ELEC 5304 Multi-Dimensional Signal Processing
Project II Report: Image Denoising with Deep Learning

Yang Wang
460273989
ywan3308@uni.sydney.edu.au

Mengwei Yuan
460101132
myua7783@uni.sydney.edu.au

Abstract

The deep learning (deep neural network) approach has shown its enormous potential in solving ranges of image denoising tasks. However, there are still difficulties in achieving both high training speed and outstanding performance. In this report, we investigate some existing neural network models and evaluate their performance by considering not only the image denoising capability but also the time cost for training the model. A classical model named Denoising Convolutional Neural Network (DnCNN) which utilizes the residual learning principle is found to have the property of relatively short training time and high restoration performance for Gaussian noise. Through further study and experiments on this model, we target on several key parameters to adjust them more suitable for short training time and perform some modifications on the structure of the model. These changes make the model shallower and have fewer parameters which allows it to learn quickly and effectively with nearly same performance on image denoising tasks with a certain level of Gaussian noise.

***Keywords*—** Deep Learning, Image Denoising, Convolutional Neural Networks

1. Introduction

As one of the leading-edge technology in image processing, image denoising has been applied to a large range of fields such as electrical devices [1], medical sensor [2], photography [3] and even military [4]. With the help of such algorithms, the performance of corresponding systems could be significantly improved. In different application fields, the objectives and algorithms used for image denoising tasks vary. Among these algorithms, the Gaussian noise reduction is one of the most popular and widely used for the reason that the Gaussian noise, which comes from the natural sources, almost appears in every application.

The algorithm for eliminating the Gaussian noise can be categorized into two classes: the step by step optimization [5] and deep learning method [6]. Both of them utilize the gradient descent theory [7] which is a mature and classical optimization approach. The gradient algorithm converts tasks into a minimum point finding problem. It starts from an initial point of a multi-dimensional surface (hyper-surface) and takes the step in the direction of the gradient. The optimal point is found by iterating this process.

In terms of the step by step optimization approach, after many years study and exploration on computation algorithms, it has been well developed by the contributors all around the world and achieved high performance on reducing the Gaussian noise from images [5]. However, the disadvantage still exists in the aspect of time cost. This approach performs well on individual images but takes a relatively long time period of computing to finish. For a denoising task which contains a large number of images, this step by step optimization approach will be inefficient.

With an increasing number of data emerging and the systemic data sets being built, another approach which is deep learning algorithm is introduced in recent years. Different from the previous approach, the deep learning algorithm generalizes the denoising processes by building and training a model with a set of parameters. This approach significantly saves time for unnecessary computing. The model is trained by a large set of data and can be used as a fixed formula for all the input rather than calculating the gradient and taking steps for every input image.

The deep learning approach shows the potential in image denoising tasks. Many models with high denoising performance are developed such as BM3D [8], WNNM [9] and TNRD [10]. To balance the trade-off between computational time cost and denoising performance, the DNCNN [11] is evaluated as the best model which satisfies both conditions. This model enhances the learning capability by implementing an identity shortcut so that the network performs the residual learning. In addition, the model includes several batch normalization layers to statistically bias the batch data. Both of the techniques avoid the vanishing gradient problem in deep learning [12]. Consequently, the model achieves the highest performance within limited training time and parameters storage space. In this report, therefore, the work will focus on developing this type of model and applying related training procedure to further improve the denoising capability for a certain level of Gaussian noise (σ= 25).

The report is organized in the way of following. Section 2 summarises some previous research about denoising algorithms. Section 3 describes the backbone of the selected neural network model and some modifications. Section 4 provides the experiment results and analysis of key parameters adjusted. The conclusion is drawn in section 5 with future research direction followed in section 7.

