ECE239AS Final Project Report: Unrolled Optimization with Deep Priors

Che-Hsien Lin, Neng Wang
University of California, Los Angeles
iced2000@ucla.edu, nengwang19@ucla.edu

Abstract

In this project, we reproduce the work of Diamond et al. *[4]. At the beginning, we showed how to use MAP estimation to solve the inverse imaging problem, and introduced the proposed framework, unrolled optimization with deep priors (ODP). Then we provided our analysis of ODP and compared it with the deep plug-and-play method [12, 13]. After that, we presented our experiments details and analysis on denoising, deblurring and CS MRI tasks. Lastly, We discussed some encountered problems and future works.*

1. Introduction

Inverse imaging is a core of many essential problems in computational imaging and low-level computer vision. The purpose of inverse imaging is to recover the latent image x from an observation $y = Ax + v$ where A is a linear operator and v is the sensor noise such as read, photon-shot and dark-shot noise. In this project, we mainly focus on two subproblems, denoising and deblurring where the corresponding linear operator is identity matrix and blurring operator. We also include experiment with CS MRI problem in the project.

1.1. Bayesian model

From Bayesian perspective, we can solve inverse imaging problem via maximum-a-posteriori (MAP) estimation. Suppose the latent image x follows a prior distribution $\Omega(\theta)$ with parameter θ and the observation y is drawn from the noise distribution $\omega(Ax)$. Let the probability of x sampling from $\Omega(\theta)$ $P(x;\theta)$, the probability of y sampling from $\omega(Ax)$ $P(y|Ax)$ and the probability of x yielding y $P(x|y)$, then the relationship is given by

$$P(x|y) \propto P(y|Ax)P(x;\theta), \tag{1}$$

and the MAP estimation of x is

$$\begin{aligned} x &= \arg\max_x P(x|y) \\ &= \arg\max_x P(y|Ax)P(x;\theta) \\ &= \arg\min_x f(y, Ax) + r(x,\theta), \end{aligned} \tag{2}$$

where $f(y, Ax) = -\log P(y|Ax)$ is the data term and $r(x,\theta) = -\log P(x;\theta)$ is the prior term.

1.2. Classical and deep approaches

The solution to the inverse imaging problem can be divided into two main categories, i.e. classical method and deep method, and both of them have their respective merits and drawbacks. Classical algorithms are based on formal optimization and solve the data term in equation (2) easily. However, they have to employ human-crafted image priors which is less representative of the prior distribution of natural images. On the other hand, deep methods try to learn the prior term in equation (2) by the loss function on a training data pair directly, but they are unable to systematically incorporate the image formation model.

To combine their advantages, an integration of both methods is necessary. In the paper of Diamond *et al*. [4], they proposed a general framework called unrolled optimization with deep priors (ODP) to integrate the image formation model into deep neural networks, and their work outperforms a number of prior research among various imaging problems.

1.3. Proximal gradient method

To solve equation (2) efficiently, many classical iterative methods do not minimize the prior term directly but use the proximal gradient method.

For an unconstrained optimization problem with cost function split into two components

$$\min_x f(x) = g(x) + h(x), \tag{3}$$

the proximal gradient algorithm is given by

$$\begin{aligned} u &= x_k - t\nabla g(x_k) \\ x_{k+1} &= \arg\min_z h(z) + \frac{1}{2t} \parallel z - u \parallel_2^2, \end{aligned} \tag{4}$$

where $g(x)$ must be a differentiable function.

Starting with an initial point x_0, using equations (4) to map x_k to $x_k + 1$ repeatedly, eventually we can obtain an optimal x^* matching equation (3).

There are some key points in equations (4). First, the latter equation often has a close-form solution with many important $h(x)$. Second, the latter equation does not depend on $g(x)$, which gives us an insight to replace $g(x)$ by a neural network since all we need is its gradient.

2. Related works

Since ODP is the integration of classical and deep methods, we divide related works into two parts.

2.1. Classical method

There are several classical works to solve equation (2) efficiently for convex data and prior terms, e.g. FISTA [2] and ADMM [3], and the majority of these algorithms are based on iterative methods.

For the prior term in equation (2), there are some works that use $l1$-regularization to model the sparsity of natural images [5, 10]. On the other hand, other than parameterizing the prior term, some works used the gradient or proximal operator of the prior term as field-of-experts (FoE) [6, 7].

2.2. Deep method

There are some recent works that directly solve inverse imaging problem by convolution neural network (CNN) [8, 11]. Even though their model resembles ODP, their design motivation is different. Zhang *et al*. [12, 13] proposed deep plug-and-play method for denoising and deblurring problem, and their models are almost the same as ODP. The comparison between them will be covered in section 3.2.

