Optimal Eye Surgeon: Finding image priors through sparse generators at initialization

Avrajit Ghosh [1] Xitong Zhang [1] Kenneth Sun [2] Qing Qu [2] Saiprasad Ravishankar [1 3] Rongrong Wang [1]

Abstract

We introduce *Optimal Eye Surgeon* (OES), a framework for pruning and training deep image generator networks. Typically, untrained deep convolutional networks, which include image sampling operations, serve as effective image priors (Ulyanov et al., 2018). However, they tend to overfit to noise in image restoration tasks due to being overparameterized. OES addresses this by adaptively pruning networks at random initialization to a level of underparameterization. This process effectively captures low-frequency image components *even without training, by just masking*. When trained to fit noisy image, these pruned subnetworks, which we term *Sparse-DIP*, resist overfitting to noise. This benefit arises from underparameterization and the regularization effect of masking, constraining them in the manifold of image priors (Figure-3). We demonstrate that subnetworks pruned through OES surpass other leading pruning methods, such as the Lottery Ticket Hypothesis, which is known to be suboptimal for image recovery tasks (Wu et al., 2023). Our extensive experiments demonstrate the transferability of OES-masks and the characteristics of sparse-subnetworks for image generation. Code is available at https://github.com/Avra98/Optimal-Eye-Surgeon.git.

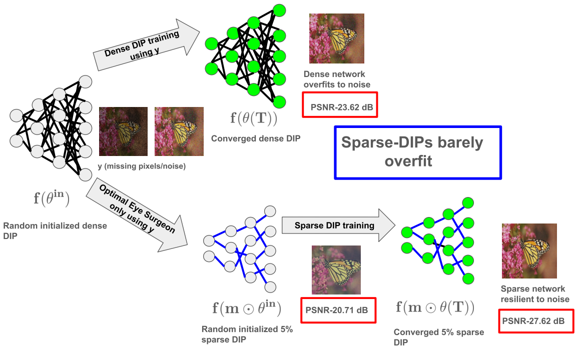

Dense DIP training using y
Dense network overfits to noise
PSNR-23.62 dB
$f(\theta(T))$
Converged dense DIP
Sparse-DIPs barely overfit
y (missing pixels/noise)
$f(\theta^{in})$
Random initialized dense DIP
Optimal Eye Surgeon only using y
Sparse DIP training
$f(m \odot \theta^{in})$
Random initialized 5% sparse DIP
PSNR-20.71 dB
$f(m \odot \theta(T))$
Converged 5% sparse DIP
Sparse network resilient to noise
PSNR-27.62 dB

Figure 1. Sparse-DIP lessens overfitting

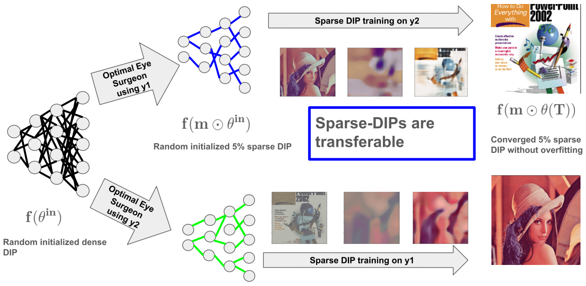

Sparse DIP training on y2
Optimal Eye Surgeon using y1
$f(m \odot \theta^{in})$
Random initialized 5% sparse DIP
Sparse-DIPs are transferable
$f(m \odot \theta(T))$
Converged 5% sparse DIP without overfitting
$f(\theta^{in})$
Random initialized dense DIP
Optimal Eye Surgeon using y2
Sparse DIP training on y1

Figure 2. Sparse-DIPs are transferrable

1. Introduction

Overparameterization has been central to the success of deep learning especially in image classification tasks. Empirically it is observed that bigger models (at scale) generalize better. Eventually, it was found that sufficiently sparse subnetworks can be found within these deep dense networks that can reach as high test accuracy as their dense counterparts. These sparse networks are called matching subnetworks. This led researchers to further study neural network pruning. However, the impact of overparameterization in deep convolutional neural networks (CNNs) hasn't been thoroughly studied for image reconstruction and inversion tasks although overparameterization is important in many image recovery tasks. Jin et al. (2017) empirically showed that trained deep CNNs are better substitutes to regularized iterative algorithms and direct inversion (Katsaggelos, 1989). The initial works further led to deploying deep convolutional networks inside the typical iterative image reconstruction framework, where it is fused with the physics or the forward model of the image generation problem (Venkatakrishnan

[1]Department of Computational Mathematics, Science and Engineering (CMSE), Michigan State University, MI, USA. [2]Department of Electrical Engineering and Computer Science (EECS), University of Michigan – Ann Arbor, MI, USA. [3]Department of Biomedical Engineering, Michigan State University, MI, USA. Correspondence to: Avrajit Ghosh <ghoshavr@msu.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

et al., 2013).

Going one step ahead, Ulyanov et al. (2018) showed that *untrained* deep convolutional networks can recover images directly from corrupted measurements. Hourglass architectures like Unet/Skipnet having downsampling, upsampling and convolutional operations are natural image priors, as they bias the output towards the prior distribution of natural images. These networks are known as Deep Image Prior (DIP). When trained to reconstruct a corrupted image, DIPs first learn the natural image component of the corrupted image and then overfit to the noise as they are highly overparameterized. This phenomenon is known as spectral bias (Chakrabarty & Maji, 2019). Hence, some early stopping time criteria need to be adopted before these models overfit to the noise or artifacts in the image. Finding an estimate of early stopping time typically requires knowledge of the clean image and noise-corruption level, which are usually unknown, making this an active area of research (Wang et al., 2021).

Underparameterized models[1] emerged as a good substitute to deep Unets as means to prevent overfitting. Heckel & Hand (2018) proposed deep decoder which consists of only upsampling layers and convolutional layers with kernel size 1×1. Deep decoders prevent overfitting to a large extent but as they are sufficiently underparameterized, they are not rich image priors. They fail to capture detailed image information (Wu et al., 2023).

In this work, we bridge this gap between overparameterized models like deep image prior and underparameterized models like deep decoder. We aim to find a sparse sub-network within a dense DIP network that can act as an image prior and doesn't overfit to noise because of underparameterization. Our main contributions are as follows:

1. We propose a principled approach of pruning a deep image prior network at *random initialization* with only the *corrupted measurement for a single image* and train the pruned network till convergence (Figure-1).

2. We show that the masked subnetwork output gives a low frequency approximation of the clean image by just masking at random network initialization. Further training these subnetworks to convergence alleviates overfitting.

3. We show that these sparse networks are transferable, i.e., masks learned on one image are transferable for recovering a different image (Figure-2).

2. Image reconstruction with DIP

The general framework for image reconstruction involves corrupted measurements $\mathbf{y}$ produced from a clean image $\mathbf{x}$ undergoing a corruption process $\mathbf{y} = \mathbf{A}(\mathbf{x}) + \epsilon$, where $\mathbf{A}(.)$ represents the corruption operation and ϵ is a noise vector drawn from any standard normalized distribution (e.g., Gaussian). The objective is to determine $\mathbf{x}$ given $\mathbf{y}$. Image reconstruction entails finding the MAP (Maximum A Posteriori) solution, which maximizes the posterior distribution $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$. Assuming Gaussian noise, the likelihood term $p(\mathbf{y}|\mathbf{x})$ focuses on minimizing $||\mathbf{y} - \mathbf{A}(\mathbf{x})||_2^2$ to identify the optimal fit. However, since the forward operator $\mathbf{A}(.)$ typically has a large null space, making the inverse problem ill-posed, additional insight into the prior distribution $p(\mathbf{x})$ is required.

Deep image prior proposed by Ulyanov et al. (2018) showed that by reparameterizing the reconstruction variable $\mathbf{x}$ as the output of an untrained deep Unet $\mathbf{x} = G(\boldsymbol{\theta}, \mathbf{z})$, we can regularize the solution-space of the output to look like natural images. For example, $G(.)$ denotes the hourglass convolutional architecture, $\boldsymbol{\theta}$ are the model parameters and $\mathbf{z}$ is a random input to the network. Here, the image prior is implicit, as the output space of $G(\boldsymbol{\theta}, \mathbf{z})$ inherently encapsulates the unique characteristics of a natural image. For image denoising, we minimize the loss $||\mathbf{y} - G(\boldsymbol{\theta}, \mathbf{z})||_2^2$ w.r.t network parameters $\boldsymbol{\theta}$, the target of the network being the corrupted image $\mathbf{y}$. Early in the training, the deep Unet architecture regularizes solutions towards natural images, giving an estimate of the clean image. However, as the model is highly overparameterized, the training loss $||\mathbf{y} - G(\boldsymbol{\theta}, \mathbf{z})||_2^2$ will be driven to 0, essentially ensuring $G(\boldsymbol{\theta}, \mathbf{z})$ fits the noisy image $\mathbf{y}$. Hence, some early-stopping strategy is required to obtain the clean image, which is difficult without the knowledge of the ground-truth clean image $\mathbf{x}$.

Several works in recent years have approached the challenge of finding the early-stopping time or preventing overfitting to noise, which broadly falls into two classes as discussed next.

2.1. Through regularization

Cheng et al. (2019) considers a Bayesian approach to inference, by conducting posterior inference using stochastic gradient Langevin dynamics which delays overfitting. Jo et al. (2021); Shi et al. (2022); Metzler et al. (2018) control the deep network capacity by regularizing the layer-weights or the Jacobian of the network. These methods incur an additional computational and backpropagation cost. Liu et al. (2019); Mataev et al. (2019); Sun et al. (2020); Cascarano et al. (2021); Bell et al. (2023) use additional regularizers on the deep, dense models such as the total-variation norm or trained denoiser or external guidance. These methods require the right regularization level which depends on the

[1] In image restoration, underparameterized models are defined as networks with fewer parameters than the number of image pixels. So, the image fit loss $\|G(\boldsymbol{\theta}, \mathbf{z}) - \mathbf{y}\|_2^2$ may not be zero at convergence.

noise-type, level, and image class to avoid overfitting. You et al. (2020) model sparse additive noise as an explicit term in the optimization. Ding et al. (2021) explore subgradient methods with diminishing step size schedules for impulse noise with ℓ_1 loss. These methods are limited to the types and the levels of noise they target. Finally, Wang et al. (2021) develop a general-purpose early-stopping detection criterion for all of the above methods. Their approach to detecting the transition from clean to noisy reconstruction is by estimating the running variance of the reconstructed image over the iteration window. However, in certain cases, their detection peak is sometimes off by certain iterations, as acknowledged by the authors. All of these works attempt to avoid overfitting while optimizing overparameterized dense models which incurs additional cost on storage and computational time.

2.2. Through underparameterization

On the contrary, the performance of under-parameterized networks for image recovery is significantly less approached. Heckel & Hand (2018) first proposed Deep-decoder, an underparameterized network consisting only of the decoder part of the Unet architecture. Underparameterization provides a barrier to overfitting, allowing the deep decoder to denoise without much overfitting. However, due to the same reason, deep decoders slightly underperform when the underlying ground-truth image has fine-grained texture details. Hence, for images with rich detail information, deep decoders underperform. However, their ability to prevent overfitting for most denoising problems makes them attractive for image restoration problems compared to typical DIP networks and their variants. The recent success of underparameterized networks like deep decoder motivates the question:

Q1 : *Can under-parameterization prevent overfitting and at the same time recover high-quality images?* If the answer to question Q1 is yes, then the next question is how to build these underparameterized networks. As a first step to this question, we start with an overparameterized Unet and attempt to study a principled pruning strategy to obtain an underparameterized network. Thus, we study the second and more interesting question:

Q2 : *Can we design a principled pruning method with only the corrupted measurements* $\mathbf{y}$ *to obtain an underparameterized network that satisfies Q1?* Our answers to both questions are positive and our findings reveal some interesting phenomena on overparameterization, initialization and their relation to capturing image priors.

3. Optimal Eye Surgeon: Pruning image generators at initialization

Neural network weight pruning dates back to as early as the early 90's (LeCun et al., 1989; Hassibi et al., 1993). Pruning can be broadly classified into three classes based on when networks are pruned: 1) *Pruning at Initialization (PAI)* methods prune deep networks at random initialization. The resultant sparse sub-network at initialization is then trained to convergence at inference time. Pruning at Initialization (PAI) techniques, like SNIP (Lee et al., 2018), GraSP (Wang et al., 2020), and SynFlow (Tanaka et al., 2020), focus on effective weight pruning in neural networks at random initialization. SNIP removes weights minimally impacting loss, GRASP preserves information flow, and SynFlow, a data-free method, maintains total synaptic flow under sparsity constraints. Our proposed method falls under this category. 2) *Pruning while Training (PWT)* takes a randomly initialized dense network and jointly trains and prunes a neural network by updating weights and masking the weights during training. Different strategies can be adopted for determining importance scores like random dropout, magnitude, or back-and-forth pruning (Evci et al., 2020; Zhao et al., 2019; He et al., 2018). The benefits of pruning early in training were also shown in You et al. (2019). 3) *Pruning After Training (PAT)* involves a Pretrain-Prune-Retrain cycle and is essential for obtaining matching subnetworks at non-trivial sparsity levels. The Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2018) advocates for Iterative Magnitude Pruning (IMP), which removes a percentage of weights based on the magnitude from a pretrained network, then retrains the remaining weights from their original initialization. For complex networks and large datasets, weight rewinding to an early-epoch state (Frankle et al., 2019) and learning-rate rewinding (Renda et al., 2020) were deemed essential to obtain matching subnetworks. The weight magnitudes at the end of training are crucial, as highlighted in ongoing research (Paul et al., 2022).

Overparameterization seems to be a crucial factor for finding sparse matching subnetworks. Zhou et al. (2019); Ramanujan et al. (2020) showed that when a network is sufficiently large, even learning a mask at random initialization (termed as *supermasks*) can show competitive performance like training a network. This phenomenon is termed as strong lottery ticket hypothesis, and was recently proved by Malach et al. (2020); da Cunha et al. (2021) under certain network assumptions. Supermasks were also used to generate different subnetworks for various tasks from the same dense network (Wortsman et al., 2020; Mallya et al., 2018), with applications also in graph networks (Huang et al., 2022). Our work is the first to show the existence and effectiveness of supermasks for image reconstruction. We further highlight the notable diffrences of pruning for image classification and image reconstruction in Table-10.

3.1. Suboptimality of LTH for DIP

LTH-based methods are very reliable to obtain matching sparse subnetworks at non-trivial sparsities for various ML

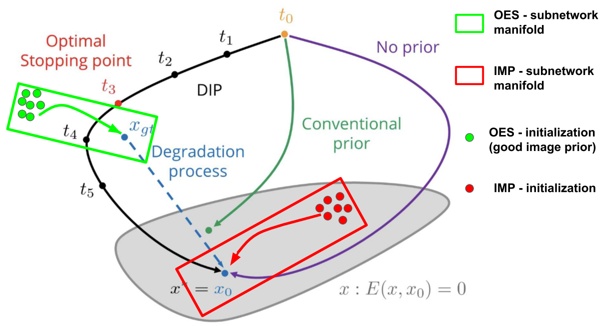


Figure 3. Subnetworks learned by OES are image generators with good image priors. On the contrary, the range of subnetworks learned by LTH is close to the overfitted noisy image, which is far from being an image prior. Image adapted from (Ulyanov et al., 2018)

tasks, a feat unachieved by other pruning methods. Given the success of LTH on a variety of machine learning tasks, at first-sight, it might be tempting to apply LTH on image reconstruction based tasks involving deep image prior. However, for unsupervised learning schemes like *DIP which overfit to noise at convergence, using the magnitudes at convergence, to determine which weights to prune, is in fact detrimental (Figure-3).*

> For image reconstruction, network output overfits to noise at convergence. Subnetworks obtained by LTH at convergence (without early stopping time) perform poorly on denoising tasks.

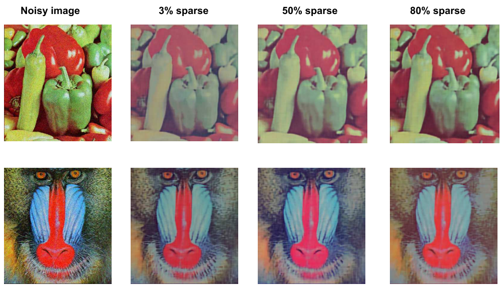


Figure 4. Masking network at initialization induces image prior. Figures show the images after masking image generator parameters at random-initialization $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$. The mask $\mathbf{m}^*$ was learned using the OES algorithm. Images corresponding to several sparsity levels are shown. We show that Strong Lottery Ticket Hypothesis also holds for image reconstruction.

Two possible ways to apply LTH to DIP for image reconstruction tasks are: a) using the clean image $\mathbf{x}$ to train the DIP model which will not require any early-stopping and b) the early stopping (ES) time can be obtained from the knowledge of $\mathbf{x}$ and the weight magnitude at ES can be used to obtain the mask. Wu et al. (2023), adopted method a) to obtain the mask, which might not be practical for most image reconstruction problems (see Section D.3 for detailed comparison) as we do not have knowledge of the clean image $\mathbf{x}$ nor an assumption of an early-stopping time (Figure-21). We show the effect of using LTH-based methods (with loss involving $\mathbf{y}$) as the mask in Figure-7 and in Section E (Appendix). Further, we study in detail, the architecture of the pruned network derived from LTH pruning in Figure-10a which sheds light on why IMP-masks may underperform in image reconstruction tasks. In our work, we propose Optimal Eye Surgeon (OES), a framework to prune image generators at random initialization which is optimal for pruning image generator networks.

3.2. Masking at initialization

Let $G(\boldsymbol{\theta}, \mathbf{z})$ be a dense and deep image generator network with random input $\mathbf{z}$. Let the random input $\mathbf{z} \in \mathbb{R}^q$, and let $\boldsymbol{\theta}$ be vectorized parameters of a dense Unet, $\boldsymbol{\theta} \in \mathbb{R}^d$. Let $\mathbf{x}, \mathbf{y}$ be the clean and noisy/corrupted RGB image such that $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{3\times H\times W}$, where H and W are respectively the height and width of the image. As DIPs are sufficiently overparameterized, i.e., the number of parameters is much more than the number of image pixels, it is usually the case that $d \gg 3HW$. Let $\boldsymbol{\theta}_{in}$ be the random initialized neural network, where the uniform Kaiming initialization is used. Let $\mathbf{m} \in \{0,1\}^d$ be the binary mask that we aim to learn at initialization. To learn an s-sparse mask, i.e., with only s non-zero parameters out of d, we would have to solve an integer problem:

$$\mathbf{m}^*(\mathbf{y}) = \arg\min_{\mathbf{m}\in\{0,1\}^d} ||G(\boldsymbol{\theta}_{in} \circ \mathbf{m}, \mathbf{z}) - \mathbf{y}||_2^2$$
$$\text{such that} \quad ||\mathbf{m}||_0 \leq s. \tag{1}$$

Equation (1) involves discrete optimization for deep networks, where d is very large (in millions). To get around this difficulty, we propose a Bayesian relaxation of (1) that is differentiable and unconstrained and can be solved by a local iterative algorithm such as gradient descent. We attempt this by reformulating (1) as learning Bernoulli dropout probability parameters $\mathbf{p}$ with the mask $\mathbf{m}$ being sampled from the Bernoulli distribution with mean $\mathbf{p} \in \mathbb{R}^d$.

$$\mathbf{m}^*(\mathbf{y}) = C(\mathbf{p}^*) \quad \text{such that}$$
$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \underbrace{\mathbb{E}_{\mathbf{m}\sim Ber(\mathbf{p})}\left[||G(\boldsymbol{\theta}_{in} \circ \mathbf{m}, \mathbf{z}) - \mathbf{y}||_2^2\right]}_{R(\mathbf{p})}$$
$$+ \lambda KL(Ber(\mathbf{p})||Ber(\mathbf{p}_0)). \tag{2}$$

The deterministic inequality constraint $||\mathbf{m}||_0 \leq s$ is changed into an unconstrained penalty which ensures that the learned Bernoulli distribution $Ber(\mathbf{p})$ is close to a prior Bernoulli distribution $Ber(\mathbf{p}_0)$, the known prior distribution depends on the desired sparsity level s. We fix $p_0 = \frac{s}{d}$.

