PyTorch implementation of Noise2Noise

Arturo Cerasi (342010), Francesco Borg (343230), Tomás Feith (342553)
Deep Learning (EE-559), EPFL, Switzerland

I. Introduction

Images are not always received in high quality, noise-free format. There are several steps along the image retrieval pipeline that may introduce noise or distortion into an image, such as distortions from the camera, loss of information from compression, or bad photography conditions. However, for several applications, this noise can have serious implications for their accuracy. This need justifies the existence of the statistical data analysis branch of Image Restoration, which studies image reconstruction techniques from incomplete or corrupt signals.

In 2018, Lehtinen *et al.* [1] introduced a method called Noise2Noise, which showed that is possible to "learn to turn bad images into good images by only looking at bad images". The purpose of this project is to create a PyTorch implementation of the Noise2Noise method and expand on it by studying several modifications to the model architecture, as well as different augmentation strategies.

Our main contributions are

- While we end up using Transposed Convolutions on the decoding branch of the U-Net, we show that there is no consistent advantage in this complexity gain;
- We study the effect of local and global crops as a data augmentation strategy, and see it leads to mixed results;
- We show that the insertion of an explicit sharpening operator after the U-Net leads to a better performance.

II. Models and Methods

A. Network Architecture

The default architecture used in this project will be a replica of the one used on the Noise2Noise paper, i.e. a U-Net. However, a few modifications will be implemented.

The original paper uses nearest-neighbor interpolation for the upsampling branch of the U-Net. It is possible to increase the complexity of these layers, and thus the representation power of the network, by replacing them with Transposed Convolutional layers. These allow to represent more complex transformations than a nearest-neighbor approach and thus should be able to better learn how to convert one image to the other.

To prevent the potential blurring introduced by the max-pooling and upsampling operations, and by the MSE loss (see section IV), we consider introducing an explicit sharpening operator at the end of the model. This operator is called unsharp masking [2], and it performs sharpening by computing an edge mask of the image and adding it to the original image. The edge mask is computed by subtracting to the original image a blurred version of it. This operation is represented as

$$f_{sharp} = f_{orig} + \tau_{sharp}(f_{orig} - h(f_{orig})) \tag{1}$$

where f_{orig} and f_{sharp} are the original and sharpened images, respectively, τ_{sharp} is the sharpening factor, and $h(\cdot)$ is the blurring transformation, which in this project will be a gaussian blur.

Finally, to make sure that the network's output was in [0, 255], the output was thresholded using two ReLUs, `x = ReLU(x)` and `x = x - ReLU(x - 255)`. The consecutive application of these enforces the range [0, 255] in the output.

B. Weight Initialization

He *et al.* (2015) [3] introduced a method for weight initialization that has since become standard, Kaiming Initialization. Under this method, the weights are initialized under a normal distribution, with mean 0 and variance given by $2/n_l$, where n_l is the number of inputs for the layer.

Following this, Kaiming Initialization was applied to all the Convolutional and Transposed Convolutional layers.

C. Loss Functions

The loss function is a key factor in the success of a deep learning model. This metric quantifies the quality of the predictions, in comparison with the expected values. As such, the choice of a proper loss function is vital.

In this project two loss functions will be considered, the MSE loss and the L_1 loss.

D. Optimizers

Two optimizers will be considered in this project: Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam). An in-depth overview of gradient descent optimization algorithms was performed by Ruder (2016) [4], and these were chosen because of their characteristics.

SGD updates the weights for every training batch. This makes the loss function fluctuate more, making it less likely for the model to get stuck on a local non-optimal minimum. However, these fluctuations also make SGD less stable as it approaches the optimal minimum since it can keep overshooting. A decreasing learning rate strategy can help prevent this.

Adam uses information from previous stages to compute the next gradient, via exponential moving averages. This means the algorithm is less fluctuating, converging faster. Furthermore, it is more likely to break symmetry, unlike SGD, which can get stuck in structures like saddle points. Nevertheless, this method is more likely to get stuck in local optimums, making the choice of starting conditions and a larger learning rate more important.

E. Data Augmentations

In the original paper, the only augmentation used was the addition of noise to the images. While on this project this will still be used, further elements will be studied. Specifically, the usage of crops (global and local), and horizontal and vertical random flips. Global crops are defined as including at least 70% of the original image, while local crops contain less than 40% of it. These transformations increase the variance of the training dataset, thus increasing the generalizability of the model.

