Compressive Sensing with Deep Image Prior, Learned Regularization, and Low Shot Learning

*Note: This is the final project report for the CSE 8803-Imaging with Data-Driven Models course at Georgia Tech

Reagan Kan
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
rkan3@gatech.edu
reagankan@gmail.com
May 2022

***Abstract*—Recent work in inverse problems has seen the use of deep neural networks as natural image priors for solving image inverse problems, such as denoising, in-painting, super-resolution, and compressive sensing. There are two ends to the spectrum of leveraging deep neural networks, (i) using an untrained neural network and (ii) training a neural network on thousands of training examples. Two recent works have explore the middle ground, training neural networks on a low shot dataset before solving inverse problems. Compressed Sensing with Deep Image Prior and Learned Regularization proposed a novel regularization method that is learned from a handful of data. Low Shot Learning with Untrained Neural Networks for Imaging Inverse Problems pre-trains a neural network on a small amount of data to improve image reconstruction performance. In this work, a hybrid low shot learning method that incorporates elements from both of these approaches is proposed and tested on the MNIST and STARE retinopathy datasets. The hybrid approach produces better image reconstructions when compared to both prior works while using a comparable amount of pre-training data.**

***Index Terms*—deep image prior, low shot learning, learned regularization**

I. Introduction

The project aims to solve the compressive sensing inverse problem which recovers an unknown image $x^* \in \mathbb{R}^n$ given a set of noisy, linear measurements $y \in \mathbb{R}^m$ defined by:

$$y = Ax^* + \eta \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$ is the measurement matrix and $\eta \in \mathbb{R}^m$ represents noise. Since $m < n$, the system of noisy linear equations is underdetermined and ill-posed.

Traditional methods assume image sparsity in hand-crafted bases $\Psi : \mathbb{R}^n \to \mathbb{R}^k$, such as the wavelet basis [1], and solve the following optimization problem:

$$\begin{aligned} z^* = argmin_{z \in \mathbb{R}^k} \frac{1}{2} ||y - A\Psi(x)||^2 + \lambda J(x), \\ x^* = \Psi^{-1}(z^*). \end{aligned} \tag{2}$$

where $z \in \mathbb{R}^k$ resides in the basis space and $J : \mathbb{R}^n \to \mathbb{R}$ is a regularization term which is controlled by a hyperparameter λ. Now, images x are recovered by taking the inverse transform Ψ^{-1} of z.

With rising computational resources, more work has been done to leverage deep neural network methods which tend to outperform these traditional methods. For example, several methods assume the reconstructed image falls within the range of the neural network and replace Ψ with a neural network $G(z; w)$, parameterized by network weights $w \in \mathbb{W}$. With the additional network parameter space $\mathbb{W}$, the optimization from (2) can occur over R^k, $\mathbb{W}$, or both. Ignoring the regularization term, we have:

$$\begin{aligned} w^* = argmin_{w \in \mathbb{W}} \frac{1}{2} ||y - AG(z; w)(x)||^2, \\ x^* = G(z; w^*) \\ z \text{ is randomly initialized.} \end{aligned} \tag{3}$$

$$\begin{aligned} z^* = argmin_{z \in \mathbb{R}^k} \frac{1}{2} ||y - AG(z; w)(x)||^2, \\ x^* = G(z^*; w) \\ w \text{ is randomly initialized.} \end{aligned} \tag{4}$$

$$\begin{aligned} w^*, z^* = argmin_{w \in \mathbb{W}, z \in \mathbb{R}^k} \frac{1}{2} ||y - AG(z; w)(x)||^2, \\ x^* = G(z^*; w^*). \end{aligned} \tag{5}$$

Broadly speaking, these methods fall under three categories:

- pre-training $G(z; w)$ with large amounts of data
- pre-training $G(z; w)$ on low shot examples
- $G(z; w)$ is untrained

Compressive Sensing using Generative Models (CSGM) [11] employs the first method and pre-trains Deep Convolutional Generative Adversarial Networks (DCGAN) and Variational Autoencoders to generate images by using hundreds of thousands of training images. In contrast, Deep Image Prior (DIP) [2] employs the third method, which requires zero pre-training of the U-net neural network. Although DIP does not solve the compressive sensing problem, it still solve

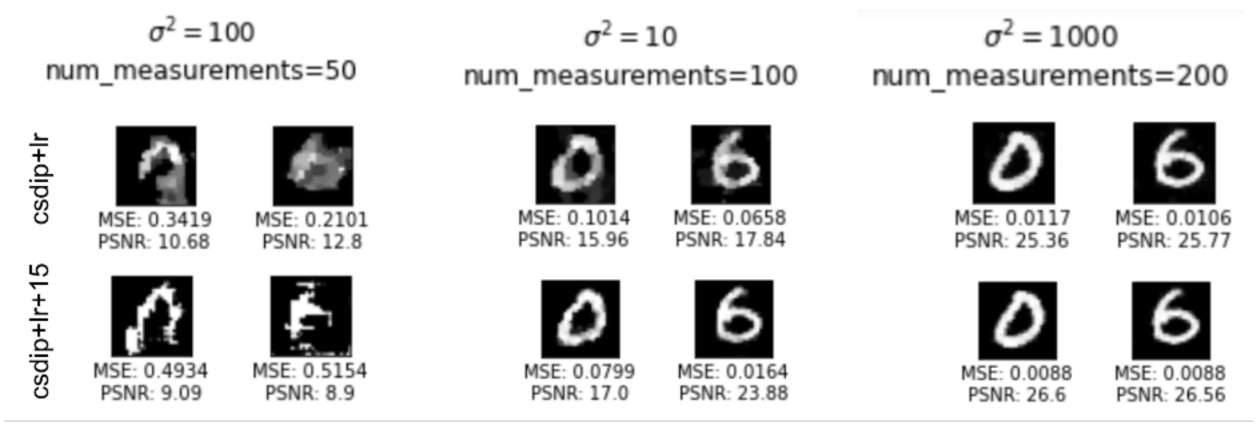


Fig. 1. MNIST Reconstructions. It is unclear if PSNR changes are due to methodology design or latent code initialization

inverse imaging problems and the approaches can be applied to compressive sensing.

In fact, this is exactly what the authors of Compressed Sensing with Deep Image Prior and Learned Regularization [3] did; one of their contributions adapted DIP for compressive sensing. Perhaps more interestingly, they proposed a learned regularization (LR) term for the network parameters w, which is learned from a small set of low shot training examples. In a similar vein, Low Shot Learning with Untrained Neural Networks for Imaging Inverse Problems [4] leveraged a small number of training examples to pre-train the network before image reconstruction.

Both of these methods fall under the second category and form the basis of this paper. Since both approaches utilize low shot training examples before test time image reconstruction, this work proposes a hybrid low shot learning approach (Section III) and demonstrates that it can improve reconstruction results on the STARE retinopathy dataset (Section IV).

II. Problem Statement

The compressive sensing inverse problem aims to recovers an unknown image $x^* \in \mathbb{R}^n$ given compressed $(m < n)$ linear measurements $y = Ax^* + \eta \in \mathbb{R}^m$ where $A \in \mathbb{R}^{m \times n}$ is the measurement matrix and $\eta \in \mathbb{R}^m$ denotes noise.

It is assumed that the image x^* belongs in a data distribution $\mathcal{D}$ and there is access to small number of example images $\{x_i\}_{i=1}^S$ from the same distribution. In other words, $x^* \sim \mathcal{D}$ and $x_i \sim \mathcal{D}$ s.t. $1 \leq i \leq S$.

III. Methodology

A. Learned Regularization

The LR assumes the neural network parameters to belong to a Gaussian distribution, as they are randomly initialized with Gaussian weights. Then, using a low shot number of examples S (around 10), the DIP problem is solved, generating S number of "optimal" network parameters $\{w_i\}_{i=1}^S$. This sample of network weights is used to compute the mean μ_w and covariance Σ_w matrices in order to represent an "optimal" Gaussian network parameter distribution $\mathcal{N}(\mu_w, \Sigma_w)$. Then, the LR term is defined as:

$$LR(w) = (w - \mu_w)^T \Sigma_w^{-1} (w - \mu_w) \tag{6}$$

which effectively measures the distance between the neural network parameters (w) and the low shot estimate for the optimal network parameter Gaussian distribution $\mathcal{N}(\mu_w, \Sigma_w)$. The exact procedure for estimating parameters μ_w and Σ_w is detailed in Algorithm 1 in the appendix of [3].

