EE-559 Miniproject 1 – Spring 2022

Ali Garjani (Sciper: 336721), Sepehr Mousavi (Sciper: 338673), Hossein Taji (Sciper: 322030)

I. Introduction

The objective of the project is to implement and train the Noise2Noise model presented in [1] for image denoising, i.e., removing noise from input images. The train dataset for this project consists of 50000 pairs of 32x32 pixels noisy images. After the training, the model is used on a validation dataset consisting of 1000 pairs of 32x32 pixels images, in which the first element of each pair is the input noisy image, and the second element are the target clean images. Through this report, first, we explain the architectures used for the denoising model. Then, we explore the effect of applying different loss function in order to study the type of the noise in the dataset. Furtheremore, we try different methods to augment the train dataset such as adding extra Gaussian noise, random flip and rotate, and random pixel exchange between the image pairs, which are explained in the next sections. Finally, we discuss the implementation details and results obtained by the mentioned methods for denoising.

II. Model architecture

A. U-net

As depicted in Figure 1, U-net, an encoder-decoder CNN architecture first presented in [2], is composed of two main parts: an encoder side during which the features of interest for denoising are extracted, and a decoder side during which the output is reconstructed using the high-level features and the convolutions from the first part. Each side consists of several blocks with a convolutional layer in each block. To avoid the original information loss, the output of each block in the encoder is concatenated to the output of the previous block of the mirrored block in the decoder. As for the architecture, we used the same architecture represented in the Noise2Noise paper.

B. Resnet encoder-decoder

This network has the same general architecture as the U-net. However, in this network there are no concatenations of output of encoder layers with the mirrored decoder layers. To avoid the gradient and information loss, instead of using normal blocks with one layer of convoloution, a Resnet block with two convolution layer is used.

III. Loss functions

Since the type of the initial noise in the dataset is unknown, we try two different loss functions to examine which one suites best with the noise distribution in the dataset. The examined losses are the $L1$ norm and the $L2$ norm. The $L1$ norm is used to estimate the median of the distribution, whereas the $L2$ norm is used for estimating the mean of the distribution. As concluded in [1], for noise distributions such as Gaussian or Poisson, the $L2$ norm would be the best choice for the loss function. But in some cases where the average of the noise can cause a color shift on image pixels (such as text removal applications), the $L1$ norm would be a better loss function only if less than 50% of pixels are noisy. If more than 50% of the pixels are noisy we will also see a shift in color for the output of this loss, as the $L1$ norm tries to estimate the median of these pixels.

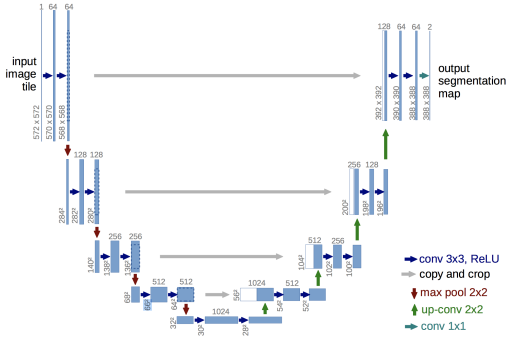


Figure 1: U-Net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.[2]

IV. Data augmentation

Although the given dataset already contains noisy images, having the same noise pattern in each iteration of training for one image can cause the model to overfit. In order to avoid overfitting and even generalize the model, we used different augmentation methods such as flipping and rotating the image, random exchange of the pixels, and adding additional Gaussian noise to the training images. In the next section, we will explain how each data augmentation method is applied.

A. Flipping and Rotating

For this augmentation method, at each training epoch, each image and its target are randomly (uniform distribution) flipped horizontally, vertically, or not flipped at all. After the first transformation, they are rotated by a degree randomly chosen from $\{0, 90, 180, 270\}$.

B. Random Exchange of Pixels

In this augmentation method, for each noisy image pair, the values of the pixels are randomly swapped between the two images in one pair. In other words, if we assume the i_{th} pixel of the first image and the second image from one pair to be p_i and q_i respectively, then by 0.5 chance we swap the values of p_i and q_i. Using this augmentation, even if we do not know the noise distribution, we are artificially generating more data with the same distribution. This data augmentation method takes advantage of the fact that both series of images in the train dataset are noisy.