III. Results

A. Full Ablation Study

1) Visual Comparison of Results: On figure 1 there are several examples of outputs from models in table I. By direct visual inspection, the outputs seem similar, but there are several key points where it seems that the modifications introduced in this work lead to better performance than the original Noise2Noise. Specifically, the new models performed better at erasing the misplaced pixels and softening the edges.

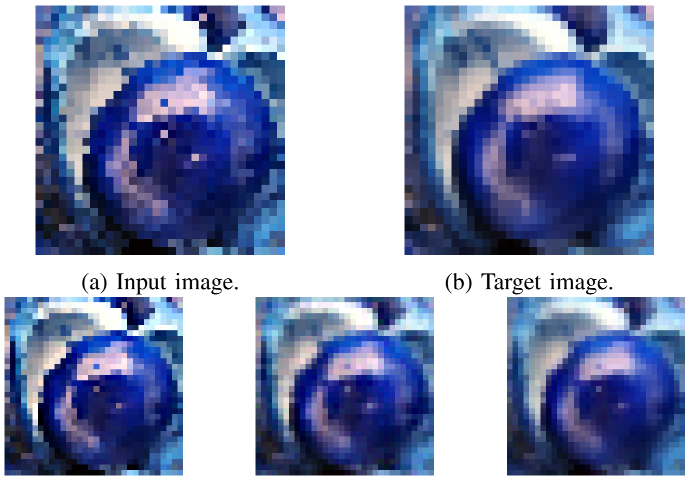
(a) Input image.
(b) Target image.
(c) Results using default Noise2Noise[1].
(d) Results with MSE loss and TConv.
(e) Results with crops and sharpening.

Fig. 1: Examples of outputs from different model variations.

2) Quantitative Comparisons: The variations in section II were studied, and the results are in table I. The PSNR values were computed as the mean of 100 evaluations of a random subset of 500 samples from the validation data, and the uncertainties are the standard deviations. This way we prevent overfitting to the validation dataset.

TABLE I: Ablations performed for training/architecture choices. PSNR_1 is the PSNR achieved with a TConv architecture and PSNR_2 is the PSNR achieved with an Upsample architecture. Models with crops trained for 10 epochs, and models without for 20. In bold are the top-3 candidates.

Loss	Optim	Sharp	Crop	PSNR_1 (dB)	PSNR_2 (dB)
L_1	Adam	✗	✗	20.82±0.14	20.74±0.14
L_1	Adam	✓	✗	23.57±0.13	23.57±0.13
L_1	Adam	✗	✓	21.57±0.12	21.63±0.13
L_1	Adam	✓	✓	23.55±0.12	23.61±0.13
L_1	SGD	✗	✗	20.54±0.12	20.85±0.11
L_1	SGD	✓	✗	22.49±0.10	22.61±0.13
L_1	SGD	✗	✓	20.62±0.10	20.97±0.11
L_1	SGD	✓	✓	21.59±0.09	21.52±0.10
MSE	Adam	✗	✗	22.61±0.14	22.53±0.12
MSE	Adam	✓	✗	**23.75±0.12**	**23.76±0.11**
MSE	Adam	✗	✓	21.86±0.11	22.01±0.12
MSE	Adam	✓	✓	21.29±0.09	22.53±0.10
MSE	SGD	✗	✗	22.49±0.11	23.32±0.11
MSE	SGD	✓	✗	**23.62±0.11**	23.60±0.12
MSE	SGD	✗	✓	22.48±0.12	22.25±0.11
MSE	SGD	✓	✓	22.80±0.10	22.34±0.11

On the iterations where cropping was used, one local and one global crop were considered. It should be noted that one epoch using one local crops uses the same number of images as two epochs without local crops. For the learning rate, an annealing strategy was considered, where after half of the epochs were complete, the learning rate became $lr \longrightarrow lr/10$.

[1]The default Noise2Noise model considered was the best achieving model from the original paper: Upsample, L_1 loss, Adam optimizer, no sharpening, no crops.

For notation simplification, the reference to *TConv architecture* refers to the model with Transposed Convolutions on the decoder branch, while an *Upsample architecture* refers to the model with nearest-neighbor upsampling.

