EFFECTIVENESS OF U-NET IN DENOISING RGB IMAGES

Rina Komatsu and Tad Gonsalves

Department of Information & Communication Sciences,Faculty of Science & Technology,Sophia University, Tokyo, Japan

ABSTRACT

Digital images often contain "noise" which takes away their clarity and sharpness. Most of the existing denoising algorithms do not offer the best solution because there are difficulties such as removing strong noise while leaving the features and other details of the image intact. Faced with the problem of denoising, we tried solving it with a Convolutional Neural Network architecture called the "U-Net". This paper deals with the training of a U-Net to remove 3 different kinds of noise: Gaussian, Blockiness, and Camera shake. Our results indicate the effectiveness of U-Net in denoising images while leaving their features and other details intact

KEYWORDS

Deep Learning, Image Processing, Denoising, Convolutional Neural Network, U-Net.

1. INTRODUCTION

Today's imaging devices such as digital cameras, smart phones and video cameras invade our lives, we become constantly aware of the need for more and more clear images. However, there is a rather formidable obstacle named "*noise*" which makes the digital images taken by these devices hard to appreciate. Noises are generated from transferring images in the digital media, due to human error while taking pictures, and while restoring compressed images back to their original size. The kinds of noises are various: some of them make the image gloomier, some make them sandy and so on. There are denoising solutions to deal with each kind of noise. However, the denoising process may take time in detecting the kind of noise and applying the relevant correcting algorithm with optimized parameters. There are other limitations, too. For example, if the noise is too strong, it is hard to recognize the objects in the image.

To remove various noises no matter how strong, technology like the human brain which can distinguish between noise and the original image is needed. In recent years, AI deep learning algorithms and techniques are rapidly progressing to handle recognition, segmentation, re-production of images, etc. Using deep learning techniques, we investigated if it is possible to build a model that can effectively denoise noisy input images and get clear output images without any traces of noise.

In this paper, we picked up the CNN architecture "U-Net" as a typical denoising deep learning model and developed our own "Reformed U-Net" to handle strong noise in the images. In Section 2, we introduce 3 kinds of noises we used as denoising targets. Section 3 does a brief U-Net overview and introduces comparative studies of the 2 models based on U-Net; Section 4 and 5 describes programming environment and experimental setup; Section 6 indicates denoising results; Section 7 concludes this study.

2. OVERVIEW: NOISE USED IN OUR EXPERIMENT

In our study, 3 kinds of noises: Gaussian noise, Blockiness and Camera shake are selected as the target for image correction. This section introduces these types of noises and their effects.

2.1. Gaussian Noise

Gaussian Noise makes image quality sandy due to the problem in electronic circuits or sensors. As the value consists of Gaussian distribution covering the original image, RGB pixels in the original image are collapsed and lose the special information such as contrast, brightness and so on. Figure1 shows the original image and the one in which Gaussian noise is added.

In image proceeding field, filtering process is usually used to reduce the effect of this noise [1]. This process is convolute to noisy image with a filter such as median filter, gaussian filter etc. The problem with this solution is filtered image become less brightness since the pixels near the noise is also convoluted.

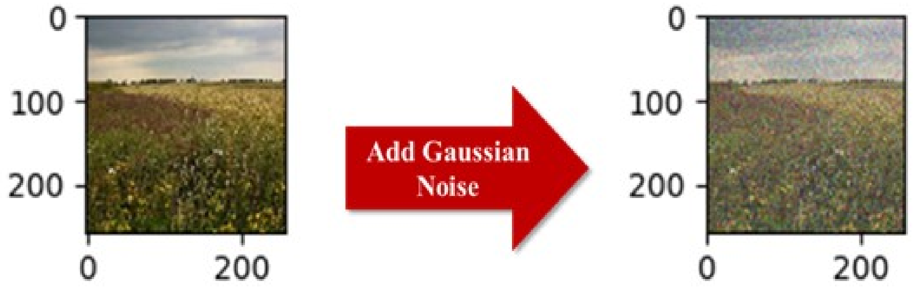

Figure1. Adding Gaussian noise

2.2. Blockiness

When uploading images on to internet servers, compression becomes necessary because of their large quantity of bytes. When restoring the compressed images, block effect comes into play blurring the smooth edges. Figure 2 shows generating blockiness by resizing the original image into a smaller size and then returning the compressed image back to the original size. Filtering process is also used to extract edge information [2].

