# Generator Surgery for Compressed Sensing

**Jung Yeon Park**[1*] **Niklas Smedemark-Margulies**[1*] **Max Daniels**[1] **Rose Yu**[2]

**Jan-Willem van de Meent**[1] **Paul Hand**[1]

[1]Northeastern University
{park.jungy,smedemark-margulie.n,daniels.g,
j.vandemeent,p.hand}@northeastern.edu

[2]University of California San Diego
roseyu@ucsd.edu

## Abstract

Recent work has explored the use of generator networks with low latent dimension as signal priors for image recovery in compressed sensing. However, the recovery performance of such models is limited by high representation error. We introduce a method to reduce the representation error of such generator signal priors by cutting one or more initial blocks at test time and optimizing over the resulting higher-dimensional latent space. Experiments demonstrate significantly improved recovery for a variety of architectures. This approach also works well for out-of-training-distribution images and is competitive with other state-of-the-art methods. Our experiments show that test-time architectural modifications can greatly improve the recovery quality of generator signal priors for compressed sensing.

## 1 Introduction

In inverse imaging problems, we recover an image signal from undersampled measurements. As the problem is underdetermined, additional structural assumptions are required. Classical methods leverage sparsity or compressibilty in a known basis [1, 2]. Recent developments in deep generative modeling, such as generative adversarial networks (GANs [3]) and variational autoencoders (VAEs, [4]) have led to their use as a signal prior for these problems. While Bora et al. [5] showed that generator signal priors outperform sparsity priors at low undersampling ratios, recovery quality is limited by low latent dimension and high *representation error* (formal definitions below). We focus on improving these generator-based signal priors.

To reduce representation error, it is natural to seek models with high latent dimension, but existing optimizers cannot reliably train such models. This has motivated a variety of approaches for obtaining generator signal priors with more degrees of freedom [6–11], though these methods are often computationally expensive or complicated. For example, Gu et al. [6] optimize multiple latent codes, composing them at an intermediate layer with optimizable weights. Abu Hussein et al. [7] optimize both the latent code *and* the generator's weights. Untrained methods such as Deep Image Prior [11] and Deep Decoder [10] recover an image by optimizing the weights of a randomly-initialized network.

We address the problem of high representation error in generator signal priors and substantially improve their performance in compressed sensing with a simple and inexpensive technique we call

[*]Equal contribution

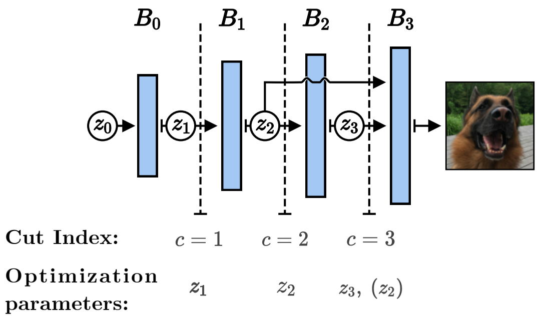

Figure 1: Applying *Generator Surgery*.

**Algorithm 1:** Compressed Sensing using *Generator Surgery*

```
Input: Pre-trained $\mathcal{G}_0$, $c$, $A$, $y$, $T$, $\alpha$
$\mathcal{G}_c \leftarrow cut(\mathcal{G}_0, c)$ ; // cut c blocks
Initialize $z^{(0)}$ ;
for $t = 0$ to $T - 1$ do
    $x^{(t)} \leftarrow \mathcal{G}_c(z^{(t)})$ ;
    $L \leftarrow \|y - Ax^{(t)}\|$ ;
    $z^{(t+1)} \leftarrow z^{(t)} - \alpha \nabla_{z^{(t)}} L$ ;
end
Output: $\hat{x} = x^{(T)}$
```

*Generator Surgery*. Using a pre-trained generator with low latent dimension, we 'cut' one or more early blocks from the network at test time, and treat the intermediate activations as a new latent image representation. By cutting early blocks at test time, we trade the generative sampling capability of the model for increased recovery quality.

Our method is applicable for a wide variety of network architectures, and is competitive with other recent baselines. We also show that our method can even recover high-quality images that are out-of-training-distribution; this is important as real-world applications like MRI imaging often include target images with novel features. Our results suggest more broadly that test-time architectural modifications can allow a model trained on one task (image generation) to perform well on a different task (image recovery).