For Bernoulli distributions, the distance measure between two distributions as given by Kulbick-Luiber divergence has a closed form and is given by $KL(Ber(\mathbf{p})||Ber(\mathbf{p}_0(s))) = \sum_i \left(p_i \log \frac{p_i}{p_{0i}} + (1-p_i)\log\frac{1-p_i}{1-p_{0i}}\right)$, where p_i and p_{0i} denotes the Bernoulli mean probability corresponding to the i^{th} weight parameter of $\mathbf{p}$ and $\mathbf{p}_0$. We solve this optimization problem by learning $\mathbf{p}$ via the Gumbel-softmax trick. We delay the details of the algorithm to the Appendix section C. After obtaining the converged $\mathbf{p}$, we prune the weights based on the ranking/ordering of $\mathbf{p}$ to obtain the desired sparsity level, which is denoted by the $C(.)$ function. We discuss the importance of KL regularization compared to L_1 regularization (Sreenivasan et al., 2022) or no regularization (Zhou et al., 2019) in Section I of Appendix. Previous work on Bernoulli mask learning and pruning on network initialization only focused on image-classification tasks, whereas our work applies it to image-reconstruction tasks and develops many new findings that might provide important insight into new structure design for DIP.

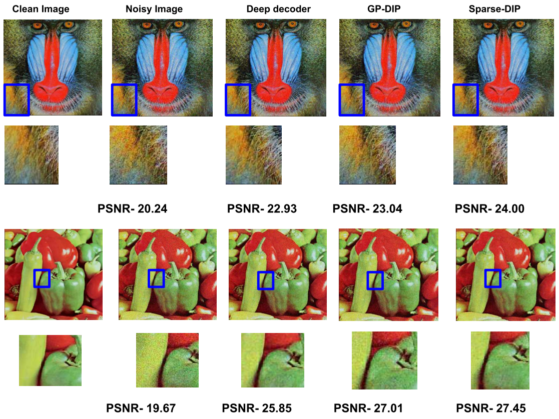


Figure 5. Comparative Analysis of Denoising Performance on 'Baboon' and 'Pepper' Images at $\sigma = 25$ dB.

Our proposed OES (Optimal Eye Surgeon) algorithm consists of two steps:

- Solve the optimization problem in (4) to learn the mask $\mathbf{m}^*(\mathbf{y})$ using OES.
- Train the sparse subnetwork $G(\boldsymbol{\theta} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$ to convergence to fit the corrupted image $\mathbf{y}$.

We summarize the important observations from applying our algorithm to DIP for image reconstruction. These findings will be supported by extensive numerical experiments in the next section and appendix.

- **Finding-1**: Masks learned by Step 1 of OES when applied at initialization induce a relatively good image prior (Figure-16). We term the sparse subnetwork $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$ at initialization as *Sparse-DIP*. It gives a good low-frequency approximation of the clean image by just the masking network.
- **Finding-2**: OES effectively recovers the clean image and exhibits minimal or no overfitting for denoising problems (Figure-6).
- **Finding-3**: On image recovery tasks, the training of subnetworks identified by OES is much more effective than those discovered by the current best Pruning At Initialization (PAI) methods. Furthermore, masks created by methods based on the Lottery Ticket Hypothesis (LTH) are not ideal for reconstructing images, a point we explore in detail in Section 4.3.
- **Finding-4**: Sparse-DIPs are transferable across images, datasets and corruption processes. More specifically, a mask learned by OES from one image can be used to successfully reconstruct other images, from completely different datasets.
- **Finding-5**: The encoder part of DIP is more compressible (prunable) than the decoder part. (Section 6)
- **Finding-6** (Appendix): The irregularly pruned sparse-DIP is better than the regular deep decoder of a similar size. (Figure-11).
- **Finding-7** (Appendix): Mask trained based on the initial weights is more transferrable than that based on the magnitude of the final trained weights like LTH. (Section-E in Appendix).

4. Experimental support of the findings

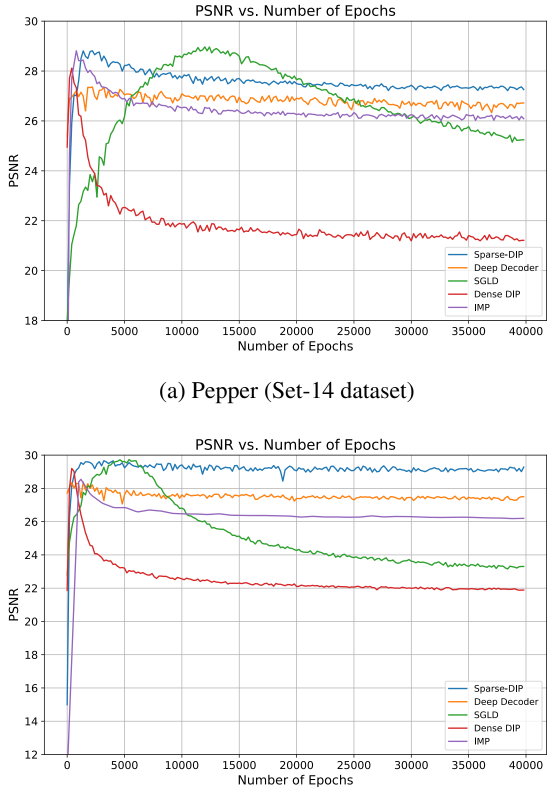
Through extensive experiments, we confirm our findings. We use images from three popular datasets: the Set-14 dataset (Zeyde et al., 2012), the standard image dataset (Ulyanov et al., 2018) and the Face dataset (Bevilacqua et al., 2012). In Finding-1 (4.1), and Figure-16, we study the quality of images that are produced by just masking. In Finding-2 (4.2), we compare the denoising performance of OES with overparameterized DIP, Gaussian process DIP (Cheng et al., 2019) and underparameterized deep decoder (Heckel & Hand, 2018). In Finding-3 (4.3), we show results of OES against state-of-the-art pruning methods. Finally, we compare the transferability of OES and IMP across various combinations of images and datasets in Finding-4 (Section 4.4).

4.1. Finding-1: Masking at initialization induces image prior

Masking at initialization with masks learned by OES inherently captures low frequency components of the image. In Figure-16, we display the results of $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$ alongside the original corrupted image $\mathbf{y}$ for images in the Set-14 dataset. Images across three different levels of sparsities 3%, 50%, and 80% are shown. OES-masked images for other datasets are shown in Figure-17 in the appendix. We observe that OES can effectively reconstruct the simpler, low-frequency parts of an image from the corrupted version $\mathbf{y}$, but it struggles with the more intricate details. This means that while OES can denoise an image, some information is lost in the process. Masking has its limitations compared to regular training. It can only represent a limited number of functions, up to 2^d, where d is the number of parameters in the network. Consequently, due to this limitation in function representation, the training loss with masking cannot reach zero. In our study, we found that OES primarily reconstructs the simpler, low-frequency parts of images. Since natural images usually contain more low-frequency elements, focusing on these parts allows for the greatest reduction in loss. Additionally, because the model described in (1) lacks sufficient function representation capability, it never achieves a training loss of zero. However, even in this limited setting, OES is effective in finding a mask that represents the image $\mathbf{y}$ as closely as possible. For experiments in the manuscript, $\mathbf{y}$ is the Gaussian noise corrupted image.

4.2. Finding-2: Sparse DIPs prevent overfitting

In OES, we further train the remaining subnetwork within the obtained mask ($G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$) till convergence to perform the image reconstruction task, image denoising. We conduct experiments on the denoising capabilities of these subnetworks over several noise levels and various images across 3 popularly used datasets, which we report in Figure-6. We perform a comparison with the following enumerated network based denoising methods. 1) Dense DIP which is the overparameterized network originally proposed by Ulyanov et al. (2018). The encoder part has 6 layers (*Conv* → *ReLU* → *Batchnorm* → *Downsample*) followed by 6 layers of decoder (*Upsample* → *ReLU* → *Batchnorm*). The convolution patch size in both the encoder and decoder parts is 3×3. The input $\mathbf{z}$ is fixed to be a random tensor drawn from the Gaussian distribution of dimension $H \times W \times 32 \times 3$. The total number of parameters in Dense-DIP is 3008867 (3 million) and the image dimension ($\mathbf{y}$) is $3 * 512 * 512 = 786432$ (0.7 million). The network is overparameterized. 2) Gaussian Process-DIP (GP-DIP) is the network trained by SGLD and proposed by Cheng et al. (2019) to alleviate overfitting to an extent, and 3) Deep Decoder, proposed by Heckel & Hand (2018) is an underparameterized network that prevents overfitting. Deep decoder contains only the decoder part of Unet. It has 1×1 convolution layer and upsampling layers (1×1 *Conv* → *Up sample* → *ReLU* → *channelnorm*). Standard decoder architecture proposed by Heckel & Hand (2018) uses channel dimension of 128 with 6 layers as optimal denoising architecture. For this architecture, deep decoder has 100224 (0.1 million) parameters [2]. *Sparse-DIP* is the pruned architecture obtained at initialization by our OES method. We perform denoising with a 3% sparse subnetwork found by OES which has approximately 90217 parameters (0.09 million), slightly less than the number of parameters in deep decoder. We use the ADAM optimizer with learning rate 10^{-2} (as reported in Ulyanov et al. (2018)) in all our experiments for training both the dense and sparse networks. In Figure-6 and Table-2, we report the results without applying early stopping and running the optimization procedure for a large number of iterations (40k). Sparse-DIP outperforms deep decoders and the overparameterized models (with regularization) for majority of the images. In Figure-5, we plot the denoising results on Baboon and Pepper images. When closely zoomed in the area of focus, we observe that the deep decoder suffers from oversmoothing the edges, while GP-DIP overfits to noise due to overparameterization. We study this phenomenon in detail in Section 5. The OES framework can also be extended to general noisy inverse problem settings involving a forward operator (with a non-trivial nullspace). We extend our framework to MRI reconstruction from undersampled k-space measurements in Appendix H.

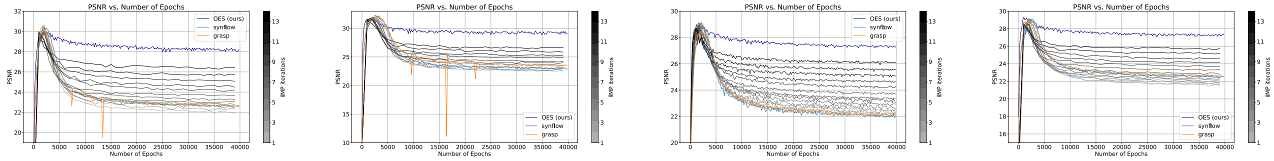


(a) Pepper (Set-14 dataset)

(b) Door (Standard Dataset)

Figure 6. Denoising results of various methods on noisy images ($\sigma = 25$ dB) across 3 popularly used datasets.

[2] Further reduction of number of layers to 5 makes the denoising performance poor as mentioned by the authors and also confirmed by our experiments. We use the 6-layer deep decoder as the standard for our experiments in the paper, unless specified otherwise



(a) Dataset-3 (Standard dataset) (b) Face-1 (Face dataset) (c) Pepper (Set-14 dataset) (d) Lena (Set-14 dataset)

Figure 7. Comparison of denoising performances for subnetworks found by various pruning methods (GRASP, Synflow, IMP, and OES). IMP utilizes 14 pruning iterations with 20% weight reduction at iterations. All the masks are 5% sparse. IMP undergoes an additional 14 steps of training and pruning before obtaining the final mask. The gray curves indicate the progression of IMP iterations, with darker shades representing higher iteration counts. Each IMP iteration is shown. The detailed result for all images in 3 datasets can be found in Table-4 in the Appendix.

4.3. Finding-3: OES is superior to other pruning methods

We compare OES with the state-of-the-art pruning methods. For pruning at initialization, we compare with GraSP, SynFlow[3], magnitude and random scores in Table-4. For LTH based pruning, we report for two pruning schedules and observe that gradual pruning for larger pruning iterations yields better performance. We evaluate the denoising performance of these subnetworks at 5% sparsity level. In Figure-7, we observe that at 5% sparsity level, OES masks shows minimal to no overfitting. LTH masks are obtained based on ranking the magnitude of the weights at convergence (at 40k epochs) and subnetworks obtained by LTH show overfitting, when masks are at the same level of sparsity. We demonstrate the adverse effect of LTH on DIP in Figure-7, which we orignally motivated in Section 3.1. However, when PSNR curves with LTH masks are plotted at every pruning iteration in Figure-7, we observe that the effect of overfitting becomes less severe when networks become more sparse. Both Synflow and Grasp show signs of overfitting for image denoising. In magnitude and random pruning methods applied *at initialization*, it is often observed that layers with a large number of parameters (large width) and those with fewer parameters (small width) are respectively at a higher risk of being entirely pruned. We consistently observe that with magnitude and random pruning at initialization, at 5% sparsity level, there is *layer-collapse*. This phenomenon occurs when an entire layer gets pruned and the output is a constant image.

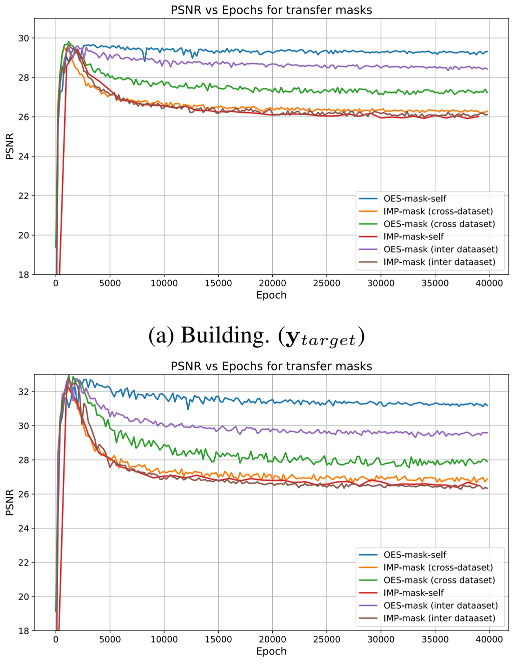
4.4. Finding-4: OES masks are transferable

We perform experiments on transferring the masks obtained by Step 1 of OES on one image and show the masked subnetwork can be used for denoising a different image. We compare the transferability of the OES masks with IMP masks at the same level of sparsity (5%). We also show that OES masks can be transferred not only to images within the same dataset, but also to those from a different dataset. In Figure-8, we compare the denoising performance for different sets of learned masks for both IMP and OES. Say there are two image datasets: Dataset-A (face) and Dataset-B (standard dataset), each of which contains noisy images. Also, let us term the image that is used to learn the mask as $\mathbf{y}_{source}$ and the image on which denoising is performed as $\mathbf{y}_{target}$. Then we explore three possibilities: 1) **self-masking**: $\mathbf{y}_{source} = \mathbf{y}_{target}$, the same corrupted image is used to learn the mask, and the mask is used for denoising; 2) **inter-dataset masking**: $\mathbf{y}_{source} \neq \mathbf{y}_{target}$, but both $\mathbf{y}_{source}$ and $\mathbf{y}_{target}$ belong to the same dataset; and 3) **cross-dataset masking**: $\mathbf{y}_{source} \neq \mathbf{y}_{target}$ and both of them belong to different datasets (say $\mathbf{y}_{source} \in$ Dataset-A and $\mathbf{y}_{target} \in$ Dataset-B or vice-versa). In the experiments, we use images from a standard image dataset (Ulyanov et al., 2018) and the face-dataset (Bevilacqua et al., 2012) to show the extent of transferability between inter and cross datasets. We note that the images in this dataset are visually diverse as face images have different characteristics than those in the standard image dataset. We observe in most cases, self-masking by OES provides the best performance. IMP masks provide the worst performance irrespective of the source and target image. Inter-dataset masking and cross-dataset masking by OES also gives good PSNR at convergence but the performance slightly degrades when compared to self-masking. More experiments comparing IMP based masking with OES are provided in Section-13 in the Appendix.

5. Noise impedance of sparse-DIP

Sparse-DIPs often outperform deep decoder even with lower levels of parameter count. Based on our experiments, we observe that with images having edges, this difference becomes prominent. To further investigate, we study the noise impedance of the network (denoted as $f(\mathbf{y})$) when trained to fit random Gaussian noise by minimizing the loss $\|G(\boldsymbol{\theta}, \mathbf{z}) - \mathbf{y}\|_2^2$ w.r.t. the parameters of the network (dense or sparse), where $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$. This is to see how each network has the capacity to fit white Gaussian noise. *Dense DIP* fits the noise in the image perfectly with

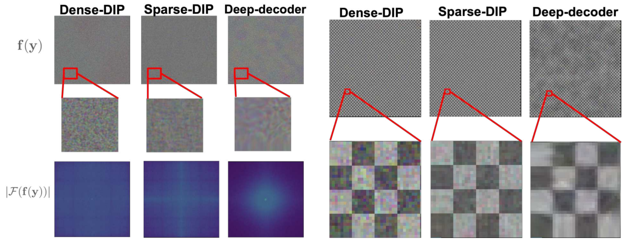
[3] Performance of SNIP is not upto par with GraSP and SynFlow and hence we don't report it.



(a) Building. ($\mathbf{y}_{target}$)

(b) Face-3. ($\mathbf{y}_{target}$)

Figure 8. Performance of masks trained from several images for denoising with noise level ($\sigma = 25$ dB). Self denotes a mask learned from the same image. Inter-dataset denotes mask learned from images from the same dataset. Cross-dataset denotes mask learned from images of different dataset. The standard dataset and the face dataset were used in this experiment. For Figure-a) Inter-datset mask ($\mathbf{y}_{source}$) is House, Cross-dataset mask ($\mathbf{y}_{source}$) is Face-0. For Figure-b) Inter-datset mask ($\mathbf{y}_{source}$) is Face-0, Cross-dataset mask ($\mathbf{y}_{source}$) is House. All the masks used in this figure are 5% sparse.



(a) Noise impedance

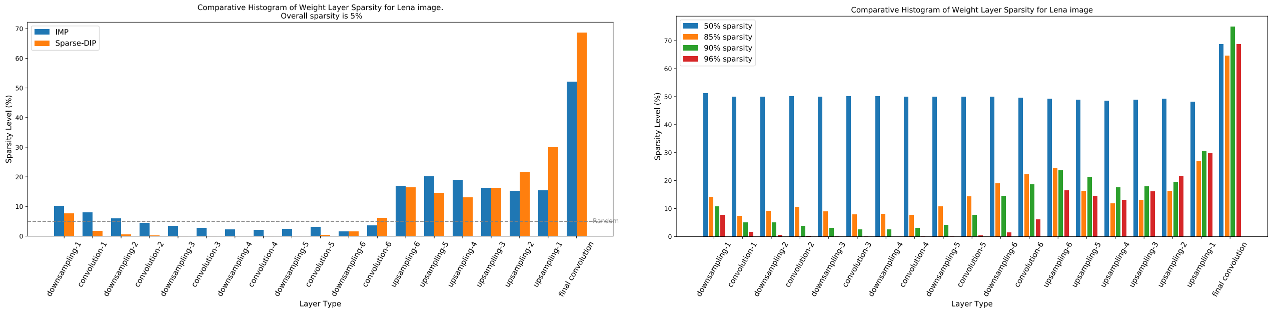
(b) Recovery of edges

Figure 9. Figure-a) shows the ability of networks to fit noise. $f(\mathbf{y})$ is the network output and $|\mathcal{F}(\mathbf{y})|$ is the magnitude of Fourier coefficients. Figure-b) shows quality of recovering edges.

zero training loss and in Fourier domain $\mathcal{F}[f(\mathbf{y})]$ shows a constant wide-band spectrum which is quite typical for Gaussian white noise. *Deep Decoder* is underparameterized and the training loss does not go to 0 indicating that noise $\mathbf{y}$ lies outside of the output range space of the network. Deep decoder smoothens out the noise to a large extent. The magnitude of the Fourier Transform of the output shows that the cut-off frequency is small essentially making it act like a low-pass filter with small bandwidth. *Sparse-DIP* is also underparameterized and obtained by masking 97% weights of a dense DIP. The magnitude spectrum of $\mathcal{F}[f(\mathbf{y})]$ shows that x-axis and y-axis of the spectra have much higher magnitude than that of deep decoder, hence it can reconstruct directional edges better than deep decoder. To further explore the image representation and noise impedance capacity, we fit these three networks to a noisy chessboard image, where the strip frequency is very high. We observe from Figure-9b that Dense DIPs recover the high-frequency edges but overfit to noise. Deep decoder has very low cut-off frequency (Figure-9a). It fails to recover the vertical and horizontal edges of the chessboard, although it does smoothen out the noise. Sparse-DIP recovers the edges and does not overfit to noise. The ability of Sparse-DIP to reconstruct high-frequency edges better than deep decoder (with similar number of parameters) explains why it showed superior denoising performance in Figure-6 and Table-2.