2. Previous Work

2.1. Gaussian Denoising Neural Network Models

To identify the neural network model for our improvement, several different structures of networks are investigated. Haoyus team [13] proposed a DN-ResNet which is designed slightly different from the traditional residual neural network. Instead of including the convolutional layer, batch normalization layer into the residual block, this model includes different components into several residual blocks and inserts blocks into a traditional convolutional neural network (CNN). The performance of this model is higher than most of currently existing ones but with large parameters space and a long time for training. With the similar problem, another model named DPDNN [14] stacks 6 denoiser blocks which are connected by the learnable weighted shortcut shown in Figure 1. The denoiser blocks utilize the downsampling and upsampling techniques to reduce the training time but it is still hard to converge. To shorten the training time, DnCNN is proposed [11] which combines the residual learning and batch normalization with adjustable network depth.

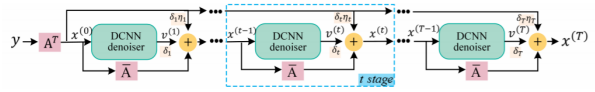

Figure 1. DPDNN Architecture

2.2. Vanishing Gradient

In deep neural networks, addressing gradient vanished problem is the key for a functional model. The vanished gradient is caused by the complex and large number of steps of computation stacking on the input parameters. Those parameters can hardly keep the original feature after times of mathematical computing. This type of problem is addressed by the residual learning concept [12]. It is thought that to learn the residual of input and output value is easier than to learn the output directly. This residual learning usually implemented by a shortcut or identity map between layers so that enhance the ability to keep the original features at each layer. In DnCNN model, the residual learning is constructed by an identity map from the input of the network to the output layer [11]. This long shortcut increase the capacity of the network, in other words, allow the network to be deeper [12].

Another way of solving the gradient problem is the batch normalization [11]. The batch normalization maps data to a distribution with mean of 0 and variance of 1. This normalization keeps the data in its original scale so that keeping the original features. The disadvantage of this is that some outliers may lead to the overbias of data. To filter out these outliers, some non-linear activation functions are applied such as sigmoid which keeps the data in the range of [-1,1] and reduces the impact of outliers. With the batch normalization, the neural network can be constructed deeply and achieve high performance [11] [12] [15].

2.3. Batch Size and Learning Rate

The performance can also be influenced by the training procedure. The batch size and learning rate in the training parameters are important for generating a reliable model [15]. With large batch size, the direction of the gradient is more confidently towards the optimal point however a overlarge batch size may cause the problem that the model converges into a local optimal rather the global optimal. The large batch size usually aligns with the large scale of learning rate. On the other hand, small learning rate provides a high uncertainty level for training, which allows the model to have more possibilities to converge to the global optimal, but sacrifice its speed of convergence for the reason that the learning rate is scaled down by considering that the gradient is calculated based on a small piece of data and has less confidence towards the optimal point.

3. Backbone Network - DnCNN

The proposed network is based on the given DNCNN model. We perform the modifications on its depth, structure, and training procedure.

3.1. Network Depth and Structure

After comparison performance and time cost through experiment on networks of different depth, the network is reduced to 9 blocks from the original DnCNN which contains 17 blocks for faster training speed. The main body of the network is constructed by 6 sequentially connected convolutional blocks. For each block, a convolutional layer, a batch normalization layer, and an activation function are stacked together. The input is connected to the main body via a convolutional layer and an activation function. The output is connected via a smoothing block and an output layer. The filter number is 64 for all convolutional layers except the one inside the smoothing block which contains 32 filters. The kernel size is 3 with the padding of 1 so that the image size can be kept the same as original through the convolution processes. Same with DnCNN, the network employs the identity map to achieve residual learning. Considering the sampling method may cause information lost during the propagation, the layers remain the same size as input to protect the feature distortion and enlarge the trainable parameter space. Figure 2 shows the actual architecture.

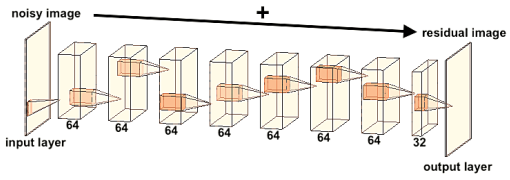

Figure 2. Proposed Architecture

3.2. Training procedure

Considering our objective is to make the network achieve both short time training and high performance, the threshold of 20 epochs is set to make sure the model can converge within a reasonably short period. The model is optimized by the Adam. Through experiments, we set a multi-step learning rate scheduled at 10, 14, 17 epochs where the learning rate will be scaled down by the factor of 0.5. This would accelerate the model to learn more effectively.

