university of groningen / faculty of science and engineering / artificial intelligence

Improving natural image denoising using a multilayer perceptron

Bachelor's Project Thesis

Herman Groenbroek, s2593386, h.g.groenbroek@student.rug.nl
Supervisor: Dr. M.A. Wiering

Abstract: Image noise reduction is a complex task with no perfect solution. Clever algorithms exist to reduce noise with the disadvantage that details in the image are also reduced. A multilayer perceptron, being a universal approximator, can be used to optimize the problem of noise reduction while preserving image details. In this thesis we find which hyperparameters work best for improving image denoising performance on colour images. Comparing the two activation functions TanH and ReLU, the latter performs best when used with a single hidden layer and without the use of dropout. Our results show superior performance over a baseline denoising algorithm. The results on parameter tuning may aid future image denoising with multilayer perceptrons.

1 Introduction

Noise reduction has been heavily studied in the field of computer science. Any sensor that captures a signal is inherently prone to noise, where noise is an unwanted, unpredictable change in the signal. In electronic sensor systems, random fluctuations in electrical current contribute to the noise in the signal. Such systems are furthermore affected by thermal noise in accordance with the fluctuation-dissipation theorem (Callen and Welton, 1951). In digital image sensors we find that many more types of noise exist, such as photon shot noise and anisotropic noise. As such, reducing the noise thereby increasing the signal-to-noise ratio is complex but highly beneficial for any sensor system. In this thesis we focus on the digital image sensor and thus on reducing noise in natural images.

Image noise manifests itself as changes in pixel values. Certain types of noise follow a specific distribution. For instance, photon shot noise corrupts images according to the Poisson distribution, meaning that only a small portion of the image has a large amount of noise. Magnetic resonance images often feature noise following the Rice distribution. The most common type of noise found in images however follows the Gaussian distribution. Many researchers in this field assume the noise to be AWG, i.e. additive, white and Gaussian (Yu and Sapiro, 2011; Dabov, Foi, Katkovnik, and Egiazarian, 2007; Elad and Aharon, 2006). Some studies have focused on smoothing out the noise in an image (Tomasi and Manduchi, 1998; Wink and Roerdink, 2004) whereas others define algorithms to cleverly use properties of natural images to improve denoising, BM3D being a prime example of this (Dabov et al., 2007). With machine learning gaining traction in the scientific world, it should come as no surprise that researchers have used neural networks as an attempt for a supervised learning approach to image denoising (Egmont-Petersen, de Ridder, and Handels, 2001). In fact, it has been shown that a neural network approach can match and even surpass the performance of state-of-the-art denoising algorithms (Burger, Schuler, and Harmeling, 2012).

Noise reduction in general can be seen as a problem of mapping noisy data to a noise-free variant. That is, we look to find a function f such that $y = f(x)$ where x is data corrupted by noise and y is the data without noise. The complexity of this problem lies in distinguishing what part of the data is considered noise and what part belongs to the original signal. Neural networks can assist us in finding this function. Convolutional neural networks (CNNs) work especially well on image data due to the network structure utilizing local receptive fields, decreasing the number of parameters and thereby increasing potential efficiency. CNNs have been successful at denoising images (Jain and Seung, 2008). Contrary to CNNs however, mul-

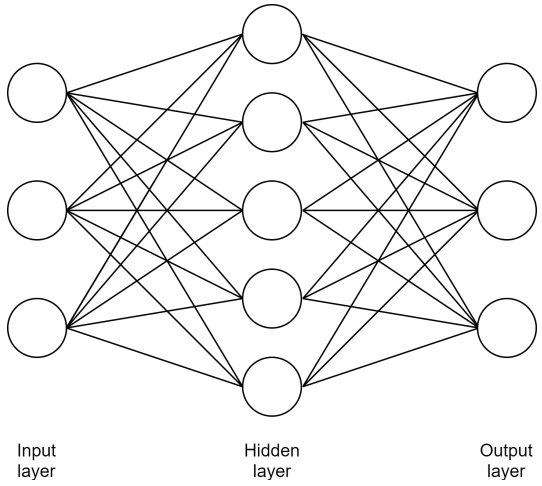


Figure 2.1: Basic model of a multilayer perceptron with a single hidden layer, also described by '(3,5,3)-MLP'.

tilayer perceptrons (MLPs) have been shown to be universal approximators (Hornik, 1989, 1991), given sufficient hidden neurons and training data. As such, we will be focusing on improving MLPs for image denoising, continuing on the work by Burger et al. (2012). We will take a closer look at the parameters of the network and deduce what is beneficial for the network to further improve image denoising, coming one step closer to the ideal image denoising function.