6. Pruned architecture study - Finding 5

Throughout all the experiments, we used Unet without skip connection as the Dense-DIP architecture. In Figure-10a, we show how the different layers of Unet are pruned with OES and IMP. These may shed light on the superior performance of OES when compared to IMP. In Figure-10b, we show the pruning pattern for OES masking for various levels of sparsity. We make the following observations: 1) *Importance of first and last layer*: The first layer of the encoder (*convolution+dowsampling*) layer and the last layer of decoder (*convolution* layer) have large number of remaining weights. The final reconstructed image is formed after convolution in the last layer, so it justifies the observation that the final convolution layer has the least amount of pruned weights. 2) *Towards the emergence of deep-decoder*: In Figure-10b, we observe that for various levels of sparsity, the decoder part of the architecture is pruned the least. This leads to the observation that for image generation, the upsampling layers play a crucial role, also observed in Liu et al. (2023). This further justifies the use of Deep decoder proposed by Heckel & Hand (2018), where the authors only use the decoder part of the Unet. 3) *Encoder layers play a role in overfitting*: When comparing the architecture of IMP-pruned vs OES-pruned networks, we observe that IMP prefers the layers in the encoder much more than OES. 4) *Importance of encoder-decoder junction*: The junction between the encoder-decoder is important as it has lot of non-zero parameters after pruning. This part is responsible for the generation of the low-frequency information of the image, which composes the majority of the information for natural images. This is because the spatial feature in this layer (because of simultaneous downsampling) is comparable to convolutional patch filter size, making its receptive



(a) OES vs IMP pruned network

(b) Layer-wise sparsity for pruned Unet for various sparsity levels using OES.

Figure 10. a) Layerwise sparsity in Unet architecture for pruning methods IMP and OES. b) Distribution of layerwise parameters for various sparsity levels using OES. The corrupted image $\mathbf{y}$ used was Lena. The overall sparsity in the architecture is 5%.

field larger. 5) *Pruning pattern for various sparsities*: We observe a similarity in the sparsity pattern across different layers in the shape of 'W'. For various pruning percentages $85\%, 90\%$ and 96%, we observe a similarity in the sparsity pattern across different layers. The three most important layers for the Unet seem to be the first layer (also the first layer of the encoder), the encoder-decoder junction and the last layer (final convolution).

7. Limitations

While our work presents a novel method to prevent overfitting, it is essential to acknowledge few limitations:

- Sparse networks tend to overfit slightly for transferring across different domains (Figure-8).
- Finding the mask adds computational overhead due to the Gumbel Softmax reparameterization. Since the masks are transferable, this overhead is not significant.
- Specialized tasks, such as MRI image processing, require unique architectures (e.g., two-channel Unet), limiting the transferability of OES subnetworks across different tasks with different architectures.

8. Conclusion

In this work, we demonstrate for the first time that in a dense deep image generator network, there exists a hidden subnetwork (sparse DIP) at initialization that shows potential of reconstructing low-frequency information of an image from only its noisy measurements. Sparse DIPs show significant potential for image reconstruction and transferability, surpassing traditional pruning methods. We believe that the connection between sparsity in the generator network and the low-dimensionality of the image output (situated in the manifold of images) prompts further theoretical investigation. We aim to further explore the role of these sparse networks within diffusion model-based generative frameworks, aiming to expedite the process and enhance the quality of generated images.

Acknowledgment

AG acknowledges Shijun Liang (PhD student, Michigan State University (MSU)) for his help with the MRI reconstruction experiments and Ismail Alkhouri and Evan Bell from MSU for helpful discussions. AG, SR, RR, and KS acknowledge support from NSF CCF-2212065. QQ acknowledges support from NSF CAREER CCF-2143904, NSF CCF-2212066, NSF CCF-2212326, ONR N00014-22-1-2529, and a gift grant from KLA. The authors also acknowledge the comments of the anonymous reviewers, which helped improve the manuscript.

Impact Statement

Our research aims to enhance image reconstruction efficiency using sparse generator networks. There are some potential societal impacts of this advancement, particularly in the domain of image generation and recovery; however, we do not feel that any specific impacts need to be highlighted here.

References

Arican, M. E., Kara, O., Bredell, G., and Konukoglu, E. Isnas-dip: Image-specific neural architecture search for deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1960–1968, 2022.

Bell, E., Liang, S., Qu, Q., and Ravishankar, S. Robust self-guided deep image prior. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096631.

Bevilacqua, M., Roumy, A., Guillemot, C., and Alberi-Morel, M. L. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012.

Cascarano, P., Sebastiani, A., Comes, M. C., Franchini, G., and Porta, F. Combining weighted total variation and deep image prior for natural and medical image restoration via admm. In *2021 21st International Conference on Computational Science and Its Applications (ICCSA)*, pp. 39–46. IEEE, 2021.

Chakrabarty, P. and Maji, S. The spectral bias of the deep image prior. *arXiv preprint arXiv:1912.08905*, 2019.

Chen, Y.-C., Gao, C., Robb, E., and Huang, J.-B. Nas-dip: Learning deep image prior with neural architecture search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pp. 442–459. Springer, 2020.

Cheng, Z., Gadelha, M., Maji, S., and Sheldon, D. A bayesian perspective on the deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5443–5451, 2019.

da Cunha, A., Natale, E., and Viennot, L. Proving the lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations*, 2021.

Ding, L., Jiang, L., Chen, Y., Qu, Q., and Zhu, Z. Rank over-specified robust matrix recovery: Subgradient method and exact recovery. *arXiv preprint arXiv:2109.11154*, 2021.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

Hassibi, B., Stork, D. G., and Wolff, G. J. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pp. 293–299. IEEE, 1993.

He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

Heckel, R. and Hand, P. Deep decoder: Concise image representations from untrained non-convolutional networks. *arXiv preprint arXiv:1810.03982*, 2018.

Huang, T., Chen, T., Fang, M., Menkovski, V., Zhao, J., Yin, L., Pei, Y., Mocanu, D. C., Wang, Z., Pechenizkiy, M., et al. You can have better graph neural networks by not training weights at all: Finding untrained gnns tickets. *arXiv preprint arXiv:2211.15335*, 2022.

Jin, K. H., McCann, M. T., Froustey, E., and Unser, M. Deep convolutional neural network for inverse problems in imaging. *IEEE transactions on image processing*, 26 (9):4509–4522, 2017.

Jin, T., Carbin, M., Roy, D., Frankle, J., and Dziugaite, G. K. Pruning's effect on generalization through the lens of training and regularization. *Advances in Neural Information Processing Systems*, 35:37947–37961, 2022.

Jo, Y., Chun, S. Y., and Choi, J. Rethinking deep image prior for denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5087–5096, 2021.

Katsaggelos, A. K. Iterative image restoration algorithms. *Optical engineering*, 28(7):735–748, 1989.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Lee, N., Ajanthan, T., and Torr, P. H. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Liu, J., Sun, Y., Xu, X., and Kamilov, U. S. Image restoration using total variation regularized deep image prior. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7715–7719. Ieee, 2019.

Liu, Y., Li, J., Pang, Y., Nie, D., and Yap, P.-T. The devil is in the upsampling: Architectural decisions made simpler for denoising with deep image prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12408–12417, 2023.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pp. 6682–6691. PMLR, 2020.

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 67–82, 2018.

Mataev, G., Milanfar, P., and Elad, M. Deepred: Deep image prior powered by red. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Mehta, R. Sparse transfer learning via winning lottery tickets. *arXiv preprint arXiv:1905.07785*, 2019.

Metzler, C. A., Mousavi, A., Heckel, R., and Baraniuk, R. G. Unsupervised learning with stein's unbiased risk estimator. *arXiv preprint arXiv:1805.10531*, 2018.

Paul, M., Chen, F., Larsen, B. W., Frankle, J., Ganguli, S., and Dziugaite, G. K. Unmasking the lottery ticket hypothesis: What's encoded in a winning ticket's mask? *arXiv preprint arXiv:2210.03044*, 2022.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11893–11902, 2020.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

Shi, Z., Mettes, P., Maji, S., and Snoek, C. G. On measuring and controlling the spectral bias of the deep image prior. *International Journal of Computer Vision*, 130(4):885–908, 2022.

Sreenivasan, K., Sohn, J.-y., Yang, L., Grinde, M., Nagle, A., Wang, H., Xing, E., Lee, K., and Papailiopoulos, D. Rare gems: Finding lottery tickets at initialization. *Advances in Neural Information Processing Systems*, 35: 14529–14540, 2022.

Sun, Z., Sanchez, T., Latorre, F., and Cevher, V. Solving inverse problems with hybrid deep image priors: the challenge of preventing overfitting. *arXiv preprint arXiv:2011.01748*, 2020.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9446–9454, 2018.

Venkatakrishnan, S. V., Bouman, C. A., and Wohlberg, B. Plug-and-play priors for model based reconstruction. In *2013 IEEE global conference on signal and information processing*, pp. 945–948. IEEE, 2013.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

Wang, H., Li, T., Zhuang, Z., Chen, T., Liang, H., and Sun, J. Early stopping for deep image prior. *arXiv preprint arXiv:2112.06074*, 2021.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

Wu, Q., Chen, X., Jiang, Y., and Wang, Z. Chasing better deep image priors between over-and under-parameterization. *Transactions on Machine Learning Research*, 2023.

You, C., Zhu, Z., Qu, Q., and Ma, Y. Robust recovery via implicit bias of discrepant learning rates for double over-parameterization. *Advances in Neural Information Processing Systems*, 33:17733–17744, 2020.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957*, 2019.

Zeyde, R., Elad, M., and Protter, M. On single image scale-up using sparse-representations. In *Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7*, pp. 711–730. Springer, 2012.

Zhao, C., Ni, B., Zhang, J., Zhao, Q., Zhang, W., and Tian, Q. Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2780–2789, 2019.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32, 2019.

Appendix

In the appendix section, we provide further extensive results to support our findings presented in the manuscript. We present the following sections sequentially:

1. Section-A presents denoising results which were reported in Finding-2 of the manuscript (4.2). Here, we present denoising performance for several noise levels.

2. Section-B contains the performance of OES method with standard pruning methods in literature over images in three different datasets.

3. Section-C summarizes the details of the Gumbel softmax reparameterization trick that was utitlized in learning the mask by OES.

4. Sectioin-D summarizes related works and confirms similar findings with related works. Here, we also highlight the difference of our work and show how OES is a more generalizable approach compared to the related works. We highlight that *no clean image is needed or no prior assumption on architecuture is required for finding a good subnetwork.*

5. Section-E highlights the difficulty in using IMP for pruning networks for image reconstruction tasks. We also consider the oracle case, where *clean image is used for IMP and we show that it has poor transferrability compared to OES.*

6. Section-F shows transfer to a different task (here inpainting). We test OES masks learned on inpainting and denoising tasks and compare them on the respective tasks.

7. Section-G shows the robustness of hyperparameter λ when KL regularization is used.

8. Section-H extends the OES framework for MRI reconstruction from undersampled k-space measurements.

9. Section-I shows the comparison and disadvantages of finding mask through L_1 regularization as done in Sreenivasan et al. (2022).

10. Section-J studies the sensitivity of masks obtained at different initialization distribution/initialization scale and when IMP masks are learned at early stop time.

11. Section-K shows the adverse effects of pruning an already underparameterized deep decoder.

12. Section-L highlights the difference in neural network pruning for image classification and image reconstruction. To the best of our knowledge, our work shows the phenomenon of Stong Lottery Ticket Hypothesis in image reconstruction for the first time.

A. Denoising Results

In this section, we report the denoising performance for all the images in the 3 datasets. In Table-1, we report the number of parameters used in each network. In Table-2, we report the PSNR at convergence for images across 3 datasets. We further plot the PSNR convergence curves of a subset of these images in Figure-11. In these figures, we want to emphasize that dense DIPs overfit to noise at convergence. With Sparse-DIP's obtained at OES, the overfitting is reduced by a large extent. We also observe that 80%-sparse DIP is more prune to overfitting than 3%-sparse DIP.

Table 1. Number of parameters count of sparse and dense networks. Number of pixels in image is $512 \times 512 \times 3 = 786432$ (0.7M)

Model	**Dense DIP**	**Dense Decoder**	**Sparse-decoder** (50%)	**Sparse-DIP**(3%)	**Sparse-DIP**(4%)
Number of parameters	3008867(3M)	100224(0.10M)	50112 (0.05M)	90217 (0.09M)	120354 (0.12M)

Table 2. Denoising capabilities comparison without early stopping on Set-14 dataset. The decoder has 100,224 parameters. Dense DIP has 3,008,867 parameters. Sparse networks are 3% sparse and have 90217 parameters. The PSNR values are noted at the end of convergence of training after 40000 epochs. The average of three runs using different random seeds are noted. For each implementation, a random $\mathbf{z}$ and network initialization is used for evaluation. For X_y, X denotes the average of three runs and y denotes the standard deviation.

	$\sigma = 25dB$				$\sigma = 12dB$				$\sigma = 17dB$			
Image	Dense DIP	GP DIP	Deep Decoder	Sparse-DIP	Dense DIP	GP DIP	Deep Decoder	Sparse-DIP	Dense DIP	GP DIP	Deep Decoder	Sparse-DIP
Pepper	$21.21_{0.22}$	$25.14_{0.29}$	$27.01_{0.25}$	$\mathbf{27.45}_{0.22}$	$27.34_{0.18}$	$29.17_{0.26}$	$28.81_{0.18}$	$\mathbf{30.41}_{0.36}$	$24.39_{0.39}$	$26.02_{0.16}$	$27.52_{0.32}$	$\mathbf{28.92}_{0.22}$
Foreman	$20.69_{0.17}$	$21.84_{0.29}$	$24.20_{0.24}$	$\mathbf{25.15}_{0.13}$	$26.59_{0.07}$	$28.33_{0.05}$	$29.81_{0.08}$	$\mathbf{30.55}_{0.14}$	$23.71_{0.22}$	$25.14_{0.17}$	$27.14_{0.28}$	$\mathbf{27.70}_{0.16}$
Flowers	$22.27_{0.18}$	$23.76_{0.35}$	$27.09_{0.22}$	$\mathbf{27.10}_{0.18}$	$28.55_{0.17}$	$30.78_{0.14}$	$30.33_{0.09}$	$\mathbf{31.07}_{0.25}$	$25.51_{0.13}$	$27.48_{0.11}$	$29.21_{0.25}$	$\mathbf{29.39}_{0.22}$
Comic	$20.63_{0.22}$	$21.56_{0.29}$	$23.22_{0.12}$	$\mathbf{24.03}_{0.21}$	$26.45_{0.36}$	$28.27_{0.36}$	$28.01_{0.06}$	$\mathbf{28.57}_{0.09}$	$23.68_{0.03}$	$24.90_{0.18}$	$25.82_{0.02}$	$\mathbf{26.53}_{0.19}$
Lena	$21.28_{0.39}$	$22.76_{0.22}$	$\mathbf{26.80}_{0.35}$	$26.40_{0.29}$	$27.50_{0.28}$	$29.48_{0.19}$	$\mathbf{30.96}_{0.24}$	$30.89_{0.43}$	$24.52_{0.11}$	$26.30_{0.18}$	$\mathbf{29.33}_{31}$	29.03_{13}
Barbara	$23.90_{0.10}$	$23.49_{0.35}$	$25.30_{0.09}$	$\mathbf{26.50}_{0.37}$	$28.27_{0.20}$	$\mathbf{30.81}_{0.25}$	$27.50_{0.31}$	$27.81_{0.21}$	$25.15_{0.13}$	$27.55_{0.08}$	$26.65_{0.04}$	$\mathbf{27.63}_{0.30}$
Monarch	$23.62_{0.36}$	$23.35_{0.14}$	$\mathbf{27.87}_{0.33}$	$27.62_{0.03}$	$28.25_{0.05}$	$31.17_{0.24}$	$32.00_{0.22}$	$\mathbf{32.12}_{0.17}$	$25.14_{0.14}$	$27.20_{0.25}$	$30.29_{0.18}$	$\mathbf{30.42}_{0.30}$
Baboon	$21.68_{0.14}$	$23.04_{0.31}$	$22.93_{0.18}$	$\mathbf{24.00}_{0.29}$	$27.27_{0.39}$	$27.03_{0.22}$	$24.12_{0.21}$	$\mathbf{25.91}_{0.04}$	$24.80_{0.11}$	$25.57_{0.20}$	$23.78_{0.27}$	$\mathbf{25.04}_{0.32}$
Ppt3	$24.07_{0.35}$	$24.57_{0.39}$	$26.81_{0.35}$	$\mathbf{26.90}_{0.22}$	$28.88_{0.10}$	$31.94_{0.13}$	$31.73_{0.18}$	$\mathbf{32.41}_{0.17}$	$25.85_{0.21}$	$28.78_{0.20}$	$29.49_{0.23}$	$\mathbf{29.96}_{0.22}$
Coastguard	$20.53_{0.01}$	$21.23_{0.21}$	$23.71_{0.20}$	$\mathbf{24.19}_{0.06}$	$26.50_{0.07}$	$28.13_{0.10}$	$29.43_{0.17}$	$\mathbf{30.60}_{0.11}$	$23.54_{0.04}$	$24.53_{0.14}$	$26.36_{0.08}$	$\mathbf{27.09}_{0.35}$
Bridge	$21.77_{0.31}$	$25.07_{0.02}$	$25.55_{0.26}$	$\mathbf{26.12}_{0.30}$	$28.58_{0.20}$	$30.47_{0.10}$	$28.10_{0.09}$	$\mathbf{29.23}_{0.31}$	$25.31_{0.08}$	$28.17_{0.42}$	$27.04_{0.28}$	$\mathbf{28.08}_{0.38}$
Zebra	$21.94_{0.08}$	$23.46_{0.02}$	$27.37_{0.19}$	$\mathbf{27.40}_{0.29}$	$28.45_{0.17}$	$30.93_{0.20}$	$30.81_{0.12}$	$\mathbf{31.54}_{0.20}$	$25.25_{0.38}$	$27.39_{0.34}$	$29.21_{0.29}$	$\mathbf{29.42}_{0.05}$
Face	$21.03_{0.07}$	$21.76_{0.30}$	$24.32_{0.11}$	$\mathbf{24.53}_{0.22}$	$26.90_{0.02}$	$27.81_{0.37}$	$29.93_{0.36}$	$\mathbf{29.93}_{0.02}$	$24.10_{0.06}$	$24.96_{0.12}$	$27.01_{0.25}$	$\mathbf{27.23}_{0.38}$
Man	$21.98_{0.31}$	$24.18_{0.10}$	$26.27_{0.33}$	$\mathbf{26.59}_{0.39}$	$28.45_{0.39}$	$31.22_{0.19}$	$29.84_{0.25}$	$\mathbf{30.94}_{0.31}$	$25.12_{0.26}$	$28.63_{0.29}$	$28.77_{0.20}$	$\mathbf{29.11}_{0.13}$

Table 3. Denoising capabilities comparison without early stopping for $\sigma = 25$dB on Face Dataset and Standard dataset.

