IPiIs Final Project Report

Pim te Rietmole
Jinyao Tian
Jesse van Werven

1. Introduction

The process of removing noise from images is called denoising. The noise usually consists of undesired artefacts of the imaging process, resulting in a grainy or unclear image.

Denoising can be formulated as the following inverse problem: given a noisy image, is it possible to reconstruct the original image?

We investigate the effectiveness of a Denoising Variational Autoencoder (DVAE) for image denoising problems on the MNIST-Fashion dataset. We look into the effects of the latent dimension on the quality of the reconstruction, and compare this to other dimensionality reduction algorithms, namely PCA and Fourier filtering. We try to find the optimal level of regularization for all three methods for several levels of noise. Then for each level of noise, we denoise using optimal regularization, and compare the methods to see which is most effective. Structural Image Similarity Score (SSIM), Peak signal-to-noise ratio (PSNR) and Mean Squared Error (MSE) are used to quantify the quality of each of the denoising algorithms.

2. Theory

2.1. Problem formulation

In general, an inverse problem consists of a forward operator K, an input signal u, and noisy output signal f^δ

$$f^\delta = Ku + \epsilon \tag{1}$$

Here $|\epsilon| < \delta$ represents additive noise, which is bounded by delta. In our case we consider only problems where K = I, and ϵ represents the addition of Gaussian noise. We have that u is a 28×28 pixel image, as is f^δ. The objective is, given a measurement f^δ, to find a u such that equation 1 holds.

Theorem 1 in the appendix explains that an artificial neural network with arbitrary width (a varying number of neurons in a layer) can be used to approximate functions arbitrarily well. We can use this property of Neural Networks to motivate learning the inverse function for our problem: since an Neural Net can learn any type of function arbitrarily well, we can use it to approximate the inverse of our forward problem. However, the theorem only states the weights

w_i exist, not how to find them, and too many layers can cause overfitting of the learned function onto the data.

2.2. Denoising Variational Autoencoder (DVAE)

Auto-encoders are a type of multi-layered feed-forward neural network that specializes in dimensionality reduction. The number of neurons in the first layer and the last layer are equal to the number of dimensions of our data, n. The auto-encoder is trained to match its output to be the same as its input. This is done by setting the loss function to be equal to the L_2-norm difference between the network output and the ground truth formed by the clean image. By adding a "bottle-neck" layer in the middle of the network, of dimension $k < n$, we prevent the network from learning the identity function. A schematic picture of an auto-encoder can be seen in figure 1.
A successfully trained auto-encoder provides a lower-dimensional representation of the input data in the form of its "latent representation", i.e. the corresponding neuron values in the bottle-neck layer. The first half of the network forms an "encoder", and the second half a "decoder", that take us to and from this latent representation back to the original data representation. The process of first encoding and then decoding the original data does not always perfectly give back the original input. The degree of discrepancy between the input and output of the auto-encoder depends on the intrinsic dimensionality of the data and the number of neurons in the bottle-neck layer.

Auto-encoders tend to ignore or remove features that lie outside of their training distribution. This property can be used to our advantage to remove noise from images. By additionally adding noise to our training images while still requiring the auto-encoder to reconstruct their noiseless counterparts, we ensure that the encoder part of the auto-encoder is robust against noise. This way the latent representation of a noisy input will be close to the latent representation of its clean counterpart. This fact is important for obtaining an accurate reconstruction of the clean image. Auto-encoders that make use of this method of denoising are called "Denoising Auto-Encoders" (DAE).

Auto-encoders and Principal Component Analysis, which will be introduced later in this section, are closely related. It is possible to perform PCA with a linear autoencoder. [1] However, a large part of the power of deep neural networks is the non-linear activation functions, which allow the network to represent non-linear functions as well as linear functions.

A second variation on auto-encoder is a "Variational Auto-Encoder" (VAE)[2]. Variational auto-encoders are generative, in the sense that we can create new data by taking convex linear combinations of the latent representations of input data. This allows us to interpolate between data. An intuitive example is interpolating between the faces of two people to estimate what their children might look like.
In order to interpolate in this sense, we require some conditions on the latent space. The first is continuity; if two input images are "similar" (e.g. have a low difference under the L_2-norm), we require that their latent representations are also similar, and vice versa. The second requirement is convexity. The cloud of points formed by the latent representations of the training data should not have a "hole" in the middle. If a linear combination between latent representations ends up in such a hole, we do not expect the decoder part of the network to be able to reconstruct a sensible output.
In order to have our network fulfill these two requirements, we make two major changes. First of all, instead of mapping our inputs directly to points in the latent space, we map them into a distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$. We then sample this distribution to get the input for

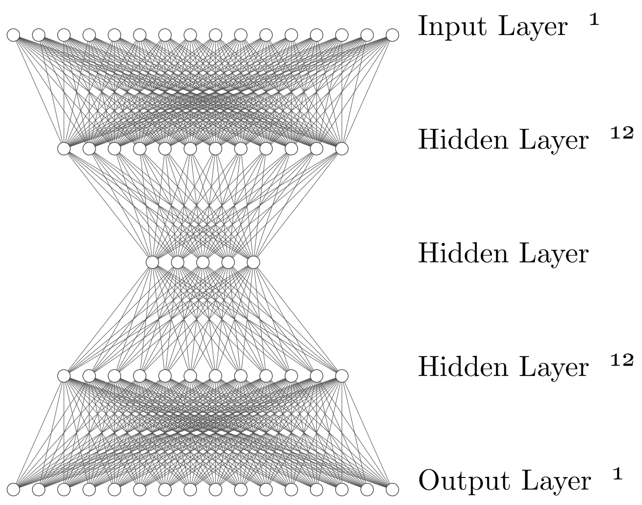

Figure 1: Schematic view of an auto-encoder neural network. The bottleneck typical to standard auto-encoders can be seen. The network dimensions are not to scale of our implementation, but do represent the approximate relative layer dimensions.

the decoder half of the network. Secondly, we add a KL divergence term to our loss function. This term applies a penalty when the distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ differs too much from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. This prevents the network from being able to learn latent distributions that are separated from each other by so much that we may as well have continued to simply map to points in the latent space.

In the context of our research, we use a “Denoising Variational Auto-Encoder” (DVAE). This is an auto-encoder that combines the changes described in the above paragraphs into a single neural network.
Choosing the latent dimension for the autoencoder is an important choice: choosing it too low results in the AE not being able to capture all of the information in an image, whereas a higher latent dimension can in turn risk overfitting, as the network can become more complex than the actual model that represents the data. An analogy can be drawn from interpolation, where a polynomial of degree n can always be fitted through n-1 datapoints. However, the actual data model could be very simple and a linear fit that does not directly reach all datapoints might be a more appropriate model than fitting a polynomial through all datapoints.
In fact, choosing the latent dimension close to the size of the input dimension results in the autoencoder learning the identity mapping, as expected.