## 2 Method

### 2.1 Generator Surgery (GS)

We define a generator network $\mathcal{G}_0 : \mathbb{R}^{k_0} \mapsto \mathbb{R}^n$ that maps a low-dimensional latent code $z_0 \in \mathbb{R}^{k_0}$ to an image $x = \mathcal{G}_0(z_0)$. The generator is composed of $d$ blocks:

$$\mathcal{G}_0(z_0) = B_{d-1} \circ \ldots \circ B_0(z_0), \tag{1}$$

where we define $z_i \in \mathbb{R}^{k_i}$ as the input to block $B_i$. Each block $B_i$ can consist of arbitrary neural network operations (such as convolution, upsampling, and activation functions). We assume $k_i \geq k_0, \forall i = 1, \ldots d-1$, which holds for all architectures considered in this paper. We refer to Eqn. (1) as an 'uncut' generator or 'No Generator Surgery (no GS)'.

After training the model in Eqn. 1 or using pre-trained weights, we cut away the first $c \geq 1$ blocks, resulting in a 'cut generator' $\mathcal{G}_c : \mathbb{R}^{k_c} \mapsto \mathbb{R}^n = B_{d-1} \circ \ldots \circ B_c$ which accepts a latent code $z_c \in \mathbb{R}^{k_c}$ as input. We refer to this model as 'Generator Surgery (GS)', and we apply it on inverse problems.

It is important to note that the cut generator is not a generative model. In effect, we are intentionally trading our ability to approximately sample from the learned distribution over images for increased expressivity of the latent representation in inversion tasks. Furthermore, since we remove blocks relative to the original generator, the cut generator requires fewer backpropagation steps and thus less computation at inversion time.

**Additional inputs** Our method is applicable to many different architectures, including models with skip connections or class-conditional models. Specifically, if we cut the first $c$ blocks of a model and the new initial layer $B_c$ requires multiple inputs, all of these inputs are treated as optimizable parameters at inversion time. We use several architectures in experiments, one of which includes skip connections (BEGAN [12]).

### 2.2 Compressed Sensing

We consider the problem of noisy compressed sensing (CS). All norms $\|\cdot\|$ are $L_2$ unless otherwise stated. For an unknown target image $x \in \mathbb{R}^n$, we are given a measurement matrix $A \in \mathbb{R}^{m \times n}, m \ll n$

and measurements $y = Ax + \eta$, where $\eta$ is noise. Our goal is to find $\hat{x} \in \mathbb{R}^n$ that minimizes $\|x - \hat{x}\|$. Note that the problem is underdetermined, since $m < n$. Using a generator $G : \mathbb{R}^k \mapsto \mathbb{R}^n$, we can estimate $\hat{x}$ by finding a latent code $z$ that optimizes the loss function:

$$\min_z \|y - AG(z)\| \tag{2}$$

We study the case of a Gaussian measurement matrix $A_{ij} \sim \mathcal{N}(0, 1/m)$ and i.i.d. Gaussian noise $\eta \sim \mathcal{N}(0, \sigma^2)$. We follow Bora et al. [5] and set $\sigma = 0.1$ for 64px images, and scale the noise level for larger images such that $\mathbb{E}\left[\|\eta\|^2/\|Ax\|^2\right]$ is kept constant. Applying 'Generator Surgery' to compressed sensing and using gradient descent to optimize Eqn. (2) results in Alg. 1.

### 2.3 Theoretical Motivation

Define the *representation error* of $\mathcal{G}_0$ w.r.t. an image $x$ as $\text{Err}_{rep}(\mathcal{G}_0, x) = \min_{z \in \mathbb{R}^{k_0}} \|x - \mathcal{G}_0(z)\|$, and analogously for $\mathcal{G}_c$. Bora et al. [5] show in their Theorem 1.2 that for an $L$-Lipschitz network, overall error in image recovery using a neural network signal prior is bounded w.h.p. by four terms: representation error, the norm of the measurement noise $\eta$, optimization error $\epsilon$, and an additive error $\delta$ based on the sensing matrix.