(a) Standard Dataset

$\sigma = 25dB$				
Image	Dense DIP	GP DIP	Deep Decoder	Sparse DIP
Flight	20.49	22.02	23.99	**24.02**
House	21.88	23.30	28.35	**29.27**
Building	21.93	23.55	27.23	**27.23**
Door	21.85	23.31	27.02	**28.18**
Hats	21.76	24.12	24.86	**26.07**

(b) Face Dataset

$\sigma = 25dB$				
Image	Dense DIP	GP DIP	Deep Decoder	Sparse DIP
Face-1	22.40	26.72	28.90	**29.07**
Face-2	22.02	26.02	29.50	**29.58**
Face-3	21.96	25.88	**28.27**	27.91
Face-4	21.83	26.37	**28.31**	27.89

B. Comparison with Standard Pruning Methods

In section-4.2, we briefly showed some results on comparison of OES with standard pruning methods that comprised of pruning at Initialization methods like Synflow, Grasp and magnitude/random based pruning and pruning after training methods like Iterative magnitude pruning. In Table-4, we show all the results for images in three different datasets: Set-14, Face, and Standard image. All the PSNR values were noted at convergence. Our observation suggests that OES outperforms the traditional pruning methods at initialization. We did not report the performance of SNIP as it resulted in layer collapse for Unet. We see that magnitude and random choice of parameters serve as a bad indication of importance score and most often than not leads to layer-collapse. We explored this part in the manuscript in Section 4.2. Synflow, Grasp pruning at initialization leads to overfitting when run for longer iterations. Lastly, our comparison with IMP (Iterative magnitude pruning), shows that using the mask obtained from converged DIP training easily leads to overfitting of the masked subnetwork. We implement IMP with two schedules: IMP-$(0.8)^{14}$ denotes pruning and training was run for 14 iterations and at each iteration 20% of the remaining weights were pruned, IMP-$(0.2)^3$ denotes pruning and training was run for 3 iterations and at each iteration 80% of the remaining weights were pruned. Having gradual pruning performed better when compared to aggressive pruning. This further shows that runnign IMP to get good masks can be costly since we need to run more iterations of pruning to reach a desired sparsity level.

C. Details of the Gumbel Softmax Reparameterization Trick

Let s be the final number of non-zero elements we want to have in the subnetwork and d is the total number of parameter. Then we fix the prior to be $\mathbf{p}_0 = \frac{s}{d} \times \mathbf{1}$, which means each parameter will have a prior probability p_0 for selecting the weight. We solve the following optimization problem using the Gumbel softmax reparameterization trick, but first we

Table 4. Denoising capabilities ($\sigma = 25dB$) comparison without early stopping for standard pruning methods for images from 3 different datasets. PAI refers to Pruning At Initialization. PAT refers to Pruning After Training. All networks have sparsity level of 5%. IMP refers to Iterative Magnitude Pruning (IMP) with weight rewinding. IMP-$(1-p)^n$ denotes at each pruning iteration $p\%$ of weights have been deleted and has been run for n number of pruning iterations. None of the methods use clean image for training.

	Image	PAI				PAT		Ours (PAI)
		GraSP	SynFlow	Magnitude	Random	IMP-$(0.8)^{14}$	IMP-$(0.2)^{3}$	OES
Set-14	Pepper	22.22	22.07	12.42	10.80	25.52	25.66	**27.45**
	Foreman	21.67	20.93	12.13	10.75	21.66	23.78	**25.15**
	Flowers	23.02	23.07	12.22	10.61	26.11	26.43	**27.10**
	Comic	21.07	21.50	12.13	11.75	22.13	22.42	**24.03**
	Lena	13.39	22.19	14.37	13.33	25.73	25.75	**26.40**
	Barbara	23.45	23.51	13.56	13.03	26.20	26.05	**26.50**
	Monarch	22.67	22.93	14.52	12.73	26.27	26.52	**27.62**
	Baboon	22.42	22.56	12.50	11.61	23.75	23.49	**24.00**
	Ppt3	23.42	24.23	9.54	8.51	26.34	26.22	**26.90**
	Coastguard	20.78	20.73	13.31	13.16	21.38	21.90	**24.19**
	Bridge	24.25	23.29	13.36	13.10	**26.27**	26.20	26.12
	Zebra	22.92	23.22	13.29	12.34	26.56	26.54	**27.40**
	Face	21.38	21.14	10.74	9.50	21.80	22.50	**24.53**
	Man	23.62	23.71	12.89	11.30	**26.82**	26.72	26.59

	Image	PAI				PAT		Ours (PAI)
		GraSP	SynFlow	Magnitude	Random	IMP-$(0.8)^{14}$	IMP-$(0.2)^{3}$	OES
Face	Face-1	22.88	22.97	12.64	8.41	26.89	26.62	**29.07**
	Face-2	22.64	22.90	12.35	10.40	26.74	26.19	**29.58**
	Face-3	22.74	22.80	13.46	11.86	26.94	26.50	**27.91**
	Face-4	22.71	22.57	12.16	11.61	26.33	26.46	**27.89**

	Image	PAI				PAT		Ours (PAI)
		GraSP	SynFlow	Magnitude	Random	IMP-$(0.8)^{14}$	IMP-$(0.2)^{3}$	OES
Standard	House	20.46	20.24	13.51	13.20	26.61	26.88	**29.27**
	Building	22.72	22.52	15.32	13.23	26.30	26.02	**27.23**
	Door	21.87	21.80	12.32	10.49	26.80	26.46	**28.18**
	Hats	22.43	21.60	11.20	12.45	25.97	25.92	**26.07**

explain the challenges of solving this optimization problem:

$$\begin{aligned} &\mathbf{m}^*(\mathbf{y}) = C(\mathbf{p}^*) \quad \text{such that} \\ &\quad \mathbf{p}^* = \arg\min_{\mathbf{p}} \underbrace{\mathbb{E}_{\mathbf{m}\sim Ber(\mathbf{p})}\left[||G(\boldsymbol{\theta}_{in}\circ \mathbf{m},\mathbf{z}) - \mathbf{y}||_2^2\right]}_{R(\mathbf{p})} \\ &\quad + \lambda KL(Ber(\mathbf{p})||Ber(\mathbf{p}_0)) \end{aligned} \tag{3}$$

The standard way to minimize $R(\mathbf{p})$ is to obtain a direct Monte Carlo estimate of $\partial_{p_i}R(\mathbf{p})$ for every $i = 1, 2, .., d$ by several random realizations of the network. Let $Q := Ber(\mathbf{p})$ denote the posterior distribution. Then for every i, let $e_i(m_i') = \mathbb{E}_Q\left[||G(\boldsymbol{\theta}_{in}\circ\mathbf{m},\mathbf{z}) - \mathbf{y}||_2^2 | m_i = m_i'\right]$, we have $R(\mathbf{p}) = p_i e_i(1) + (1-p_i)e_i(0)$, which yields $\partial_{p_i}R(\mathbf{p}) = e_i(1) - e_i(0)$. Finding the Monte Carlo estimate of $\partial_{p_i}R(\mathbf{p})$ is computationally infeasible because of computing the conditional expectation for every i. The loss $R(\mathbf{p})$ depends on $\mathbf{p}$ in an implicit way and calculating the gradient $\partial_{\mathbf{p}}R(\mathbf{p})$ using Monte Carlo samples is not straightforward.

To make the relation of the loss $R(\mathbf{p})$ and variable $\mathbf{p}$ explicit for gradient based methods, a classical approach called the *reparameterization trick* is used to find the mapping that makes it explicit. For discrete Bernoulli distribution, the reparamterization trick is called the Gumbel-Max (GM) trick which is a method of sampling from discrete random variables using explicit dependence on the probabilities of each state. The GM trick allows straightforward simulation of discrete variables, but it is not practical for gradient computing because it involves differentiation through a max function. To overcome this disadvantage of GM trick, Maddison et al. (2016) introduced the Gumbel-Softmax trick which relaxes the discrete distribution to CONCRETE distribution: CONtinuous relaxations of disCRETE random variables. Let $T \in \mathbb{R}^+$ denote the temperature which controls the degree of relaxation from the discrete distribution to the continuous distribution. The sampling from the Concrete distribution $Concrete(p_i, 1 - p_i)$ is as follows:

1. fix T and sample $G_k, G_l \sim$ Gumbel i.i.d ($-\log(-\log(U[0, 1]))$).

2. set $\hat{m_i}(p_i) = \frac{\exp\,(\frac{(\log(p_i)+G_l)}{T})}{\exp\,(\frac{(\log(p_i)+G_l)}{T})+\exp\,(\frac{(\log(1-p_i)+G_k)}{T})} \sim Concrete(p_i, 1 - p_i)$ for $i = 1, 2, .., d$.

Here, $\hat{\mathbf{m}}$ obeys a *continuous* Concrete distribution denoted as Concrete($\mathbf{p}$,$\mathbf{1} - \mathbf{p}$) instead of discrete Bernoulli distribution $Ber(\mathbf{p})$. This continuous approximation of discrete distribution is controlled by the temparature variable T. As T tends to 0, Concrete($\mathbf{p}$,$\mathbf{1} - \mathbf{p}$) distribution converges to the $Ber(\mathbf{p})$ distribution, however, for small T, there are numerical instability issues in estimating $\hat{m_i}$. In our experiments, T is fixed to 0.2 for all the experiments, as it gives a good approximation to the discrete distribution without suffering from the numerical instability issue. Note that unlike in Bernoulli distribution $\mathbf{m} \sim Ber(\mathbf{p})$, where the dependence of $R(\mathbf{p})$ on $\mathbf{p}$ was implicit, making (4) challenging to optimize, for Concrete distribution the dependence on $\mathbf{p}$ is explicit, making it amenable to solve by gradient based optimizers. So, given random network initialization, $\boldsymbol{\theta}_{in}$, and the noisy corrupted image $\mathbf{y}$, the steps to learn mask $\mathbf{m}$ to the model parameters are as follows.

Algorithm 1 Optimal Eye Surgeon (Learning Mask at initialization)

```
Input: $\boldsymbol{\theta}_{in}$, $\mathbf{p}_0$, $\mathbf{y}$, $G(., \mathbf{z}), C(.)$, number of samples $K$
Output: Final mask $\mathbf{m}^*(\mathbf{y})$
Initialize $\mathbf{p} = 0.5 \times \mathbf{1}$, set $T = 0.2, \lambda = 1e - 9$
for each iteration do
    for $k = 1$ to $K$ do
        $\hat{\mathbf{m}}^k(\mathbf{p}) \leftarrow$ Concrete($\mathbf{p}, 1 - \mathbf{p}$)
        $L^k(\mathbf{p}) \leftarrow \|G(\boldsymbol{\theta}_{in} \circ \hat{\mathbf{m}}^k(\mathbf{p}), z) - \mathbf{y}\|_2^2$
    end for
    $L_C(\mathbf{p}) \leftarrow \frac{1}{K}\sum_{k=1}^{K} L^k(\mathbf{p}) + \lambda \mathrm{KL}\,(\mathrm{Ber}(\mathbf{p})||\mathrm{Ber}(\mathbf{p}_0))$
    Compute $\nabla_{\mathbf{p}} L_C(\mathbf{p})$, do GD : $\mathbf{p} \leftarrow \mathbf{p} - \eta \nabla_{\mathbf{p}} L_C(\mathbf{p})$
end for
$\mathbf{m}^*(\mathbf{y}) \leftarrow C(\mathbf{p}^*)$, where $\mathbf{p}^*$ is the converged probability mean.
```

While optimizing (4) by Algorithm-1, we reparameterize the optimization variable $\mathbf{p}$ through a sigmoid function $\mathbf{p} = sigmoid(\boldsymbol{v})$, which maps the domain of the variable $\mathbf{p}$ from $[0, 1]$ to the optimization variable $\boldsymbol{v}$:$[-\infty, \infty]$. So our initialization, which ensures unbiased selection of weights is at $\boldsymbol{v} = sigmoid^{-1}(\mathbf{p}) = sigmoid^{-1}(0.5) = 0$. The prior probability which controls sparsity, is also related as $\boldsymbol{v}_0 = sigmoid^{-1}(\mathbf{p}_0)$ where $\mathbf{p}_0$ is the prior probability vector. This reparameterization of the optimization variable ensures that the variable domain is not restrictive.

Once $\mathbf{p}^*$ is obtained by gradient descent, mask $\mathbf{m}^*(\mathbf{y}) \leftarrow C(\mathbf{p}^*)$ is obtained by ranking the elements of $\mathbf{p}^*$ and setting the indices of $\mathbf{m}^*(\mathbf{y})$ corresponding to the top $s\%$ values of $\mathbf{p}^*$ to be 1, and 0 otherwise. This way the sparsity of the mask is set to be the desired sparsity s and is accomplished by the $C(.)$ function. $C(.)$ is a ranking function, which ranks the values of $\mathbf{p}$ and then thresholds the weight indices corresponding to $s\%$ highest values of $\mathbf{p}$ to achieve the desired sparsity. We chose the initialization $\mathbf{p} = 0.5 \times \mathbf{1}$, so that there is no bias towards any weight selection and all weights have equal probability of selection/pruning at initialization. Although with prior knowledge, for certain layers $\mathbf{p}$ can be initialized to higher probability values, but in our preliminary experiments, we do not introduce bias towards any weights in any layers.

D. Differences with Related Work

While we provide an interesting insight on the image generation capability of hour-glass Unet architecture, we acknowledge the existence of previous works which further substantiate our current findings. In the following points, we highlight the difference of our work with the following and also mention the similarity of the findings:

D.1. Comparison to NAS-DIP (Chen et al., 2020) and ISNAS-DIP (Arican et al., 2022)

NAS-dip proposes to apply the NAS (Neural architecture search) algorithm on DIP framework. They build a searching space for upsampling cells in the decoder and the skip connections between encoder and decoder. Then they leverage reinforcement learning (RL) with RNN controller and use *PSNR wrt clean image* as reward to guide the architecture. After the network search, they transfer the best-performing architecture and optimize the model the same way as DIP. We highlight the points of difference:

- *Architecture search vs. pruning*: Chen et al. (2020) and Arican et al. (2022) search for the best architecture. The final architecture found by NAS-DIP is a dense architecture. Instead, we start with a dense deep Unet architecture and then make it sparse. Instead of searching for the best architecture combination, we focus on each weight parameter and evaluate it's importance in the context of image generation. Infact, a NAS-DIP model found by Chen et al. (2020) can be further pruned by OES.

- *Limited search space*: NAS-DIP searches over only the upsampling and residual connections. For OES, a 6 layer encoder-decoder network is the base architecture and each parameter gets it's individual importance metric through learning $\mathbf{p}$. We believe that although upsampling layers play a crucial role, the encoder layers can't be entirely discarded.

- *Using clean image to find architecture*: We want to emphasize this is the main point of difference between the previous works like Chen et al. (2020); Arican et al. (2022); Wu et al. (2023) and ours. *We do not need to use the clean image for pruning the network. Masking at initialization induces image prior even when trained against a corrupted image.* We discuss this phenomenon in detail in Finding-1.

D.2. Comparison to The Devil is in the Upsampling (Liu et al., 2023)

Liu et al. (2023) proposed a heursitic strategy for designing appropriate architecture by analyzing the frequency response of architecture parts of DIP. Their observation was that the bilinear upsampling layers are the most important parts for image generation. Followed by the convolutional layers as they observed that only when the decoder part is used, convolutional decoders performed better than non-convolutional or MLP decoders. Furthermore, they suggest whether to increase/decrease depth or width or whether to keep skip connections (or not) based on signal processing intuition and sanity check based experiments. Our Alorithm OES relies on the mask learning algorithm to convey the similar information obtained in Liu et al. (2023) and both these works agree on three findings.

1. *Importance of decoder*: In Figure-10b, we also find that given a hour-glass Unet architecture, the decoder part seems to be more important while the encoder part is more compressible. This is the main finding in Liu et al. (2023) based on the frequency response of the upsampling layer. However, in OES, the final converged value of $\mathbf{p}$ conveys this information.

2. *Reduced depth in Unet*: For hour-glass architecture, the authors observe that increased depth can lead to oversmoothing of final image. Hence, for decoder architectures, the authors advocate reduced upsampling operations and for Unet architecture, they advocate decreasing the depth of the network. In Figure-10b, we see that the converged and thresholded value of $\mathbf{p}$ conveys the same finding. For 6 layer Unet architecture, the middle layer of the encoder-decoder architecture seems to get pruned the most showing a 'W' shape in encoder-decoder hour glass architecture. This denotes that we can do with reduced depth.

3. *Not using skip connections*: The authors notice that the skip connections ameliorate the oversmoothing issue when the network has large depth. Hence, they may lower the effective upsampling rate, making deep networks perform similarly to shallower ones. Thus in our base architecture, we use the simple hour-glass Unet architecture. Trying to understand and analyze OES with skip architecture can be more complicated and we leave it as future work.

We also want to highlight one point of difference in the findings between these two works. We observe that using an irregular pruned Hour-glass architecture outperforms deep decoder based architecture. Hence, although *devil is in the upsampling layers*, we observe that the encoder-decoder junction also plays a crucial role.

D.3. Comparison to Lottery Image Prior (Wu et al., 2023)

The main message of our paper was to advocate learning the mask at initialization instead of learning the mask based on magnitude obtained post-training. However, we acknowledge that Wu et al. (2023) is the first work to apply unstructured pruning for image reconstruction based problems. However, they use the early stopping time to obtain the mask through Lottery Ticket Hypothesis. Generally speaking, identifying the early stopping time in image reconstruction tasks is itself a challenging task. It is hard to come up with an estimate of an early-stopping time based on observations from different images and corruption levels. For example in Figure-21, we see that for two different images with two different corruption levels, the early stopping time can vastly vary. We also observe that in their python script in their github repo they use the clean image to train the mask for a single image. In our experiments, we show the adverse effects of LTH based masks obtained at convergence. But further we also compare our method when LTH masks are obtained at early-stopping time or using clean images. We observe that LTH based masks obtained at early stop time perform well when the image used for training the mask is also used for denoising in Figure-14-a. But when a different image is used for denoising, the transferability of OES masks seems to be better (Figure-14-b). In Table-5, we compare the transferability of OES masks and IMP based masks. Here, OES masks are obtained at initialization and LTH based masks are obtained by training the network to convergence but with a clean image. We study the pruning pattern of Unet architecture in details and compare OES and IMP methods, something that was not studied comprehensively (Wu et al., 2023).

E. Comparing IMP-based Denoising

In Finding-4, in the manuscript, we discussed the transferability of OES masks and compared how these masks transferred with the same image, within images of the same dataset and within images of varying datasets. Here, in this section, we report additional performance where we use the mask learned on Lena image (clean and noisy) at 5% sparsity. In Table-5, we compare the performance of OES masks with IMP masks at convergence for several noise levels. Here in Table-5, the IMP masks were learned on the noisy Lena images. We demonstrate the corresponding figures in Figure-13. We run the denoising algorithm till 40k iterations. We see that in Figure-13, the IMP based masks overfit to noise, whereas OES-masks learned at initialization do not overfit. For this particular experiment, we do not use any knowledge of early-stopping time, so at convergence the parameters overfit to noisy Lena image. The IMP mask in Table-5 is obtained based on the magnitude of these parameters.

Table 5. Comparison of denoising capabilities (for various noise levels) of transferred masks for OES vs IMP based pruning. $\mathbf{y}_{source}$ used is the **noisy Lena image**. All masks are 5% sparse.

Image (y_{target})	$\sigma = 25dB$		$\sigma = 12dB$		$\sigma = 17dB$	
	$m(IMP)$	$m(OES)$	$m(IMP)$	$m(OES)$	$m(IMP)$	$m(OES)$
Pepper	26.57	**27.05**	29.66	**30.37**	27.92	**28.55**
Flowers	26.17	**27.10**	30.03	**31.02**	28.54	**29.31**
Lena (self)	25.85	**26.35**	29.36	**30.95**	28.45	**28.89**
Barbara	25.31	**26.34**	28.60	**30.36**	27.30	**28.43**
Monarch	26.45	**27.38**	31.01	**32.84**	29.14	**30.23**
Baboon	23.26	**23.91**	24.87	**25.25**	24.24	**24.89**
Ppt3	26.11	**26.96**	30.92	**32.32**	29.05	**29.57**
Bridge	25.09	**26.17**	28.06	**28.74**	26.93	**27.54**
Zebra	26.34	**27.20**	30.54	**31.45**	28.80	**29.87**
Man	25.83	**26.92**	29.67	**30.22**	27.99	**29.13**

To further make an apple-to-apple comparison with Wu et al. (2023), we compare our method when the clean Lena and Pepper images were used to learn the mask. We observe that when IMP uses clean Lena and Pepper image for learning the mask, the denoising performance is improved as compared to when IMP only used the corrupted image. Like in Table-5, the final PSNR achieved when IMP used the noisy image was 25.85dB (Lena-self in Table-5) whereas when IMP used the clean image to learn the mask, the denoising performance improved to 26.65 dB (Lena-self in Table-6b). But the improvement, for denoising other images does not increase when compared to OES. For example, *the PSNR of IMP masks using the clean image (26.65 dB (Lena-self in Table-6b) is still less than when OES used the corrupted image (27.05 dB in Table-5).* When the target image was different, say for Barbara image, the mask learned on clean Lena image using IMP gives a PSNR of 25.66 dB (Table-6b) but using OES mask with a corrupted image gives PSNR of 26.34 dB (Table-5, Barbara). We further explore this phenomenon of transferability in Figure-14 where IMP masks learned on clean image performed well, when it was used for denoising on the same image but performed worse than OES when it was used for a different image.