3. Proposed method

3.1. Unrolled optimization with deep priors

Diamond *et al*. proposed ODP to incorporate the image formation model to deep convolution network. Inspired by classical method, ODP utilizes iterative method to update its parameters. In this project, we mainly focus on the proximal gradient network, and the algorithm is summarized in Algorithm (1)

To derive the algorithm, we first let the $g(x)$ and $h(x)$ in equation (3) be $f(y, Ax)$ and $r(x, \theta)$ in equation (2), and replace the gradient of the prior term by CNN. Using ϕ to initialize x from measurement y and N to serve as the number of iterations the algorithm unrolled we can obtain the ODP proximal gradient network.

Although the ODP network is originally aimed to minimize the objective in equation (2), they are actually trained to minimize the higher-level loss, e.g. mean-squared-error or PSNR. From the perspective of deep learning, the image formation model is like a layer in deep neural network, and the loss function is the metric between the network's output and the ground-truth image x. If we denote the network by $\Gamma(y, \theta)$ and the loss function by $l(\cdot, \cdot)$, then training the ODP model is to minimize

$$E_x E_y l(x, \Gamma(y, \theta)). \tag{5}$$

Since it resembles training a CNN model, we can utilize some efficient stochastic optimizers such as Adam. However, similar to every deep learning task, the optimal model structure is distinct among different tasks and requires experiments to figure out.

3.2. Comparing to deep plug-and-play method

Before comparison, we will derive the deep plug-and-play method [12, 13] first. In half quadratic splitting (HQS) method, by introducing a variable z, we can rewrite the equation (2) to

$$x = \arg\min_x f(y, Ax) + r(z, \theta) + \frac{1}{2} \parallel x - z \parallel_2^2 . \tag{6}$$

This equation can be solved via the iterative algorithm

$$\begin{aligned} x_{k+1} &= \arg\min_x f(y, Ax) + \mu \parallel x - z_k \parallel_2^2 \\ z_{k+1} &= \arg\min_z r(z, \theta) + \frac{\mu}{2} \parallel z - x_{k+1} \parallel_2^2 . \end{aligned} \tag{7}$$

Similar to proximal gradient method, HQS method also divides the equation (2) into two subproblems. Surprisingly, the latter equation in (7) coincides with the denoising problem, whose objective function can be written as

$$z^* = \arg\min_z g(z) + \frac{1}{2t} \parallel z - x \parallel_2^2 . \tag{8}$$

Therefore, by training an end to end CNN model for denoising problem, we are able to learn the prior distribution of natural images implicitly. After that, we can plug the model into equation (7) and solve the iterative problem.

$$\begin{aligned} x_{k+1} &= \arg\min_x f(y, Ax) + \mu \parallel x - z_k \parallel_2^2 \\ z_{k+1} &= pretrainedCNN(x_{k+1}). \end{aligned} \tag{9}$$

Compared to ODP framework, deep plug-and-play method has three major differences. First, their high-level intuition is different. In ODP framework, their purpose is

Algorithm 1 ODP proximal gradient network

Initialization: $x_0 = \phi(f, A, y, \theta_0), \alpha_k = C_0 C^{-k}, C_0, C > 0$
for $k \leftarrow 0$ to $N-1$ **do**
  $x_{k+1/2} \leftarrow \text{CNN}(x_k, \theta_k)$
  $x_{k+1} \leftarrow \arg\min_x \alpha_k f(Ax, y) + \frac{1}{2} \parallel x - x_k + x_{k+1/2} \parallel_2^2$
end for

trying to incorporate image formation model to deep neural networks, while in deep plug-and-play method, they aim to incorporate CNN model into the classical iterative algorithm. Second, if we compare algorithm 1 with equation (9), we can find that unlike ODP framework, the deep plug-and-play method use CNN to replace not only the gradient of the prior term but also the term x_k. Last but not least, the way they train their model is different. In ODP framework, the model is trained end to end by high-level loss; on the contrary, the deep plug-and-play method uses iterative algorithm with a pretrained CNN. The reason why ODP framework is better than deep plug-and-play method is that by end to end training, the data and prior terms in equation (2) can efficiently share information during training process and allow networks to learn task-specific priors. And with the same reason, the ODP framework requires fewer iterations than deep plug-and-play method, which saves a number of inferencing time.

4. Experiments

In this section, we show our implementation of the ODP framework stated above. We built 4 ODP proximal gradient networks separately for denoising, deblurring(disk), deblurring(motion) and CS MRI task. We use PyTorch and train the networks on a GTX1080Ti GPU.

