Deep Learning Projet 1 Report
noise2noise

Erwan Serandour, Julian Blackwell, Ghali Chraibi

May 2022

1 Introduction

For this first miniproject, we implement a noise2noise model: a network that learns to reconstruct corrupted images without any clean targets. We compare the performance and convergence time of a range of deep convolutional architectures for this task.

In this report, we first describe how we are compared the different architectures. Then we evaluate the different networks between each other and discuss the results. Finally, we select the best model based on our exploration for the submission.

2 Methodology

The comparison of two deep architectures can be quite tricky. Hyper-parameters such as the learning rate can significantly influence the results and bias the analysis. Therefore, we will attempt to make the comparisons as fair as possible, and focus on the significant differences between these models instead of chasing numbers.

To compare all subsequent networks, we fix beforehand the following hyper-parameters. For the optimizer, we choose to use the **Adam optimizer** as it is widely used in deep learning and is generally pretty efficient. It also gives the models the opportunity to adapt their learning increments during training. For the the learning rate, we select a **constant learning rate of 3e-4** for all the networks. We train each network for **1000 gradients steps** with **a batch size of 128** which, depending on the network, represents between one and five minutes of runtime on a Colab GPU. As our loss metric, we use the **Mean squared error** (MSE) since maximising the peak signal-to-noise ratio (PSNR) corresponds to minimizing the mean squared error. This statistical argument serves as the basis upon which the noise2noise papers rest on. Finally, we compute the PSNR for each model using a validation set containing 500 corrupted images with their clean version as target.

3 Networks comparison

3.1 Architectures description

In our exploration we decide to consider 4 networks, all of which are convolutional but with interesting differences. Since our models work with image data, it seems appropriate to use convolutional layers which are good at extracting information while keeping images' structural links. Hence, we compare an 8-layer convolutional neural network (ConvNet or CNN), a residual neural network (ResNet), a Unet and a modified Unet.

The ConvNet serves as our baseline. The ResNet is similar to the ConvNet, but adds skip connections [1] that helps to tackle the vanishing gradient problem especially for very deep neural network. Unet [2] is a standard network for image segmentation and denoising. We also adapt the Unet architecture based on the task and our intuition. As an additional note, after each convolution (unless noted) we add a **BatchNorm2d** layer and use a **ReLU** activation for the network to be able to learn non-linear models.

ConvNet The ConvNet has 8 convolutional layers with respectively the following number of output channels: [64, 128, 128, 256, 256, 128, 64, 3].

ResNet The ResNet is composed of 4 ResNet blocks with the following number of channels: $[64, 128, 128, 256]$. We insert a convolution layer with a 3x3 kernel between ResNet blocks to adapt the number of channels (see Fig. 3a).

Unet A Unet is a network that has a symmetric contracting and expanding path. We implement each path using a combination of Resnet blocks and maximum pooling layers for downsizing and interpolation layers for upsamling (see Fig. 3b).

Unet + concatenation After analysing results from previous architectures, we identify that sometimes the model has difficulties at picking the right color. Thus, we decide to add a 3x3 convolution at the end of the Unet which takes as input the concatenation of the model output with the input image.

3.2 Results

We can observe both significant differences in training time as well as in the obtained PSNR between architectures (see Table 1). First, we compare our models with the simple convolutional baseline. We can see that both introducing skip connections as well as including a contracting and expanding path improves the obtained PSNR by at least 1 dB. The main difference being that the Unet trains more than 2 times faster than the ResNet, although it takes a higher number of gradient steps to achieve a similar PSNR score (see Fig. 3). The ResNet architecture on the other hand, starts off strong but seems to only slowly improve, with even some diminishing returns at times. As a last alternative, concatenating the input image with the Unet output as a final step of the model before the last activation once again improves the results by 0.7 dB. This architecture starts strongly only second to ResNet but quickly improves

to surpass the other models' performance. Clearly, the Unet with an added concatenation step seems to be the best preforming model for this task.

