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, we define the architecture of the model we have considered, in Section 3 we analyse the performance of the selected models over various training and model hyper-parameters, and finally, in Section 4 we provide the results with the chosen Noise2Noise model and training hyper-parameters.

2 Architectures

We have chosen CNN with a U-Net structure as the denoising model as this is the widely used architecture for this application. 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 that there can exist 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. So, we included it in our models. The Conv2d and ConvTranspose2d were chosen to have kernels of size $(3,3)$, stride$=1$ and padding$=1$.

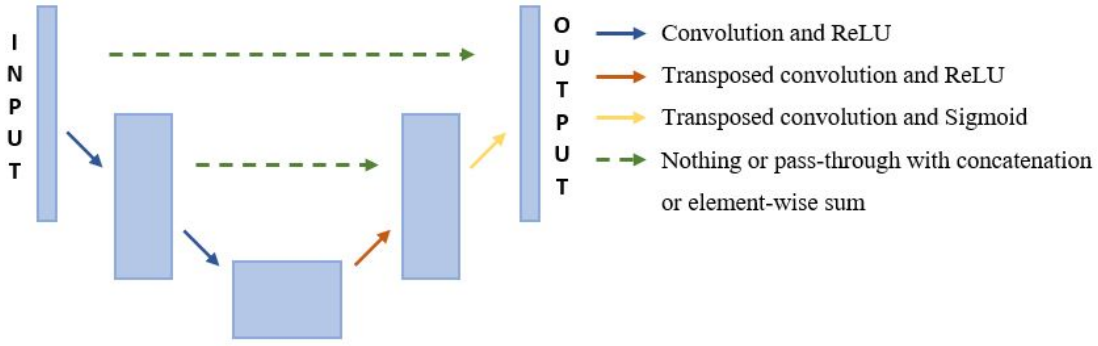

Figure 1: U-Net Architecture

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)

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

[1]We did not include batch normalization (BatchNorm2d) 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, we report on how we chose the model (among the four selected models mentioned in the previous section) and the corresponding hyper-parameters for training. We fix the number of output channels of each layers to be the same and it is denoted 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

We considered batch size 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, we did not consider higher values for these hyper-parameters. For the learning rate lr, we considered $(0.1, 0.05, 0.02, 0.01, 0.008, 0.005, 0.001)$ as possibilities. All these possible values for the hyper-parameters and network sizes were selected after some trial-and-error experiments.

For each of the models, we employed the following procedure to find the best training hyper-parameters: We first train the model 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, we choose the best batch size bs and number of channels m depending on the final PSNR (or validation loss) reached by this configuration. With this selection for bs and m, we next find the best learning rate lr [3].

Once we have the best training and model hyper-parameters for each of the models, we choose the model with best performance. We noted that Net-6 model (6 layers: 3 Conv2d and 3 ConvTranspose2d) with $bs = 500$, $m = 50$ and $lr = 0.005$ gave us the best performance. With the best model, we obtained a mean PSNR value of 25.44 for the validation set.

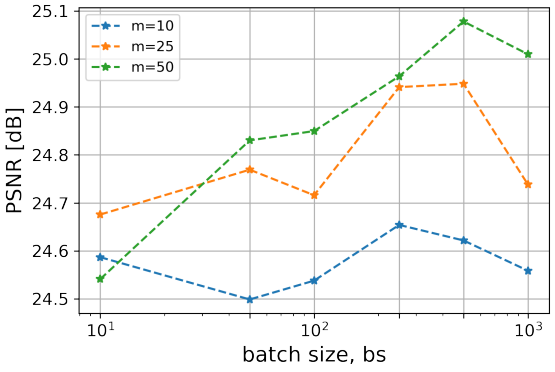

Figure 2: Variation of PSNR over batch size (bs) and number of channels (m) for Net-6. $m = 50$ and $bs = 500$ was found to give the best PSNR value

