École Polytechnique Fédérale de Lausanne

Deep Learning

Professor François Fleuret

MiniProject 1

Students	**Sciper**
Gabriel Bessette	283364
Louis Ferment	288331

May 27, 2022

Table of Contents

1 Mini-Project 1 **1**
1.1 Introduction . . . 1
1.2 Network Structure . . . 1
1.3 Optimizer . . . 1
1.4 Weight Initialisation . . . 2
1.5 Parameter Investigation . . . 2
1.6 Additional Discussion . . . 2
1.6.1 Network . . . 3
1.6.2 Residual and Batch Normalization . . . 3
1.7 Conclusion on the Network and its performance . . . 3

1 Mini-Project 1

1.1 Introduction

The purpose of this mini-project was to implement and optimize an image denoiser using the Pytorch framework and a provided Data Set. The main challenge was to build a network that could denoise any RGB image with the constraint that the network must be trained in a short time.

In order to estimate the performance of the network, the Peak Signal-to-Noise Ratio (PSNR) criterion will be used.
Note : for the optimization process of the network, the network will be train on samples of 10'000 images of the train data set for 10 epochs and the resulting network will be tested on the 1000 images of the test data set. The convergence and the mean PSNR over the test data set will be investigated.

1.2 Network Structure

Initially, a network inspired by the literature [1] was implemented and then tested. This network was a Convolutional Neural Network (CNN) arranged in an Encoder-Decoder structure and therefore was mainly composed of 2-dimensional Convolution and transposed Convolution, Relu, Upsampling and Sigmoid layers. After some tests of the Network, it turned out that the network was performing well and was giving promising PSNR results. However the performance in terms of time was poor and the Network was taking too long to train compared to the requirements. These performances are caused by the complexity of the network i.e., the number of layers and the number of parameters for each layers.
Therefore, new smaller networks based on the structure of the previous tested network were created. The idea behind these networks was to keep the Encoder-Decoder structure presented in [2] and minimize the number of layers to speed up computation time while keeping good PSNR results. Hence, a very simple but promising network was developed. This network has the structure presented in the figure 1.

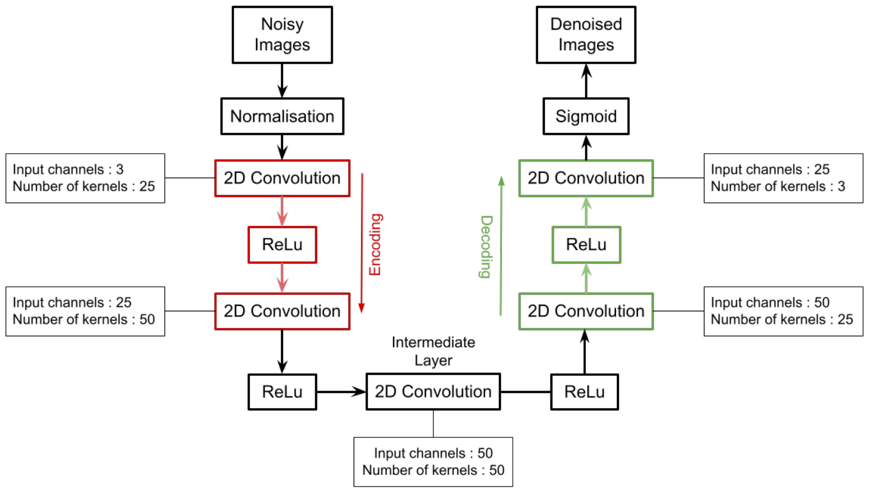

Figure 1: *Simple Denoiser Network*

Starting from this network, some parameters and criterions of the network can be modified and adjusted in order to optimize the network and to achieve better results.

1.3 Optimizer

Initially, the Stochastic Gradient Descent (SGD) has been used as optimizer to update the parameters of our network (weights and biases). A first significant improvement could be achieve by implementing the Adam optimizer (for the default parameters : $\eta = 0.001$, and $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$ for adam). The results are given in table 1 below.

	SGD	Adam
PSNR $[dB]$	16.24	24.16

Table 1: *Optimizer : Adam vs SGD* $\eta = 0.001$

It can be observed that the Adam optimizer brings an important improvement in term PSNR.

1.4 Weight Initialisation

Then, improvements can be made by initialising the weights and biases differently from the default initialisation (uniform distribution). The Xavier initialisation was investigated. The results given in the table below 2 show that with the Xavier Normal initialisation, a better PSNR is reached in 10 epochs.