B. Low Shot Pre-training

The pre-training procedure follows directly from [4]; before solving the reconstruction optimization, use $\{x_i\}_{i=1}^S$ low shot example images to initialize neural network parameters $\hat{w}$ and latent codes $\{\hat{z}_i\}_{i=1}^S$. The following gives the optimization objective with $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ being the $\ell-2$ loss function:

$$argmin_{w, z_1, \ldots, z_S} \frac{1}{S} \sum_{i=1}^{S} \mathcal{L}(G(z_i; w), x_i) \tag{7}$$

C. Reconstruction Optimization: Step 1

Similar to [4], the reconstruction optimization is split into two steps. The first step remains the same; optimize for the latent code $\hat{z}$ that is specific to the image to be reconstructed using the pre-trained initializations $\hat{w}$ and $\{\hat{z}_i\}_{i=1}^S$:

$$argmin_z \frac{1}{2} ||AG(z; \hat{w}) - y||_2^2 \tag{8}$$

At the start of this optimization, the latent code is initialized using a random sample from a multivariate Gaussian distribution fit of $\{\hat{z}_i\}_{i=1}^S$.

D. Reconstruction Optimization: Step 2

Here, the proposed method diverges from [4] in two ways. First, rather than optimize over the latent space and the network parameter space, the proposed method optimizes solely over the network parameter space. This is done for two reasons: (i) to allow a fair comparison against [2], [3] which only optimize for the network parameters and (ii) to speed up the optimization. The second modification comes in

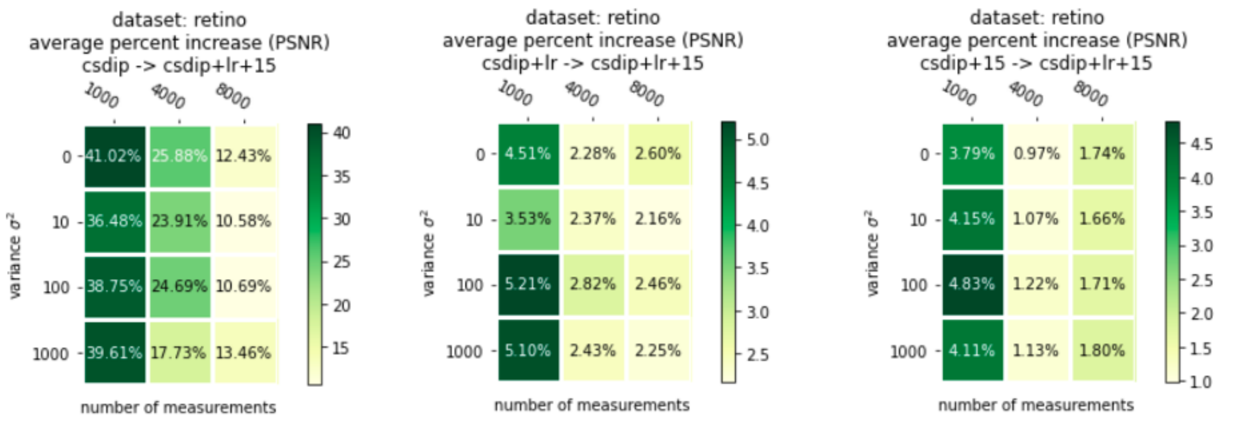


Fig. 2. PSNR Percent Increase (averaged over 5 test images)

the addition of two regularization penalty terms from [3], the learned regularization and total variation (TV) [10]. The final optimization problem is:

$$min_w \frac{1}{2}||AG(\hat{z};w)-y||_2^2+\lambda_T TV(G(\hat{z};w))+\lambda_L LR(w) \quad (9)$$

where λ_T and λ_L are hyperparameters that control the influence of the regularization terms (see IV-A-Implementation Details for specific values).

IV. Results

A. Experiment Setup

Datasets: Following [3], the first 100 test images from the MNIST dataset [5] and the first 20 images from the STARE retinopathy dataset [6] are used to evaluate the proposed method in the experiments. Pixel intensity values range from -1 to 1.

Measurement Matrix: The conventional measurement matrix from compressive sensing literature is used. $A \in \mathbb{R}^{m\times n}$ is Gaussian i.i.d sampled from $\mathcal{N}(0, \frac{1}{m})$, where n is the number of pixels in the ground truth image and m is the number of measurements. For MNIST ($n = 784$), $m \in [50, 100, 200]$. For STARE ($n = 16384$), $m \in [1000, 4000, 8000]$.

Noise: The noise vector η has entries drawn from $\mathcal{N}(0, \frac{\sigma_\eta^2}{m})$ where $\sigma_\eta^2 \in [0, 10, 100, 1000]$

Learned Regularization: The experiments use the pre-trained LR parameters that are available in the GitHub [7].

Low Shot Pre-training: $S \in [0, 10, 50, 100]$ for MNIST images and $S \in [0, 15]$ for STARE images. 5 hold-out images from both datasets were used for test-time reconstruction evaluation.

Baselines: The proposed method is compared against three baselines: untrained CSDIP (DIP with a DCGAN architecture as in [4]), untrained CSDIP+LR (DIP with LR enabled as in [3]), and DIP pre-trained on S low shots, or CSDIP+S, (same as [4] except for the modification to step 2 of the reconstruction optimization as in the proposed method (see III-D)). Since [3] requires it, all baselines will be use the TV regularization in the reconstruction optimization loss for a fair comparison across methods.

Metrics: Reconstruction performance is quantitatively measured using Peak Signal-to-Noise Ratio (PSNR).

Implementation Details: The learned regularization parameters used in the experiments are the exact same as those found in the GitHub repository [7] for [3]. The low-shot pre-training hyperparameters are the same as in [4]. Specifically, (7) is optimized using Adam optimizer [8] for 50,000 iterations with a learning rate of 10^{-3}. Then, (8) is solved using Adam for 1250 iterations with a learning rate of 5×10^{-2}. Finally, (9) is solved using RMSProp optimizer [9] for 1000 iterations with a learning rate of 10^{-3} and momentum of 0.9. For MNIST, $\lambda_T = 10^{-2}$ and $\lambda_L = 0$. For STARE, $\lambda_T = 2\times10^{-2}$ and $\lambda_L = 1000$. These values were found using a gridsearch by the authors of [3].

Additional implementation details, like neural network architectures, are in the code, which is publicly available on a GitHub repository: https://github.com/reagankan/compsensing_dip/tree/python3

B. Experiment Results

1) MNIST: As noted in [3], the reconstructions are highly sensitive to latent code initializations since the optimization objective tends to have local minima and the DCGAN latent dimensions (128) are so close to the number of pixels $n = 784$. Even with low shot pre-training and the extra latent code optimization step, reconstructions varied with different latent code initialization. For this reason, method comparison using MNIST dataset is forgone. Fig. 1 shows some example MNIST reconstructions.

2) STARE: In summary, the proposed method beats all baselines with a 10.58% to 41.02% average PSNR percent increase over CSDIP, 2.16% to 5.21% increase over CSDIP+LR, and 0.97% to 4.83% increase over CSDIP+15. The benefit is most prominent when the measurements are noisy and/or compressed. Fig. 2 shows all average PSNR percent differences.

To examine how each method responds to varying noise and compression levels, PSNR is plotted against each of these dimensions in Fig. 3. All methods can handle different levels of noise. However, CSDIP is most severely impacted by high compression rates. Finally, example reconstructions with corresponding PSNR scores are showed in Fig. 4.

CONCLUSIONS

This work presents a hybrid low shot learning method for Deep Image Prior that outperforms previous low shot learning approaches at the compressive sensing inverse problem. Improvement tends to be strongest for noisy and compressed measurements. Code for experiments is released publicly on GitHub https://github.com/reagankan/compsensing_dip/tree/python3.

REFERENCES

[1] David Donoho, Michael Lustig, and John M. Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. Magnetic Resonance in Medicine, 58(6):1182–1195, 2007.

[2] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. arXiv preprint, arXiv:1711.10925, 2017.

[3] Dave Van Veen, Ajil Jalal, Mahdi Soltanolkotabi, Eric Price, Sriram Vishwanath, and Alexandros G. Dimakis. Compressed sensing with deep image prior and learned regularization. arXiv preprint, arXiv:1806.06438, 2019.

