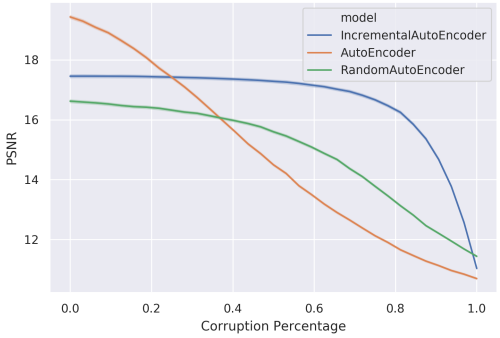
B Image Quality Measures Graphs



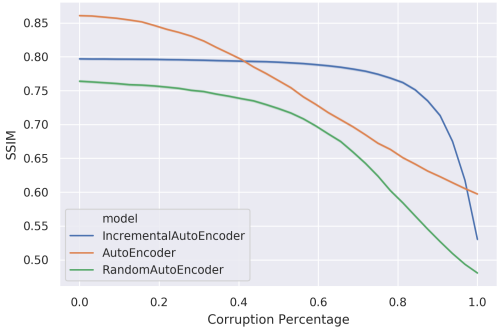
(a) MSE



(b) NRMSE

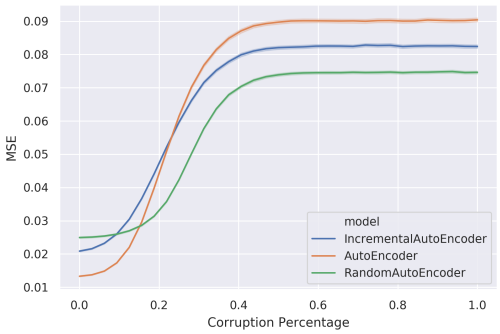


(c) PNSR

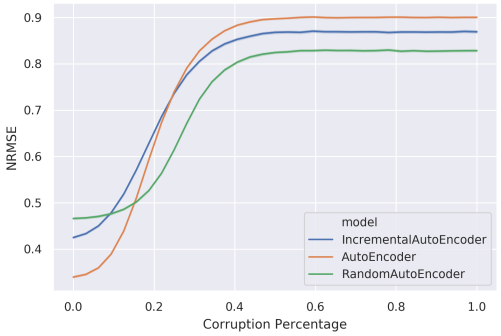


(d) SSIM

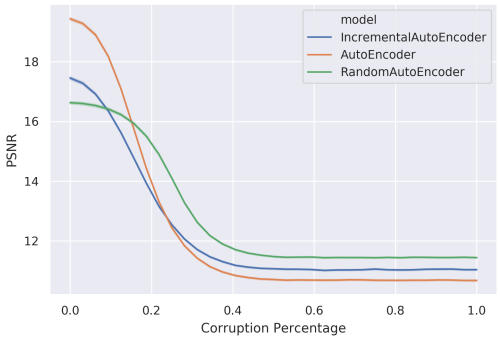
Figure 21: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with incremental corruption on the MNIST data set.



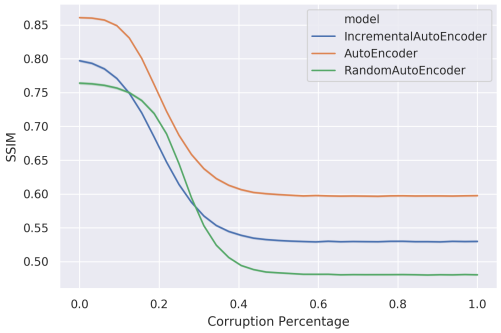
(a) MSE



(b) NRMSE

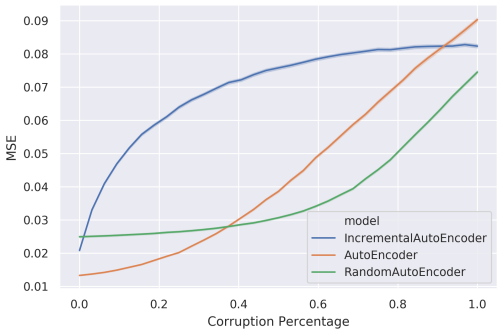


(c) PNSR

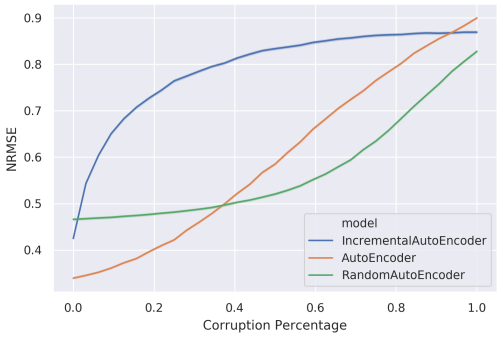


(d) SSIM

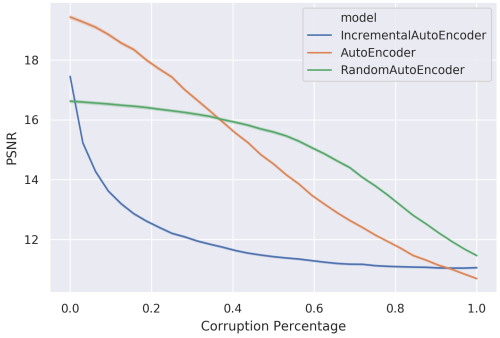
Figure 22: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with random corruption on the MNIST data set.



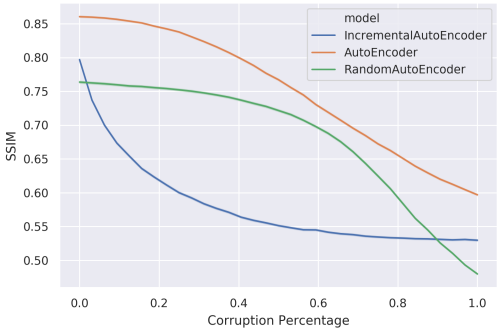
(a) MSE



(b) NRMSE

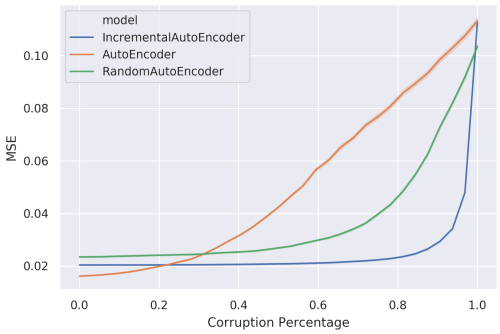


(c) PNSR

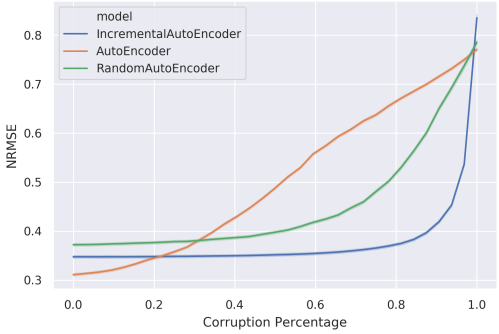


(d) SSIM

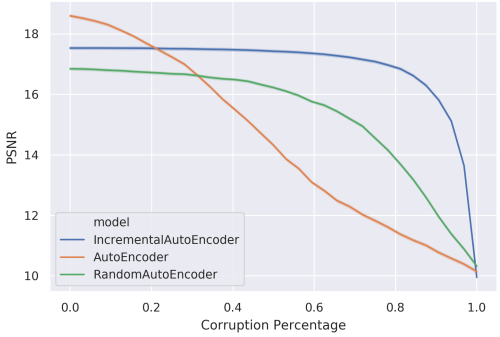
Figure 23: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with reverse incremental corruption on the MNIST data set.