Through the exploration of the relationship of batch size and learning rate, the model with the batch size of 32 and the initial learning rate of 1e-3 achieves the best performance among others.

We keep the training data same with the DnCNN model which includes different data augmentations such as rotating and mirroring to diverse the training data. The patch size is 40x40, which is proved to be the most effective patch size for this type of structure and level of noise.

4. Experiments and Analysis

In order to achieve better performance than the given DnCNN model, some of the initial thoughts with respect to improving the performance are tested correspondingly. The PSNR value is chosen as the indicator of the general performance of models and the PSNR of the original DnCNN model is set as the baseline, which is 29.70. The results of some testing and comparisons based on them are shown below.

4.1. Redesign the Network Architecture

According to the previous work that has been done, the plan of improvements over the network architecture is basically based on three models.

4.1.1 DPDNN

An implementation of DPDNN using Pytorch is reconstructed strictly as the paper suggested [14]. However, the testing result of this model is a disappointment and the PSNR only reaches 14.67, which is way much less than the baseline. Another drawback is that due to the complex structure, the model has totally 9x6 = 54 layers and takes a long time and relatively large data sets to train.

What the team also tried is to simply build one basic denoiser block of 9 layers and get rid of the overall connection of 6 denoiser units. The result is even worse maybe because the whole structure is changed.

4.1.2 FFDNet

Taking the limited time into consideration, the team referred to the FFDNet proposed by Kai Zhang and his team [16]. In this network, images are downsampled into four sub-images as input and reconstructed together as output. However, the input patch size in this problem is set to 40x40, which will result in limited information within each sub-image. The reconstruction of the input data generation is not considered for now.

4.1.3 DnCNN

After trying with other models and failing to get comparable results, the team decided to get back to the original DnCNN model and make some adjustments on the parameters to improve its performance.

The first thing to consider is the depth of the architecture because deeper networks usually have better performance. Set the epoch to be 5, a quick test among different depths is done and the result is shown in Table 1.

Depth	5	7	9
PSNR	28.31	28.58	28.21

Table 1. Comparison on Depth

The second thing worth thinking about with the layer structure is the order of Rectified Linear Unit (ReLU) and Batch Normalization (BN). Putting ReLU before BN helps the parameters get well distributed and mean-centered. Putting ReLU after BN will result in half of the normalized data wiped out and much information is not fully used, sometimes produces activation with stable distribution though. There is no certain answer to how to arrange them so we trained both models and here is the comparison of the result in Table 2. The resulting PSNR value slightly changed so we stick to the original order which is ReLU after BN.

Order	ReLU before BN	ReLU after BN
PSNR	29.87	29.89

Table 2. Comparison on Order of ReLU & BN

The team also came up with an idea of adding a layer to gently connect the 64 channels at last, which is so-called a smoothing layer. This is achieved by firstly cutting 64 channels down to 32 and then to 1 to return the output. Although not a big change in the result is made according to Table 3, adding this layer does help with a better outcome.

Smoothing layer	With	Without
PSNR	30.14	30.10

Table 3. Comparison on the Smoothing Layer

Changing the weight initialization to Xiaver is also tried and the resulting PSNR is 27.8.

4.2. Training Procedure

Apart from the model itself, there are still a lot can be done to achieve goals. During the training procedure, the setup of parameters will impact the time spent on training and resulting performance. The adjustments on parameters are considered in the following scopes.

4.2.1 Hyper-parameters

Taking the batch size to be smaller than the total number of samples, a larger batch size will result in longer training time and worse accuracy according to our tests shown in Table 4. In this case, the trained model with batch size of 64 outperforms.

Batch Size	256	128	64	32
PSNR	29.93	30.14	30.17	30.16

Table 4. Comparison on Batch Size

By applying the original model and training in different epochs, it is found that the best choice of epoch number depends on the

complexity of the model, which is quite simple in our case. Considering the trade-off between training time and performance, a suitable value of epoch number is around 20.