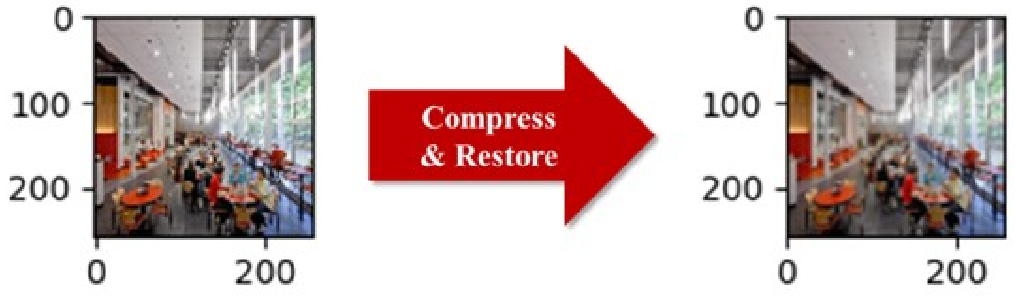

Figure 2. Generating Blockiness

2.3. Camera Shake

Camera shake happens when we take pictures with trembling or non-steady hands. As a result, the output becomes a picture which is out of focus and it is hard to detect objects in such blurred images (Figure 3). There is the technique named blind deconvolution which correct images without hardware like gyro sensor, only blurred image. As the example study in detecting blur kernel, output Fourier transformed image from blurred one and get blur vector and length [3] is picked up.

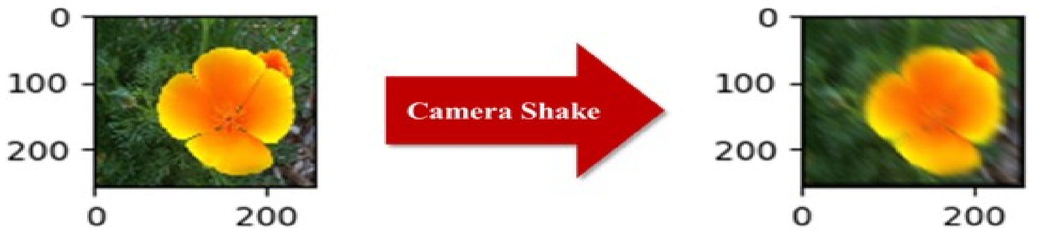

Figure 3. Generating Camera shake

3. Overview: Deep Learning Models

In deep learning field, Vincent et al proposed learning architecture named Denoising Autoencoder [4]. This architecture train neural network model to be able to reconstruct clear data from noisy input. As a denoising neural network model, this section introduces 2 models.

3.1. U-Net

In our study, U-Net is adopted as the generator model. U-Net is the one of the Convolutional Neural Network (CNN) architectures proposed by Ronneberger et al. The architecture of the network shown in Figure 4 describes the alphabet “U”. The left side which consists of the Convolution layers is called the contracting path and the right side which consists of Deconvolution layers is called the expansive path. As explained in [5], the contracting path gets the context, while the expansive path holds the precise localization.

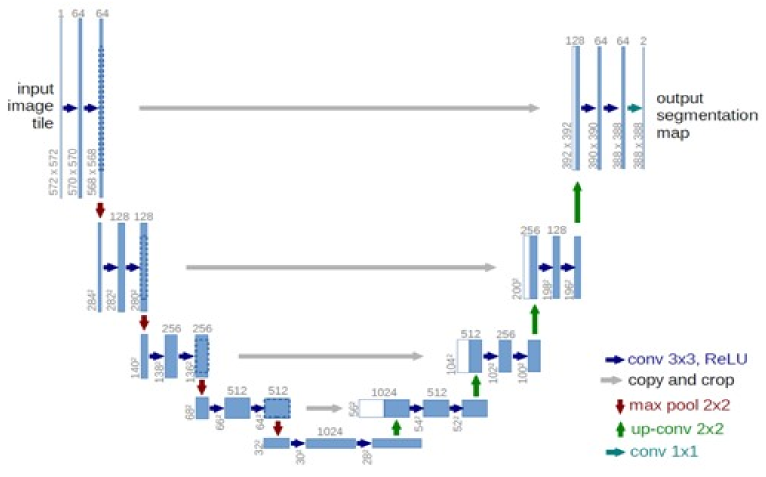

Figure 4. Architecture of U-Net (quote from Fig.1. from [5])

