Training a Noise2Noise Denoising Autoencoder with PyTorch

Deep Learning MiniProject 1

Francesco Salvi **Lukas Van Den Heuvel** **Jiří Lhotka**

EPFL, Lausanne, Switzerland
{cornelius.vandenheuvel,francesco.salvi,jiri.lhotka}@epfl.ch

Abstract

In this report, we explore different architectures for training a denoising autoencoder using noisy images as both input and target. We present TNet, a simple architecture which outperforms UNet (Lehtinen et al., 2018) — the original Noise2Noise architecture — on our dataset, while using 20x fewer parameters. We explore different regularization and optimization techniques and present their results. The final TNet achieves a PSNR of (25.32 ± 0.01) dB while using only $43\,064$ parameters.

1 Introduction

The task of image denoising is defined as removing unwanted disturbance from an image. Beginning in the 2010's, denoising autoencoders have been successfully used for this task (Goodfellow et al., 2016). Such autoencoders learnt the mapping between noisy and clean data, but clean data is not always available. Recently, Lehtinen et al. (2018) have shown that it is also possible to train a denoising autoencoder on noisy data, provided that we have two representations of each image where each one contains different noise. Lehtinen et al. (2018) call this idea Noise2Noise and prove the success on this task of an architecture based on UNet (Ronneberger et al., 2015).

In this work, we implement several of our own versions of Noise2Noise, considering different design choices and evaluating their impact on performance. Finally, we present TNet, short for Thin-Network, an architecture that outperforms UNet on 3x32x32 images while using substantially fewer parameters and consequently being much easier to train.

2 Methods

In this section, we discuss different suitable techniques, their usage during our experimentation, and their effect on our results.

2.1 Baseline (BaseNet)

As a baseline, we have adapted a simple model presented in the lectures (Fleuret, 2022), used for MNIST Autoencoding. This simple model, that we called BaseNet, interleaves 6 layers of 2D convolutions with ReLU activation units, embedding images in a 8-dimensional vector latent space.

2.2 TNet

2.2.1 Residual Connections

Observing the initial results of BaseNet aroused a suspicion that for a network this deep, a simple feed-forward structure will make it almost impossible for the network to recover full image information from the latent representation. For this reason, inspired by Lehtinen et al. (2018), we added residual connections, which enable the model to learn the difference between the input and the target.

2.2.2 MaxPooling

Additionally, we added Max Pooling layers, which are ubiquitous in Computer Vision Deep Learning (Goodfellow et al., 2016) and which were also used in UNet (Ronneberger et al., 2015). Max Pooling Layers enable the model to reduce the dimensionality of the representation while preserving the most prominent (and therefore often the most important) features in a given region.

2.2.3 ADAM Optimizer

Thirdly, we directly opted for using the ADAM optimizer with the default parameters. The ADAM optimizer has been shown to make gradient descent more stable and more performant, while having very few downsides (Kingma and Ba, 2015). Indeed, it has also been used for training UNet in Lehtinen et al. (2018).

2.2.4 TNet

The final architecture is presented in Figure 1.

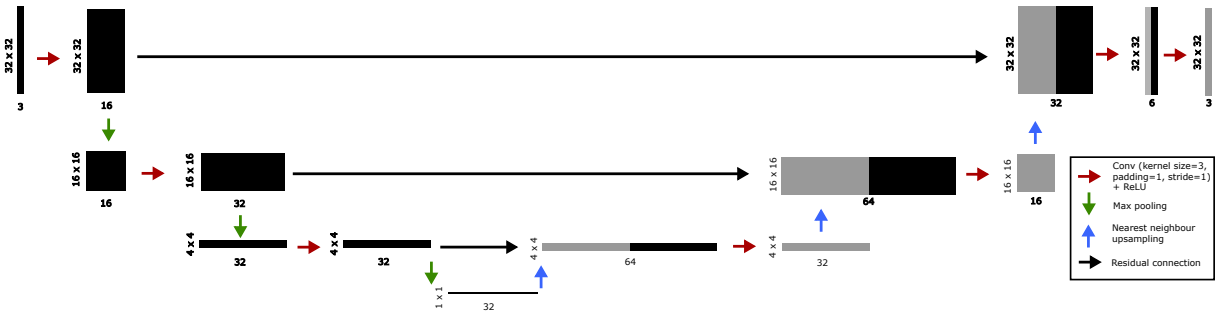

Figure 1: High-level view of TNet.