Consider the performance of the uncut generator $\mathcal{G}_0$ and cut generator $\mathcal{G}_c$ applied to the same set of measurements $y$. Since we use the same measurements $y$, both the sensing error $\delta$ and the measurement noise $\eta$ are equal for the two generators. We can assume that both models are optimized to below a fixed error level $\epsilon$ (we empirically find that optimization error is indeed low). Thus, in order to compare overall recovery quality between the two models, we only need to compare their representation error:

$$\text{Err}_{rep}(\mathcal{G}_c, x) = \min_{z \in \mathbb{R}^{k_c}} \|x - \mathcal{G}_c(z)\| \le \min_{\tilde{z} \in \mathbb{R}^{k_0}} \|x - \mathcal{G}_c(B_{c-1} \circ \dots B_0(\tilde{z}))\| = \text{Err}_{rep}(\mathcal{G}_0, x) \tag{3}$$

The inequality in Eqn. (3) arises by comparing the minimization of $z \in \mathbb{R}^{k_c}$ to the constrained minimization of $\tilde{z} \in \mathbb{R}^{k_0}$ for $k_c \ge k_0$. Intuitively, we can see that cutting blocks increases the range of the model and thereby reduces representation error.

## 3 Experiments

**Experimental Design.** We apply *Generator Surgery* (GS) to DCGAN [13], BEGAN [12], and VAE [4] architectures. We measure recovery performance using average peak signal-to-noise ratio (PSNR). All models are trained on CelebA [14]. We measure average PSNR on recovery of test images from CelebA and COCO 2017 [15]. For each architecture, we choose the cut index $c$ that maximizes average PSNR over 100 validation images on CelebA. For all experiments, we use 100 images per dataset, and run Alg. 1 with 3 random restarts.

We first compare against the standard sparsity-based Lasso-DCT to demonstrate utility [16]. Lasso-DCT solves the Lasso optimization $\hat{z} = \min_z \|y - A\Phi z\|_2^2 + 0.01\|z\|_1$ . Next, we compare against several recent learned and unlearned methods: IAGAN [7], Deep Decoder (DD) [10], and mGANprior [6]. To compare GS against baselines, we use the same generator architecture for *Generator Surgery*, IAGAN, and mGANprior. For Deep Decoder (DD), we use a similar number of parameters as *Generator Surgery* for a fair comparison. We follow hyperparameters given in the literature, except for mGANprior as the given settings were not applicable.

**Generator Surgery for CS.** Fig. 2 shows recovery on images from CelebA test set as well as COCO. For CelebA, the "No GS" model matches the results of Bora et al. [5], outperforming Lasso-DCT at low measurement regimes for all models. The "GS" model improves recovery performance substantially for all architectures, and outperforms the Lasso-DCT baseline over almost all undersampling ratios.

For COCO images, cutting blocks still gives a large jump in recovery quality. The performance increase over "No GS" is similar to CelebA images; this may indicate that much of the bias towards the training domain is contained in the early blocks of the generator.

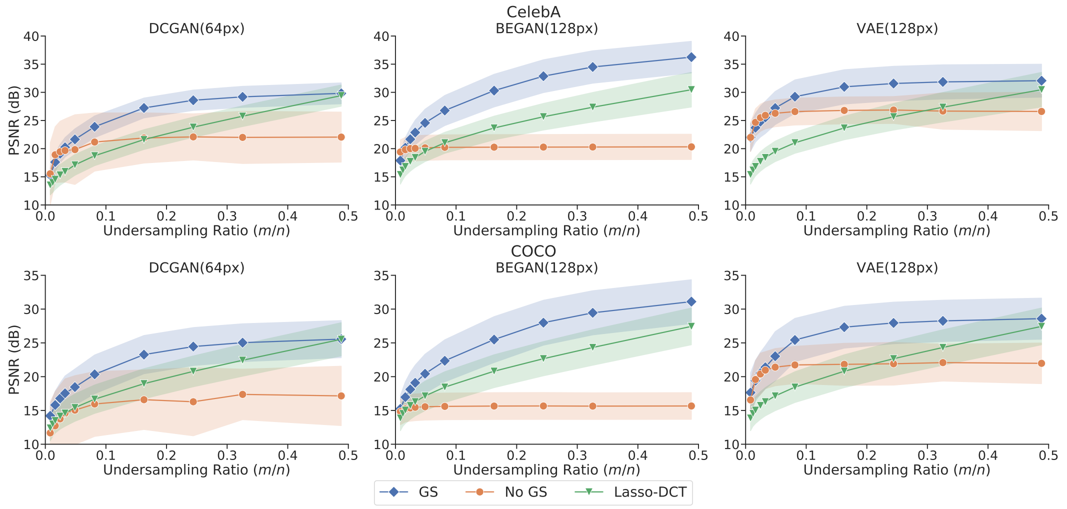

Figure 2: Effect of *Generator Surgery* for DCGAN, BEGAN, and VAE on CS for CelebA (top) and COCO (bottom). Shaded bands show 1 standard deviation. All models were trained on CelebA.