This U-Net is not only used in image segmentation. Isola and others who propose image architecture “px2px”, utilizes U-Net as generator model to transform daylight scene into night scene, to generating a map from an aerial photograph, and painting from an edge image and so on [6].

Referring to Figure 4, the sizes of the input and the output image are different. When convolute to input, output size relies on the parameter: filter size, stride size and padding size. Under formula (1) shows the output sizes of height (h_output) and width (w_output). (Formula (1) referred from a part of document [7])

$$h_O = \frac{h_I + 2h_P - h_K}{s_Y} + 1$$
$$w_O = \frac{w_I + 2w_P - w_K}{s_X} + 1 \qquad (1)$$

$(h_O, w_O) \coloneqq output\ image\ size$

$(h_I, w_I) \coloneqq input\ image\ size$

$(h_K, w_K) \coloneqq filter\ size$

$(s_Y, s_X) \coloneqq stride\ direction$

$(h_P, w_P) \coloneqq padding\ size$

To be able to compare images of identical size, we reconstruct U-Net by adjusting several parameters in each layer. Figure 5 shows the U-net which outputs same sized images used in our study.

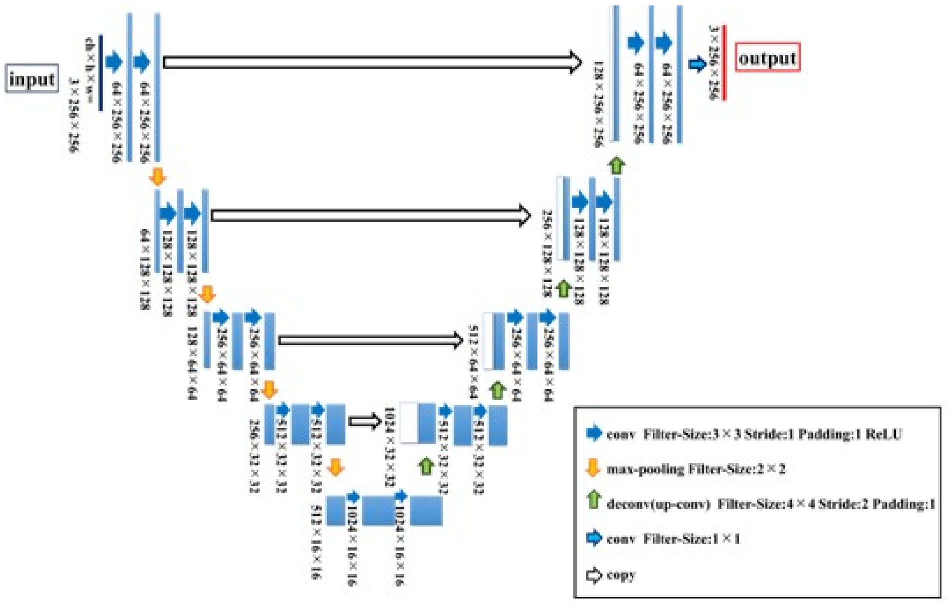

Figure 5. Architecture of U-Net used in this study

3.2. Reformed U-Net

According to some techniques adopted by DCGAN: Deep Conditional GANs [8], performance is enhanced by replacing pooling layers with fractional-stride convolutions and using Batch Normalization (Batch Normalization normalizes layer's input and helps higher learning rate [9].) to each layer's output in model. Following these techniques, we reformed the U-Net shown in Figure 5. We named this proposed U-net "Reformed U-Net". Figure 6 shows the architecture of Reformed U-Net.

