Compressed Sensing via Measurement-Conditional Generative Models

Kyung-Su Kim, Jung Hyun Lee, Eunho Yang
kyungsukim@kaist.ac.kr, junghyunlee@kaist.ac.kr, eunhoy@kaist.ac.kr

1Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea

Abstract

A pre-trained generator has been frequently adopted in compressed sensing (CS) due to its ability to effectively estimate signals with the prior of NNs. In order to further refine the NN-based prior, we propose a framework that allows the generator to learn measurement-specific prior distribution, yielding more accurate prediction on a measurement. Our framework has a simple form that only utilizes additional information from a given measurement for prior learning, so it can be easily applied to existing methods. Despite its simplicity, we demonstrate through extensive experiments that our framework exhibits uniformly superior performances by large margin and can reduce the reconstruction error up to an order of magnitude for some applications. We also explain the experimental success in theory by showing that our framework can slightly relax the stringent signal presence condition, which is required to guarantee the success of signal recovery.

1 Introduction

Compressed Sensing (CS) has been a popular approach for decades to recover signals when the number of devices is larger than the size of measurements like in communications [1, 2, 3] or measurements are extremely expensive such as in medical imaging [4, 5, 6] and optical imaging [7]. CS aims to estimate a signal \( \mathbf{x} \in \mathbb{R}^d \) given an undersampled measurement vector \( \mathbf{y} \in \mathbb{R}^m \) under the following linear relationship:

\[
\mathbf{y} = \mathbf{A}\mathbf{x} + \omega,
\]

where \( \mathbf{A} \in \mathbb{R}^{m \times d} \) is a given sensing matrix such that \( m < d \), and \( \omega \) is an unknown noise. Since (1) is an underdetermined linear system, it requires some underlying assumption about the signal to guarantee a unique solution. Classical literature on CS postulates that \( \mathbf{x} \) would be sparse in some known basis and solves (1) by \( \ell_1 \)-minimization.

As neural networks (NNs) have accomplished numerous successes in both supervised learning including regression and classification tasks and unsupervised learning such as clustering and density estimation tasks, many researchers have recently devoted much effort to leveraging NNs as a structural assumption for CS [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 6]. In particular, it is proved in [22] that CS using pre-trained generators (CSPG) is able to reconstruct signals by

*equal contribution
taking advantage of a domain-specific prior, instead of universal sparsity prior. Although methods using LASSO [25] only capture signal sparsity from data transformed by a certain operator (e.g., the wavelet transform), real-world data possess a wide variety of features other than the sparsity. As a result, training a generative model enables its generator to learn a domain-specific distribution, which allows for signal recovery even with a fewer number of measurements than theoretical lower bounds under sparsity prior.

Unfortunately, prior works [22, 18, 26, 27, 10, 8, 14] in CSPG solely deal with training generators to infer the a priori probability distribution of signals itself. As the measurement vector of each training signal is available in training a generator, we can additionally refine our generator by making generator dependent on the measurement and learning the signal distribution conditioned on the measurement. In this paper, we concentrate on how to help generators learn this measurement-conditional distribution.

Our contribution is threefold:

- We propose a framework that enables a generator to learn measurement-specific prior distribution $p(x|y)$. To best of our knowledge, our framework is the first attempt to insert $y$ into generative models for CS.
- We provide supporting theoretical insight for learning measurement-conditional prior that our framework alleviates the stringent signal presence assumption, thus making signal reconstruction much more successful.
- We empirically show consistent and considerable improvements on a wide variety of prior works. We further demonstrate the practicability of our method on real-world data that are difficult for previous generative models to reconstruct (i.e., MRI image reconstruction).

2 Related Work

We introduce several lines of research in CS using deep NNs, which can be largely split into two groups relying on whether to make use of generators or not.

**CS via NNs without generators** This group of studies is mainly concerned with devising NN architectures for certain special purposes. First of all, [28] suggested that the update step in the iterative shrinkage-thresholding algorithm (ISTA) can be represented as each layer of a NN, and proposed a deep architecture as a learned variant of ISTA (LISTA). As LISTA optimizes the network whose form is initially set to ISTA, LISTA directly improves the performance of ISTA by using its architecture. Motivated by this unfolding procedure, extensive studies [29, 30, 31, 32, 33, 24, 23, 6, 19] have been conducted by unfolding state-of-the-art CS algorithms (e.g., approximate message passing, sparse Bayesian learning, and alternating direction method of multipliers and so on) and mapping them to certain network structures. In addition to unfolding-based research, [11] proposed a variant of a convolutional autoencoder to accelerate signal recovery and induce a data-driven dimensionality reduction. [9] presented how to learn a sensing matrix via designing an autoencoder inspired by the projected subgradient method. [20] studied the case where the regularization functional is built as a NN.
Figure 1: Figure 1(a) shows a visualization result based on t-SNE [49] for the marginal generator versus conditional counterpart. Gray points represents outputs of a generator trained to infer $p(x)$ and red/blue/green-colored points indicate outputs of a generator trained to estimate $p(x|y)$. These colored points are much closer to each target signal marked by 'x' than gray points. Figure 1(b) confirms that the reconstruction error decreases in both marginal and conditional generators as the latent optimization ((22) or (5)) progresses. Here, the red curve corresponds to our approach using the measurement-conditional generator, giving smaller reconstruction error for all iterations than the original generator (the blue curve). Here, we consider CSGM as an example, using $m = 1000$ and DCGAN. Each experimental setting is described in Appendix in detail.

CS via NNs with generators The second group is further classified into CS using pre-trained generators (CSPG) and CS using untrained generators (CSUG) depending on whether to train a generator or not. CSPG indicates algorithms to recover signals by the help of generators trained over data. [22] first exploited pre-trained generators to reconstruct signals, providing a recovery guarantee. A number of studies [18, 26, 14, 27, 8, 10] have been done to enhance the performance of CSPG thereafter. In contrast to CSPG, CSUG such as [17, 34] represents methods based on the deep image prior [35] so that the weights of an untrained generator can be trained using only one measurement vector $y_{te}$. Although CSUG is able to recover a signal even in the absence of training data, CSPG is more widely used than CSUG since training data can be usually obtained in practice and CSPG commonly outperforms CSUG in various aspects (e.g., CSUG [36, 34] performs similarly to a wavelet-based LASSO method, whereas CSPG [22, 18] outperforms it).

In this work, we propose a new framework that is easily applicable to CSPG while significantly and uniformly improving existing CSPG methods by modeling conditional generative models. While there are several options in building conditional generative models, our framework is particularly based on generative adversarial network (GAN) [37] due to its popularity in CSPG. As a result, our framework also can be understood as a special case of conditional GAN (cGAN) [38] along with recent studies applying cGAN in various fields [39, 40, 40, 40, 41, 41, 42, 43, 44, 45, 46, 42, 46, 42, 47, 48]. However, all these works solely aim at generation tasks, not at recovery, especially at CS.

3 A Framework of Measurement-Conditional Generative Models for CS

In this section, we outline an essential background of CSPG and propose a simple yet effective scheme to cope with this limitation.
Notation  We use $G_{\theta}(z)$ to denote a generator with parameters $\theta$ and latent variables $z$. Similarly, we use $D_{\phi}(x)$ to denote a discriminator with parameters $\phi$ and input $x$. $(x_{tr}, y_{tr})$ (or $(x_{te}, y_{te})$) indicates a pair of a training (or test) signal $x_{tr}$ (or $x_{te}$) and the corresponding measurement vector $y_{tr}$ (or $y_{te}$) given by (1). We use $p(x, y)$, $p(x)$, and $p(x | y)$ to denote the joint distribution of $(x, y)$, the marginal distribution of $x$, and the conditional distribution of $x$ given $y$, respectively.