**Baseline Comparisons.** Fig. 3 shows the comparison to baselines for BEGAN. We emphasize that our focus is not to produce the highest performance but to be competitive with other methods while being simpler. GS generally beats mGANprior and performs similarly to DD (GS is slightly worse at lower measurements and slightly better at higher measurements). IAGAN outperforms all methods by a fair margin; this is expected as IAGAN is highly overparametrized. In general, performance increases with increasing overparametrization ratio (see Table 1). Qualitative comparisons are provided in the Appendix.

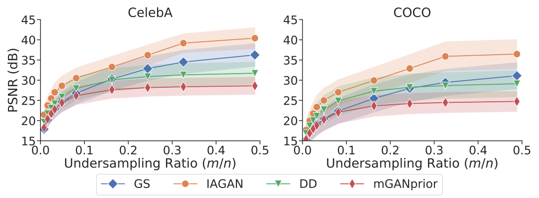

Figure 3: CS on CelebA and COCO. GS using BEGAN model outperforms mGANprior and performs similarly to Deep Decoder.

Table 1: Overparameterization ratio

| | BEGAN |
|---|---|
| mGANprior | 0.195 |
| **GS** | 0.833 |
| DD | 0.838 |
| IAGAN | 43.7 |

## 3.1 Understanding GS Effect

**Optimization and Representation Error.** We explore the relative magnitude of sources of error in our models. We set $A = I$ and $\eta = 0$ to eliminate measurement error and sensing error, leaving only representation error and optimization error. We then evaluate recovery performance for cut and uncut models on images generated from the model (for which representation error is zero by definition), as well as CelebA train, test, and COCO images (Fig. 4, left). The uncut generator $\mathcal{G}_0$ achieves a very high PSNR on generated images, confirming that it has low optimization error, but performs poorly on all real images, indicating it suffers from high representation error. By contrast, the cut generator $\mathcal{G}_c$ achieves high quality recovery on generated images, indicating low optimization error, and also performs well on real images, indicating reduced representation error.

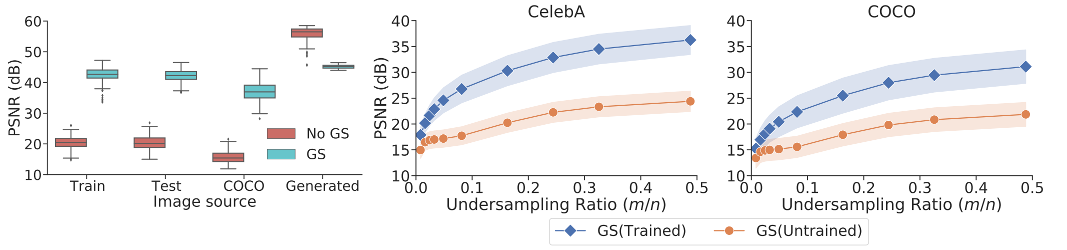

Figure 4: Left: Average PSNR for BEGAN model. Right: Trained vs. untrained GS for BEGAN model on CS task.

**Do Learned Weights Matter?** We examine whether the performance increase from GS is solely due to increased degrees of freedom, or whether this performance benefits from the learned weights. Fig. 4 (right) shows recovery performance using a trained and untrained BEGAN generator in Alg.1. The pre-trained model weights produce significantly higher recovery quality in both image domains.

**Other considerations** All results for this subsection are provided in the Appendix. We confirm that directly training a cut generator fails for a variety of hyperparameter settings, indicating that cutting blocks must be done at test time. We also show that there is a tradeoff in choosing how many blocks to cut; while cutting initial blocks increases the generator's expressivity and reduces bias towards the training domain, cutting too many blocks results in poor image quality. We also consider different initialization strategies and how they impact performance. Finally, we show samples from the cut generator, demonstrating it no longer behaves as a generative model.