B. Finding Best Candidate

1) PSNR evolution: Based on the previous section, the top-3 models were considered viable best candidates. To determine the one with the best balance between accuracy and computational cost, a study of the validation PSNR over the training epochs was performed. To simplify, shorthand notation will be used to refer to each of the candidates. Its explanation is in table II.

TABLE II: Explanation of the shorthand notation used for the three candidates for best model.

Notation	Arch	Loss	Optim	Sharp	Crop
Candidate 1	Upsample	MSE	Adam	✓	✗
Candidate 2	TConv	MSE	Adam	✓	✗
Candidate 3	TConv	MSE	SGD	✓	✗

From figure 2 it is evident that the models trained with Adam optimizer converged faster and with less fluctuation towards its optimal configuration, and to a better minimum. Candidates 1 and 2 perform very similarly, however Candidate 2 slightly outperforms 1 in the beginning of the training schedule. Since this model is intended to be used with short training cycles of 20 epochs, that makes Candidate 2 as the best choice.

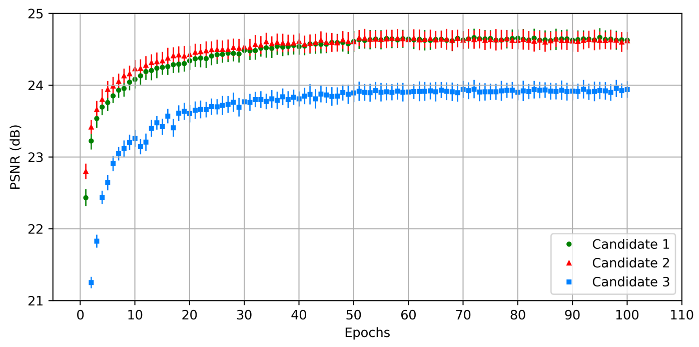

Fig. 2: Full evolution of the PSNR metric over the 200 epochs.

2) Parameter Fine-tuning: To achieve the best results from the model chosen above, its parameters can be fine-tuned. This optimization will be performed on the learning rate, both in its value and the decay strategy, the strength of the gaussian noise introduced in the data and the sharpening factor. The results obtained are in table III.

For the learning rate annealing strategy, the parameter changed was the number of steps performed and, on each step, the learning rate became 10× smaller. To exemplify, considering an initial rate of 5×10^{-2}, 3 steps, and 20 epochs, then after epochs 5, 10, and 15 the learning rate becomes 5×10^{-3}, 5×10^{-4}, and 5×10^{-5}, respectively. In the table, σ_n is the standard deviation of the gaussian noise introduced in the data, lr_0 is the initial learning rate, and τ_{sharp} is the sharpening factor. Omitted from the table are learning rate values larger than 0.001, as for these, the network would not converge, leading to values of PSNR~ 5. Like in table I, the PSNR values were computed as the mean of 100 evaluations of a random subset of 500 samples from the full validation data, and the uncertainties are the standard deviations from these evaluations.

TABLE III: Parameter fine-tuning was performed on a model with Transposed Convolutions on the decoder branch, an explicit sharpening operator, Adam optimizer, and MSE loss, trained for 20 epochs. For PSNR$_{base}$ no annealing was performed, and for PSNR$_{annealing}$ one decreasing step was performed. In bold we have the top-3 results.

σ_n	τ_{sharp}	lr_0	PSNR$_{base}$ (dB)	PSNR$_{annealing}$ (dB)
1	1	0.0001	23.89±0.12	23.64±0.12
1	1	0.0005	**24.42±0.12**	24.28±0.11
1	1	0.001	**24.46±0.12**	24.34±0.13
1	2	0.0001	16.48±0.09	22.20±0.08
1	2	0.0005	24.06±0.11	23.86±0.12
1	2	0.001	24.05±0.12	24.00±0.11
5	1	0.0001	24.00±0.14	23.71±0.12
5	1	0.0005	24.38±0.13	24.20±0.12
5	1	0.001	**24.46±0.13**	24.38±0.13
5	2	0.0001	23.73±0.11	15.10±0.08
5	2	0.0005	24.11±0.11	23.95±0.10
5	2	0.001	24.16±0.11	23.98±0.11
10	1	0.0001	23.83±0.12	23.72±0.12
10	1	0.0005	24.27±0.10	24.10±0.11
10	1	0.001	24.32±0.12	24.19±0.12
10	2	0.0001	23.72±0.11	23.59±0.10
10	2	0.0005	24.11±0.11	23.76±0.13
10	2	0.001	24.04±0.11	24.03±0.10

IV. Discussion

From table I, there is no proof that the complexity gained by using Transposed Convolutional layers, instead of nearest-neighbor upsampling, is justified. Indeed, over all iterations, this modification lead to an average PSNR variation of -0.13±0.04 dB. So, on average, TConv led to worse results than Upsample.

