Regularization and Applications of a Network Structure Deep Image Prior

Timothy Anderson
Department of Electrical Engineering
Stanford University
Stanford, CA
timothy.anderson@stanford.edu

Abstract

Finding a robust image prior is one of the fundamental challenges in image recovery problems. Many priors are based on the statistics of the noise source or assumed features (e.g. sparse gradients) of the image. More recently, priors based on convolutional neural networks have gained increased attention, due to the availability of training data and flexibility of a neural network-based prior.

Here, we present results for an entirely novel neural-network based image prior that was introduced in [18], known as a network structure deep image prior. We test the effect of regularization on convergence of the network structure prior, and apply this prior to problems in deconvolution, denoising, and single-pixel camera image recovery.

Our results show that regularization does not improve the convergence properties of the network. The performance for improving deconvolution results and single-pixel camera image recovery are also poor. However, the results for denoising are comparable to baseline methods, achieving a strong PSNR for several test images. We believe with further work, this method can be readily applied to similar computational imaging problems such as inpainting and demosaicking.

1. Introduction

Finding an effective prior is one of the most fundamental challenges in computational imaging. The general image denoising or restoration problem is of the form:

$$\min_{\boldsymbol{x}} E(\boldsymbol{b}, \boldsymbol{x}) + \lambda \Gamma(\boldsymbol{x}) \tag{1}$$

where $E(\cdot)$ is an energy function dependent on the observed image $\boldsymbol{b}$ and reconstructed image $\boldsymbol{x}$, λ is a hyperparameter, and $\Gamma(\cdot)$ is a prior used to regularize the reconstructed image. In a typical image reconstruction problem, the energy function will be almost entirely task-dependent [18], meaning the quality of the reconstruction depends heavily on the prior $\Gamma(\cdot)$. Image priors are either applied directly in the optimization process (if they are differentiable), or used in an iterative process where first the prior is applied, then the energy function is minimized.

Classical image priors are typically based off some existing or assumed structure of the image. For example, the isotropic total variation (TV) prior $\Gamma(\boldsymbol{x}) = \sqrt{|\boldsymbol{D}_x \boldsymbol{x}|^2 + |\boldsymbol{D}_y \boldsymbol{x}|^2}$ is based off the assumption that natural images contain sparse gradients. The non-local means (NLM) prior $\Gamma(\boldsymbol{x}) = NLM(\boldsymbol{x}, \sigma)$ is similarly based off the assumption that natural images have self-similarity [3].

2. Background

Deep neural networks have exhibited stunning success in a variety of computer vision and image processing tasks [12, 16, 10], and provide a general framework for learning function approximators from data. Consequently, there has been much interest recently in applying neural network architectures to image recovery and restoration tasks.

Several methods apply neural networks to directly find the latent image. That is, these methods train a neural network to directly solve the inverse problem in eqn. 1. For example, SRCNN from [9] showed state of the art super resolution results using a convolutional neural network (CNN) to super-resolve an input image. [14] similarly introduced the SRResNet and SRGAN architectures to perform super-resolution. [4] introduced a blind deconvolution method which uses a neural network to predict the motion blur kernel. [19] develops an iterative non-blind deconvolution method which alternately deblurs an image using quadratic optimization and denoises using a CNN.

Beyond using the network to directly perform image recovery or restoration, much recent work has also centered on using neural networks as priors for image recovery and restoration. Image priors bifurcate into two distinct classes: those learned from data, and "data-free" priors based only on a single input image. The vast majority of neural-network priors fall into the former class, where a

prior function is learned from multiple examples.

The most direct approach to using a CNN-based prior is to substitute a trained CNN for the prior term $\Gamma(\boldsymbol{x})$ in eqn. 1. This is the approach taken by [13], which integrates FFT-based deconvolution with a CNN-based image prior. [8] presents a more general framework for integrating a CNN prior with off-the-shelf optimization algorithms.

A popular optimization technique for solving the split-objective optimization problem in eqn. 1 is the Alternating Direction Method of Multipliers (ADMM) [2]. ADMM is an attractive framework for imaging inverse problems because there often exist computationally-efficient closed-form solutions for the proximal operators [1]; several neural network-based image priors have been developed with the goal of preserving the optimization structure provided by ADMM, and therefore the acceleration provided by these methods. [5] takes a similar approach to [19], but embeds their method in an ADMM optimization framework. Instead of learning the prior itself, [15] attempts to directly learn the proximal operators for ADMM-based deconvolution.