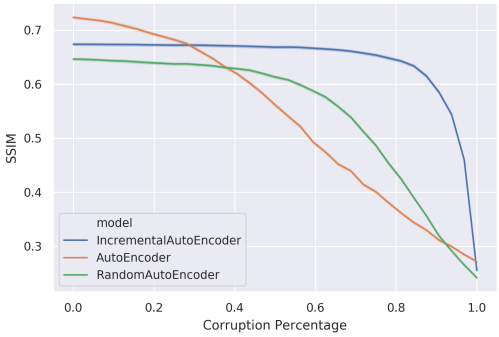
(a) MSE



(b) NRMSE

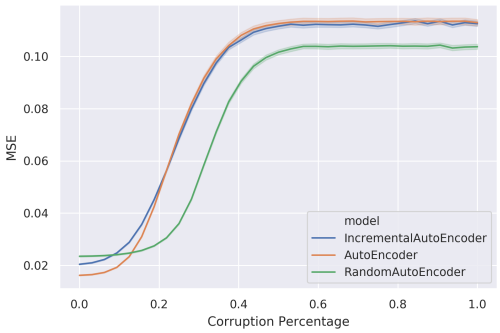


(c) PNSR

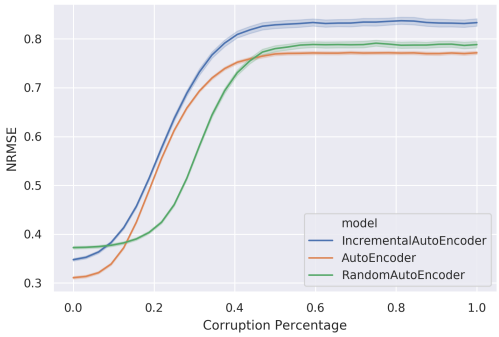


(d) SSIM

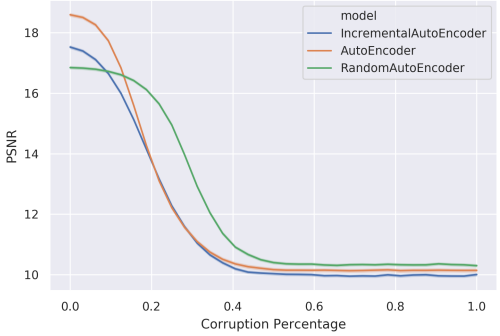
Figure 24: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with incremental corruption on the Fashion-MNIST data set.