## References

[1] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[2] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

[5] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 537–546. JMLR. org, 2017.

[6] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *CVPR*, 2020.

[7] Shady Abu Hussein, Tom Tirer, and Raja Giryes. Image-Adaptive GAN based Reconstruction. *AAAI Conference on Artificial Intelligence*, 2020.

[8] Muhammad Asim, Max Daniels, Oscar Leong, Paul Hand, and Ali Ahmed. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *Proceedings of Machine Learning and Systems 2020*, pages 4577–4587, 2020.

[9] ShahRukh Athar, Evgeny Burnaev, and Victor Lempitsky. Latent convolutional models. In *International Conference on Learning Representations (ICLR)*, 2019.

[10] Reinhard Heckel and Paul Hand. Deep decoder: Concise image representations from untrained non-convolutional networks. *International Conference on Learning Representations*, 2019.

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[12] David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[13] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations (ICLR)*, 2016. URL `http://arxiv.org/abs/1511.06434`.

[14] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[16] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

## A Appendix

### A.1 Baseline Comparisons

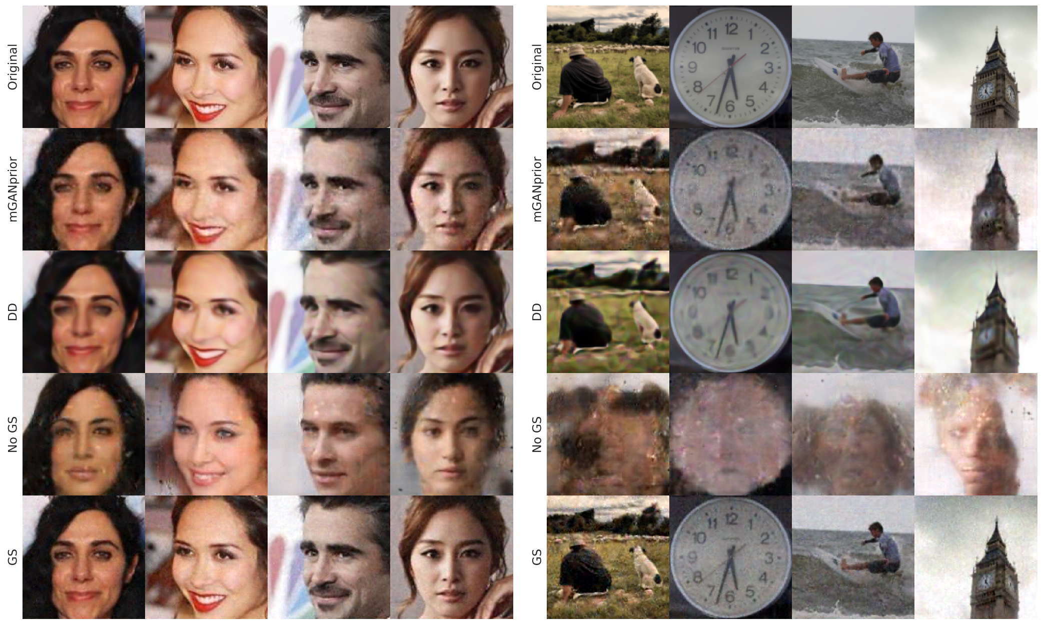

Figure 5: Compressed sensing of CelebA (left) and COCO (right) images for BEGAN(128px) at $m/n \approx 0.16$.

Fig. 5 show recovered images from CelebA and COCO for compressed sensing at $m/n \approx 0.16$. We use BEGAN for GS. First, we can clearly see that "no GS" recovers a clearly different face than the original for CelebA and hallucinates a non-existent face for COCO (this is expected as the models are trained on CelebA). This qualitatively demonstrates the large representation error of the uncut generator. mGANprior produces slightly lower quality results than GS; it is grainier and sometimes creates different colors than the original. DD and GS show subtle qualitative differences. DD often produces smoother images, sometimes losing fine textural details such as eyes, hair strands, and clock numerals; GS produces grainier images but recovers such fine details.