3.1 Preliminary: compressed sensing using pre-trained generators

Algorithms in CS using pre-trained generators (CSPG)\(^1\) can be typically divided into the following two phases, training (2) and test (22), respectively.

The training phase aims to find optimal parameters $(\theta^*, \phi^*)$ of generator $G_{\theta}$ and discriminator $D_{\phi}$ given a set of training signals, $\{x_{tr}\}$:

\[
(\theta^*, \phi^*) = F_{(\theta, \phi)}^{opt} \left[ \mathcal{L}_{tr}(G_{\theta}(z), D_{\phi}(\bar{x})) \right],
\]

where $\bar{x}$ denotes either the training signal $x_{tr}$ or the fake signal generated by $G_{\theta}(z)$, $\mathcal{L}_{tr}(\mathcal{M})$ indicates a loss function for training involved with models $\mathcal{M}$, and $F_{\mathcal{T}}^{opt}[\cdot]$ is defined by an operator to optimize its input with respect to $\mathcal{T}$. In general, $\mathcal{M} = \{G_{\theta}(x), D_{\phi}(\bar{x})\}$ and $\mathcal{T} = \{\theta, \phi\}$, but $\mathcal{T}$ can vary depending on $\mathcal{M}$.

In the test phase, a target $x_{te}$ is estimated as $\hat{x}$ by the following two-stage process: we first find the optimal latent variables $z^*$ given trained $\theta^*$ and measurement $y_{te}$:

\[
z^* = F_{z}^{opt} \left[ \mathcal{L}_{te}(G_{\theta^*}(z), y_{te}) \right],
\]

where $\mathcal{L}_{te}(G_{\theta^*}(z), y_{te})$ represents an objective function for inference. Then, given estimated $z^*$ and trained $\theta^*$, we recover the target $x_{te}$ as $\hat{x} = G_{\theta^*}(z^*)$.

3.2 Measurement-conditional pre-trained generators

In contrast to existing CSPG methods to learn the marginal probability of our interest, $p(x)$, under the framework of the original GAN such as DCGAN [50], our framework provides a way to learn the measurement-conditional distribution $p(x | y)$ by leveraging the concept of conditional GAN [38] framework. Essentially, we generate the signal distribution not just by ‘noise’, but with specific ‘measurement information’, thus further refining the prior with the additional information of measurement vectors. Under the classical assumption in CS on the existence of an inverse map from $y$ to $x$ for all $(x, y)$ satisfying $p(x, y) > 0$, a generator could recover $x_{te}$ successfully from $y_{te}$ if it exactly inferred $p(x | y)$. In practice, of course, hardly does such an ideal inverse map exist. Nevertheless, as empirically shown in Figure 1(a), a conditional generator can still provide a more refined prior for the target signal, thereby increasing the chance of success in the subsequent signal reconstruction stage like (22).

---

\(^1\)While there are some exceptions, most state-of-the-art CSPG methods are based on Generative Adversarial Networks (GANs) [37, 50]. Hence, we focus on GAN-based prior generators throughout the paper, but it can be seamlessly extended to other generative models such as Variational Autoencoder (VAE) [22].
Overall, in contrast to (2) and (22), we inject the measurement information into \( G_\theta \) and \( D_\phi \) as an input for both the training and test phases so that \( \theta \) is able to learn the conditional prior:

\[
(\theta^*, \phi^*) = \mathcal{F}^{\text{opt}}_{(\theta, \phi)} \left[ \mathcal{L}_{tr}(G_\theta(z, y_{tr}), D_\phi(x, y_{tr})) \right] \quad \text{for training phase},
\]

\[
z^* = \mathcal{F}^{\text{opt}}_{z} \left[ \mathcal{L}_{te}(G_\theta(z, y_{te}), y_{te}) \right], \quad \hat{x} = G_\theta^*(z^*, y_{te}) \quad \text{for test phase}.
\]

This approach, dubbed ‘Inserting Measurements’ (IM), can be readily applicable to any CS method using pre-trained generators (CSPG) as shown in Figure 2.

Unlike discriminative models (i.e., learning NNs for CS that directly map measurements to target signals without any generator), a generative model for CS has the latent variable vector \( z \), which can be optimized in the test phase via (22) to find a more closer estimate to the true signal. In this process, a variety of samples from the generative model can be simultaneously used to help achieve this objective effectively. Our approach improves the generative model for CS by taking similar spirit of discriminative approach while inheriting the advantages of generative models, as demonstrated in Figure 1(b).

4 Revising Existing CSPG Models under Our Framework

We now provide examples showcasing our framework using IM for varied CSPG models such as Compressed Sensing using Generative Models (CSGM) [22] and Projected Gradient Descent GAN (PGDGAN) [26]. The applications of IM to Deep Compressed Sensing (DCS) [10] and SparseGen [18] are deferred to Appendix.

4.1 Compressed Sensing using Generative Models (CSGM)

Compressed Sensing using Generative Models (CSGM) [22] is the first work to propose the GAN-based CSPG framework in (2) and (22). While it allows any GAN models, if the standard GAN training objective including DCGAN [50] is used as showed in [22], a generator \( G_\theta \) and a discriminator \( D_\phi \) are trained by solving the following min-max problem:

\[
(\theta^*, \phi^*) \leftarrow \arg \min_{\theta} \arg \max_{\phi} \mathbb{E}_{x_{tr} \sim p(x)} \left[ \ln D_\phi(x_{tr}) \right] + \mathbb{E}_{z \sim p_z(z)} \left[ \ln \left( 1 - D_\phi(G_\theta(z)) \right) \right]. \quad (6)
\]
Another reasonable GAN instance under its framework is BEGAN. In this case, the training phase of CSGM can be given as

\[
(\theta^*, \phi^*) \leftarrow \text{argmax}_{\theta} \text{argmin}_{\phi} \mathbb{E}_{x_{tr} \sim p(x), z \sim p_z} \left[ R_{\phi}(x_{tr}) - \zeta R_{\phi}(G_{\theta}(z)) \right],
\]

where \( R_{\phi}(\bar{x}) = \| \bar{x} - D_{\phi}(\bar{x}) \|_p \) denotes the reconstruction loss determined by \( D_{\phi} \) in terms of \( p = 1 \) or \( 2 \) norms and the parameter \( \zeta \) controls the balance between auto-encoding true images and distinguishing real images from fake ones. While \( D_{\phi} \) in (6) outputs a scalar value (probability) indicating whether its input is a real image or not, \( D_{\phi} \) in (7) reconstructs an image based on its input.

CSGM in the test phase computes an estimate \( \hat{x} = G_{\theta^*}(z^*) \) of target signal \( x_{te} \) by minimizing the following loss with respect to the input noise \( z \) of the pre-trained generator \( G_{\theta^*} \):

\[
z^* \leftarrow \text{argmin}_{z} \| y_{te} - AG_{\theta^*}(z) \|^2, \quad \hat{x} = G_{\theta^*}(z^*).
\]

CSGM-IM We now show how the proposed scheme can be applied to CSGM under our framework. We name this application CSGM-IM.