2 Methods

We are using a multilayer perceptron (MLP), a feedforward type of neural network. This type of network can be regarded as a graph of at least three layers of nodes: an input layer that receives the input values, one or more hidden layers and an output layer that yields the result. Each node is fully connected to the nodes in the following layer, see Figure 2.1. The connections between nodes are weights that the values are multiplied with. The input values of a node are then summed and an activation function is applied. To train the network, the weights are adjusted through backpropagation based on the error of the obtained output compared to the target output.

An MLP has a large number of hyperparameters that require tweaking for better results. The number of hidden layers for instance and the number of nodes in a hidden layer both affect the complexity of a function that the MLP learns. Too few hidden nodes results in overgeneralization: the network will not have the required complexity to approximate the function and thus yields poor performance. On the other hand, too many hidden nodes may result in overfitting, meaning that the network learns to optimize performance for the training data only, leading to worse results when the network is run with other data. The activation function also affects how well the network learns. Commonly used activation functions include the hyperbolic tangent, or TanH,

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \qquad (2.1)$$

and the rectified linear unit, ReLU

$$f(x) = \begin{cases} 0 & for \quad x < 0 \\ x & for \quad x \geq 0\,. \end{cases} \qquad (2.2)$$

Finally, since overfitting is a common problem in deep neural networks, there is a method referred to as dropout which aims to prevent this (Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov, 2014). Dropout removes a percentage of random nodes from the network, creating a "thinned" network for training. For every new presentation of training data, a new thinned network is randomly sampled. This results in a regularization technique that reduces overfitting. With careful selection of the network hyperparameters, the activation function and the percentage of dropout, we should be able to further optimize image denoising with MLPs.

2.1 Obtaining Image Data

In order to reduce noise in an image with a supervised learning approach, many noise-free images are needed as training data. Since we assume image noise to be additive, white and Gaussian (AWG), training data is easily obtained by manually corrupting noise-free images with AWG noise, see Figure 2.2. We use the CIFAR-10 dataset which contains 50,000 training images and 10,000 test images (Krizhevsky, 2009)*. In our experiments we use

*The test images "cat", "fruits" and "monarch" were obtained from the University of Wisconsin-Madison as 'public-

(a) Original test image "Lenna."

(b) "Lenna" with AWG noise, $\sigma = 25$.

Figure 2.2: Corrupting standard test image "Lenna" with additive white Gaussian noise

5000 randomly sampled images from the training portion for training, and 1000 randomly sampled images from the testing portion for validation. All images in the dataset are downscaled photographs with 32x32 pixels, which ensures they are essentially noise-free.

The idea behind denoising with an MLP is to split an image up into patches, feed the pixel values of a patch to the network and obtain a denoised variant of the patch from the network. Once all patches of an image have been denoised they can be reconstructed into a denoised full image. It is important to split an image into small patches as the input and output dimensionality of the network would otherwise grow to an unusable extent. However, the patches still have to be large enough to contain enough data for the network to learn to denoise with. There is a trade-off between having enough data and being efficient in use.

TensorFlow is used to construct the MLP to further increase efficiency of training. Neural networks are notorious for the extensive training required, as they require innumerable matrix multiplications. Tensorflow is able to parallellize these matrix multiplications when paired with supported Graphics Processing Units (GPUs) in order to speed up the neural network training. The results in this thesis have been obtained using an NVIDIA GTX 650 GPU.

domain test images'. The standard test image "Lenna" was obtained from Wikimedia Commons.

