Mini-Project 1

Servane Lunven, Anya-Aurore Mauron, Louis Poulain--Auzéau
Deep Learning EE-559, EPFL Lausanne, Switzerland

***Abstract*—In 2018, the Noise2Noise paper [1] showed that an autoencoder was capable of reconstructing an image by only seeing corrupted versions of the image. In this paper, we try to mimic the results but with a time constraint for the training step: 10 minutes on a small GPU and 45 on a CPU. The majority of the work is done on T4 and P100 Nvidia GPU. We evaluate the performance of our network using two different metrics: the Peak-to-Signal Noise Ratio (PSNR) and the Structural Similarity (SSIM).**

1. Introduction

Image denoising is a well studied subject, for instance in the field of MRI or simply in photography. Several implementations use pairs of clean and corrupted images to estimate the optimal parameters of a convolutional neural network. However, it is sometimes very hard to get clean versions of an image. In 2018, Lehtinen et al. ([1]) showed that it was possible to recover clean images only by seeing corrupted versions of the image, even without any prior knowledge on the noise. In this project, we try to replicate the same on a new dataset, and with a time constraint on the training time. We will use the well-known library Pytorch [2] to construct our auto-encoder.

A. Dataset

The training data is composed of 50000 pairs of images, corrupted with an unknown noise. The test set is composed of 1000 pairs of images, one being the ground truth and the other a corrupted version of it.

B. Evaluation of the performance

PSNR is a well known metric, used to measure the efficiency of a reconstruction. Unfortunately, it tends to prefer blurring over a better reconstruction of edges as shown in Figure **??**. This is why a second measure of performance is used for all of our tests: SSIM. It is designed to evaluate the reconstruction from a human perspective. Finally, driven by the experiments of Zhu et al. ([3]), we implemented a loss function based on wavelet coefficients, often used for denoising tasks in the field of image processing.

2. Theoretical Background

A. SSIM

Peak signal-to-noise ratio (PSNR) is a wildly used full-reference image quality and distortion assesment algorithm. However, it does not correlate well with perceived quality. That is why perceptual image quality metrics, like *Structural similarity* (SSIM), were created. It came from the idea that the human visual system is highly adapted for extracting structural information from a scene. More advanced metrics like *multi-scale* structural similarity were implemented. Nevertheless, in this study, we stick to a much simpler approach. To measure the quality of the denoising task with a perceptual image quality metric, the single-scale structural similarity is used [4].

1) Single-scale structural similarity: Let $\boldsymbol{x} = [x_1, ..., x_N]^\top$ and $\boldsymbol{y} = [y_1, ..., y_N]^\top$ be two discrete non-negative signals that have been aligned with each other. Let μ_x, σ_x^2 and σ_{xy} respectively be the mean of $\boldsymbol{x}$, the variance of $\boldsymbol{x}$, and the covariance of $\boldsymbol{x}$ and $\boldsymbol{y}$. The luminance, contrast and structure comparison measures are defined as : $l(\boldsymbol{x},\boldsymbol{y}) = \frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}$, $c(\boldsymbol{x},\boldsymbol{y}) = \frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2}$ and $s(\boldsymbol{x},\boldsymbol{y}) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}$, where $C_1 = (K_1 L)^2$, $C_2 = (K_2 L)^2$ and $C_3 = C_2/2$. The dynamic range is given by L, and the constants K_1 and K_2 are set to $K_1 = 0.01$ and $K_2 = 0.03$. The general form of the structural similarity index between signals $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by $\mathrm{SSIM}(\boldsymbol{x},\boldsymbol{y}) = [l(\boldsymbol{x},\boldsymbol{y})]^\alpha [c(\boldsymbol{x},\boldsymbol{y})]^\beta [s(\boldsymbol{x},\boldsymbol{y})]^\gamma$, where parameters α, β and γ are set to 1. We thus obtain

$$\mathrm{SSIM}(\boldsymbol{x},\boldsymbol{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}.$$