The training phase of CSGM-IM learns \((\theta, \phi)\) according to the following optimization (9) (for DCGAN) and (10) (for BEGAN), which are revised from (6) and (7), respectively:

\[
(\theta^*, \phi^*) \leftarrow \text{argmin}_{\theta} \text{argmax}_{\phi} \mathbb{E}_{x_{tr} \sim p(x)} \left[ \ln D_{\phi}(x_{tr}, y_{tr}) \right]
\]

\[
+ \mathbb{E}_{z \sim p_z(z)} \left[ \ln \left( 1 - D_{\phi}(G_{\theta}(z, y_{tr}), y_{tr}) \right) \right], \quad (9)
\]

\[
(\theta^*, \phi^*) \leftarrow \text{argmax}_{\theta} \text{argmin}_{\phi} \mathbb{E}_{x_{tr} \sim p(x), z \sim p_z} \left[ R_{\phi}(x_{tr}, y_{tr}) - \zeta R_{\phi}(G_{\theta}(z, y_{tr}), y_{tr}) \right], \quad (10)
\]

where \( R_{\phi}(x, y) = \| x - D_{\phi}(x, y) \|_p \) represents the reconstruction loss determined by the modified discriminator \( D_{\phi}(x, y) \).

In the test phase, CSGM-IM estimates the target signal \( x_{te} \) as \( \hat{x} = G_{\theta^*}(z^*, y_{te}) \) by consistently feeding the test measurement to the learned generator:

\[
z^* \leftarrow \text{argmin}_{z} \| y_{te} - AG_{\theta^*}(z, y_{te}) \|^2, \quad \hat{x} = G_{\theta^*}(z^*, y_{te}).
\]

We illustrate how CSGM-IM differs from CSGM in Figure 2 and defer their algorithms to Appendix.

### 4.2 Projected Gradient Descent GAN (PGDGAN)

Projected Gradient Descent GAN (PGDGAN) [26] is a representative work of applying a projected gradient method to the CSPG framework. Similarly to CSGM, the training phase of PGDGAN can be any learning schemes to optimize a generator \( G_{\theta} \) like (6) or (7). PGDGAN in the test phase however restores signals by jointly reducing Euclidean measurement error [26] and making a signal estimate belong to the range of pre-trained generator. Specifically, PGDGAN in the test phase
computes an estimate $\hat{x} = x_T$ of the target signal $x_{te}$ by iteratively solving the following recursive formula with respect to the input noise $z$ of a pre-trained generator $G^{\theta^*}$:

$$x_{t+1} = P_{G^{\theta^*}} \left( x_t - \alpha \frac{1}{2} \nabla \| y_{te} - Ax_t \|^2 \right) = G^{\theta^*} \left( \arg \min _{z} \| x_t - \alpha A^T (Ax_t - y_{te}) - G^{\theta^*} (z) \| \right),$$

(12)

for $t = 0, \cdots, T - 1$, where $P_{G^{\theta^*}} (h) = G^{\theta^*} (\arg \min _{z} \| h - G^{\theta^*} (z) \|)$ denotes a projection operator mapping the input $h$ to a point nearest to $h$ in the range of the pre-trained generator $G^{\theta^*}$, $x_0 = 0$ or $A^T y_{te}$ in general, $\alpha$ is a learning rate, and $T$ is the total number of iterations. Note that the inner and outer calculations of the second term in (12) update an estimate to the direction of decreasing Euclidean measurement error and project this updated estimate into the range of $G^{\theta^*}$, respectively. Therefore, PGDGAN can be regarded as solving a least squares problem with a network-based regularizer rather than with a sparsity promoting regularizer conventionally used in CS.

**PGDGAN-IM** We now describe how IM can be applied to PGDGAN under our framework. We dub this application PGDGAN-IM.

As PGDGAN has the same training phase as CSGM, PGDGAN-IM also has the same training phase as CSGM-IM ((9) or (10)). In the test phase, PGDGAN-IM estimates the target signal $x_{te}$ as $\hat{x} = x_T$ by consistently feeding the test measurement to the trained generator:

$$x_{t+1} = G^{\theta^*} \left( \arg \min _{z} \| x_t - \alpha A^T (Ax_t - y_{te}) - G^{\theta^*} (z, y_{te}) \|, y_{te} \right).$$

(13)

We provide the algorithms of PGDGAN and PGDGAN-IM in Appendix.

### 5 Theoretical Insight

In this section, we provide a theoretical insight for the reason why inserting the measurements into the generative model improves the performance of signal reconstruction. To demonstrate this, we consider PGDGAN in Section 4.2 as an example and show how PGDGAN-IM enhances the performance of PGDGAN. Both algorithms require a specific $(S, \gamma)$-RIP for a given sensing matrix $A$, which is easily satisfied in practice with rich theoretical background/guarantees [51, 22, 18, 34]:

**Definition 1.** For a parameter $\gamma > 0$, a matrix $A \in \mathbb{R}^{m \times d}$ satisfies $(S, \gamma)$-RIP, if for all $x \in S$,

$$(1 - \gamma) \| x \| \leq \| A x \| \leq (1 + \gamma) \| x \| .$$

(14)

Under $(S, \gamma)$-RIP, we present a sufficient condition for PGDGAN or PGDGAN-IM to recover signals.

**Theorem 1.** Let $\{x_t\}_{t=0}^T$ be a set of outputs obtained from each iteration of PGDGAN or PGDGAN-IM, with learning rate $\alpha$ small enough. Let $x_{te}$ and $y_{te}$ be a target signal vector and its measurement vector, respectively. Suppose $A^{m \times d}$ satisfies $(S, 1 - \gamma, 1 + \gamma)$-RIP with high probability where $S$ denotes arbitrary set including $\{x_t\}_{t=0}^T$ and $x_{te}$. Suppose further that

(a) In the case for PGDGAN, there exists a vector $z^*$ satisfying $G^{\theta^*} (z^*) = x_{te},$
Table 1: The reported numbers are the mean of probability of $z$ satisfying the condition (a) or (b) in Theorem 1, with the standard deviation of the means computed across 64 random samples in the test set of CelebA. $\theta^*$ comes from (6) and (9). Here, $m = 1000$ and $\epsilon = 0.125$. Each probability is calculated by sampling 1000 $z$'s.

| Probability for condition (a) or (b) to hold | Mean ± Standard Deviation |
|---------------------------------------------|---------------------------|
| $P_z(\|G_{\theta^*}(z) - x_{te}\|^2 < \epsilon)$ | $0.003 ± 0.007$ |
| $P_z(\|G_{\theta^*}(z, y_{te}) - x_{te}\|^2 < \epsilon)$ | $0.984 ± 0.124$ |

(b) In the case for PGDGAN-IM, there exists a vector $z^*$ satisfying $G_{\theta^*}(z^*, y_{te}) = x_{te}$.

Then, there exists $T \propto \log(1/\epsilon)$ such that the signal estimate $\hat{x} = x_T$ satisfies $\|\hat{x} - x_{te}\| \leq \epsilon$.

Theorem 1 shows that the condition required for successful signal recovery is changed from (a) to (b), with the introduction of IM into PGDGAN. Both conditions (a) and (b) require that the target signal to recover is included in the range of a pre-trained generator, but we empirically show in Table 1 that the condition (b) with IM is much easier to be satisfied than (a), which is consistent with our motivation introduced in Section 3.2. Overall, IM improves the performance of signal recovery by relaxing the condition for signal existence from (a) to (b).