While CNN-based image prior methods have been successful, these methods often require extensive training and/or large amounts of data to achieve strong performance. Methods not based on neural-networks (e.g. BM3D and CBM3D from [6]) have also been successful, but rarely surpass neural network-based methods in generality or performance. Consequently, there exists a need for a method with the speed and generality of neural network methods, but does not require large amounts of training data.

3. Network Structure Deep Image Prior

To overcome the difficulties of learned and data-free priors, [18] introduced an entirely novel type of image prior based on the structure of a neural network. The basic optimization problem for a neural network is:

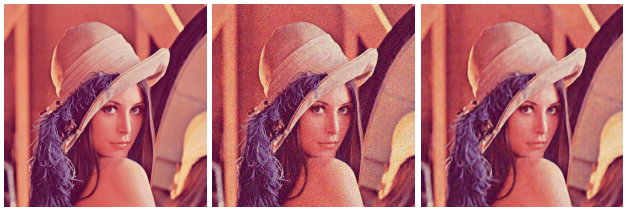
$$\min_{\boldsymbol{\theta}} \ \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})$$

where $\ell(\cdot)$ is the loss function, $\boldsymbol{\theta}$ the vector of network parameters, $\boldsymbol{x}$ the input image, $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ the output, and $\boldsymbol{y}$ the desired output of the network.

In the approach used by [18], they train a neural network to recreate a corrupted/distorted input image, and in doing so are able to remove the noise or artifacts present in the image. Let $\boldsymbol{z}$ be a random input vector ([18] uses $\boldsymbol{z} \sim \mathcal{U}(0, 1/10)$ i.i.d.), $f_{\boldsymbol{\theta}}(\boldsymbol{z})$ be the neural network output, and $\boldsymbol{x}_0$ the image we would like to reconstruct. Then, for fixed $\boldsymbol{z}$, they solve:

$$\min_{\boldsymbol{\theta}} \ \ell(f_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{x}_0)$$

From hereon, we will refer to this type of prior as a **network structure (NS) prior**. This setup effectively minimizes over the space of natural images realizable by the neural network. To apply the NS prior to an image, the above optimization is run until some stopping procedure is met ([18] uses early stopping in all of their experiments). An example of denoising with this method is shown in fig. 3.



(a) Ground truth (b) Noisy image (c) Denoised

Figure 1: Denoising using a NS deep image prior.

The NS prior is based on the observation that a neural network will gravitate towards natural images more quickly than unnatural (e.g. noisy) images. If we realize the predicted image $\boldsymbol{x}$ as the output of a neural network, then the output of the network will first produce natural-looking images, and in this sense acts as an implicit image prior. The primary advantage of the NS prior over many other neural network-based methods is that it does not require any training data, and instead relies on the network structure to act as a prior. As shown in [18], the network structure greatly affects the results, depending on the task and the network structure used.

4. Regularization

4.1. Regularization Approaches

In order to improve network generalization, regularization techniques are often applied to the network, either during the training process or in the form of a penalty in the optimization process. Here, we explore three common regularization methods for neural networks.

