Deep Learning - EE-559

Report of Project 1

Noise2Noise implementation

Louise Coppieters
Suhan Narayana Shetty
Centamori Frank

May 27, 2022

1 Introduction

This report describes our implementation of a model to denoise images by implementing a Noise2Noise model using the tools from PyTorch. The Noise2Noise architecture relies on the following mathematical equation:

$$\text{argmin}_\theta \mathbb{E}\Big[||\phi(X+\epsilon;\theta)-(X+\delta)||^2\Big] = \text{argmin}_\theta \mathbb{E}\Big|\phi(X+\epsilon;\theta)-X||^2\Big] \quad (1)$$

This equation describes mathematically that seeking the parameters that learns the best a noisy image from another noisy version of the same image gives the same results if the purpose was to find the clean version of the image. The report is organised as follows, in Section 2 the architecture of the considered model is described, in Section 3 the performance of the selected models over various training and model hyper-parameters is analysed, and finally, in Section 4 the results with the chosen Noise2Noise model and training hyper-parameters is discussed.

2 Architectures

A widely used architecture for the Noise2Noise model is the CNN with a U-Net structure. The U-Net is an encoder-decoder structure where the encoder is composed of convolution layers (Conv2d) and decoder is composed of transposed convolutions (ConvTranspose2d) (see fig.5). One of the main features of this architecture is the skip connections between the layers of encoder and decoder. These skip connections between Conv2d and ConvTraspose2d layers have been emphasised to be very effective in the literature. After several tests, the Conv2d and ConvTranspose2d were chosen to have kernels of size $(3, 3)$, stride$= 1$ and padding$= 1$.

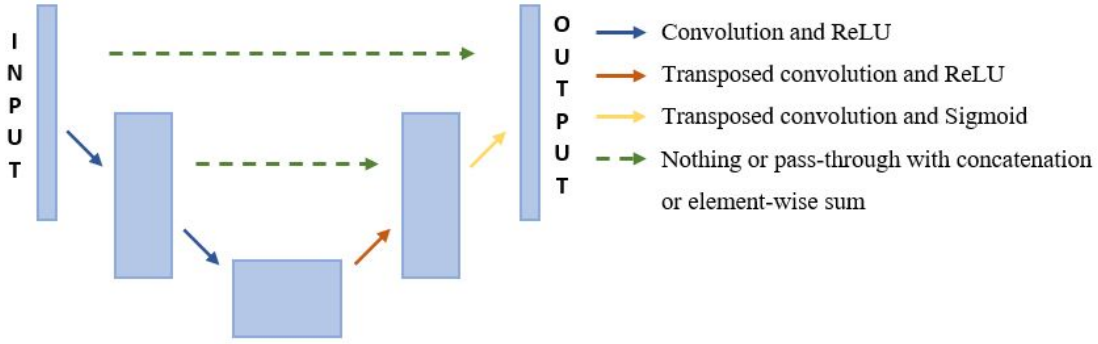

Figure 1: U-Net Architecture

The number of layers in the denoising model remains a decision parameter. We considered four different denoising models with U-Net architecture :

- Net-2 (2 layers: 1 Conv2d and 1 ConvTranspose2d),
- Net-4 (4 layers: 2 Conv2d followed by 2 ConvTranspose2d with skip connections),
- Net-6 (6 layers: 3 Conv2d followed by 3 ConvTranspose2d with skip connections),
- Net-8 (following the same logic)

Through trial-and-error these models appeared to be expressive enough to get a PSNR $>$ 24dB [1].

[1] In the final model, the batch normalization (BatchNorm2d) was not included between the convolution layers as it did not improve the performance presented in the next section.

3 Model and Hyper-parameter Selection

In this section, the choice of the model is explained (among the four selected models mentioned in the previous section) as well as the corresponding hyper-parameters for training. The number of output channels of each layers is fixed and is given by m.[2]

The training and model hyper-parameters include:

- m: Number of output channels of each layer
- bs: The batch size
- lr: Learning rate

The batch size has been considered bs to be among the tuple $(10, 25, 50, 250, 500, 1000)$ and the number of output channels m to be in $(10, 25, 50)$. To reduce the memory requirement, the higher values for these hyper-parameters were not considered. For the learning rate lr, the tests were made for the following values: $(0.1, 0.05, 0.02, 0.01, 0.008, 0.005, 0.001)$. The range of possible values for the hyper-parameters and network sizes was selected after some trial-and-error experiments.

For each of the models, the following procedure to find the best training hyper-parameters was used: first, the model was trained for the different choices of m and bs. The training time was kept short and number of epochs was set to be $1 + \text{floor}(0.25 \times \frac{bs}{10})$ so that we are not biased for any batch size in terms of number of gradient descent steps. Then, in the next step, the best batch size bs and number of channels m is chosen depending on the final PSNR (or validation loss) reached by this configuration. With this fixed bs and m, we find the best learning rate lr [3].

Once obtaining the best training and model hyper-parameters for each of the models, model with best performance was chosen. The Net-6 model (6 layers: 3 Conv2d and 3 ConvTranspose2d) with $bs = 500$, $m = 50$ and $lr = 0.005$ gave the best performance. The best model resulted in a mean PSNR value of 25.55 for the validation set.

