Deep Learning EE-559 - Noise2Noise

Antoine Olivier Salaün, Henrik Øberg Myhre, Utku Görkem Ertürk

27th of May 2022

***Abstract*—In this project, we predict the clean versions of noisy images. This is done by using a Noise2Noise-technique [1], which means that the training data only consists of pairs of noisy images. The AI-model has hence never seen a clean reference image while training, but it still manages to remove noise from images because the noise in the pairs of noisy images is statistically independent and unbiased. The creation of a network structure and tuning of hyperparameters were important to achieve a good result.**

I. Introduction

When collecting real-world image data, the images may be blurry or contain other kinds of unwanted noise. One might want clean images, but lack the ability to capture images of cleaner quality. This can be solved by using a dataset of paris of images with random noise. A Noise2Noise-model can remove noise after being trained on pairs of images with independent, unbiased noise. In this report, we will present an implementation of a Noise2Noise-model that uses PyTorch to remove noise from images.

II. Noise2Noise

A Noise2Noise-model can create clean images from noisy images after being trained on pairs of noisy images. This works if the noise of the pairs in the training data is independent from each other and unbiased. When training on many independent samples, the resulting model prediction turns out to be the clean versions of the images. In other words, since the noise is independent and unbiased, no model can predict the noise without overfitting, thus the best predictions that the model can perform is the denoised images.

This can be proven mathemtically as follows (where ϵ and δ are unbiased and independent noises):

$$\begin{aligned}
&\mathbb{E}\left[||\phi(X+\epsilon;\theta)-(X+\delta)||^2\right] \\
&= \mathbb{E}\left[||(\phi(X+\epsilon;\theta)-X)-\delta||^2\right] \\
&= \mathbb{E}\left[||\phi(X+\epsilon;\theta)-X||^2\right] \\
&-2\times\mathbb{E}\left[\delta^T(\phi(X+\epsilon;\theta)-X\right]+\mathbb{E}\left[||\delta||^2\right] \\
&= \mathbb{E}\left[||\phi(X+\epsilon;\theta)-X||^2\right]
\end{aligned}$$

The last equation holds because the expected value of δ, and hence the expected value of the L_2-norm of δ, is equal to 0 when the noise is unbiased.

III. Network Structure

Figure 1 shows the final network structure of the Noise2Noise-model. As the figure illustrates, we use two different encoders and four different decoders. The encoders include 2D-convolution, ReLU, batch-norm, max pooling and dropout-layers. The decoders use all of these, except max pooling, in addition to 2D-transposed convolutions to upscale the images back to their original shape. Also we are using skipping connections from encoder layers to decoder layers. The dimension of the input and output images are (channels, height dimension, width dimension) = $(3, 32, 32)$. In the network, they are downsampled to (64, 4, 4) by the encoders before being upsampled to their original shape by the decoders.

IV. Data Augmentation

A very simple trick is performed to increase the amount of training data. As both the input and output of the model are of the same nature (noisy 32×32 images), inputs and outputs can be swapped. Outputs will be used to predicts inputs, thus doubling the number of training samples.

V. Tuning of Hyperparameters

Hyperparameter tuning is done to optimize the Noise2Noise-model learning. In this paper, we present three hyperparameters that was optimized during the creation of the model. The hyperparameters and the values they were tested with are shown in table I. We will test every possible combination to find the values that produce the model with the highest PSNR-score.