2.2 Experimental Setup

We split the CIFAR-10 images into non-overlapping patches of 15x15 pixels so that every image yields four patches to train with, as we disregard the unused pixels. This gives a total of 20,000 input patches. Using coloured images, every pixel contains three values: the red, green and blue colour channel of the pixel. As such, this gives $15 \times 15 \times 3 = 675$ input nodes for the network; the same holds for the number of output nodes since the MLP will learn a mapping from an input patch to the corresponding noise-free output patch. We corrupt each patch with a significant amount of AWG noise: we take $\sigma = 25$, as was used by Burger et al. (2012). With the ReLU activation function the pixel values are normalized by division of the maximum pixel value. Using the TanH activation function however, every pixel value p is normalized according to LeCun, Bottou, Orr, and Müller (1998) for efficient backpropagation,

$$(p/255 - 0.5) \times 2\,, \tag{2.3}$$

and the output of the network is inversely denormalized. We now have normalized data for noisy patches and their perfect noise-free counterpart. For supervised training we need to quantify the error between the output patch and the clean patch. Since the peak signal-to-noise ratio (PSNR) is a common technique to measure image quality (Horé and Ziou, 2010) and it has a monotonic relation with the mean squared error (MSE), we find this to be a simple and effective error function,

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(p_i - y_i)^2\,, \tag{2.4}$$

where n is the total number of output nodes, p_i is the predicted pixel value of output node i and y_i is its target value, obtained from the original noise-free image. The Adam optimizer is used to minimize the error (Kingma and Ba, 2015). The default settings are used: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. When the network has finished the training session, we may take a test image such as "Lenna", Figure 2.2b, to deconstruct into patches of 15x15 pixels with a stride of 3. Any remaining part of the image that lies outside of the patched area is discarded. Burger et al. (2012) found that smaller strides increase the denoising performance,

although they saw little improvement when using a stride of 1 over a stride of 3. For reconstruction, the weighted average is taken per pixel for better denoising results.

In order to get the results of which hyperparameter settings allow the network to denoise best, we need to pick initial settings that will allow the network to find an approximation to the denoising function. The network is initially run for 20,000 epochs and with a batch size of 300 patches. To ensure the network has sufficient hidden nodes for approximation, we take three hidden layers each with 256 nodes. We start with a dropout of 30% to lower the chances of getting stuck at local minima. The goal is to find the values for these settings that work best, initially testing activation functions with various learning rates.

3 Results

3.1 Learning Rate

The best learning rate for the MLP can be obtained empirically. We are using the two common activation functions TanH and ReLU. Varying the learning rate for both activation functions we find that both allow for approximating the denoising function with our default parameters, see Figure 3.1. With these parameters TanH achieves better results, as long as it passes a local optimum which outputs a noise-free grey-scale image. Interestingly, whether the learning rate using TanH is 0.001, 0.0001 or 0.00001, all MLPs seem to reach this local optimum of a noise-free grey-scale image, indicating that regardless of learning rate the MLPs take a similar optimization route for denoising.

The ReLU starts with a larger error due to a different normalization of the image data. When the learning rate for the ReLU is too large, every output becomes a patch of middle grey, a local optimum. This corresponds to the two lines without an image, the ReLUs with learning rates 0.001 and 0.0001 in Figure 3.1. The learning rate for the ReLU needs to be much lower than the TanH to get usable results. At a learning rate of 0.00001, the output of the network contains the shape of the input and is noise free meaning that the original image is recognizable in the output, but the colours do not match the original image and it is noticeably blurred. A learning rate a tenfold lower was tested using the ReLU but yielded results that were similar yet considerably slower, far from convergence, hence its exclusion from the figure.

Continuing our tests we are updating our default parameters to have a learning rate of 0.001 when using TanH and one of 0.00001 when using ReLU. Neither activation function can be disregarded since both allow for approximating the denoising function.