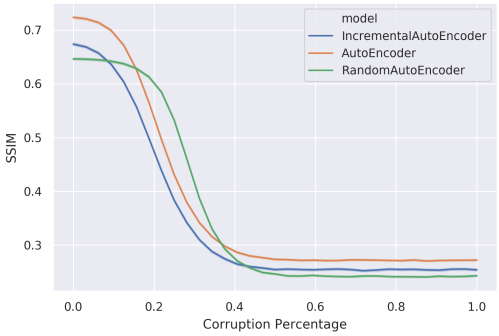
(a) MSE



(b) NRMSE

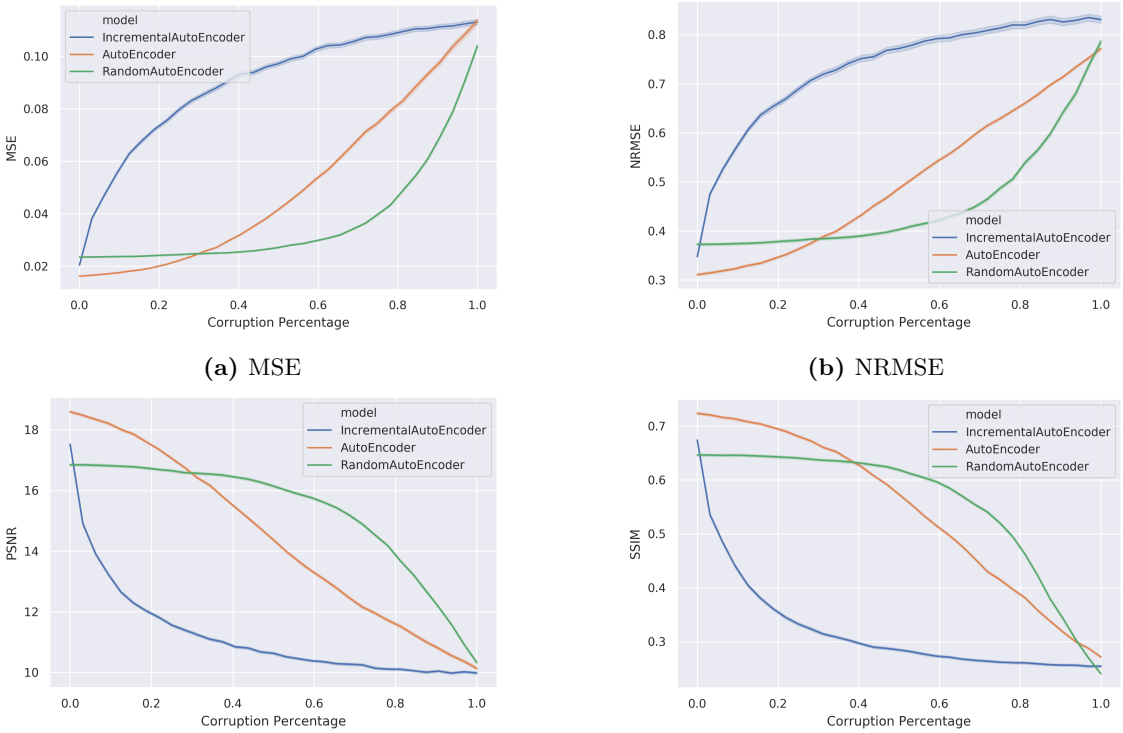


(c) PNSR



(d) SSIM

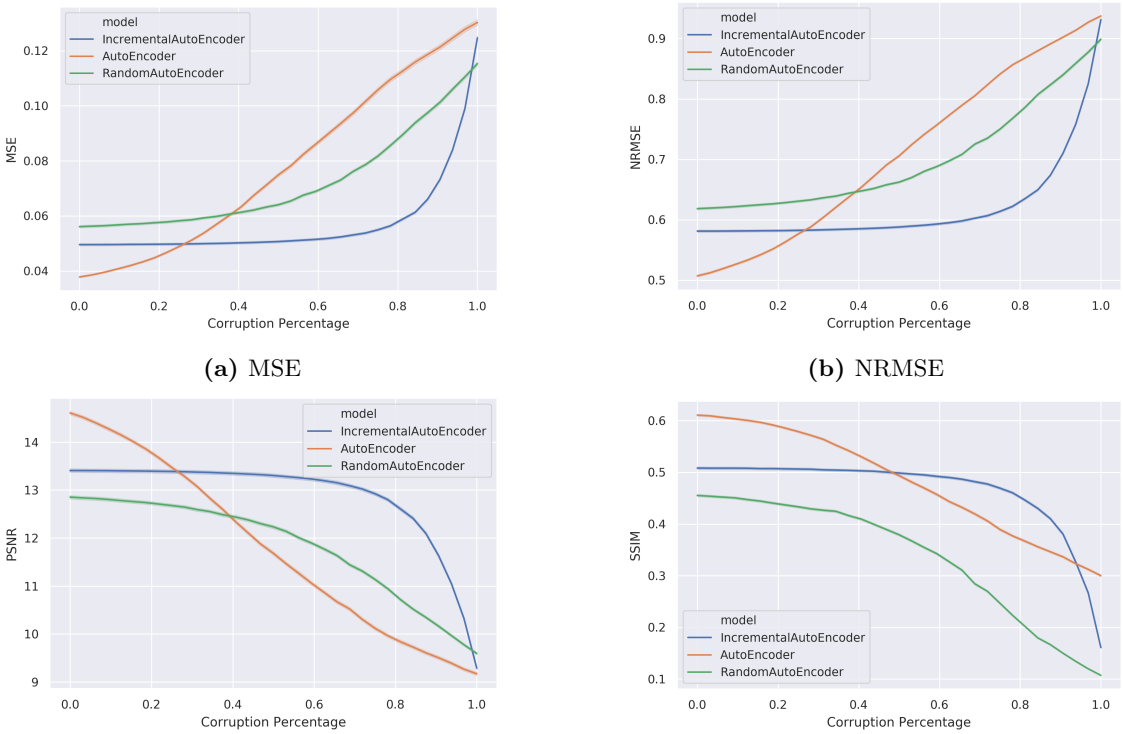
Figure 25: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with random corruption on the Fashion-MNIST data set.



(a) MSE **(b)** NRMSE

(c) PNSR **(d)** SSIM

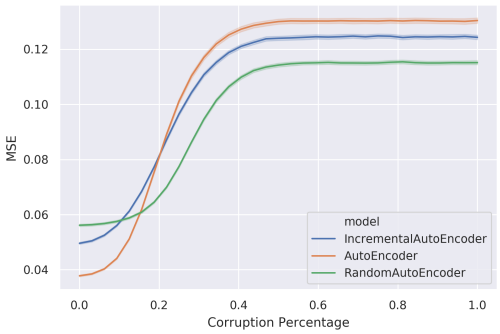
Figure 26: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with reverse incremental corruption on the Fashion-MNIST data set.



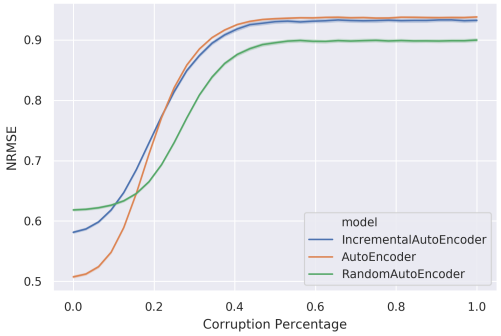
(a) MSE **(b)** NRMSE

(c) PNSR **(d)** SSIM

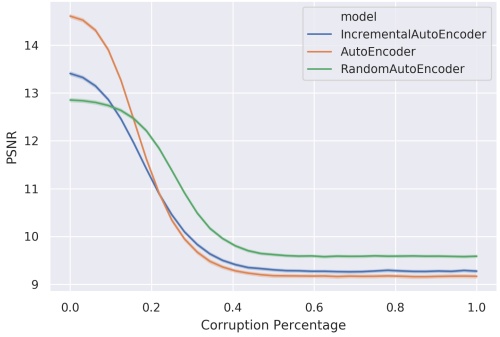
Figure 27: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with incremental corruption on the KMNIST data set.



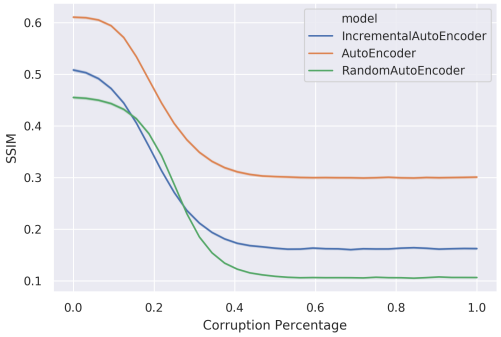
(a) MSE



(b) NRMSE

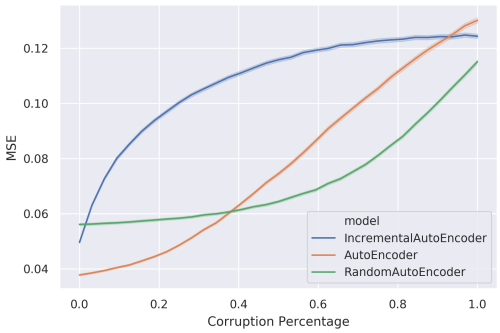


(c) PNSR

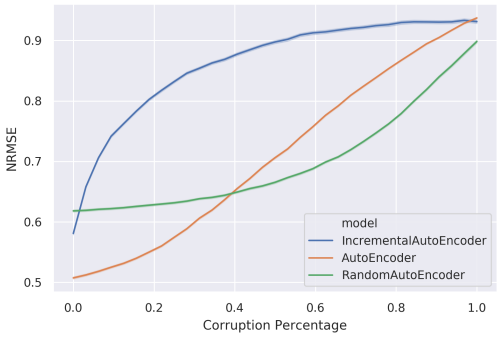


(d) SSIM

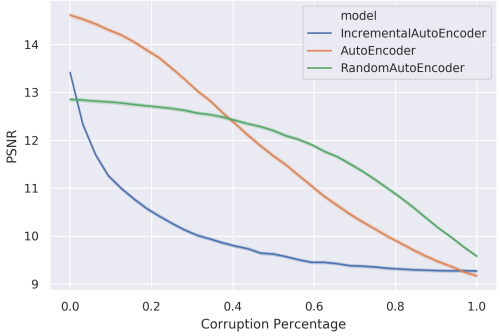
Figure 28: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with random corruption on the KMNIST data set.



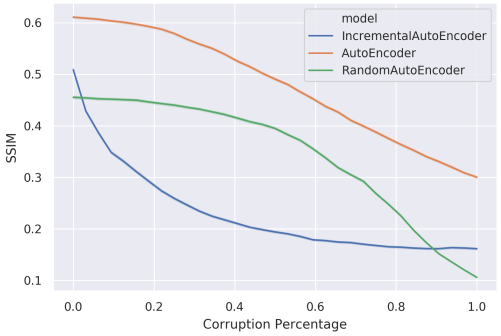
(a) MSE



(b) NRMSE



(c) PNSR



(d) SSIM

Figure 29: Graphs of the image quality according to the similarity measures plotted against the corruption percentage, with reverse incremental corruption on the KMNIST data set.