[4] Oscar Leong and Wesam Sakla. Low shot learning with untrained neural networks for imaging inverse problems. arXiv preprint arXiv:1910.10797, 2019.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[6] AD Hoover, Valentina Kouznetsova, and Michael Goldbaum. Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. IEEE Transactions on Medical imaging, 19(3):203–210, 2000.

[7] https://github.com/davevanveen/compsensing_dip

[8] Diederik Kingma and Jimmy Ba. Adam. Adam: A method for stochastic optimization. arXiv preprint, arXiv:1412.6980, 2014.

[9] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2):26–31, 2012.

[10] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. Physica D: nonlinear phenomena, 60(1-4):259–268, 1992.

[11] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. arXiv preprint arXiv:1703.03208, 2017.

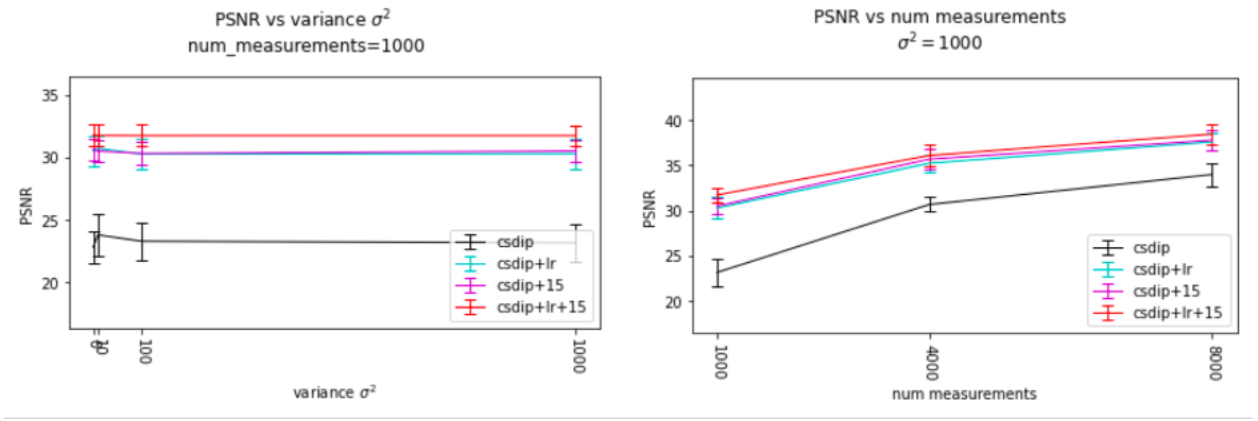


Fig. 3. Left: PSNR vs Noise. All methods are robust against noise. Right: PSNR vs Number of Measurements. CSDIP degrades the fastest with compression.

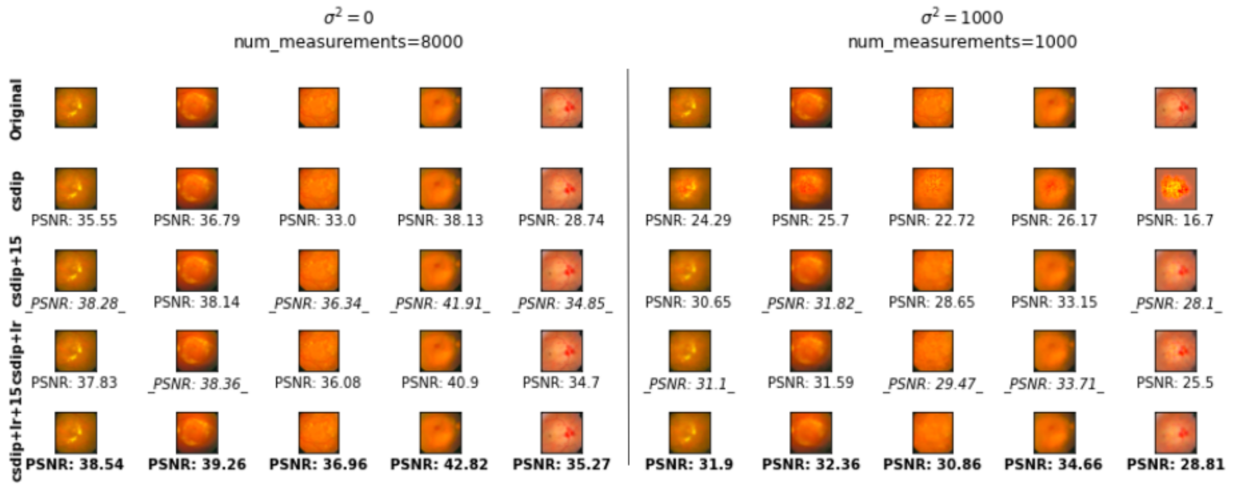
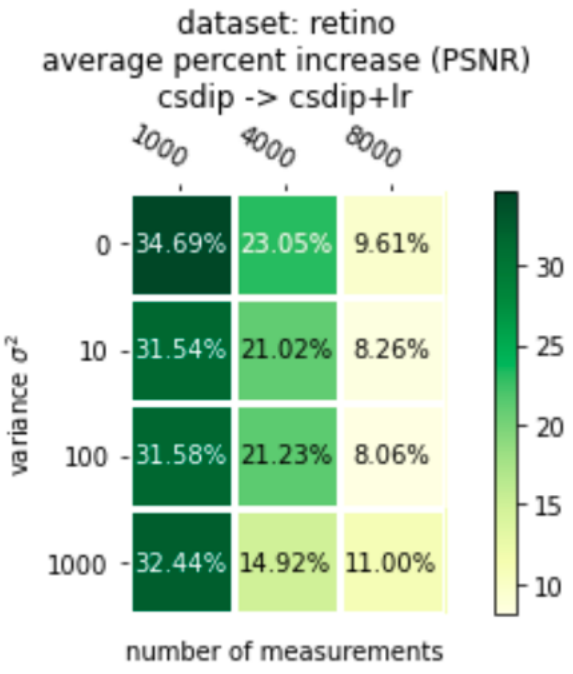


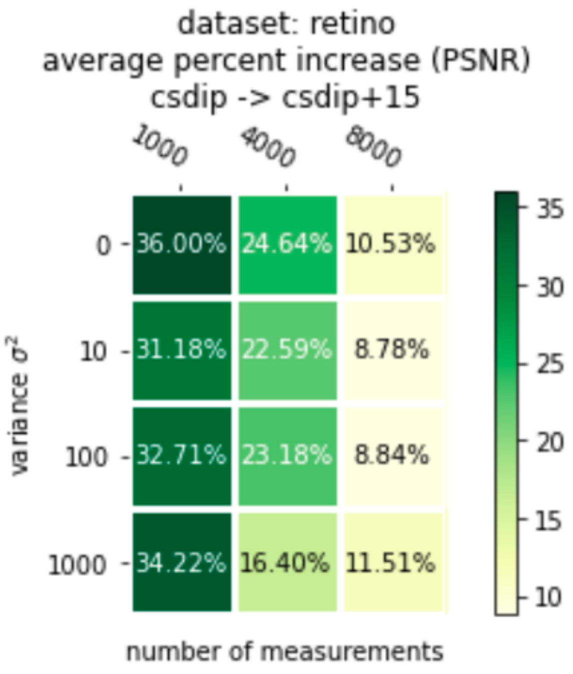
Fig. 4. STARE reconstructions with PSNR. Bold denotes the best score and _underline_ denotes 2nd best. Left: least noise/compression. Right: most noise/compression.