2.3. Metrics

PSNR (Peak Signal-to-Noise Ratio)

PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is most easily defined via the mean squared error (MSE). Given a noise-free m×n monochrome image I and its noisy approximation K, MSE is defined as:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{2}$$

The PSNR (in dB) is defined as:

$$PSNR = 10\cdot\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\cdot\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) = 20\cdot\log_{10}(MAX_I) - 10\cdot\log_{10}(MSE) \tag{3}$$

Here, MAX_I is the maximum pixel value of the image.
SSIM
The difference with respect to other techniques mentioned previously such as MSE or PSNR is that these approaches estimate absolute errors; on the other hand, SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. Structural information is the idea that the pixels have strong inter-dependencies, especially when they are spatially close. These dependencies carry essential information about the structure of the objects in the visual scene. Luminance masking is a phenomenon whereby image distortions (in this context) tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is a significant activity or "texture" in the image.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{4}$$

As can be seen in figure 2, the SSIM can represent different information about the image than the MSE. While the two images have the same MSE, the shifted image is clearly more similar to the original than the noisy image. This is reflected in the SSIM metric.
An advantage of using the SSIM to estimate our algorithms performance is that the loss function has no direct relationship to this metric, so we

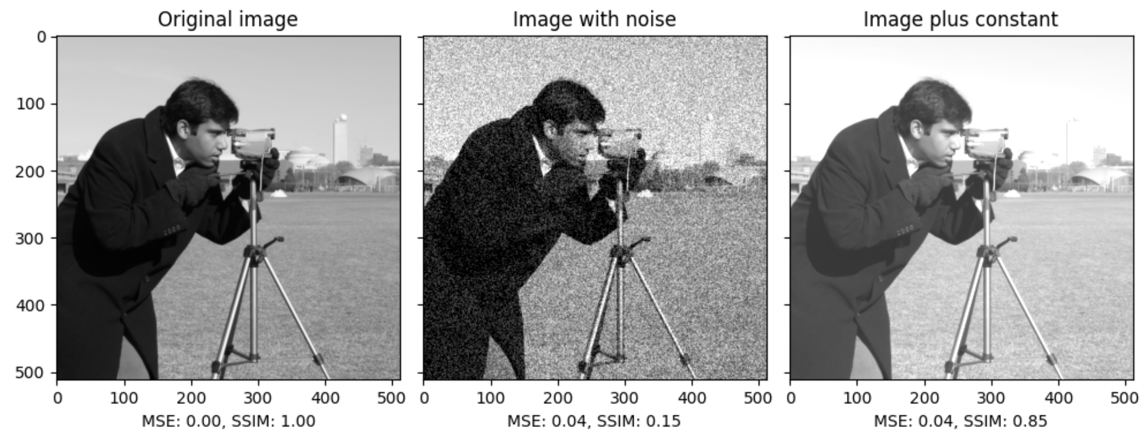

Figure 2: Difference between metrics for cases with linear pixel shift and noise addition. The SSIM in this case provides a better metric since it sees the shifted image as very similar to the original, and the noisy image as not very similar. Image adapted from `https://scikit-image.org/docs/dev/auto_examples/transform/plot_ssim.html`.

2.4. Fourier Analysis
A 2D image permits a Fourier deconstruction by performing a frequency domain analysis in two directions. In this way it is possible to characterize the functions in the 2D x-y image plane with a frequency domain in u and v, with $\sin(ux + vy)$. An intuitive motivation for this is given in [3]

Fourier techniques are often used when we have some prior knowledge about the noise, such as the approximate frequency the noise resides at. This is even more so when the noise is strongly periodic. This is because the Fourier transform enforces periodicity by effectively extending the image with itself, placing an infinite, periodic sequence of images along which it performs frequency analysis. As such, any function can be approximated using a linear combination of sinusoids. However, when the function is strongly periodic this frequency is more meaningful and perhaps a Notch[4] filter can be applied to remove periodic noise. When the noise is not periodic, it can still be represented by a periodic function, however, the frequency of this function will be very high in order to capture the irregularities. This is why Gaussian noise is often removed using a low-pass (smoothing) filter. However, a negative side effect of this is that the high frequency part of the Fourier spectrum contains most of the information about the edges of the image[5], since higher frequency functions are necessary to represent the more complex behaviour of the edges. Therefore, implementing such a low pass filter results in edges getting blurred.

2.5. Basic PCA: Principal component analysis

Principal component analysis is an orthogonal transformation that seeks the direction of maximum variance in the data and is commonly used in dimensionality reduction of the data. Data with maximum variance contains most of the information needed to present the whole dataset. In image denoising, one has to take care of the compromise between noisy data and preserving the high variance image data detail. We can start by looking into the PCA analysis to see how PCA inherently tries to reduce the noise in an image.

The basic intuition behind denoising the image is that any components with variance much larger than the effect of the noise should be relatively unaffected by the noise. So if you reconstruct the data using just the most significant subset of principal components, you should be preferentially keeping the signal and throwing out the noise.

We tried the plain PCA method for one image (a boot) from the MNIST-Fashion dataset. Our approach is:

- Take the dataset of a single image (1*28*28)
- Add three levels of random Gaussian noise to the image (std= 0.01, std=0.1, std=1)
- Plot the variance vs Component curve to determine the component storing the highest variation. And find the optimal number of components we need for the best PCA denoising performance.
- Apply inverse PCA to get the image back using the components derived in the above step.
- Visualize the dataset again to see the difference.

3. Methods

3.1. Dataset

We perform our experiment on the MNIST-Fashion dataset, which consists of images of dimensionsality 28x28 images. Our data is split 80-20, with 80% training, 20% test. We do not need a validation set since we do not run the risk of overfitting hyperparameters onto the test data.

3.2. Noise

In this forward process, the noise added is additive Gaussian noise. We vary the mean and the variance of the Gaussian noise. The entire image is subjected to this additive noise process.

Between training epochs for the DVAE, the noisy images are varied, since the autoencoder should be able to denoise any type of noise, so we expected it would generalize better if we do not use the same noisy version of each image every epoch, as this would effectively mean the VAE has seen more distinct realizations of the noise than it would otherwise.

3.3. DVAE training

In order to investigate the effect of the latent dimension on the reconstruction accuracy, we train multiple autoencoders on the same dataset (and with the data samples given in exact the same order with each trained autoencoder). This way we keep all variables the same except for the latent dimension. The latent dimension is varied from 1 to 20.

The input dimension for the DVAE is 28x28, and the output dimension is also 28x28. In between the bottleneck layer are two smaller linear layers. These extra layers stack the auto-encoder, allowing it to represent a more complex model. The network is thus effectively a fully connected multi-layer perceptron with a hidden layer of which the size is varied in order to estimate the effect the dimension of this bottleneck layer has on the reconstruction quality.

The training algorithm consists of adding noise to the images in the data-set using the process described in 3.2. The network is given as input the corrupted image. The loss function for the network is given as the L_2 norm of the difference between the true image and the reconstructed image. In this way, the network learns to reconstruct denoised version of the noisy images. We train the network weights using backpropagation and the ADA optimizer of PyTorch[6].

3.4. Fourier Denoising

As explained in section 2.4, we can deconstruct a 2D image into a 2D Fourier spectrum. We investigate what percentage of the fourier coefficients is optimal for denoising, by reconstructing the image starting with 0% of the coefficients, then 1% of the coefficients, until we have all coefficients, following the implementation of [7]. We select the coefficients starting around the origin and symmetrically, as shown in figure 3

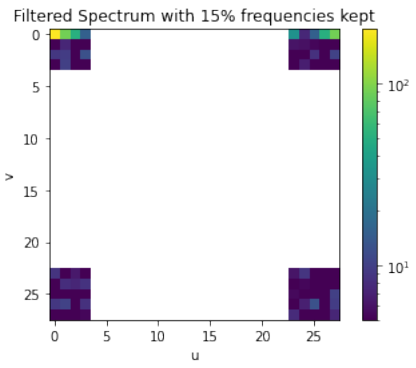

Figure 3: Frequencies kept for the Fourier transform. The frequencies are at the edges because they have not been fftshifted, they represent the frequencies around the origin of the Fourier spectrum

We then use the modified Fourier spectrum to construct the filtered image by applying the 2D inverse Fourier transform.

