EE-559 Mini-project 1: Implement Noise2Noise model with PyTorch

Hao Zhao, Xufeng Gao, Qiming Sun

Abstract

We consider the problem of investigating the effect of blind denoising algorithm, Noise2Noise. Prior image restoration work typically requires to access clean reference images when learning, making them inefficient and even unavailable in some practical settings. However, by applying basic statistical reasoning, the Noise2Noise study first made it possible to learn to restore images by only looking at corrupted examples. In this project, we showed that a single model mainly composed of convolutional layers and transpose convolutional layers can achieve notable noise removal effects, even without explicit image priors and noise information. Using a baseline model without adjusting the training pipeline, we obtained 25.35 db PSNR on unseen noisy images. The highest PSNR, 25.84 dB on the validation dataset, is obtained after adding data augmentation and tuning the loss function.

1. Introduction and Related Works

We would often like to reconstruct a signal from high-dimensional measurements that are corrupted, undersampled, or otherwise noisy. For example, when taking videos or pictures with ultra-fast frame rates at very low-illumination, each individual feature can become quite noisy. Fortunately, the objects being studied are often very well structured and the values of different features are highly correlated. To put it simply, if the latent dimension of the space of signals under study is much lower than the dimension of the measurement, it may be possible to implicitly learn that structure, denoise the measurements, and recover the signal without any prior knowledge of the signal or the noise [1].

In [2], Jain et al. first apply CNNs for the denoised task. They introduced a basic setup that is still widely used by many methods today including Noise2Noise: the model learns to minimize a loss for regression problem calculated between its prediction and clean ground truth data. But a key weakness of this type of denoising is the posterior may be non-deterministic, possibly multi-modal. In [7], Zhang et al. achieve SOTA performance by introducing a very deep CNN architecture with residual learning and batch normalization. At about the same time, Mao et al. introduce a complementary very deep encoder-decoder architecture [5] for the denoising task.

Whenever ground truth images are not provided, above methods cannot be trained and are therefore useless. In [3], Noise2Noise proposed by Lehtinen et al. attempts to learn a mapping between pairs of independent degraded versions of the same clean image, i.e. $(s + n, s + n')$. Each image pair incorporates the same signal s, but independently drawn noise n and n'. Naturally, a model fails to learn to predict one noisy target from another one. However, networks trained on this impossible training task can produce results that converge to the same deterministic predictions as traditionally trained networks.

2. Theoretical Background

Typical regressor learning task for a set of input-target pairs (x_i, y_i) can be represented as following,

$$\underset{\theta}{argmin}\, \mathbb{E}_{(x,y)} L(f_\theta(x), y) \tag{1}$$

where the network function $f_\theta(x)$ is parameterized by θ. The optimization problem in (1) can be decomposed to several minimization problem based on every training sample. After some basic manipulations, we can see that (1) is equivalent to

$$\underset{\theta}{argmin}\, \mathbb{E}_x\{\mathbb{E}_{y|x}\{L(f_\theta(x), y)\}\} \tag{2}$$

In this way, we can incorporate the internal expectation of equation (2) into the neural network and force it to update model parameters toward the direction of reducing the expectation loss. For equation (2), no matter what the underlying distribution of $p(y|x)$ looks like, the optimal θ is a group of parameters that always make the estimate equal to the expectation of targets $\mathbb{E}_{y|x}\{\hat{y}|\hat{x}\}$ if we use L_2 loss as our objective function. Similarly, the estimate will converge to the median of observations for L_1 loss. Thus, we can learn a denoiser by minimizing the empirical risk

$$\underset{\theta}{argmin} \sum_i L(f_\theta(\hat{x}_i), \hat{y}_i) \tag{3}$$

where both inputs and targets are corrupted images.

3. Experimental Setup

3.1. Backbone networks

For the main experiments, we use the U-network [6], which was the default network structure in the original paper, to implement blind denosing. In addition, we tested the algorithm on another model architecture: REDNet30 [5].