C Image Quality Visualization

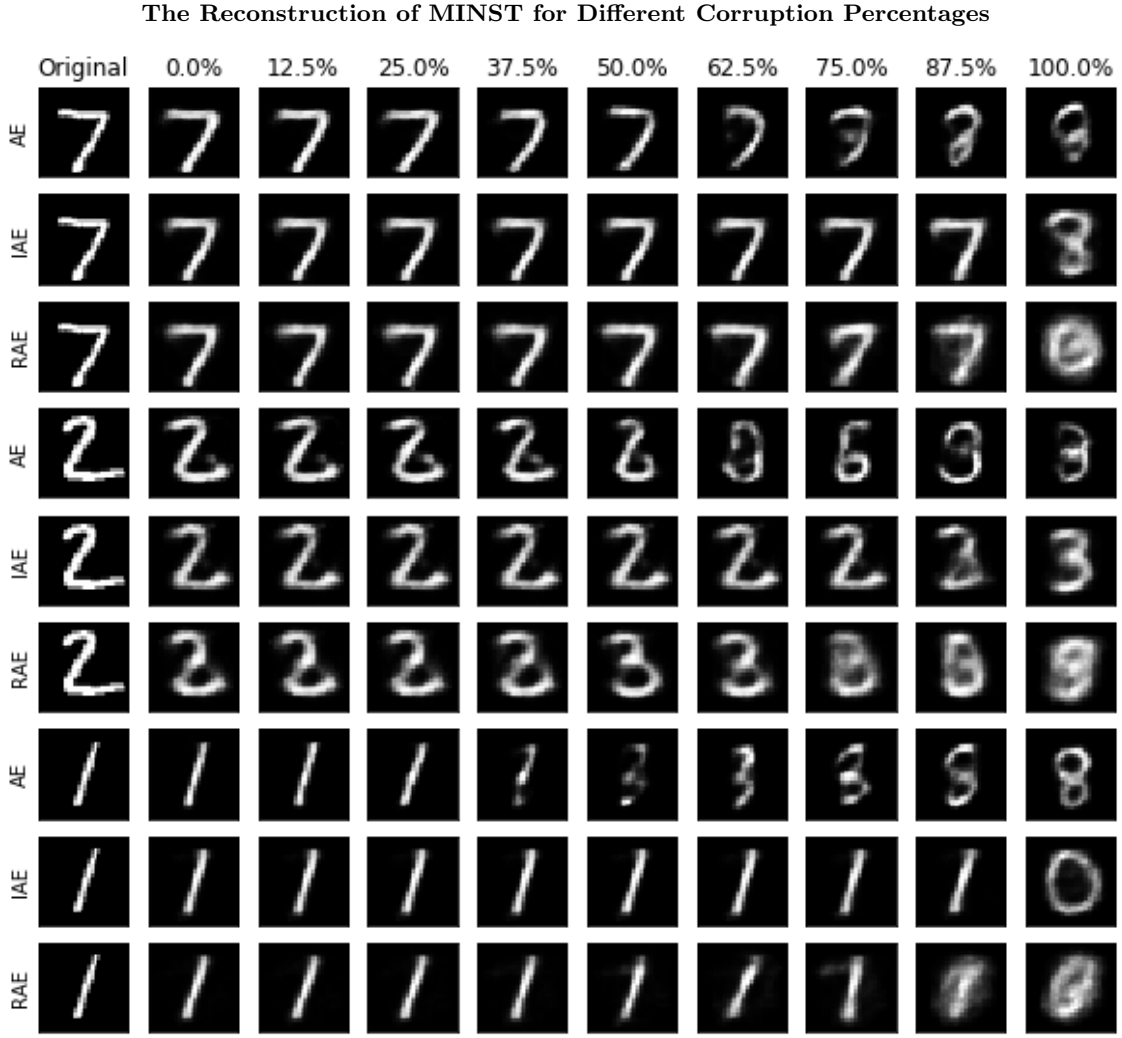


Figure 30: Visualization of the reconstructed images from the MNIST data set after incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

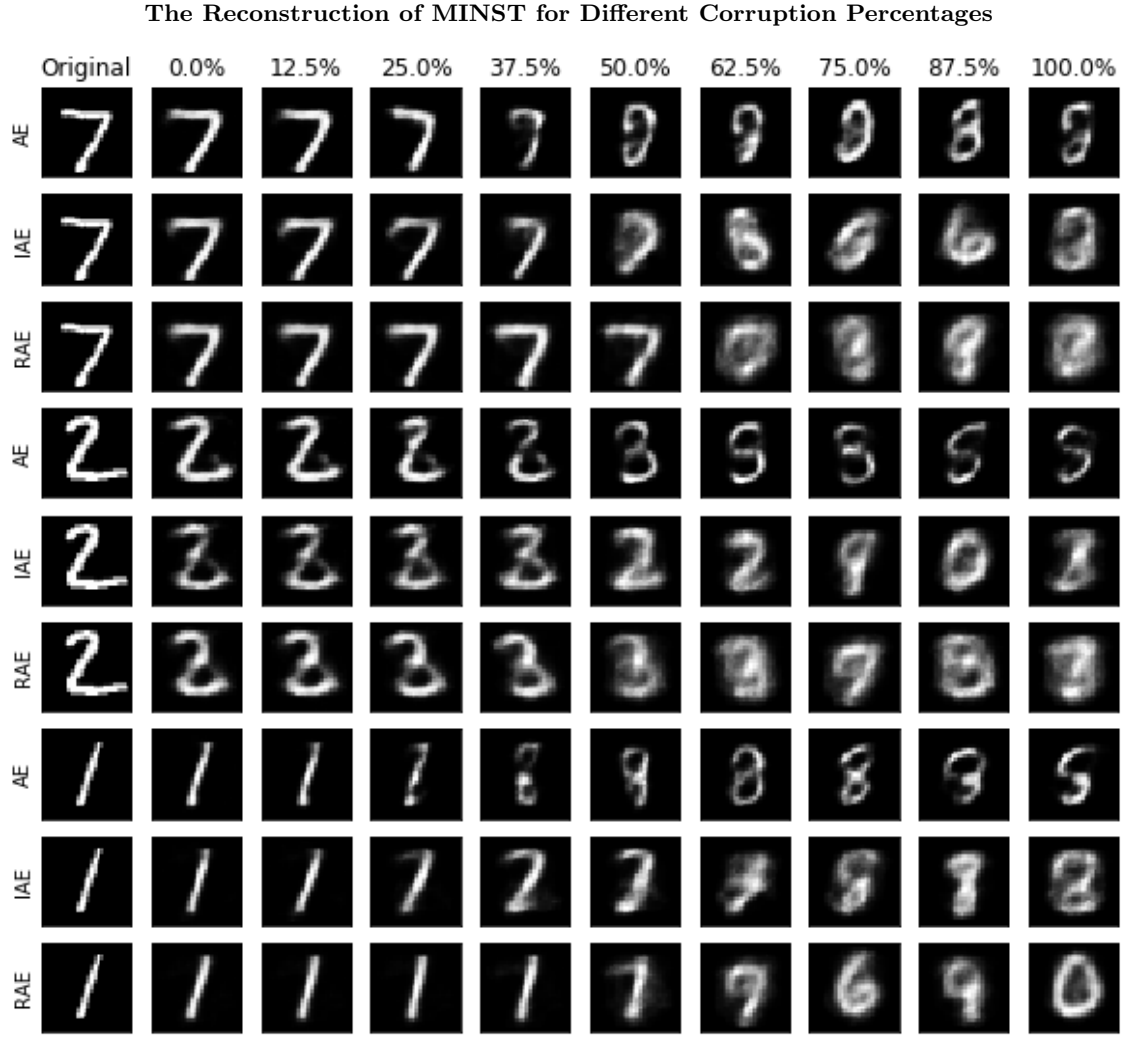


Figure 31: Visualization of the reconstructed images from the MNIST data set after random corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

