Project 1 - Noise2Noise Model for Image Denoising Without Clear Images

EE-559 Deep Learning Course

Naravich Chutisilp (341752), Veniamin Veselovsky (337188)

May 2022

1 Introduction

Noise2Noise [1] is able to reconstruct a signal from learning exclusively from the corrupt signals. In this case, we use this idea to denoise blurry images without training the model with clear images. With the U-Net model that we implemented according to the paper, we have achieved peak signal to noise ratio (PSNR) of **25.55 db** on the validation set.

2 Networks

In general, the network is an auto encoder-decoder. All networks receive the images as an input, and return the images of the same size as the output. The main idea is the to use multiple convolutional layers to encode the input images to a smaller size matrices. Then, up-sample the encoded matrices back to the output images.

The input images have the values in range of $[0, 255]$. The network model also outputs the denoised images which have the values in range of $[0, 255]$. However, during the training, we normalize the input value to be between $[0, 1]$. This allows the weights to be small. As result, the memory usage during both training and evaluation is reduced. The prediction function would then scale, and clip the values of the output images to be between $[0, 255]$ again.

2.1 U-Net

U-Net is implemented exactly according to [1]. This network has a skipping layer, which allows the decoding part of the network to have additional information about the encoding. All the convolutional layers use the padding *same* configuration. That is, the output of the layer has the same size as the input. The activation function after each convolutional layer is Leaky Relu. In the encoding part, the Max Pooling 2D layer is used to reduce the size of the intermediate value. In the decoding part, the Nearest Neighbor Upsampling 2D is used to up-sample the size of the intermediate value. At each up-sampling step, we concatenate the encoded value from the encoding part. As mentioned before, this is the skipping step allowing the decoding part to obtain additional information. Lastly, the convolutional layer is applied without activation function. The network can be summarized in Figure 1.

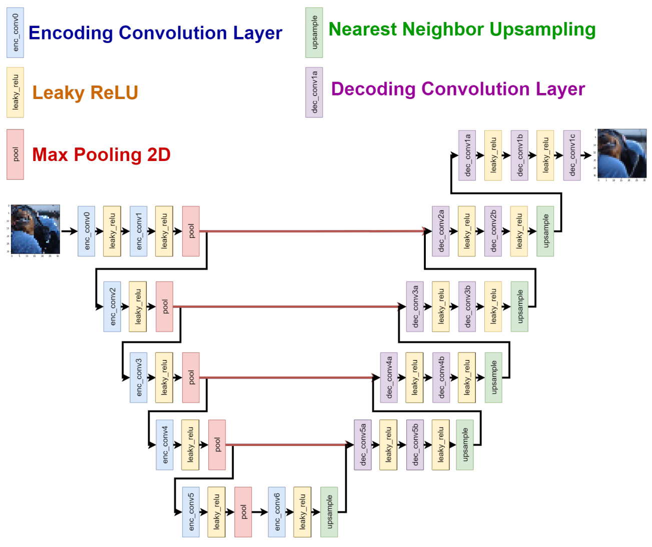

Figure 1: U-Net Network: Red arrow is the skipping step where the output is being concatenated with the intermediate output from the encoding part

We implement the code according to the paper. In fact, the code repository of the paper implements this network in Tensorflow-Keras. However, we reimplement this network using Pytorch framework.

2.2 Simple Sequential

Four other architectures are implemented in a simpler sequential model. The variations are their up-sampling layers (upsampling3 and upsampling4) and the last activation layer (activation). The model can be summarized as the Table 1.

We implement four variations of this sequential model by using up-sampling layers to be either Nearest Neighbour Upsampling 2D (scale factor 2) + Convolutional Layer (with padding same and kernel size 3x3, stride 1), or Convolutional Transpose (2x2 kernel, stride 2). The activation function is either Relu, or Sigmoid. Therefore, there are 4 variations from these architecture choices. They are 1) *NN + Relu*, 2) *NN + Sigmoid*, 3) *Convtrans + Relu* and 4) *Convtrans + Sigmoid* according to the up-sampling layers and the layer activation layer. Note that although we use abbreviation *NN*, we refer to Nearest Neighbour + Convolutional Layer.