4.1. Denoising

Algorithm 2 shows the ODP proximal gradient network for denoising task. We followed [4] and use a network with 4 iterations and a 10 layer, 64 channel residual CNN prior with ReLu nonlinearities. We used Xavier initialization for the CNN and parameterized α_k as $C_0 C^{-k}$ with C_0 and C learnable, initialized to $C_0 = 0$ and $C = 2$. We used a mean-squared-error.

Algorithm 2 ODP proximal gradient denoising network

Initialization: $x_0 = y, C_0 = 0, C > 0$
for $k \leftarrow 0$ to $N-1$ **do**
  $\alpha_k \leftarrow C_0 C^{-k}$
  $x_{k+1/2} \leftarrow \text{CNN}(x_k, \theta)$
  $x_{k+1} \leftarrow (\alpha_k y + x_k + x_{k+1/2})/(\alpha_k + 1)$
end for

We used a training set of 400 images from BSDS500 dataset [1], and used the rest 100 images as a validation set. We randomly cropped the input images to 180×180 and used random vertical, horizontal flipping and rotation by 90 degrees for data augmentation. For additive noise, we used random Gaussian noise with standard deviation $\sigma = 25$. We started with learning rate 0.001 decayed exponentially by factor 0.5 every 300 epochs following [4], but later changed the initial learning rate to 0.0001 due to the instability in the training process. We used Adam $\beta_1 = 0.9$, Adam $\beta_2 = 0.999$, and weight decay 0.0001. While [4] used a batchsize of 4, we tried using larger batchsize with 8 or 16, and made the training faster and more stable. Figure 1 shows the loss curve in average PSNR(dB).

As a result, we achieved average PSNR of 29.78dB at 1278 epoch in the denoising task, which is close to the result from [4] and outperforms many state-of-art methods. Figure 2 shows some results of the denoising network.

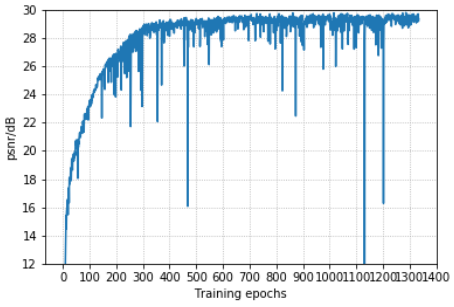

Figure 1. Loss curve in average PSNR(dB) for denoising network

4.2. Deblurring

Algorithm 3 shows the ODP proximal gradient network for denoising task. We followed [4] and use a network with 8 iterations and a 5 layer, 64 channel residual CNN prior with ReLu nonlinearities. We used Xavier initialization for the CNN and parameterized α_k as $C_0 C^{-k}$ with C_0 and C learnable, initialized to $C_0 = 1000$ and $C = 2$. We used a mean-squared-error.

We trained two models, one for disk blur kernel and another for motion kernel. We rewrote the $fspecial()$ function from MATLAB in python and used it to genarate kernels. We trained the disk deblurring model with a disk7 kernel, and the motion deblurring model with a motion kernel, of which the length is 21, and the angle is 45 degree in

Figure 2. Result of ODP proximal gradient denoising network at $\sigma = 25$. The first row is the noisy input image, the second row is the output result, the last row is ground truth.

Algorithm 3 ODP proximal gradient deblurring network. $\mathcal{F}$ is the DFT, k is the blur kernel, and K is the DFT of k

Initialization: $x_0 = k^T * y, C_0 > 0, C > 0$
for $k \leftarrow 0$ to $N - 1$ **do**
  $\alpha_k \leftarrow C_0 C^{-k}$
  $x_{k+1/2} \leftarrow \text{CNN}(x_k, \theta)$
  $x_{k+1} \leftarrow \mathcal{F}^{-1} diag(\alpha_k |K|^2 + 1)^{-1} \mathcal{F}(\alpha_k k^T * y + x_k + x_{k+1/2})$
end for

a counter-clockwise direction.

Instead of training with the $1.2e6$ ImageNet training dataset mentioned in [4], we still used the 400 images training set from BSDS500 dataset [1], and used the rest 100 images as a validation set. We randomly cropped the input images to 180×180 and used random vertical, horizontal flipping and rotation by 90 degrees for data augmentation. We used random Gaussian noise with standard deviation $\sigma = 5.7020$. We started with learning rate 0.0001 decayed exponentially by factor 0.5 every 300 epochs as same as the denoising task. We used Adam $\beta_1 = 0.9$, Adam $\beta_2 = 0.999$, and weight decay 0.0001. And we used batch-size of 8 for training. Figure 4 shows the loss curve in average PSNR(dB).