	PSNR	gradient steps	lr rate	time
ConvNet	22.5	1000	3e-4	2:50
ResNet	23.5	1000	3e-4	4:41
Unet	23.8	1000	3e-4	**1:45**
Unet + concatenation	**24.5**	1000	3e-4	1:51

Table 1: Comparison of performance

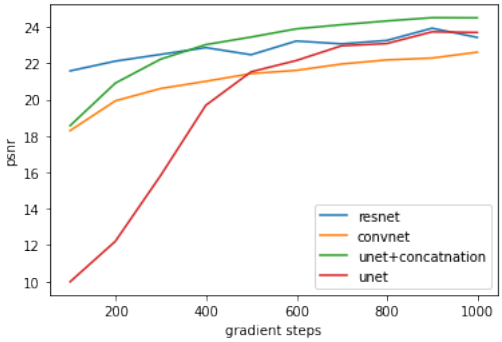


Figure 1: Validation PSNR results

4 Final model results

We train the Unet with the added concatenation layer for 5:13 minutes and 3000 gradient steps and obtain a PSNR of 25.26 db on the full validation set an improvement of 3.2 db compared to a simple baseline. We train the model only for 5 minutes because we found that the model started to overfit afterwards and, due to lack of time, we chose to use early stopping for regularisation.

5 Conclusion

In this report, we compared the impact of adding skip connections, using symmetric contracting and augmenting paths, as well as concatenating the input image before the last layer of the model. All of these methods have shown significant positive impact compared to a simple baseline CNN. For this specific task, we found that the Unet architecture was most efficient for a high performing model. After understanding one of the model's difficulties in learning the correct colours, we adapted the final stage of the architecture to include the original input. This led to a notable gain in performance with only a marginal measured computational tradeoff. With more time, we would have liked to be able to compare different image augmentations such as mixup augmentation or adding noise to our image to avoid overfitting.

References

[1] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: 7 (Dec. 2015).

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: vol. 9351. Oct. 2015, pp. 234–241. ISBN: 978-3-319-24573-7. DOI: 10.1007/978-3-319-24574-4_28.

Appendix

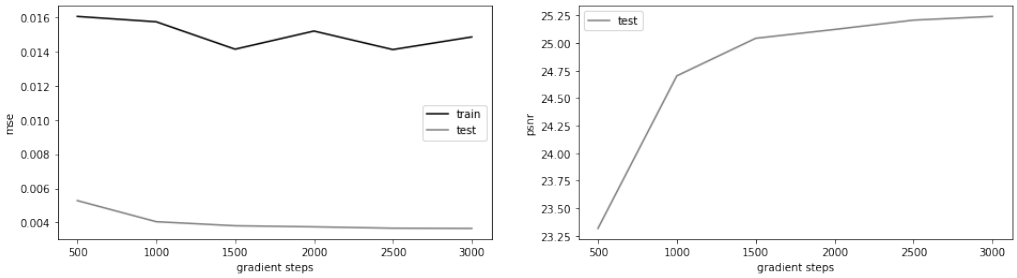
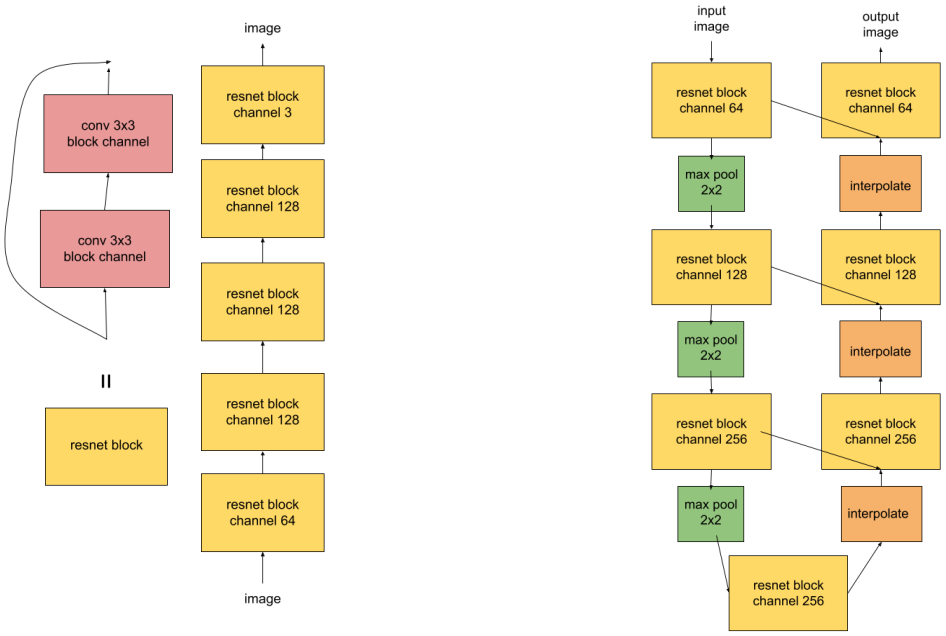


Figure 2: Final model training - Note that the MSE of the test set is lower than for the train set as we used clean label for the former



(a) ResNet (b) Unet

Figure 3: Network architectures