C. Additional Gaussian Noise

The last implemented data augmentation method is adding a Gaussian noise during the training. Although this extra Gaussian noise might not have the same distribution as the initial noise in the training dataset, it can help the model to avoid overfitting. To generate the noise, during each training iteration, the standard deviation of the Gaussian distribution, σ, is uniformly randomly chosen from the range $[0, \sigma_{max}]$.

V. Results

We used the PyTorch library to implement the model. The model is trained on the training dataset and tested on the validation dataset, with several different configurations and hyper-parameters. For each configuration the model is trained for at least 30 epochs until the loss on the validation dataset is no more improving. The models are trained using a batch size of 128 with an initial learning rate of 0.001. At each epoch, the model parameters are saved if there is an improvement in the result of that epoch. The decision on each feature or hyper-parameter is made based on the results presented in this section. For the experiments, first we tested different model and loss function configurations to find the best model and loss for this dataset. After fixing the model and loss function, we experimented different augmentation methods. In all the experiments, the evaluations are done using the PSNR and SSIM metrics. All of the mentioned experiments were done using GPU on Google Colab. While running on Google Colab each epoch took at most 78 seconds to run.

A. Model and Loss function

Figure 2 illustrates the average PSNR values obtained on the validation dataset for each epoch. As represented in this figure, U-Net with the $L2$ loss function outperforms the configurations. The dashed line in the plot is the baseline, which is the average PSNR of the noisy image and the clean image in the validation dataset. We also observed that the Resnet autoencoder model (represented as AE in the plot) performs even worse than the baseline. This bad performance can be due to information loss, as the Resnet autoencoder is deep and cannot propagate the image's useful information for reconstruction as good as the U-Net architecture. For both models, we can see that the $L2$ loss performs better than $L1$, hence we can conclude that in terms of the quality of the images, the mean estimate of the noisy images gives us a better result than the median estimate.

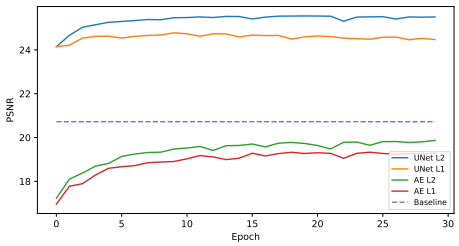
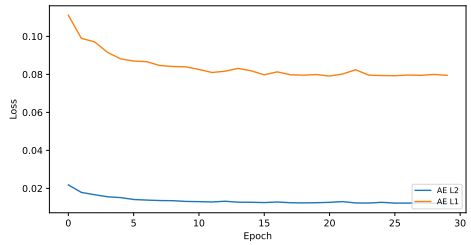
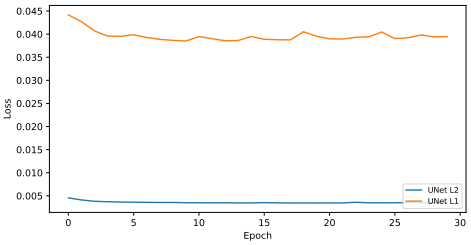


Figure 2: The blue and orange curves in this figure show the results of the U-Net for loss functions $L2$ and $L1$ respectively. The green and red curves also show the results of the Resnet autoencoder for $L2$ and $L1$ loss functions. The dashed purple line in the figures represents the baseline.

Figure 3a and Figure 3b show the $L1$ and $L2$ loss values of the two model over different epochs. In Figure 3a, we observe convergence of the Resent autoencoder model after around the 12^{th} epoch for both loss functions. However, for U-Net in Figure 3b, the convergence occurs around the 5^{th} epoch.



(a)



(b)

Figure 3: Figures showing the loss values for different epochs (Figure 3a for the Resent autoencoder, and Figure 3b for the UNet model). In Figure 3a the Resnet model converges after around 12 epochs, whereas the U-Net model in Figure 3b shows convergence after 5 epochs.

Since the U-Net model with the $L2$ loss has the highest performance amongst other configurations, we select this

model as a baseline to compare different data augmentation methods explained in the next section.

B. Data Augmentation

Regarding the additional Gaussian noise, Figure 4 illustrates the performance of the model under different values of σ_{max} for the additional loss. Based on the plot, adding the Gaussian noise does not enhance the performance significantly. Although this extra noise might help generalizing the model, it changes the initial noise distribution, which can deteriorate the performance of the model on the validation dataset.