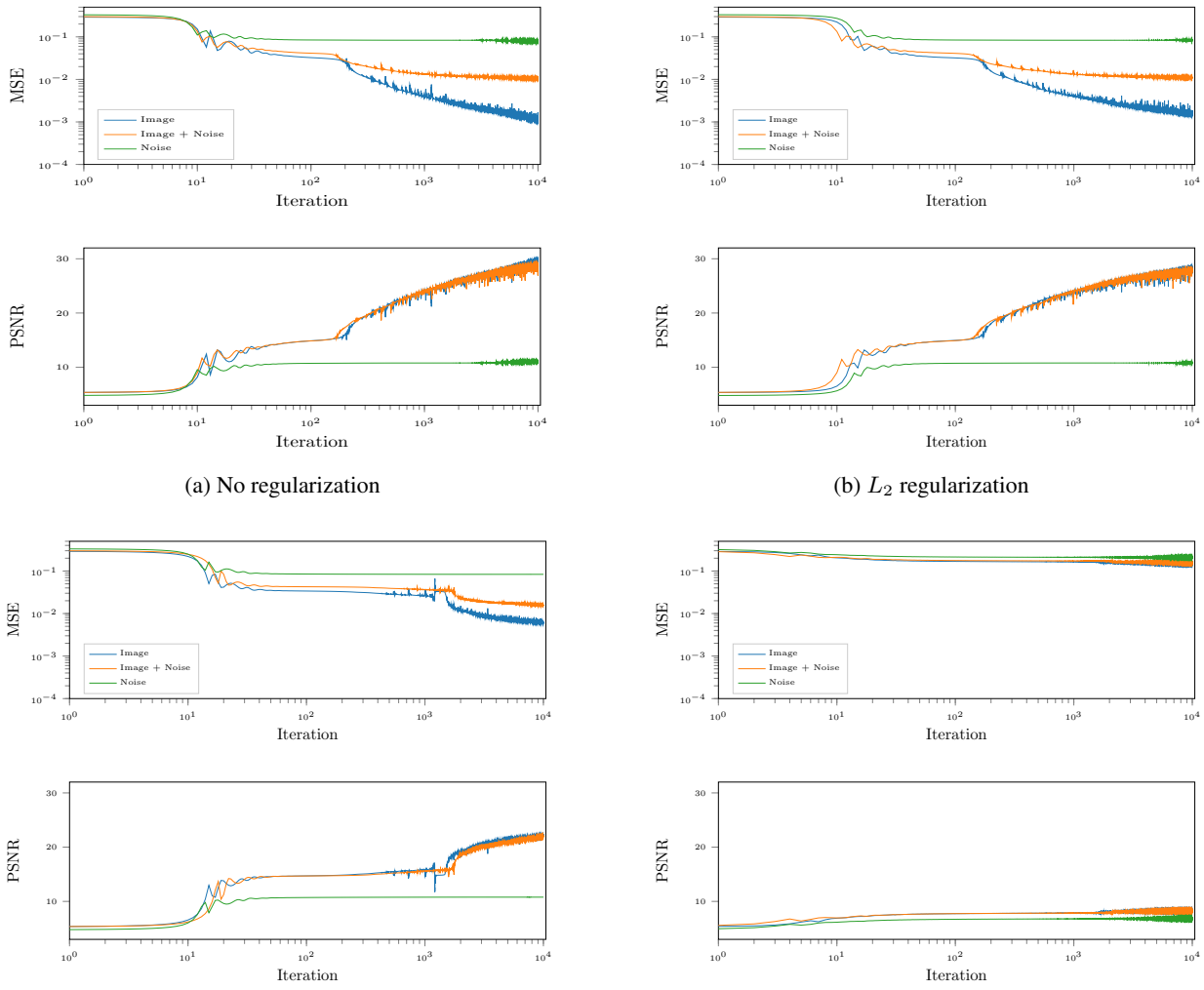
Dropout [17] is a training technique where features are passed to the next network layer with some probability p_{keep} i.e.

$$\boldsymbol{x}_{i+1} = \boldsymbol{m}_i \circ \boldsymbol{x}_i$$

where $\boldsymbol{x}_i$ is the feature map at the i^{th} network layer and $\boldsymbol{m}_i \sim Bernoulli(p_{keep})$ is a Bernoulli random vector. Randomly passing feature maps to the next layer effectively applies a stochastic relaxation to the optimization process of the network, thereby allowing the network to find a more general local minimum or saddle point.

L_1 and L_2 regularization differ from dropout in that they impose a penalty on the objective, so the optimization objective becomes:

$$\min_{\boldsymbol{\theta}} \ \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}) + \lambda R(\boldsymbol{\theta})$$



(a) No regularization

(b) L_2 regularization

(c) L_1 regularization

(d) Dropout

Figure 2: Regularization results. We observe that adding regularization does not improve convergence. All three types of regularization also decreases the PSNR for all three cases.

where $R(\boldsymbol{\theta})$ is the regularization function and λ the associated regularization parameter. For L_1 regularization, the regularization function has the form:

$$R(\boldsymbol{\theta}) = \sum_i ||W_i||_1$$

where W_i is the weight matrix for each layer of the network. Similarly, for L_2 regularization:

$$R(\boldsymbol{\theta}) = \sum_i \frac{1}{2}||W_i||_2^2$$

L_2 regularization is by far the most commonly employed method of regularization in neural networks, and has been shown to improve network generalization in a wide range of contexts.

4.2. Convergence Issues

The primary drawback of the approach in [18] is that early stopping during the training process must be employed to achieve strong results. As shown in fig. 3 of [18], in applications such as denoising or JPEG artifact removal, in the early stages of training the network gravitates towards the natural (noise- or artifact-free) image. However, in later stages of training the network begins to overfit to the noise and artifacts, causing the output image to almost exactly match the input and rendering the result useless.