4.2.2 Optimizer

As Stochastic Gradient Descent (SGD) takes smaller local optimum steps while Adam makes the decision of direction considering previous movements, the training of SGD is expected to take a longer time and result in more accurate results. The obstacle of building up SGD training is tuning the learning rate in this case and improper learning rates will lead to loss explosion quickly from our experience.

4.2.3 Scheduler

The setup of the learning rate can greatly affects the training results. With the chosen scheduler to be multistep learning rate, adjustments over initial learning rate, milestones, and gama are available. According to the baseline performance, the default gama and initial learning rate values have already fitted enough to this model, so more focus is put on tuning the milestones.

The loss function curve during training when the milestones are set to be [10,14,17] is shown in Figure 3. It can be clearly observed that after each milestone point the loss drops faster than before, which shows it helps with better convergence.

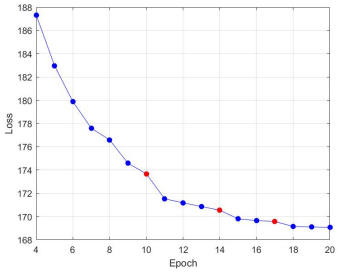

Figure 3. Loss Convergence with Milestones [10,14,17]

5. Our Improvement

From the analysis upon all testing results and comparisons, the final choice of model as well as setup of parameters are:

Network Architecture	DnCNN
Adjustment on Model	Adding smoothing layer
Optimizer	Adam
Depth	9
Epoch	20
Batch size	64
Patch size	40

Table 5. Setup of Final Model

During the decision making of the final choice of the models, the team has come through some milestone models where some certain parameters have shown their great impacts on the performance. Their PSNR curves are plotted in Figure 4, from which the comparisons are clear and significant. All chosen models basically converge after around 10 epochs and barely fluctuate.

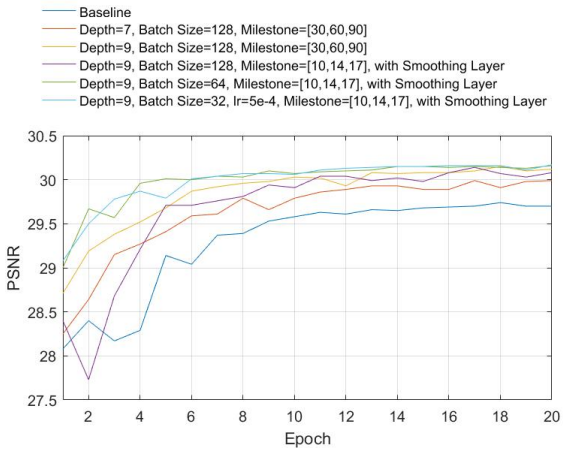

Figure 4. Comparison on PSNR

The performance is obviously enhanced comparing to the benchmark set initially according to Table 6.

	PSNR	SSIM
Baseline	29.70	0.9158
Improved model	30.14	0.9245

Table 6. Comparison of Baseline & Final Model

Figure 5 shows some results for visualization of the image denoising procedure.