3.5. PCA Denoising

Following the step in section 2.5, we need to show the original digit image before adding noise comparing with the noisy images by adding three different levels of random Gaussian noise. The digit data-set looks like this:

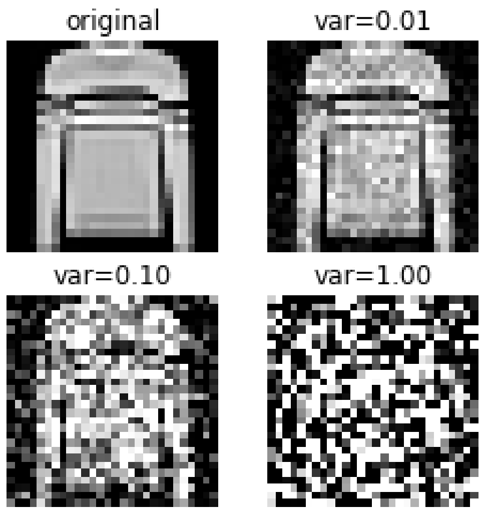

Figure 4: Comparison between original image and three noisy images

Then we explore how the variance is distributed across principle components for individual pixels for different noise levels. The below are the results for original image and noisy images with three different std levels shown in Figure 4:

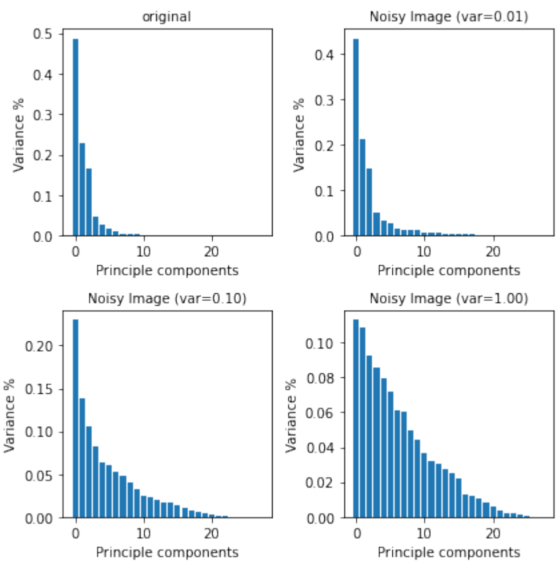

Figure 5: variance of signal's and variance of noise's distribution on principle components

From the above plots it can be observed that variance of a signal will concentrate on a small subset of principle components, while the variance of noise will evenly spread over the whole dataset. So, based on the estimated value of sigma least significant components can be dropped

and image can still be reconstructed with good quality. Most of the dropped components will contain noise, so the reconstructed image will be a denoised image.
But one of the main drawback of PCA method is that, as the std value increase energy gets too much distributed among all the components, which makes it difficult to estimate the number of principal components to be retained for image reconstruction.

4. Results

4.1. Optimal regularization

4.1.1. Optimal number of latent dimensions in the DVAE

In order to denoise as successfully as possible using the DVAE, we need to pick the correct number of latent dimensions. If we use too few latent dimensions, we might not be able to reconstruct the original image due to the limitations on the information that can pass through the neural network. If, on the other hand, we use too many latent dimensions, we might end up over-fitting our training data. To prevent these problems, we will study the optimal number of latent dimensions.

We will consider this problem from both the perspective of using MSE and SSIM.

Take MSE (mean-squared-error) as an example. The procedure is as follows: we pick a number of latent dimensions, and train the DVAE. We calculate the mean-squared-error between a clean image from our test data-set, and the corresponding denoised image. We repeat for a large number of images. Then we take the average. This gives an estimate how well the DVAE is denoising for that number of latent dimensions. We train a new DVAE for a new number of latent dimensions, and repeat the process.

The expectation is that the resulting graph of average mean-squared-error versus latent dimension will have a minimum somewhere. This minimum is then an estimate for the optimal number of latent dimensions.

An analogous procedure is done using SSIM instead of MSE. The resulting graph, for noise with variance of 0.1, can be seen in figure 6. Figures for the remaining noise levels can be found in the appendix in figure 15

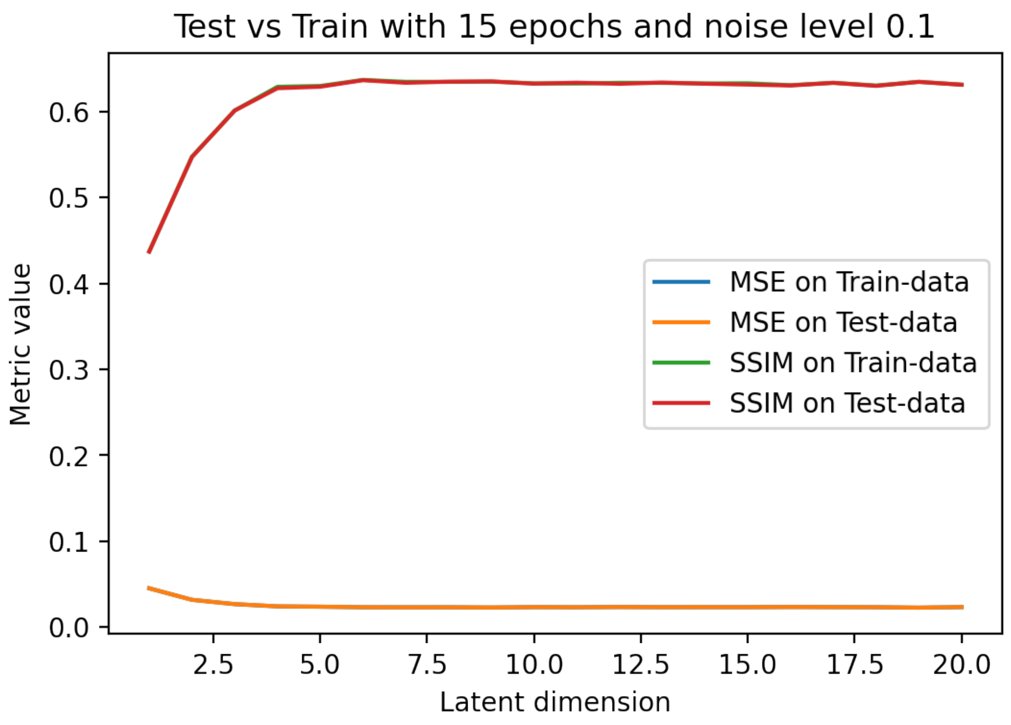

Figure 6: This graph shows the MSE and SSIM measures between the clean and denoised images. The denoised images follow from the DVAE trained on noise with variance 0.1 and mean 0. Along the x-axis we change the latent dimension of the DVAE. Performance rises as the minimum dimensionality to represent the data is reached, and flattens after this threshold. Performance on test and train data is nearly identical, indicating that overfitting is not a problem in this context.