4.3. Regularization Results

To test the effect of regularization on convergence, we trained a NS deep image prior to reproduce a the original Lena image, Lena with additive noise, and an image of ran-

dom noise. In these experiments, we apply the autoencoder architecture (without skip connections) from [18], and train the network with the Adam optimizer (from [11]). The results are shown in fig. 2.

The results show that regularization is actually harmful to the effectiveness of the deep image prior. Specifically, for all three regularization methods we see that the PSNR after 10^4 iterations is lower than for the unregularized network. Furthermore, then loss plots do not show improved convergence for the regularized networks.

The negative results for regularization are likely because the regularization methods limit how much of the network's structure the deep prior is able to exploit. Specifically, L_2 regularization penalizes the parameters for moving too far from the origin, L_1 regularization enforces sparsity, and dropout randomizes the network structure itself. A possible explanation for the poor performance of L_1 regularization is that sparsity limits how many network connections the network can use to recreate the input image.

The poor performance for dropout in many ways validates the conclusions in [18]. There, the authors claimed that the deep image prior arises because of a synergy between the network structure and the image data. Dropout creates uncertainty in the network structure, meaning that it is impossible for the network to synchronize with the image. As a corollary, this result may partially explains why dropout is a robust regularization method. The uncertainty in the network structure prevents the network structure from becoming too aligned with the data in the training set, forcing it to seek more general features to perform classification.

5. Applications

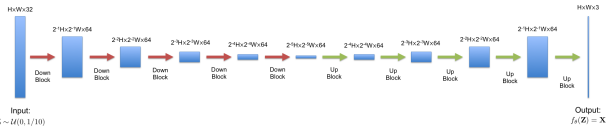
In this following section, we apply the NS prior to several image recovery and reconstruction problems and compare to three well-known baseline methods: total variation prior and non-local means. In all experiments, we use an autoencoder architecture (shown in fig. **??**) similar to the autoencoder architecture in [18], and train the NS prior using the Adam optimizer with learning rate of 10^{-4}.

5.1. Deconvolution

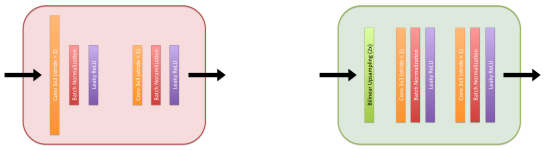
The first application is image non-blind deconvolution. The basic image deconvolution problem can be stated as:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda\Gamma(\boldsymbol{x})$$

where $\boldsymbol{C}$ applies a (known) blur kernel to the image $\boldsymbol{x}$, and $\boldsymbol{b}$ is the observed (blurry) image. There are several approaches to solving this problem. When using a uniform prior, Wiener deconvolution is known to be optimal. When using a known prior such as total variation (TV), a popular



(a) Network structure



(b) Downsampling block (c) Upsampling block

Figure 3: Network architecture and convolutional blocks. This autoencoder architecture was used for all experiments presented.

approach is to use ADMM [2] to solve the resulting optimization problem:

$$\min_{\boldsymbol{x}} \frac{1}{2}||\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda TV(\boldsymbol{x})$$

Table 1 summarizes results for deconvolution using Wiener filtering, ADMM + TV prior (with two different regularization parameters), and a NS prior applied after Wiener filter. We see that the NS prior causes a negligible change in the PSNR when it is applied to Wiener filtering. The NS prior assumes no information on the image other than the image is natural, and therefore is unable to perform operations such as deconvolution. This illustrates one of the major shortcomings of the NS prior: it has no knowledge of blurry vs. non-blurry, just an implicit knowledge of natural vs. unnatural images, and therefore cannot differentiate a blurry and non-blurry image.

5.2. Denoising

We also compare the NS prior to two popular denoising algorithms: non-local means (NLM) from [3] and BM3D/CBM3D from [6]. We test each algorithm on four different images (taken from [6]): two grayscale and two color. To generate the noising images, we add (clipped) Gaussian noise to each image for a range of σ values. Table 2 summarizes the results for denoising with each denoising algorithm, and fig. 3 shows the results for an example image.

At lower noise levels, the NS prior performs comparably to or worse than the other methods. However, at the highest noise levels, the NS prior significantly outperforms NLM and BM3D. For all images, the NS prior shows the strongest performance for $\sigma = 0.5$, and in general the performance of

$\sigma \backslash$ Method	Wiener	ADMM + TV ($\lambda = 5 \times 10^{-3}$)	ADMM + TV ($\lambda = 5 \times 10^{-2}$)	Wiener + NS
10^{-3}	**24.9376**	23.5814	22.4688	24.8879
10^{-2}	23.5545	**23.5759**	22.4694	23.4526
10^{-1}	18.1195	**23.0354**	22.3920	17.6956

(a) Results for House image.