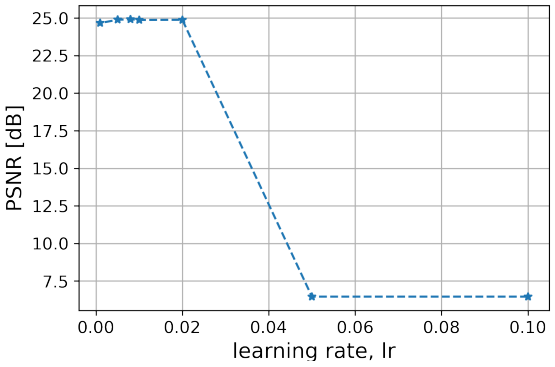

Figure 3: Variation of PSNR over the learning rate

[2] As we found satisfactory results with the above architectures which were not "very" deep (also the size of the input image was relatively small), we did not include max-pooling nor drop-out.

[3] We employ the same procedure: for the given model with the chosen m and bs values, train the model for $\text{floor}(0.25 \times \frac{bs}{10})$ and choose the lr which results in the best performance

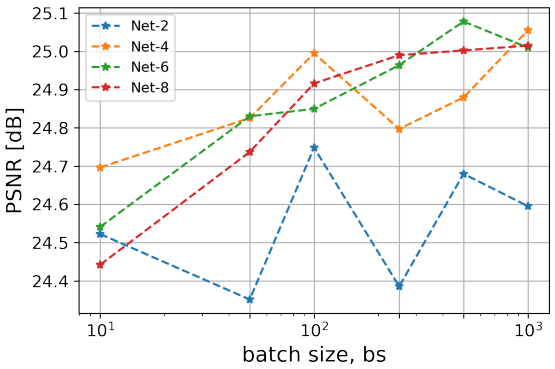

Figure 4: Variation of PSNR for $m = 50$ (number of channels in each layer) over the various networks considered

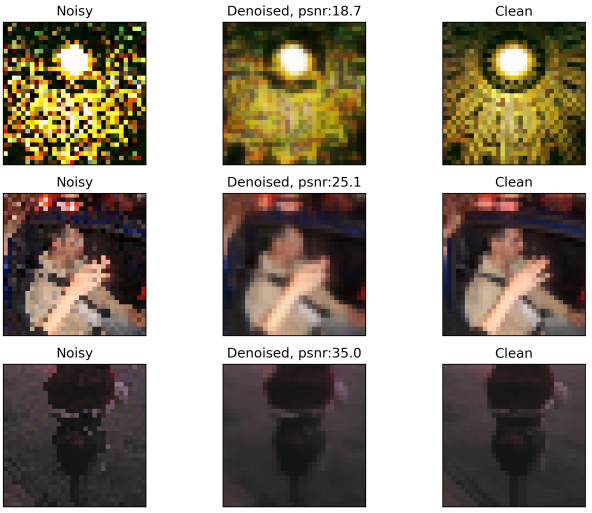

Figure 5: Input (left column), Denoised (center) and Clean Images (right)

4 Results

4.1 Discussion:

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

- In all the considered models, a higher value of batch size bs (>100) gave better results of the PSNR value. This could be because in the Noise2Noise modeling as both the input and output image are noisy versions of an underlying clean image, a higher batch size might be giving a better gradient towards the clean images (i.e., the mean is approximated better with a larger batch size)
- 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 we had to limit its values.
- We could potentially improve the best model we obtained by varying the number of output channels m for each layer. As our best model already gave a PSNR $>$ 25 dB, we did not consider improving it further.
- The choice of optimization method played a significant role. We tried Adam and SGD-with-momentum. Adam optimizer outperformed the SGD in all our experiments and the tuning of the learning rate was much easier.
- We also considered optimizing the parameters w.r.t to $L1$ loss, however,we did not observe 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 generated from a Gaussian noise (hence optimizing the model over MSE loss is preferable)