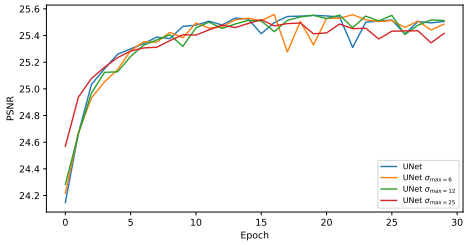


Figure 4: This figure shows the performance of the additional gaussian noise data augmentation for σ_{max} values: 6, 12 and 25. In general, adding this additional noise does not enhance the results very much, as it changes the intial noise distribution.

Figure 5 shows the effect of each data augmentation method on the validation PSNR score. Among all the augmentation methods, random exchange of pixels and random flip and rotate have the best gain in performance. For the random exchange of the pixels, we can see the enhancement in performance even in the first few epochs.

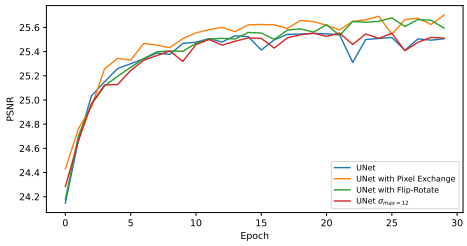


Figure 5: The PSNR values of different augmentation methods for each epoch. As shown, random flip and rotation and random exchange of pixels, enhance the performance, as they keep the initial noise distribution the same while augmenting the data.

Table I summarises the top results for different settings experimented. The metrics used are the PSNR score and the SSIM score. The SSIM metric measures the structural similarity between the two images. The SSIM is measured using a window size of 11 and a window sigma of 1.5. As shown in the table, the best results are obtained by doing a random flip and rotate or random exchange of pixels on U-Net with the $L2$ loss function. Figure 6 shows visual examples of the denoised images, using our top model, for two samples from the validation dataset. As illustrated, the model removes the noise from the images, but as a result, it also introduces a blur effect on the image.

Table I: Results of performance of different models using different loss functions and data augmentations. The quality of image reconstruction is measured using the PSNR and the SSIM metrics. the U-Net model trained with the $L2$ loss and augmented with random flips and rotates or random pixel exchange beats other configurations.

Model	Loss	Data Augmentation	PSNR (dB)	SSIM
UNet	$L2$	-	25.55	0.846
UNet	$L1$	-	24.78	0.838
AE	$L2$	-	19.91	0.504
AE	$L1$	-	19.44	0.487
UNet	$L2$	Gaussian Noise $\sigma_{max} = 12$	25.55	0.846
UNet	$L2$	Random Flip and Rotate	**25.77**	**0.852**
UNet	$L2$	Random Exchange of Pixels	**25.74**	**0.852**

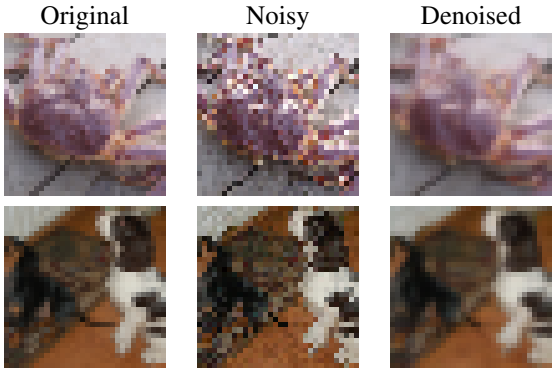


Figure 6: Visual examples of image denoising using the obtained top model. The PSNR score of the denoised images for the first row and the second row are 24.35 dB and 26.18 dB. The model manages to remove the noise from the image, however, it also makes the image blurry.

VI. Conclusion

The aim of the project was to implement, train and test the Noise2Noise denoising model. This paper argues that the denoising can also be done only using the noisy images and without the clean targets. For implementing this model, we experimented on different architectures using different loss function and data augmentation methods, and the best result was obtained by the U-Net model trained with the $L2$ loss function, where the training dataset was augmented using random flip and rotate or random exchange of the pixels.

References

[1] Jaakko Lehtinen et al. "Noise2Noise: Learning image restoration without clean data". In: *arXiv preprint arXiv:1803.04189* (2018).

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].