The disk and motion deblurring network achieved average PSNR of 23.94 dB and 24.74 dB respectively on the validation set. And the overall performance was close to which was described in the supplement material of [4]. Because of the overfitting problem, which can also been seen from the plot, we later deleted one layer from the CNN prior. The performance of the network was not largely influenced, but the overfitting problem was also not completely solved. So our next plan would be training this deblurring network on a larger dataset, like the VOC2012 dataset and ImageNet dataset. We'll also try working with larger image size like 256×256 . Figure 3 shows some results of our ODP proximal gradient deblurring network.

4.3. CS MRI

Algorithm 4 shows the ODP proximal gradient network for CS MRI task. We followed [4] and use a network with 8 iterations and a 7 layer, 64 channel residual CNN prior with ReLu nonlinearities. We used Xavier initialization for the CNN and used a mean-squared-error.

We used the data from the GitHub repository of the work from [9]. We used 80 of the 100 chest images for training, and used the rest 20 images as a validation set. We used 4

Figure 3. Result of ODP proximal gradient deblurring network. The first row is the blurry input image with Gaussian noise at $\sigma = 5.7020$, the second row is the output result, the last row is ground truth. The three columns of the left half are results from the disk blur kernel 'disk7'. The three columns of the right half are results from the motion kernel with length 21 and angle 45 degree.

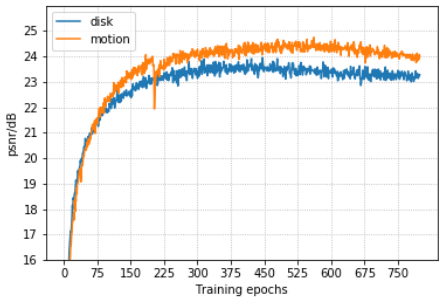

Figure 4. Loss curve in average PSNR(dB) for deblurring network

Algorithm 4 ODP proximal gradient CS MRI network. $\mathcal{F}$ is the DFT, P is the binary, diagonal subsampling matrix.

Initialization: $x_0 = \mathcal{F}^{-1}Py$
for $k \leftarrow 0$ to $N-1$ **do**
  $x_{k+1/2} \leftarrow \text{CNN}(Re(x_k), \theta)$
  $z_{k+1} \leftarrow \mathcal{F}(x_k + x_{k+1/2})$
  $x_{k+1} \leftarrow \mathcal{F}^{-1}(Py + (I-P)z_{k+1})$
end for

subsampling patterns ranging from sampling 20% to 50% of the Fourier domain. Instead of training one model for all the patterns like [4] did, we trained each model separately for each pattern.

All the input images are at the size of 256×256. We used random vertical, horizontal flipping and rotation by 90 degrees for data augmentation. Under noise free assumption, by applying the image formation model $y = P\mathcal{F}x$, We finished the preprocessing of the input data. We started training with learning rate 0.005 decayed exponentially by factor 0.5 every 300 epochs. We used Adam $\beta_1 = 0.9$, Adam $\beta_2 = 0.999$, and weight decay 0.0001. And we used a batchsize of 4 for training. We didn't use batchsize 8 as [4] did because of memory limitation. Since reducing the layer number would hurt the performance, we decided to use a smaller batch. Figure 5 shows the loss curve of the 4 models in average PSNR(dB).

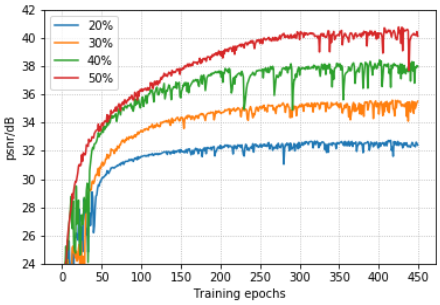

Figure 5. Loss curve in average PSNR(dB) for CS MRI network

Table 1 shows our validation results of the four models on chest images. We also test our models on two brain images, which worked out well but the testing amount is too small to give a convincing result. These results don't look as