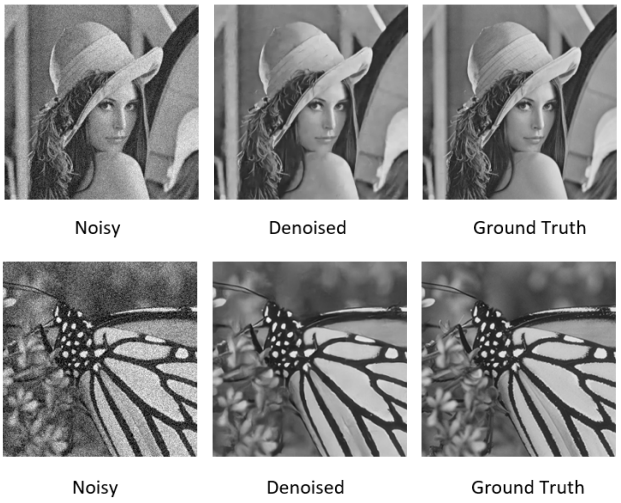

Figure 5. Denoising Results

6. Conclusion

In this project, we made careful adjustments to the provided model and enhance the performance of it. The resulting PSNR of this improved model over testing set Set12 reaches 30.17 and the improvement is obvious compared to the baseline PSNR (29.70). The original DnCNN is proved to function very well and there are not many changes can be done to improve. Two major factors in the improvements are the learning rate and batch sizes. They have shown great impacts on increasing resulting PSNR values. The failure of other models may be attributed to not fitting this problem or some other parameters during the implementation of these models are not properly set up.

7. Future Work

Since the same testing set (Set12) is used to evaluate the performance every time, this might lead to overfitting of the model to the training set and testing set. To avoid this, validation sets can be used to evaluate the performance and get a model suitable for more general image denoising problems.

During the tuning of parameters, comparisons over each group of parameters are simply considered ignoring the impacts from other fixed parameters in this case. More comparisons are supposed to be done to reach a better result if given more time.

So far, our efforts are put on the improvement of the model itself. Changing the way how image data is preprocessed might also result in better results and is worth further tests and analysis.

8. Group Contribution

All the work during the project is conducted by the team members together with fully in time discussion and communication.

All the training and testing are equally assigned to both team members and result data are shared in time. For the improvements of the model, Mengwei is in charge of the implementation of other models and modifications of the DnCNN architecture. Yang takes the charge of tuning parameters of training procedure. Most changes of the model are carefully made after detailed discussion about previous results of the group. Each group member individually did research about the theories behind the parts they were working on and shared the gained knowledge and understanding.

As regards presentation of the final results, Mengwei has mainly prepared for the speech content and Yang arranged the content and formatted slides correspondingly.

The final report is done with equal division of workloads to both group members and integrated by Yang for formatting and error checking.

References

[1] P. D. P. Cattin, "Image restoration: Introduction to signal and image processing," https://miac.unibas.ch/SIP/06-Restoration.html, 2016, accessed: 2019-06-02.

[2] S. Li, H. Yin, and L. Fang, "Group-sparse representation with dictionary learning for medical image denoising and fusion," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 12, pp. 3450–3459, 2012.

[3] A. Buades, B. Coll, and J.-M. Morel, "Nonlocal image and movie denoising," *International Journal of Computer Vision*, vol. 76, no. 2, pp. 123,139, 2008-02.

[4] V. Soni, A. K. Bhandari, A. Kumar, and G. K. Singh, "Improved sub-band adaptive thresholding function for denoising of satellite image based on evolutionary algorithms," *IET Signal Processing*, vol. 7, no. 8, pp. 720–730, October 2013.

[5] J. M. Bioucas-Dias and M. A. T. Figueiredo, "Multiplicative noise removal using variable splitting and constrained optimization," *IEEE Transactions on Image Processing*, vol. 19, no. 7, pp. 1720–1730, July 2010.

[6] H. R. Shahdoosti and Z. Rahemi, "Edge-preserving image denoising using a deep convolutional neural network," *Signal Processing*, vol. 159, pp. 20 – 32, 2019.

[7] X. Luo, H. Liu, G. Gou, Y. Xia, and Q. Zhu, "A parallel matrix factorization based recommender by alternating stochastic gradient decent," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 7, pp. 1403 – 1412, 2012.

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, Aug 2007.

[9] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 2862–2869.

[10] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, June 2017.

[11] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, July 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun.

[13] H. Ren, M. El-Khamy, and J. Lee, "DN-ResNet: Efficient Deep Residual Network for Image Denoising," *arXiv e-prints*, p. arXiv:1810.06766, Oct 2018.

[14] W. Dong, P. Wang, W. Yin, and G. Shi, "Denoising prior driven deep neural network for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[15] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv e-prints*, p. arXiv:1502.03167, Feb 2015.

[16] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, Sep. 2018.