As expected, a very low number of latent dimensions leads to a less accurate denoised image. However, in the range of latent dimensions shown in the figure, we see that the mean-squared-error never seems to significantly decrease as a result of increasing the number of latent dimensions. Also, MSE and SSIM values change quickly at low latent dimension, but plateau for latent dimensions above 6.
Together with the fact that the MSE and SSIM measures for training data and test-data overlap, we conclude that, at least in this range of latent dimensions, overfitting does not seem to be a problem. This likely also has to do with the size of our training data-set. If we have more training examples, overfitting is less likely to occur.
The fact that we reach this plateau around a number of latent dimensions equal to 6, means we can pick this number of latent dimensions as an approximately optimal choice for denoising.
Of course, this result might depend also on the level of noise. Figures 15in the appendix (section 7) show the same data as figure 6 for different noise levels. The optimal number of latent dimensions derived from these are noted in table 1 towards the end of this section.

We can also think of the mean-squared-error as being a combination of two separate terms. The first term is error due to bias. We expect this term to be large when we over-regularize, so especially for a low number of latent dimensions.
The second term is error due to variance. We expect this term to be large when we under-regularize. However, based on the results of figure 6, in the context of a DVAE we expect this

term to increase very little, if at all, for increasing number of latent dimensions.

To estimate these two terms, we first create a hundred noisy images from a single clean image. Then we pass each of these through the DVAE. The average output image is our MAP estimate of the denoised image. We define our estimate of the variance term in the error to be the average MSE between the MAP estimate and the individual denoised images. The bias term is then estimated as the MSE between the MAP estimate and the clean image.
Figure 7 shows estimates of these bias and variance terms respectively, for differing number of latent dimensions, and for noise with mean 0 and variance 0.1. The x-axis shows the number of latent dimensions, ranging from 1 to 15. The y-axis shows the MSE values as described in the last paragraph. The red points show the estimates of the bias term, and the blue points the estimates of the variance term. The bias term decreases sharply up until a number of latent dimensions equal to 3. After that, it decreases more slowly, and eventually plateaus. The variance term is much smaller than the bias term on average, and seems to change relatively little. It is slightly larger for a large number of latent dimensions.
These observations seem to be consistent with our results in figure 6. From here as well, we can conclude that using 6 latent dimensions seems like a good choice for denoising, as the bias and variance at that number of latent dimensions are low. We see that the most important factor affecting our choice of latent dimensions should be the bias term, as it is relatively large and changes more when varying the number of latent dimensions compared to the variance term. The variance term is small, and changes relatively little when varying the number of latent dimensions. This is in line with our conclusion that we do not seem to be overfitting in this range of latent dimensions.

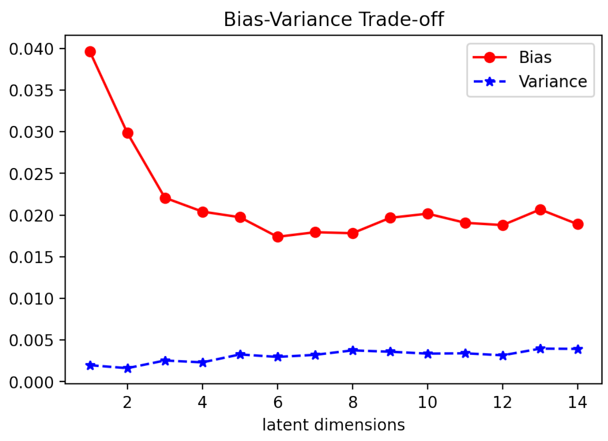

Figure 7: This figure shows estimates of these bias and variance terms respectively, for differing number of latent dimensions, and for noise with mean 0 and variance 0.1. The x-axis shows the number of latent dimensions, ranging from 1 to 15. The y-axis shows the MSE values as described in the last paragraph. The red points show the estimates of the bias term, and the blue points the estimates of the variance term. The bias term decreases sharply up until a number of latent dimensions equal to 3. After that, it decreases more slowly, and eventually plateaus. The variance term is much smaller than the bias term on average, and seems to change relatively little. It is slightly larger for a large number of latent dimensions.

4.1.2. Optimal number of principal components in PCA

When denoising by truncating the principal components of an image, the main question we need to ask ourselves is: what number of principal components should we choose to have the optimal denoising result? As has been analyzed above, using more principal components results in more noise contained (shown in Figure 5). However, the less components we use, the less signal information we have. We are dealing with a bias-variance trade-off problem.
Figure 8 shows a plot of the cumulative explained variance for different numbers of components at which to truncate.

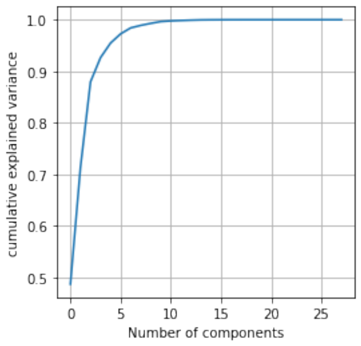

Figure 8: cumulative explained variance

It shows that approximately the first 10 principal components already contribute to almost all the variance of the signal image.

Now we want to know the optimal number of principal components to keep for the purpose of denoising. We can deduce this by looking at the PSNR and SSIM metrics between the denoised image and the clean image while varying the number of principal components.
This is shown in figure 9. The x-axis shows the number of principal components kept, while the y-axis shows the PSNR and SSIM metric values. We can clearly see that the graph is sharply peaked at 3 principal components. This indicates that truncation at 3 principal components is optimal for a denoising an image with noise variance equal to 0.1.

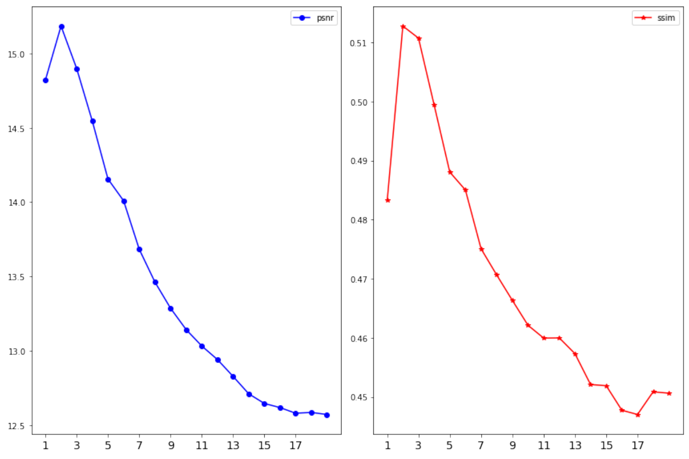

Figure 9: This figure shows the denoising quality of the PCA method for noise with variance of 0.1. The x-axis shows the number of principal components kept, while the y-axis shows the PSNR and SSIM metric values. We can clearly see that the graph is sharply peaked at 3 principal components. This indicates that truncation at 3 principal components is optimal for a denoising an image with noise variance equal to 0.1. The results are averaged over 100 images to ensure they represent the dataset.

The optimal number of principal components for other noise levels were similarly determined for other levels of noise. The graphs for these different noise levels are found in the appendix (section 7). The resulting values can be seen in 1 towards the end of this section.

4.1.3. Optimal frequency threshold in Fourier Method As described in the Methods section, we find the optimal fraction of Fourier coefficients to keep by finding the average reconstruction error per fraction of coefficients kept. This results in a clear pattern: higher noise requires us to perform more smoothing and the Fourier spectrum is more polluted, so we keep a much lower fraction of components. The filter thus acts very similar to a blurring filter. The SSIM is still considerably better when denoising with the optimal parameter than the noisy image, as can be seen in figure **??** in the Appendix. We can conclude that per noise level, an optimal regularization exists, but that they depend heavily on the noise level, which has to be known in advance.