To measure the SSIM index between two discrete signals, a sliding window approach is used. A window (of size 7×7) slides pixel by pixel across the whole image, and, at each step, the SSIM index is calculated within the local window. As mentioned by Wang [5], this method often exhibits undesirable blocking artifacts. That is why a circular-symmetric Gaussian weighting function $\boldsymbol{w} = [w_1, ..., w_N]^\top$, with standard deviation of 1.5 samples and normalized to unit sum ($\sum_{i=1}^N w_i = 1$), is used. Hence the values μ_x, σ_x and σ_{xy} become $\mu_x = \sum_{i=1}^N w_i x_i$, $\sigma_x = \left(\sum_{i=1}^N w_i (x_i - \mu_x)^2\right)^{1/2}$ and $\sigma_{xy} = \sum_{i=1}^N w_i (x_i - \mu_x)(y_i - \mu_y)$. In the end, the mean of all the indices obtained by sliding the window is computed to get the SSIM of the whole image.

Measuring the SSIM index can be used as a way to assess the quality of a denoising task's result. Moreover, as suggested in [6], it can also be used in the loss function of the optimization of the neural network of Noise2Noise [1]. In our case, we use the single-scale SSIM instead of the suggested multi-scale SSIM, which gives

$$L_{H+SSIM}(\boldsymbol{x},\boldsymbol{y}) = (1-\alpha)\cdot L_H(\boldsymbol{x},\boldsymbol{y}) + \alpha \cdot \mathrm{SSIM}(\boldsymbol{x},\boldsymbol{y}), \quad (1)$$

where $\alpha \in (0,1)$ is a constant and L_H is the Huber Loss.

B. Wavelet Transform

In an image, several features are encoded by different frequency coefficients. For instance, high frequency coefficients contain information about the edges of the image, and low frequency coefficients contain information on the texture, the colors. The wavelet transform consists in recursively separating the low frequency coefficients from the high frequency ones. To achieve this, two filters, one lowpass and one highpass are applied along the columns of the image. The two results obtained are then downsampled by a factor 2 and concatenated, as shown in Figure 1. The same procedure is then applied along the lines. The resulting image is then decomposed into four rectangles of same size. The top left one contains low frequency coefficients (LL), the bottom right one contains high frequency coefficients (HH) and

the other contain the rest (LH and HL). The same procedure can then be applied on the top left rectangle to produce a wavelet transform of higher order. The idea of using a wavelet transform to denoise an image is not a new idea in the field of image processing. Indeed, given a noisy image, the noise is uniformly spread in its wavelet transform, allowing it to be more easily removed.

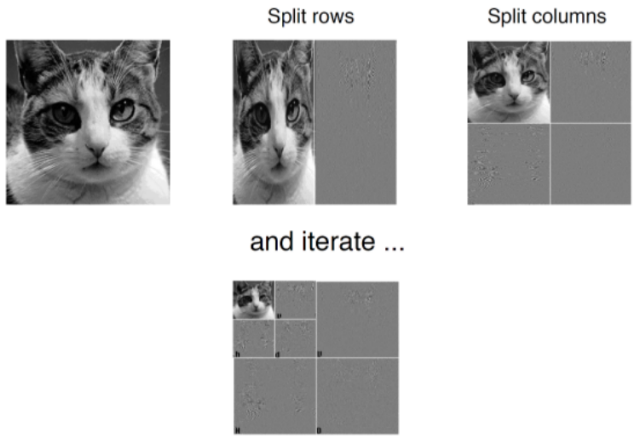

Figure 1: Illustration of the wavelet transform on a cat image. Taken from [7].