LR $\times 10^{-3}$	0.1	**0.5**	0.9
PSNR (dB)	25.31 ± 0.01	$\mathbf{25.33 \pm 0.02}$	25.32 ± 0.02
Loss(L2)	0.70896 ± 0.00035	$\mathbf{0.70859 \pm 0.00056}$	0.70869 ± 0.00083

Table 1: Learning Rate Comparison (average $\pm$ standard deviation after 5 runs)

2.3 Hyperparameter Search & Experiments

In this section, we will describe hyperparameter search as well as other experiments with the architecture that we did to tune TNet.

2.3.1 Learning Rate Grid Search

The learning rate is often the most important hyperparameter (Ng, 2021). While ADAM dynamically changes the learning rate — it can tune the learning rate up to a factor of 10 (assuming $\beta_1 = 0.9$, the default) — the influence of the learning rate on training performance remains high. Our initial exploration suggested that the optimal learning rate is around 0.5×10^{-3}. We therefore performed a grid search with 3 values on a linear scale around 0.5×10^{-3}, training TNet on 50,000 noisy image pairs for 50 epochs with L2 loss. To quantify model performance, we used both the mean peak signal-to-noise ratio (PSNR) over 1000 validation images and the final training loss after 50 epochs. Table 1 shows the results, which show that the difference in performance and loss between the learning rate is small. Still, 5×10^{-4} is the optimal learning rate in terms of both PSNR and loss, so we used this value in further TNet experiments below.

2.3.2 Batchnorm and L1 Loss

In Table 2, we present a comparison between the tested models. We compare BaseNet and TNet to the original UNet, for which we use an implementation by Tsinghua University (2020), while modifying the kernel sizes to fit our data without changing the architecture. For TNet, we tested two additional training conditions, Batch Normalization and L1 loss, which both did not improve the model's performance. All models were trained for 50 epochs with 5 random initializations. We used the optimal learning rate of 5×10^{-4} for all models except for UNet, where we adopted 10^{-3} as used in Lehtinen et al. (2018).

In addition to these trials, we also experimented with dropout, the LHR loss proposed in Lehtinen et al. (2018), and with increasing the number of dimensions in TNet's embedding space. None of these modifications improved the performance presented in Table 2.

Model	#P[1]	Res[2]	Loss	BN[3]	PSNR[4]
BaseNet	167.0	No	L2	No	6.95 ± 0.32
UNet	991.0	Yes	L2	No	25.24 ± 0.08
TNet	**43.0**	**Yes**	**L2**	**No**	$\mathbf{25.32 \pm 0.01}$
TNet+L1	43.0	Yes	L1	No	25.29 ± 0.01
TNet+BN	43.5	Yes	L2	Yes	24.45 ± 0.05

Table 2: Comparison of models
1 – # of parameters (thousands), 2 – Residual connections, 3 – BatchNorm, 4 – average $\pm$ std after 5 runs

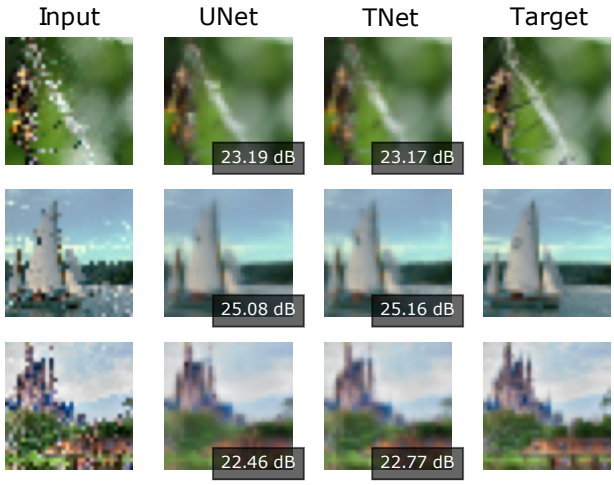

Figure 2: Examples of denoised predictions.