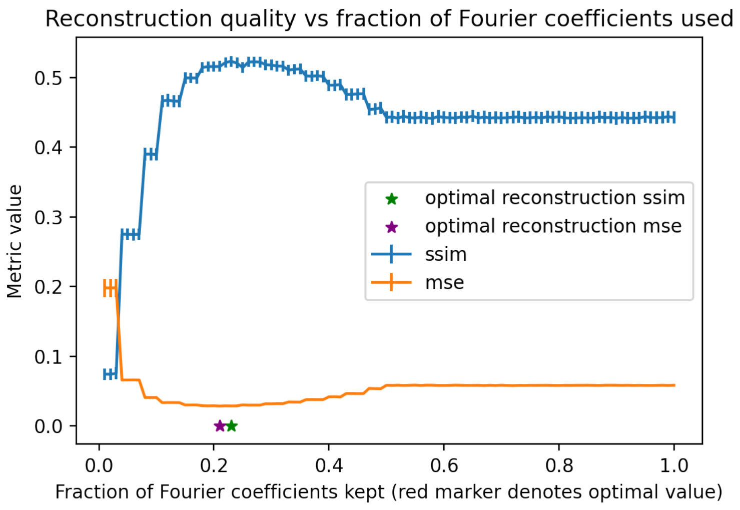

Figure 10: Analysis of 100 images with noise with mean 0 and variance 0.1, reconstructed with Fourier with different amounts of coefficients kept. Metrics SSIM and MSE are used. Error bars represent the spread, as different number of coefficients kept might be better for different images. It is clear that there is a distinct optimum at 21% and 23% at which minimal noise is present, but still enough Fourier coefficients are kept to not lose too much information about the image. Here the mean error is low and the images are structurally the most similar. The error bars are very small, indicating that the optimal choice does not vary greatly per image and all images with 23% of coefficients are optimal

4.2. Comparing VAE, PCA, and Fourier methods

In figure 12 we compare the methods for different noise levels. We see that the Fourier and PCA method work better than the DVAE for low variance noise, and that the DVAE works better for higher variance noise. The parameters used are taken from 1.
It makes sense that the DVAE outperforms the other methods on higher noise levels, since the DVAE is specifically trained to remove this noise level, whereas for the other methods, we follow the same denosing steps but take a different number of components and fourier coefficients. However, these still contain a large part of noise if the noise is high. The DVAE is able to deal with this noise because it has learned how to reconstruct an image that resembles an article of clothing out of any noisy image through training.

We also expect the DVAE might be performing well on the metrics because all images in the dataset have the same background, and the DVAE might be very good at reconstructing this background.
All methods seem to act a lot like low-pass filters/averaging filters, introducing blur in the right locations to counteract noise.

Additionally, all methods seem to prefer a lower dimensional representation of the data when the noise variance becomes higher. For the DVAE, this change is less obvious, since there is a small peak in performance at the low latent dimension (as can be seen in 15), but the performance more or less levels out for higher dimensions. However, more dimensions means more parameters and thus more training time, so the lower latent dimensions are optimal for higher noise levels.

