Implementation of Noise2Noise models to learn image restoration without clean data

Miniproject 1

Camillo Nicolò De Sabbata, Stefan Igescu, Giovanni Monea
Department of Computer Science, EPFL, Switzerland

***Abstract*—This report presents the implementation of a deep neural network that allows to restore images by only looking at other corrupted examples, without using clean data. The implementation follows the work of Lehtinen et al., 2018 [1].**

I. Introduction

In this project, the goal is to find the best neural networks to learn noise removal in absence of a clean reference. The data training set is composed of a training set of 50000 RGB image pairs of size 32×32, while the validation set is composed of pairs of noisy-clear images. Random data augmentation techniques were applied, as explained later in the report, without any improvement in the result. The performance metric used was the Peak Signal-to-Noise Ratio (PSNR), defined as the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation, measured in decibel.

II. Methodology

We examined the performance of three types of networks: a classical denoising autoencoder, used as baseline, a REDNet (from the work of Mao et al., 2016 [2]) and the UNet used by Lehtinen et al., 2018 [1].

A. Autoencoder

As a baseline for our study we chose a standard autoencoder, a network used to learn efficient encodings of unlabeled data. It is composed of two parts: an encoder that maps the input into the latent space, and a decoder that maps the the representation in the latent space to a reconstruction of the input. The encoder serves the function of a feature extractor, maintaining the primary components of objects in the image while removing corruptions. The decoder recovers the image content's information. Our network is composed of 10 convolutional layers for the encoding and 10 symmetric transposed convolutional layers for the decoding.

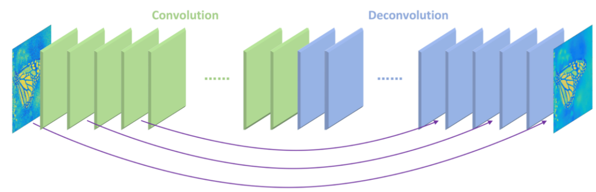

Fig. 1. REDNet by Mao et al., 2016

B. RedNET

In this deep Residual Encoder-Decoder Networks (REDNet), the encoder is composed of 15 convolutions and the decoder of 15 symmetric transposed convolutions; rectification layers are added after each convolution and deconvolution. Moreover, skip connections are also added from a convolutional layer to its corresponding mirrored deconvolutional layer.

C. UNet

Finally, the UNet is composed of the following modules:

- Conv: convolution and rectification
- Down: Conv and downscaling
- Up: Upscaling and double Conv
- OutConv: convolution

These modules are organized in the following way:

1) Conv
2) Down
3) Down
4) Down
5) Down
6) Conv
7) Up
8) Up
9) Up
10) Up
11) OutConv

NAME	N_{out}	FUNCTION
INPUT	n	
ENC_CONV0	48	Convolution 3×3
ENC_CONV1	48	Convolution 3×3
POOL1	48	Maxpool 2×2
ENC_CONV2	48	Convolution 3×3
POOL2	48	Maxpool 2×2
ENC_CONV3	48	Convolution 3×3
POOL3	48	Maxpool 2×2
ENC_CONV4	48	Convolution 3×3
POOL4	48	Maxpool 2×2
ENC_CONV5	48	Convolution 3×3
POOL5	48	Maxpool 2×2
ENC_CONV6	48	Convolution 3×3
UPSAMPLE5	48	Upsample 2×2
CONCAT5	96	Concatenate output of POOL4
DEC_CONV5A	96	Convolution 3×3
DEC_CONV5B	96	Convolution 3×3
UPSAMPLE4	96	Upsample 2×2
CONCAT4	144	Concatenate output of POOL3
DEC_CONV4A	96	Convolution 3×3
DEC_CONV4B	96	Convolution 3×3
UPSAMPLE3	96	Upsample 2×2
CONCAT3	144	Concatenate output of POOL2
DEC_CONV3A	96	Convolution 3×3
DEC_CONV3B	96	Convolution 3×3
UPSAMPLE2	96	Upsample 2×2
CONCAT2	144	Concatenate output of POOL1
DEC_CONV2A	96	Convolution 3×3
DEC_CONV2B	96	Convolution 3×3
UPSAMPLE1	96	Upsample 2×2
CONCAT1	$96+n$	Concatenate INPUT
DEC_CONV1A	64	Convolution 3×3
DEC_CONV1B	32	Convolution 3×3
DEV_CONV1C	m	Convolution 3×3, linear act.

Fig. 2. UNet by Lehtinen et al., 2018

As in the case of the REDNet, skip connections are also added from a convolutional layer to its corresponding mirrored deconvolutional layer.