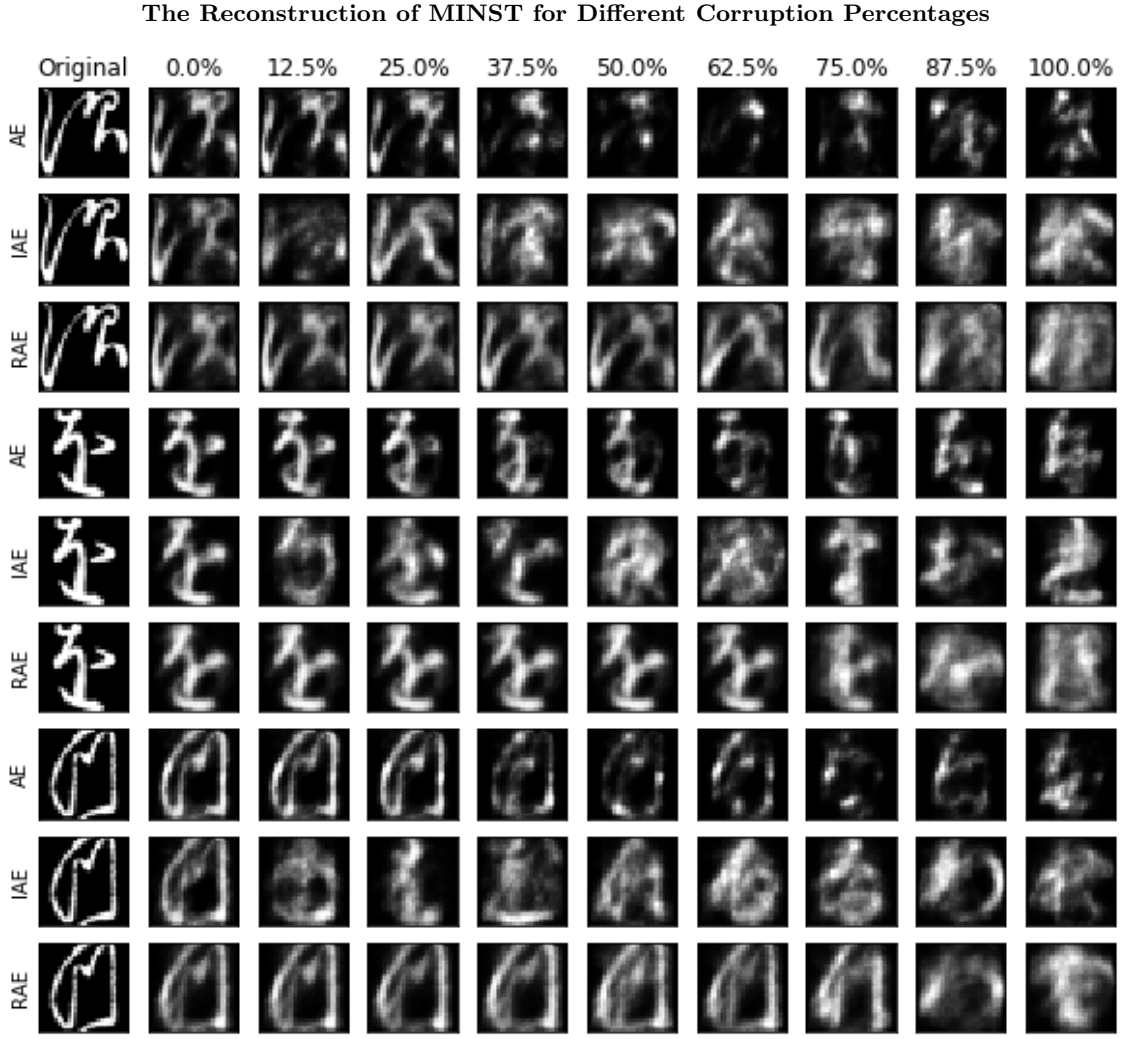


Figure 32: Visualization of the reconstructed images from the MNIST data set after reverse incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