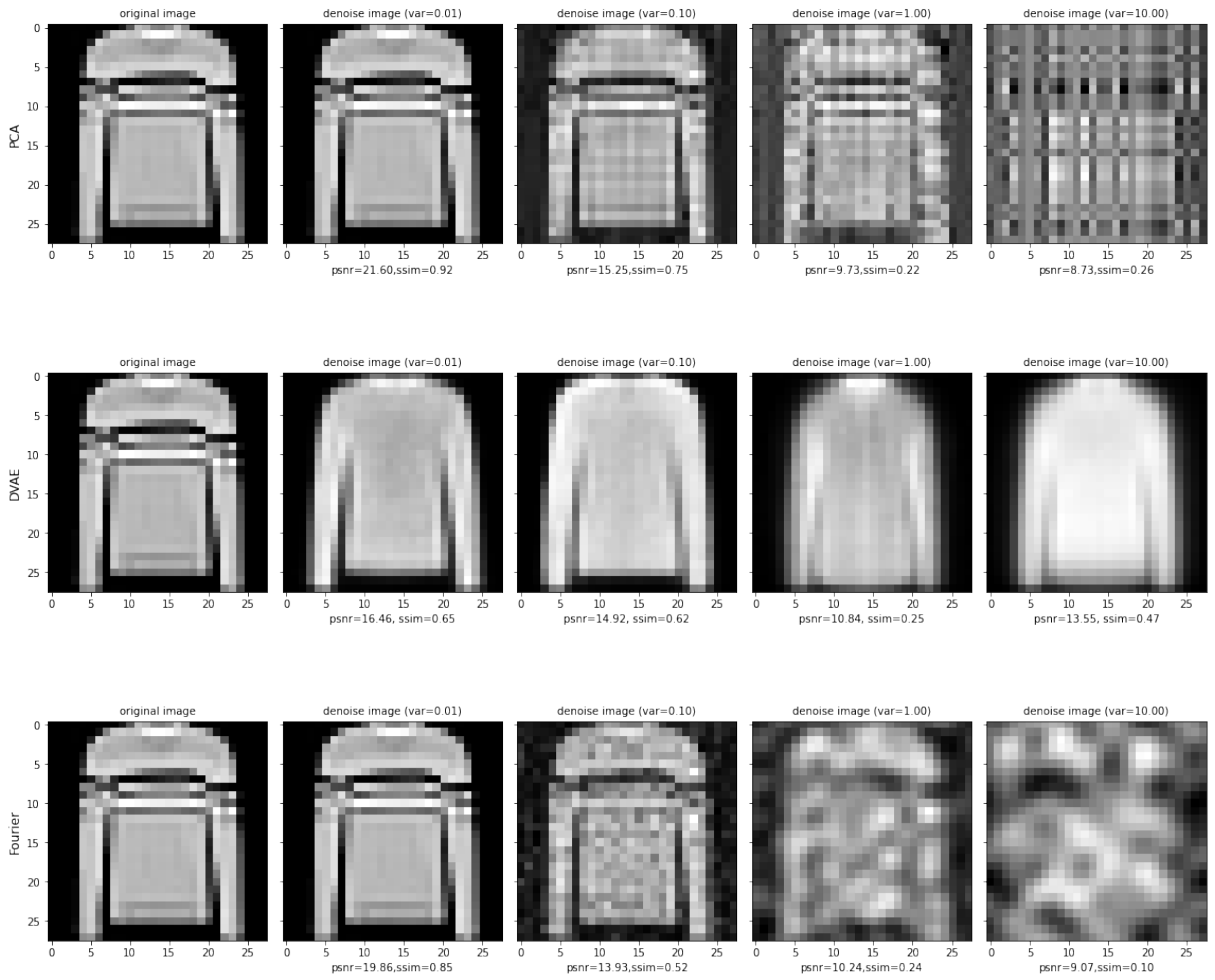

Figure 11: Comparison of denoised results between three methods

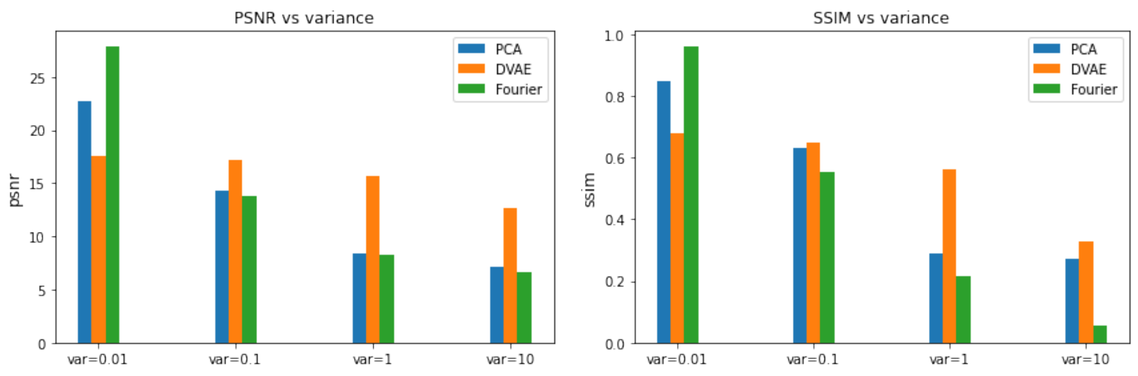

Figure 12: Quantitative analysis of denoised results. The figure shows reconstruction accuracy with different metrics, on different noise levels. The results were found by first determining the optimal parameters for each algorithm. The DVAE latent dimensions are taken from table 1. PCA performs best for low noise variance, whereas DVAE obtains significantly better results than the other methods for high variance noise. The results are averaged over 100 images to ensure they represent the dataset.

Noise variance	**DVAE Latent Dimension**	**Fourier fraction kept**	**PCA components kept**
0.01	6	0.44	5
0.1	6	0.23	2
1	4	0.16	1
10	2	0.10	1

Table 1: Algorithms and optimal parameters for different noise levels (0.01, 0.1, 1, 10). The latent space dimensions for the DVAE were found by training the DVAE on data with noise corresponding to each noise level, sequentially. Then the optimal value was found by evaluating on test data with the same noise distribution as the train data

5. Conclusion

The DVAE trained on high variance noise performs quite well on that noise level compared to the optimal filtered image using PCA and Fourier filtering. The latent dimension of the DVAE has a threshold above which the performance flattens, as can be seen in figure 6. This threshold dimension gets lower as more noise is introduced into the data. This is in line with the results from PCA and Fourier analysis as well: more noise means we need a lower dimensional representation/less components of the image to reconstruct it effectively. The optimal values for each method on each noise level can be found in table 1.
Thus, the DVAE provides a promising way to denoise high variance noise more efficiently than traditional Fourier and PCA based methods, and we see similar patterns in the dimensionality of the DVAE as with the Fourier and PCA methods, though the change in performance is much less drastic for the DVAE.

6. Discussion

6.1. Failed attempts

There are some things we attempted, that didn't make it into this report. For some this was due to time and page-limit concerns. For others, it was because we suspect the results might be inaccurate due to some mistake on our part.

For the largest part, these consist of our attempts to quantify a image-specific uncertainty for the denoising process formed by the DVAE. Initially we planned to estimate these uncertainties by separately estimating the bias and variance term of the error. In the end it turned out our approach was founded on a misinterpretation of how to calculate the variance estimate, and we had to abandon the idea.
Another idea was to study the variance of the latent distributions that the encoder part of the DVAE maps into. Specifically, we wanted to know whether the norm of this variance might be correlated to the mean-squared-error between the clean and the denoised image. If such a correlation exists, it would be a strong argument that an image-specific uncertainty estimate could be formulated based on the variance of the latent distribution. Unfortunately, due to time constraints we were not able to implement this.

Another thing that didn't make it into this report is an alternative method for estimating the optimal number of latent dimensions for the DVAE. This method was based on the histograms of the pixel-wise intensity difference between the denoised image and the noisy image. By comparing this histogram to a Gaussian distribution with the mean and variance of the noise, we can tell if we are over- or under-regularizing. We ultimately opted to omit these figures for the sake of brevity.

6.2. Limitations

Like most neural networks used in image processing, a DVAE is limited by its input dimension: the network will have no idea how to denoise images of 50×50, unless we downscale those images to the resolution of the network (28×28), and in doing so we lose a lot of information. In fact, the only networks that we are aware of that do not have this limitation are Fully Convolutional Networks (FCNs).

Additionally, the autoencoder works well for lower dimension, but the number of connections grows exponenentially with the input dimension of the image. Therefore, we expect that for images of 512x512, the algorithm training time becomes very long and the number of variables in the network becomes incredibly large.
A potential method to solve this issue could be fully convolutional networks (FCN). These have already been used in 3D reconstruction of scenes from 2D images [8] and we expect that these can also be used in denoising applications.

Additionally, there is no direct dependence implied in the model that an autoencoder uses: the neural network inputs are processed independently, and the only relationship between the input nodes is how the edge weights attached to them interact with the activation function. This could also be solved by introducing convolutional layers into the autoencoder since these exploit the spatial structure of the image directly.

The DVAE is also trained on one type of noise, and we are not sure whether it generalizes well to different types of noise, such as non-Gaussian noise, or other noise levels.

6.3. Further Works

It could be that for higher noise levels, the DVAE reconstructs an image that is structurally similar to the original image because it has any form of structure, and not because it has the right structure. This is because for such high noise levels, we expect that anything that resembles a clothing article will be structurally somewhat similar to another clothing article. It could be worth investigating if the class of reconstructed clothing article changes a lot in this reconstruction.

Currently the forward model uses an identity operator as the focus of this paper is on denoising applications. However, we expect that this can easily be extended into an ability to learn how to invert a forward operator. This is motivated by our previous explanation of the universal approximation theorem for deep neural networks. An application of this could be, for example, learning face inpainting, where the forward operator sets a section of the face's pixels to zero (or some other arbitray value), but it could extend itself to fields beyond pure imaging techniques, such as system learning in control systems. It would then be interesting to see how this would compare with regularized pseudo-inverse methods and other traditional inversion techniques.

More complicated network structures could be explored, such as stacking more layers of sparse autoencoders, and using pretraining to train them quicker, by training each layer as a Restricted Boltzmann Machine, as suggested in this work[9], would allow us to perform this experiment more easily on larger networks.

7. Appendix

Theorem 1 (Universal Approximation Theorem[10]) *Let $\sigma : R \rightarrow R$ (activation function) be a bounded, real, non-constant, continuous function. Let I_m denote the m-dimensional hypercube $[0,1]^m$. The space of real-valued continuous functions defined on I_m is denoted by $C(I_m)$. Then given any $\epsilon > 0$ and any function $f \in C(I_m)$, there exists an integer N, real constants $v_i, b_i \in R$ and real vectors $w_i \in R^m$ for $i = 1...N$ such that we may define*