3. Experiments

A. Loss function

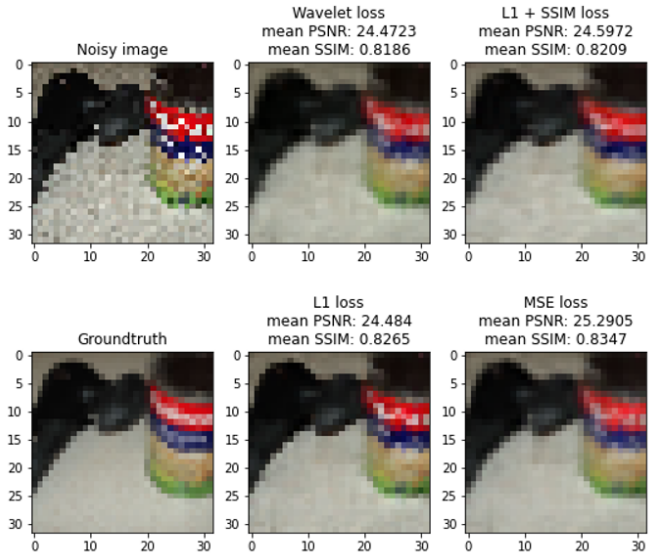

Figure 2: Comparison of several losses. The performance of each loss is indicated as a title.

As in [1], using the mean squared error (MSE) as a loss function proved to output a better PSNR, but a more blurry image. This is shown in Figure **??**. This is why we used in most of our models the L1-Loss. We also implemented a so-called wavelet loss ([3], Wloss). This loss is characterized by its relation with the wavelet transform described in the previous section and can be decomposed into four classical losses (MSE loss for the LL part and L1 loss for the 3 other parts). After the noisy image is passed to the model we compute the output's wavelet coefficients and also the wavelet coefficients of the second noisy image. To help the network to learn in priority the edges of the image, a weight of 100 is applied on the loss comparing the HH coefficients, a weight of 10 for the LH and HL parts and a weight of 10^{-4} is applied on the loss comparing the LL coefficients. The four losses are then summed up to form the wavelet loss and the backward step is performed.

B. Models

Considering the time limit of the training, the first step is to build very small autoencoders with only 3 to 5 layers for both the encoder and the decoder. The two first attempts were destined to build a network with a different architecture than that presented in [1]: no pooling layer and transposed convolutions were used in the decoder (models 1 & 2, mean PSNR of 23.9 and 23.5).

Since the results were not good enough, the architecture was changed to use pooling layers and classical convolutional layers (model 3, mean PSNR of 19.4). Then, to improve the results, we began concatenating the outputs of the encoder inside the decoder, to provide more information to the decoder about the encoding steps (model 4, mean PSNR of 24.3). Following this, the next step was to construct several models to evaluate the performance of little changes. First, we tried to concatenate twice the input to give more weight to the encoding step inside the decoder (models 5 & 6, mean PSNR of 24.6 for model 6). In model 7, we tried using average pooling layers instead of max-pooling ones to retain more information on the pixel values but the score didn't change much.

Following ideas of image processing, we also changed the padding modes (models 8, 9 & 10, respectively 'zeros', 'circular', 'replicate') to try to introduce less boundary artifacts when convolving the images. Surprisingly, the score didn't improve (mean PSNR of around 24 for the three models). Furthermore, we modified the activation function to use SELU instead of leaky-ReLU (model 11, mean PSNR of 24.7). Although we got better results, the computational time was too big when running on a cpu (around 20 minutes for one epoch). Finally, we used the wavelet loss to try to improve the score, but it didn't improved compared to model 6 (model 12, mean PSNR of 24.5). In all of our models, the last activation is a clipping operation to the range $[0,1]$. All the results can be found in Table A.3.