6 Experiments

In order to evaluate the effectiveness of our algorithm, we focus on conducting experiments on CelebA [52] dataset, which is a common but more difficult task than on MNIST [53] or OMNIGLOT [54] datasets. The images are cropped at the center to the size $64 \times 64 \times 3$ ($d = 12288$) and normalized into the range $[-1, 1]$. For inference, we utilize 64 random images in the test set and compute the reconstruction error $\|x_{te} - \hat{x}\|^2$ with 95% confidence interval of 5 trials. Each entry of the sensing matrix $A$ is sampled from the normal distribution $N(0, 1/m)$. We follow the same experimental setting as [22] unless otherwise specified.

When adding $y$ into generative models, $y$ is concatenated to $z$ in $G_\theta$ in both DCGAN and BEGAN. $y$ is also concatenated to the embedding layer in $D_\phi$ in BEGAN. Owing to the absence of an embedding layer in $D_\phi$ in DCGAN, we emulate an architecture suggested in [55]. Further information about experimental setting and the result of experiments on different $m$ are given in Appendix.

CSGM-IM As illustrated in Table 2, IM considerably improves CSGM for all $m$. On average, IM decreases the reconstruction error by above 30%. Although IM greatly enhances the performance of CSGM for small $m$, it still exhibits performance saturation like CSGM. When exploiting BEGAN, CSGM-IM demonstrates much more compelling performance as well as overcomes such a limitation. Comparing with the last column in Table 2 and 2, the reconstruction error per pixel of CSGM-IM using BEGAN is smaller than that using DCGAN for every $m$. More strikingly, for $m \geq 1000$ in Table 2, the reconstruction error per pixel of CSGM-IM using BEGAN is almost less than that of CSGM using BEGAN by an order of magnitude. As a result, not only does CSGM-IM using BEGAN recover images similar to the original ones when $m = 1000$ ($m/d \approx 8.14\%$), but reconstructs images almost the same as the original ones when $m = 5000$ ($m/d \approx 40.69\%$) as shown in Figure 3.

PGDGAN-IM For fair comparisons, we follow the same hyperparameters as [26]. In Table 2, PGDGAN-IM outperforms PGDGAN by reducing the reconstruction error by above 30% on average.
Table 2: Reconstruction error per pixel for existing methods and those with the IM framework. IM significantly and uniformly improves all existing methods. DCGAN is utilized unless ‘B’ is marked where ‘B’ stands for the use of BEGAN.

| Method          | $m = 20$      | $m = 100$     | $m = 500$     | $m = 1000$    | $m = 5000$    |
|-----------------|---------------|---------------|---------------|--------------|--------------|
| CSGM [22]       | 0.304 ± 0.068 | 0.104 ± 0.012 | 0.039 ± 0.004 | 0.033 ± 0.003 | 0.029 ± 0.003 |
| CSGM-IM         | 0.209 ± 0.011 | 0.072 ± 0.007 | 0.029 ± 0.003 | 0.022 ± 0.003 | 0.018 ± 0.002 |
| CSGM (B)        | 0.213 ± 0.031 | 0.104 ± 0.012 | 0.071 ± 0.011 | 0.068 ± 0.010 | 0.066 ± 0.011 |
| CSGM-IM (B)     | 0.185 ± 0.008 | 0.058 ± 0.007 | 0.015 ± 0.002 | 0.008 ± 0.001 | 0.002 ± 0.000 |
| PGDGAN [26]     | 0.630 ± 0.076 | 0.128 ± 0.010 | 0.049 ± 0.006 | 0.038 ± 0.004 | 0.032 ± 0.004 |
| PGDGAN-IM       | 0.438 ± 0.017 | 0.094 ± 0.007 | 0.031 ± 0.003 | 0.023 ± 0.003 | 0.019 ± 0.002 |
| DCS [10]        | 0.246 ± 0.005 | 0.159 ± 0.007 | 0.114 ± 0.007 | 0.087 ± 0.003 | 0.082 ± 0.002 |
| DCS-IM          | 0.110 ± 0.008 | 0.049 ± 0.005 | 0.017 ± 0.002 | 0.010 ± 0.001 | 0.002 ± 0.000 |
| SparseGen [18]  | 0.374 ± 0.060 | 0.118 ± 0.018 | 0.038 ± 0.005 | 0.030 ± 0.004 | 0.024 ± 0.003 |
| SparseGen-IM    | 0.224 ± 0.022 | 0.072 ± 0.005 | 0.028 ± 0.003 | 0.021 ± 0.002 | 0.017 ± 0.002 |

Table 3: Reconstruction error per pixel with 95% confidence interval of 5 trials for Section 4.1 using BEGAN.

| Method          | $m = 500$     | $m = 1000$    | $m = 2500$    | $m = 5000$   |
|-----------------|---------------|---------------|--------------|-------------|
| CSGM [22]       | 0.0674 ± 0.0118 | 0.0671 ± 0.0097 | 0.0655 ± 0.0101 | 0.0659 ± 0.0123 |
| CSGM-IM         | 0.0159 ± 0.0007 | 0.0120 ± 0.0007 | 0.0092 ± 0.0005 | 0.0082 ± 0.0005 |

**DCS-IM** To verify the validness of our method in the absence of $D_\phi$, we apply IM to DCS. Table 2 shows that IM cuts down on the reconstruction error of DCS by above 70%. When $m \leq 100$, DCS-IM particularly outperforms all the other methods, which implies that IM makes the strength of DCS in the small $m$ far more outstanding. Moreover, IM reduces the reconstruction error per pixel of DCS by an order of magnitude for $m \geq 1000$, which leads to successful signal recovery like CSGM-IM using BEGAN as illustrated in Figure 3.

**SparseGen-IM** We only consider the wavelet basis due to the fact that [18] recommend the wavelet basis rather than the discrete cosine transform. Table 2 indicates that SparseGen-IM surpasses SparseGen, averagely curtailing the reconstruction error by around 38%.

7 Application to Magnetic Resonance Imaging

To validate the practicality of IM on real-world data, we run experiments on the fastMRI [56] dataset. In particular, we utilize the knee dataset. Owing to the low quality of test slices, we regard the validation slices as the test set. Similarly to [56], the images are cropped at the center to the size $256 \times 256$ and downsampled to the size $128 \times 128 (d = 16384)$. Any data augmentation is not used to show that IM can encourage previous methods to perform well even on a small number of data. For test, we utilize 64 random images in the validation slices. Further information is given in Appendix.

Same as Section 6, each entry of the sensing matrix $A$ is also sampled from the normal distribution $\mathcal{N}(0, 1/m)$, and the reconstruction error is used as an evaluation metric. Unlike (1), noise is not
Figure 3: Reconstructed images on CelebA when $m = 1000$ ($m/d \approx 8.14\%$) and $m = 5000$ ($m/d \approx 40.69\%$). The first row represents the original images. The second and third row indicate images recovered by existing methods and existing methods + IM when $m = 1000$. The fourth and last rows display images recovered by existing methods and existing methods + IM when $m = 5000$.

added when creating a measurement vector $y_{te}$. In this experiment, we solely employ the BEGAN architecture due to the fact that it works best among our experiments.

Similarly to Section 6, CSGM-IM achieves much better performance than CSGM. As seen in Table 3, IM decreases the reconstruction error of CSGM by closely an order of magnitude for $m \geq 2500$. Not only that, as illustrated in Figure 4, CSGM-IM recovers images highly analogous to the original ones whereas CSGM cannot at all.