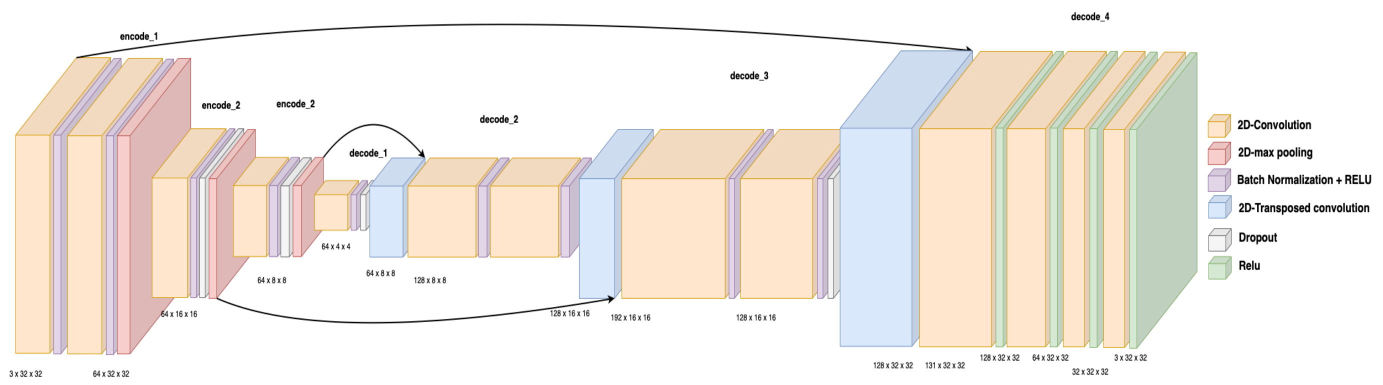

Fig. 1: The network structure for the best model

Batch sizes:	32	64	128
Loss functions:	MSE	L_1	
Learing rates:	1,00E-04	1,00E-03	2,00E-03

TABLE I: Vaues for tuning hyperparameters

VI. Results

The network shown in figure 1 produced the results shown in table II. The training losses and validation losses are not comparable between the L_1-loss and MSE-loss, because the values are calculated differently based on the loss-type. However, the PSNR-scores are comparable, and show that the model generally perform better with MSE-loss than with L_1-loss. Figure 2 shows an example from the best model.

We also see in figure 3 that the performance (PSNR-score) is strongly decreasing when the batch size increases. Additionally, we see in table II that the training loss is most often increasing when the batch size is increasing. The validation loss is a bit more ambiguous, but the lowest losses are often obtained with lower batch sizes. We also see in figure 3 that learning rate 1×10^{-3} performs the best.

Figure 4 shows the training speed for the best model. We see that the PSNR-score changes varies a lot, and it is not proportional to the MSE training loss. in contrast to the performance score, the training loss steadily becomes better thoughout the epochs.

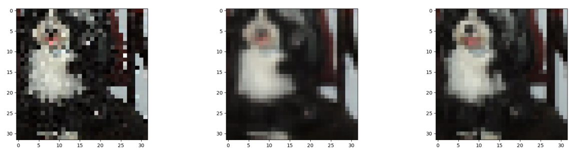
(a) Noised (b) Denoised (c) Clean

Fig. 2: Model demonstration on an example image

Loss type	Learning rate	Batch Size	Performance (dB)	Train loss	validation loss
L_1	1,00E-04	32	23,46	18,14	9,69
L_1	1,00E-04	64	23,23	18,26	10,10
L_1	1,00E-04	128	23,33	18,40	9,93
L_1	1,00E-03	32	23,37	17,98	9,88
L_1	1,00E-03	64	23,26	18,02	10,06
L_1	1,00E-03	128	23,07	18,10	10,79
L_1	2,00E-03	32	23,50	18,08	9,75
L_1	2,00E-03	64	23,46	18,07	9,71
L_1	2,00E-03	128	23,21	18,12	10,34
MSE	1,00E-04	32	23,91	933,16	225,23
MSE	1,00E-04	64	23,85	936,81	227,92
MSE	1,00E-04	128	23,82	941,59	231,76
MSE	1,00E-03	32	**24,15**	924,77	217,67
MSE	1,00E-03	64	24,07	925,65	223,32
MSE	1,00E-03	128	24,03	928,47	220,71
MSE	2,00E-03	32	24,10	928,31	221,06
MSE	2,00E-03	64	24,06	927,23	219,24
MSE	2,00E-03	128	23,94	931,15	227,27

TABLE II: Tuning of hyperparameters

VII. Discussion

A. Network structure

The network structure was settled after a long period of testing. As expected in image analysis, the network heavily relies on convolutions and takes the typical shape of a (pretty heavy) auto-encoder.

Tests demonstrated the importance of the U-net structure. It is believed these layers of encoders and decoders mostly learn how to denoise an image. They may lose the information of the input image through the deep network. Thus it is important to reinject this data in the network.