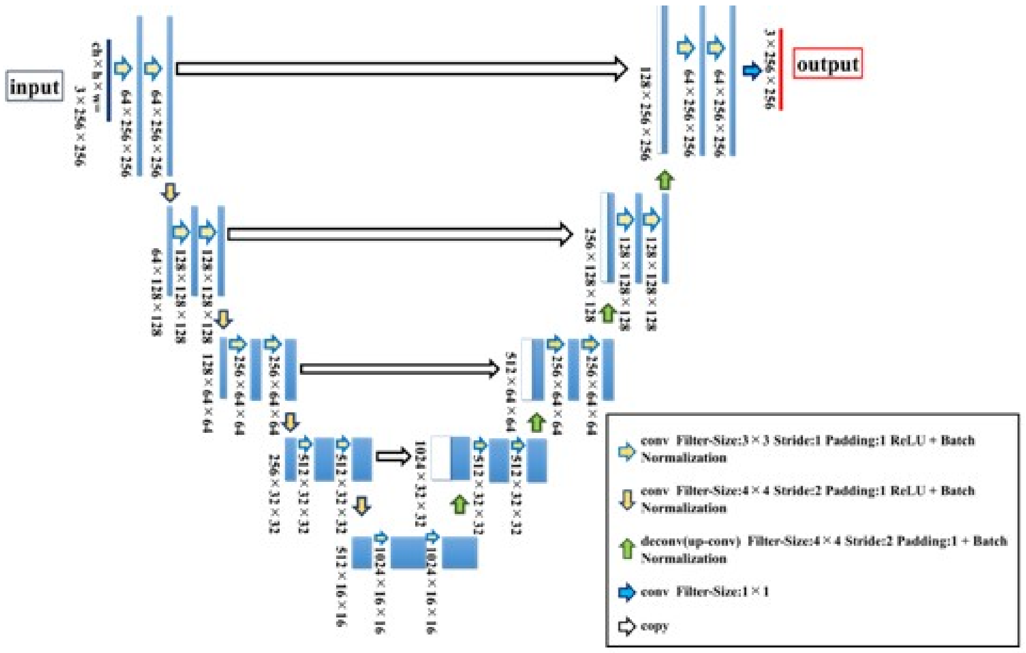
input
output
conv Filter-Size:3×3 Stride:1 Padding:1 ReLU + Batch Normalization
conv Filter-Size:4×4 Stride:2 Padding:1 ReLU + Batch Normalization
deconv(up-conv) Filter-Size:4×4 Stride:2 Padding:1 + Batch Normalization
conv Filter-Size:1×1
copy

Figure 6. Architecture of Reformed U-Net

4. Experiment Environment

4.1. Image Dataset

As the materials to adding noise, we choose the ADE20K dataset [10]. This dataset was used in the image segmentation competition and is rich in the variety of landscape pictures. This dataset has been split in 20,210 images for training and 3,352 images for testing in advance.

4.2. Programming

Our study tried using "chainer" [11] as the deep learning framework. This framework is used from building deep learning models to training and testing. Also, to deal with image processing in programming, the image library "OpenCV" is employed. Noisy images are generated using OpenCV, while inputting the model and training it to be able to output clear images is performed using the functions provided by the chainer framework.

5. Experiment

5.1. Creating Noisy Images

First, all images, training as well as testing were resized to 256 x 256 pixels. Noise is then added to each resized image. Gaussian noise is generated by overlapping a clear image with Gaussian distribution whose parameter σ is in the range [15, 50]. In generating Blockiness, images are first resized in the scale [1/4, 1/2] and then reframed back to the original 256 x 256 px size. Camera shake is generated on purpose by overlapping a few copies of the same images, each with a varying focus.

5.2. Training

Next is training each model to generate clear images from noisy ones. Each of the 3 steps described below counts as 1 epoch. Training is carried on for 100 epochs with mini batches of 15 images per batch.

Step 1. Load noisy image as the input of the model and non-noisy clear image as the target.

Step 2. Normalize noisy image by dividing by 255. Then, input to the model and get output.

Step 3. Calculate loss by Mean Absolute Error (MAE) between output and target and update the parameters in U-Net by optimizer Adam [12] which is attached chainer's library.