	Uniform	Xavier Uniform	Xavier Normal
PSNR $[dB]$	24.16	24.24	24.72

Table 2: *Initialisation : Uniform distribution vs Xavier*

1.5 Parameter Investigation

Considering the Adam optimizer and Xavier Initialisation for the network structure discussed in the section 1.2, the parameters of the different convolutions used in the network can be investigated to improve the performances. Especially, the influence of the number of channels for each layer is important. Furthermore, the choice of the batch size B, and the learning rate η can be investigated. Therefore, the following tables show the influence of each parameters by imposing a reference configuration ($C_1/C_2 = 25/50$, $\eta = 0.001$,$B = 5$) and then modifying one parameter at a time. The table 3 shows some combinations of the number of channels C_1/C_2 while tables 4 and 5 show the influence of the learning rate and the batch size.

C_1/C_2	16/32	25/50	32/64	50/100	64/128
PSNR $[dB]$	24.33	24.72	24.71	24.50	24.68

Table 3: *Number of Channels* ($\eta = 0.001$, $B = 5$)

η	0.0005	0.001	0.005	0.01
PSNR $[dB]$	24.60	24.72	23.92	22.83

Table 4: *Learning rate* ($C_1/C_2 = 25/50$, $B = 5$)

B	2	5	10
PSNR $[dB]$	24.66	24.72	24.48

Table 5: *Batch size* ($C_1/C_2 = 25/50$, $\eta = 0.001$)

These results suggest that the configuration ($C_1/C_2 = 25/50$, $B = 5$, $\eta = 0.001$) is more efficient i.e., for 10 epochs and on a sample of the train data set, the resulting PSNR is higher. Concerning the number of channels, as the number of channels increase to larger number (64/128), the PSNR increases. However, it requires larger number of parameters and increases the computation time and therefore the combination 25/50 is kept. Also, for a batch size of 2 images, the PSNR is close, but a larger batch size is preferred to speed up the training.

1.6 Additional Discussion

In the previous section, multiple convincing ways to optimize the network were presented. However, other solutions were explored to increase the performance of the network but were either not successful or needed excessive computation time. These solutions are discussed in this sections.

1.6.1 Network

The network implemented for this project was very simple and different options were explored to increase its complexity and achieve better results. First, a higher number of layers was tested to increase the number of parameters. But in general, adding layers to the network decreases the performances and the PSNR. For instance, a similar network with four layers for the encoder and the decoder and one additional layer between the encoder and decoder and a maximum number of channel of 256 gave at most a PSNR of 24.26 after some optimizations.

1.6.2 Residual and Batch Normalization

Also, batch normalization and residual layers were implemented and tested in the network. It turned out that the batch normalization did not work well with the network. A reason could be that with the low number of layers and parameters of this network, batch normalization ”drowns” the information of the initial images by normalizing the results of the convolution layers. Similarly, the residual layer tends to restore the information of the initial noisy images and therefore to bring back the noise. This problem could eventually be avoided with a larger number of parameters and a more complex network, as shown in the literature, but as discussed previously, this costs in term of computation and therefore in term of time.

1.7 Conclusion on the Network and its performance

In this part, it was shown how the denoiser network was developed and the methods used to optimize it. This results in a network depicted in figure 2 and the performance after a training on the 50’000 images of the training set for the configuration ($C_1/C_2 = 25/50$, $B = 5$, $\eta = 0.001$) are summarized in the table 6. Note : the PSNR is computed on the 1000 images of the testing set.

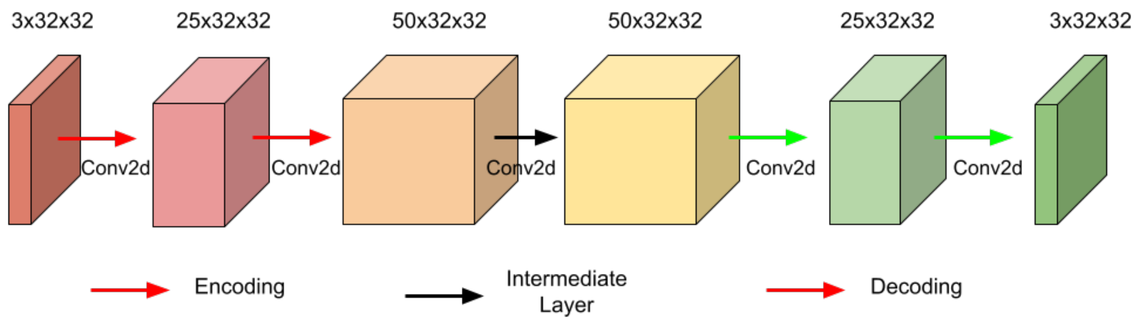

Figure 2: *Denoiser Network*

	Average PSNR $[dB]$	Time per epoch during training
Results	25.02	39 *seconds*

Table 6: *Number of Channels* ($\eta = 0.001$, $B = 5$)

References

[1] *Rina Komatsu and Tad Gonsalves, Comparing U-Net Based Models for Denoising Color Images, 2020.*

[2] *J. Lehtinen et al., Noise2Noise: Learning Image Restoration without Clean Data, 2018.*