3.2. Hyperparameters and model selection

Model selection significantly affects performance. Different tradition classification problem, we cannot use standard training-domain validation for selecting hyperparameters because all targets in the training set are noisy. Thus we implemented model selection on a subset of testing set. We conduct a random search of 20 trials over a joint distribution of all hyperparameters to train the base model. More specifically, learning rate is selected from $10^{Uniform(-5,-3.5)}$ and batch size is selected from $[2^1, 2^2, 2^3, 2^4, 2^5]$. According to the result of random search, we finally used $LR = 0.0005$ and $BS = 4$. For the results in Mini-project 1, we used three seeds $\{0, 1, 2\}$.

For simplicity, we only focus on the selection of optimizers and use their default hyperparameters. It will have the risk of catastrophic forgetting to use SGD during training, so it's a better choice to use Adam as the optimizer. This per-parameter learning rate method provides heuristic approach without requiring expensive work in tuning hyperparameters. However, in [4], Loshchilov et al. pointed out that the way weight decay is implemented in Adam in every library seems to be wrong and they proposed a slightly different weight decay to fix it. The new optimizer is named as AdamW. It has been shown in some study that AdamW is a faster way to train neural nets than Adam and does not harm the performance.

4. Results

Data augmentation improves the performance of the model. Different from object recognition problem, we should be careful with using data augmentation methods. In case of introducing extra loss, we should apply the same augmentation steps onto both inputs and targets. Besides, some augmentation techniques like posterize and solarize are difficult to control their influence on the loss, so we should prevent from using these augmentation methods. In our project, we explored two kinds of augmentation techniques: flip and rotation. For an input-target image pair, they have 50% probability to implement horizontal flip first, and then they will have another 50% probability to flip vertically. In addition to flip, a pair of images has the same probability to simultaneously rotate $0°$, $90°$, $180°$, or $270°$. The first two columns of Table 1 lists the results on testing data, in which only single operation has positive effect on the denoising effect. The reason why using both techniques failed to improve the performance is that both flip and rotation are similar augmentation methods. In some occasions, images flip first and then implement rotation after which images return to the original state. Thus, these two augmentation methods have some antagonization effect.

Rotate	Flip	Swap	PSNR (dB)
✗	✗	✗	25.35
✓	✗	✗	25.41
✗	✓	✗	25.42
✓	✓	✗	25.39
✓	✓	✓	25.54

Table 1. Effectiveness of data augmentation techniques and input-target image pairs swapping. PSNR (dB) metric of trained models is reported on testing data.

L_1	L_2	L_{bid}	PSNR (dB)
✓	✗	✗	25.46
✗	✓	✗	25.47
✓	✗	✓	25.58
✗	✓	✓	25.60

Table 2. Effectiveness of loss function combinations. PSNR (dB) metric of trained models is reported on testing data.

Introducing a swap loss significantly improves the performance of the model. Previous section has stated the unexpected benefits of L_1 and L_2 loss. Because other loss variants do not have such elegant conclusions, we mainly focused on L_1 and L_2 loss in our project. Table 2 shows the results on validation data with different loss functions. We can see that L_2 loss achieved similar PSNR performance with L_1 loss.

Another significant observation is that for the Noise2Noise scenario both inputs and targets are noisy images. Naturally, we can swap inputs and targets i.e. feeding the target image into the network and comparing its output with the original input image. Because the underlying clean signal s is the same for each image pair, only with independent noise n and n', we can obtain a more robust denoiser by learning from a bidirectional loss without worrying about harming its performance. For combining loss contribution from both image pairs, we design a bidirectional loss as following,

$$\mathcal{L}_{bid} = \sum_{i=1}^{N} \frac{1}{2} L(f_\theta(\hat{x}^i), \hat{y}^i) + \frac{1}{2} L(f_\theta(\hat{y}^i), \hat{x}^i) \qquad (4)$$

where L can be L_1 or L_2 loss. In table 2, we can see that denoisers using the new bidirectional version of loss gained better performance than the ones using the original loss. This result makes sense from the perspective of data augmentation.