5.3. Evaluating The Trained Model

To evaluate how the trained model could render clear images from the input noisy images, we employed 2 different estimate methods. The 1st method relies on the loss changes which calculates the MAE during the training to estimate whether the output from the model comes near to the target. The 2nd method is visualizing the output in RGB image and comparing it with the target image by calculating the MAE from pixel values between the visualized output image and the target image. Here we use PNSR: Peak signal-to-noise ratio which is the criterion for determining how close the output resembles the target.

6. RESULTS

6.1. Loss Changes

To evaluate whether there was an over-fitting in the trained, we compared the loss changes using the training and the testing datasets. Figure 7 shows the graph of loss change recorded from epoch 1 to 100. From the start to finish, both the models steadily reduce the value of the loss function.

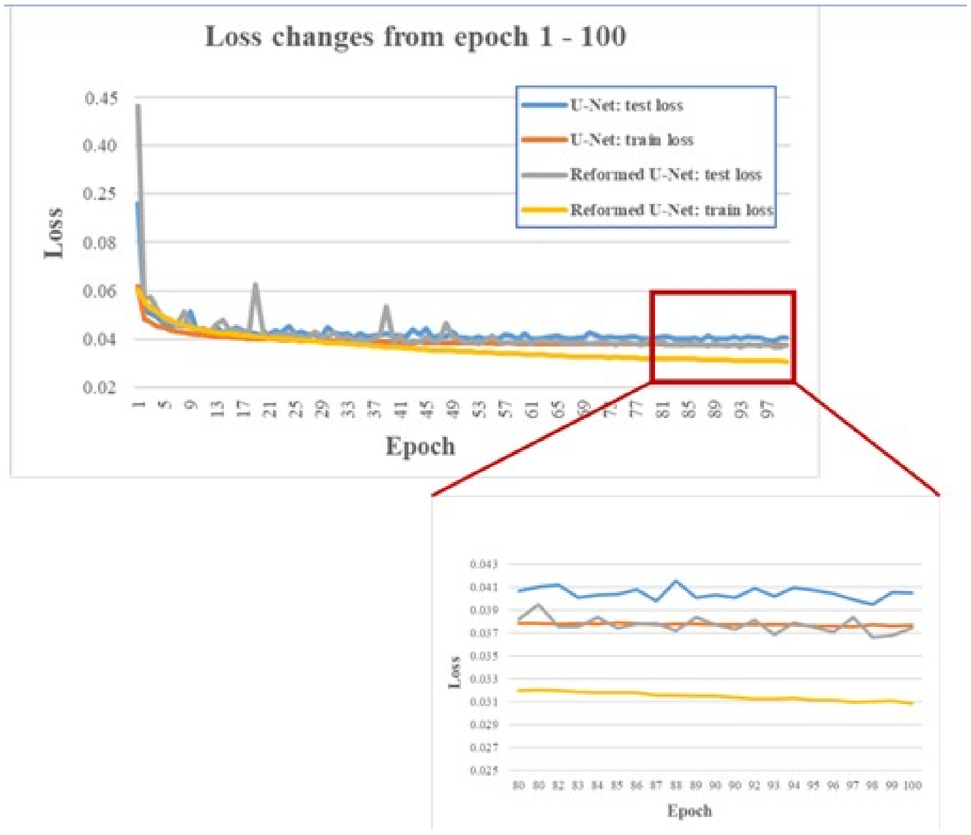

Figure 7. Loss changes from epochs 1-100 in training and testing

Referring to the loss in training and testing in Figure 7, we notice that each loss becomes less and less as the epoch proceeds. Both the models did not show any overfitting, a phenomenon in which the loss change in the case of testing remains relatively flat while that of the training phase steadily decreases. On finishing training and testing the model, the loss of U-Net settled on 0.038 in training and 0.041 in testing, while the Reformed U-Net settled on 0.031 in training and 0.037 in testing.

6.2. Visualizing Output Of The RGB Image

Since both the models could decrease the loss in training and testing, we should evaluate how clearly the model succeeds in removing noise from a given noisy image. Figure 8 shows the visualized output result using the model trained through 10, 50, and 100 epochs. Using the PNSR and MAE criteria, both models could denoise image as PNSR increases and MAE decreases compared to the noisy image. Also, the model trained over a greater number of epochs produced improved results than the one with a smaller number of epochs.