$\sigma \backslash$ Method	Wiener	ADMM + TV ($\lambda = 5 \times 10^{-3}$)	ADMM + TV ($\lambda = 5 \times 10^{-2}$)	Wiener + NS
10^{-3}	**26.6473**	25.9782	25.2714	25.7898
10^{-2}	25.8921	**25.9751**	25.2696	25.2730
10^{-1}	19.6848	**25.3060**	25.1263	19.5468

(b) Results for Parrots image.

Table 1: PSNR values for various deconvolution methods. σ represents the additive noise standard deviation, and λ is the regularization parameter for ADMM. All images blurred with a Gaussian blur kernel with $\sigma_{\text{blur}} = 5$px.

the NS prior changes very little except at the highest noise level.

These results correspond with those in [18] for both denoising and inpainting. Denoising at very high noise levels is similar to image inpainting in that there are many pixels distributed across the image for which the information is missing. The NS prior does very well with inpainting tasks; the strong performance for denoising images with large noise seems to be an extension of this same phenomenon.

5.3. Compressive Imaging

Finally, we show results for reconstructing images taken with a single pixel camera using a NS prior. Single-pixel camera image recovery is based on solving an underdetermined linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ is the measurement matrix (for N pixels and M measurements) and $\boldsymbol{b} \in \mathbb{R}^M$ is the vector of measurements.

Most problems have a compression factor $N/M > 1$ [7], so in order to make the problem well-posed we must impose a prior $\Gamma(\boldsymbol{x})$ on the recovered image, so we seek to solve:

$$\min_{\boldsymbol{x}} \ \frac{1}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2 + \lambda\Gamma(\boldsymbol{x})$$

Two common priors in single pixel camera image recovery are the TV prior and a self-similarity prior (e.g. NLM). With the self similarity priors, NLM and BM3D, we simply deploy these algorithms as priors in the ADMM iteration. The NS prior is also effectively a self-similarity prior [18], so we similarly use it as a proximal operator.

We test all four priors at compression factors (CF) 1, 2, 4, and 8 for two different images: the Stanford logo, and Barbara image from the Set14 dataset. Table 3 shows the results for these methods, and fig. 4 shows the recovered images for the Stanford logo.