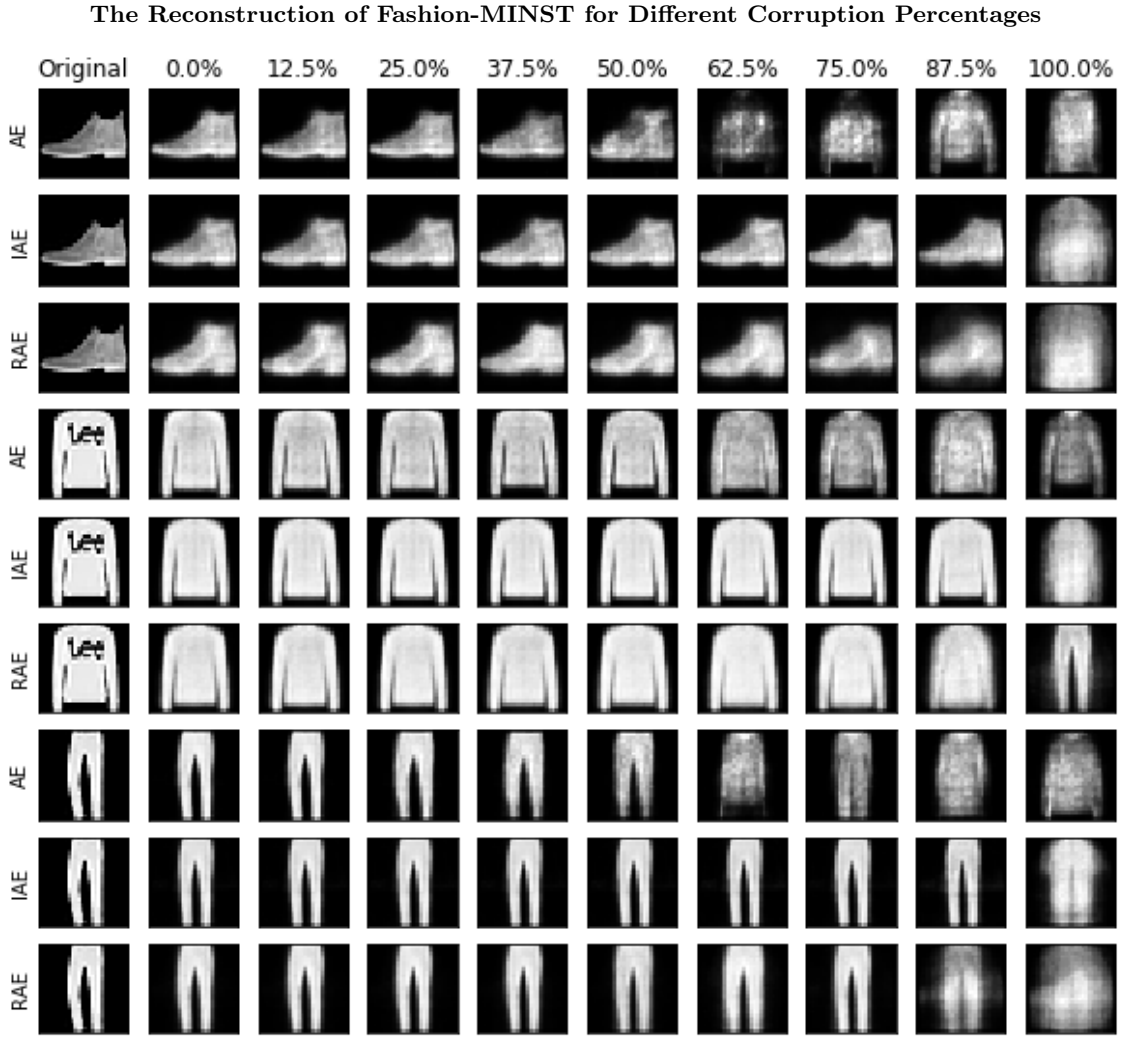


Figure 33: Visualization of the reconstructed images from the Fashion-MNIST data set after incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

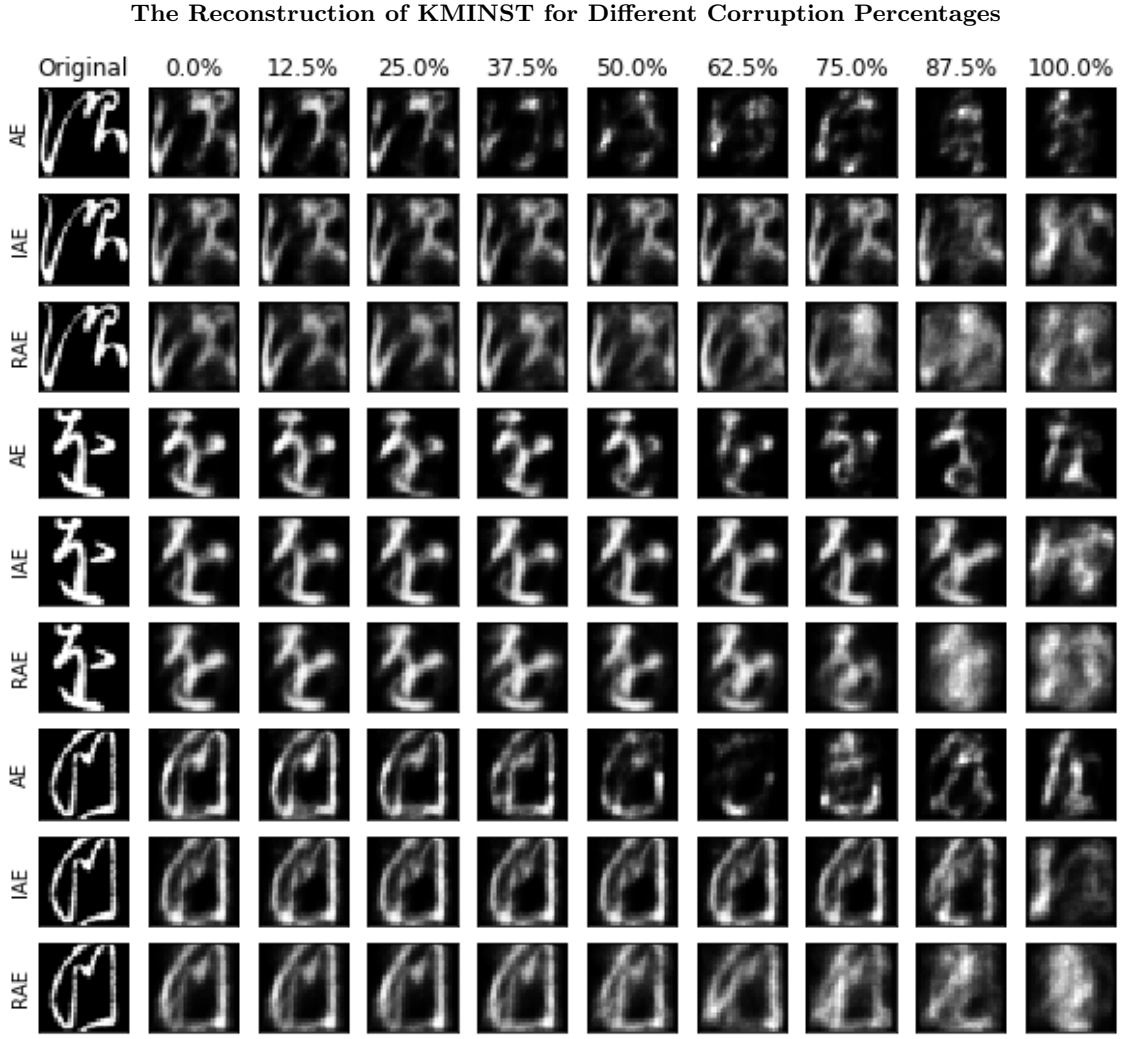


Figure 34: Visualization of the reconstructed images from the KMNIST data set after random corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

