Mini-Project 1

Jozef Coldenhoff, Bonan Sun

I. Introduction

In this project, we build a network to denoise images without clean samples, following the idea of the so-called *noise2noise* proposed in [1]. Consider that we are given multiple noisy measurements of a single image, such as can happen in astro-photography, we want to enhance the image by removing the noise from the measurement. Traditionally, what could be done is to train a neural network to transform this noisy image into a clean target. However, in many cases, such clean targets do not exist, as is the case in astro-photography. [1] proposes that we should still be able to train a de-noising network by leveraging a key insight. In their paper they explain that when we have pairs of the same image corrupted with different zero-mean noise, we can train the network to transform one such image into another. When conducting this training, the network learns to predict the image that is most likely given the noisy input image. This prediction of the most likely image has the effect that the zero-mean noise is essentially averaged out by the network.

II. Methods

A. Noise2Noise

Why we can train a denoising network with corrupted data can be explained by the following derivations. Consider two independent random noises ε and δ that are unbiased, i.e., $\mathbb{E}[\varepsilon] = \mathbb{E}[\delta] = 0$. Denote clean image by X and the model by ϕ, which is in a hypothesis space $\mathcal{H}$. Then we have

$$\begin{aligned}&\mathbb{E}[\|\phi(X+\epsilon) - (X+\delta)\|^2] \\ =&\mathbb{E}\left[\|\phi(X+\epsilon) - X\|^2\right] - 2\mathbb{E}[\langle\delta, \phi(X+\epsilon; -X\rangle] + \mathbb{E}\left[\|\delta\|^2\right] \\ =&\mathbb{E}\left[\|\phi(X+\epsilon) - X\|^2\right] + \mathbb{E}\left[\|\delta\|^2\right].\end{aligned}$$

Thus

$$\begin{aligned}&\underset{\phi\in\mathcal{H}}{\text{argmin}}\mathbb{E}\left[\|\phi(X+\epsilon) - (X+\delta)\|^2\right] \\ =&\underset{\phi\in\mathcal{H}}{\text{argmin}}\mathbb{E}\left[\|\phi(X+\epsilon) - X\|^2\right]\end{aligned}$$

meaning that minimizing the mean square loss between two noisy images with unbiased noise is equivalent to minimizing the mean square loss between noisy and clean ones. This result can be easily generalized to the case with zero median noise, in which case we only need to replace the mean square loss with the L_1 loss. In this project, we only consider the mean square loss since the images we were given to are corrupted by zero mean noise rather than zero median noise.

B. U-net auto-encoder

The original noise2noise paper uses a U-net [2] to remove noise from the images. This U-net architecture is not unlike an auto-encoder, in that it has a symmetric contracting and expanding path which forces the network to learn a lower dimensional representation, and hopefully, in the process throw away noise, as it is not useful for the network to generate the other noisy image. However, in U-net the authors make use of one more important ingredient which are skip connections. These skip connections, take representations learnt in the contracting path of the network, and pass them to the corresponding layer in the expanding path, which is visualized in Fig. 1 as the grey arrows.

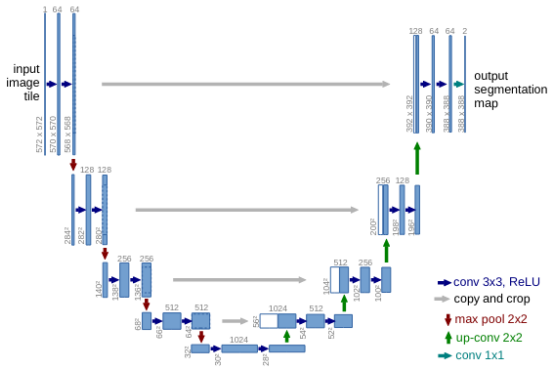

Fig. 1. The network architecture of the original U-net implementation [2]

However, unlike the original noise2noise paper, our pairs of noisy images are significantly smaller at 32x32 pixels, which allows us to decrease the amount of layers in the network.

III. Experiments

A. Find the best network architecture

We choose to use a U-net-like architecture, which is an auto encoder with *addition* or *concatenation* skip connections. We train these two networks with depth from 2 to 10, kernel size from 2 to 6 and out put channel size of the first layer (short by up_channel) from 4 to 10, which are $2 \times 5 \times 5 \times 7 = 350$ different configurations. We train each of the models for 10 epochs. We fix the batch size, the learning rate and the activation functions as 128, 10^{-3} and ReLU to use an Adam optimizer with betas (0.9, 0.999) to train at this stage. After finding the best model architecture, we will tune these parameters.

1) Influence of Kernel Size: For most of the cases, the best kernel size is 2, either in the case of addition or concatenation skip connections. We also plot the PSNR with respect to the kernel size for one pair of configuration of up_channel and depth in Figure 2. The fact that smaller kernels perform better is likely because our images are so small that if we use a large kernel, we will not be able to detect sufficient features.