8 Conclusion

We propose a simple yet effective method, Inserting Measurements, which enables a generator to learn $p(x|y)$ instead of $p(x)$. Even in the IM framework, the characteristic of generative models remains, which allows us to find a more closer estimate to the true signal by the latent optimization. By leveraging both advantages of discriminative and generative models, IM can yield much smaller reconstruction error than existing methods up to an order of magnitude. We therefore expect IM to be useful for a variety of CS applications as well as image recovery.
Figure 4: Reconstructed images on fastMRI when \( m = 5000 \) \((m/d \approx 30.52\%)\). The first column has the original image. The second and last column show images recovered by CSGM and CSGM-IM using BEGAN.

References

[1] Hengtao He, Chao-Kai Wen, Shi Jin, and Geoffrey Ye Li. Deep learning-based channel estimation for beamspace mmWave massive MIMO systems. *IEEE Wireless Communications Letters*, 7(5): 852–855, 2018.

[2] Jinyoup Ahn, Byonghyo Shim, and Kwang Bok Lee. Expectation propagation-based active user detection and channel estimation for massive machine-type communications. In *IEEE International Conference on Communications Workshops*, pages 1–6, 2018.

[3] Kyung-Su Kim and Sae-Young Chung. Tree search network for sparse estimation. *Digital Signal Processing*, 100:102680, 02 2020. doi: 10.1016/j.dsp.2020.102680.

[4] Yo Seob Han, Jaejun Yoo, and Jong Chul Ye. Deep residual learning for compressed sensing CT reconstruction via persistent homology analysis. *arXiv preprint:1611.06391*, 2016.

[5] Michael Lustig, David L. Donoho, Juan M. Santos, and John M. Pauly. Compressed sensing MRI. In *IEEE Signal Processing Magazine*, 2007.

[6] Jian Sun, Huibin Li, Zongben Xu, et al. Deep ADMM-Net for compressive sensing MRI. In *Advances in Neural Information Processing Systems*, pages 10–18, 2016.

[7] Rebecca M Willett, Roummel F Marcia, and Jonathan M Nichols. Learning a compressed sensing measurement matrix via gradient unrolling. 97, 2019.

[8] Aditya Grover and Stefano Ermon. Uncertainty autoencoders: Learning compressed representations via variational information maximization. In *International Conference on Artificial Intelligence and Statistics*, pages 2514–2524, 2019.

[9] Shanshan Wu, Alexandros Dimakis, Sujay Sanhavi, Felix Yu, Daniel Holtmann-Rice, Dmitry Storcheus, Afshin Rostamizadeh, and Sanjiv Kumar. Learning a compressed sensing measurement matrix via gradient unrolling. 97, 2019.

[10] Yan Wu, Mihaela Rosca, and Timothy Lillicrap. Deep compressed sensing. In *International Conference on Machine Learning*, pages 6850–6860, 2019.

[11] Ali Mousavi, Gautam Dasarathy, and Richard G Baraniuk. A data-driven and distributed approach to sparse signal representation and recovery. In *International Conference on Learning Representations*, 2019.
[12] Rabeeh Karimi Mahabadi, Junhong Lin, and Volkan Cevher. A learning-based framework for quantized compressed sensing. *IEEE Signal Processing Letters*, 26(6):883–887, 2019.

[13] Qi Lei, Ajil Jalal, Inderjit S Dhillon, and Alexandros G Dimakis. Inverting deep generative models, one layer at a time. In *Advances in Neural Information Processing Systems*, pages 13910–13919, 2019.

[14] Ankit Raj, Yuqi Li, and Yoram Bresler. GAN-based projector for faster recovery with convergence guarantees in linear inverse problems. In *IEEE/CVF International Conference on Computer Vision*, pages 5601–5610, 2019.

[15] Sriram Ravula and Alexandros G Dimakis. One-dimensional deep image prior for time series inverse problems. *arXiv preprint:1904.08594*, 2019.

[16] Fabian Latorre Gómez, Armin Eftekhari, and Volkan Cevher. Fast and provable ADMM for learning with generative priors. *arXiv preprint:1907.03343*, 2019.

[17] Dave Van Veen, Ajil Jalal, Mahdi Soltanolkotabi, Eric Price, Sriram Vishwanath, and Alexandros G Dimakis. Compressed sensing with deep image prior and learned regularization. *arXiv preprint:1806.06438*, 2018.

[18] Manik Dhar, Aditya Grover, and Stefano Ermon. Modeling sparse deviations for compressed sensing using generative models. In *International Conference on Machine Learning*, pages 1222–1231, 2018.

[19] Morteza Mardani, Qingyun Sun, David Donoho, Vardan Papyan, Hatef Monajemi, Shreyas Vasanawala, and John Pauly. Neural proximal gradient descent for compressive imaging. In *Advances in Neural Information Processing Systems*, pages 9573–9583, 2018.

[20] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. Adversarial regularizers in inverse problems. In *Advances in Neural Information Processing Systems*, pages 8507–8516, 2018.

[21] Chinmay Hegde. Algorithmic aspects of inverse problems using generative models. In *Annual Allerton Conference on Communication, Control, and Computing*, pages 166–172. IEEE, 2018.

[22] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. JMLR. org, 2017.

[23] Hao He, Bo Xin, Satoshi Ikehata, and David Wipf. From Bayesian sparsity to gated recurrent nets. In *Advances in Neural Information Processing Systems*, pages 5554–5564, 2017.

[24] Chris Metzler, Ali Mousavi, and Richard Baraniuk. Learned D-AMP: Principled neural network based compressive image recovery. In *Advances in Neural Information Processing Systems*, pages 1772–1783, 2017.

[25] Robert Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
[26] Viraj Shah and Chinmay Hegde. Solving linear inverse problems using GAN priors: An algorithm with provable guarantees. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4609–4613. IEEE, 2018.

[27] Maya Kabkab, Pouya Samangouei, and Rama Chellappa. Task-aware compressed sensing with generative adversarial networks. In *AAAI Conference on Artificial Intelligence*, 2018.

[28] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning*, pages 399–406, 2010.

[29] Thomas Moreau and Joan Bruna. Understanding neural sparse coding with matrix factorization. In *International Conference on Learning Representations*, 2017.

[30] Raja Giryes, Yonina C Eldar, Alex M Bronstein, and Guillermo Sapiro. Tradeoffs between convergence speed and reconstruction accuracy in inverse problems. *IEEE Transactions on Signal Processing*, 66(7):1676–1690, 2018.

[31] Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In *Advances in Neural Information Processing Systems*, pages 9061–9071, 2018.

[32] Eric W Tramel, Angélique Drémeau, and Florent Krzakala. Approximate message passing with restricted Boltzmann machine priors. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(7):073401, 2016.

[33] Mark Borgerding, Philip Schniter, and Sundeep Rangan. AMP-inspired deep networks for sparse linear inverse problems. *IEEE Transactions on Signal Processing*, 65(16):4293–4308, 2017.

[34] G Jagatap and C Hegde. Algorithmic guarantees for inverse imaging with untrained network priors. *Advances in Neural Information Processing Systems*, 2019.

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

[36] R Heckel et al. Deep decoder: Concise image representations from untrained non-convolutional networks. In *International Conference on Learning Representations*, 2019.

[37] 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.

[38] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint:1411.1784*, 2014.

[39] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[40] Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. Towards diverse and natural image descriptions via a conditional GAN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979, 2017.
[41] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional GANs for image editing. *arXiv preprint:1611.06355*, 2016.

[42] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. Face aging with conditional generative adversarial networks. In *IEEE International Conference on Image Processing*, pages 2089–2093. IEEE, 2017.

[43] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[44] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[45] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8183–8192, 2018.

[46] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014(5):2, 2014.