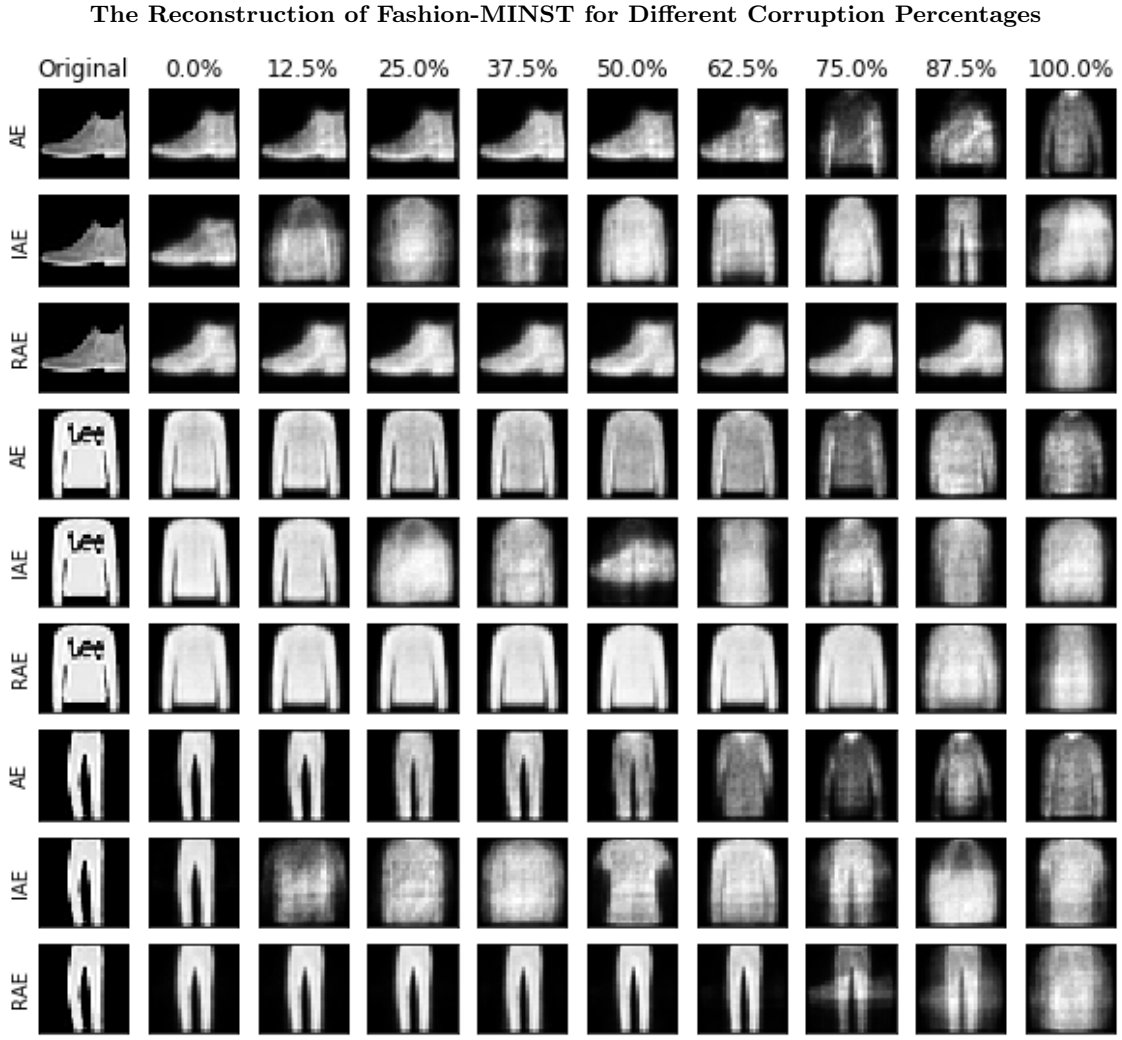


Figure 35: Visualization of the reconstructed images from the Fashion-MNIST data set after reverse incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

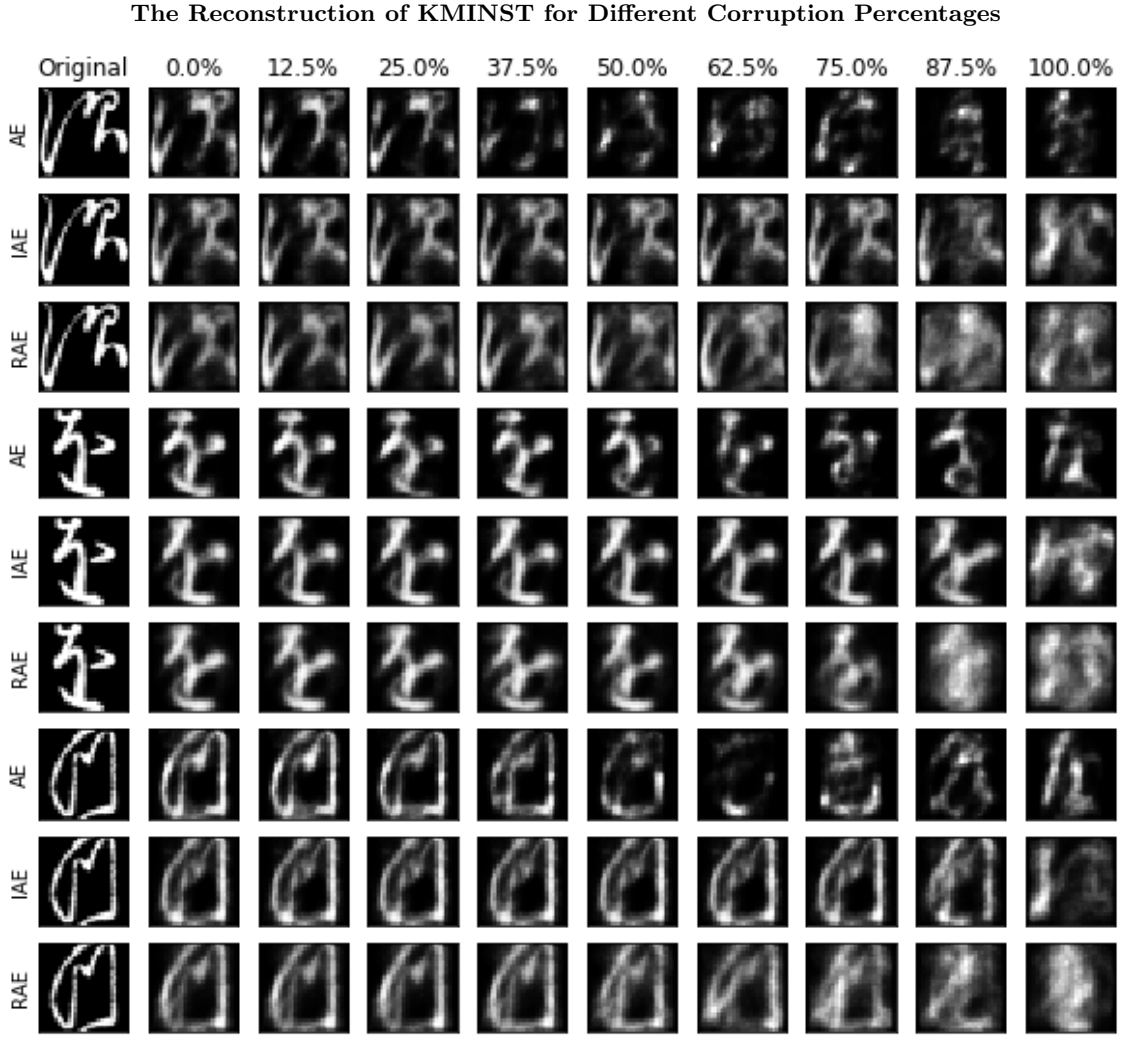


Figure 36: Visualization of the reconstructed images from the KMNIST data set after incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