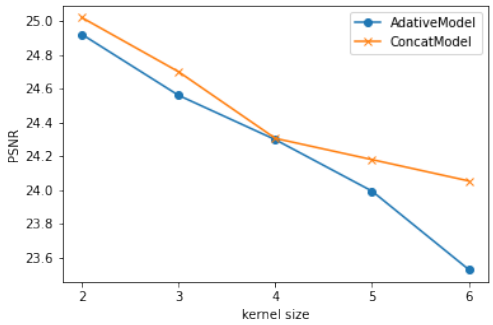

Fig. 2. Validation PSNR v.s. epochs for different kernel sizes of addition and concatenation models with 8 layers and up_channel = 10

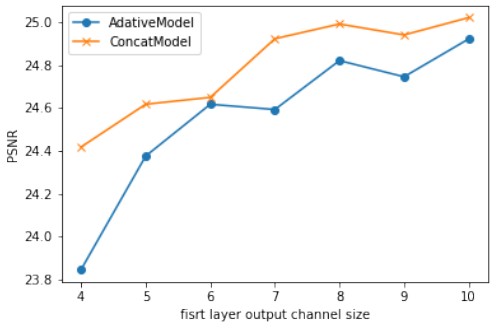

Fig. 4. Validation PSNR v.s. epochs for different kernel sizes of addition and concatenation models with 8 layers and kernel size 2

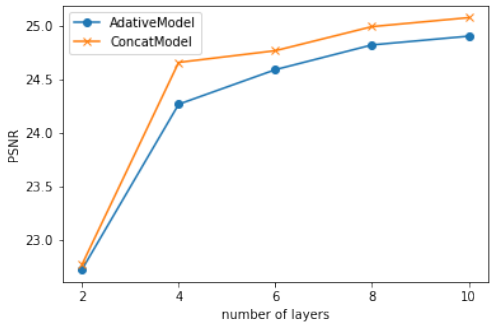

Fig. 3. Validation PSNR v.s. epochs for different layers of addition and concatenation models with kernel size 2 and up_channel = 8

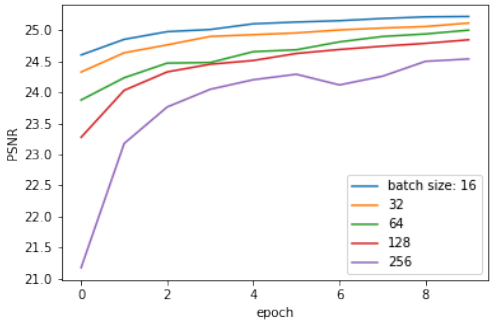

Fig. 5. PSNR v.s. epochs for the best model with different batch size

2) Influence of Depth: We plot the PSNR with respect to the number of layers in the decoder (i.e., doubled in the whole network) in Figure 3 and we can see that the more layers the network has, the better performance it gives. This is as expected since the benefits of depth have been realized especially in recent years. The more parameters we have, the more complicated function we can fit, while the computational effort and execution time also grow exponentially. Therefore, we choose a 10 layer network, which can be trained in reasonable time and gives a sufficiently good result at the same time.

3) Influence of up_channel: As we can see in Figure 4, the PSNR increases as the size of the first layer output channels increases. This is firstly because of the same reason as the influence of the depth: more parameters implies bigger hypothesis space. Secondly, more channels in the first layer can extract more features to be used in the downstream network. However, in our experiments which are not shown here, the performance may decease when the up_channel is too large. Therefore, we choose a moderate size of 10.

4) Summary: In summary, the network structure we choose has 10 layers, uses a kernel size of 2, and 10 output channels in the first layer.

An issue we found too late however, was that in the experiments above, we computed the PSNR on the unclipped values of the model output, this could have led to a slightly wrong PSNR (though limited as ReLU bounds negative numbers). However, since the experiments took around 25 hours to run, and because we are at this point only interested in the best model, and not necessarily an exact estimate of the PSNR, we still decided to use the conclusions of the experiment to select the model.

B. Tuning the best model

After choosing the best architecture, we need to fine tune the training procedure of the model.

1) Batch Size: We test different batch sizes in the training process, as shown in Figure 5. We can see that a smaller batch size works better, which is reasonable since smaller batch size means more steps within one epoch, thus the optimizer can make corrections more frequently leading to a faster convergence rate.

2) Learning Rate: We also test different learning rates, which should be of proper size, otherwise the optimizer will converge very slowly or even diverge. It is not known a priori what the learning rate should be for a complicated model like neural networks. Therefore, we start with some empirical choices of learning rate and compare them, as shown in Figure 9 in the appendix. A learning rate of 0.01 seems to bounce around a local minimum and do not converge, while a learning

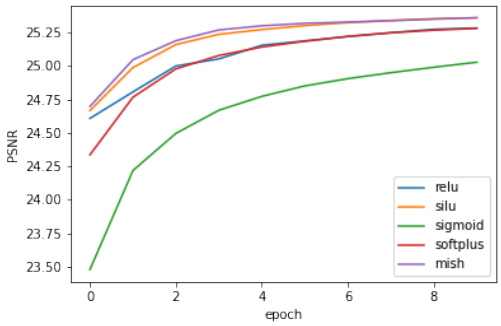

Fig. 6. PSNR v.s. epochs for the best model with different activation functions

rate of 10^{-4} converges slowly. Therefore, we choose 10^{-3}, which makes the convergence fast and stable.