Sequential	Nout
conv1 (2x2 kernel, stride 2)	48
relu1	48
conv2 (2x2 kernel, stride 2)	96
relu2	96
upsampling3	48
relu3	48
upsampling4	3
activation	3

Table 1: The summary of the general sequential model. The variation happens in the upsampling3, upsampling4, and activation

3 Training

We use mean square error loss according to the Noise2Noise paper. However, for the U-Net, Adam optimizer with the learning rate 0.001, betas of (0.9, 0.99) is used. While for all the sequential models, Stochastic Gradient Descent (SGD) optimizer with learning rate of 0.1 and momentum of 0.9. All the models are trained for 20 epochs.

All these hyper-parameters are tested that they will work on the training data, by first training the model with these parameters on only one batch size for 5 epochs. The parameters work if the model can overfit the this one batch of training data.

We keep track of the training loss, and validation loss for each epoch. Furthermore, we also calculate the PSNR at the end of each epoch.

4 Result

The PSNR of the validation data set is the metric we use to select our best model. The training loss and validation loss over epoch show that all the model saturate very fast. That is, the loss does not decrease so much after 8th epoch for the training loss (see Figure 3), and 5th epoch for the validation loss (see Figure 4). However, the highest PSNR is obtained at the 11th epoch in U-Net, which is 25.55 db. The best sequential model is Convtrans + Relu which has the highest PNSR of 24.05 db. All the best PSNRs for each model are summarized in the Table 2.

	Best PSNR (db)
U-Net	**25.55**
NN + Relu	23.79
NN + Sigmoid	23.29
Convtrans + Relu	**24.05**
Convtrans + Sigmoid	23.44

Table 2: The summary of the general sequential model. The variation happens in the upsampling3, upsampling4, and activation

In training process, Figure 2 shows that U-Net beats other sequential models since the first epoch. This is unsurprising, because U-Net architecture is deeper and larger. Thus, the model could learn more parameters, which leads to better result. The model also has the skipping layer, which allows the decoding part to use the additional information.

Comparing the four sequential models, activation layer using Relu yields better PSNRs. This complies with the result of U-Net. In U-Net, the last output layer does not have the activation function after. This has the same effect as applying Relu because in prediction we would clip the output value to 0 and 255. That is, we clip all the negative values to 0.

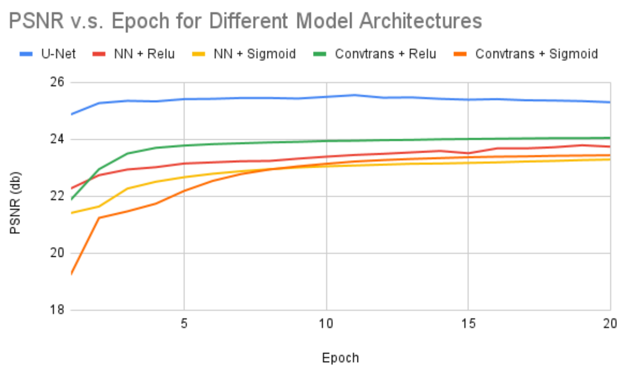

Figure 2: Peak Signal to Noise ratio v.s. Epoch for Different Model Architecture: U-Net Model is more complex, and yield higher PSNR since the very first epochs. The PSNRs of all models start to saturate at around 10th epoch

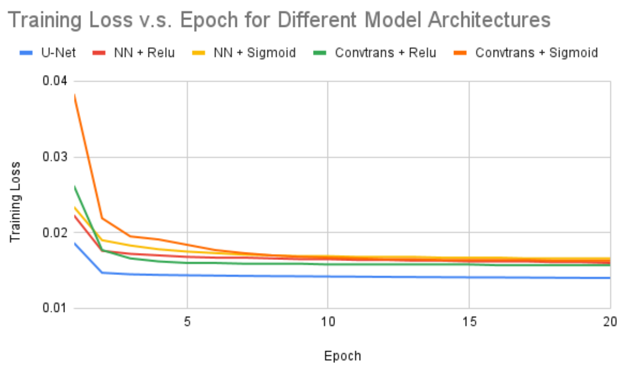

Figure 3: Training Loss v.s. Epoch for Different Model Architecture: This result complies with the Figure 2. U-Net yields better training loss. All the models start to saturate at 8th epoch. Even though this figure may not show obviously due to decimal places, U-Net actually saturate at 10th epoch