On the other hand, the sharpening operator was very beneficial, consistently improving the results. Comparing results with and without sharpening, there is an average increase of +1.18±0.04 dB.

As far as the loss function and the optimizer, the MSE loss gave consistently better results than the L_1 loss, and likewise Adam performed better than SGD. Adam may outperform SGD because the latter is slower, needing more epochs to converge, as is supported by figure 2. The results from the loss function are contrary to the ones in the original Noise2Noise paper. As they explain, there is a well-known tendency for the MSE loss to make the network learn to output the average of all plausible explanations, blurring the predictions. This was one of the reasons for introducing the explicit sharpening operator at the end of the network. Even though the sharpening operator improved the results, there doesn't seem to be a strong blur in the predictions. One hypothesis for this observation is that the images considered are too small, and the effects of blurriness are not significant at this scale. In future work we recommend the usage of different image sizes to measure its effect.

For the data augmentation strategy, the crops yielded mixed results. For some of the architecture choices they slightly improved the PSNR, but for others they led to a drop, especially when combined with the sharpening operator. This might happen because, with such small images, by selecting a local crop and reshaping it back to the full size, the blurring is too strong for the model to be able to take any benefits. Nevertheless, there are several hyper-parameters that can be better tuned, like the number and scales of local and global crops. A more in-depth study of these would be required for a conclusive statement regarding their usefulness.

Using the results from section III-B it is evident that the Adam optimizer converges faster than the SGD. Since it is desired to find a balance between results and computational cost, that makes Adam the best choice. Since Candidate 1 outperforms Candidate 2 in the beginning, and they converge to similar scores, Candidate 2 was selected as the best choice. After fine-tuning the training parameters for 20 epochs, the annealing strategy didn't show any advantages, possibly due to the short training schedule. From table III, it is also clear that higher learning rates lead to better results, and that when the gaussian noise gets too strong the performance decreases. Finally, increasing the strength of the sharpening operator decreased the performance of the model.

As a final remark, it is relevant to discuss the metric used to evaluate the model's accuracy. It has been shown by Ponomarenko *et al.* (2007) [5] that PSNR does not correlate well with human perception. As such, to better train the network concerning human perception one may consider a more complex metric, namely PSNR-HVS-M, which is a modified version of the PSNR introduced in that same paper that takes into account the Contrast Sensitivity Function and relates more closely to human perception. In future work this metric might be considered, potentially improving the visual quality of the outputs.

V. Summary

This project led to a PyTorch implementation of a Noise2Noise model. From the ablations performed it was identified that the optimal network is a U-Net with Transposed Convolutions on the decoder branch, with an explicit sharpening operator with factor τ_{sharp} of 1, using an Adam optimizer and MSE loss for training, as well as gaussian noise given by $\mathcal{N}(0, 5^2)$ for data augmentation. A network such as this, trained for 20 epochs with no annealing strategy and a learning rate of 0.001 can achieve a PSNR of 24.45 dB on the full validation set. Further work may focus on parameter tuning for the cropping-based data augmentation strategy, or on a different accuracy metric, such as the PSNR-HVS-M.

References

[1] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: Learning Image Restoration without Clean Data," *CoRR*, vol. abs/1803.04189, 2018. [Online]. Available: http://arxiv.org/abs/1803.04189

[2] E. R. Davies, "Machine vision: Theory, algorithms, practicalities." Oxford, England: Morgan Kaufmann, 2005, ch. 3.7.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 1026–1034. [Online]. Available: https://doi.org/10.1109/ICCV.2015.123

[4] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[5] N. Ponomarenko, F. Silvestri, K. Egiazarian, M. Carli, J. Astola, and V. Lukin, "On between-coefficient contrast masking of dct basis functions," *Proc of the 3rd Int Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 01 2007.