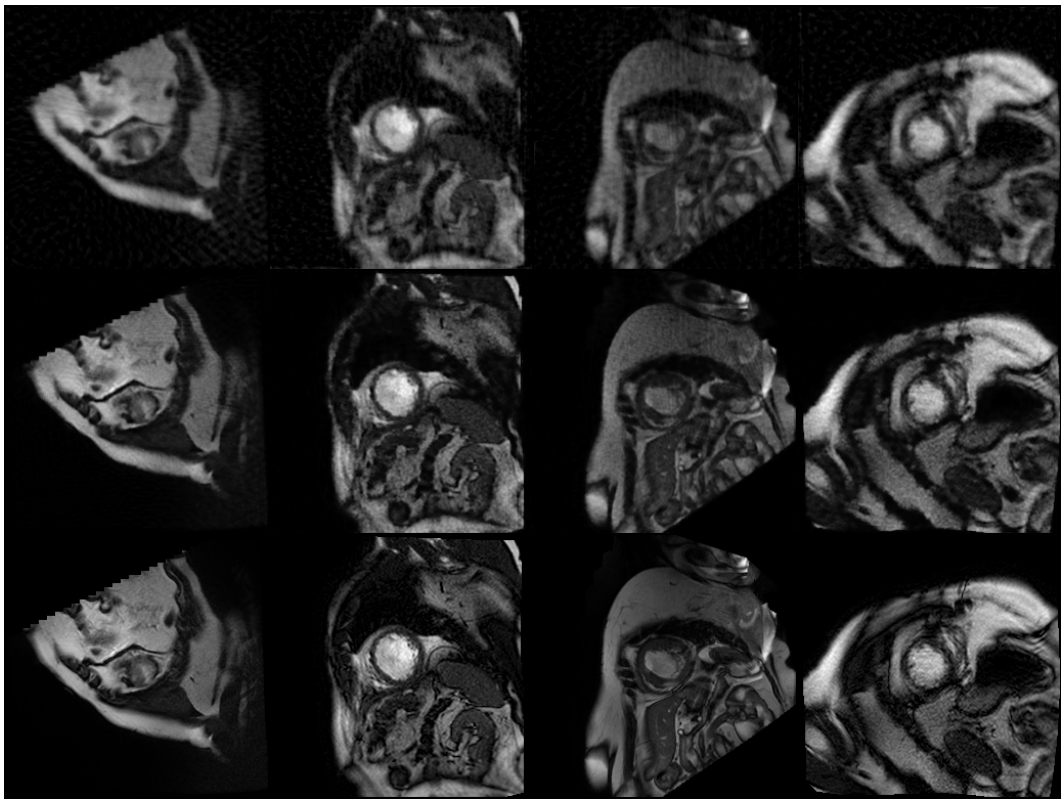

Figure 6. Result of ODP proximal gradient CS MRI network on chest images. The first row is the subsampled input image with 20% of the Fourier domain. The second row is the output result, the last row is ground truth.

Subsamping Rate	20%	30%	40%	50%
PSNR(dB)	32.73	35.58	38.42	40.75

Table 1. Results of CS MRI on chest images

good as [4] currently, but due to data difference, we cannot simply compare the results. In addition, we are still trying to figure out how [4] managed to train a single model with the 4 patterns. We tried interpreting P in Algorithm 4 as the matrix $y! = 0$ and trained with all 4 sampling patterns and P unfixed. But the training was unstable and gave PSNR of 37 dB at its best, which is not a very good result either. So our next step for this part would be exploring the method to bring in all patterns and finding the correct data to train and evaluate our model.

5. Contributions

Che-Hsien Lin: 50 %

Built the model for denoising task and set the basic framework. Conducted further research into plug-and-play and other related methods.

Neng Wang: 50 %

Built the models for deblurring(disk & motion) and CS MRI. Tuning, training, and analysis of the four built models.

6. Conclusion

In this report, we first introduced the proposed framework, unrolled optimization with deep priors (ODP) [4]. Then we showed our analysis into the algorithms and compared it to the plug-and-play method [12, 13]. Further, we post our experiments details and analysis on reproducing [4]'s work on denoising, deblurring and CS MRI. We also discussed some problems we encountered and possible future strategies.

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] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[4] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein. Unrolled optimization with deep priors. *arXiv preprint arXiv:1705.08041*, 2017.

[5] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 399–406. Omnipress, 2010.

[6] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 860–867. Citeseer, 2005.

[7] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781, 2014.

[8] C. J. Schuler, H. Christopher Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1067–1074, 2013.

[9] J. Sun, H. Li, Z. Xu, et al. Deep admm-net for compressive sensing mri. In *Advances in neural information processing systems*, pages 10–18, 2016.

[10] S. Wang, S. Fidler, and R. Urtasun. Proximal deep structured models. In *Advances in Neural Information Processing Systems*, pages 865–873, 2016.

[11] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Advances in neural information processing systems*, pages 1790–1798, 2014.

[12] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3929–3938, 2017.

[13] K. Zhang, W. Zuo, and L. Zhang. Deep plug-and-play super-resolution for arbitrary blur kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1671–1681, 2019.