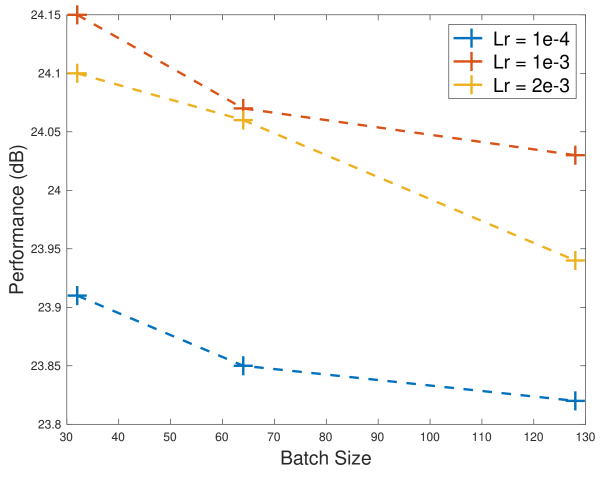

Fig. 3: Perfomance (in dB) in function of Batch Sizes for each learning rate with the MSE loss function

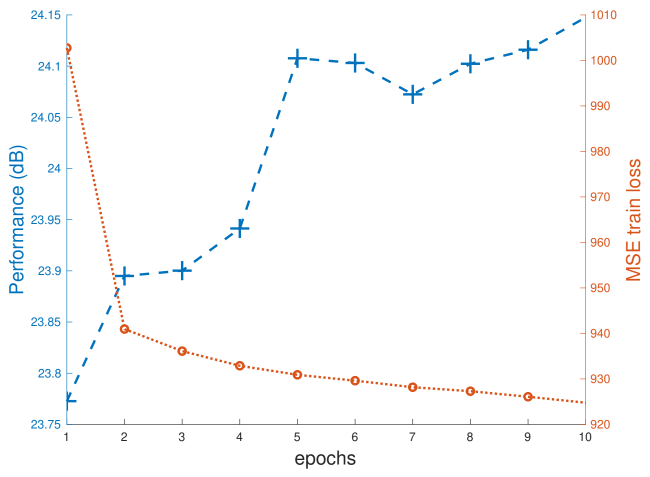

Fig. 4: Validation perfomance (in dB) and MSE train loss for the optimal model

B. Loss function

MSE loss achieves much a better training than a L_1 loss. MSE corresponds to a L_2 measure which seem to highly fit our data. Hence, the input images were noised based on a unbiased normal distribution (i.e. mean = clean image). L_1 would target the median of the noise, while L_2 targets the mean (i.e. the input signal). This works well for this dataset but it could be considered overfitting as it would not perform as well for a different type of noise. It was observed that MSE sometimes results in blurring in the predictions.

C. Batch Size

We notice that the performance improves and the training loss decreases when the batch size decreases. Additionally, the validation loss most often decreases. One potential reason for this is that larger batch sizes leads to more internal competition for weight change within the batch, and in the end a less accurate result than if they were split in smaller batches and done sequentially [2]. See figure 5.

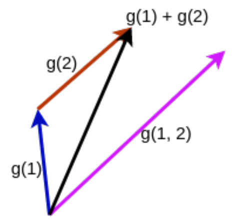

Fig. 5: Splitting the batches them in two batches may lead to a different result at $g(1) + g(2)$ compared to $g(1,2)$

D. Time limit

The selected model is deep and has many parameters to train. However, it is still performant with only ten epochs (24.15 dB). However, figure 4 shows that at the 10th epoch, the model is still learning. The model is limited to ten epochs because of the ten min-training time limit although more epochs would definitly increase the performance. When training with more time, the model, with best parameters, achieved a PSNR-score of 25,75 dB after 25 epochs.

E. Improvements

The model proposed in [1] reaches a higher PSNR performance (31.83 dB). This paper used different images of different dimensions, but it is likely possible for our model to achieve a performance closer to theirs. To do so, the priorities would be to tune hyperparameters more finley, perform data augmentation by croping, rotating and alter the dataset. Additionally, we could tune number of channels more systematically. However, increasing the training time could be the most effective way to improve the performance, for reasons mentioned in section VII-D.

VIII. SUMMARY

In order to denoise a dataset of 32×32 images, a UNet auto-encoder was implemented with the architecture shown in figure 1. Inspired by [1], it trains to reproduce images noised differently and it achieves to clean the images. Hyperparameter tuning, architecture testing and a wise choice of loss and optimiser let one achieve a performance of 25,75 dB, or 24,15 dB with the 10-minute limitation.

References

[1] Jaakko Lehtinen et al. *Noise2Noise: Learning Image Restoration without Clean Data*. 2018. DOI: 10.48550/ARXIV.1803.04189. URL: https://arxiv.org/abs/1803.04189.

[2] Kevin Shen. *Effect of batch size on training dynamics*. 2018. URL: https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e.