[47] Hao Ye, Geoffrey Ye Li, Biing-Hwang Fred Juang, and Kathiravetpillai Sivanesan. Channel agnostic end-to-end learning based communication systems with conditional GAN. In *IEEE GLOBECOM Workshops*, pages 1–5. IEEE, 2018.

[48] Hao Ye, Le Liang, Geoffrey Ye Li, and Biing-Hwang Juang. Deep learning based end-to-end wireless communication systems with conditional GAN as unknown channel. *IEEE Transactions on Wireless Communications*, 2020.

[49] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[50] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint:1511.06434*, 2015.

[51] Kiryung Lee, Yoram Bresler, and Marius Junge. Subspace methods for joint sparse recovery. *IEEE Transactions on Information Theory*, 58(6):3613–3641, 2012.

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

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

[54] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350:1332–1338, 2015.
[55] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069, 2016.

[56] Jure Zbontar, Florian Knoll, Anuroop Sriram, Matthew J Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J Geras, Joe Katsnelson, Hersh Chandarana, et al. FastMRI: An open dataset and benchmarks for accelerated MRI. *arXiv preprint arXiv:1811.08839*, 2018.

[57] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, volume 70, pages 1126–1135. PMLR, 2017.
A Revising Existing CSPG Models under Our Framework

A.1 Deep Compressed Sensing (DCS)

DCS

To recover signals faster and more accurately than CSGM, [10] proposed Deep Compressed Sensing (DCS) by jointly training the latent variables $z$ and the weights $\theta$ of a generator without any discriminator $D_\phi$ via meta-learning [57]. More concretely, the training phase of DCS is given as follows: for each training sample $x_{tr}$, the latent optimization is carried out by minimizing $\|y_{tr} - AG_\theta(z)\|$ while keeping $\theta$ fixed, then $\theta$ are subsequently trained in DCS:

$$z^*_{x_{tr}} = \arg\min_z \|y_{tr} - AG_\theta(z)\|, \quad (15)$$

$$\theta^* = F_{\text{opt}}\theta \left[ L_{tr}(G_\theta(z^*_{x_{tr}})) \right] = \arg\min_{\theta} \mathbb{E}_{x_{tr}\sim p(x)} \left[ \|y_{tr} - AG_\theta(z^*_{x_{tr}})\|^2 \right]. \quad (16)$$

Note that the test phase of DCS is the same as that of CSGM (8), so it is not shown here.

DCS-IM

Similarly to (9) and (10), the IM framework makes the measurement information also taken by $G_\theta$ as additional input in DCS. Therefore, by applying IM to DCS, (15) and (16), the procedure for training $\theta$, are substituted with (17) and (18), respectively.

$$z^*_{x_{tr}} = \arg\min_z \|y_{tr} - AG_\theta(z, y_{tr})\|, \quad (17)$$

$$\theta^* = F_{\text{opt}}\theta \left[ L_{tr}(G_\theta(z^*_{x_{tr}}, y_{tr})) \right] = \arg\min_{\theta} \mathbb{E}_{x_{tr}\sim p(x)} \left[ \|y_{tr} - AG_\theta(z^*_{x_{tr}}, y_{tr})\|^2 \right]. \quad (18)$$

As the formula of DCS in the test phase is the same as that of CSGM (8), the formula of DCS with the IM framework is also identical to that of CSGM-IM (11), but notice that a couple of gradient descent steps are sufficient to implement (8) in DCS and (11) in DCS with the IM framework.

In such a case of applying IM to DCS in the training and test phases, we name it DCS-IM. The algorithms of DCS and DCS-IM are given in Section C.

A.2 Sparse deviations for compressed sensing using Generative models (SparseGen)

SparseGen

[18] came up with the idea to combine a domain-specific generative model prior with sparsity prior to enhance the generalization of CSGM, called SparseGen. Similarly to CSGM, the training phase of SparseGen can be any learning scheme to optimize a generator $G_\theta$ like (6) or (7). In the test phase of SparseGen, given trained parameters $\theta^*$ of a generator, sparse deviations from the support set of $G_\theta^*$ are allowed to consider signals even outside the range of $G_\theta^*$, which results in an estimate being of the form $G_\theta^* + \nu$ where $\nu$ is an augmented sparse estimate. Hence, $L_{te}$ should be involved with $\ell_1$ minimization with respect to $\nu$ as well as the optimization of $G_\theta^*$ with respect to $z$:

$$(z^*, \nu^*) = F_{\text{opt}}(z, \nu | L_{te}(G_\theta^*(z), \nu | y_{te})) = \arg\min_{z, \nu} \|A(G_\theta^*(z) + \nu) - y_{te}\| + \lambda \|B\nu\|_1, \quad (19)$$

where $B$ is a transform matrix promoting sparsity of the vector $B\nu$, and $\lambda$ is the Lagrange multiplier. By using $(z^*, \nu^*)$ given in (19), SparseGen estimates the target signal as $G_\theta^*(z^*) + \nu^*$.  

16
**SparseGen-IM** IM can be run on SparseGen by adjusting (19) to (20):

\[(z^*, \nu^*) = \mathcal{T}_{(z, \nu)}^{opt}[\mathcal{L}_{te}(G_{\theta^*}(z, y_{te}), \nu | y_{te})] = \arg\min_{z, \nu} \|A(G_{\theta^*}(z, y_{te}) + \nu) - y_{te}\| + \lambda \|B\nu\|_1, \quad (20)\]

which we name SparseGen-IM. In this case, \(G_{\theta^*}(z^*, y_{te}) + \nu^*\) becomes an estimate of \(x_{te}\). As SparseGen has the same training phase as CSGM, SparseGen-IM also has the same training phase as CSGM-IM ((9) or (10)).

The algorithms of SparseGen and SparseGen-IM are given in Section C.

**B Proof of Theorem 1**

The proof of Theorem 1 is based on that of Theorem 1 in [34].

Let the following algorithm be the test phase of PGDGAN (without blue notes) and PGDGAN-IM (with blue notes) in the noiseless setting, \(y = Ax\).

**Input:** \(y_{te} \in \mathbb{R}^m, A \in \mathbb{R}^{m \times d}, G_{\theta^*} : \mathbb{R}^v \mapsto \mathbb{R}^d, \alpha \in \mathbb{R}_+, T \in \mathbb{N}\)

**Initialize:** \(x_0 = 0 \in \mathbb{R}^d\)

1. for \(t = 0\) to \(T - 1\) do
2. \(w_t = x_t - \alpha A^\top(Ax_t - y_{te})\)
3. \(z_t = \arg\min_{z} \|w_t - G_{\theta^*}(z, y_{te})\|\)
4. \(x_{t+1} = G_{\theta^*}(z_t, y_{te})\)
5. end for

**Output:** the signal estimate \(\hat{x} = x_T\)

Referring to the above algorithm, we prove Theorem 1 under the condition (b) (i.e., the guarantee of signal recovery in PGDGAN-IM). The proof of Theorem 1 under the condition (a) (i.e., the guarantee of signal recovery in PGDGAN) is trivial if we remove all the blue notes and use the condition (a) instead of (b).

