DL Miniproject 1 - A Denoising Model Trained using Noisy Labels

PHILIPPE, Jocelyn (288228) jocelyn.philippe@epfl.ch
DEVAUD, Quentin (287452) quentin.devaud@epfl.ch
KREMER, Iris (337635) iris.kremer@epfl.ch

I. Introduction

The goal of this mini-project 1 is to implement a model capable of denoising images while being trained with noisy images.

II. Data exploration and augmentation

The dataset comprises 50,000 pairs of noisy images of 32x32 pixels with 3 color channels. We do not know the exact type of noise applied to the images. We are also provided with a validation dataset comprised of 1,000 pairs of noisy images and their ground-truth.

In order to improve the robustness of our model, we implemented five different transformations to augment our dataset. The first one is a rotation transformation that randomly rotates the image by 90°, 180° or 270°. We restricted the rotation angle to these three values because otherwise some parts of the original image were not present in the rotated one, and we had to fill the remaining pixels with zeros (Figure 1). The four others transformations are related to the values of each pixel: images can undergo brightness, contrast, saturation and hue transformations. When we load the data, we pass it through an iterator that randomly transforms each image separately. Each transformation has a probability of 50% to be applied to each image. This iteration is performed once at the beginning of the training. We tried to re-iterate during the training (e.g. every 4 epochs) to shuffle the transformation but it was taking too much computation time and thus reducing the total number of epochs we could run under the original time constraints of the project.

At first, we also wanted to implement affine transformation (zoom, rotation, translation and shear) but it was not suitable in our case. Translation and zoom-out had the same problem than rotation with non-fixed angles: parts of the image had to be filled with zeroes which induced an important loss of information (Figure 1). Zoom in and shear could be used without filling pixels but the resolution of our image was already so small that we thought this kind of transformation would be too drastic to improve the robustness of our model.

III. Models and methods

Models presented in this report were evaluated with the Peak Signal to Noise Ratio (PSNR) function defined as:

$$PSNR = -10 \cdot log_{10}(MSE)$$

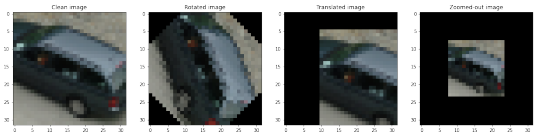

Fig. 1. Example of problematical affine transformations

with MSE standing for the mean squared error between the output of the model and the ground truth images.

At first, we decided to implement the Noise2noise model as described in the reference paper [1]. However, we wanted our model to output values between 0 and 1, and the initial implementation does not have an activation layer at the end, meaning the output is unbounded. To solve this, we thought about applying a min-max scaling to the output, but it implied that the scaling was different for each image. Therefore, a pixel value of x in output would not give the same color depending on what are the maximum and minimum value of the output. Hence, we decided to use the hard sigmoid function. We chose this version over the usual sigmoid, because it allows to output values between 0 and 1 included. This is not the case for the sigmoid where the output only converges towards 0 and 1, yet never reaches them. In fact, during our tests with sigmoid, the model was not able to output values lower than 3 or higher than 250 (when scaled between 0 and 255).

The Noise2noise model is a U-net taking any rectangular image[4] as input. It uses a series of five convolutions + max-pooling blocks, dividing x and y dimensions of the image by 32. The convolutions extract features from the image to the different channels, and the max-pooling layers filter the information to only retain the most important parts, i.e. the maximum values. Then, five upscale layers composed of nearest neighbor upsampling + convolution reconstruct the image in its original shape with less noise using the extracted features. Upsampling + convolution blocks are usually used instead of transpose convolution to avoid checkerboard artefacts as described in [2]. There is a skip connection from each max-pooling to each upscale (i.e. the first max-pooling layer is connected to the last upscale and so

[4]Both x and y dimensions shall be a multiple of 32

forth) to help retaining the original information of the image. Considering the amount of convolution operations performed by the model, training it is quite demanding. We hypothesised that simplifying the model could reduce the computational complexity, thus allowing us to train for more epochs in the allowed running time without losing performance. On top of that, simplified model would have less weights to tune and thus the training would me more efficient. We recursively created four simplified models based on the architecture of the original Noise2noise model. Each of them cuts the last remaining dimension reduction step, so that the first simplified model (Noise2noiseSimplified1) divides each dimension by 16 and the last one (Noise2noiseSimplified4) only divides them by 2.

We experimented with three different loss functions during the training: the mean absolute error (MAE), the mean squared error (MSE) and the Huber loss (HU). MSE is tested because it appears in the PSNR's formula, so it seems consistent to minimize it. Then we tested MAE because it would make sense that distant pixel values do not have higher weight on the loss. We also chose HU because it is a good compromise between MSE and MAE and it might benefits from both functions' advantages. We will only use the loss function leading to the best results to train the four simplified models.

