

EE239AS Final Project Report

Zach (James) Harris

jzharris@g.ucla.edu

Dennis Wang

dmwang626@g.ucla.edu

Bryan Bednarski

bryanbed@g.ucla.edu

Joshua Hannan

jhannan@g.ucla.edu

University of California, Los Angeles
Los Angeles, CA

Abstract

The classic computational imaging problem of denoising and deblurring images reduces to extracting latent images that are drawn from some prior distribution. In this project, we implement the unrolled optimization with deep priors (ODP) network on noisy and blurry images. This network combines convolutional neural network priors with the proximal gradient method, achieving state of the art performance. We demonstrate that our implementation achieves similar performance to Diamond et al. [2].

1. Introduction

Traditionally, image reconstruction problems have been solved in one of two ways: with classical optimization algorithms or with deep neural networks. Classical algorithms utilize rigorous convergence analysis to find a solution. In contrast, neural networks find representative features to reconstruct these images. Thus, if we are able to combine the two methods, we will have a robust algorithm that combines prior information with complex models.

Diamond *et al.* [2] propose a general framework for incorporating the two methods, called unrolled optimization with deep priors (ODP). ODP unrolls an optimization algorithm, *e.g.* proximal gradient method or ADMM, and incorporates a deep convolutional neural network (CNN) prior to give the algorithm more knowledge of representative features of the image to more accurately invert the image formation operation. We achieve similar results to those found by Diamond *et al.*, outperforming current state of the art models.

Additionally, we compare our results implementing the unrolled ODP network to the “plug and play” method proposed by Zhang *et al.* [8]. The approach of this paper is to build a generalized image restoration (IR) framework, that finds a functional middle-ground between traditional

model-based optimization and modern learning methods. This method allows users to achieve good results in traditional de-blurring applications, utilizing networks trained by Zhang *et al.* for a variety of IR methods. We find that our unrolled ODP image deblurring approach described here demonstrates PSNR results comparable to these generalized results, as we will analyze later in this report.

2. Motivation

As described by Diamond *et al.*, the ODP framework derives from finding the maximum-a-posteriori (MAP) estimate of the latent image. In essence, we want to solve equation 1 for our specific problem, whether it be denoising or deblurring. For the proximal gradient ODP algorithm, we choose $r(x, \theta) = \frac{1}{2}\|x - x^k - x^{k+\frac{1}{2}}\|_2^2$ so that equation 1 becomes the proximal gradient.

$$x = \arg \min_x f(y, Ax) + r(x, \theta) \quad (1)$$

Algorithm 1: General proximal gradient ODP algorithm

Input: # of iterations N ; images

Initialize: x^0 ; $\alpha_k = C_0 C_k$; C_0 ; C_k ;

for $k = 0$ *to* $N - 1$ **do**

 Update $x^{k+\frac{1}{2}} = \text{CNN}(x^k, \theta^k)$;

 Update $x^{k+1} =$

$\arg \min_x \alpha_k f(y, Ax) + \frac{1}{2}\|x - x^k - x^{k+\frac{1}{2}}\|_2^2$;

end

2.1. Convolutional neural networks

Inspired by human perception of images, CNNs use a network of layers to learn a representation of images. Unlike most algorithms, CNNs do not require handcrafted fea-

ture engineering to learn the importance of both local and global features. CNNs are also highly robust and are easily scalable to complex large-scale image datasets. As a result, CNNs achieve state-of-the-art performance in many applications including image denoising and deblurring.

2.2. Proximal gradient method

The proximal gradient method uses a generalized form of the projection to efficiently solve non-differentiable optimization problems. Specifically, the proximal gradient method solves the problem

$$\text{minimize } f(x) = g(x) + h(x) \quad (2)$$

where $g(x)$ is convex and differentiable and $h(x)$ is convex and possibly non-differentiable. In the context of image denoising and deblurring, it is typically not necessary to achieve extremely precise results (such as when $f(x^k) - f^* \leq 1 \times 10^{-5}$), which methods such as ADMM achieve quickly. Instead, it's usually sufficient to achieve only a few orders of improvement, which the accelerated proximal gradient achieves in <10 iterations [6]. Hence, we train our algorithm with 4 iterations for image denoising and 8 iterations for image deblurring to achieve good results.

3. Framework

Classical optimization methods generally run until they reach some convergence criterion; often this criterion is having the solution within some $\varepsilon < 10^{-3}$ from the true value. However in imaging problems, this strict criterion is often unnecessary as this miniscule difference is not discernable to the human perception. Therefore we can “un-roll” the optimization algorithm, only running it for a few iterations, to combine the best of neural networks and classical methods.