3.2 Hidden Layers & Nodes

The number of hidden layers and nodes tells us something about the complexity of the function that the MLP is approximating. A highly complex function needs many hidden nodes and possibly multiple layers. With our neural network, more than one hidden layer hinders the performance, having a larger error, see Figure 3.2. Using TanH, the performance with two or three hidden layers stops noticeably improving around 20,000 epochs. With the ReLU the performance is initially worse using multiple hidden layers as compared to one, although around 20,000 epochs it is still improving. The difference in performance between one hidden layer with 128 and 256 hidden nodes is minimal at the end of the training session. The lowest error is obtained with these settings by TanH using a single hidden layer with 256 hidden nodes, hence the default settings for the following tests will use this number of hidden layers and hidden nodes.

3.3 Dropout

Dropout is a regularization technique which allows a neural network to better train its sub-networks in order to minimize overfitting and increase performance (Srivastava et al., 2014). Figure 3.3 shows the performance of TanH and ReLU on the validation dataset for various percentages of dropout. TanH shows the best results after 20,000 epochs when using 30% dropout. Each step of 10% dropout away from 30% decreases performance of the network, with the worst results when no dropout is used. The output images show very similar results, showing a denoised version of the input image although with a slightly different colour tone. The output without dropout retains noise in certain

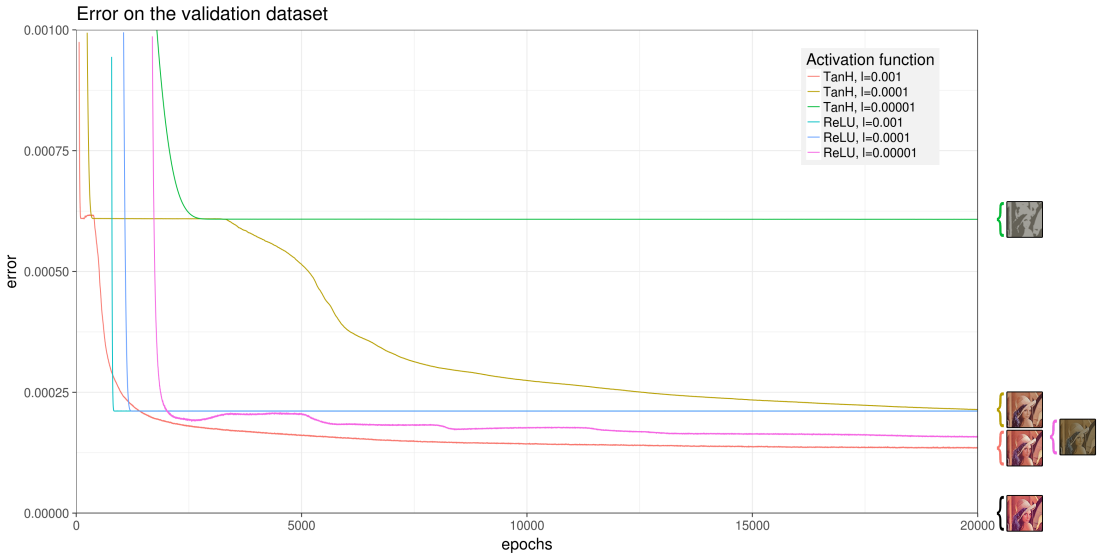


Figure 3.1: Error on the validation dataset with variable learning rates for TanH and ReLU. "Lenna" shown on the right shows the quality of denoising after 20,000 epochs. Note that the error cannot be compared between TanH and ReLU as the data is differently normalized.