The results show that our vanilla TNet performs best, achieving comparable results to UNet even though its number of parameters is more than 20× less. Surprisingly, adding batch normalization did not improve the model's performance. Indeed, the advantage of batch normalization for deep neural networks is still under debate (Yang et al., 2019), and it does not seem beneficial for TNet.

3 Smoothed Gradient Visualization

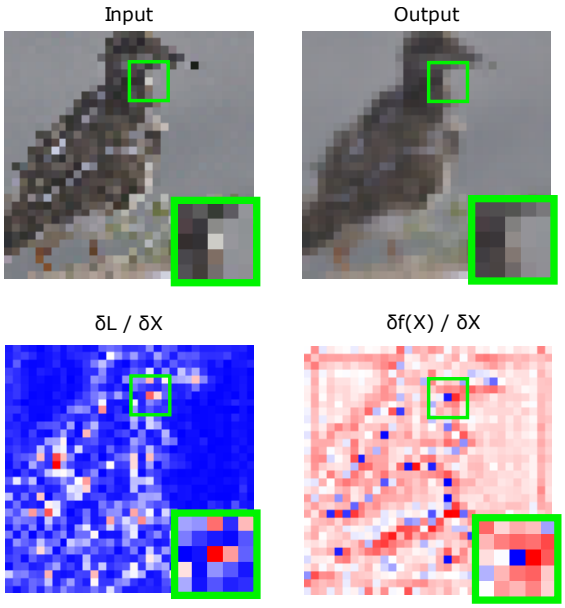

Figure 3: Example of input, output and smoothed gradients.

What does the model do to denoise? To answer this question, we adopted the SmoothGrad method proposed by (Smilkov et al., 2017). SmoothGrad was originally introduced to visualize which regions in an input image are influential for a decision made by a classifier network. In brief, the gradient of the network output with respect to the input perturbed by random noise is calculated for several perturbed inputs, and then averaged to enhance visualisation. Even though the output of TNet is an image and not a classification, we adopt the same method to visualize which pixels in the input, when perturbed, drastically change (1) the loss and (2) the output.

Figure 3 shows an example of a smoothed gradient of (1) the loss with respect to the input $(\partial L(f(X))/\partial X_{i,j})$ and of (2) the output with respect to the input $(\partial f(X)/\partial X_{i,j})$. Naturally, the network mainly impacts the intensity of the noisy pixels in the input, such as the one in the center of the green square. Accordingly, the gradient of the loss w.r.t. the input is high for noisy pixels. On the other hand, the gradient of the output w.r.t. the input is low for noisy pixels, suggesting that they have very little influence on the denoised image produced by the network. Instead, the gradient at pixels *surrounding* a noisy pixel is high. In conclusion, the network identifies which are the most noisy pixels by looking at their surroundings, and it replaces their intensity to match the surroundings better. When doing so, the network is aware of the difference between object and background: of all pixels surrounded by the green square in Figure 3, the background pixels (at the right-hand side) have more influence on the output than the pixels belonging to the object (the bird). This content-awareness is what makes the use of deep-learning-based denoising more powerful than classical operations like mean or gaussian filtering.

4 Conclusion

In this project, we investigated various architectures for denoising in a a Noise2Noise paradigm. We presented TNet, a feed-forward network using only 43k parameters, which leverages residual connections, pooling, convolutions and upsampling to achieve performance comparable to the much deeper UNet, reaching (25.32 ± 0.01) dB on small RGB images of $3 \times 32 \times 32$px.

We tuned TNet by doing a grid search on its hyperparameters, employing the ADAM optimizer, and we experimented with tweaking the loss and adding batch normalization. We further explored what happens inside TNet by visualizing the gradients w.r.t the loss and the input, which led to the insight that the model suppresses noisy pixels by looking at the pixels surrounding them in a content-aware manner.

References

François Fleuret. 2022. Autoencoders lecture notes.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*.

Andrew Ng. 2021. Improving deep neural networks: Hyperparameter tuning, regularization and optimization.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. SmoothGrad: removing noise by adding noise. Technical report, arXiv.

The Future Lab, Tsinghua University. 2020. UNet pytorch implementation.

Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. 2019. A Mean Field Theory of Batch Normalization. Technical Report arXiv:1902.08129, arXiv.