$$F(x) = \sum_{1}^{N} v_i \phi(w_i^T x + b_i) \tag{5}$$

as an approximation of the function f, that is,

$$|F(x) - f(x)| < \epsilon \tag{6}$$

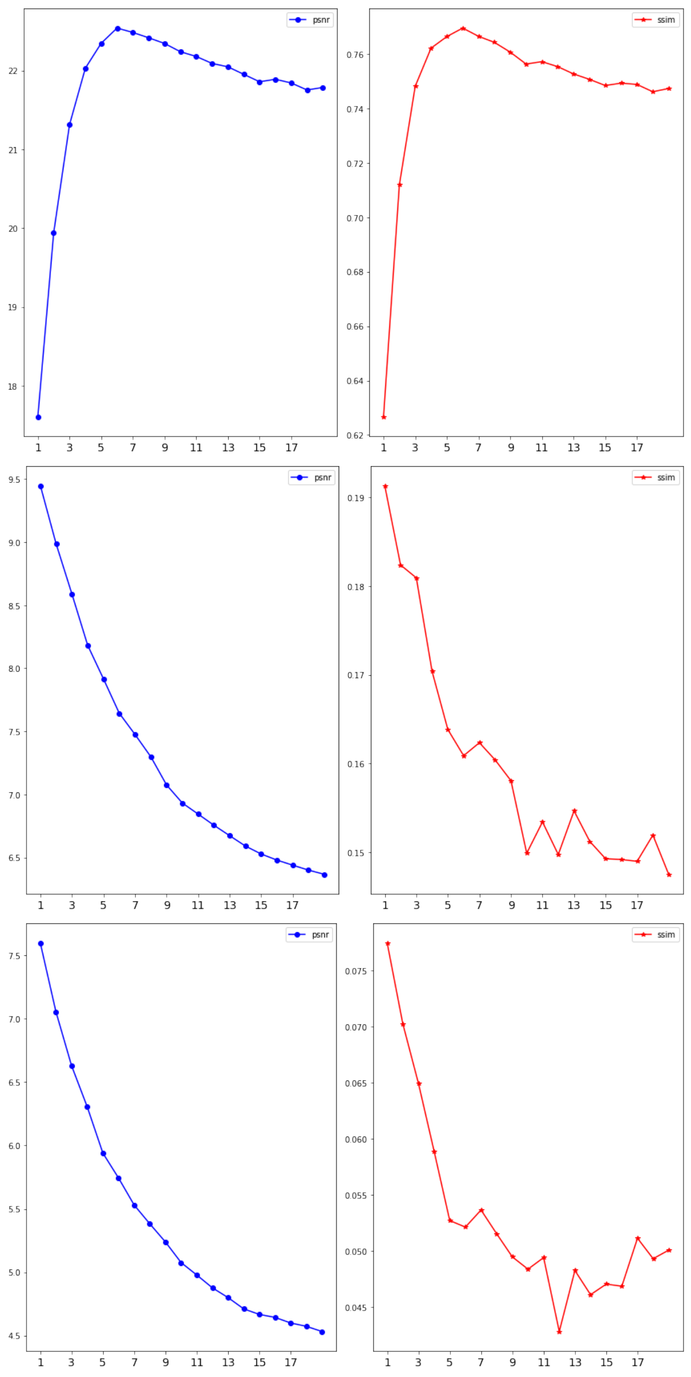

Figure 13: This figure shows the denoising quality of the PCA method for noise with variance of 0.01, 1 and 10. The x-axis shows the number of principal components kept, while the y-axis shows the PSNR and SSIM metric values.

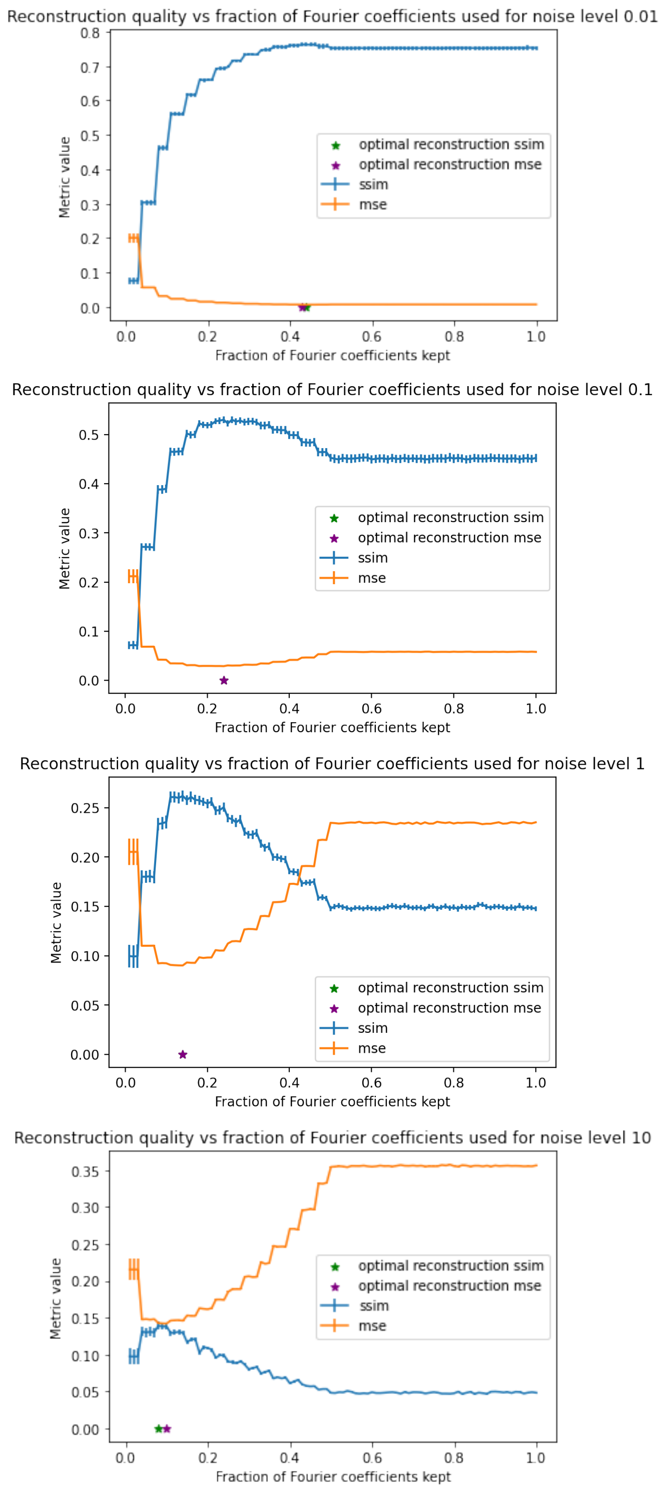

Figure 14: Fourier denoising performance at different noise levels. It is clear that for low noise, as much of the information in the image should be used as possible, where for higher noise levels, the reconstruction suffers a lot from more frequencies taken. The Fourier denoiser is thus very sensitive to the noise level and does not perform as well for higher variance noise, though relative to the unfiltered image it does improve the reconstruction a lot even for high variance noise, going from an SSIM of 0.05 to 0.15 in the last plot, an improvement of around 300%

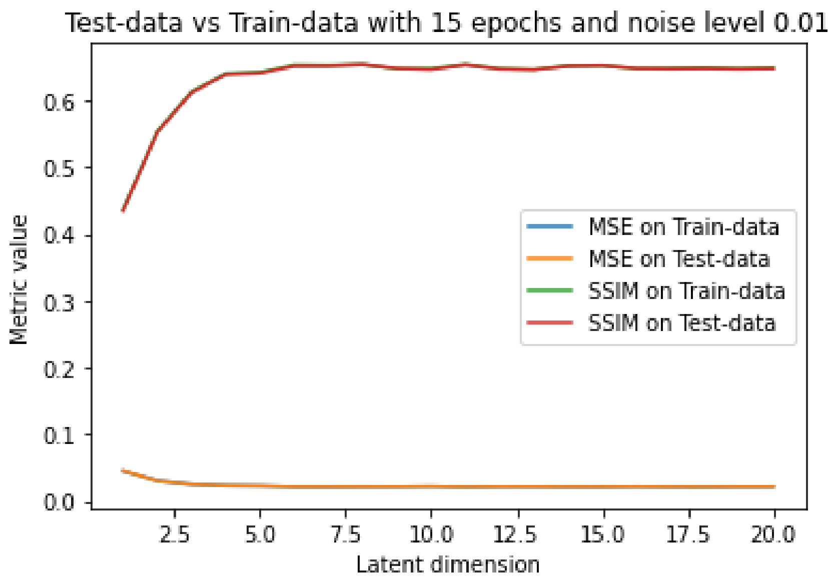

Figure 15: SSIM, MSE for VAE trained on noise with variance 0.01 and mean 0. The curve increases until the minimum number of dimensions to represent the data is reached, then flattens. The test and train performance is identical