When the CNN priors are incorporated to the unrolled algorithm, we essentially get a deep network that is highly tailored to finding latent images. As described in algorithm 1, each iteration of the optimization algorithm can be viewed as a CNN with many layers followed by a proximal gradient layer. Thus we have a series of CNNs and proximal gradient layers, creating a deep network. This entire network is then trained end-to-end using standard machine learning optimization methods, such as gradient descent or Adam [4].

3.1. Data set

For this project, all training, validation, and testing was done on the BSDS500 dataset [1], which consists of 500 natural images. For the denoising experiment, noisy images were generated as $y = x + z$, where $z \sim \mathcal{N}(0, \sigma^2)$ and

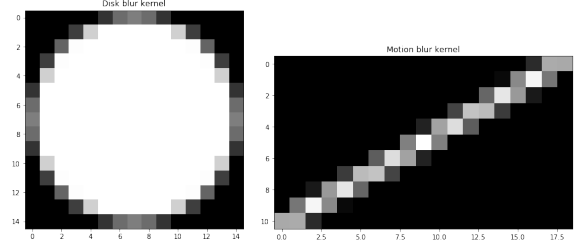


Figure 1. Disk blur and linear motion blur kernels used.



Figure 2. Gaussian noise and white noise applied to training image of mushrooms.



Figure 3. Disk blur (rad = 7) and white noise applied to training image of mushrooms.



Figure 4. Linear motion blur (len = 21, theta = 30) and white noise applied to training image of mushrooms.

$\sigma = 25$. Figure 2 shows an example of a training image of a mushroom before and after the Gaussian blur kernel and random noise was applied to the training image. For deblurring, blurry images were generated as $y = k * x + z$, where $z \sim \mathcal{N}(0, \sigma^2)$ and $\sigma = 5.702$. Two blur kernels were used: disk blur with radius 7 and linear motion blur with length 21, as shown in Figure 1. We trained separate networks for each case. Figure 3 shows an example of a training image of a mushroom before and after both the disk blur kernel was applied. Figure 4 shows an example of a training image of a mushroom before and after both the motion blur kernel was applied. For all of these images, random Gaussian white noise was added.

3.2. Design choices

For all of the following experiments, the models consist of the unrolled proximal gradient method with CNN priors. We initialize the step size $\alpha_k = C_0 C^k$, where k is the ODP iteration (starting from 1), where both C_0 and C_k are learnable. We stop training once the validation set levels out. Sections 3.2.1 and 3.2.2 will describe other hyperparameters for each experiment.

3.2.1 Denoising

The proximal gradient step for the Gaussian denoising problem reduces to solving equation 3. Here, we absorb σ^2 into the step size α_k , as in real applications we would not know the true noise. Instead, we let the algorithm learn it. The analytical solution to this minimization problem is described in algorithm 2.

We used the same hyperparameters as [2]. Specifically, each CNN prior was a 10 layer, 64 channel network with 3×3 kernels for each layer. The entire ODP network had 4 iterations and was trained end-to-end with mean-squared-error (MSE) loss and Adam [4] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay of 0.0001. C_0 and C_k were initialized to 0 and 2^{-k} , respectively. For training, all images were cropped to 180×180 , with data augmentation via random crops, flipping, and random rotations by 90 degrees. The learning rate began at 0.001 and decayed exponentially by a factor of 0.5 every 300 epochs.

$$\text{minimize } \frac{\alpha_k}{2\sigma^2} \|x - y\|_2^2 + \frac{1}{2} \|x - x^k - x^{k+\frac{1}{2}}\|_2^2 \quad (3)$$

Algorithm 2: ODP algorithm for denoising

Input: # of iterations N ; original and noisy images

Initialize:

$x^0 = y$; $\alpha_k = C_0 C_k$; $C_0 = 0$; $C_k = 2^{-k}$;

for $k = 0$ **to** $N - 1$ **do**

 Update $x^{k+\frac{1}{2}} = \text{CNN}(x^k, \theta^k)$;

 Update $x^{k+1} = \frac{\alpha_k y + x^k + x^{k+\frac{1}{2}}}{\alpha_k + 1}$;

end

3.2.2 Deblurring

The proximal gradient step for the deblurring problem with blur kernel k reduces to solving equation 4. Again, we absorb σ^2 into α_k to obtain the analytical solution in algorithm 3, where $\mathcal{F}(\cdot)$ is the Fourier transform.