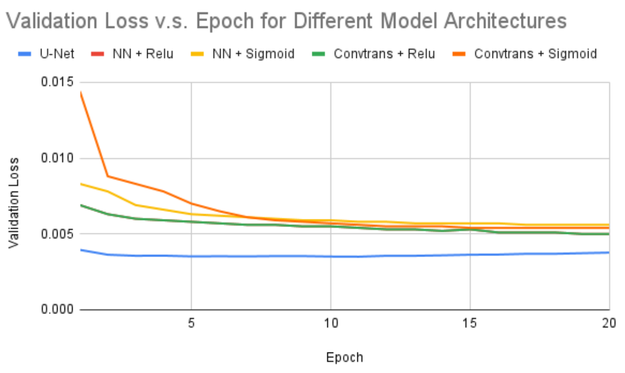

Figure 4: Validation Loss v.s. Epoch for Different Model Architecture: This result complies with the Figure 2 and Figure 3. U-Net yields better validation loss. All the models start to saturate at 5th epoch

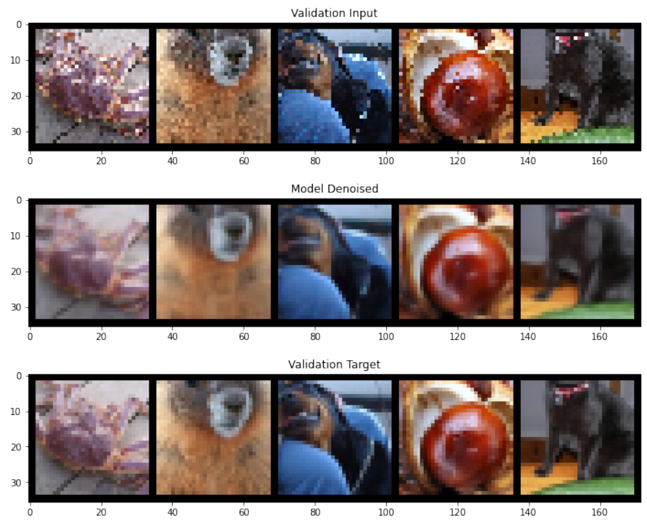

Figure 5: Qualitative Result of U-Net: The first row are the images from the validation input. The second row are the images that are denoised by the U-Net. The third row are the images from the validation target. Even though the images are denoised, U-Net cannot fully bring the high frequency signal of the images back

Qualitatively, U-Net have denoised the output to some degree. However, the model cannot bring the high frequency part of the image back. In other words, the denoised images tend to be blurry compared to the target images. This can be seen in the Figure 5. Nevertheless, compared to the best sequential model, Convtrans + Relu, the denoised images from this model are more pixelated (see Figure 6).

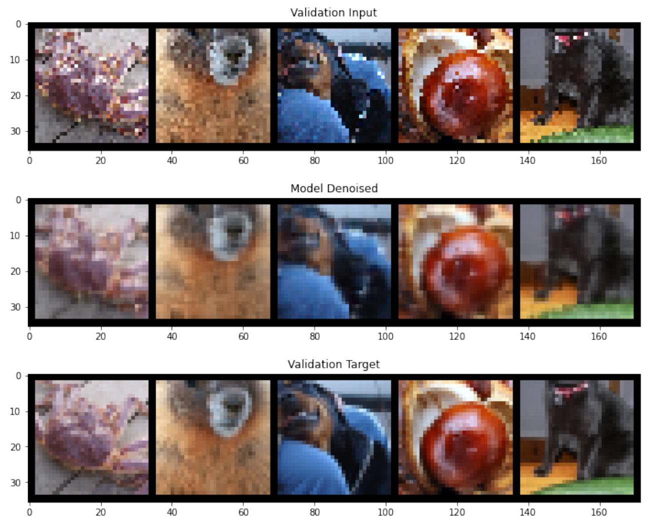

Figure 6: Qualitative Result of Convtrans + Relu: The first row are the images from the validation input. The second row are the images that are denoised by the Convtrans + Relu sequential model. The third row are the images from the validation target. Compared for Figure 5, this model's denoised results are more pixelated

5 Conclusion

We have implemented five models, and train them to denoise images without showing the clear images using the technique from [1]. U-Net, which is the architecture suggested by [1], yields the best result qualitatively and qualitatively. This may be because the U-Net is larger and deeper than other sequential models that we have experimented, which leads to more learnable parameters. U-Net also have skipping layer which allows more information to be processed in the decoding process.

References

[1] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *CoRR*, vol. abs/1803.04189, 2018. [Online]. Available: http://arxiv.org/abs/1803.04189