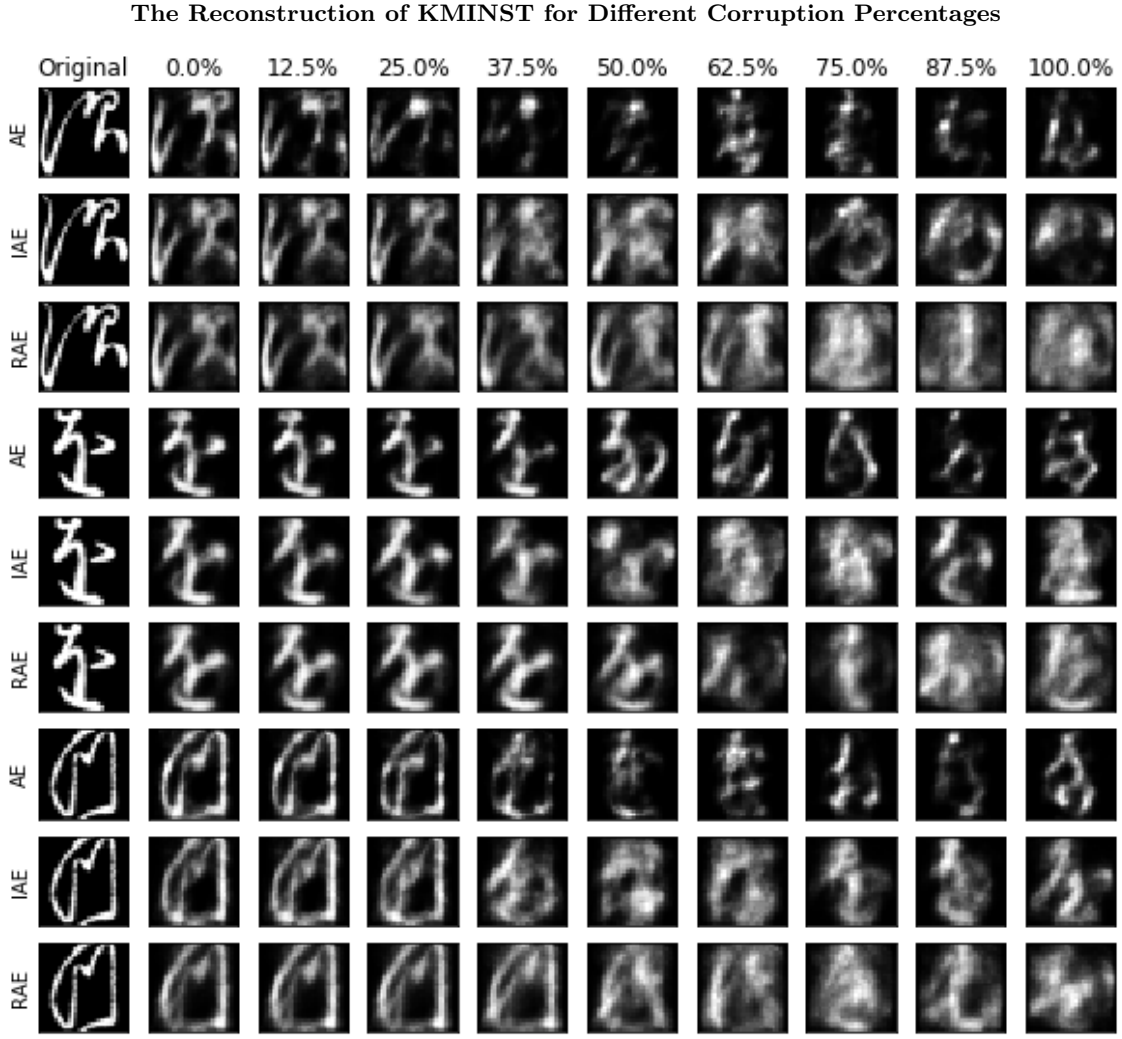


Figure 37: Visualization of the reconstructed images from the KMNIST data set after random corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.

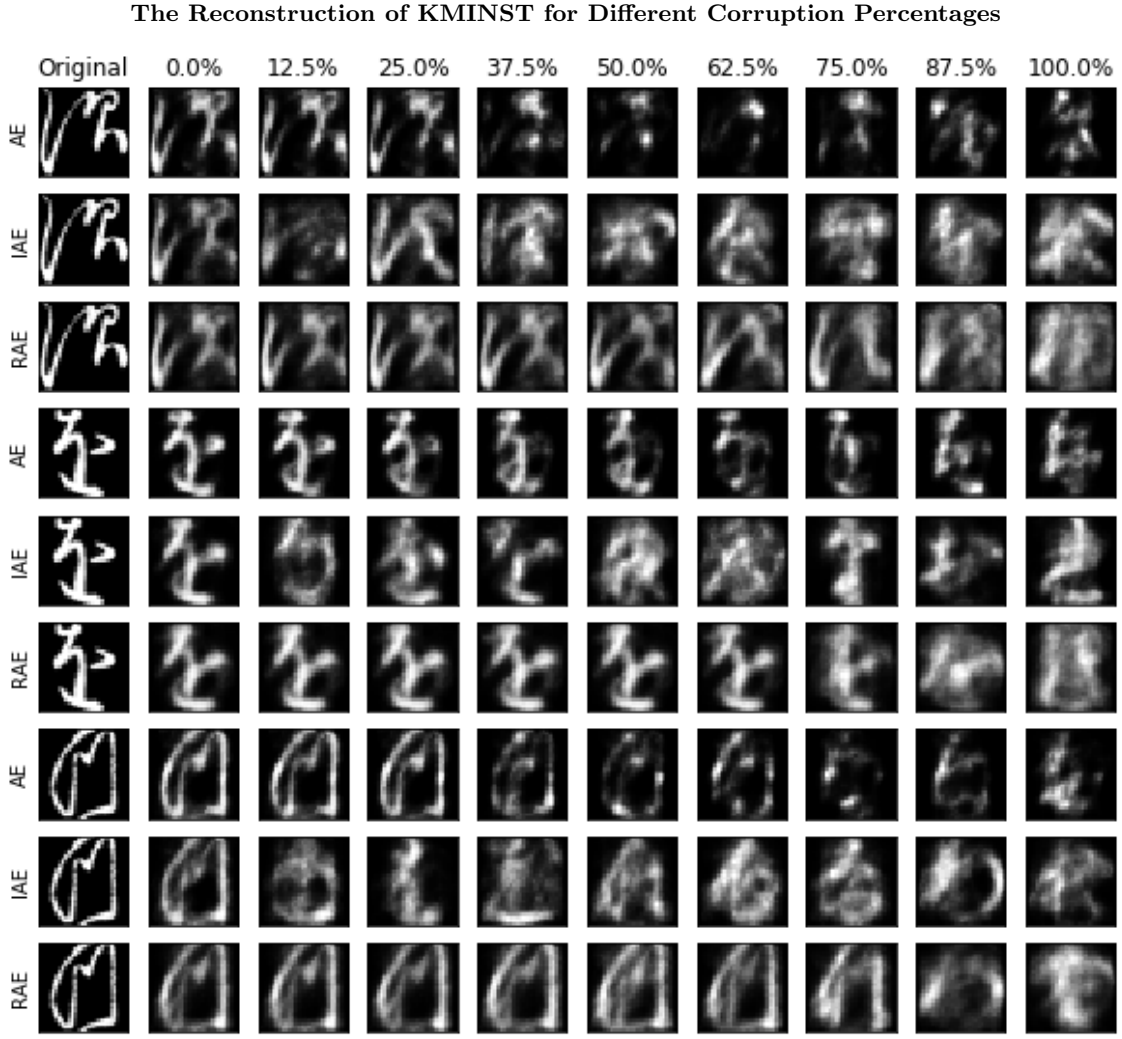


Figure 38: Visualization of the reconstructed images from the KMNIST data set after reverse incremental corruption. The leftmost image is the original image, with the other images being reconstructions. The corruption percentage for the reconstructed image is displayed above the column. The type of AE used for the reconstruction is displayed next to the rows.