Again, we used the same hyperparameters as [2]. Specifically, each CNN prior was a 5 layer, 64 channel network with 3×3 kernels for each layer. The entire ODP network

had 8 iterations and was trained end-to-end with MSE loss and Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay of 0.0001. C_0 and C_k were initialized to 1000 and 2^{-k} , respectively. For training, all images were cropped to 256×256 , with data augmentation via random crops, flipping, and random rotations by 90 degrees. The learning rate began at 0.001 and decayed exponentially by a factor of 0.5 every 20000 iterations.

$$\text{minimize } \frac{\alpha_k}{2\sigma^2} \|k * x - y\|_2^2 + \frac{1}{2} \|x - x^k - x^{k+\frac{1}{2}}\|_2^2 \quad (4)$$

Algorithm 3: ODP algorithm for deblurring

Input: # of iterations N ; original and blurry images

Initialize:

$x^0 = y$; $\alpha_k = C_0 C_k$; $C_0 = 1000$; $C_k = 2^{-k}$;

for $k = 0$ **to** $N - 1$ **do**

 Update $x^{k+\frac{1}{2}} = \text{CNN}(x^k, \theta^k)$;

$X^k = \mathcal{F}(x^k)$;

$X^{k+\frac{1}{2}} = \mathcal{F}(x^{k+\frac{1}{2}})$;

$K = \mathcal{F}(k)$;

$S_{ij} = \frac{1}{\alpha_k |K_{ij}|^2 + 1}$;

 Update

$x^{k+1} = \mathcal{F}^{-1}(S \cdot (\alpha_k K \cdot Y + X^k + X^{k+\frac{1}{2}}))$;

end

4. Results

In the following sections, we describe the training and testing results for each of the denoising, disk deblurring, and motion deblurring experiments.

4.1. Denoising

The training PSNR for the denoising network averages 34.6 dB, while the validation PSNR averages 33.6 dB. This matches [2]. In table 1, we see that the model was able to remove almost all of the noise, at the cost of smoothing out the image.

4.2. Disk deblur

The training and validation PSNR for the disk deblurring model average 24.8 dB and 24.3 dB respectively, which again is very close to the results found by [2]. They have a slightly higher training PSNR. This can be attributed to the fact that they trained on the entirety of ImageNet, which is approximately $1.2e6$ training images, while we only trained on 400 BSDS500 images. Even though we have significantly fewer training images, we still attain comparable results. Also, regarding the two dips in PSNR, these are likely

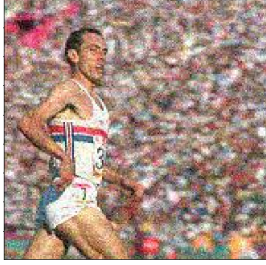
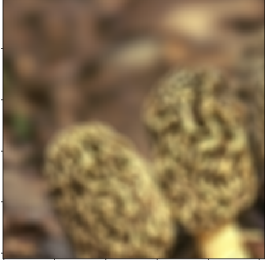

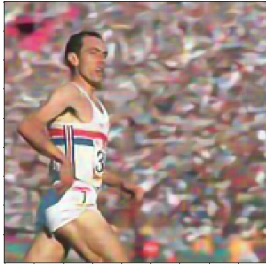
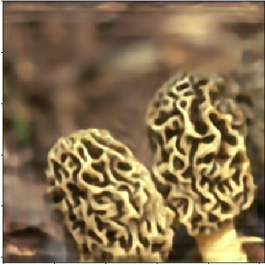




	Denoising	Disk deblurring	Motion deblurring
Input image			
ODP-predicted image			
Ground truth			

Table 1. Sample results for all 3 experiments.