C. Best model

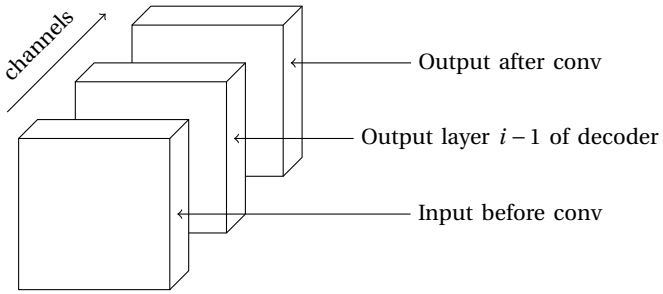

Figure 3: Schematizing of the input in layer i of the decoder for the best model.

As explained in the previous subsection, our best model corresponds to model 6. The encoder is composed of 6 convolutions, each of which, but the last, is followed by a pooling layer. We keep track of the state before and after entering the next convolutional layer. The main idea, schematized in Figure 3, is to reuse these two pieces of information in the decoder. In each of the layers of the decoder, we keep track of

the state of the tensor after the previous decoding step. Then, before entering the next layer of the decoder, we concatenate it between the two pieces of information that have the same shape.

D. SSIM

Loss (1) is used for the simple neural network of model 6. The result is compared with the one obtained with the same model, but with the Huber Loss L_H only. In the experiment, α is fixed to 0.5. We use then another loss defined as:

$$L_{l1+SSIM}(\boldsymbol{x}, \boldsymbol{y}) = (1-\alpha) \cdot L_{l1}(\boldsymbol{x}, \boldsymbol{y}) + \alpha \cdot SSIM(\boldsymbol{x}, \boldsymbol{y}),$$

where $L_{l1}(\boldsymbol{x}, \boldsymbol{y})$ is the $l1$ loss function. In table I, it is noticeable that the PSNR results do not vary much as a function of the loss function used. The SSIM indices do not vary noticeably either, but, as expected, the highest final SSIM values are the one obtained with the losses including them used in the training part.

	Huber	Huber + SSIM	L1	L1 + SSIM
PSNR	24.586	24.577	24.48	24.6
SSIM	0.819	0.823	0.827	0.821

Table I: Comparison of the different results, with respect to various losses which include the SSIM loss or not.

Comparing the denoised images with the groundtruth on figure A.4, the one obtained with the $l1$ loss produces the sharpest image. It seems that the SIMM loss does not add any improvement to the denoising task.

E. Data augmentation

Data augmentation allows having more training data in order to avoid overfitting. We apply the same transformation on the pairs of noisy images during training.

According to Calvarons [6], simple augmentations can be used in order to increase the number of noisy images. These augmentations are horizontal and vertical flips. Also, it is suggested to switch pixels in between the two noisy images, under the assumptions that the images are well aligned, and that the noise is not correlated. In this way, new pair of images are generated and act as noise surrogate of the original pairs. In Krull's paper [8], rotations of 90, 180 and 270 degrees were also used in order to have more training data in a denoising task. We do not use data augmentation like cropping, since the images are already small, nor do we add noise, as we are trying to denoise the images.

Each of these transformations is applied under a probability of 80%. Such a sequence of transformations (vertical flip, horizontal flip and rotation) is applied on 80% of the training set, at each epoch (so a pair of images undergoes a specific transformation with a probability of 64%).

Moreover, we apply a pixel swapping transformation in between a pair of noisy images with a random number of pixels in between 50 and 250, on 50% of the training set. This window of pixels is reasonable, since an image in our data set contains 1024 pixels (so less than a quarter of the image is swapped).

We compare the PSNR and the SSIM indices of four models; without data augmentation, with simple transforms (80% of the training set is transformed), with simple transforms (50%), with pixels swapping (50%). The quality indices are computed in between each pair of the test samples, and a mean is computed. All models are instances of the same model (model 6) and their parameters are equal before the training. We compare visually the efficiency of the 4 models on a test image, see figure A.5. We notice that the quality of the denoised image is not noticeably improved by data augmentation.