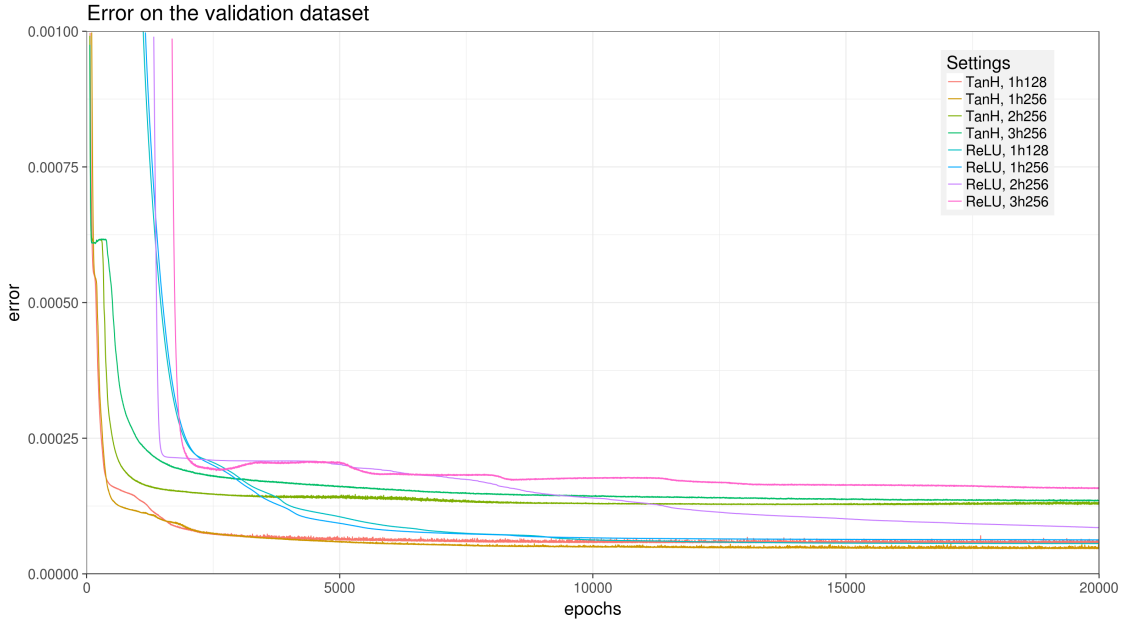


Figure 3.2: Error on the validation dataset with a variable number of hidden layers and nodes, '2h256' depicting two hidden layers with 256 nodes each.

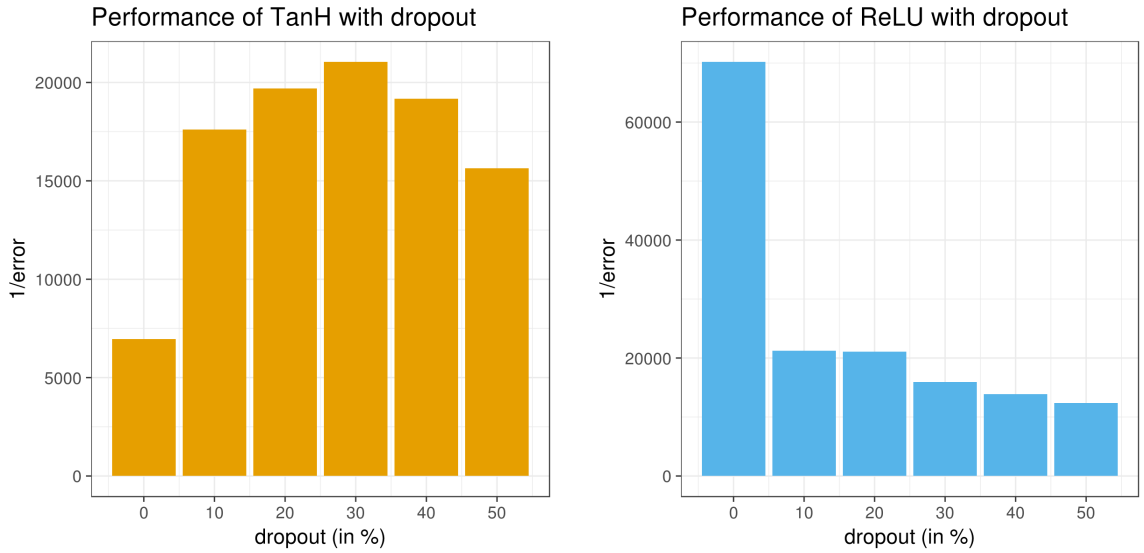


Figure 3.3: Performance of TanH and ReLU with and without dropout. Note that the performance cannot be compared between TanH and ReLU as the data is differently normalized.

parts of the image although less chromatic than the input.