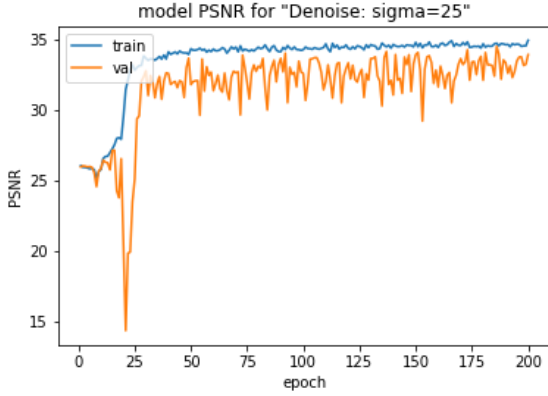


Figure 5. Model PSNR during training

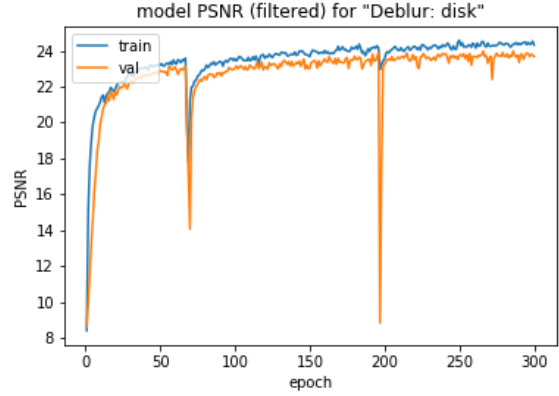


Figure 6. Disk deblurring model PSNR during training

anomalous states in the training that are quickly fixed by the model, and thus not significant. In table 1 we observe that the model successfully removed most of the blur from in-focus objects in the foreground of the image. However, the ODP-predicted image is less smooth and has artifacts in the background parts. This suggests that the ODP model per-

forms better on the salient parts of the image that have more distinct features.

4.3. Motion deblur

Similar to the disk deblurring, we attain an average training and validation PSNR of 25.4 dB and 24.9 dB respec-

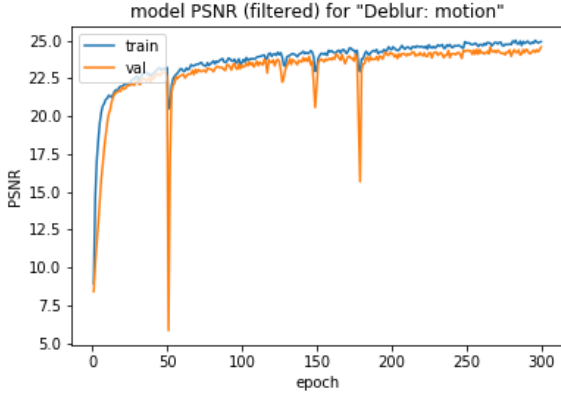


Figure 7. Motion deblurring model PSNR during training

tively. Again, this model was only trained on 400 images, in contrast to the $1.2e6$ images used by [2]. Similarly to the disk deblurring example, we notice in table 1 that the motion deblurring model performed well for the in focus part of the image, but didn't work as well for the out of focus regions. Larger sample sizes could improve this short-coming by increasing the robustness of the model.

5. Novelty

The plug and play method proposed by Zhang *et al.* [8], generalizes an optimization and learning approach to create a CNN frameworks capable of deblurring in a number of capacities. In their paper, Zhang *et al.* introduce a method that uses the half quadratic splitting technique, generalized with a denoiser prior to build a network which can be trained on images with a variety of blur kernels applied, and utilized for general inverse problems as a result. The method proposed in this paper [8], ICRNN, was developed from the generic DnCNN network [7] and at a high level is composed of a 7-layer network consisting of an iterative combination of three different types of blocks ("Dilated Convolution+ReLU" block in the first layer, five "Dilated Convolution+Batch Normalization+ReLU" blocks in the middle layers, and "Dilated Convolution" block in the last layer").

This network was then trained on a data set consisting on a total of 400 BSD, 400 ImageNet and 4,744 Waterloo Exploration Dataset images. Taking a random $N=256 \times 400$, 35×35 px frames from these images to train on [7]. The results were fairly robust due to the large training set. When implemented for this assignment, 101 validation images from the BSDS500 image set were passed through the ICRNN network after blurring the images with identical disk and motion blur kernels that our own unrolled ODP was tested with.

The results of the ICRNN plug and play network in deblurring disk-blur images were as follows: from the vali-

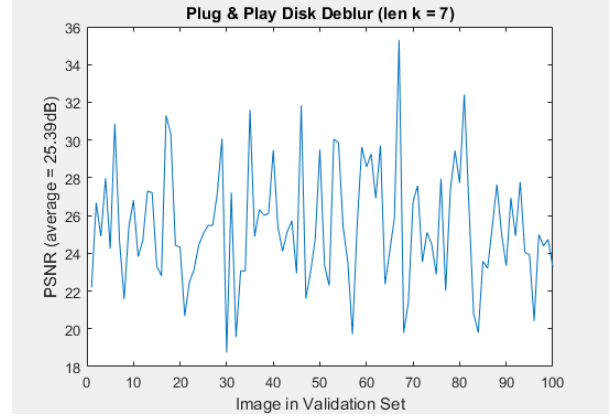


Figure 8. ICRNN network disk kernel (rad = 7) deblurring results, PSNR (in dB) for each image in the BSDS500 validation set of 101 images.