Table 6. Comparison of Denoising Capabilities of Transferred Masks Obtained from Sparse-DIP Pruning at Initialization vs IMP/OES Based Pruning. Here, both the OES and IMP masks were learned on *clean Lena image.*

(a) Masks learned on **clean Pepper image**.

Image	$\sigma = 25dB$	
	$m(IMP)$	$m(OES)$
Pepper (self)	26.89	**27.68**
Flowers	26.48	**26.80**
Lena	25.96	**26.38**
Barbara	25.42	**26.32**
Monarch	26.73	**27.40**
Baboon	23.49	**23.89**
Ppt3	26.36	**26.84**
Bridge	25.75	**26.03**
Zebra	26.58	**27.20**
Man	26.19	**26.94**

(b) Masks learned on **clean Lena image**.

Image	$\sigma = 25dB$	
	$m(IMP)$	$m(OES)$
Foreman	26.39	**26.67**
Lena(self)	26.65	**26.83**
Barbara	25.66	**26.46**
Monarch	26.61	**27.35**
Baboon	23.65	**24.02**
Ppt3	23.19	**26.85**
Bridge	25.81	**26.21**
Man	26.31	**26.88**

We observe a similar phenomenon in transferability in Figure-14 when IMP masks were obtained at an early stopping time.

F. Transfer to Different Task: Inpainting

In the manuscript, we performed on learning mask $\mathbf{m}(\mathbf{y})$ from noisy images $\mathbf{y}$, where $\mathbf{y}$ is corrupted by additive Gaussian noise with standard deviation σ. In this section, we show the efficiency of OES masks, when $\mathbf{y}$ is a masked image with probability of masking $p = 0.5$, i.e, on average 50% of the image pixels are missing. We compare the masks learned when $\mathbf{y}$ had missing pixels (referred to as Sparse-DIP-in) and when $\mathbf{y}$ was corrupted with Gaussian noise (referred to as Sparse-DIP-den). We evaluate the relative comparison of both these masks against deep decoder and dense DIP at convergence. We report the results of these masks in the inpainting task in Table-7. Furthermore, we use this set of masks for denoising in two different noise levels $\sigma = 25dB$ and $\sigma = 12dB$ and report in Table-8. Based on this observation in Table-7, we see that sparse-DIPs (mask learned from missing pixel $\mathbf{y}$ or noisy $\mathbf{y}$) seems to perform comparably with Deep-decoder and Vanilla DIPs. This is because for inpainting tasks, the effect of overfitting is not as pronounced as compared to denoising tasks. Both the masks learned from denoising task and the inpainting task seem to perform comparably in Table-8.

Table 7. Comparison of inpainting capabilities of transferred masks (denoising training) obtained from Sparse-DIP pruning at initialization vs IMP based pruning. $p = 0.5$

	Dense DIP	Deep Decoder	Sparse-DIP (den)	Sparse-DIP (in)
Ppt3	**28.62**	28.40	28.43	28.33
Baboon	21.36	22.13	**22.60**	22.38
Coastguard	**27.80**	27.27	27.45	27.45
Man	25.47	**26.63**	26.13	26.16
Zebra	31.20	29.52	31.62	**31.27**
Pepper	28.40	28.45	30.76	**30.81**
Face	28.64	**31.27**	31.12	29.10
Comic	22.36	**24.53**	22.55	22.54
Flowers	30.73	29.61	**31.10**	30.85
Bridge	24.85	**25.16**	25.01	24.78
Foreman	31.57	33.60	30.75	31.53
Monarch	30.54	31.08	31.44	**31.70**
Barbara	**27.62**	25.71	27.23	27.28
Lena	28.85	**31.23**	31.17	31.17

However, for denoising task in Table-8, we see that Sparse-DIP (learned through denoising and inpainting loss) outperforms both Deep decoder and Dense-DIP due to severe overfitting. This is something we already explored in Table-2.

Table 8. Denoising capabilities comparison without early stopping on Set-14 dataset. The Sparse-DIP masks have been generated with two procedures. ”denoise” denotes the masks generated from the denoising operation. ”inpaint” denotes the masks generated from the inpainting operation.

	$\sigma = 25dB$				$\sigma = 12dB$			
Image	Dense	Deep	Sparse-DIP		Dense	Deep	Sparse-DIP	
	DIP	Decoder	denoise	inpaint	DIP	Decoder	denoise	inpaint
Pepper	21.21	27.08	27.45	**27.46**	27.34	28.81	29.89	**30.40**
Foreman	20.69	24.20	**25.15**	24.36	26.59	29.81	30.12	**30.37**
Flowers	22.27	27.03	**27.10**	27.10	28.55	30.33	**31.07**	31.07
Comic	20.63	23.35	**24.03**	23.96	26.45	28.01	28.57	**28.81**
Lena	21.28	**26.85**	26.40	26.23	27.50	**30.96**	30.89	30.85
Barbara	23.90	25.30	**26.50**	26.25	28.27	27.50	29.85	**30.09**
Monarch	23.62	27.87	**27.87**	27.82	28.25	32.00	32.12	**32.67**
Baboon	21.68	22.93	24.00	**24.10**	27.27	24.12	**25.91**	25.90
Ppt3	24.07	26.81	**27.20**	26.64	28.88	31.73	32.41	**32.50**
Coastguard	20.53	23.71	**24.19**	24.16	26.50	29.43	**30.60**	29.80
Bridge	21.77	25.19	**26.12**	26.10	28.58	28.10	**29.23**	28.98
Zebra	21.94	**27.40**	27.32	27.29	28.45	30.81	31.54	**31.62**
Face	21.03	24.14	**24.18**	24.05	26.90	29.93	**29.93**	29.86
Man	21.98	26.32	**26.59**	26.55	28.45	29.84	**30.94**	30.56

G. Sensitivity of λ in OES Mask Learning and Selectivity of $\mathbf{p}_0$

G.1. Sensitivity of λ

We found empirically that OES algorithm is robust to the choice of λ, given a network architecture with fixed number of parameters (Unet in this case). We fix $\lambda = 1e - 9$ for all our experiments. For this particular experiment, we take $\mathbf{p}_0 = 0.05 \times \mathbf{1}$ and threshold 95% of the weights by ranking $\mathbf{p}$. The initialization value was taken to be at $\mathbf{p} = 0.5 \times \mathbf{1}$, where all the weights have equal chance of selection or deletion. λ controls the regularization balance on fitting the image (first part of the loss) or by making the distribution $Ber(\mathbf{p})$ close to $Ber(\mathbf{p}_0)$ (second part of the loss). Note that $\mathbf{p}_0$ is the pre-specified prior probability that is same for all the parameters of the network. As $\lambda \to \infty$, then $\mathbf{p} \to \mathbf{p}_0$, at this limit a) there is no image generation at initialization, as the first part of the loss is not minimized and b) there is no separation among the converged values $\mathbf{p}$ and the probabilities for all the elements will collapse to $\mathbf{p}_0$. So the mask can’t be formed by ranking and thresholding at the desired sparsity level.

$$
\begin{aligned}
\mathbf{m}^* = C(\mathbf{p}^*) \quad & \text{such that} \\
\mathbf{p}^* = \arg\min_{\mathbf{p}} & \underbrace{\mathbb{E}_{\mathbf{m}\sim Ber(\mathbf{p})}\left[||G(\boldsymbol{\theta}_{in} \circ \mathbf{m}, \mathbf{z}) - \mathbf{y}||_2^2\right]}_{R(Q)} + \lambda KL(Ber(\mathbf{p})||Ber(\mathbf{p}_0(s)))
\end{aligned}
$$

$\lambda = 1e - 3$ correspond to this observation in Figure-18. Increasing it to $\lambda = 1e - 6$, we observe that $\mathbf{p}$’s for different weights start to vary and are not entirely localized at $\mathbf{p}_0$. However, even in this case, ranking the values of $\mathbf{p}$, leads to layer collapse. Layer collapse happens in this phenomenon because important weights are thresholded. For smaller $\lambda = 1e - 13$, we observe that the distribution $\mathbf{p}$, is uniform around the initialization $\mathbf{p} = 0.5 \times \mathbf{1}$. Although the image is formed by masking in this case, the distribution remains uniform. We see that at $\lambda = 1e - 9$, the distribution of $\mathbf{p}$ seems to have two modes. We see a clear distinction where some of the $\mathbf{p}$’s are localized at 1 and other is centered around $\mathbf{p}_0$. This leads to better separation while thresholding and pruning the weights based on $\mathbf{p}$. However, we note that the value of the KL would depend on the size of the network, for the current Unet architecture we are using, which has 3 million parameters, we found that $1e - 9$ works the best among all the other values in logarithmic scale.

H. OES pruning for MRI reconstruction

We extend the OES pruning and sub-network training framework to the setting of multi-coil magnetic resonance image (MRI) reconstruction from undersampled k-space measurements. In previous literature [4], dense networks based DIP was used for MRI reconstruction as folows:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{c=1}^{N_c} \|\mathbf{A}^{(c)} G(\boldsymbol{\theta}, \mathbf{z}) - \mathbf{y}^{(c)}\|_2^2 \qquad \text{(P1:Vanilla DIP)}$$

For multi-coil MRI, let there be N_c number of coil sensitivity maps denoted as $\mathbf{S}_c \in \mathbb{C}^{q \times q}$, $c = 1, 2.., N_c$. The corresponding $\mathbf{A}^c$ denotes the undersampled forward linear operator $\mathbf{A}^c(\mathbf{M}) = \mathbf{M}\mathcal{F}\mathbf{S}_c$. $\{\mathbf{M} \in \{0,1\}^{q \times q}\}$ is the sampling mask in k-space, $\mathcal{F} \in \mathbb{C}^{q \times q}$ denotes the Fourier Transform operator and $\mathbf{y}^{(c)} \in \mathbb{C}^q$ denotes the undersampled k-space measurements. $G(\boldsymbol{\theta}, \mathbf{z})$ is an overparameterized Unet with two channels that processes the real and complex channel separately and with trainable parameters $\boldsymbol{\theta}$ and fixed input $\mathbf{z}$. For our experiments, we use multi-coil fastMRI knee and brain datasets [1,2] which are available publicly. The coil sensitivity maps were obtained using the BART toolbox [3]. When the dense network [4] is trained with generic optimizer like ADAM, the above suffers from overfitting (Figure-24a). In the OES framework, we first learn the mask for the subnetwork, denoted as $\mathbf{m}^*(\mathbf{A}, \mathbf{y})$ (not to be confused with the k-space mask $\mathbf{M}$), where $\mathbf{A}(\mathbf{M}) = [\mathbf{A}^c(\mathbf{M})]_{c=1}^{Nc}$ and $\mathbf{y} = [\mathbf{y}^c]_{c=1}^{Nc}$. For the sake of notation, we will omit the coil dependency c as the loss can be combined across coils and written in terms of one forward operator $\mathbf{A}$ and measurements $\mathbf{y}$.

$$\begin{aligned} \mathbf{m}^*(\mathbf{y}, \mathbf{A}) = C(\mathbf{p}^*) \quad & \text{such that} \\ & \mathbf{p}^* = \arg\min_{\mathbf{p}} \mathbb{E}_{\mathbf{m} \sim Ber(\mathbf{p})} \left[\|\mathbf{A} G(\boldsymbol{\theta}_{in} \circ \mathbf{m}, \mathbf{z}) - \mathbf{y}\|_2^2 \right] \\ & + \lambda KL(Ber(\mathbf{p}) || Ber(\mathbf{p}_0)). \end{aligned} \tag{4}$$

In Figure-23, we show the 4 MRI scans that are used in the following experiment. $\mathbf{x}$ denotes the ground truth MRI image (obtained from a full set of k-space measurements), $\mathbf{M}_{4\times}$ and $\mathbf{M}_{8\times}$ denote the $4\times$ and $8\times$ undersampling masks for k-space or Fourier space (white lines are sampled), respectively. $\mathbf{A}^H(\mathbf{M}_{4\times})\mathbf{y}$ and $\mathbf{A}^H(\mathbf{M}_{8\times})\mathbf{y}$ denote the conventional zero-filling MRI reconstructions that produce aliasing artifacts. We will denote the set of the forward operator and measurement pair as $(\mathbf{A}_i(\mathbf{M}_{4\times}), \mathbf{y}_i)$ for data index $i = 1, 2, 3, 4$ for $4\times$ undersampling rate. For $8\times$ undersampling rate, we denote the pair as $(\mathbf{A}_i(\mathbf{M}_{8\times}), \mathbf{y}_i)$. In our experiments, we train the OES mask using the pair $(\mathbf{A}_1(\mathbf{M}_{4\times}), \mathbf{y}_1)$, and then use the mask subnetwork to reconstruct MRI in four different scenarios across various network sparsity levels:

1. **Self + same undersampling**: The target reconstruction pair is $(\mathbf{A}_1(\mathbf{M}_{4\times}), \mathbf{y}_1)$. We denote this experiment as $P(\mathbf{A}_1(\mathbf{M}_{4\times}), \mathbf{y}_1)$.
2. **Self+higher undersampling**: The target reconstruction pair is $(\mathbf{A}_1(\mathbf{M}_{8\times}), \mathbf{y}_1)$. We denote this experiment as $P(\mathbf{A}_1(\mathbf{M}_{8\times}), \mathbf{y}_1)$
3. **Cross + same undersampling**: The target reconstruction pair is $(\mathbf{A}_i(\mathbf{M}_{4\times}), \mathbf{y}_i)$ for $i = 2, 3$ and 4. We denote this experiment as $P(\mathbf{A}_i(\mathbf{M}_{4\times}), \mathbf{y}_i)$
4. **Cross + higher undersampling**: The target reconstruction pair is $(\mathbf{A}_i(\mathbf{M}_{8\times}), \mathbf{y}_i)$ for $i = 2, 3$ and 4. We denote this experiment as $P(\mathbf{A}_i(\mathbf{M}_{8\times}), \mathbf{y}_i)$.

Note that transfer to a higher undersampling rate demonstrates the capability of transferring to a different level of degradation. Once the mask $\mathbf{m}^*(\mathbf{A}_1, \mathbf{y}_1)$ is obtained, the subnetwork at initialization is further trained to convergence with the following optimization. Similar notations extend to $8\times$ undersampling rate.

$$\min_{\boldsymbol{\theta}} \|\mathbf{A}_i(\mathbf{M}_{4\times}) G(\boldsymbol{\theta} \circ \mathbf{m}^*(\mathbf{y}_1, \mathbf{A}_1), \mathbf{z}) - \mathbf{y}_i\|_2^2 \qquad \text{(P}(\mathbf{A}_i(\mathbf{M}_{4\times}), \mathbf{y}_i)\text{: Sparse-DIP)}$$

We make the following observations from the PSNR curves in Figure-24.

- *Sparse-DIP reduces overfitting:* Vanilla Dense DIP produces artifact-affected images in all the cases. This is due to the nullspace of the forward operator that does not offer any control over nonsampled frequencies. Sparse DIP has very less overfitting.
- *Sparse-DIP is robust to higher undersampling rate:* For higher undersampling factor, i.e, $8\times$ undersampling, vanilla dense DIP overfits much more. Sparse DIP at higher sparsities (above 90%) seems to be robust to overfitting even at $8\times$ undersampling.

- *Moderate overfitting at moderate sparsity*: With moderate sparsity level ($50\%, 80\%$), subnetwork overfits artifacts when cross transfer tasks take place (different image's measurements) or when the undersampling rate is $8\times$. However, overfitting (at moderate sparsity levels) takes place to much less extent when self transfer takes place with the same undersampling rate $4\times$.

- *Limited representation capability at very high sparsity:* For higher sparsity levels (90% or higher), overfitting rarely happens in any of the scenarios (cross-transfer or higher undersampling rate). For very high sparsity level 97%, the PSNR curve fails to rise very high, denoting that the network has already reached its representation capability.

H.1. Selectivity of Prior $\mathbf{p}_0$ and Thresholding

In our experiments, after the final convergence of our algorithm, we rank and threshold the value of $\mathbf{p}$ to reach the desired sparsity level. An avid reader may ask the question that since the sparsity level is achieved through ranking and thresholding of the $\mathbf{p}$ values, so is the selection of the prior $\mathbf{p}_0$ important in getting a good ranking? Ideally speaking, the ranking should be based on the importance of the parameter in contributing to the loss. That means, a parameter w_1 is considered more important than parameter w_2 in the following case: if the objective when we fix $w_1 = 0$ (say $T(w_1)$) would be more than the objective when we fix $w_2 = 0$ (say $T(w_2)$). So in this case, if $T(w_1) \geq T(w_2)$ then a proper ranking would imply $p(w_1) \geq p(w_2)$. We observe that for $\lambda = 1e-9$ choosing the prior $\mathbf{p}_0$ to be the same as the desired sparsity level provides a good ranking that separates the important parameters from the non-important ones. Like in Figure-18 with $\lambda = 1e-9$, when the prior $\mathbf{p}_0$ is chosen to be the same as the desired sparsity level (5%), we observe that most of the distribution is centered around $\mathbf{p}_0$ with some values $\mathbf{p}$ at 1. In the previous subsection, we discussed how the choice of λ affects this distribution. In this section, we empirically show that the choice of $\mathbf{p}_0$ is crucial in getting OES masks that are suitable for denoising. Fixing $\lambda = 1e-9$, we perform a denoising experiment with different ranges of values where $\mathbf{p}_0$ is as high as $0.5, 0.8$ or as low as $0.05, 0.03$. After the convergence of the loss we rank and threshold 95% of the weights based on the value of $\mathbf{p}$. We see that choosing a high value of $\mathbf{p}_0$ outputs a mask that suffers from layer collapse and hence when further trained to denoise, completely breaks down. This is because, when $\mathbf{p}_0$ is set to be high as 0.5 or 0.8, the distribution of $\mathbf{p}$ across the network is centered at 0.5 or 0.8 respectively. Now when 95% of the weights are thresholded after ranking, w.h.p all the weights in one layer are getting pruned because of improper ranking of $\mathbf{p}$. This phenomenon of layer collapse seems to be avoided when the value of $\mathbf{p}_0$ is chosen to be close to the pruning level. $\mathbf{p}_0 = 0.03$ or $\mathbf{p}_0 = 0.05$ seem to give the same denoising performance when 95% of the weights are pruned.

I. Comparison with L_1 Regularization

In the image classification literature, supermasks have been used for obtaining a subnetwork by Bernoulli masking for example in Zhou et al. (2019) without sparsity control. Sreenivasan et al. (2022) used the ℓ_1 regularization to control the sparsity of the mask and used iterative freezing at every epoch to reach the desired sparsity level. However, we observed that using ℓ_1 regularization can't give a good ranking of $\mathbf{p}$ based on the importance score. The optimal ℓ_1 regularization coefficient can vary for different images but with KL regularization it is the same for all the images.

Through extensive study, we find that in our experiments:

1. That masking based on the ranking of the $\mathbf{p}$ is sensitive to the choice of λ when used in L_1 regularization like in Sreenivasan et al. (2022), i.e, the optimal λ_{L_1} is not the same for two different images (Figure-29). The best λ_{L1} for the Pepper image can lead to a layer-collapse when used for the Flowers image for a desired sparsity level.

2. Controlling the sparsity level through KL regularization leads to a better ranking in $\mathbf{p}$ that can clearly separate out the important weights (Figure-18). Using no (or extremely small) KL regularization, does not lead to a proper ranking of $\mathbf{p}$ based on importance. The best ranking is obtained when the desired sparsity level is the same as the prior probability used in the KL. This alleviates the need to tune the prior $\mathbf{p}_0$ and can be fixed to the target sparsity we want to achieve. We also demonstrate that using a severely different $\mathbf{p}_0$ than the target sparsity can lead to improper ranking which leads to layer-collapse (Figure-15).