Finally, we obtain the following indices in table II. Overall, the SSIM indices as well as PSNR are approximately the same, with or without data augmentation. It means that we already had enough training samples beforehand, and that we were not overfitting. However, data augmentation might be useful with a smaller training data set.

	Without data augmentation	Simple transforms (80%)	Simple transforms (50%)	Pixels swapping (50%)
Mean PSNR	24.4926	24.4862	24.5292	24.4997
Mean SSIM	0.828	0.829	0.828	0.827

Table II: Mean PSNR and SSIM indices on the validation set, depending of the presence of data augmentation and the type of data augmentation.

4. Discussion

We have shown that a simple auto-encoder can resolve a denoising task, only with pairs of noisy images and under a tight time constraint. This technique of comparing a pair of noisy images is useful, even necessary, as having a pair composed of a ground-truth and a noisy image can be logistically complicated, even impossible in certain areas like fluorescence microscopy or medical imaging. Greatly inspired by [1], various experiments have been conducted to adapt the Noise2Noise model to our noisy data set, using different loss functions, model structures, and quality measurements of the final results.

One important question we have explored is the quality measurement of the denoising task. How do we assess two images' similarity ? Based on two works of Wang ([4] and [5]), the SSIM index was used for its perceptual image quality metrics, which are closer to the human visual system by being more sensitive to structural information. With PSNR and SSIM indices, we could evaluate our denoising task in a more subtle way.

Another approach was to combine image processing techniques with the auto-encoder approachby using the wavelet loss. Indeed, wavelet transform uniformly spreads the noise across the coefficients, which could help to denoise the image. Unfortunately, we didn't obtained better results, and the images were still blurry. One path that we didn't explore was to add another constraint on the absolute value of the HH coefficients, as done in [3].

Moreover, data augmentation was used, but did not improved the efficiency of the denoising task.

Overall, the main constraint of our implementation was the time restriction of 10 minutes for the training on a GPU. Because of that, we had to simplify our model a lot, by implementing less layers of smaller size.

Finally, even using two metrics to assess the efficiency of our models, other better measurements could be used, like the multi-scale SSIM ([4]). Moreover, as mentioned in [1], the "L0" loss could be employed and tested.

References

[1] J. Lehtinen, J. Munkberg, J. Hasselgren, *et al.*, "Noise2noise: Learning image restoration without clean data", *arXiv preprint arXiv:1803.04189*, 2018.

[2] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library", in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[3] Q. Zhu, H. Wang, and R. Zhang, "Wavelet loss function for auto-encoder", *IEEE Access*, vol. 9, pp. 27 101–27 108, 2021. DOI: 10.1109/ACCESS.2021.3058604.

[4] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment", in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Ieee, vol. 2, 2003, pp. 1398–1402.

[5] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[6] A. F. Calvarons, "Improved noise2noise denoising with limited data", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 796–805.

[7] M. Unser, D. Van de Ville, M. Liebling, and D. Sage, *Image Processing*. EPFL, May 2022.

[8] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2129–2137.

Appendix

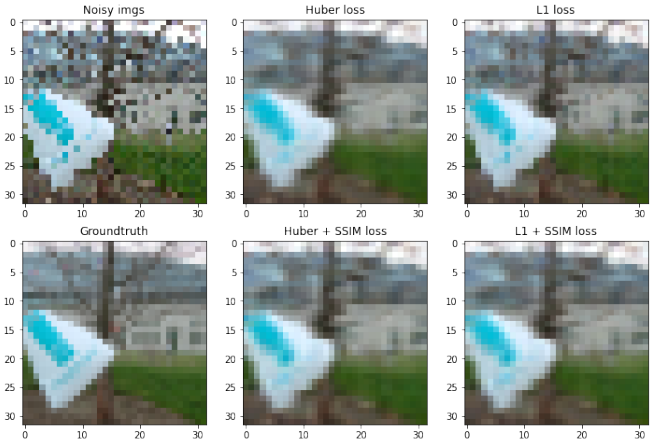

Figure A.4: Comparison of the denoised images, obtained by different models with losses including SIMM or not (see titles of the images), with the groundtruth (down left).