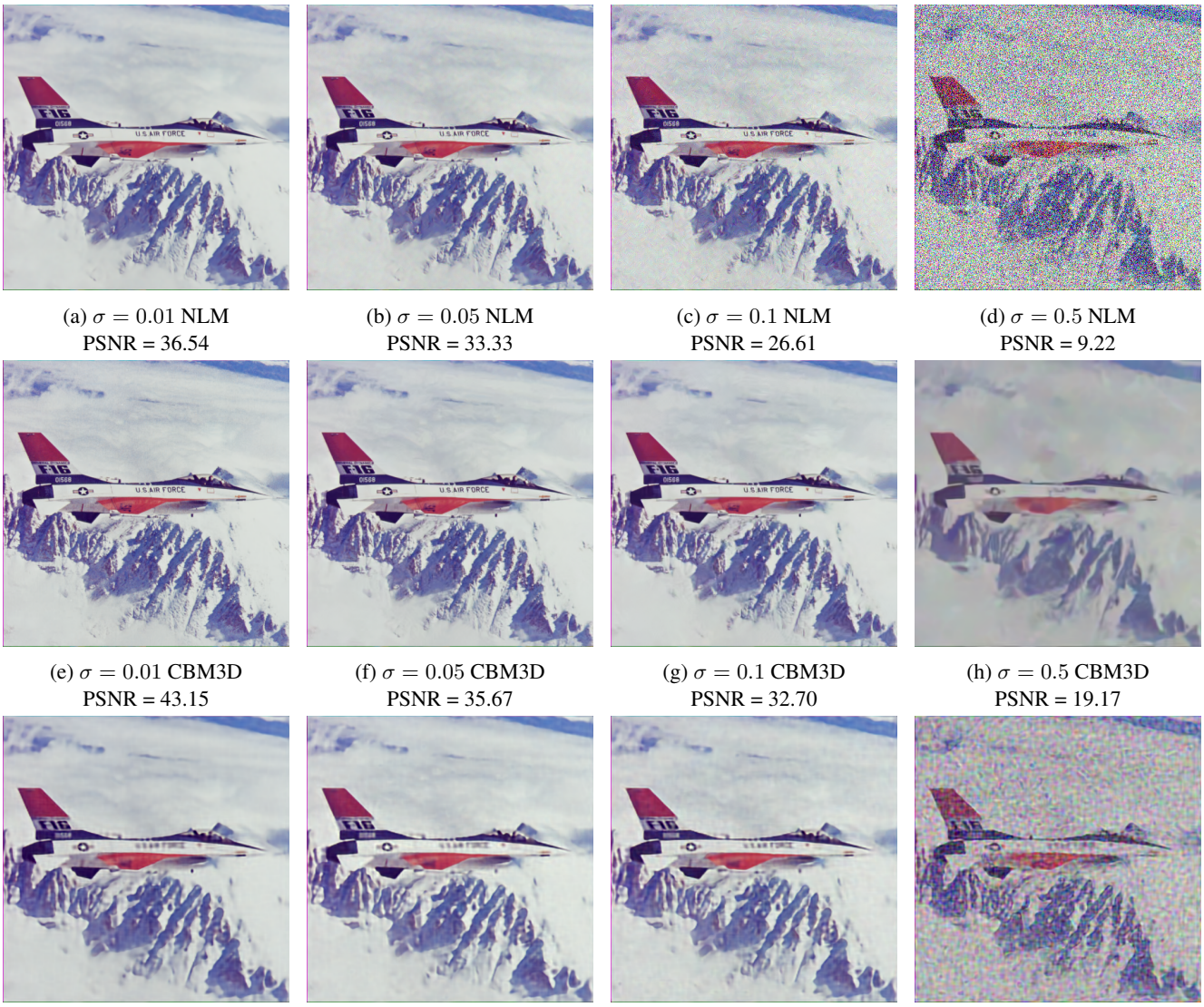
Overall, the network structure prior performs very poorly in compressive imaging recovery compared to the other priors. Furthermore, the NS prior is extremely computationally intensive: it can take up to an hour for a single image to process even on GPU hardware, whereas the other methods can converge on the order of seconds to minutes. In all cases, BM3D seems to be the best method for recovering images with a single pixel camera.

6. Conclusion

The work presented here shows that regularization for a network structure deep image prior decreases performance and does not improve convergence. We also present results for numerous experiments applying the NS prior to various image processing tasks. These experiments showed that the NS prior works well for image denoising, but performs poorly for tasks where image information—specifically, information in the frequency domain—is entirely missing, such as deconvolution and single pixel camera image recovery.

While some of the results for the NS prior are impressive, it is important to note that all NS prior computations take several minutes on a GPU. For example, each denoising experiment took around 15 minutes on an Nvidia 1080 Ti GPU. Most other algorithms discussed here (e.g. NLM) can run efficiently on a CPU on the order of seconds. So, the NS prior method is intriguing insofar as its flexibility for various image processing tasks and potential theoretical insights into CNNs, but it is not a computationally efficient approach.

For this reason, future work on the NS prior should focus primarily on analyzing the connection between the structure of a network and its ability to reproduce and process natural images. There are still many deep mysteries surrounding neural networks, and studying the synergy between a CNN architecture and input images may provide significant insight to this end.



(a) $\sigma = 0.01$ NLM
PSNR = 36.54

(b) $\sigma = 0.05$ NLM
PSNR = 33.33

(c) $\sigma = 0.1$ NLM
PSNR = 26.61

(d) $\sigma = 0.5$ NLM
PSNR = 9.22

(e) $\sigma = 0.01$ CBM3D
PSNR = 43.15

(f) $\sigma = 0.05$ CBM3D
PSNR = 35.67

(g) $\sigma = 0.1$ CBM3D
PSNR = 32.70

(h) $\sigma = 0.5$ CBM3D
PSNR = 19.17

(i) $\sigma = 0.01$ NS
PSNR = 29.24

(j) $\sigma = 0.05$ NS
PSNR = 28.38

(k) $\sigma = 0.1$ NS
PSNR = 28.56

(l) $\sigma = 0.5$ NS
PSNR = 18.17

Figure 4: Denoising results.

7. Acknowledgements

Many thanks to Vincent Sitzmann for his advice and mentorship on this project.

References

[1] M. S. C. Almeida and O. C. Mar. Deconvolving Images with Unknown Boundaries Using the Alternating Direction Method of Multipliers. pages 1–12.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. 3(1):1–122, 2011.

[3] A. Buades, B. Coll, and J.-M. J.-M. Morel. A non-local algorithm for image denoising. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2(0):60–65 vol. 2, 2005.