V. Appendix

A. Ablation Study: CSDIP vs CSDIP+LR vs CSDIP+15

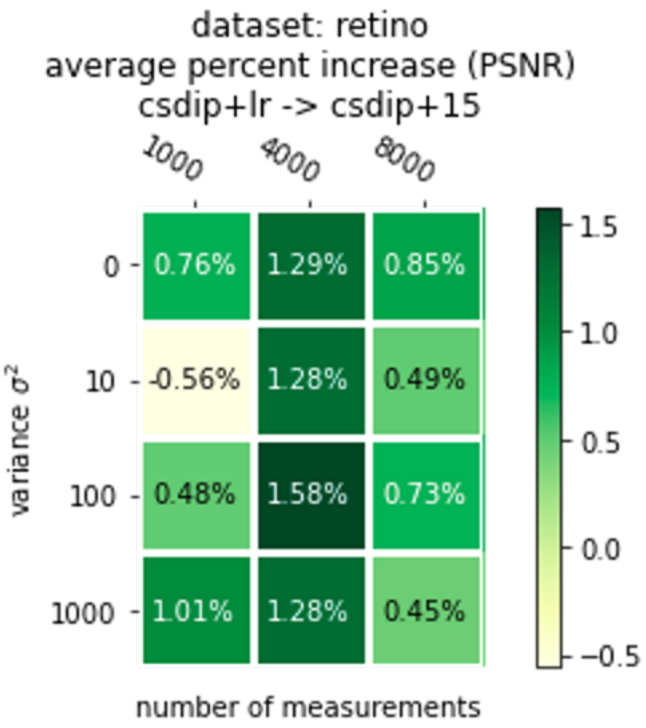
To examine the contribution of LR and Low Shot Pre-training separately, the following figure shows the average PSNR percent increase comparing CSDIP+LR against CSDIP and comparing CSDIP+15 against CSDIP.





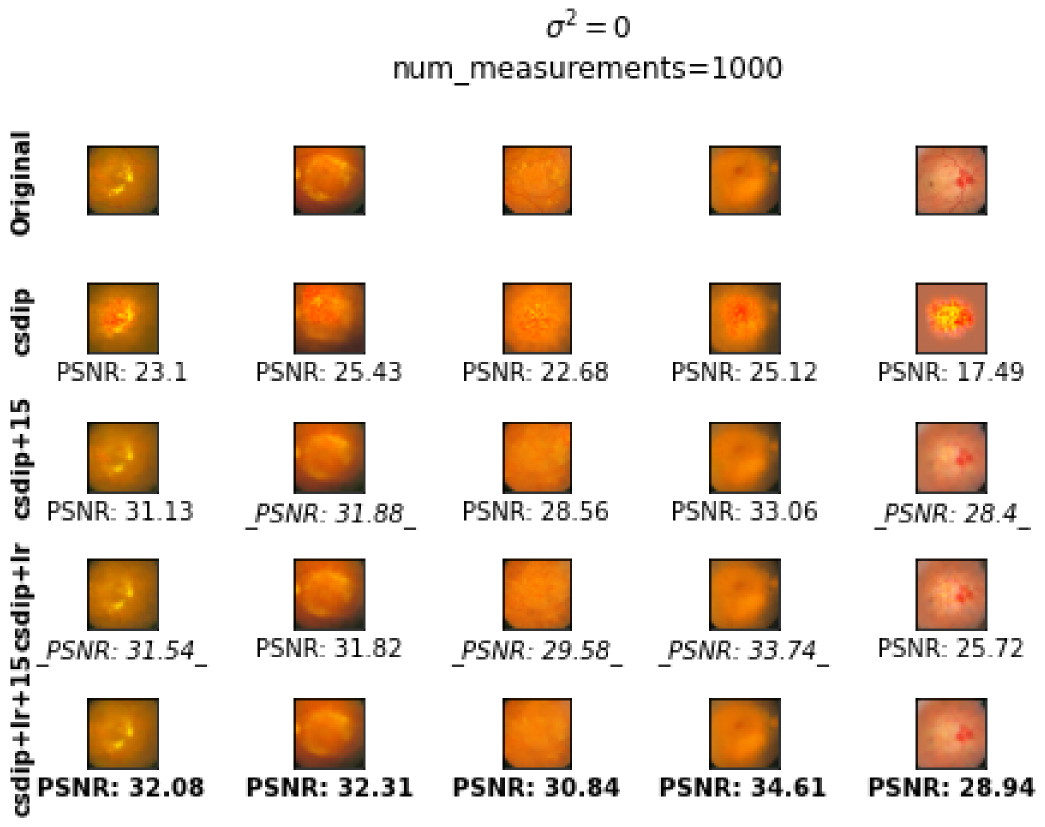
Left: CSDIP vs CSDIP+LR. Right: CSDIP vs CSDIP+15

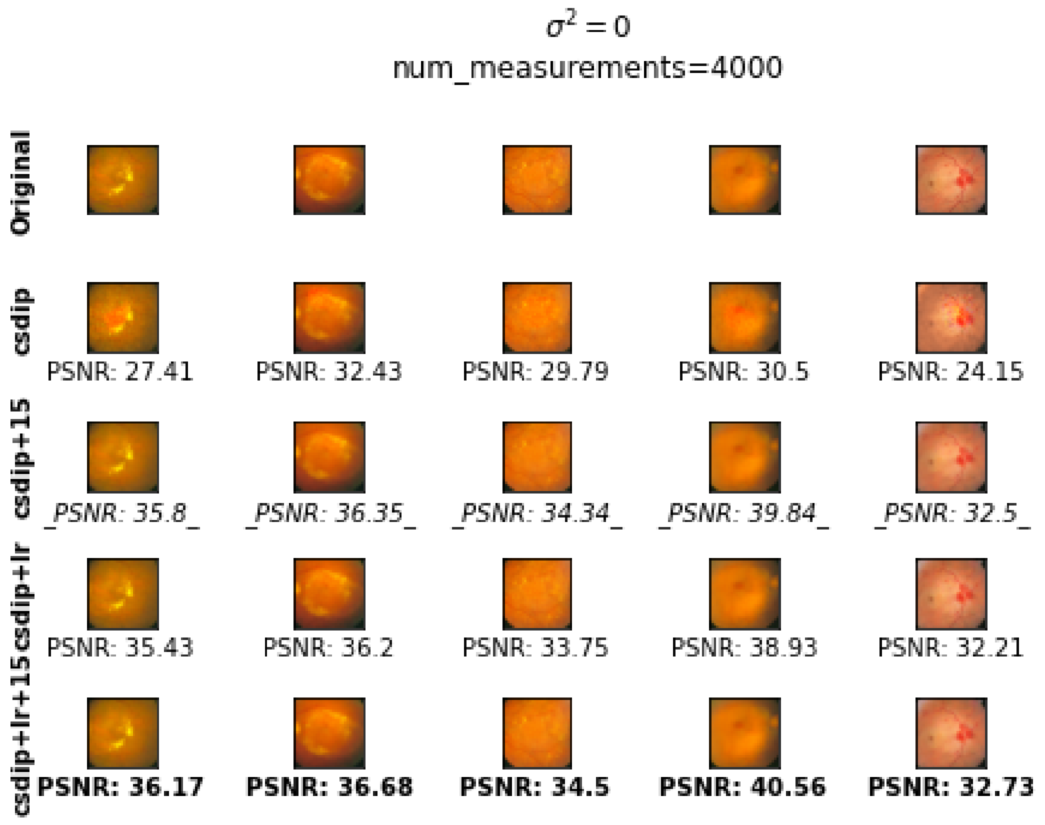
Except for when $\sigma^2 = 10$ and $m = 1000$, CSDIP+15 provides the most reconstruction improvement. We can see the relative PSNR percent increase in the next figure.

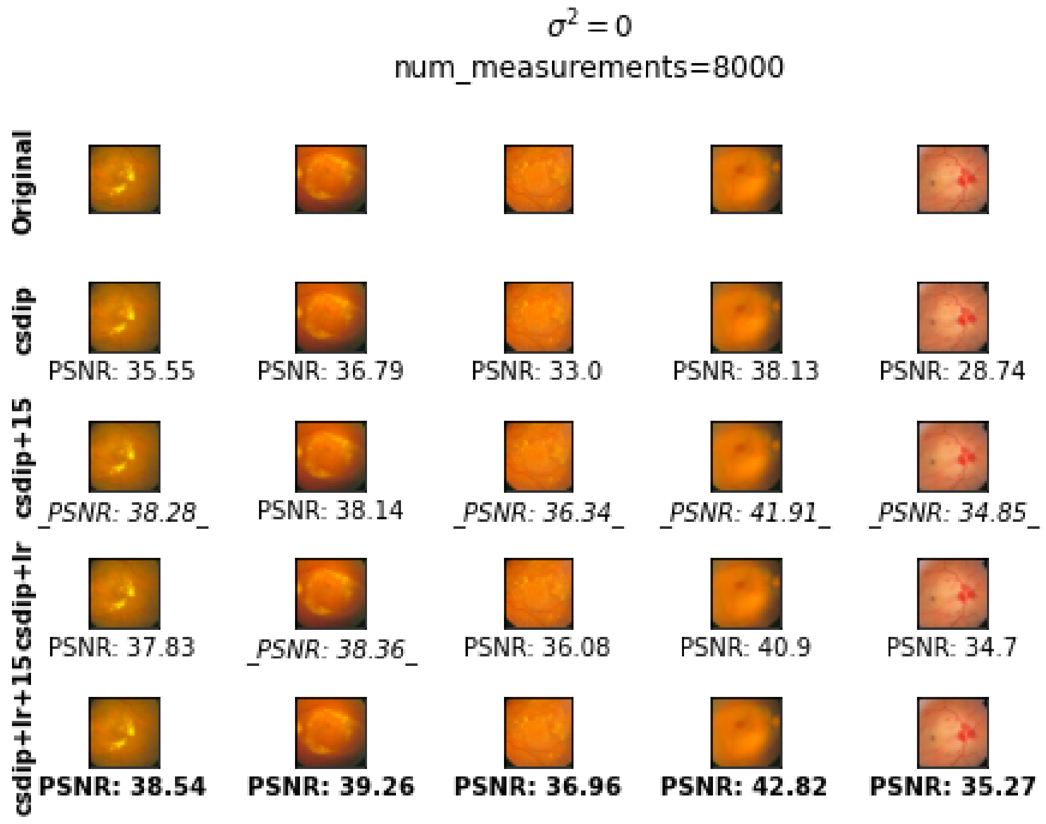


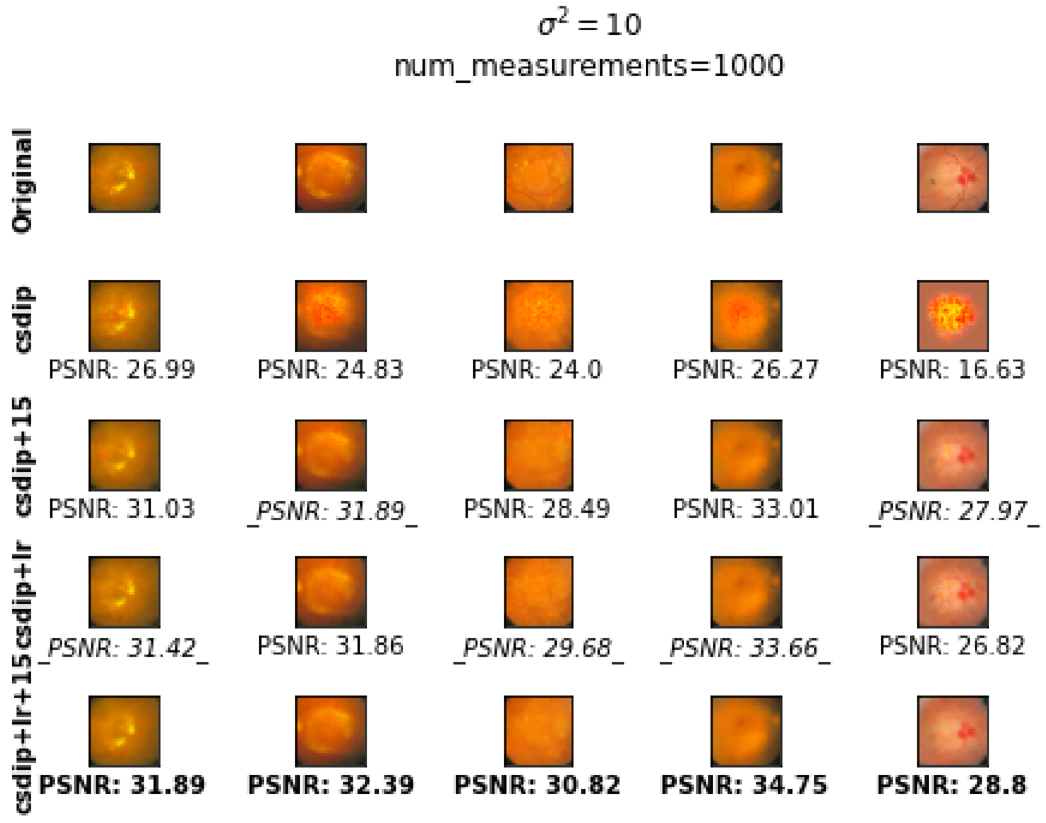
CSDIP+LR vs CSDIP+15

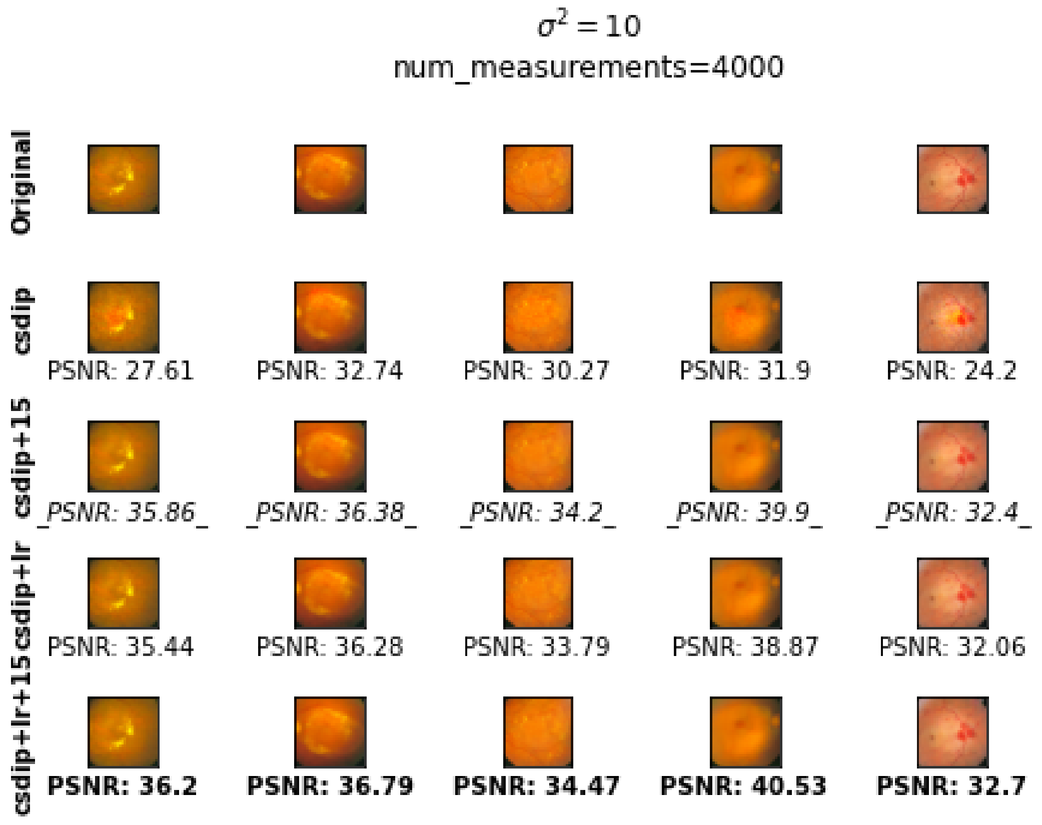
B. All STARE Reconstructions

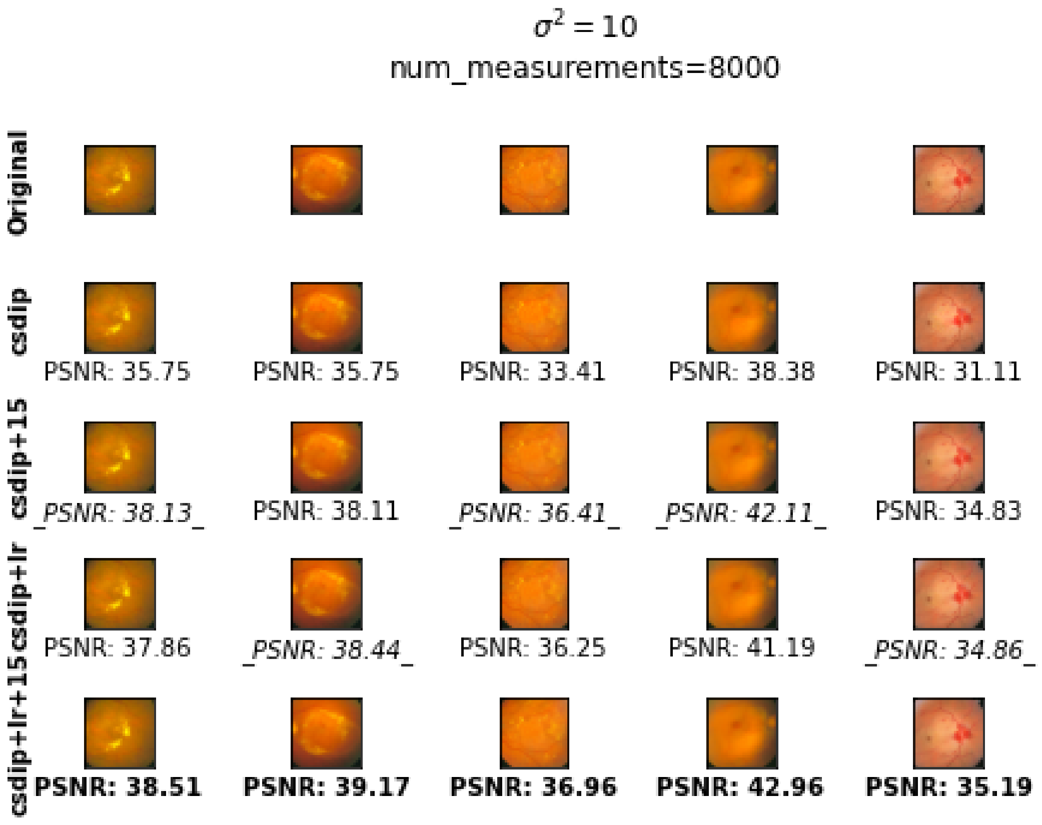


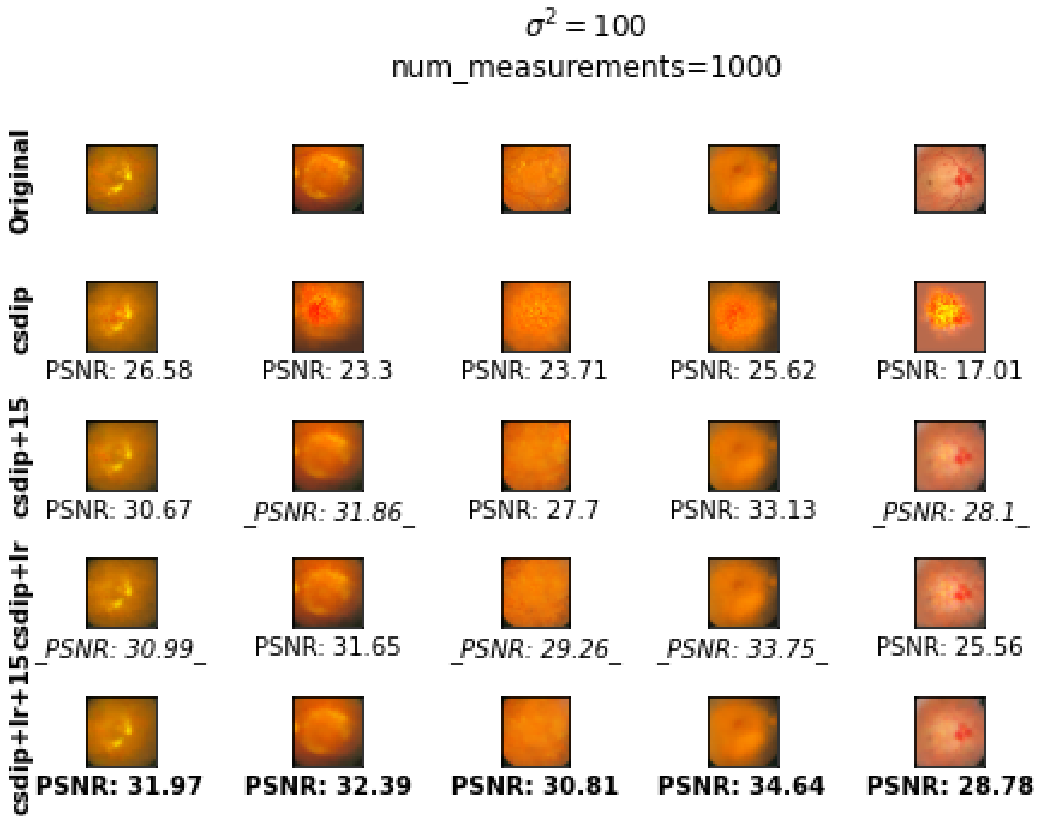


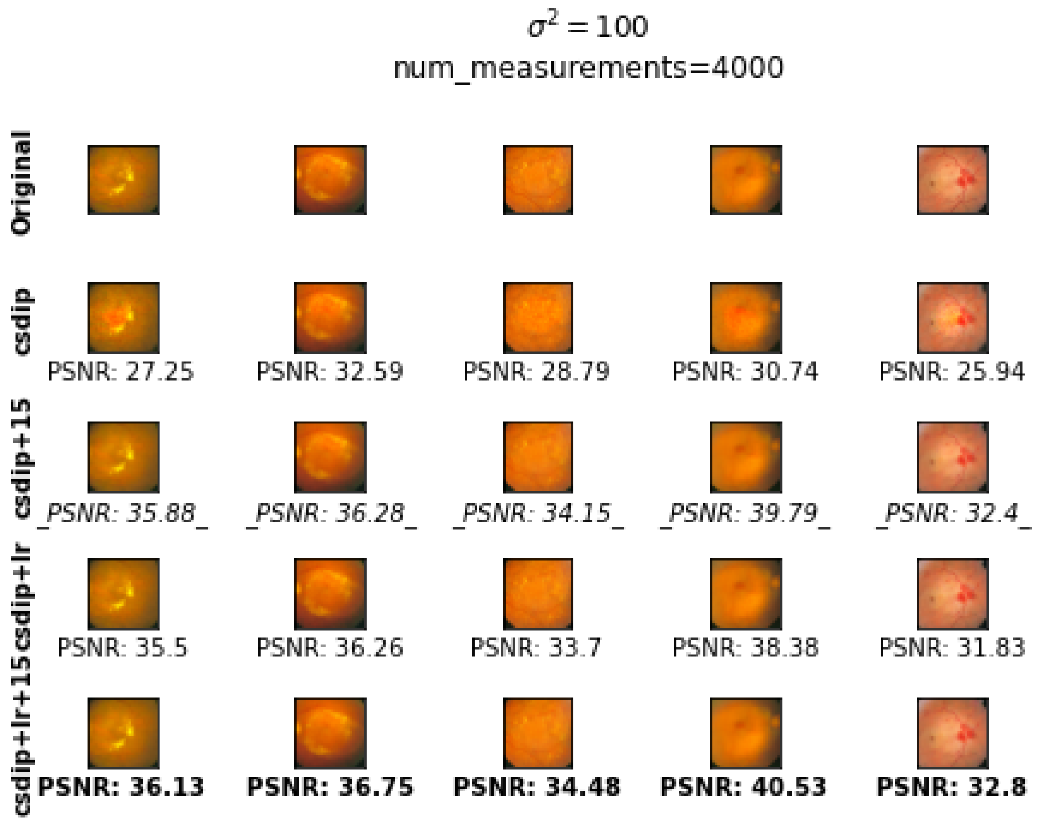

$\sigma^2 = 0$
num_measurements=8000
Original
csdip
PSNR: 35.55
PSNR: 36.79
PSNR: 33.0
PSNR: 38.13
PSNR: 28.74
csdip+15
PSNR: 38.28
PSNR: 38.14
PSNR: 36.34
PSNR: 41.91
PSNR: 34.85
csdip+lr
PSNR: 37.83
PSNR: 38.36
PSNR: 36.08
PSNR: 40.9
PSNR: 34.7
csdip+lr+15
PSNR: 38.54
PSNR: 39.26
PSNR: 36.96
PSNR: 42.82
PSNR: 35.27

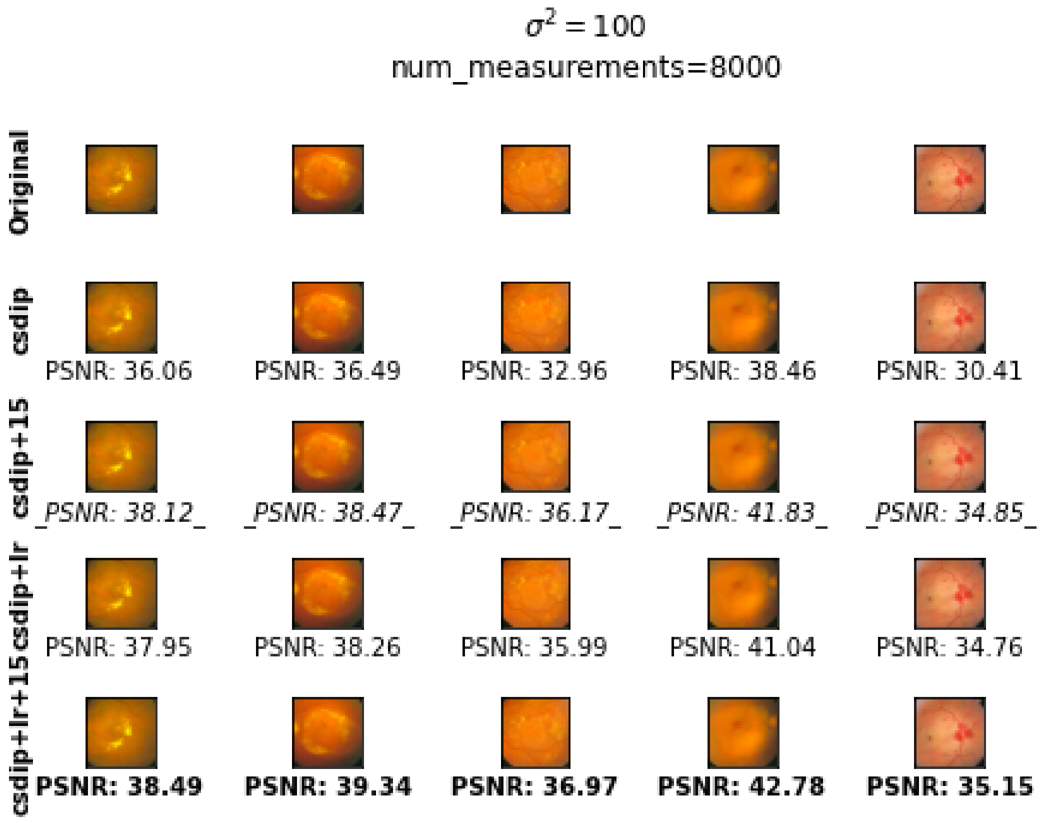

$\sigma^2 = 10$
num_measurements=1000
Original
csdip
PSNR: 26.99
PSNR: 24.83
PSNR: 24.0
PSNR: 26.27
PSNR: 16.63
csdip+15
PSNR: 31.03
PSNR: 31.89
PSNR: 28.49
PSNR: 33.01
PSNR: 27.97
csdip+lr
PSNR: 31.42
PSNR: 31.86
PSNR: 29.68
PSNR: 33.66
PSNR: 26.82
csdip+lr+15
PSNR: 31.89
PSNR: 32.39
PSNR: 30.82
PSNR: 34.75
PSNR: 28.8

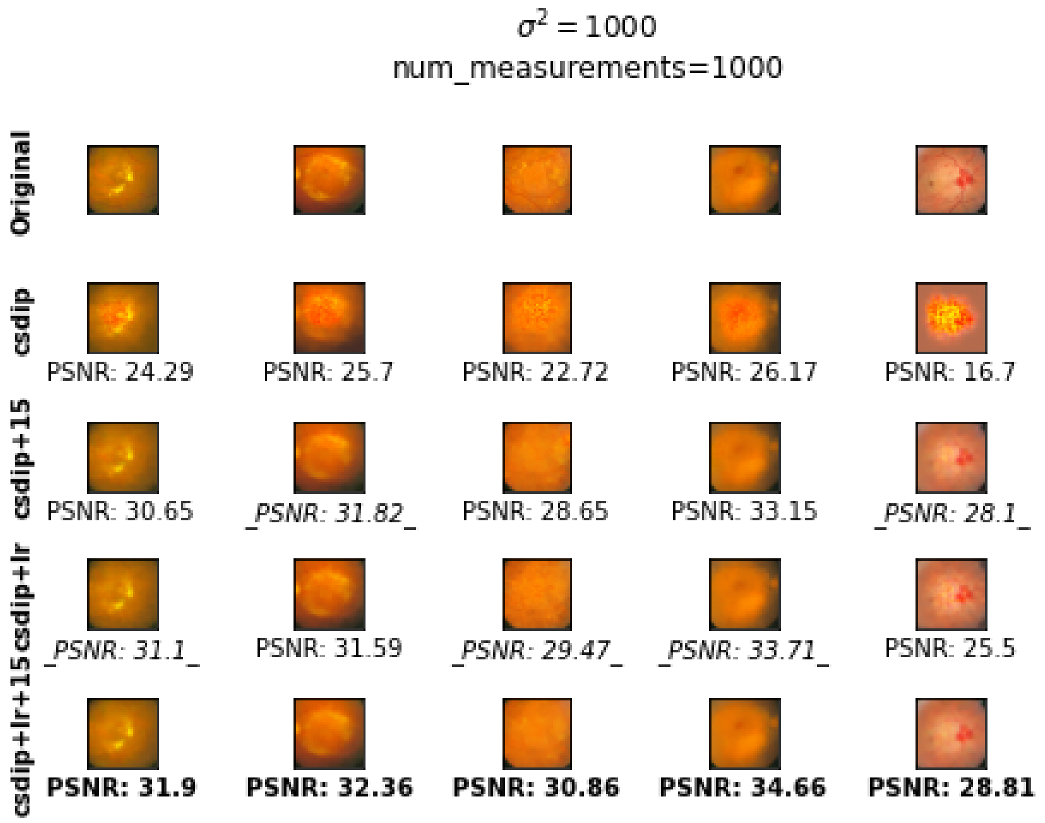

$\sigma^2 = 10$
num_measurements=4000
Original
csdip
PSNR: 27.61
PSNR: 32.74
PSNR: 30.27
PSNR: 31.9
PSNR: 24.2
csdip+15
PSNR: 35.86
PSNR: 36.38
PSNR: 34.2
PSNR: 39.9
PSNR: 32.4
csdip+lr
PSNR: 35.44
PSNR: 36.28
PSNR: 33.79
PSNR: 38.87
PSNR: 32.06
csdip+lr+15
PSNR: 36.2
PSNR: 36.79
PSNR: 34.47
PSNR: 40.53
PSNR: 32.7

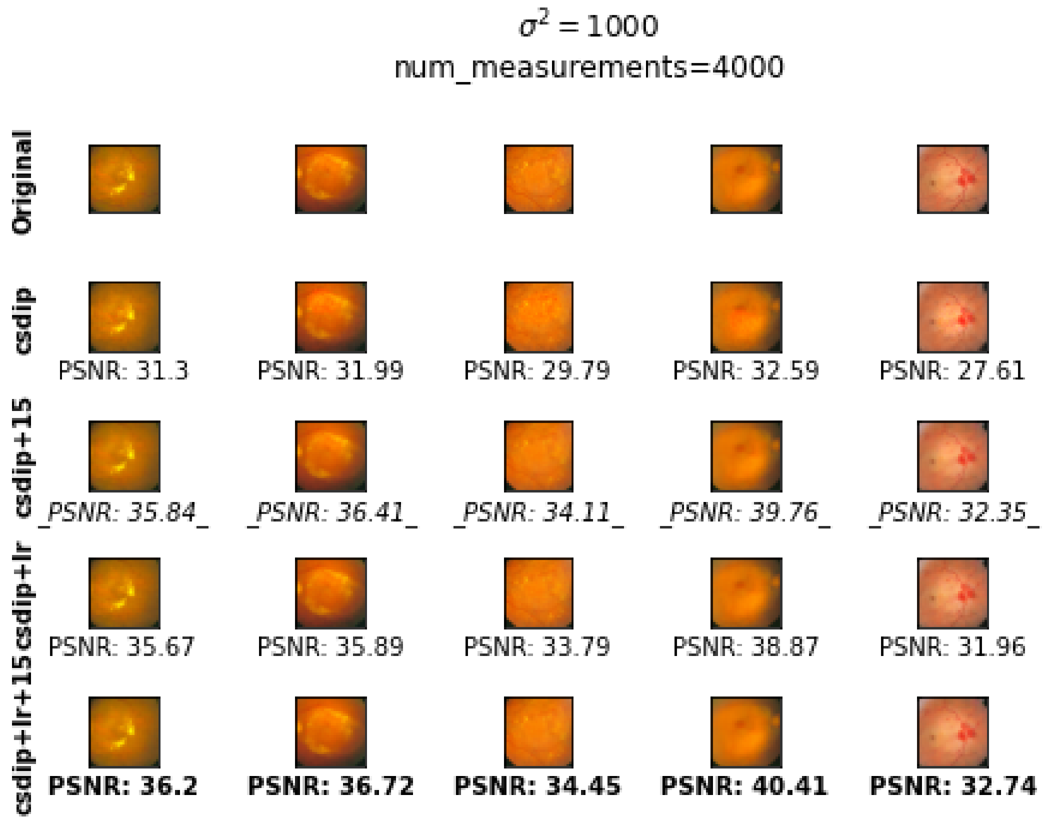

$\sigma^2 = 10$
num_measurements=8000
Original
csdip
PSNR: 35.75
PSNR: 35.75
PSNR: 33.41
PSNR: 38.38
PSNR: 31.11
csdip+15
PSNR: 38.13
PSNR: 38.11
PSNR: 36.41
PSNR: 42.11
PSNR: 34.83
csdip+lr
PSNR: 37.86
PSNR: 38.44
PSNR: 36.25
PSNR: 41.19
PSNR: 34.86
csdip+lr+15
PSNR: 38.51
PSNR: 39.17
PSNR: 36.96
PSNR: 42.96
PSNR: 35.19

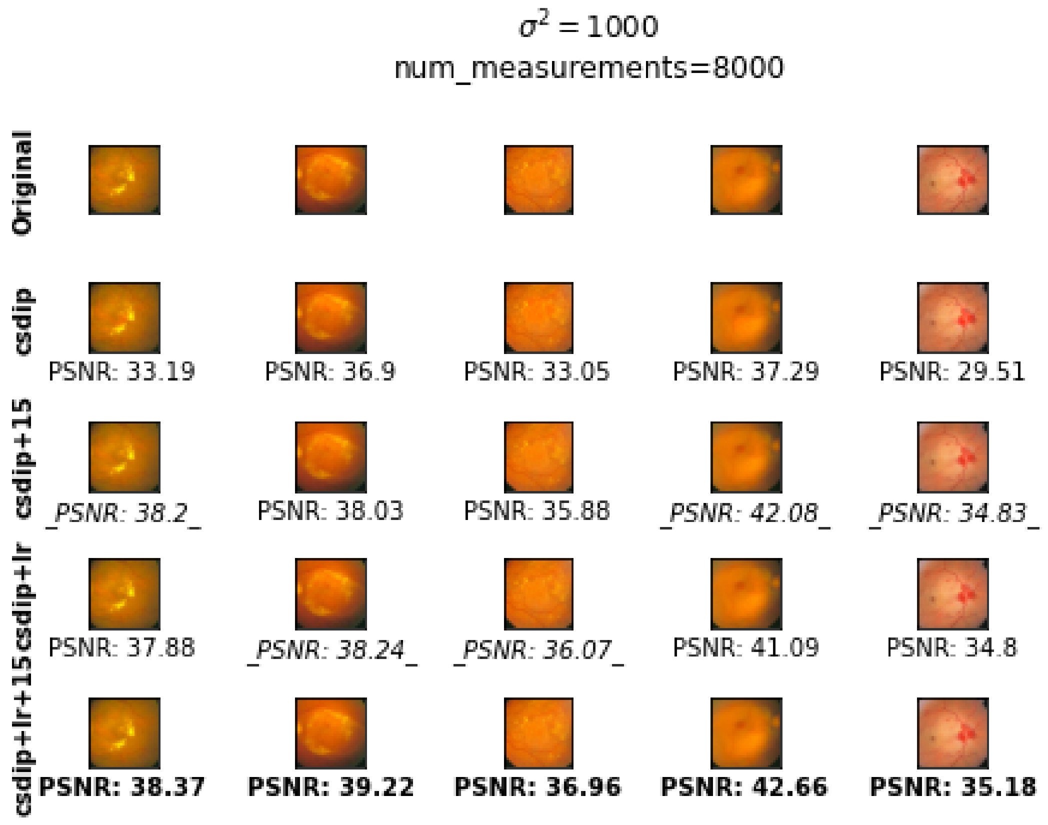

$\sigma^2 = 100$
num_measurements=1000
Original
csdip
PSNR: 26.58
PSNR: 23.3
PSNR: 23.71
PSNR: 25.62
PSNR: 17.01
csdip+15
PSNR: 30.67
PSNR: 31.86
PSNR: 27.7
PSNR: 33.13
PSNR: 28.1
csdip+lr
PSNR: 30.99
PSNR: 31.65
PSNR: 29.26
PSNR: 33.75
PSNR: 25.56
csdip+lr+15
PSNR: 31.97
PSNR: 32.39
PSNR: 30.81
PSNR: 34.64
PSNR: 28.78


$\sigma^2 = 100$
num_measurements=4000
Original
csdip
PSNR: 27.25
PSNR: 32.59
PSNR: 28.79
PSNR: 30.74
PSNR: 25.94
csdip+15
PSNR: 35.88
PSNR: 36.28
PSNR: 34.15
PSNR: 39.79
PSNR: 32.4
csdip+lr
PSNR: 35.5
PSNR: 36.26
PSNR: 33.7
PSNR: 38.38
PSNR: 31.83
csdip+lr+15
PSNR: 36.13
PSNR: 36.75
PSNR: 34.48
PSNR: 40.53
PSNR: 32.8


$\sigma^2 = 100$
num_measurements=8000
Original
csdip
PSNR: 36.06
PSNR: 36.49
PSNR: 32.96
PSNR: 38.46
PSNR: 30.41
csdip+15
PSNR: 38.12
PSNR: 38.47
PSNR: 36.17
PSNR: 41.83
PSNR: 34.85
csdip+lr
PSNR: 37.95
PSNR: 38.26
PSNR: 35.99
PSNR: 41.04
PSNR: 34.76
csdip+lr+15
PSNR: 38.49
PSNR: 39.34
PSNR: 36.97
PSNR: 42.78
PSNR: 35.15


$\sigma^2 = 1000$
num_measurements=1000
Original
csdip
PSNR: 24.29
PSNR: 25.7
PSNR: 22.72
PSNR: 26.17
PSNR: 16.7
csdip+15
PSNR: 30.65
PSNR: 31.82
PSNR: 28.65
PSNR: 33.15
PSNR: 28.1
csdip+lr
PSNR: 31.1
PSNR: 31.59
PSNR: 29.47
PSNR: 33.71
PSNR: 25.5
csdip+lr+15
PSNR: 31.9
PSNR: 32.36
PSNR: 30.86
PSNR: 34.66
PSNR: 28.81


$\sigma^2 = 1000$
num_measurements=4000
Original
csdip
PSNR: 31.3
PSNR: 31.99
PSNR: 29.79
PSNR: 32.59
PSNR: 27.61
csdip+l5
PSNR: 35.84
PSNR: 36.41
PSNR: 34.11
PSNR: 39.76
PSNR: 32.35
csdip+lr
PSNR: 35.67
PSNR: 35.89
PSNR: 33.79
PSNR: 38.87
PSNR: 31.96
csdip+lr+l5
PSNR: 36.2
PSNR: 36.72
PSNR: 34.45
PSNR: 40.41
PSNR: 32.74


$\sigma^2 = 1000$
num_measurements=8000
Original
csdip
PSNR: 33.19
PSNR: 36.9
PSNR: 33.05
PSNR: 37.29
PSNR: 29.51
csdip+l5
PSNR: 38.2
PSNR: 38.03
PSNR: 35.88
PSNR: 42.08
PSNR: 34.83
csdip+lr
PSNR: 37.88
PSNR: 38.24
PSNR: 36.07
PSNR: 41.09
PSNR: 34.8
csdip+lr+l5
PSNR: 38.37
PSNR: 39.22
PSNR: 36.96
PSNR: 42.66
PSNR: 35.18