D. Parameter tuning

We trained the three networks on a random sub sample of 5000 training pairs. To simplify the procedure, we set up an automated pipeline that iterated over different values of the following parameters:

- Optimizer: SGD and ADAM
- Loss: Mean Absolute Error and Mean Squared Error
- Number of features in convolutional layers: 32 and 64
- Batch size during training: 10, 50 and 200

The following tables present the various results we obtained for each combination.

Optim	Loss	Features	Batch size	PSNR
SGD	MAE	32	10	12.67
SGD	MAE	32	50	9.30
SGD	MAE	32	200	9.29
SGD	MAE	64	10	12.67
SGD	MAE	64	50	10.04
SGD	MAE	64	200	8.90
SGD	MSE	32	10	12.68
SGD	MSE	32	50	9.28
SGD	MSE	32	200	9.27
SGD	MSE	64	10	12.68
SGD	MSE	64	50	10.03
SGD	MSE	64	200	8.89
ADAM	MAE	32	10	21.62
ADAM	MAE	32	50	21.52
ADAM	MAE	32	200	19.02
ADAM	MAE	64	10	21.65
ADAM	MAE	64	50	6.42
ADAM	MAE	64	200	19.40
ADAM	MSE	32	10	22.54
ADAM	MSE	32	50	21.54
ADAM	MSE	32	200	17.96
ADAM	MSE	64	10	21.71
ADAM	MSE	64	50	6.42
ADAM	MSE	64	200	20.85

TABLE I
BASELINE AUTOENCODER PARAMETER TUNING

Optim	Loss	Features	Batch size	PSNR
SGD	MAE	32	10	20.72
SGD	MAE	32	50	20.74
SGD	MAE	32	200	20.63
SGD	MAE	64	10	20.81
SGD	MAE	64	50	20.75
SGD	MAE	64	200	20.65
SGD	MSE	32	10	20.83
SGD	MSE	32	50	20.61
SGD	MSE	32	200	20.17
SGD	MSE	64	10	21.41
SGD	MSE	64	50	20.73
SGD	MSE	64	200	20.23
ADAM	MAE	32	10	23.93
ADAM	MAE	32	50	23.94
ADAM	MAE	32	200	23.57
ADAM	MAE	64	10	24.18
ADAM	MAE	64	50	24.05
ADAM	MAE	64	200	23.81
ADAM	MSE	32	10	24.90
ADAM	MSE	32	50	24.70
ADAM	MSE	32	200	24.36
ADAM	MSE	64	10	24.77
ADAM	MSE	64	50	24.75
ADAM	MSE	64	200	24.40

TABLE II
REDNET PARAMETER TUNING

Optim	Loss	Features	Batch size	PSNR
SGD	MAE	32	10	23.96
SGD	MAE	32	50	21.22
SGD	MAE	32	200	17.72
SGD	MAE	64	10	23.99
SGD	MAE	64	50	21.43
SGD	MAE	64	200	17.92
SGD	MSE	32	10	22.37
SGD	MSE	32	50	18.75
SGD	MSE	32	200	16.95
SGD	MSE	64	10	23.00
SGD	MSE	64	50	19.38
SGD	MSE	64	200	17.06
ADAM	MAE	32	10	24.39
ADAM	MAE	32	50	24.69
ADAM	MAE	32	200	24.24
ADAM	MAE	64	10	20.08
ADAM	MAE	64	50	24.59
ADAM	MAE	64	200	24.18
ADAM	MSE	32	10	24.83
ADAM	MSE	32	50	25.04
ADAM	MSE	32	200	24.19
ADAM	MSE	64	10	17.94
ADAM	MSE	64	50	24.98
ADAM	MSE	64	200	24.33

TABLE III
UNET PARAMETER TUNING

III. RESULTS AND DISCUSSION

As can be seen by the previous tables, the best PSNR was obtained for the UNet with Adam Optimizer, MSE Loss, 32 features and batch size of 50. We can observe how the application of residuals to the architecture brings great benefits to the learning and to the final score.

We proceeded by training this model with the whole data, obtaining a PSNR of **25.47**. Various data augmentation techniques were randomly applied (vertical flip, horizontal flip, rotation, crop); however, this did not lead to any improvement in the PSNR.

REFERENCES

[1] J. Lehtinen, J. Munkberg, J. Hasselgren, *et al.*, "Noise2noise: Learning image restoration without clean data," *35th International Conference on Machine Learning*, 2018.

[2] X. Mao, C. Shen, and Y. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," *Advances in Neural Information Processing Systems*, 2016.