EE559-Deep Learning Project 1
Using the standard PyTorch framework

Malena Aguiriano Calvo - Henry Papadatos - Aitana Waelbroeck

***Abstract*—In this report, we tackle the task of creating an image denoising network, which is trained without a clean reference image.**

I. Introduction

Deep learning methods for image denoising predominantly use discriminative images where the training is done on pairs of noisy input images and clean target images. However clean targets can be rare and difficult to obtain. Recent advances in the field however have shown that images can be denoised just as well if not better, by using only noisy images, such as in the paper Noise2Noise [1]. In this report we will develop our own deep learning network in order to denoise images without a clean reference. For the first part, we will explain how we developed the model, while in the second part we will analyze the results obtained.

II. Data

For our data we have 10000 test images and 50000 train images, that are 32 by 32 pixels and have 3 channels. We performed the data normalization by dividing by the maximal value of a pixel (255) so the range was between 0 and 1.
To measure our network's performance, we used the Peak Signal-to-Noise Ratio (PSNR) metric.
The data for the predicted images was clipped at 255, so it remained in the range 0-255.

III. Results

A. Hyper Parameters Selection

We experimented with multiple learning rates, η, in order to select one that would work with our data. From the graph below we can see that a learning rate of 0.002 is acceptable for our network.

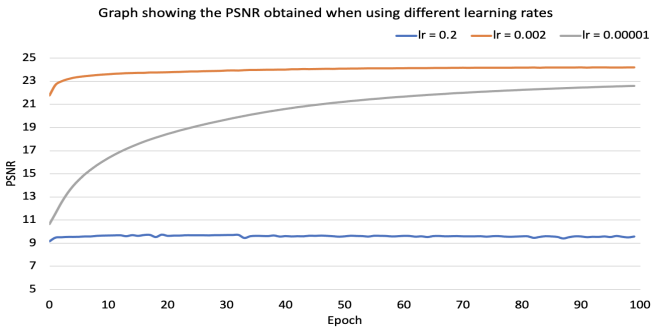

Fig. 1. Learning rate

We selected the Mean Squared Error (MSE) criterion for our network. MSE measures the mean squared error between each element in the input and target. When compared with the cross-entropy criterion, below we can clearly see a better performance from the MSE. In the following graph we see their performances:

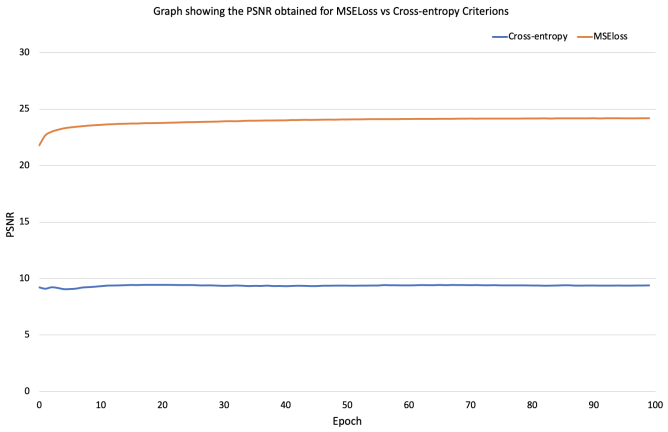

Fig. 2. Criterion Comparison

It makes sense that the cross-entropy criterion is not adapted to our problem. Cross-entropy modifies each output in function of the others (it is adapted for classification).

We used the Adam optimizer for our network, as it had a better performance than other optimizers, such as SGD. The graph below shows the comparison in performances:

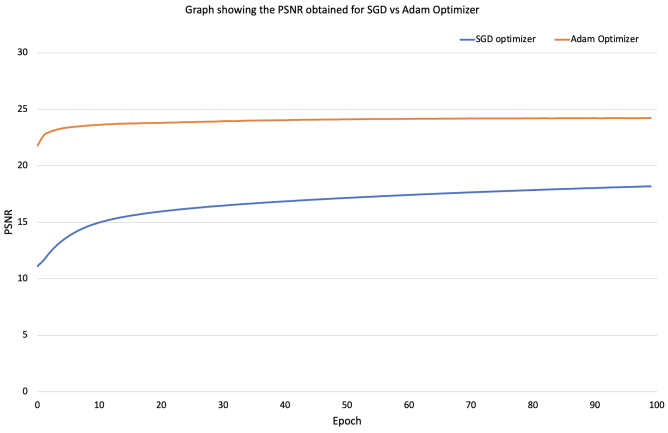

Fig. 3. Optimizer Comparison

In order to see if we would need to use all 50000 images in our dataset, we compared its performances with 30000 images. We also performed data augmentation to see how it would compare. This can be seen in the following graph:

Fig. 4. Number of images used

We can see that a smaller dataset has a slightly lower performance than when using out full dataset. We can also see that with the data augmentation we used we get a similar behavior, minus the oscillation to the full set. Thus we used the 50000 images to train our network.
Summary of the hyperparameters we selected:

Hyperparameter	Value
Criterion	MSE
Optimizer	Adam
Learning rate	0.002
Number of images used	50000

We also used batch sizes of 1000 images.