The ReLU on the other hand shows surprising results. The less dropout, the better the network performs. When no dropout is used, the performance is significantly better than with any dropout. It has to be noted however that the ReLU performance without dropout is inconsistent. More often than not, training with these settings yields worse performance than any tested percentage of dropout. Since training only needs to be done once in order to use the approximated denoising function on any image, the better results are shown in the figure. The output images with any number of dropout are denoised but are lacking in vibrance, resulting in a very different look from the original image. The better performing MLP without dropout on the other hand shows images with the exact same colour scheme as the original image contrary to the results from TanH. All AWG noise is reduced, albeit at the cost of reducing sharpness.

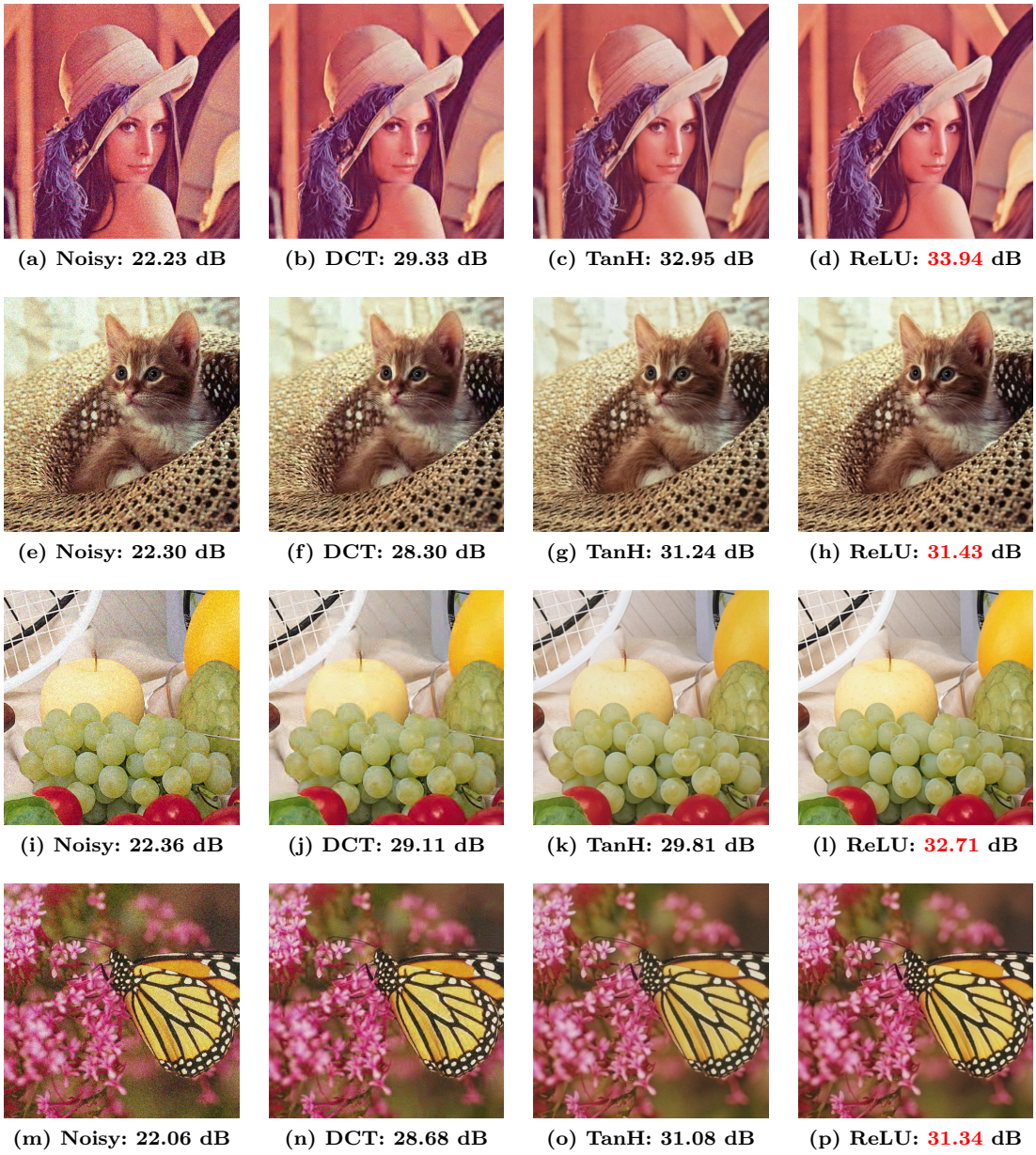
3.4 Activation Function

The best performing MLP hyperparameters have been found for TanH and ReLU from the previous tests. TanH works well with 30% dropout, one hidden layer of 256 nodes and a learning rate of 0.001. ReLU works well without dropout, with one hidden layer of 256 nodes and a learning rate of 0.00001. The final TanH and ReLU MLPs can be compared to existing denoising algorithms. The peak signal-to-noise ratio is used for a fair comparison of the performance of each denoising method. The denoising algorithm by Yu and Sapiro (2011), referred to as DCT, can be considered a baseline for comparison with denoising methods. The comparison of DCT with our methods can be seen in Figure 3.4.

DCT claims to be a simple and effective denoising algorithm. Its simplicity shows in the artefacts that the denoised images contain. It reduces the visible noise, making it indeed effective. Comparing the PSNR with the noisy input, DCT improves the image quality significantly. If we look at our MLP results, TanH improves over DCT on all four test images. ReLU however improves over TanH on all these images as well, making the ReLU the best denoising method for this dataset.

4 Conclusion & Discussion

This study shows the multilayer perceptron hyperparameters that work best for finding an approximation to the denoising function. Two activation functions, the hyperbolic tangent (TanH) and the



(a) Noisy: 22.23 dB (b) DCT: 29.33 dB (c) TanH: 32.95 dB (d) ReLU: 33.94 dB

(e) Noisy: 22.30 dB (f) DCT: 28.30 dB (g) TanH: 31.24 dB (h) ReLU: 31.43 dB

(i) Noisy: 22.36 dB (j) DCT: 29.11 dB (k) TanH: 29.81 dB (l) ReLU: 32.71 dB

(m) Noisy: 22.06 dB (n) DCT: 28.68 dB (o) TanH: 31.08 dB (p) ReLU: 31.34 dB

Figure 3.4: Signal to noise ratio of image denoising methods for comparison. The noisy image is the input, DCT is a baseline denoising algorithm and TanH and ReLU are our results.

rectified linear unit (ReLU), are compared. Using a patch size of 15x15 pixels and three colour values per pixel as the input of the network, we find that TanH works best with a learning rate of 0.001, one hidden layer of 256 nodes and 30% dropout. The ReLU works best with a learning rate of 0.00001, one hidden layer of 256 nodes using no dropout. Training the neural network with a ReLU, it is able to outperform TanH on natural image denoising as well as a baseline denoising algorithm without leaving visual artefacts. This makes the ReLU the preferred activation function for image denoising with a multilayer perceptron using the patching method described in this thesis.

A comparison using our ReLU network with the one by Burger et al. (2012), which uses TanH, would be ideal for reference. This is perhaps an unfair comparison as their MLP was trained on monochromatic noise on greyscale images whereas ours is trained on RGB-noise on full colour images, making the validity of any comparison debatable. It would be interesting to see if their MLP would have performed better with the results found in this study, since this study shows the ReLU achieving better results than TanH. Comparing our results with the DCT algorithm by Yu and Sapiro (2011) shows that our method significantly improves over their simple denoising method. They refer to their algorithm as a baseline for image denoising, making our comparison only valid for claiming that our results are promising. In order to put our performance in context with the best denoising methods, a comparison needs to be made with state-of-the-art algorithms such as BM3D.