### A.2 Choice of Initialization

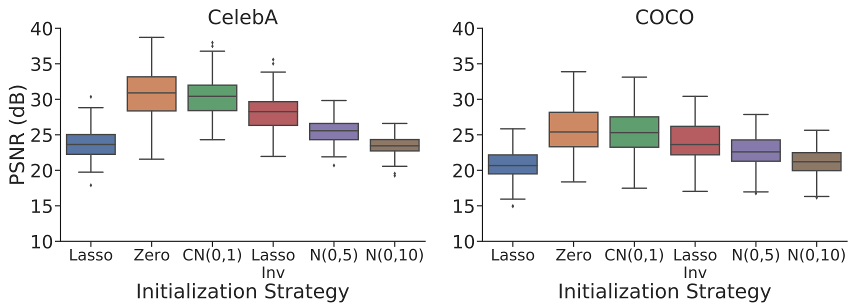

Figure 6: Initialization strategies for GS using BEGAN model on CS task. We use $m/n \approx 0.16$ over 100 images from CelebA and COCO. See text for label explanations.

We consider different initialization strategies in Fig. 6. "Zero" is the all-zeros vector and "$CN(0, 1)$" denotes a standard normal distribution censored to the range $[-1, 1]$. "LassoInv" denotes using the solution to the Lasso-DCT optimization (the latent vector $z$) as initialization. $\mathcal{N}(0, 5)$, $\mathcal{N}(0, 10)$ are normal distributions with increasing variance. All initializations were run with the best of 3 random restarts. We find that performance varies widely for different initialization strategies, supporting our claim of the source of regularization. We find that $CN(0, 1)$ and "Zero" perform best, though "Zero" has slightly higher variance. We use $CN(0, 1)$ for all experiments.

### A.3 Choosing Number of Blocks to Cut

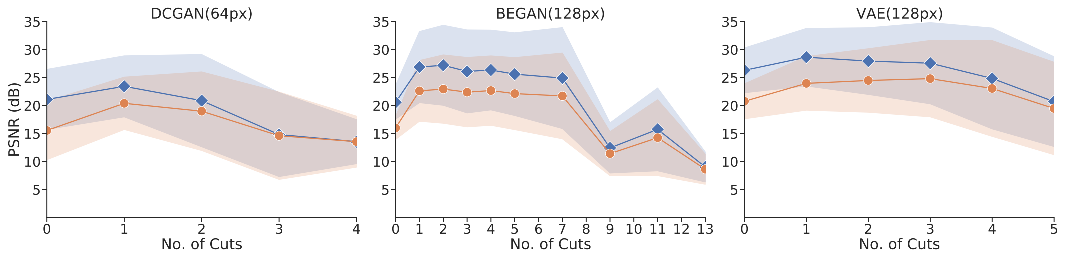

Figure 7: Varying cut index $c$ of GS using DCGAN, BEGAN, VAE models on CS task. For each model, we use 100 validation images and select $c$ that maximizes PSNR on CelebA.

We find the best number of cuts $c$ for each model by evaluating compressed sensing performance on validation images from CelebA and COCO. We use 100 images and compute the average PSNR for each dataset. Fig. 7 shows the average PSNR for these datasets when applying *Generator Surgery* to DCGAN, BEGAN, and VAE models. For all other experiments, we select the $c$ which achieves the highest PSNR on CelebA here, which is $c = 1$ for DCGAN, $c = 2$ for BEGAN, and $c = 1$ for VAE.