3. The regularization strength λ_{KL} is robust when KL regularization is used. We find $\lambda_{KL} = 1e-9$ works for all the images unlike for L_1 regularization. We show this in Figure-18.

Lastly, we want to emphasize that although learning masks by optimizing the Bernoulli probability $\mathbf{p}$ has already been used in several works before, we show that using KL-based regularization gives us robust control over the sparsity we want to achieve. We compare our mask learning method with that of L1 regularization on $\mathbf{p}$, which is known to promote sparsity in $\mathbf{p}$. Sparsity in $\mathbf{p}$ would ensure that the corresponding mask will be 0 with a very high probability. Although unlike in our formulation where we controlled the distribution $Ber(\mathbf{p})$ to be close to some prior distribution $Ber(\mathbf{p}_0)$, in L_1 regularization, we can only make $\mathbf{p}$ sparse.

$$
\begin{aligned}
&\mathbf{m}^* = C(\mathbf{p}^*) \quad \text{such that} \\
&\quad \mathbf{p}^* = \arg\min_{\mathbf{p}} \mathbb{E}_{\mathbf{m}\sim Ber(\mathbf{p})} \left[||G(\boldsymbol{\theta}_{in} \circ \mathbf{m}, \mathbf{z}) - \mathbf{y}||_2^2 \right] + \lambda \|\mathbf{p}\|_1
\end{aligned}
$$

We solve the above optimization using the same algorithm as in Algorithm-1, but change the regularization term to $\|\mathbf{p}\|_1$ (scaled by λ) instead of the KL term. Here λ would control the sparsity level of $\mathbf{p}$, a higher λ would ensure more $\mathbf{p}$ is towards zero. Just like in OES, we also rank the $\mathbf{p}$ values and threshold them at the desired level of sparsity. We observe through experiments that obtaining a reasonable network mask at initialization is sensitive for L_1 regularization. In Figure-29, we see that the optimum λ for the Pepper image and the Flowers image are different. For $\lambda = 1e-9$, the mask produced by the Pepper image gives the best image representation, while for Flowers image, the best $\lambda = 1e-8$. In fact for $\lambda = 1e-9$, the Flower image seems to suffer from layer-collapse resulting in a constant image. This is unlike in the loss used for KL regularization, where $\lambda = 1e-9$ performed consistently for all images.

I.1. Comparing KL, ℓ_1 and Centered Mean Regularizaion

We further investigate the use of ℓ_1 regularization (given as $||p||_1$) and the centered mean regularizer (given as $|mean(p) - \frac{s}{d}|$), where we take $\frac{s}{d} = 0.05$.

When we minimize the objective with the centered mean regularizer and monitor the value of $mean(p)$, we see that starting from $p = 0.5$ the loss can decrease to $p = 0.05$ but not more, where it becomes stationary and does not change over 10 thousands of iterations (Figure-25). During this phase, this penalty has the same gradient as the ℓ_1 norm regularizer. However, after $mean(p)$ reaches 0.05, the mean p becomes stationary and the loss seems to get stuck, although the penalty might behave differently than ℓ_1. So, the overall effect of ℓ_1 and the centered mean regularizer are similar.

Now, comparing ℓ_1 regularizer to the KL regularizer, we notice across various experiments that ℓ_1 regularization is less stable to the choice of the regularization strength. This is because ℓ_1 regularizer encourages sparser solutions (for centered mean, (p-0.05) is sparse) than KL regularizer. This enforces a bulk of p values to collapse on the same point. Hence the relative ranking gets lost due to this effect.

For example, when the logits corresponding to the three regularizers are plotted in Figure-28, the logits in KL regularization seems to be more well spread than the ℓ_1 and centered mean regularizer. When we look at the corresponding layerwise architecture in Figure-27, we see that the middle layers are severely pruned by ℓ_1 and centered mean regularization which may lead to layer collapse. We intentionally plot the sparsity percentage on the log scale to show the severity of this effect.

So, based on this empirical observation, we think that sparser solutions may not be ideal for bringing the data misfit loss down (since loss of rank importance may lead to layer collapse). Furthermore, enforcing sparsity shrinks the search space of gradient descent, so it may be more likely to get stuck in local minima.

I.2. On using Pointwise Regularization

From our experiments, we observed that using a pointwise regularizartion chosen with a proper regularization strength preserves the ranking. For KL regularization penalty, the ranking would remain preserved for very large range of moderate values of regularization coefficient λ, especially when compared to other pointwise regularization choices like $mean|p - p_0|$. This is because for KL regularization, the regularizer takes very low values around a large window $[p_0 - \epsilon, p_0 + \epsilon]$ (Figure-26). This is not true for linear pointwise regularizers such as ℓ_1.

We want to emphasize that pointwise regularization may allow the implementation of non-uniform prior p_0 across various weights in Unet. That's why we presented a more generic implementation, so if the user has some prior knowledge on what parameters are more important, they have the flexibility to modify the corresponding prior value.

J. Sensitivity of Masks to Weight Initialization

J.1. Change in weight distribution (Uniform/ Normal initialization)

Unlike other methods, OES learns the mask at initialization where the parameters are drawn from random Uniform distribution (He/Kaiming initializaiton) by Pytorch's default implementation. We check that the denoising performance of the mask is not affected by the distribution of the initialization. Changing the distribution to Normal Xavier distribution does not significantly affect the denoising performance of the OES masks. In Figure-31, we show that across 4 various images, the performance of masks learned either at uniform initialization or Normal Gaussian initialization remains the same.

J.2. Scale of Initiailization

We observe that the scale of the initialization seems to affect the learned mask. So, in the experiment in Figure-30, we scale the original He initialization by 0.1 and 5 times respectively and then learn the mask by OES on the Pepper image. We observe that with a smaller scale of initialization, the learned OES mask seems to perform better in terms of denoising. On the contrary, masks learned at $5\times$ initialization, seem to overfit slightly.

J.3. Initial Weights are at Early-Stopping Point

Finding an early stopping point is a challenging task without the knowledge of the ground truth image. So performing IMP based pruning at the early-stopping point is too ambitious. In the following experiment, we show that even if we had an estimate of the early-stopping point, IMP based pruning may not be the best option. In Figure-14, we compare the denoising performance of three different masks: 1) IMP masks obtained at convergence on training with Pepper, 2) OES at initialization when Pepper is used in the loss function and 3) IMP masks obtained at the early stopping point also trained on Pepper (with the assumption that early stopping point is known). In this particular setting, when the target $\mathbf{y}$ is the corrupted Pepper image, we observe that IMP obtained at early-stopping point performs as well as OES. However, when the same 3 masks are used for denoising the Flower-image in Figure-14, we observe that the performance of IMP (at early stopping) degrades with respect to the OES mask.

K. Pruning Deep Decoders by OES

In the manuscript, we showed that pruning a random-initialized Unet with 6 layers can give good starting point for further doing image reconstruction using just the masked subnetwork. Here we apply the OES methodology on the deep decoder architecture (Heckel & Hand, 2018). Deep decoders only consist of upsampling operations as the source of getting low-frequency components in an image. In Figure-32, we compare the images produced by the masked decoder at 55% sparsity and compare it with images with masked Unet at 3% sparsity, along with the corrupted versions. Since, the decoder is already underparameterized and acting as a natural image prior, masking at initialization seems to oversmoothen the image. There seems to be patches of bright and dark areas in the sparse decoder output when the parameters are just masked. On the contrary, for sparse Unets, the information lost due to oversmoothing is not that drastic. This is because decoders are already underparameterized, constraining the output space of decoder to have low frequency componenets. Further pruning by OES at masking leads to oversmoothing and loss of information. We observe that these sparse decoders are compressible by OES upto 74% after which the output image is failed to produce due to layer collapse. In Table-9, we perform denoising using the masked decoder subnetworks for three different images. We observe that at 27% sparsity level deep decoder performs comparably with it's original dense counterpart. However, for higher sparsity levels like 55% and 74%, the performance starts to detoriate. When we observe the layer-wise sparsity pattern produced in deep decoder at 3 different sparsity levels in Figure-33, we observe that the first and last layer seems to the most important. The importance of parameter layers seems to be gradually diminishing towards the middle. This is similar to the finding in Figure-10b where the middle of the encoder and the decoder architecture was pruned the most. With the study of masking deep decoder, we motivate one fundamental question :

Q3 : *Should we start with a highly overparameterized model (dense Unet) to find a subnetwork or should we start with a smaller model (dense deep decoder) to find a subnetwork?*

Our experiments suggest that we should start with a highly overparameterized model. Starting with a smaller model, imposes the prior assumption that some architecture parts are not useful. However, this might not be always the case as we see that

Sparse-DIP often outperforms deep decoders at the same level of sparsity. In all our experiments, we fix $\lambda = 1e - 9$ and fix prior $\mathbf{p}_0 = 0.5$ for all the weights in all layers. A realization of the decoder can be obtained by fixing $\mathbf{p}_0 = 1$ for the decoder part and $\mathbf{p}_0 = 0$ for the encoder part.

Table 9. Comparison of deep decoder performance across various pruning levels.

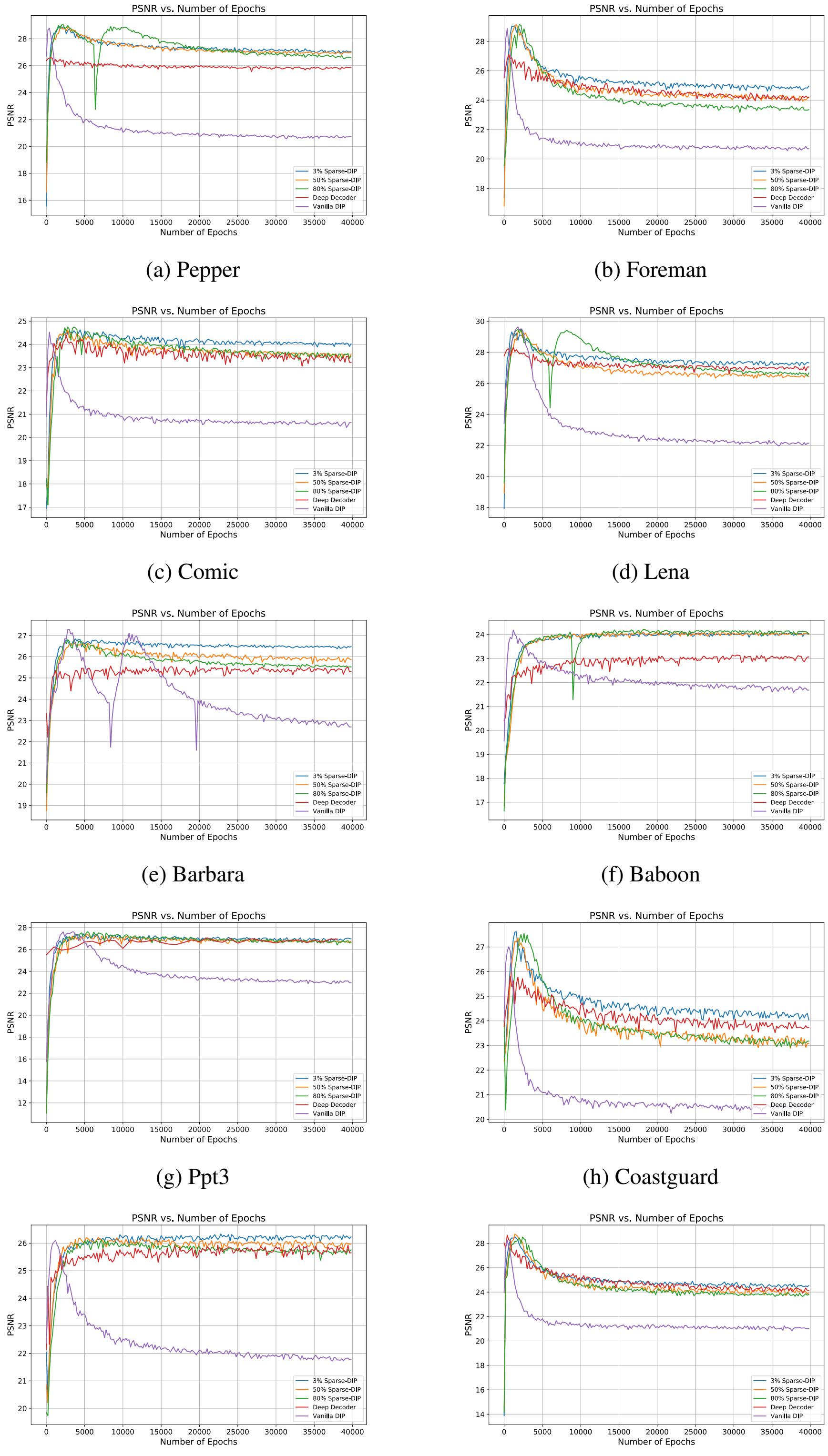
	Deep Decoder	Sparse Decoder (27%)	Sparse Decoder (55%)	Sparse Decoder (74%)
Pepper	27.01	27.06	26.17	26.35
Lena	26.80	26.94	25.35	25.15
Barbara	25.30	25.14	24.58	24.42

L. Comparison of Pruning in Image Classification and Image Reconstruction

In Table-10, we discuss the many differences in pruning networks for image classification and image reconstruction. Pruning for image classification tasks, dates back to the early 90's with a recent surge of works being done after the popularity of Lottery Ticket Hypothesis (Frankle & Carbin, 2018). To the best of our knowledge, Wu et al. (2023), is the first work to propose pruning network for image reconstruction tasks. In our work, we show the drawbacks of just applying LTH on image reconstruction tasks and propose OES that mitigates the problem. Our work also shows the Strong Lottery Ticket Hypothesis in image reconstruction networks for the first time. In Figure-19, we highlight the representation capability of OES. With no mask and or all masked, we get two extremes. In the middle ground, we can approximate any image by just masking. In Figure-20, we show the progression of transferred subnetwork through intermediate epochs, showing that the subnetwork output image is always constrained in the manifold of image priors. The images we used in this paper are shown in Figure-34 and Figure-35.

Table 10. Pruning for Image Classification vs Image Reconstruction

Criterion	**Image Classification**	**Image Reconstruction (DIP)**
Task	The pruned network is learned based on ERM loss over a set of given image/label pairs. Usually, 0-1 loss is used.	Pruned network is learned over a single image instance (extreme data-diet) and regression loss (MSE) loss is used.
Validity of LTH	LTH is essential to obtain matching subnetworks at non-trivial sparsities.	LTH is suboptimal as network overfits to image noise at convergence (post-training).
Transferability	Transferability is difficult to attain. (Mehta, 2019)	Reasonable transferability can be attained. Better transferability can be achieved through OES when compared to LTH.
Performance of matching subnetworks	Matching subnetworks can attain almost the same level of test accuracy (or slightly higher in intermediate sparsity levels (Jin et al., 2022). *Sparsity may not be necessary to get good generalization.*	Sparse subnetworks alleviate the problem of overfitting. *Sparsity is necessary to alleviate overfitting.*
Strong Lottery Ticket Hypothesis	Ramanujan et al. (2020) showed that masking a wide Resnet50 can give similar test accuracy as training a Resnet-34 on Imagenet classification. Malach et al. (2020) proved that if a ReLU fully-connected neural network with depth d and width n can fit a target by normal training, then masking a Relu network at depth $2d$ and polynomial width can approximate the same performance. For CNN's (da Cunha et al., 2021), the width required was logarithmic in depth and number of parameters.	*Our work is the first to show that Strong Lottery Ticket Hypothesis can also be observed for image reconstruction tasks.* We see that the network output can give low frequency representation of the clean image by just masking the network parameters by OES. Proving it for image reconstruction problems will be future work.



(a) Pepper

(b) Foreman

(c) Comic

(d) Lena

(e) Barbara

(f) Baboon

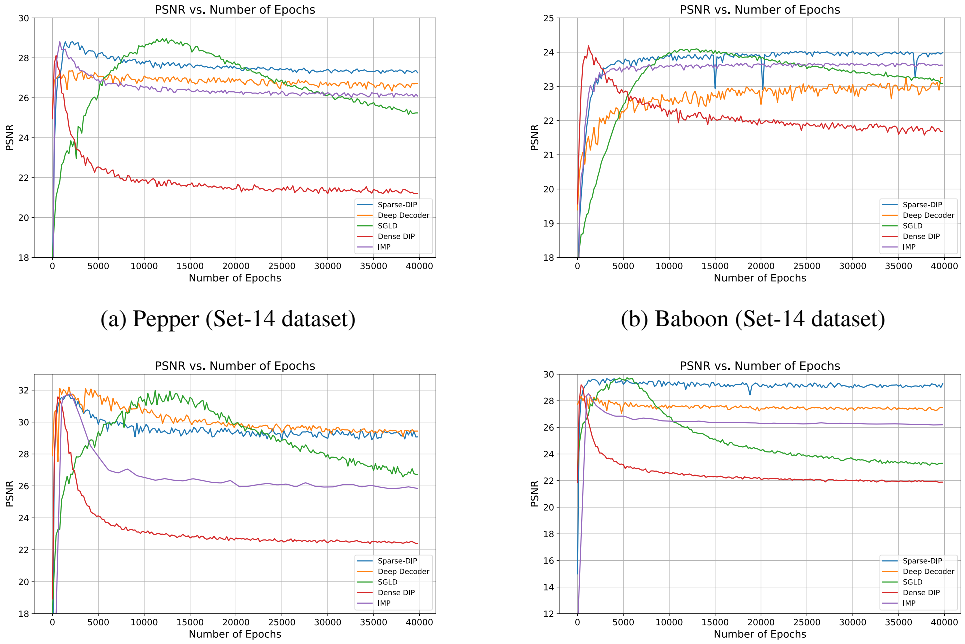
(g) Ppt3

(h) Coastguard

(i) Bridge

(j) Face

Figure 11. Denoising performances ($\sigma = 25dB$) of OES at 3 sparsity levels (3%,50%,80%) and comparison to underparamterized deep decoder and overparameterized dense DIP. We observe that the peak performance of vanilla DIP is comparable with the final convegence of sparse-DIP.



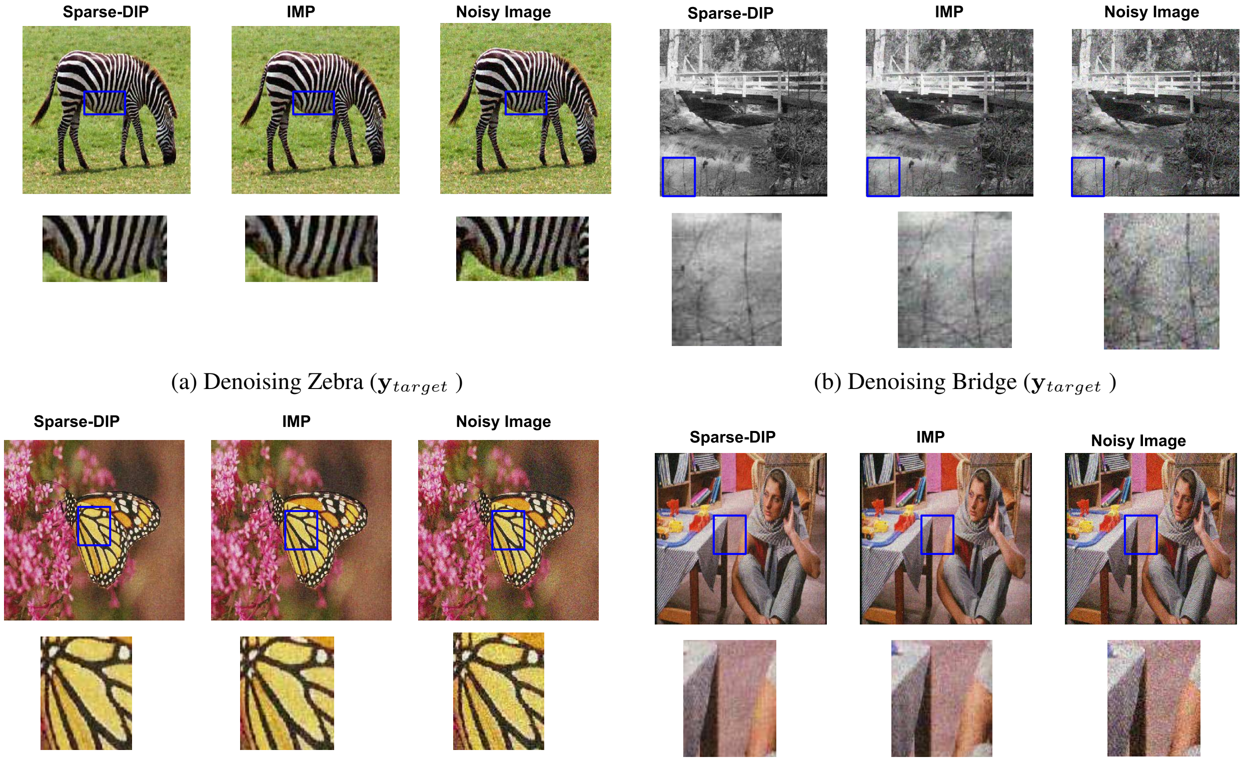
(a) Pepper (Set-14 dataset)

(b) Baboon (Set-14 dataset)

(c) Face-2 (Face dataset)

(d) Door (Standard Dataset)

Figure 12. Denoising results of various methods on noisy images ($\sigma = 25$ dB) across 3 popularly used datasets.