In Figure 3.1 the error graph seems to converge before the 20,000$^{\text{th}}$ epoch. In Figure 3.2 however we cannot state as a fact that ReLU 2h256 will remain worse than ReLU 1h256 as it is still learning. The end results of this study have been obtained by training for 20,000 epochs. Marginally better results can be obtained by training for longer. The results from Burger et al. (2012) were obtained by training the MLP for close to a month on a GPU, whereas our training sessions each took three hours to compute. As such it is important to note that this study does not present the best performing denoising method but rather the hyperparameters that allow for a better performing MLP.

This thesis shows the performance of an MLP trained solely on noise with a standard deviation (σ) of 25. Ideally in order to be more relevant, an MLP is trained on noise with any σ so that it may be used in widespread systems that deal with great amounts of image noise. Moreover, the data on which this MLP was trained is not ideal. Although the CIFAR-10 dataset ensures noise-free images due to the amount of downscaling, it also results in many harsh lines in patches that may affect the training session. For instance, since all images in this dataset contain an object and each image is split into four patches for training, it is likely that every patch in the dataset contains an edge of an object. Natural images on the other hand often contain large areas with a single colour and no edges. In fact, this is the basis of BM3D and the reason why it works well on natural images (Dabov et al., 2007).

Image denoising has come close to perfection (Levin and Nadler, 2011). So far most methods for denoising have used only the noisy version of an image to reduce the noise while keeping the edges and details in place. However, one may argue that knowledge of the objects in an image could improve denoising further than any such method by essentially drawing over a denoised image to add known details. Our results show that an MLP learns to blur an image quickly to reduce the noise, followed by adding details and colour as learning continues. The resulting images still look somewhat blurred because of this. If a new method would be able to identify objects in an image based on prototypes, it might be able to add details of the prototype, resulting in less blurred edges in the denoised image.

Many denoising methods have been created and studied using perfect additive white gaussian noise, including the one in this thesis. Natural images taken with digital image sensors however show more types of noise, meaning that not all denoising methods are equally relevant for natural image denoising. Since a multilayer perceptron uses training data for image denoising, it would be interesting for an MLP to be trained on image noise from a single camera model at various ISO levels to see if this in fact improves natural image denoising over existing methods that are based on AWG noise. A downside to this is that a sufficient number of training data is not easily obtained from a single camera model. This thesis shows that for reducing AWG noise, acceptable results can be achieved with only 20,000 training patches, whereas supervised neural

networks generally need training data in the order of millions for proper results. If the same holds true for natural image noise, this could be a simple method to improve image denoising on natural images.

References

Harold C. Burger, Christian J. Schuler, and Stefan Harmeling. Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds. *Journal of Machine Learning Research*, pages 1–38, 2012.

Herbert B. Callen and Theodore A. Welton. Irreversibility and generalized noise. *Physical Review*, 83(1):34–40, jul 1951.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing (TIP)*, 16(8):2080–2095, 2007.

M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks - a review. *Pattern Recognition*, 35:2279–2301, 2001.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing (TIP)*, 15(12):3736–3745, 2006.

Alain Horé and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. *International Conference on Pattern Recognition*, 2010.

Kurt Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2: 359–366, 1989.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4: 251–257, 1991.

Viren Jain and H. Sebastian Seung. Natural image denoising with convolutional networks. *Advances in Neural Information Processing Systems (NIPS)*, 21:769–776, 2008.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pages 1–15, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Dept. of Comp. Sci., University of Toronto, apr 2009.

Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. *Neural Networks: tricks of the trade*, 1998.

Anat Levin and Boaz Nadler. Natural image denoising: Optimality and inherent bounds. *Computer Vision and Pattern Recognition (CVPR)*, pages 2833–2840, jun 2011.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *IEEE International Conference on Computer Vision*, pages 839–846, 1998.

Alle Meije Wink and Jos B. T. M. Roerdink. Denoising functional MR images: A comparison of wavelet denoising and gaussian smoothing. *IEEE Transactions on Medical Imaging*, 23(3):374–387, mar 2004.

Guoshen Yu and Guillermo Sapiro. DCT image denoising: a simple and effective image denoising algorithm. *Image Processing On Line*, pages 292–296, oct 2011.