It follows that

\[
\|y_{te} - Ax_{t+1}\|^2 - \|y_{te} - Ax_{t}\|^2
= (\|Ax_{t+1}\|^2 - \|Ax_{t}\|^2) - 2(y_{te}^\top Ax_{t+1} - y_{te}^\top Ax_{t})
= \|Ax_{t+1} - Ax_{t}\|^2 - 2(Ax_{t})^\top(Ax_{t}) - 2(Ax_{t})^\top(Ax_{t+1}) - 2(y_{te}^\top Ax_{t+1} - y_{te}^\top Ax_{t})
= \|Ax_{t+1} - Ax_{t}\|^2 + 2(Ax_{t} - y_{te})^\top(Ax_{t+1} - Ax_{t}).
\]

(21)

Step 2 of PGDGAN-IM is given as

\[
w_t = x_t - \alpha A^\top(Ax_t - y_{te}).
\]

(22)

Afterward, by using \(w_t\) in (22), Step 3 of PGDGAN-IM updates the latent variables \(z\) so that \(G_{\theta^*}(z, y_{te})\) lies in the range of the generator \(G_{\theta^*}\) while being the closest to \(w_t\).

From Step 4, it follows that \(G_{\theta^*}(z_t, y_{te}) = x_{t+1}\). By the condition (b), there exists \(z^*\) satisfying \(G_{\theta^*}(z^*, y_{te}) = x_{te}\). Then, by using these two results and the definition of \(z^*\) in Step 3, we obtain

\[
\|x_{t+1} - w_t\|^2 \leq \|x_{te} - w_t\|^2.
\]

(23)
By applying (22) to (23), we obtain
\[ \|x_{t+1} - x_t + \alpha A^\top (Ax_t - y_{te})\|^2 \leq \|x_{te} - x_t + \alpha A^\top (Ax_t - y_{te})\|^2. \] (24)

(24) can be written as
\[ \frac{1}{\alpha} \|x_{t+1} - x_t\|^2 + 2(Ax_t - y_{te})^\top A(x_{t+1} - x_t) \leq \frac{1}{\alpha} \|x_t - x_{te}\|^2 - 2\|y_{te} - Ax_t\|^2. \] (25)

Then, by applying (21) to (25), we gain
\[ \|y_{te} - Ax_{t+1}\|^2 + \|y_{te} - Ax_t\|^2 \leq \frac{1}{\alpha} \|x_t - x_{te}\|^2 - \frac{1}{\alpha} \|x_{t+1} - x_t\|^2 + \|A(x_{t+1} - x_t)\|^2. \] (26)

From \((S, 1 - \gamma, 1 + \gamma)\)-RIP condition, the first and last terms on the right-hand side of (26) are upper bounded respectively by
\[ \frac{1}{\alpha} \|x_t - x_{te}\|^2 \leq \frac{1}{\alpha(1 - \gamma)} \|A(x_t - x_{te})\|^2 \] (27)
and
\[ \|A(x_{t+1} - x_t)\|^2 \leq (1 + \gamma) \|x_{t+1} - x_t\|^2. \] (28)

Then, it follows that
\[ \|y_{te} - Ax_{t+1}\|^2 + \|y_{te} - Ax_t\|^2 \leq \frac{1}{\alpha} \|x_t - x_{te}\|^2 - \frac{1}{\alpha} \|x_{t+1} - x_t\|^2 + \|A(x_{t+1} - x_t)\|^2 \]
\[ \leq \frac{1}{\alpha(1 - \gamma)} \|y_{te} - Ax_t\|^2 + (1 + \gamma) \|x_{t+1} - x_t\|^2 \]
\[ \leq \frac{1}{\alpha(1 - \gamma)} \|y_{te} - Ax_t\|^2, \] (29)
where (a) is satisfied by applying (27) and (28) to (26), and (b) follows from a supplementary assumption that \(\alpha < 1/(1 + \gamma)\).

By moving the second term on the left-hand side of (29) to the right-hand side,
\[ \|y_{te} - Ax_{t+1}\|^2 \leq \left(\frac{1}{\alpha(1 - \gamma)} - 1\right) \|y_{te} - Ax_t\|^2. \] (30)

By repeating (30) for \(t \in \{0, 1, \ldots, T - 1\}\), we get
\[ \|y_{te} - Ax_T\|^2 \leq \left(\frac{1}{\alpha(1 - \gamma)} - 1\right)^T \|y_{te} - Ax_0\|^2. \] (31)

From \((S, 1 - \gamma, 1 + \gamma)\)-RIP condition, we also get
\[ \|x_{te} - x_T\|^2 \leq \frac{1}{1 - \gamma} \|y_{te} - Ax_T\|^2. \] (32)
Applying (31) to (32), we obtain
\[
\|x_{te} - x_T\|^2 \leq \frac{1}{1 - \gamma} \left( \frac{1}{\alpha(1 - \gamma)} - 1 \right)^T \|y_{te} - Ax_0\|^2 \triangleq \epsilon^2. \tag{33}
\]
If we set \(\alpha\) to any constant satisfying \(\frac{1}{2(1 - \gamma)} < \alpha < \frac{1}{(1 - \gamma)}\) and \(\alpha < \frac{1}{(1 + \gamma)}\), the right-hand side of (33) converges to zero when \(T\) is sufficiently large. In other words, if we denote \(v := \left( \frac{1}{\alpha(1 - \gamma)} - 1 \right)^{-1}\), and \(T\) satisfies
\[
T = \log_v \left( \frac{\|y_{te} - Ax_0\|^2}{\epsilon^2(1 - \gamma)} \right) \propto \log \left( \frac{1}{\epsilon} \right), \tag{34}
\]
then we obtain the following equation, thereby completing the proof.
\[
\|x_{te} - x_T\| \leq \epsilon. \tag{35}
\]

C Algorithm Details

**Algorithm 1** CSGM (without blue notes) and CSGM-IM in the case of using DCGAN training objective

**Training phase**
Input: \(A \in \mathbb{R}^{m \times d}, G_\theta : \mathbb{R}^v \mapsto \mathbb{R}^d, D_\phi : \mathbb{R}^d \mapsto [0, 1]\)

\[
\text{for } i = 1 \text{ to } n \text{ do}
\]
\[
\begin{align*}
&\text{sample } x_{tr,i} \text{ from } p(x) \\
&\text{measure } y_{tr,i} = Ax_{tr,i} \\
&\text{sample } z_i \text{ from } p_z(z)
\end{align*}
\]
end for \(\triangleright\) generate \(n\) training samples

\[
(\theta^*, \phi^*) = \arg\min_{\theta} \arg\max_{\phi} \frac{1}{n} \sum_{i=1}^{n} \left[ \ln D_\phi(x_{tr,i}, y_{tr,i}) + \ln (1 - D_\phi(G_\theta(z_i, y_{tr,i}), y_{tr,i})) \right]
\]

Output: the trained parameters \(\theta^*\) of \(G\)

**Test phase**
Input: \(y_{te} \in \mathbb{R}^m, A \in \mathbb{R}^{m \times d}, G_{\theta^*} : \mathbb{R}^v \mapsto \mathbb{R}^d, \tau \in \mathbb{R}^+, T \in \mathbb{N}\)

Initialize: sample \(z_0\) from \(p_z(z)\)

\[
\text{for } t = 0 \text{ to } T - 1 \text{ do}
\]
\[
\begin{align*}
&z_{t+1} = z_t - \frac{\partial}{\partial z} \left. \|y_{te} - A G_{\theta^*}(z, y_{te})\|^2 \right|_{z = z_t}
\end{align*}
\]
end for

Output: the signal estimate \(\hat{x} = G_{\theta^*}(z_T, y_{te})\)
Algorithm 2 CSGM (without blue notes) and CSGM-IM in the case of using BEGAN training objective