In our project, we further evaluated optimization performance of three optimizers: SGD, Adam, and AdamW.

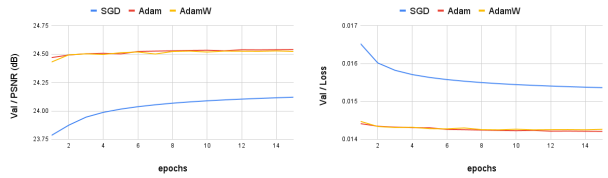

Figure 1. Effectiveness of three optimizers on learning blind denoising task. Left: calculated PSNR on validation splits. Right: observed L_2 loss on validation splits.

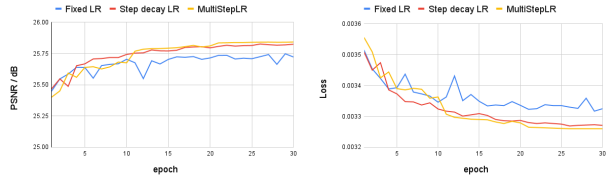

Figure 2. Testing on the provided validation dataset. Left: calculated PSNR. Right: observed L_2 loss.

We kept the other hyperparameters the same and only use the default optimizer setup. Considering the computational consumption of time, we only ran 15 epochs and increased the learning rate ($LR = 0.001$) for compensation. The results of training process using three optimizers are reported in Figure 1. We can see that the leaning efficiency of Adam and its variant is notably better than SGD in terms of PSNR and loss. However, we did not find the tendency pointed out by [4] that AdamW is a better optimizer by adjusting the weight decay term. As the result show, the model trained using AdamW achieved 24.52 dB PSNR on the training split and 0.01426 loss, while Adam obtained 24.54 dB and 0.01422 loss. In terms of consumption time, Adam spent 1h36m46s completing 15 full epochs training on NVIDIA RTX 3090 Ti and Adam spent 1h38m21s on the same machine. Thus, it is a better choice to use Adam as the optimizer in our task.

An appropriate learning rate scheduler matters. When training deep neural networks, it is often useful to reduce LR as the training progresses. This can be done using predefined schedules. In this project, we compared three LR strategies: (a) fixed LR, (b) step decay LR and (c) MultiStepLR. The initial LR equals to 5e-4 for three strategies. In the step decay LR strategy, we update LR according to the equation $(1 + \gamma \times iter\_ratio)^{-p}$ where $\gamma = 10$ and $p = 0.75$. In MultiStepLR, we set up two milestones $M = [10, 20]$ by observing the smoothed loss curve of fixed LR strategy. We update LR by $LR(t) = \gamma \times LR(t-1), if\ t \in M$. $\gamma = 0.2$ in the MultiStepLR case. In Figure2, we can see that pre-defined dynamic LR strategies outperform the fixed LR strategy on both PSNR metric and loss. Also, MultiStepLR achieved better performance than step decay LR because it combined some prior knowledge from fixed LR. Finally, we selectd step decay LR because its denoising performance and prior-free property.

Deeper model is not always a better choice. To better understand the role of backbone networks for blind denoising work, we selectd REDNet30 as the counterpart. We chose the best hyperparameter combination according to previous experiments (LR=0.0005, BS=4, bidirectional L_2 loss, MultiStepLR, and Adam optimizer) and ran 30 full epochs. Accuracy of U-net and REDNet30 is reported on the testing dataset in table 3. Naturally, having less layers, the computational cost of U-net is much less than REDNet30. U-net saved 73 seconds for each epoch training in average. However, REDNet30 did not outperform U-net in the blind denoising task although it has a deeper network. With the same experimental settings, U-net achieved 25.84 dB PSNR in the testing data, while REDNet30 only got 25.69 dB PSNR. The reasons of failure for REDNet30 come from many aspects: (a) the provided noisy dataset has limited noisy types (downsampled, pixlated images). (b) it needs more training samples and epochs to tune its parameters.

Backbone networks	PSNR (dB)	Average consumption time (s) / epoch
U-net	25.83	392.3
REDNet30	25.69	465.4

Table 3. Effectiveness of loss function combinations. PSNR (dB) metric of trained models is reported on testing data.