[4] A. Chakrabarti. A Neural Approach to Blind Motion Deblurring. pages 1–15.

[5] J. H. R. Chang, C.-l. Li, B. Poczos, B. V. Kumar, and A. C. Sankaranarayanan. One Network to Solve Them All Solving Linear Inverse Problems using Deep Projection Models. pages 1–12.

[6] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3D transform-domain collaborative filtering. *European Signal Processing Conference*, 16(8):145–149, 2007.

[7] M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly,

$\sigma \setminus$ Method	NLM	BM3D	NS
5×10^{-3}	31.3169	**47.1748**	28.7472
10^{-2}	31.2866	**42.4136**	28.6989
5×10^{-2}	30.3413	**32.6561**	28.1155
10^{-1}	26.8188	**29.1090**	27.12
5×10^{-1}	9.0870	**18.5934**	18.17

(a) Cameraman image (grayscale)

$\sigma \setminus$ Method	NLM	BM3D	NS
5×10^{-3}	32.1112	**46.4566**	29.6394
10^{-2}	32.0878	**41.3285**	29.1170
5×10^{-2}	31.3310	**32.5035**	29.5311
10^{-1}	27.4636	**29.7353**	28.7299
5×10^{-1}	9.0704	**20.9436**	19.0584

(b) Hill image (grayscale)

$\sigma \setminus$ Method	NLM	CBM3D	NS
5×10^{-3}	36.6710	**47.5239**	29.4907
10^{-2}	36.5363	**43.1490**	29.2364
5×10^{-2}	33.3319	**35.6655**	28.3844
10^{-1}	26.6053	**32.6968**	28.5641
5×10^{-1}	9.2171	**19.1744**	18.1657

(c) F-16 image (color)

$\sigma \setminus$ Method	NLM	CBM3D	NS
5×10^{-3}	28.2113	**46.1411**	21.2136
10^{-2}	28.1641	**40.4113**	20.9389
5×10^{-2}	26.8323	**29.2229**	21.0951
10^{-1}	22.9737	**25.8018**	21.1132
5×10^{-1}	9.0955	**18.0825**	17.2367

(d) Baboon image (color)

Table 2: PSNR values for denoising experiments. σ is the standard deviation of the additive noise.

CF \ Prior	TV	NLM	BM3D	NS
1	51.1815	47.8160	**56.1739**	18.9983
2	43.1129	42.3209	**50.2682**	23.4179
4	17.6178	33.6182	**42.1280**	13.5192
8	15.7656	16.4523	**20.9842**	10.9844

(a) Results for Stanford Logo image

CF \ Prior	TV	NLM	BM3D	NS
1	38.1158	40.1670	**41.6852**	23.3253
2	28.9712	31.2341	**31.8059**	20.7045
4	23.7100	25.4420	**26.2328**	19.0856
8	20.2931	20.1481	**21.6473**	16.7318

(b) Results for Barbara image

Table 3: PSNR results for single pixel camera image reconstruction with various image priors.

and R. G. Baraniuk. Single-Pixel Imaging via Compressive Sampling [. (March 2008):83–91.

[8] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein. Unrolled Optimization with Deep Priors. pages 1–11, 2017.

[9] C. Dong, C. C. Loy, and K. He. Image Super-Resolution Using Deep Convolutional Networks. pages 1–14.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1026–1034, 2015.

[11] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *Iclr*, pages 1–15, 2015.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.

[13] J. Kruse. Learning to Push the Limits of Efficient FFT-based Image Deconvolution. (October), 2017.

[14] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. 2016.