B. Network Architecture

We explored various network architectures in order to find a good one. In general, we used an hourglass shape architecture. Such an architecture is appropriate to encode the important features of an image in a lower dimension and then bringing it back to the original shape. Doing so will reduce the noise as it not represented in the important features [1].
The first major improvement we obtained was when we added skip connections. This makes sense, as the skip connections preserve the structure of the images better and therefore the model is easier to train.
We tried to use ReLU and Sigmoid as the last activation layer. Sigmoid worked better which is sensible as we normalized our images between 0 and 1.
We tried to use 4 and 6 convolutions. We had the constraint that our model should be trained after only 10 minutes on a GPU. Therefore, a 4 layered model worked better, but a 6 layered network might have performed better if it was trained longer.
We tried to add a linear layer in the bottleneck, which resulted in a very bad PSNR. This is probably caused by the fact that a linear layer completely destructures the images, which makes the network harder to train.
And finally, we tried to use various kernel sizes for the convolutions.

The following graph shows the performances of the various models we tried. Note that these models were only trained on 10 thousands training images to speed up the process.

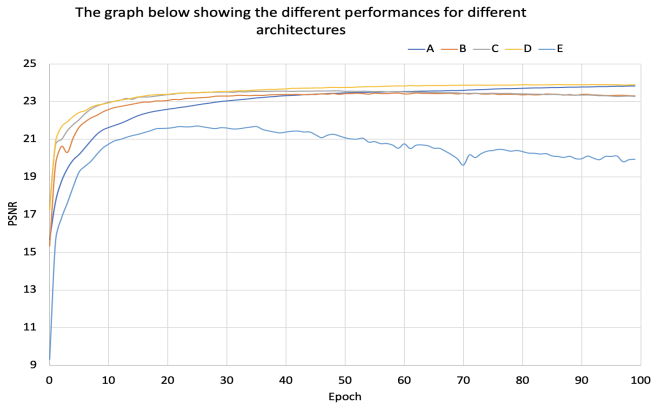

Fig. 5. Comparison of performances for different architectures. A: Uses ReLU as last activation, B: uses sigmoid as last activation, C: same as B but has 4 convolution layers instead of 6, D: same as C but increase the kernel size of the 2 middle convolutions from 3 to 5 E: has a linear layer in the bottleneck. **The model we kept is D**

The best network we found has the following architecture:

1) Conv2d(kernel size = 3, stride = 1)
2) BatchNorm
3) ReLU
4) skip connection $\rightarrow$
5) Conv2d(kernel size = 5, stride = 2)
6) BatchNorm
7) ReLU
8) skip connection $\leftarrow$
9) BatchNorm
10) skip connection $\rightarrow$
11) ConvTranspose2d(kernel size = 5, stride = 2)
12) BatchNorm
13) ReLU
14) skip connection $\leftarrow$
15) ConvTranspose2d(kernel size = 3, stride = 1)
16) Sigmoid

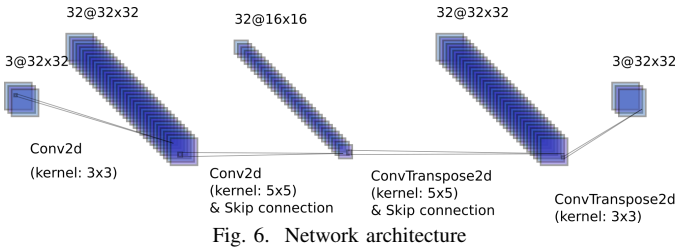

Fig. 6. Network architecture

IV. Discussion and Conclusion

Below are four images selected from the dataset, the first figure shows the clean image, the second shows the input

(noisy) and the third shows the denoised image found by our network.

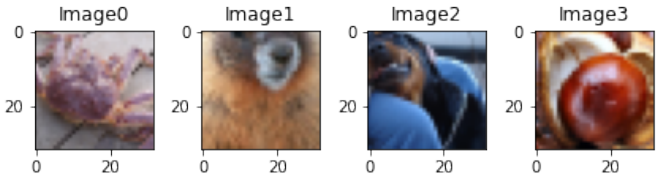

Fig. 7. Clean Images

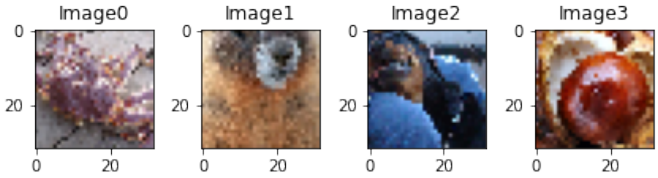

Fig. 8. Noisy Images

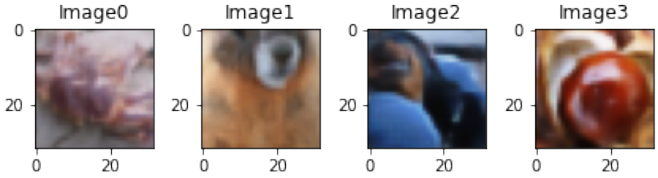

Fig. 9. Denoised image

We can see qualitatively that the clean image and the denoised image from our network are quite similar, strongly indicating that our network is performing well. The PSNR between the clean image and the noisy image was 19.62, with our network we were able to obtained a PSNR of 24.20, showing that our network was able to denoise images.

We noticed that the dimension of the tensor at the bottleneck of the model should not be too small. Otherwise, the model can not encode enough features to reconstitute the image properly. We also learned that residual networks have the capacity to preserve the structure of the data and therefore make models easier to train for our objective of reconstructing denoised images.

In conclusion, our network architecture was able to denoise images to a large extent, while only using corrupted/noisy images as input and target. Our paper shows similar conclusions to that of Noise2Noise [1], suggesting that clean targets are not necessary to train a network to denoise images.

V. Bibliography

[1] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, Timo Aila. “Noise2Noise: Learning Image Restoration without Clean Data”. In: (). URL: https://arxiv.org/pdf/1803.04189.pdf.