Noise2Noise achieves outstanding image denoising performance on the testing dataset. Summarizing the previous work, we obtained a well trained model that achieved 25.84 dB on unseen testing dataset. In the appendices section, We listed some denoised images restored by the model. Obviously, our trained Noise2Noise model indeed learns to denoise images and does a great job.

5. Conclusion

In this work, we consider to investigate the ability of Noise2Noise model on the blind denoising task. By means of experiments on different settings, we prove the efficacy of our constructed model. With this model, we are able to learn to restore noisy images without access to clean images, which is often unavailable in practical scenarios. Naturally, we can extend the denoising image application to denoise noisy signals in a broad sense. We believe the ability to simultaneously denoise multiple noise types is a highly intractable problem. And we are curious to find out the performance of Noise2Noise model when coming across images with noise in the wild.

References

[1] J. Batson and L. Royer. Noise2self: Blind denoising by self-supervision. In *International Conference on Machine Learning*, pages 524–533. PMLR, 2019.

[2] V. Jain and S. Seung. Natural image denoising with convolutional networks. *Advances in neural information processing systems*, 21, 2008.

[3] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018.

[4] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[5] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *Advances in neural information processing systems*, 29, 2016.

[6] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[7] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

Appendices

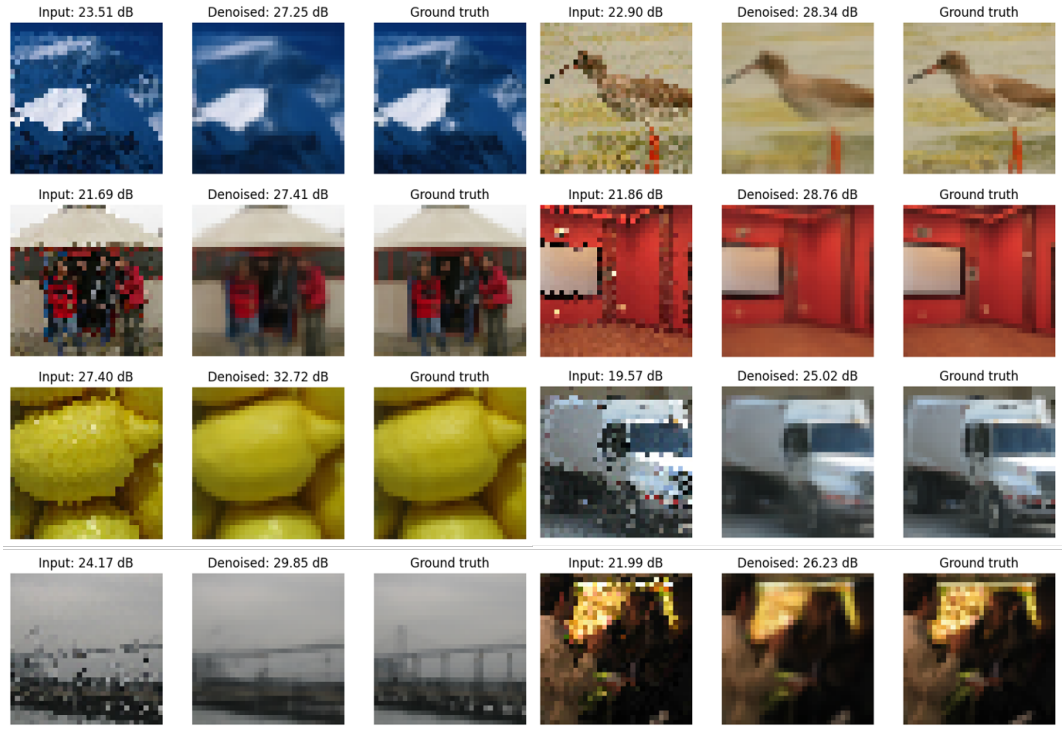

Figure 1. Performance of the trained Noise2Noise model on unseen testing noisy images.