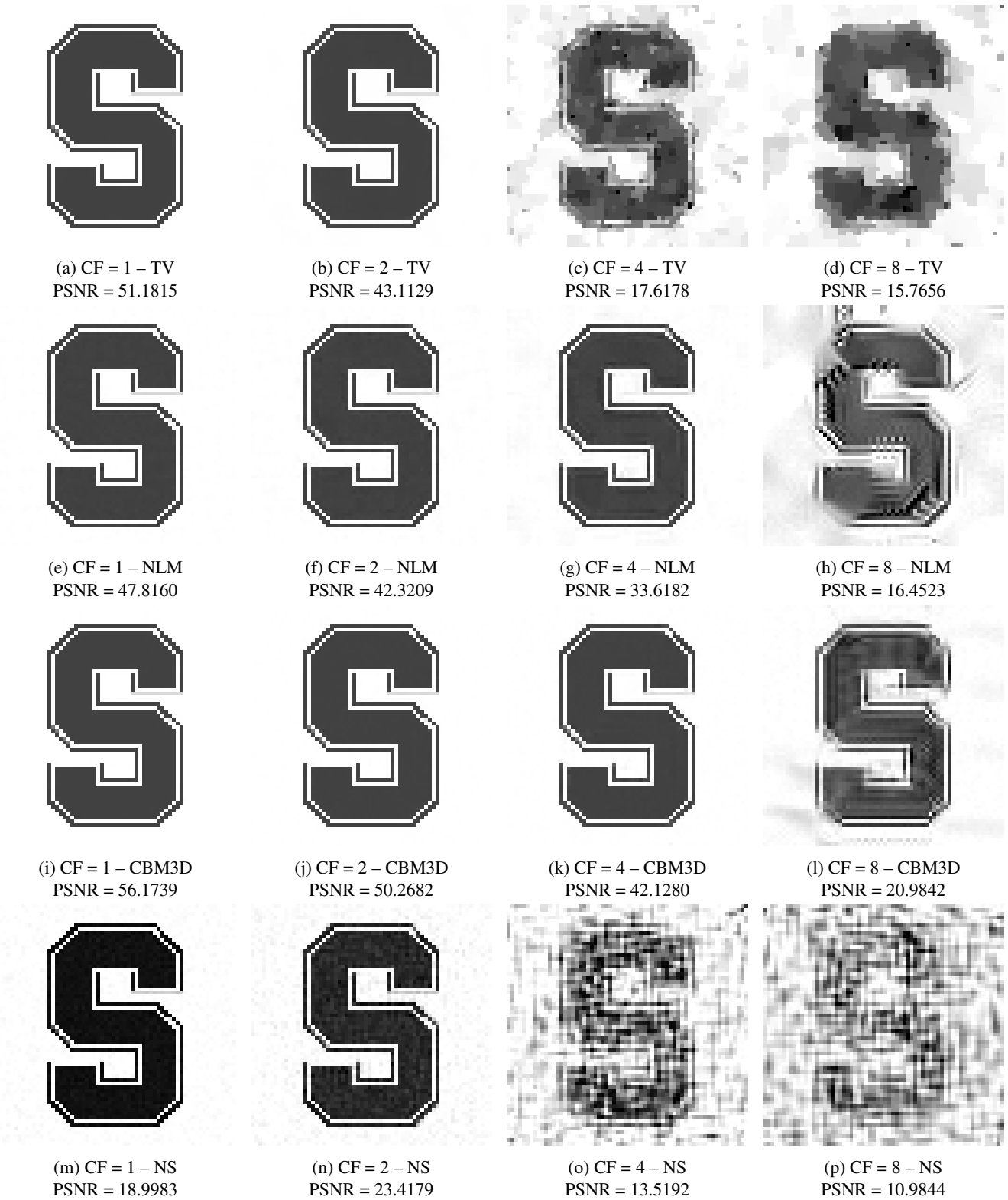
[15] T. Meinhardt. Learning Proximal Operators : Using Denoising Networks for Regularizing Inverse Imaging Problems. (October), 2017.

[16] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *Iclr*, pages 1–, 2014.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. 15:1929–1958, 2014.

[18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep Image Prior. 2017.

[19] J. Zhang, J. Pan, and W.-s. Lai. Learning Fully Convolutional Networks for Iterative Non-blind Deconvolution.



(a) CF = 1 – TV
PSNR = 51.1815

(b) CF = 2 – TV
PSNR = 43.1129

(c) CF = 4 – TV
PSNR = 17.6178

(d) CF = 8 – TV
PSNR = 15.7656

(e) CF = 1 – NLM
PSNR = 47.8160

(f) CF = 2 – NLM
PSNR = 42.3209

(g) CF = 4 – NLM
PSNR = 33.6182

(h) CF = 8 – NLM
PSNR = 16.4523

(i) CF = 1 – CBM3D
PSNR = 56.1739

(j) CF = 2 – CBM3D
PSNR = 50.2682

(k) CF = 4 – CBM3D
PSNR = 42.1280

(l) CF = 8 – CBM3D
PSNR = 20.9842

(m) CF = 1 – NS
PSNR = 18.9983

(n) CF = 2 – NS
PSNR = 23.4179

(o) CF = 4 – NS
PSNR = 13.5192

(p) CF = 8 – NS
PSNR = 10.9844

Figure 5: Single pixel camera image reconstruction results.