Regarding the batch size for the training, we did not want a small value because the estimation of the gradient would not be accurate. However, we did not want it to be too high because it would reduce the number of update per epoch, thus increasing the time of computation required to reach a good model. In the end, we chose a batch size of 64 as a good compromise. As the index of the images in the dataset are shuffled at each epoch, we can suppose that batches of 64 images are independent and identically distributed. Finally, in order to respect the time limit for training, we chose to train all our models for 13 epochs. It gives us a training time of 9 minutes on a laptop computer equiped with a GTX 1650. Models are evaluated by computing the mean PSNR on the validation dataset.

IV. Results

First of all, we test the training with two optimizers: SGD and Adagrad. Results show that Adagrad perform better in every configuration we tried, hence we decided to use it for the rest of our trainings.

We then test the different loss function in the configuration described in section III (13 epochs and batch size of 64 images) with four different learning rates: 1e-3, 2.5e-3, 5e-3 and 7.5e-3. The best PSNR obtained for each loss is shown in Table II. For every loss function, the model does not converge with a learning rate of 7.5e-3 and the maximal PSNR is reached with the learning rate equal to 5e-3.

Loss function	Best PSNR
MAE	24.18
MSE	24.36
HU	24.52

TABLE II
BEST PSNR ACHIEVED WITH EACH LOSS FUNCTION

Even though the results are close to each other, we can conclude that the Huber loss is best suited for our problem. Hence, we will use this one for the training of all the other models. Next step is to fine-tune all our models to determine which one is the best at denoising images. Table I shows the best PSNR obtained with each model and each learning rate tested.

Looking at these results, we can see that simplified models 1 and 2 have worse PSNR than the original model while still having similar training duration. It seems that the gain of efficiency is not big enough to compensate the loss of complexity, and thus accuracy. On the other hand, simplified models 3 and 4 obtain better results than baseline Noise2noise. They are also less sensitive to nonconvergence as they are able to converge with a large learning rate (i.e. 7.5e-3). It seems that they are both simplified enough to benefit from the lower computational complexity.

[2]c. = compression

[3]This result only happen once, it almost never converges with this learning rate

[4]We did not train with this learning rate as it already does not converge with a learning rate of 5e-3

Model	lr = 1e-3	lr = 2.5e-3	lr = 5e-3	lr = 7.5e-3
Noise2noise (32x c.[1])	23.06	24.20	24.52	24.64[2]
Simplified1 (16x c.)	23.09	24.29	24.37	5.9
Simplified2 (8x c.)	23.20	24.20	6.42	/[3]
Simplified3 (4x c.)	23.18	24.35	24.58	24.46
Simplified4 (2x c.)	23.40	24.29	24.53	24.84

TABLE I
PSNR FOR EACH MODEL AND EACH LEARNING RATE TESTED

Looking at several outputs of our model, we discovered interesting behaviors. First, as we can see in Figure 2, output images are blurry compared to the ground truth, even when the PSNR is very high. It is quite logic because our model is pretty simple and is trained with noisy images as targets. Hence, the model's best strategy for reducing noise is to output a blurry mean of the two images.

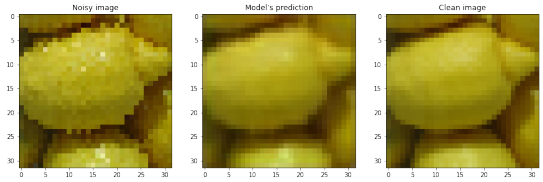

Fig. 2. Example of output of Noise2noiseSimplified4 with high PSNR (30.42)

However, there are some cases where the model is not able to properly denoise the input image. In Figure 3, light reflections on the salt shaker are interpreted by our model as noise and it tries to remove it although it should not. The same problem can appear in images containing specific geometric patterns, as shown in Figure 4 in which the model is unable to recover the pattern of the dome from the noisy image.

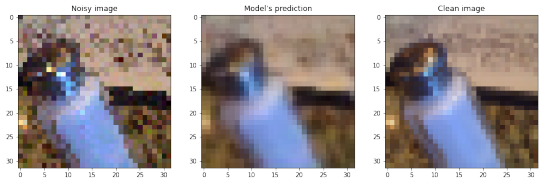

Fig. 3. Example of output of Noise2noiseSimplified4 with reflections in the input

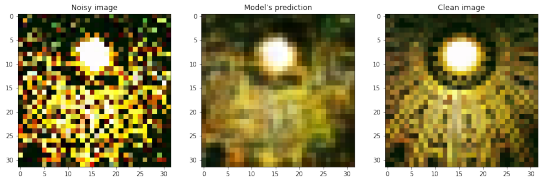

Fig. 4. Example of output of Noise2noiseSimplified4 with specific geometric patterns

V. Conclusion

In conclusion, we showed in this report that simplifying the baseline Noise2noise model is a viable solution for our task when facing training time constraints. It allows to reduce the computational complexity while still maintaining a high accuracy on small images.

References

[1] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.

[2] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard