

Few-Shot Meta-Learning for Image Denoising

Jessica Tawade and Rahul Shiv

Abstract—Image denoising is a critical component of any image processing pipeline. While there has been significant progress in this area of research, state-of-the-art denoising techniques still suffer from significant drawbacks. Deep learning approaches often require large quantities of data for training and do not generalize well to unseen noise distributions. Non-learning methods, on the other hand, tend to suffer from prohibitively long runtimes. In this paper, we address these issues by developing a "Meta-Denoiser", a few-shot, model-agnostic meta-learning (MAML) framework for image denoising. The main objective of the Meta-Denoiser is to find an ideal set of weights to initialize a denoising convolutional neural network (DnCNN). With the proper weight initialization, a DnCNN can adapt to unseen noise distributions with very few training examples. We tested our Meta-Denoiser on both synthetic and real noisy images and find that the Meta-Denoiser performs image denoising as well as, if not better than, non-learning image denoising algorithms, and does so in far less time. The source code for our project can be found here: <https://github.com/jt658/EE367-Final-Project>

Index Terms—Image denoising, meta-Learning, convolutional neural networks

1 INTRODUCTION

Image denoising algorithms generally fall into one of two categories: learning-based and non-learning based methods. Most learning-based methods, such as the DnCNN [1] result in de-noised images of much higher quality than non-learning based approaches like non-local means [2] and BM3D [3]. Moreover, after training, the DnCNN tends to denoise images faster than non-learning-based methods do. However, most of these learning-based approaches require a large number of clean and noisy image pairs [1], which can be difficult to obtain for certain applications. Many applications may only have access to a few clean and noisy image pairs, which could lead to overfitting in learning-based approaches.

This challenge is usually addressed by constructing large training datasets with clean and noisy image pairs where the noisy image is synthetically generated from the clean one. Specifically, to generate noisy images, samples are drawn from an assumed noise distribution. These samples are then applied to a clean image to create the noisy counterpart [4]. However, training with synthetically generated noisy images does not generalize well in practice due to a lack of fine-tuning during test-time [5], [6]. Furthermore, many of these learning-based models are also trained to denoise for a specific type of noise and a fixed amount of noise variance. These models, then, are not very generalizable and are unsuitable for smaller training sets.

To overcome these issues, we propose the application of meta-learning algorithms to learn few-shot image denoising with various noise distributions. We believe the combination of meta-learning and few-shot learning is a good model for the image denoising challenges described above because the model will quickly learn how to denoise images from a few synthetically generated noisy images (using various noise distributions) and use fine-tuning to efficiently adapt to denoising of a small set of real noisy images.

The key meta-learning framework we aim to explore is known as model-agnostic meta-learning (MAML) [7].

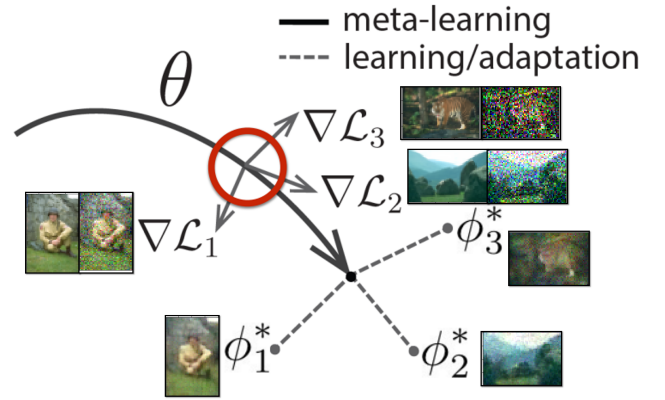


Fig. 1. MetaDenoiser is a deep learning model that finds an optimal set of parameters for initializing a DnCNN. It does so by training the DnCNN on several tasks and aggregating the final loss of the DnCNN after it has trained on said tasks.

The operating principle behind MAML is to train a neural network on a set of tasks which, in the context of this project, are a set of noise distributions. After the network has been trained on separate tasks, the MAML architecture aggregates the loss of the network across different tasks and backpropagates on this loss to generate a new set of weights [7]. This new set of weights is used to initialize the neural network for the next round of training [7]. As this process repeats, the MAML architecture finds increasingly better sets of weights for initializing the network. After all rounds of training are complete, the network's starting weights are ideally structured so that, with just a few training examples from an unseen task, the network can easily converge to an optimal mapping between noisy and clean images [7].

2 RELATED WORK

2.1 Traditional Denoising Approaches

Non-learning-based denoising methods typically rely on some form of convolutional filtering to denoise images.

Such methods often assume a Gaussian or Poisson noise model and benefit from requiring little to no training data to perform well. However, non-learning-based methods often have very long runtimes and are outperformed by learning-based approaches. This paper uses non-learning-based approaches such as non-local means (NLM) [2], bilateral filtering (BF) [8], and block-matching and 3-D filtering (BM3D) [3] as a baseline against which we measure the performance of our Meta-Denoiser.

2.2 Learned Denoising

Learning-based approaches have been quite popular for image denoising in the past couple years. These algorithms have been constructed using architectures like auto-encoders, convolutional neural networks, and multi-layer perceptrons as well. Our Meta-Denoiser relies on the DnCNN [1], in combination with meta-learning algorithms, to denoise images.

Instead of directly trying to predict the denoised output of a noisy input image, the DnCNN uses residual learning to predict the residual/noise distribution within the image. The learned noise is then subtracted from the noisy image and the result is considered to be the denoised image which is compared against the clean image [1].

2.3 Meta-Learning

The main goal of meta-learning is to train a model to learn how to quickly solve a task after training on a set of tasks all drawn from a similar distribution [7]. Most meta-learning algorithms are broken down into two parts: meta-training and meta-testing. During each iteration of meta-training, the algorithm samples a task from the distribution of training tasks and learns how to perform this task. An example of one such task would be to denoise an image that has been perturbed by a Gaussian noise distribution of a particular variance. The algorithm then uses the knowledge acquired from multiple tasks during meta-training to quickly adapt to a new unseen task during meta-testing. The meta-test tasks are sampled from a similar distribution to the meta-train tasks. For few-shot learning, each task only uses a small amount of data to train and adapt to new tasks. We employ a meta-learning framework in this paper to enable a DnCNN to quickly learn how to denoise images with unseen noise distributions.

3 PROPOSED METHOD

In order to apply meta-learning in the few-shot context for image denoising, we will be combining the optimization-based meta-learning algorithm known as MAML with the denoising convolutional neural network architecture known as the DnCNN. The Meta-Denoiser operates in two parts. The first part involves meta-training on a set of tasks that include synthetically generated noisy images. The DnCNN will learn to denoise the synthetically generated noisy images for each task and MAML will learn to use the aggregated loss from all the tasks to learn a good weight initialization for the DnCNN model. The second part then uses this initialization to perform meta-testing. During meta-testing, the DnCNN is given a new, unseen task. In our setting this

will be a denoising task that operates on a noise distribution or level that was unseen during meta-training. These two parts come together to help the DnCNN model learn how to be adaptable to denoise images affected by various types and levels of noise [5].

3.1 Problem Formulation and Tasks

Let's define our clean image as a flattened vector $\mathbf{y} \in \mathbb{R}^D$ and let's define our synthetically generated noisy image as a flattened vector $\mathbf{x} \in \mathbb{R}^D$ [5]. Rather than train our DnCNN on a specific noise distribution, or on several noise distributions at once, we instead partition our training dataset into a set of tasks, where each task represents a distinct noise distribution. We generate the dataset for each task by sampling a task $\mathcal{T}_i$ from a distribution of tasks $p(\mathcal{T})$. Each sampled task $\mathcal{T}_i$ will represent a specific type of noise distribution (represented as $h_{\mathcal{T}_i}$) and an associated set of parameters for that noise distribution. We generate our dataset for each task as follows $\{\mathbf{x}_i = h_{\mathcal{T}_i}(\mathbf{y}_i), \mathbf{y}_i | 1 \leq i \leq k\}$ [5]. k represents the number of samples we use for each task and we refer to each of these few-shot tasks as k-shot tasks.

In this project, we use three types of Gaussian noise and three types of Poisson noise, yielding a total of six tasks over which we will train the DnCNN. More information about these tasks can be found in Section 4.1.1.

The dataset used during meta-test time is separate from the data used during meta-training. The meta-test data is also split into 2 parts: the fine-tune set $\{\mathbf{X}_{ft}, \mathbf{Y}_{ft}\}$ (which contain k pairs of noisy and clean images and is used to allow the model to quickly learn a new task) and the evaluation set $\{\mathbf{X}_{eval}, \mathbf{Y}_{eval}\}$ which is used to determine the model's ability to generalize to new tasks.

3.2 Meta-Denoiser: MAML + DnCNN

The DnCNN maps noisy images to a residual noise distribution underlying those images [1]. Specifically, the DnCNN is a function f_θ that is parameterized by weights θ .

The objective of meta-learning is to compute a set of weights θ which we can use to initialize our network [7]. To compute θ , we sample a task $\mathcal{T}_i$ from a task distribution $p(\mathcal{T})$. We then train our DnCNN on each task through gradient descent with learning rate α . When it is trained on task $\mathcal{T}_i$, the DnCNN's weights are denoted as θ'_i . The first update to our network under task $\mathcal{T}_i$ is:

$$\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

We can update the weights θ'_i for each task for as many inner iterations as we would like.

After computing θ'_i for all i , MAML aggregates these losses $\mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ into a single loss term and uses the aggregated loss to update the weight initialization θ as follows (see Figure ??):

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$$

β is another learning rate and we can update the weights θ for as many outer iterations as we would like.

After this meta-training process is complete, we can use the initialization θ to fine-tune and evaluate the model using $\{\mathbf{X}_{ft}, \mathbf{Y}_{ft}\}$ and $\{\mathbf{X}_{eval}, \mathbf{Y}_{eval}\}$.

The complete algorithm can be found below.

Algorithm 1 Meta-Denoiser [5], [7]

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β : step size hyperparameters

```

1: Randomly initialize  $\theta$ 
2: for outerIteration=1, 2,  $\dots$ , m do
3:   for task=1, 2,  $\dots$ , t do
4:     Sample noise distribution  $\mathcal{T}_i \sim p(\mathcal{T})$ 
5:     Sample k clean images  $\{\mathbf{y}_i | 1 \leq i \leq k\}$ 
6:     Apply sampled noise  $\mathbf{x}_i = \{h_{\mathcal{T}_i}(\mathbf{y}_i) | 1 \leq i \leq k\}$ 
7:     Evaluate  $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$  w.r.t k generated image pairs
8:     Update task specific parameters with gradient
       descent:  $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$ 
9:   end for
10:  Update  $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ 
11: end for
12: Fine tune  $f_{\theta}(\mathbf{x})$  using  $\{\mathbf{X}_{ft}, \mathbf{Y}_{ft}\}$ 
13: Evaluate  $f_{\theta}(\mathbf{x})$  using  $\{\mathbf{X}_{eval}, \mathbf{Y}_{eval}\}$ 

```

3.3 Model Architecture Details and Hyperparameters

We run out meta-training for $m = 2000$ outer iterations and use $t = 6$ tasks. We update the task specific parameters (line 8 in Algorithm 1) for 20 inner iterations. Our learning rates are set as follows: $\alpha = 10^{-3}$ and $\beta = 10^{-4}$. Each task uses $k = 10$ image patches and we use mean square error loss to evaluate the model.

Our inner model $f_{\theta}(\mathbf{x})$ follows a standard DnCNN [1] structure with 7 convolutional layers. The inner 5 layers also employ batch normalization and a ReLU activation function.

4 EXPERIMENTAL RESULTS

4.1 Generalizing to Unseen Noise Distributions

4.1.1 Task Generation and Experiment Details

For our first set of experiments, we meta-trained and meta-validated our Meta-Denoiser model on synthetically-generated noisy images from the BSDS300 [9] dataset in order to perform few-shot image denoising on synthetically-generated noisy images from the BSDS500 [10] dataset using unseen noise distributions during meta-test time. We used the set of 200 training images from the BSDS300 [9] to perform meta-training and the set of 100 validation images from the BSDS300 to perform meta-validation. The images in BSDS300 and BSDS500 are of size $481 \times 321 \times 3$, however the DnCNN within our Meta-Denoiser model meta-trained and meta-validated on patches of size $32 \times 32 \times 3$.

During meta-training, the Meta-Denoiser model learns on 6 different few-shot tasks. For each task, we randomly sample 10 images patches from the BSDS300 training set, synthetically apply noise to these image patches, and then the Meta-Denoiser model learns to denoise these image patches as described in Section 3. For each of the first 3 meta-training tasks, we add zero-mean Gaussian noise to the 10 sampled image patches in order to create the synthetic noisy image patches. Each task, has a different standard

deviation for the Gaussian noise. Therefore, our synthetic noisy image patches for each Gaussian task are generated as follows $\mathbf{x} = h_{\tau=\mathcal{N}(\mu, \sigma)}(\mathbf{y})$ [5], where $\mu = 0$ for each task and the following standard deviations $\sigma \in [0.05, 0.1, 0.2]$ are used for each task. For the other 3 meta-training tasks we apply signal-dependent Poisson distributed noise as follows: $\mathbf{x} = h_{\tau=\mathcal{P}(\beta)}(\mathbf{y})$ where $\mathcal{P}$ represents the Poisson distribution and β represents the scaling factor [5]. To be more specific $\mathbf{x} = \mathcal{P}(\beta \times \mathbf{y})/\beta$. For each task, we used the following 3 scaling values $\beta \in [1, 5, 10]$.

We used the same 6 tasks during meta-validation in order to track the performance of the Meta-Denoiser model every 20 outer iterations. During meta-test time, we used the model weights which resulted in the highest meta-validation PSNR. The following 6 tasks were evaluated during meta-test time:

- 3 Gaussian denoising tasks with $\mu = 0$ and $\sigma \in [0.1, 0.15, 0.3]$
- 3 Poisson denoising tasks with $\beta \in [5, 7, 15]$

We wanted to include a mix of seen and unseen noise parameters during meta-test time in order to evaluate the Meta-Denoiser model's ability to generalize to various types of denoising tasks. We use the set of 200 testing images from the BSDS500 [10] dataset to perform meta-testing. The fine-tuning steps of the meta-testing stage are performed on 10 image patches of dimension $32 \times 32 \times 3$, while the evaluation is performed on the remaining full size images of dimension $481 \times 321 \times 3$.

4.1.2 Qualitative Results and Analysis

The images shown in Figure 2 compare the distinct features of various denoising algorithms used for each of the 6 meta-testing tasks the Meta-Denoiser model performed. A general trend that we see throughout these images is that there is a clear trade-off between image sharpness and noise artifacts. Amongst all 6 meta-testing tasks, Figure 2 demonstrates that the Meta-Denoiser is able to remove a significant portion of noise while still retaining the sharpness of features in the image.

However, the same cannot be said for many of the non-learning methods. While NLM does remove almost all the noise artifacts from the noisy images, it also significantly blurs the images in the meta-test tasks that apply Poisson distributed noise and the images in the meta-test tasks that apply Gaussian distributed noise with $\sigma = 0.15$ and $\sigma = 0.3$. This results in denoised images that significantly deviate from the ground truth images in most tasks. We believe that the blurring artifacts could be due to the averaging aspect of the NLM algorithm. It's possible that the NLM algorithm is excessively averaging or smoothing over all the similar patches within the image. On the other hand, it's interesting to note that NLM does fairly well for images with Gaussian distributed noise of $\sigma = 0.1$. Visually in Figure 2, NLM retains more sharpness of image features, like the texture of the bear's fur, than the Meta-Denoiser, but simultaneously the NLM result still has a few noise artifacts for this task.

Amongst all the compared methods, bilateral filtering consistently performs the worst. A large amount of the speckling pattern attributed to the noise is still present in the

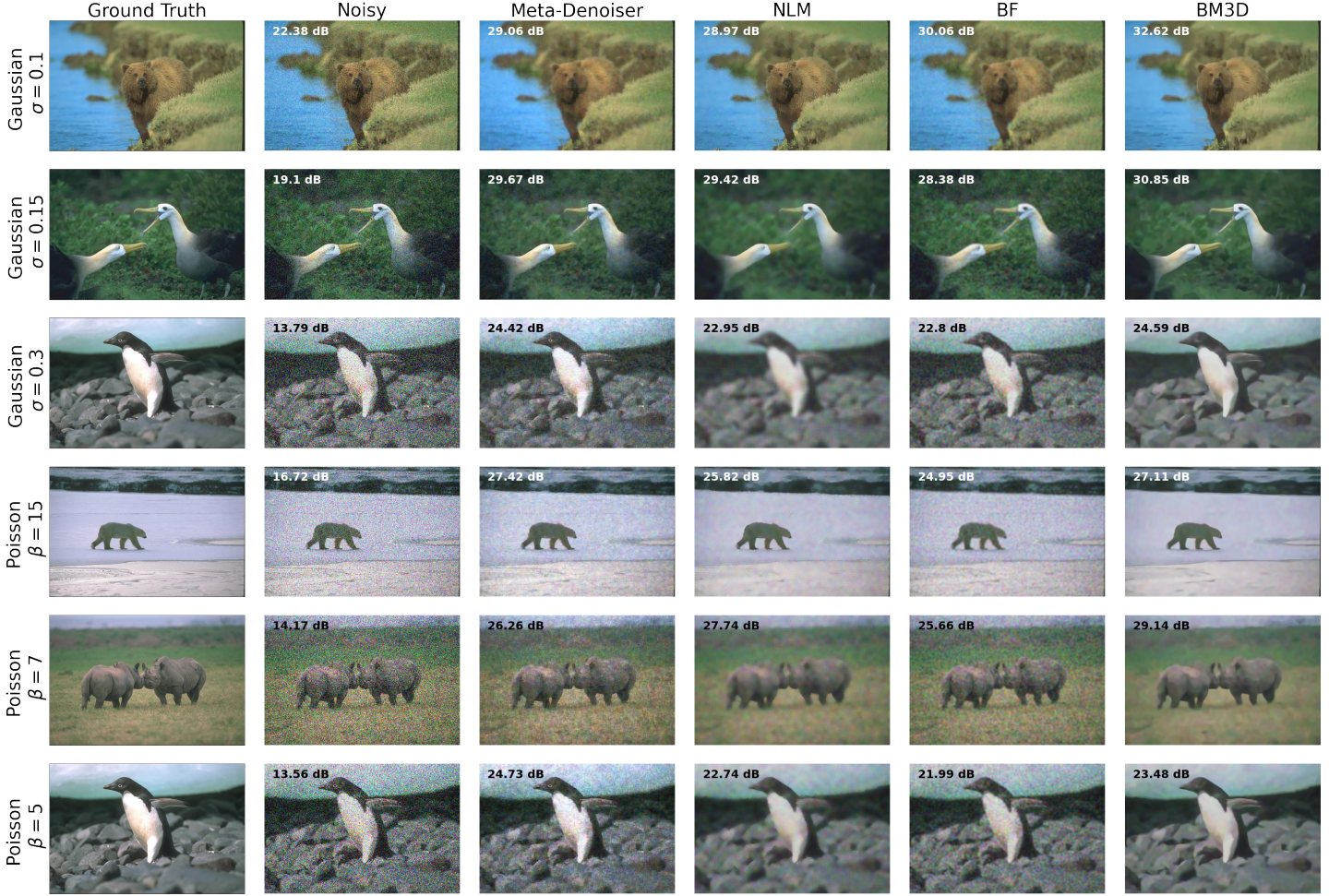


Fig. 2. **Meta-Test Results for Seen and Unseen Noise Distributions:** The above images are results from 6 different meta-test tasks. The Gaussian noise distribution, $\sigma = 0.1$ and Poisson noise distribution, $\beta = 5$ tasks were both seen during meta-training. However, the other 4 noise distributions were new to the model.

bilateral filtered images for all tasks. Furthermore, the image resolution also decreases as a result of the bilateral filtering. This result was quite surprising because in class we saw that bilateral filtering results in good quality edge-aware smoothing. However, majority of our meta-test tasks include quite a heavy amount of noise, which could be hindering the ability to accurately smooth over similar intensity values.

BM3D was the most difficult baseline for our Meta-Denoiser to beat. BM3D consistently removed majority of the noise while also maintaining a high level of image sharpness. In Figure 2, we can see that BM3D specifically does well for the Gaussian distributed denoising tasks with parameters $\sigma = 0.1$ and $\sigma = 0.15$. However, when provided with images that have a lot of noise, the BM3D algorithm results in very blurry images. This is apparent in the BM3D images for the Gaussian distributed denoising task with $\sigma = 0.3$ and all the Poisson distributed denoising tasks in Figure 2. In these types of tasks, the Meta-Denoiser results retain much sharper image features than the BM3D results, although the noise artifacts are more apparent.

4.1.3 Quantitative Results and Analysis

While Figure 2 allows us to directly compare the distinct features of the images generated by different algorithms,

TABLE 1
Average PSNR Values Per Meta-Test Task and Denoising Method

Noise Parameters	PSNR				
	Noisy	BF	NLM	BM3D	Meta-Denoiser
Gaussian, $\sigma = 0.1$	22.48	29.85	29.30	33.17	30.65
Gaussian, $\sigma = 0.15$	18.84	28.09	29.08	30.89	29.32
Gaussian, $\sigma = 0.3$	13.74	23.45	24.25	25.71	24.60
Poisson, $\beta = 5$	13.55	23.10	24.39	25.26	24.82
Poisson, $\beta = 7$	14.63	24.32	25.47	26.55	26.26
Poisson, $\beta = 15$	17.54	26.97	23.84	29.42	28.81

TABLE 2
Average Execution Time to Denoise 1 Image

Denoising Method	Execution (sec.)
Meta-Denoiser	0.0013
BF	0.0039
NLM	0.328
BM3D	12.443

Table 1 shows us the average PSNR values of each of the denoising methods for each meta-test task evaluated on a subset of the test images from the BSDS500 dataset. Table 1 indicates that on average the Meta-Denoiser is able to achieve a higher PSNR than the NLM and bilateral filtering

methods for all our meta-test tasks, which confirms the results that we visually observed in Section 4.1.2.

Out of all our non-learned baselines, the Meta-Denoiser model had the most difficulty beating the average PSNR of the BM3D method for all the meta-test tasks. In Table 1, we can see that BM3D consistently beats the Meta-Denoiser’s average PSNR by ~ 0.3 to 2 dB. It’s interesting to notice the varying difference in the PSNRs of BM3D and the Meta-Denoiser between various tasks. BM3D beats the Meta-Denoiser by a higher margin (1 to 2 dB) for the Gaussian tasks, than it does for the Poisson tasks. It’s possible that the Meta-Denoiser model is biased to perform better on noisier images based on the noise distributions we used during meta-test time, which is why the Meta-Denoiser PSNR more closely approaches the BM3D PSNR for images with Poisson noise.

Although, on average, BM3D did out-perform our Meta-Denoiser, we still believe that there are advantages to using the Meta-Denoiser over non-learned methods. For example, the Meta-Denoiser results in a significant speed up in run time for denoising each image. Meta-learning algorithms like MAML [7] can take a while to meta-train. For reference, our Meta-Denoiser model with the hyperparameters listed in Section 3.3, took approximately 2 hours to meta-train and validate on a Tesla P100 16GB GPU. However, once the model is done meta-training, with just a few steps of fine tuning during meta-test time, the model is able to denoise a single image in 0.0013 seconds (See Table 2). This is $\sim 9500\times$ faster than the time it takes to denoise one image using BM3D, which gave us the highest PSNR.

Furthermore, we believe that with more carefully tuned meta-training and fine-grained selection of meta-train tasks, the Meta-Denoiser model may be able to beat the average PSNR level of BM3D for a variety of denoising tasks. Looking at a case-by-case basis in Figure 2, we already saw that the Meta-Denoiser can result in denoised images with the highest PSNR of all the methods we compared (see rows 4 and 6 in Figure 2). The general trend we observed was that the Meta-Denoiser performs best on uniform colored images like the polar bear image in Figure 2.

4.1.4 Evaluation of the Experiment

Overall, our Meta-Denoiser model demonstrated it’s capability to generalize to new unseen denoising tasks during meta-test time, just by meta-training on few-shot denoising tasks that learn from a relatively small amount of data (10 image patches per task $\times$ 6 tasks = 60 image patches total). It’s also quite interesting that the DnCNN component of the Meta-Denoiser was able to successfully denoise images affected by signal-dependent Poisson noise. The DnCNN’s architecture is well-suited for additive Gaussian noise due to it’s residual learning component which learns to subtract out the noise from the image. However, since the Poisson noise isn’t additive, we were surprised to see that the model was able to increase the PSNR of the Poisson noisy images. This result further highlights the adaptability of meta-learning techniques like MAML [7], which allow us to find a good initialization such that the model is able to operate on 2 different types of noise distributions with various levels of noise parameters in a much shorter time frame than all of our non-learning baselines.

4.2 Generalizing to Real Noisy Images

We also briefly explored our model’s ability to generalize to real noisy images during meta-test time. We used the set of 160 images from the real noise dataset called Smartphone Image Denoising Dataset (SIDDD) [11], [12]. For this experiment our set up was almost identical to the setup described in Section 4.1.1. The only difference was that our three Poisson tasks used the following parameters: $\beta \in [128, 512, 2048]$. The Gaussian tasks remained the same as those described in Section 4.1.1.

We made this change in the meta-train tasks because when we tried meta-testing on images from SIDDD using the model trained on the tasks described in Section 4.1.1, the model was not able to denoise the images. In fact, the model further reduced the PSNR by overblurring the images. We believe this was due to the fact that the Poisson noise distributions in the original meta-training tasks resulted in images that were much more noisy than the images in SIDDD. As a result, the Meta-Denoiser model was not able to learn how to denoise images with small amounts of noise. Our new Poisson meta-train tasks include much lower levels of noise.

4.2.1 Results and Analysis

We did not have as much time to explore this experiment compared to the experiments in Section 4.1, however we were able to get some results that can be used as a stepping stone for future work.

After meta-training on the tasks described in Section 4.2, we performed meta-testing by fine tuning on 10 image patches of size $32 \times 32 \times 3$ from the SIDDD dataset. We then used the results from fine tuning to denoise the full size images from the SIDDD dataset. These images had to be downsampled by a factor of 4 in order to avoid hitting the CUDA limit on our machine.

Our meta-testing results showed that the average PSNR of the noisy images in the dataset was 35.2589 dB and our Meta-Denoiser algorithm was able to produce denoised images with an average PSNR of 36.2859 dB. This indicates that our model was generalizable enough to denoise real noisy images even though it only meta-trained on synthetically generated noisy images. With further refinement of hyperparameters and more meta-training tasks, it could be possible to improve this result.

Figure 3 provides more insight into how the Meta-Denoiser result compares against our non-learned baselines. Since the real noise present in the SIDDD images is very low, we zoomed into a patch of one of the images to highlight the denoising quality of the algorithms. In contrast to the results in Section 4.1, BM3D tends to do more poorly due to overblurring, at least for this particular image. The Meta-Denoiser and NLM generate less noisy images while still retaining sharpness of the image features. NLM slightly beats the Meta-Denoiser in PSNR. Furthermore, the text on the markers in the NLM generated image is less blurry than in the Meta-Denoiser generated image. However, we believe this is an exciting extension with room for further exploration and improvement. Many learning-based approaches require a large number of clean and real noisy image pairs [1], which can be difficult to obtain for certain applications.

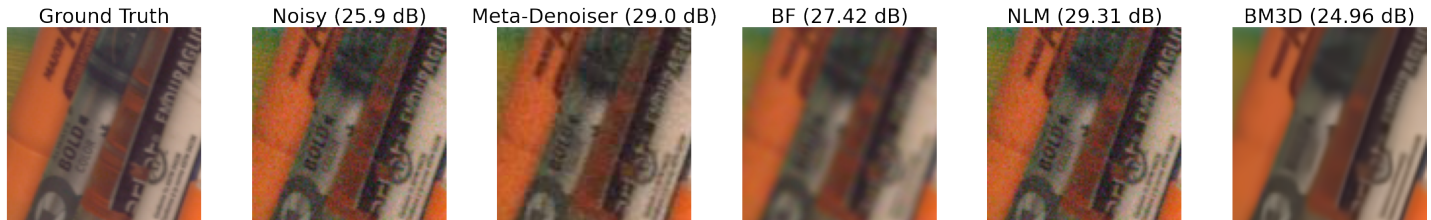


Fig. 3. **Meta-Test Results for Real Noisy Images:** The above images patches taken from an image in SIDD [12]. Images in SIDD have real noise and this figure shows how various denoising algorithm attempt to remove the noise.

The Meta-Denoiser can help overcome this issue by leveraging synthetically generated noisy images in order to extend to denoising images with real noise.

5 DISCUSSION

5.1 Limitations and Challenges

One of the main limitations with our meta-denoising approach is that the model requires quite a lot of compute power for the initial meta-training stage. Our experiments in section 4 only involved training on a small number of tasks, but it’s possible that the compute power will need to be scaled up in order to improve the model and train on more tasks. Furthermore, the meta-training will also take longer, but this should have no effect on the run time for denoising an image during meta-testing. We address potential solutions to this limitation in Section 5.2.

Other challenges in building our Meta-Denoising model included finding the right set of hyperparameters and the correct DnCNN model architecture. The original meta-denoising paper we were referring to for our project [5] recommended using 13 convolution layers with batch normalization and the ReLU activation function for the DnCNN architecture. However, with the combination of MAML and DnCNN, we found that our model’s PSNR for the denoised images decreased after we increased the number of inner layers past 7. It was also slightly challenging to find the number of inner and outer training loop iterations which wouldn’t lead to overfitting.

5.2 Future Work

Interesting future work could involve refining the selection of tasks to do better on denoising real images or even on unseen synthetically generated noisy images as discussed in Section 4.1. Increasing the number of tasks during meta-training could potentially help increase the model’s generalizability and performance. In addition to increasing the number of tasks, increasing the diversity of tasks by adding Gaussian-Poisson mixed distribution noise could also help improve the model’s ability to denoise real noisy images during meta-test time.

Aside from experimenting with task generation, we believe interesting future work could also address some of the limitations discussed in section 5.1. While compute power and training time are potential limitations to the Meta-Denoiser model, there are ways to potentially overcome or reduce the computation strain. For example there are other meta-learning algorithms which converge much faster, are

more stable, and require less computation than MAML. A meta-learning algorithm called Reptile [13] was used for meta-denoising in [5]. The authors found that using Reptile resulted in a more generalizable model that converged 4 times faster than MAML [5]. Furthermore, MAML uses second-order gradients in order to update the model initialization parameters, while Reptile is similar to a 1st order version of MAML, resulting in less computation and memory [13].

5.3 Conclusion and Takeaways

In conclusion, our Meta-Denoiser model has the ability to generalize the denoising process for multiple types of noise distributions without a dependency on large amounts of training data. We demonstrated that meta-learning can be used to find a good initialization for the DnCNN model, such that with just a few steps of fine-tuning during the meta-testing stage, the model is able to denoise images with noise distributions and levels that were not seen during meta-training.

The Meta-Denoiser was able to beat most of our non-learning algorithm baselines like NLM and bilateral filtering. Although it wasn’t able to beat the BM3D baseline, the PSNR of the denoised images resulting from the Meta-Denoiser was quite close to the PSNR of the denoised images resulting from BM3D. Furthermore, the Meta-Denoiser provides a speed boost for image denoising over BM3D and there is vast potential to further improve the performance of meta-denoising such that it will surpass methods like BM3D in the future.

ACKNOWLEDGMENTS

The authors would like to thank Professor Gordon Wetstein, Professor David Lindell, and Mark Nishimura for their support throughout the class and for access to the DnCNN and BSD300 Dataloader starter code. We would also like to thank Cindy Nguyen for her advice and help throughout our project. Last but not least, we would like to thank Edwin Pan and Anthony Vento for access to their MetaHDR MAML code as reference.

REFERENCES

- [1] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising,” *CoRR*, vol. abs/1608.03981, 2016. [Online]. Available: <http://arxiv.org/abs/1608.03981>
- [2] A. Buades, B. Coll, and J.-M. Morel, “Non-Local Means Denoising,” *Image Processing On Line*, vol. 1, pp. 208–212, 2011, https://doi.org/10.5201/ipol.2011.bcm_nlm.

- [3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.
- [4] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. 110, pp. 3371–3408, 2010. [Online]. Available: <http://jmlr.org/papers/v11/vincent10a.html>
- [5] L. Casas, G. Carneiro, N. Navab, and V. Belagiannis, "Few-shot meta-denoising," *CoRR*, vol. abs/1908.00111, 2019. [Online]. Available: <http://arxiv.org/abs/1908.00111>
- [6] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell, "Towards adapting deep visuomotor representations from simulated to real environments," *CoRR*, vol. abs/1511.07111, 2015. [Online]. Available: <http://arxiv.org/abs/1511.07111>
- [7] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *CoRR*, vol. abs/1703.03400, 2017. [Online]. Available: <http://arxiv.org/abs/1703.03400>
- [8] P. D. Patil and A. D. Kumbhar, "Bilateral filter for image denoising," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015, pp. 299–302.
- [9] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.
- [10] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011. [Online]. Available: <http://dx.doi.org/10.1109/TPAMI.2010.161>
- [11] A. Abdelhamed, R. Timofte, M. S. Brown *et al.*, "Ntire 2019 challenge on real image denoising: Methods and results," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2019.
- [12] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [13] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *CoRR*, vol. abs/1803.02999, 2018. [Online]. Available: <http://arxiv.org/abs/1803.02999>