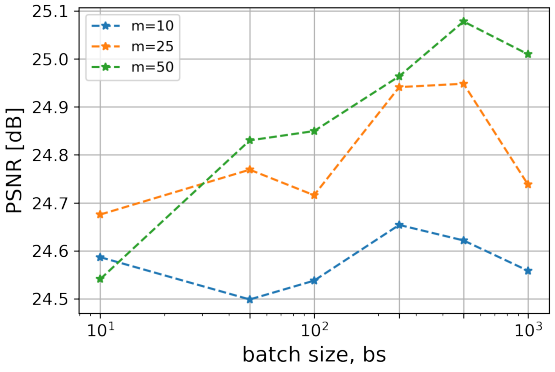

Figure 2: Variation of PSNR over batch size (bs) and number of channels (m) for Net-6. $m = 50$ and $bs = 500$ was found to give the best PSNR value.

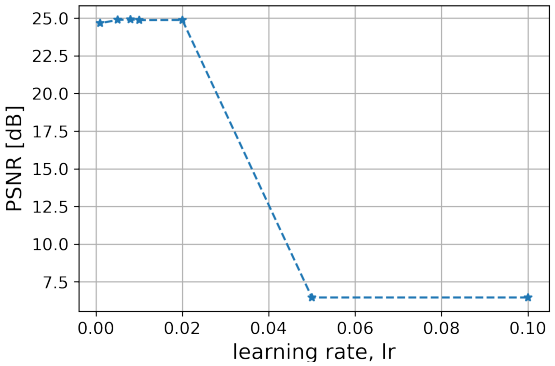

Figure 3: Variation of PSNR over the learning rate for Net-6, for a bs of 500 and a m of 50.

[2] The results with the above shallow architectures were considered already satisfying.

[3] For the choice of the learning rate, the same procedure was followed: for the given model with the chosen m and bs values, train the model for floor$(0.25 \times \frac{bs}{10})$ and choose the lr which results in the best performance

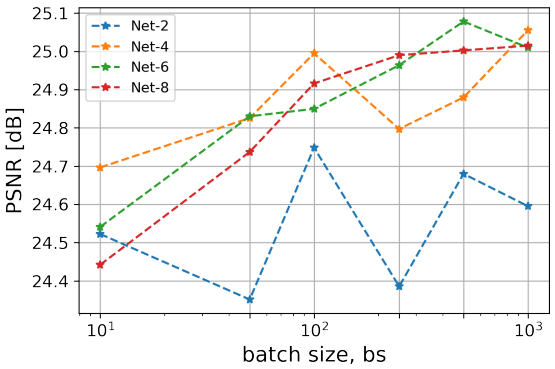

Figure 4: Variation of PSNR for $m = 50$ (number of channels in each layer) over the various networks considered.

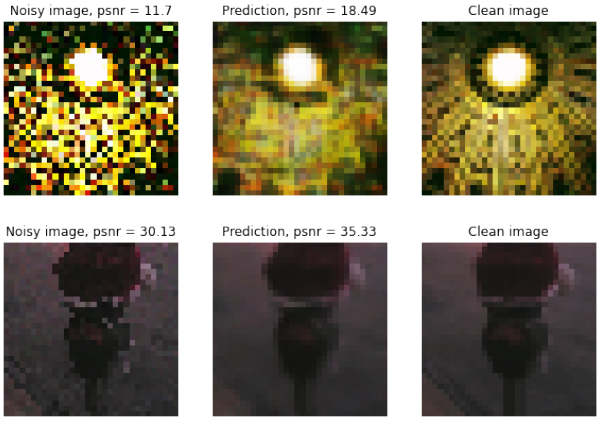

Figure 5: Input (left column), Denoised (center) and Clean Images (right)

4 Results

4.1 Discussion:

From observations and experiments, it is possible to highlight some key points of interest:

- A higher value of the number of channels m also improved the performance, potentially as it allows the model to learn a bigger amount of information on how to denoise. However, increasing it too much will result in excessive memory consumption. So this value was limited for testing.
- Moreover, improving the best resulting model by varying the number of output channels m for each layer could have been possible. As the best model already gave a PSNR $>$ 25 dB, it was not further improved.
- The choice of optimization method played a significant role, with Adam and SGD-with-momentum as potential candidates. Adam optimizer outperformed the SGD in all experiments and the tuning of the learning rate was much easier.
- Optimization of the parameters w.r.t to $L1$ loss was also considered, however, it did not show good performance - both quantitatively (in terms of PSNR) and qualitatively (visual observation of denoised images). This could be because the data of noisy images were likely generated from a Gaussian noise, hence optimizing the model over MSE loss would be preferable.
- Observations of the difference in denoising show the severity of loss on small detailed images, as seen in Figure 5. Such fine details are likely not retrievable from noisy images without more performing models.
- The requirement on the number of epochs given the batch size as discussed in Section 3 has very likely biased the results on the batch size. Fixing the number of epochs for all the batch sizes considered could have been a good choice as well. With enough training time, a smaller batch size (≈ 20) proved to lead to the same level of performance as in batch size of 500.

5 Conclusion

The results obtained show quite well and in details the number of problems related to constructing a Deep Learning model. Indeed, while creating a model that denoised quite well the images, trying to improve it demands to search among a lot of parameters and structures. This proves to be challenging and as a result, the best model of this project is quite close in quality to the model of project 2.