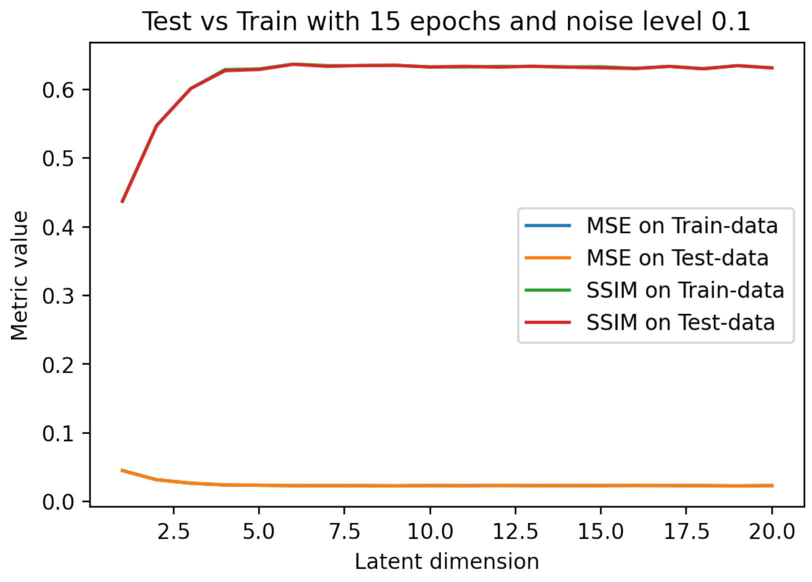

Figure 16: SSIM, MSE for VAE trained on noise with variance 0.1 and mean 0. The curve increases until the minimum number of dimensions to represent the data is reached, then flattens. The test and train performance is identical

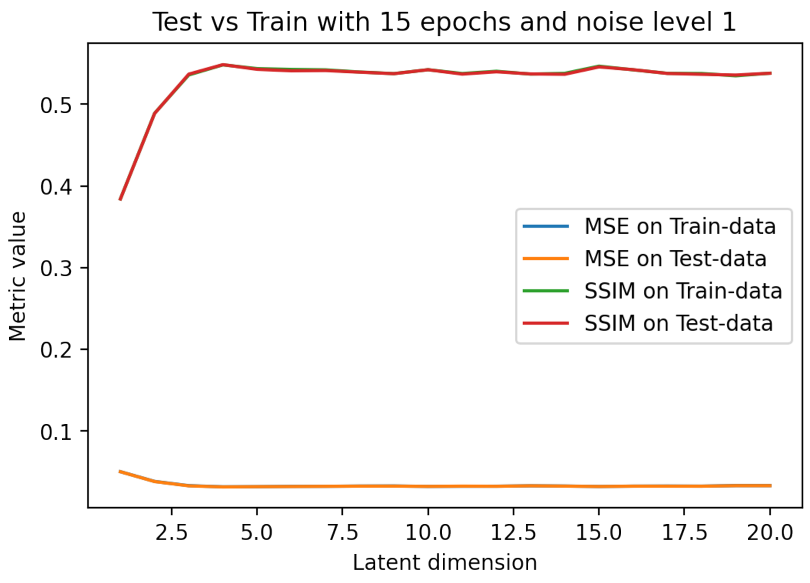

Figure 17: SSIM, MSE for VAE trained on noise with variance 1 and mean 0. The curve increases until the minimum number of dimensions to represent the data is reached, then flattens. The test and train performance is identical

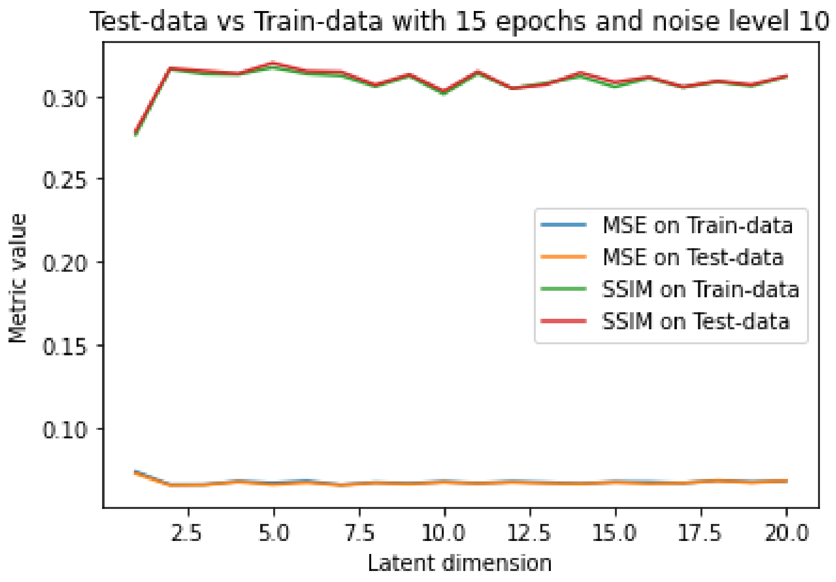

Figure 18: SSIM, MSE for VAE trained on noise with variance 10 and mean 0. The curve increases until the minimum number of dimensions to represent the data is reached, then flattens. The test and train performance is identical

References

[1] Plaut E 2018 From principal subspaces to principal components with linear autoencoders (*Preprint* `1804.10253`)

[2] Rocca J Understanding variational auto-encoders URL `https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73`

[3] Thomas R Fourier transforms of images URL `https://plus.maths.org/content/fourier-transforms-images`

[4] Wikipedia Band-stop filter URL `https://en.wikipedia.org/wiki/Band-stop_filter`

[5] christopher polack Image filtering and hybrid images URL `https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj1/html/cpolack6/index.html`

[6] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J and Chintala S 2019 Pytorch: An imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems 32* ed Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E and Garnett R (Curran Associates, Inc.) pp 8024–8035 URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`

[7] Lectures S Image denoising by fft URL `http://scipy-lectures.org/intro/scipy/auto_examples/solutions/plot_fft_image_denoise.html`

[8] Palla A, Moloney D and Fanucci L 2017 Fully convolutional denoising autoencoder for 3d scene reconstruction from a single depth image pp 566–575

[9] Hinton G E and Salakhutdinov R R 2006 *science* **313** 504–507

[10] HandWiki Universal approximation theorem URL `https://handwiki.org/wiki/Universal_approximation_theorem`

[11] te Rietmole P 2021 An introduction to dimensionality reduction Tech. rep. Utrecht University

[12] Maier A Unsupervised learning — part 2 URL `https://towardsdatascience.com/unsupervised-learning-part-2-b1c130b8815d`