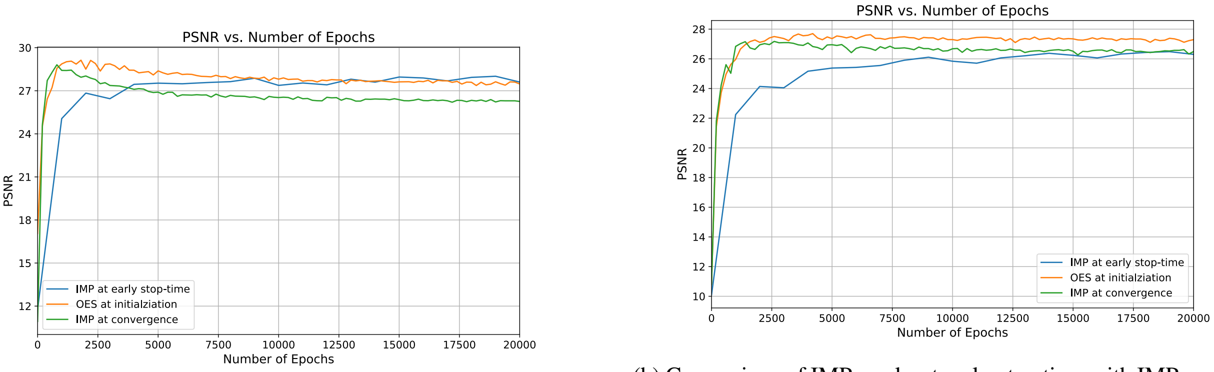
(a) Denoising Zebra ($\mathbf{y}_{target}$)

(b) Denoising Bridge ($\mathbf{y}_{target}$)

(c) Denoising Monarch ($\mathbf{y}_{target}$)

(d) Denoising Barbara ($\mathbf{y}_{target}$)

Figure 13. Comparing the denoising performance of transferred subnetworks found by OES vs subnetworks found by IMP in Set-14 dataset. Here $\mathbf{y}_{source}$ is the Lena image. Both masks are at sparsity level of 5%. IMP based subnetworks overfit to noise as shown in the zoomed version. All noisy images are corrupted with $\sigma = 25dB$. The PSNR values are found in Table-5.



(a) Comparison of IMP masks at early-stop time with IMP mask at convergence and OES at initialization on the same image (Pepper).

(b) Comparison of IMP masks at early-stop time with IMP mask at convergence and OES at initialization on different images. Mask learned with Lena, used to denoise Flowers.

Figure 14. IMP masks learned at early-stopping time performs comparatively well. But when used on transfer tasks performs worse than OES masks at initialization. All the masks are 5% sparse.

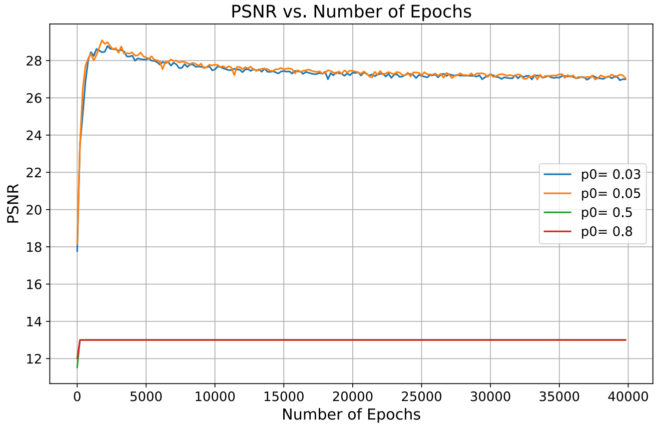


Figure 15. Performance of subnetworks trained with different prior p_0's in equation and then pruned 95% by ranking. This shows that the importance ranking of p's after training is dependent on prior p_0. Good results are expected when prior p_0 used in optimization, matches the pruning percentage.

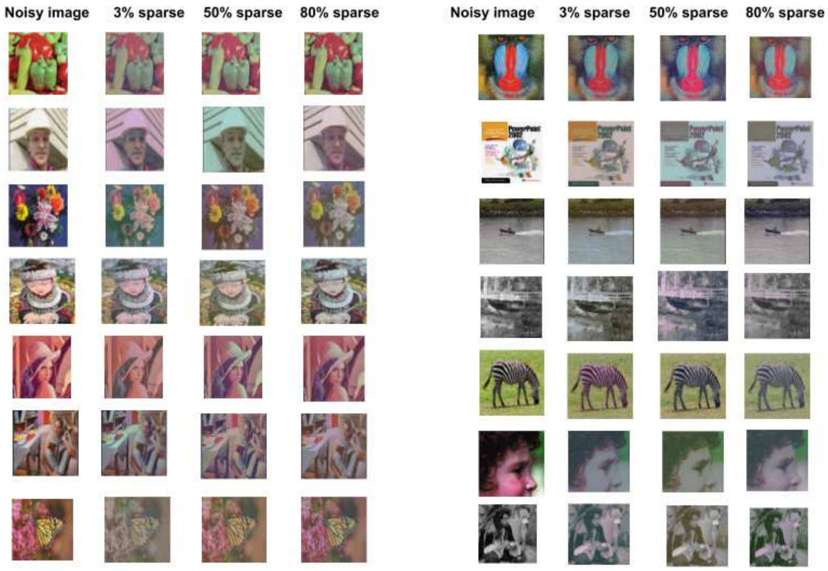
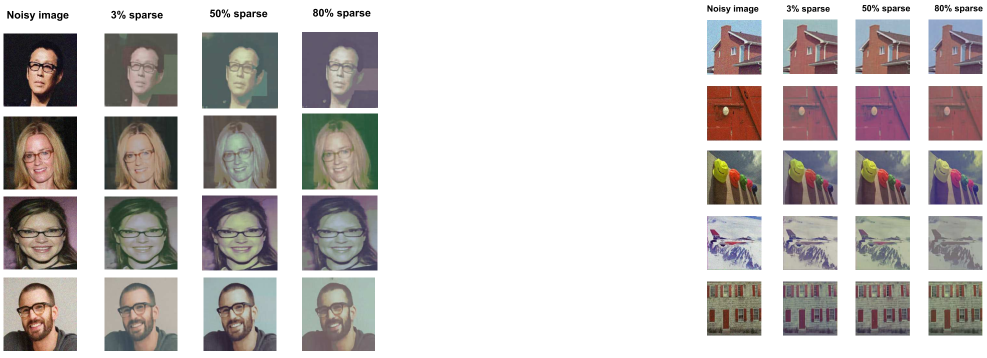


Figure 16. $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$ for Set-14 dataset. Images generated by the randomly initialized network found after applying OES mask.



(a) $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$ for Face dataset.

(b) $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$ for standard dataset.

Figure 17. Masking at initialization can induce image prior. Figures shows the images after masking image generator at initialization $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$. The mask $\mathbf{m}^*$ was learned using OES algorithm. Images corresponding to several sparsity levels are shown.

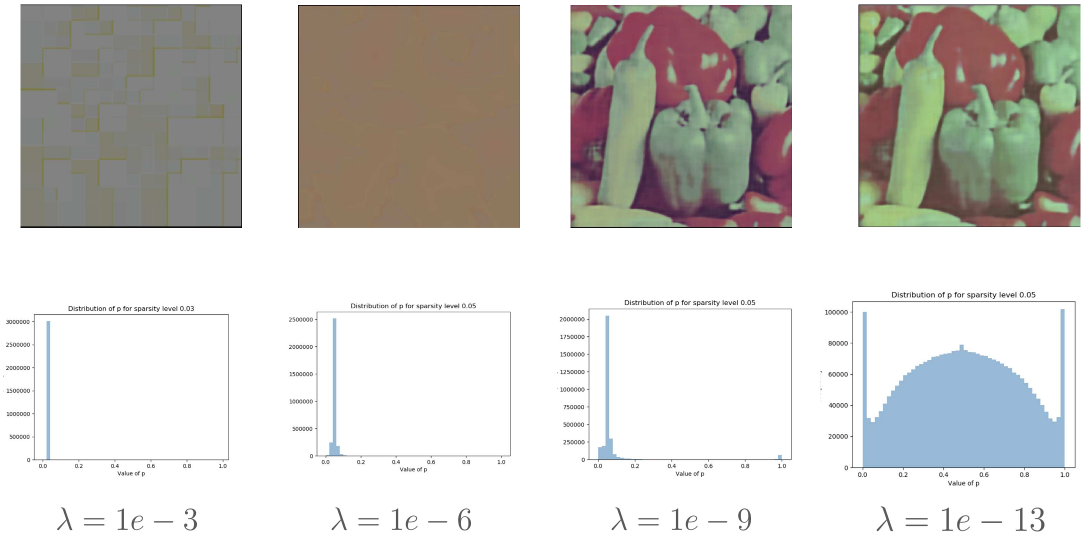


Figure 18. Distribution of logits ($\mathbf{p}$) for various λ in front of KL term and its effect of the output image ($G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$) after thresholding the logits $\mathbf{p}$ to reach the desired sparsity level. Here the prior $\mathbf{p}_0 = 0.05 \times \mathbf{1}$ and the desired threshold level is also 5% sparsity. Different strength of KL term λ leads to the distribution of logits $\mathbf{p}$ centered around the desired prior $\mathbf{p}_0$. From eq-, we observe that higher $\lambda = 1e - 3$ or $\lambda = 1e - 6$ gives more importance to the KL term and less importance to the image data-fidelity term (no image formation). $\lambda = 1e - 9$ gives the best balance of regularization and data-fidelity. For $\lambda = 1e - 9$, although the centre of distribution is at $\mathbf{p}_0$, there is some concentration near $\mathbf{p} = 1$, ensuring that there is a clear distinction between the important and non-important parameters. OES subnetwork is 5%.

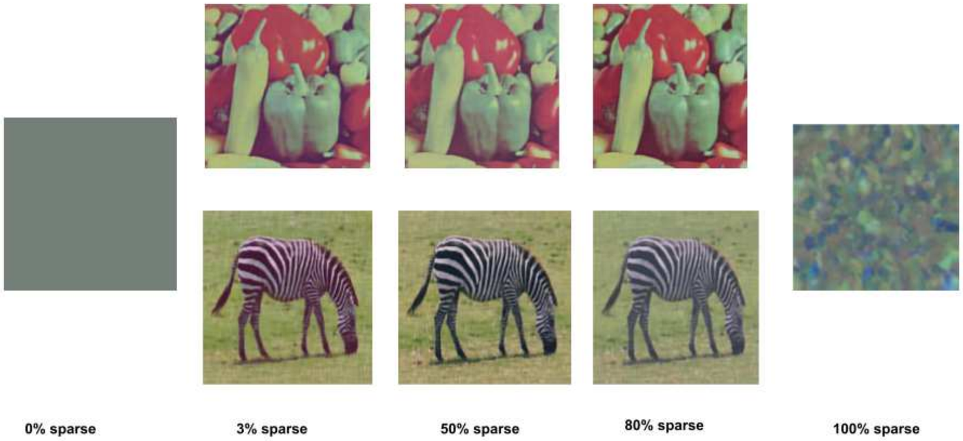


Figure 19. $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}), \mathbf{z})$: capability of image representation by just masking network parameters. When $\mathbf{m} = \mathbf{1}$, images correspond to stochastic processes producing spatial structures with self-similarity as noticed in Ulyanov et al. (2018). For $\mathbf{m} = \mathbf{0}$, it produces a constant image (assuming no bias terms). However, in the middle ground, different images (even at a fixed sparsity level) can be represented by the combination of chosing to select a weight parameter or delete it.

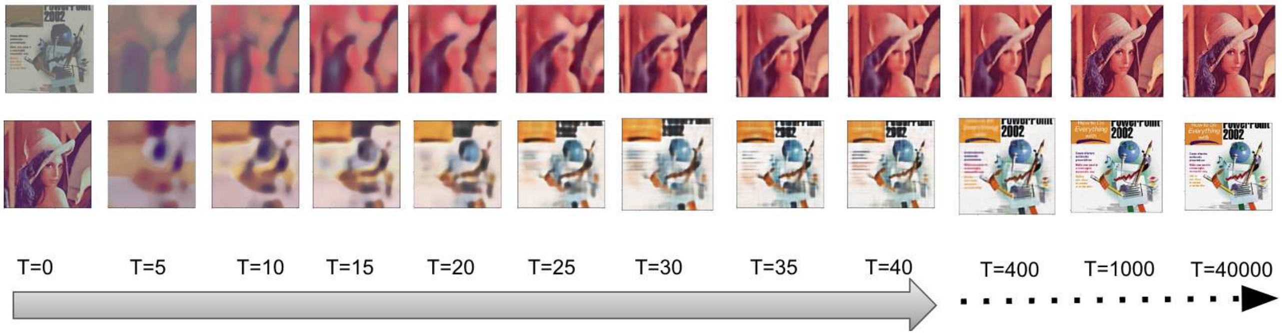


Figure 20. Transferability of OES subnetworks. OES masks trained on $\mathbf{y}_1$, denoted as $G(\boldsymbol{\theta}_{in} \circ \mathbf{m}^*(\mathbf{y}_1), \mathbf{z})$ can be used for denoising image $\mathbf{y}_2$. Here interchanging $\mathbf{y}_1$ and $\mathbf{y}_2$ in the opposite way also ensures the operation of OES. At epoch $T = 0$, just the application of mask on random network initialization (on which mask was learned), produces an image. Epoch $T = 40000$ denotes the final recovered image that does not suffer from overfitting. Underparameterization by OES subnetwork ensures that the output lies in the manifold of natural image prior.

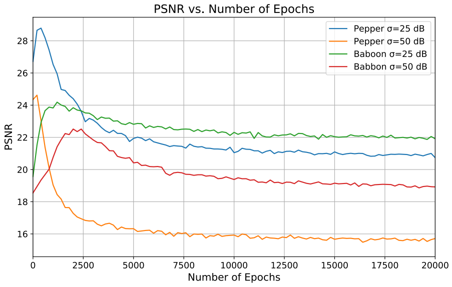


Figure 21. Early stopping time window can vary for different images and also various noise levels. Estimating this early-stopping time from an image distribution or a particular noise level can be difficult. Here we see that there can be a window as large as 2500 iterations between early stopping times of two images with different corruption levels.

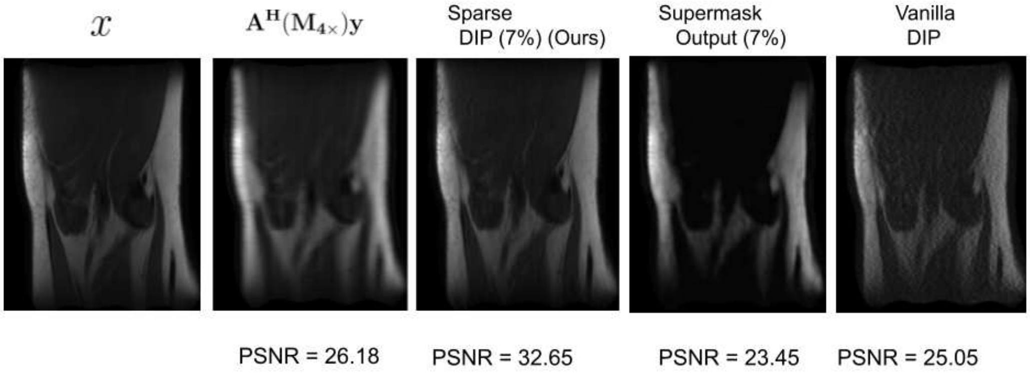


Figure 22. MRI reconstruction comparison with Sparse-DIP and Vanilla Dense DIP without early stopping. Sparse-DIP removes aliasing artifacts and preserves the important details of the images when compared to the ground-truth $\mathbf{x}$. Vanilla dense DIP overfits to artifacts (due to nullspace) and requires careful early stopping (See Figure-24a). Supermasked output at network initialization still manages to capture some important image details.

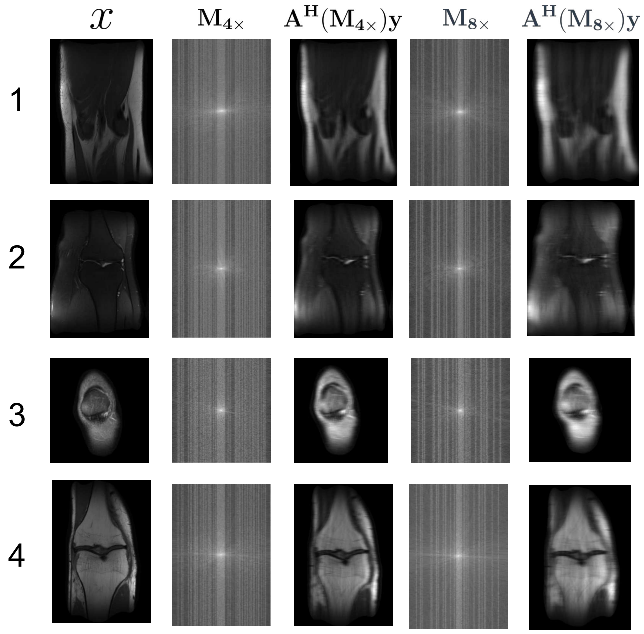
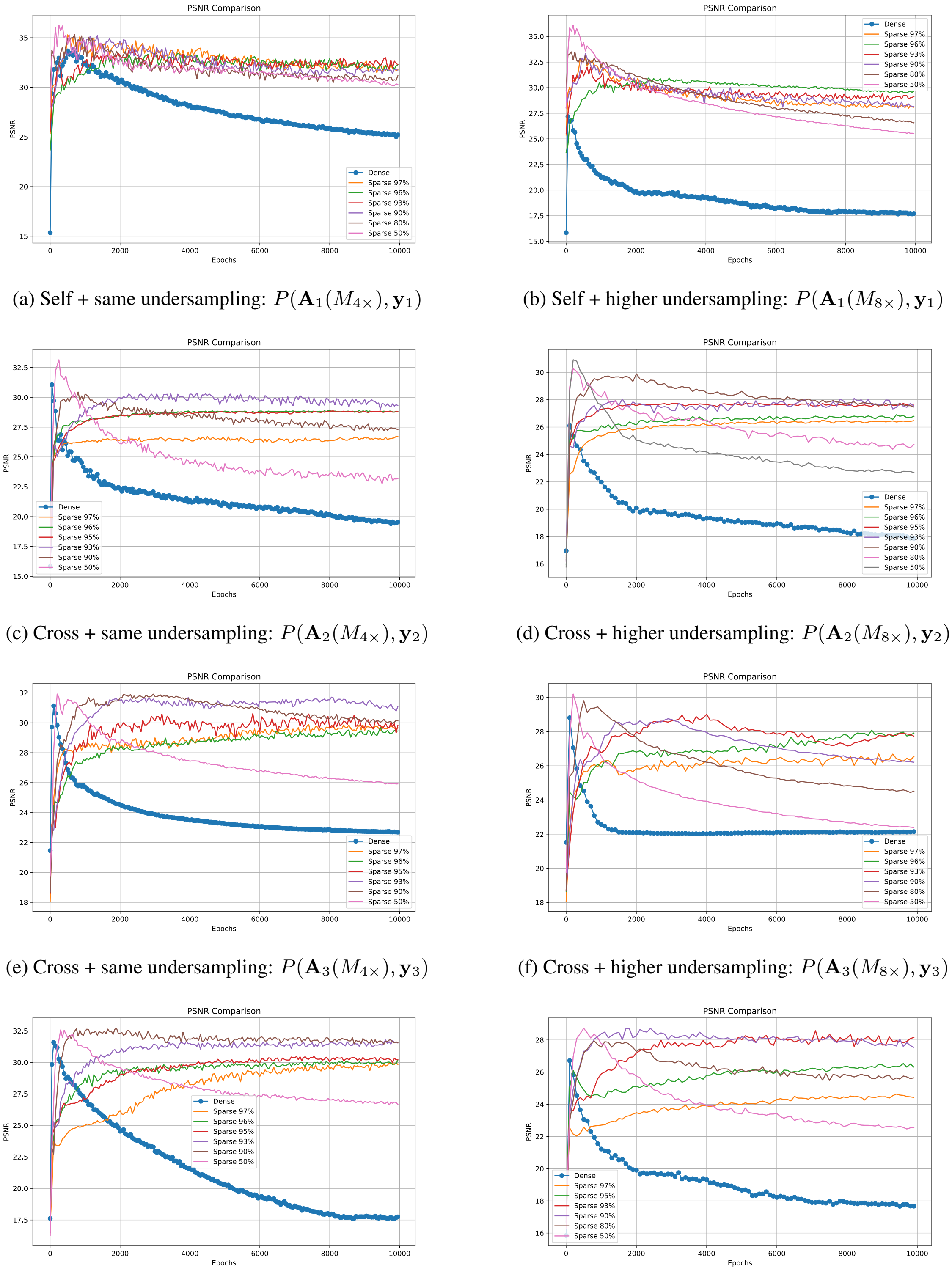


Figure 23. The 4 MRI ground-truth and measurements used in this experiment. $\mathbf{x}$ denotes the ground-truth image or full-kspace reconstruction. $\mathbf{M}_{4\times}$ and $\mathbf{M}_{8\times}$ denote the k-space undersampling masks. $\mathbf{A}^H(\mathbf{M}_{4\times})\mathbf{y}$ and $\mathbf{A}^H(\mathbf{M}_{8\times})\mathbf{y}$ denote the zero-filling reconstructions that produce aliasing artifacts.



(a) Self + same undersampling: $P(\mathbf{A}_1(M_{4\times}), \mathbf{y}_1)$

(b) Self + higher undersampling: $P(\mathbf{A}_1(M_{8\times}), \mathbf{y}_1)$

(c) Cross + same undersampling: $P(\mathbf{A}_2(M_{4\times}), \mathbf{y}_2)$

(d) Cross + higher undersampling: $P(\mathbf{A}_2(M_{8\times}), \mathbf{y}_2)$

(e) Cross + same undersampling: $P(\mathbf{A}_3(M_{4\times}), \mathbf{y}_3)$

(f) Cross + higher undersampling: $P(\mathbf{A}_3(M_{8\times}), \mathbf{y}_3)$

(g) Cross + same undersampling: $P(\mathbf{A}_4(M_{4\times}), \mathbf{y}_4)$

(h) Cross + higher undersampling: $P(\mathbf{A}_4(M_{8\times}), \mathbf{y}_4)$

Figure 24. Performance of OES subnetworks for MRI reconstruction from 4× (left column) and 8× (right column) undersampled k-space measurements. In all the experiments, the OES network mask m^* was learned from pair $(\mathbf{A}_1(M_{4\times}), \mathbf{y}_1)$. In Figure-a (self+ same undersampling), the subnetwork mask was used to reconstruct image from $(\mathbf{A}_1(M_{4\times}), \mathbf{y}_1)$. In Figure-b (self+ higher undersampling), mask was used to reconstruct from $(\mathbf{A}_1(M_{8\times}), \mathbf{y}_1)$ which has a higher undersampling. For Figures (c-h) (cross), the operator-measurement pair for image reconstruction were different from which the mask was learned $(\mathbf{A}_1(M_{4\times}), \mathbf{y}_1)$.

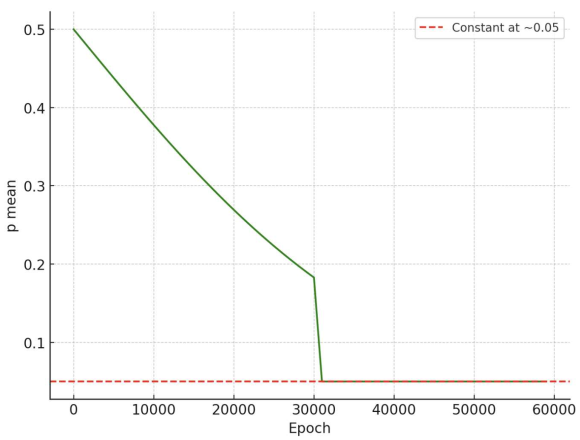


Figure 25. Mean of p across various epochs when the regularization used is $|mean(p) - \frac{s}{d}|$. In this particular experiment, $\frac{s}{d} = 0.05$.

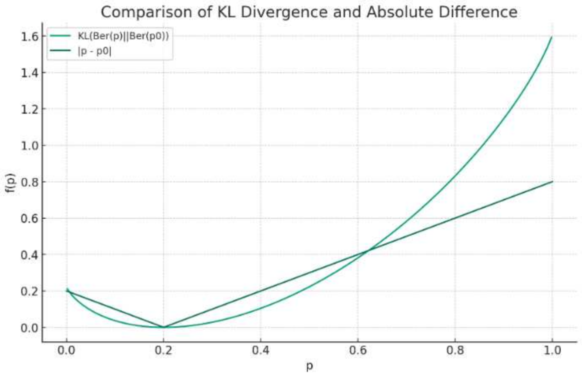
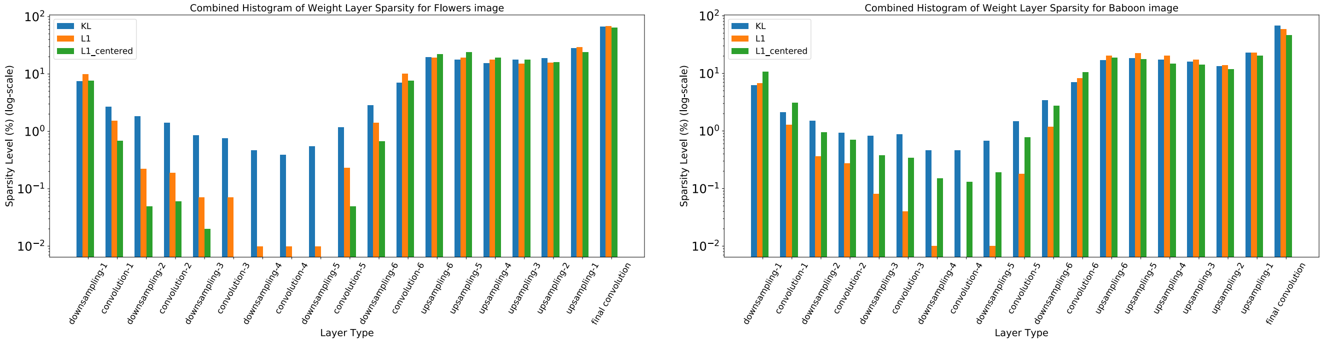


Figure 26. Comparison of KL regularization and pointwise centered ℓ_1 regularization for a scalar value. Around the prior value p_0, the KL is much smoother than ℓ_1 regularizer. $f'(p) = \log\left(\frac{\frac{p}{1-p}}{\frac{p_0}{1-p_0}}\right)$ for KL, which is very close to 0 when $p \in [p_0 - \epsilon, p_0 + \epsilon]$. However, for ℓ_1 regularization, $f'(p) = 1$ or -1 for all points except $p = p_0$.



(a) Flowers

(b) Baboon

Figure 27. Layerwise architecure pruning (sparsity percentage in log-scale) by OES at initialziaiton using three different choices of regularization, KL, ℓ_1 and centered ℓ_1 for Baboon image and Flowers image in Set-14 dataset. Centered ℓ_1 means the centered mean regularizer.

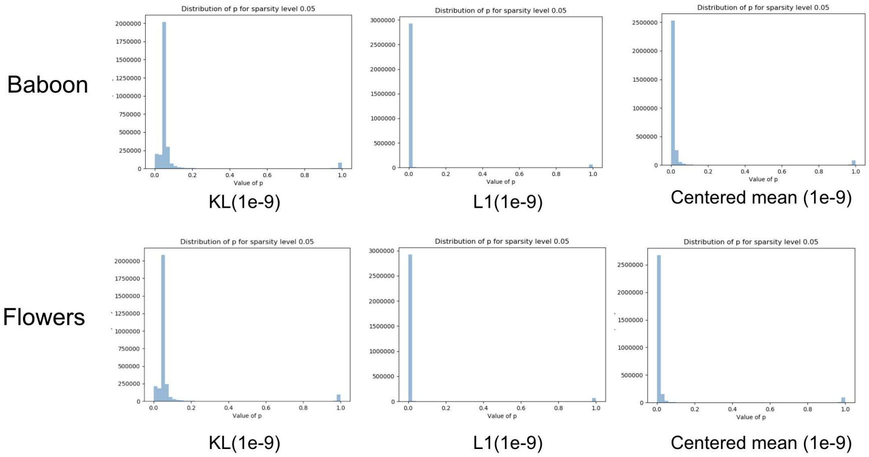


Figure 28. Histogram of logits of p when OES is ran across images with KL, ℓ_1 and centered mean regularizer. In our implementation we minimize $|\sum_i p_i - (\frac{s}{d} * numel(p))|$, to both ℓ_1 regularization and centered mean regularizer on the same scale.

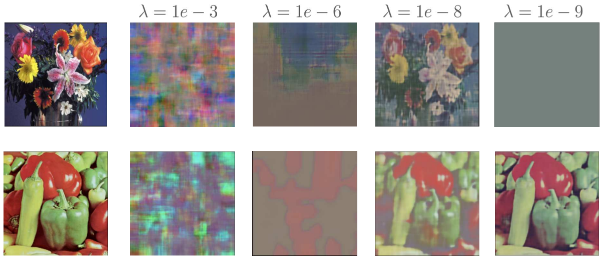


Figure 29. Sensitivity of hyperparameter λ when mask is optimized by L1 regularization. Here, the best mask is obtained for different λ for different images. For example, for the flower image, $\lambda = 1e - 8$, is the best hyperparameter, but for the Pepper image $\lambda = 1e - 9$. The masks here are 5%-sparse.

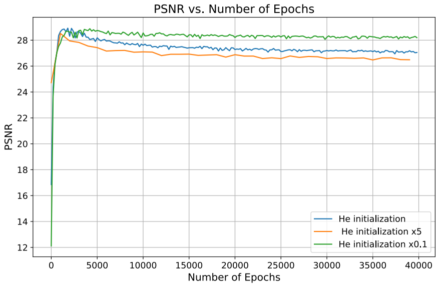
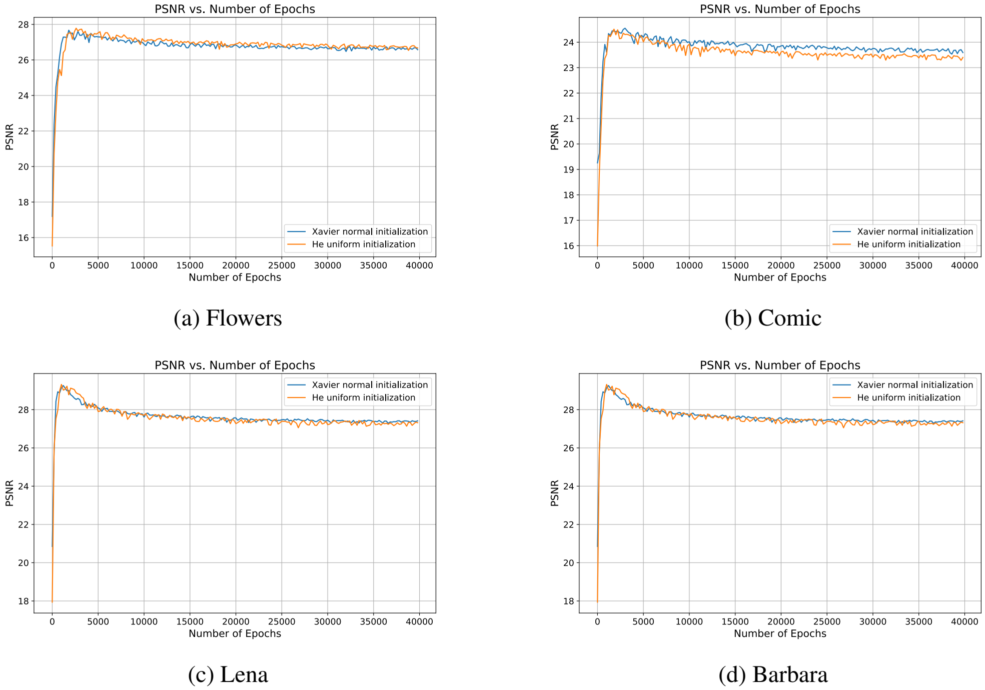


Figure 30. Comparison of denoising performance of OES masks at different initialization scales.



(a) Flowers

(b) Comic

(c) Lena

(d) Barbara

Figure 31. Denoising performance of OES masks learned at He (uniform) initialization vs at Xavier initialization (Gaussian initialization). The initialization distriubtion does not seem to play a big role in learning the mask.

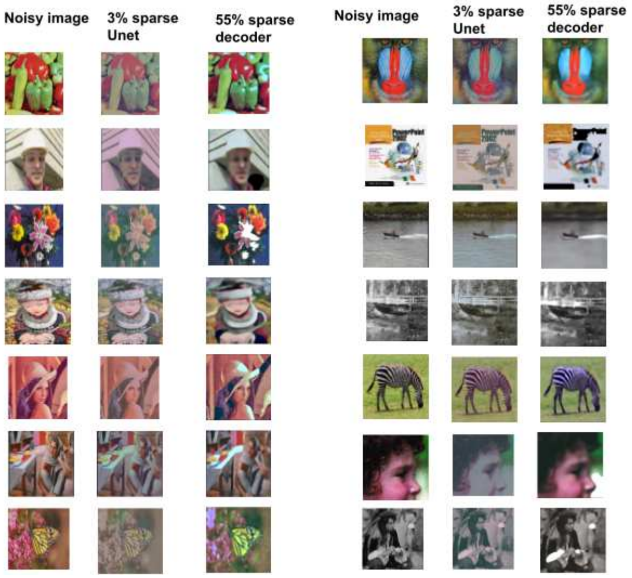


Figure 32. Comparison of OES masking in deep Unet vs in deep decoder.

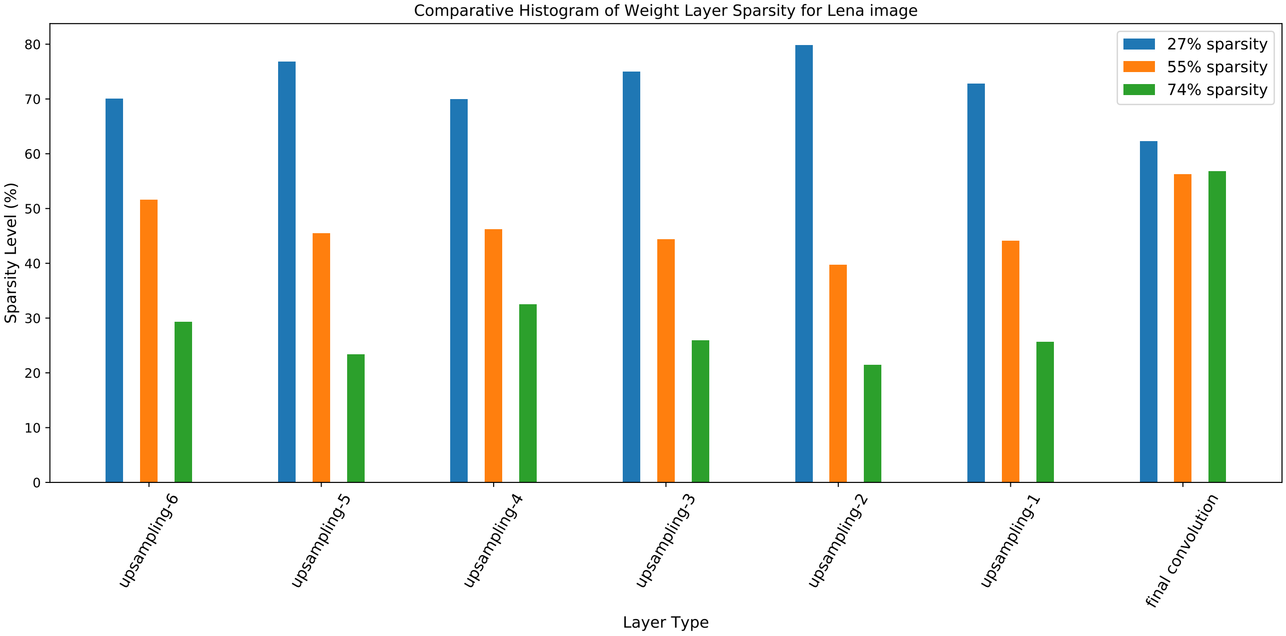


Figure 33. Layerwise pruning percentage for a deep decoder at various level sparsity levels.

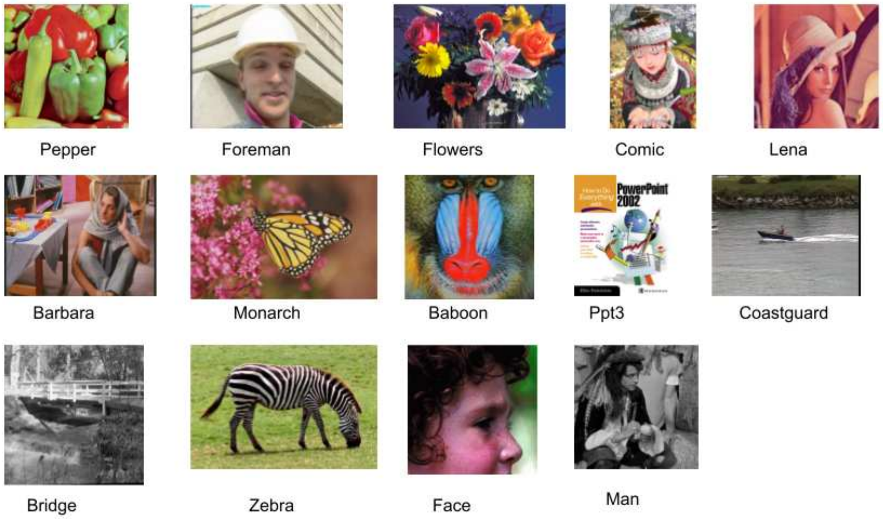


Figure 34. Set-14 dataset images used in this paper.

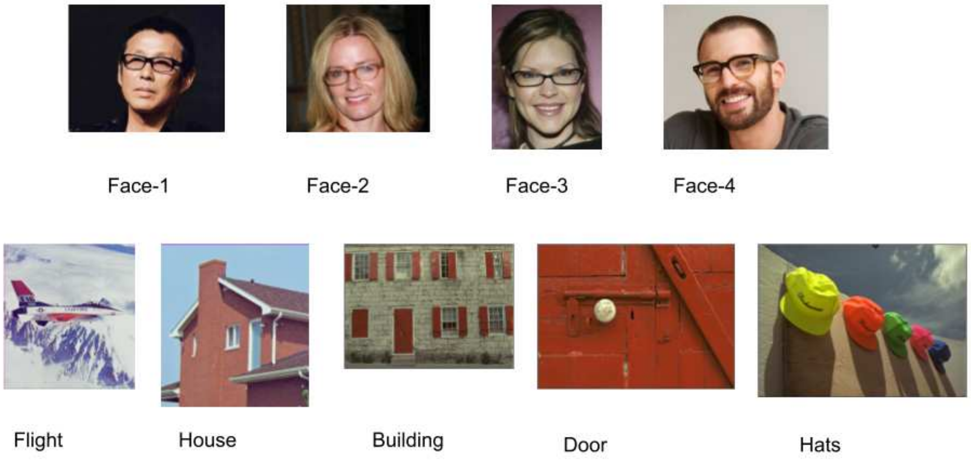


Figure 35. Images in face and standard dataset used in this paper.