### A.4 Generator samples with and without *Generator Surgery*

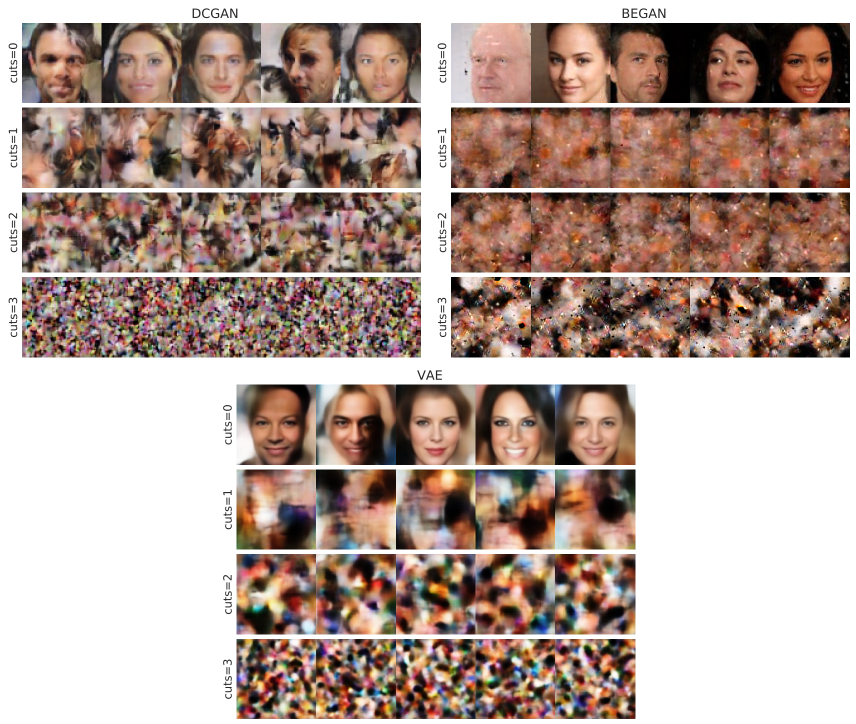

Figure 8: Generated samples from pre-trained DCGAN, BEGAN, and VAE with varying cut index.

We give samples from our pre-trained generators, before *Generator Surgery* ("cuts= 0"), and after cutting 1, 2, or 3 blocks from each model. The high quality samples in the first rows show that our original generators have been successfully trained, while the low quality samples from the cut generators show that these no longer behave as a generative model after *Generator Surgery*.

We produce samples from cut generators by sampling from a censored normal $p(z_c) = CN(0,1)^{k_c}$ in the latent space of a cut generator. We note that sampling from the prior $p(z_0) = CN(0,1)^{k_0}$ in the original latent space, and pushing that vector through the removed blocks would recover the original generative sampling procedure. There may exist a method for sampling from this "pushforward prior" without saving all parameters from the removed blocks, and such a method could provide a cheap and effective initialization for image recovery, but we defer this exploration to future work.

## A.5 Surgery before Training

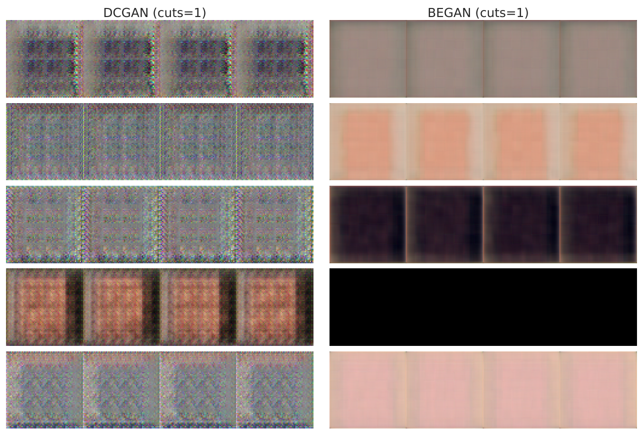

Figure 9: Models cut before training ($c = 1$). All models were trained for 20 epochs. Each row represents one hyperparameter setting, each column is a different random seed. Generated samples show failed training; no faces are produced in any iteration for any setting.

To confirm that layers cannot be cut at training time, we try training BEGAN and DCGAN with a variety of hyperparameter settings for at least 20 epochs after cutting $c = 1$ blocks. In Fig. 9, each row represents a single hyperparameter setting and each column is an independent random sample from after just one epoch. All attempts appear to result in mode collapse with poor quality output images. Note the stark difference when compared to the samples from successful training in the first rows of Fig. 8.

## A.6 Experiment Details

We use L-BFGS for all problems and use the following hyperparamters:

| Model | Task | Learning Rate | Optimizer Steps |
|---|---|---|---|
| DCGAN | CS | 0.1 | 25 |
| | Raw Reconstruction | 1 | 100 |
| BEGAN | CS | 1 | 25 |
| | Raw Reconstruction | 1 | 100 |
| VAE | CS | 1 | 25 |
| | Raw Reconstruction | 1 | 40 |

Table 2: Optimizer Settings. "CS" is for compressive measurements and "Raw reconstruction" refers to reconstructing the image with no degradation ($A = I, \eta = 0$).