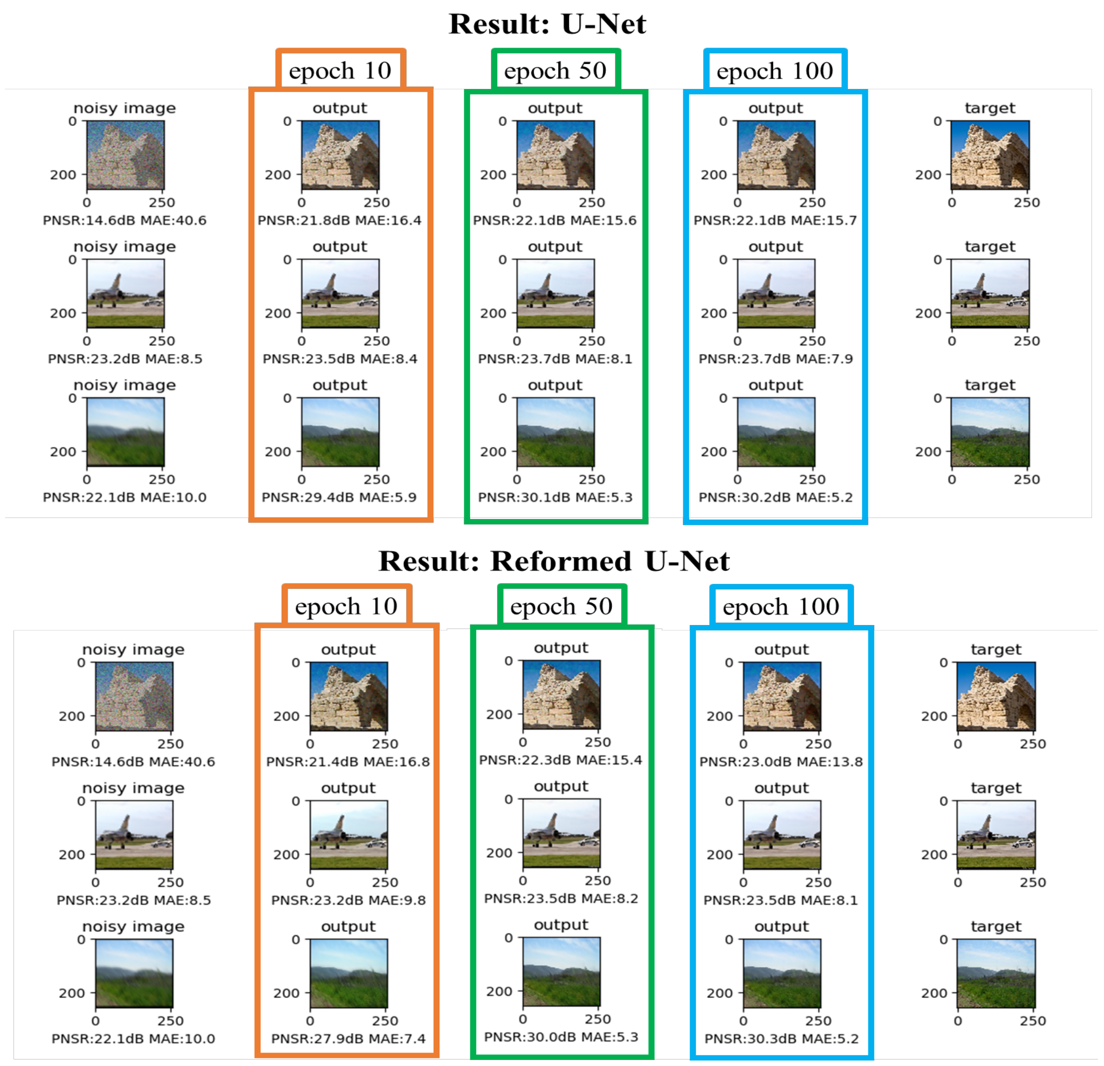

Figure 8. Visualized output result

6.3. Comparison In Dealing With Strong Noise

The results in section 6.1 and 6.2 indicate that the deep learning model based on U-Net has enough ability to get rid of different kinds of noise. This study picked up 2 models, the U-Net and the Reformed U-Net. Which model is better for denoising strong noise? To ascertain this, we had additional experiment steps described below:

Step1. Add strong noise to test images by adjusting the parameters described in section 5.1.
Step2. Input strong noisy image which has been normalized and get the output.
Step3. Calculate loss by MAE.
We try denoising each kind of noise one by one. Table 1 shows the result of loss. We observe that the Reformed U-Net is better than U-Net in dealing with strong noise.