Figure 9. Example of ICRNN network deblurring when disk blur kernel applied.

dation set of 101 BSDS500 images, the average PSNR was 25.39 dB. Figure 8 shows the calculated PSNR for each image in the data set with a high of 35 dB and low of 19 dB. Figure 9 below shows an example of the blur kernel applied to the image, deblurred by the ICRNN network, and the original image. This result outperforms the ODP network by 1.09 dB.

In future work, an extension of this novelty would be to train an additional DnCNN network on the same training sets that we trained our unrolled ODP network on. This would lead to a more accurate and direct comparison of the network architectures themselves rather than evaluating the network and training sets together. The approach taken here is also a valuable comparison. Because training time is such a costly task in terms of both time and total compute this plug-and-play comparison shows a higher-level comparison between both of these networks. Finally, it is important to consider that the results of the unrolled ODP had only slightly better PSNR evaluations than our unrolled ODP architecture, which was trained on much less data.

The results of the ICRNN plug and play network in de-

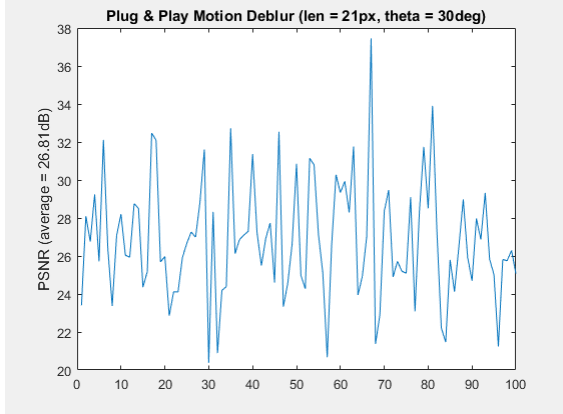


Figure 10. ICRNN network motion kernel (len = 21, theta = 30) deblurring results, PSNR (in dB) for each image in the BSDS500 validation set of 101 images.



Figure 11. Example of ICRNN network deblurring when motion blur kernel applied.

blurring motion-blur images were as follows: from the validation set of 101 BSDS500 images, the average PSNR was 26.81 dB. This outperforms the ODP network by 1.9 dB. Figure 10 shows the calculated PSNR for each image in the data set with a high of 38 dB and low of 20 dB. Figure 11 below shows an example of the motion blur kernel applied to the image, deblurred by the ICRNN network, and the original image.

6. Conclusion

In this project, we introduce, implement, and analyze the technique of using unrolled optimization with deep priors to denoise and deblur images. We compare our approach to the results found by [2] as well as a common “plug and play” method proposed by [8] and conclude that our implementation achieves comparable results. Lastly, we suggest ways in which our technique can be modified by using larger datasets and different optimization algorithms to further improve our results.

7. Future work

This project utilized the BSDS500 dataset, which only has 500 images; compared to other computational imaging machine learning applications, this is very low. Using a larger dataset such as ImageNet would greatly improve our model’s accuracy and robustness, while also reducing any overfitting that arises from such a small dataset.

The CNN models used in ODP are extremely simple by most machine learning standards, only incorporating convolutional layers and naïvely-chosen hyperparameters. More complex models such as ResNet [3] or Inception [5] can be used, along with having smarter hyperparameters can drastically improve learning and overall performance.

Additionally, we only implemented the proximal gradient method in our ODP. Future work could focus on implementing other algorithms, such as gradient descent, ADMM, Linearized ADMM, which Diamond *et al.* demonstrate achieve slightly better performance.

8. Contributions

Overall, we worked together on all aspects of the project and report, so contributions were roughly equal. Major contributions of this project by each team member are as follows: **Zach (James) Harris:** ODP implementation and training. Qualitative evaluation of results. **Dennis Wang:** Noisy/blurred image generation. Mathematical solutions to proximal gradients. Help with ODP implementation and training. **Bryan Bednarski:** Some image processing/blur application scripts. Plug-and-play setup, execution and comparison of results. **Joshua Hannan:** Analysis of optimization algorithm and CNN techniques. Help with ODP implementation and training. Evaluation of results.

References

- [1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.
- [2] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein. Unrolled optimization with deep priors. *CoRR*, abs/1705.08041, 2017.
- [3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [4] D. P. Kingma and J. Ba. 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*, 2015.
- [5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [6] L. Vandenberghe. Ece236c - optimization methods for large-scale systems (spring 2019), 2019.
- [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, Jul 2017.

- [8] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep CNN denoiser prior for image restoration. *CoRR*, abs/1704.03264, 2017.