Id	N_{in}	N_{out}	Pooling	PSNR (loss)	SSIM
1	3, 16, 24, 24, 24, 16	16, 24, 24, 24, 16, 3	None	23.9 (L1)	0.8
2	3, 16, 24, 48, 48, 48, 48, 48, 24, 16	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	None	23.46 (L1)	0.79
3	3, 16, 24, 48, 48, 48, 48, 48, 24, 16	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Max	19.4 (L1)	0.47
4	3, 16, 24, 48, 48, 96, 96, 72, 40, 19	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Max	24.3 (L1)	0.826
5	3, 16, 24, 48, 48, 144, 144, 96, 56, 22	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Max	24.56 (L1)	0.829
6	3, 16, 24, 48, 48, 96, 144, 120, 64, 35	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Max	24.64 (L1)	0.83
7	3, 16, 24, 48, 48, 96, 144, 120, 64, 35	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Avg	24.38 (L1)	0.82
8	3, 16, 24, 48, 48, 96, 96, 72, 40, 19	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Avg	23.9 (L1)	
9	3, 16, 24, 48, 48, 96, 96, 72, 40, 19	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Avg	24 (L1)	
10	3, 16, 24, 48, 48, 96, 96, 72, 40, 19	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Avg	24.3 (L1)	
11	3, 16, 24, 48, 48, 96, 96, 72, 40, 19	16, 24, 48, 48, 48, 48, 48, 24, 16, 3	Max	24.7 (L1)	
12	3, 16, 24, 24, 48, 48, 96, 72, 72, 40, 19	16, 24, 24, 48, 48, 48, 48, 48, 24, 16, 3	Max	24 (Wloss)	0.83

Table A.3: List of several models with their respective performances and architectures. Except model 11 which uses SELU, every model uses LeakyReLU with negative slope of 0.1. After the last layer we clip the data to the range $[0,1]$.

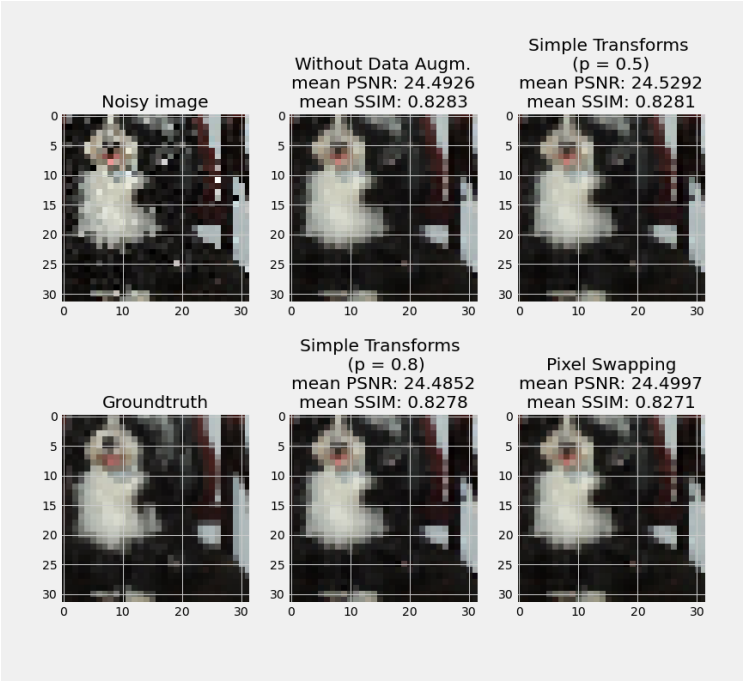

Figure A.5: Comparison of the denoised images obtained by models with different data augmentation (see title of the images), with the groundtruth (down left).