3) Activation Functions: The network uses activation functions after each convolution and transpose convolution. We test different activation functions in order to obtain the best possible model. Figure 6 shows the result on the validation set. We see that the mish [3] and silu [4] activation functions perform the best. We choose mish for the final model as it is slightly higher in the early stages of training.

C. Data Augmentation

We also test various data augmentation strategies for the training data. In our experiments we consider cropping the image to 24x24, and padding it to 32x32, randomly flipping the image in the horizontal and vertical directions and converting the image to greyscale. As we can see in Fig. 7, using no augmentation seems to perform equal or better than all other augmentation strategies. We thus use no augmentation to train the final model.

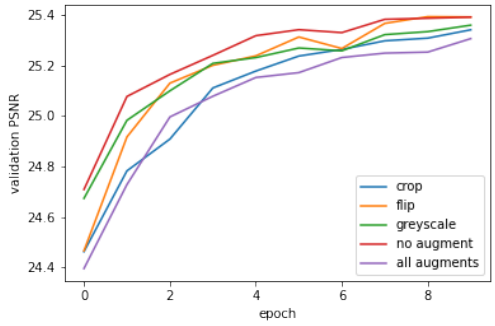

Fig. 7. Plot showing validation PSNR curves of different data augmentation strategies. Trained on the best found model using Adam optimizer with learning rate of 10^{-3}, batch size 16.

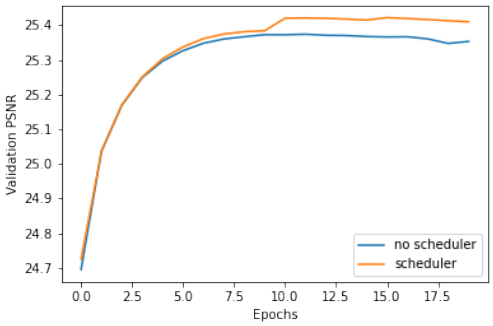

Fig. 8. The final model trained

IV. Conclusion

Name	C_{out}	Function
Input	3	
Conv1	10	Convolution 2×2
Conv2	20	Convolution 2×2
Conv3	40	Convolution 2×2
Conv4	80	Convolution 2×2
Conv5	160	Convolution 2×2
Concat	320	Concatenate output of Conv5
TransConv5	80	Transposed Convolution 2×2
Concat	160	Concatenate output of Conv4
TransConv4	40	Transposed Convolution 2×2
Concat	80	Concatenate output of Conv3
TransConv3	20	Transposed Convolution 2×2
Concat	40	Concatenate output of Conv2
TransConv2	10	Transposed Convolution 2×2
Concat	20	Concatenate output of Conv1
TransConv1	3	Transposed Convolution 2×2
Sigmoid	3	

TABLE I

THE BEST NETWORK ARCHITECTURE WE FIND. C_{out} DENOTES THE OUTPUT CHANNEL OF EACH LAYER. EVERY CONVOLUTION AND TRANSPOSED CONVOLUTION WAS FOLLOWED BY A MISH ACTIVATION, EXCEPT FOR THE LAST LAYER.

Our best model, shown in table I is trained for 20 epochs on the full training data without data augmentation. It uses 10 layers, concatenating skip connections, the mish activation function, Adam optimizer, learning rate of 1e-3, and a batch size of 16. Figure 8 shows the training curve of the final model. We did observe that without using a scheduler, the PSNR did not increase, or even decrease. We thus add a simple scheduler that halves the learning rate at the 10th and 15th epoch. We obtain a final PSNR of 25.41 on the validation set using the scheduler.

A final thing that should be noted however, is that since we selected the model by the validation PSNR, we may have overfit on the validation data. Ideally, we would have used a holdout set to evaluate the final PSNR of our best selected model. However given the small size of the validation set, and the fact that we only need to select the best model, and not necessarily present a unbiased estimate of the model performance, we believe our method should still be adequate.

REFERENCES

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

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[3] D. Misra, "Mish: A self regularized non-monotonic activation function," 2019. [Online]. Available: https://arxiv.org/abs/1908.08681

[4] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017. [Online]. Available: https://arxiv.org/abs/1710.05941

APPENDIX

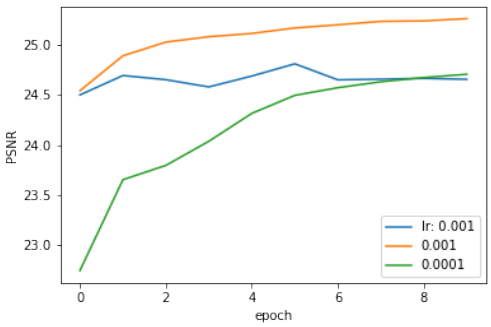

Fig. 9. PSNR v.s. epochs for the best model with different learning rate