Table 1. Comparison result dealing with strong noise.

	Gaussian Noise	Blockiness	Camera Shake
U-Net	0.0676	0.0423	0.0456
Reformed U-Net	0.0605	0.0422	0.0426

7. CONCLUSION

Our results show that Deep learning networks like the U-Net are effective in denoising RGB images. Further, the Reformed U-Net we proposed in this study can deal with strong noise more effectively than the U-Net. The reason why the neural network model based on U-Net succeeded in denoising might be associated with the connections between contracting path and expansive path. According to Mao Xiao-Jiao and others who proposed the model architecture "skip connections" (skip connection is linking between corresponding convolution and deconvolutions), skip connection has effect to gradient vanishing problem and be able to pass image details from convolution layers to deconvolutions layers which roles recovering noisy input [13]. Like U-Net which has linking architecture could learn denoising in stable and get clear output.

However, the effect in denoising blockiness was not more outstanding than the other noises. It seems to be associated with the measure used to obtain loss. The Mean Absolute Error calculates the loss depending only on the value in each pixel and not the visual aspect. To consider the visual aspect, we may need the discriminator model which learns through training to distinguish the fake images (i.e. output from the generator model) and the real (i.e. clear image before adding noise).

REFERENCES

[1] Masanao Koeda, Takayuki Nakamura & Etsuko Ueda, (2014) "Introduction to image Processing with OpenCV, 2nd ed", Kodansya, (Japanese).

[2] Y. L. Lee, H. C. Kim & H. W. Park, (1998) "Blocking effect reduction of JPEG images by signal adaptive filtering", IEEE TRANSACTIONS ON IMAGE PROCESSING, Vol.7, No.2, pp.229-234.

[3] Kenichi Yoneji, Masayuki Tanaka & Masatoshi Okutomi, (2005) "PSF Parameter Estimation for Restoration of Linear Motion Blurred Image", Computer Vision and Image Media (CVIM), vol.2005, no.38, pp47-52.

[4] Pascal Vinent, Hugo Larochelle, Yoshua Bengio & Pierre-Antoine Manzagol, (2008) "Extracting and composing robust features with denoising autoencoders", Proceedings of the 25th international conference on Machine learning, ACM, pp1096-1103.

[5] Olaf Ronneberger, Philipp Fischer & Thomas Brox, (2015) "U-net: Convolutional Networks for Biomedical Image Segmentation", International Conference on Medical image computing and computer-assisted intervention (MICCAI), Springer, Vol. 9351, pp234-241.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou & Alexei A. Efros, (2017) "Image-to-Image with Conditional Adversarial Networks", 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp5967-5976.

[7] chainer.functions.convolution_2d, URL: http://docs.chainer.org/en/stable/reference/generated/chainer.functions.convolution_2d.html#chainer.functions.convolution_2d, (access data: 23/01/2019)

[8] Alec Radford, Luke Metz & Soumith Chintala, (2015) “Unsupervised representation learning with deep convolutional generative adversarial networks”, arXiv preprint arXiv:1511.06434.

[9] Sergey Ioffe & Christian Szegedy, (2015) “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, arXiv preprint arXiv:1502.03167.

[10] ADE20K dataset, URL: http://groups.csail.mit.edu/vision/datasets/ADE20K/, (access date: 17/12/2018).

[11] Seiya Tokui, Kenta Oono, Shohei Hido & Justin Clayton, (2015) “Chainer: a next-generation open source framework for deep learning”, Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS), Vol.5, pp1-6.

[12] Diederik P. Kingma & Jimmy Lei Ba, (2014), “Adam: A Method for Stochastic Optimization”, arXiv preprint arXiv:1412.6980.

[13] Mao Xiao-Jiao, Chunhua Shen & Yu-Bin Yang, (2016), “Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections”, Advances in neural information processing systems, pp2802-2810.