Training phase
Input: $A \in \mathbb{R}^{m \times d}$, $G_{\theta}: \mathbb{R}^v \mapsto \mathbb{R}^d$, $D_{\phi}: \mathbb{R}^d \mapsto \mathbb{R}^d$, $R_{\phi}(\bar{x}, y) = |\bar{x} - D_{\phi}(\bar{x}, y)|$, $\lambda \in \mathbb{R}^+$, $\gamma \in [0, 1]$, $\zeta(0) = 0$, $K \in \mathbb{N}$
for $i = 1$ to $n$
do
  sample $x_{tr,i}$ from $p(x)$
  measure $y_{tr,i} = Ax_{tr,i}$
  sample $z^G_i$ and $z^D_i$ independently from $p_z(z)$
end for
\(\triangledown\) generate $n$ training samples
for $k = 0$ to $K - 1$
do
  $\phi(k + 1) = \phi(k) - \eta \frac{\partial}{\partial \phi} \left( \sum_{i=1}^{n} \left( R_{\phi}(x_{tr,i}, y_{tr,i}) - \zeta(k) R_{\phi}(G_{\theta(k)}(z^D_i, y_{tr,i}); y_{tr,i}) \right) \right) \bigg|_{\phi = \phi(k)}$
  $\theta(k + 1) = \theta(k) - \eta \frac{\partial}{\partial \theta} \left( \sum_{i=1}^{n} R_{\phi(k)}(G_{\theta(k)}(z^G_i, y_{tr,i}) - R_{\phi(k)}(G_{\theta(k)}(z^G_i, y_{tr,i}), y_{tr,i})) \right) \bigg|_{\theta = \theta(k)}$
  $\zeta(k + 1) = \min(\max(\zeta(k) + \lambda(\gamma R_{\phi(k)}(x_{tr,i}, y_{tr,i}) - R_{\phi(k)}(G_{\theta(k)}(z^G_i, y_{tr,i}), y_{tr,i})), 0), 1)$
end for
\(\triangledown\) optimize the network parameters
Output: the trained parameter $\theta(K)$ of $G$

Test phase
Input: $y_{te} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times d}$, $G_{\theta(K)}: \mathbb{R}^v \mapsto \mathbb{R}^d$, $\tau \in \mathbb{R}^+$, $T \in \mathbb{N}$
Initialize: sample $z_0$ from $p_z(z)$
for $t = 0$ to $T - 1$
do
  $z_{t+1} = z_t - \tau \frac{\partial}{\partial z} \left\| y_{te} - A G_{\theta(K)}(z, y_{te}) \right\|^2_{z=z_t}$
end for
Output: the signal estimate $\hat{x} = G_{\theta(K)}(z_T, y_{te})$
Algorithm 3 PGGGAN (without blue notes) and PGGGAN-IM in the case of using DCGAN training objective

Training phase:
Input: \( A \in \mathbb{R}^{m \times d}, G_\theta : \mathbb{R}^v \mapsto \mathbb{R}^d, D_\phi : \mathbb{R}^d \mapsto [0, 1] \)

\[
\text{for } i = 1 \text{ to } n \text{ do}
\quad \text{sample } x_{tr,i} \text{ from } p(x)
\quad \text{measure } y_{tr,i} = Ax_{tr,i}
\quad \text{sample } z_i \text{ from } p_z(z)
\text{end for}
\]

\[\sum \frac{1}{n} \left[ \ln D_\phi(x_{tr,i}, y_{tr,i}) + \ln (1 - D_\phi(G_\theta(z_i, y_{tr,i}), y_{tr,i})) \right] \]

Output: the trained parameters \( \theta^* \) of \( G \)

Test phase:
Input: \( y_{te} \in \mathbb{R}^m, A \in \mathbb{R}^{m \times d}, G_{\theta^*} : \mathbb{R}^v \mapsto \mathbb{R}^d, (\alpha, \tau) \in \mathbb{R}_+, (T, K) \in \mathbb{N}^2 \)

Initialize: \( x_0 = 0 \in \mathbb{R}^d \)

\[
\text{for } t = 0 \text{ to } T - 1 \text{ do}
\quad w_t = x_t - \alpha A^\top (Ax_t - y_{te})
\quad \text{sample } z_0 \text{ from } p_z(z)
\quad \text{for } k = 0 \text{ to } K - 1 \text{ do}
\quad \quad z_{k+1} = z_k - \frac{\alpha}{\tau} \left\| w_t - G_{\theta^*}(z, y_{te}) \right\|^2_{z=z_k}
\quad \text{end for}
\quad x_{t+1} = G_{\theta^*}(z_K, y_{te})
\text{end for}
\]

Output: the signal estimate \( \hat{x} = x_T \)
Algorithm 4 DCS (without blue notes) and DCS-IM

Training phase
Input: $A \in \mathbb{R}^{m \times d}$, $G_\theta : \mathbb{R}^v \mapsto \mathbb{R}^d$, $(\tau, \eta, \lambda) \in \mathbb{R}_+^3$, $T \in \mathbb{N}$
Initialize: $\theta(0) = \theta$

for $k = 0$ to $K - 1$ do
  for $i = 1$ to $n$ do
    sample $x_{tr,i}$ from $p(x)$
    measure $y_{tr,i} = Ax_{tr,i}$
    sample $z_{i,0}$ from $p_z(z)$
    for $t = 0$ to $T - 1$ do
      $z_{i,t+1} = z_{i,t} - \tau \frac{\partial}{\partial z} \|y_{tr,i} - AG_\theta(z_{i,t}, y_{tr,i})\|_2^2 |_{z=z_{i,t}}$
    end for
  end for
  $\triangledown$ generate $n$ training samples
  $M(\theta) = \frac{1}{n} \sum_{i=1}^{n} \|y_{tr,i} - AG_\theta(z_{i,T}, y_{tr,i})\|^2$
  $R(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{x_1,x_2 \in \{(x_{tr,i},G_\theta(z_{i,0},y_{tr,i}),G_\theta(z_{i,T},y_{tr,i}))\}} \left( \|A(x_1 - x_2)\| - \|x_1 - x_2\| \right)^2$
  $\theta(k+1) = \theta(k) - \eta \frac{\partial}{\partial \theta} \left( M(\theta) + \lambda R(\theta) \right) |_{\theta=\theta(k)} \triangledown$ optimize the network parameters
end for
Output: the trained parameters $\theta(K)$ of $G$

Test phase
Input: $y_{te} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times d}$, $G_{\theta(K)} : \mathbb{R}^v \mapsto \mathbb{R}^d$, $\tau \in \mathbb{R}_+$, $T \in \mathbb{N}$
Initialize: sample $z_0$ from $p_z(z)$

for $t = 0$ to $T - 1$ do
  $z_{t+1} = z_t - \tau \frac{\partial}{\partial z} \|y_{te} - AG_{\theta(K)}(z_t, y_{te})\|_2^2 |_{z=z_t}$
end for
Output: the signal estimate $\hat{x} = G_{\theta(K)}(z_T, y_{te})$
**Algorithm 5** SparseGen (without blue notes) and SparseGen-IM in the case of using DCGAN training objective

**Training phase**

Input: $A \in \mathbb{R}^{m \times d}$, $G_\theta : \mathbb{R}^v \mapsto \mathbb{R}^d$, $D_\phi : \mathbb{R}^d \mapsto [0, 1]$

for $i = 1$ to $n$

sample $x_{tr,i}$ from $p(x)$

measure $y_{tr,i} = Ax_{tr,i}$

sample $z_i$ from $p_z(z)$

end for

$\triangledown$ generates $n$ training samples

$(\theta^*, \phi^*) = \arg\min_\theta \arg\max_\phi \frac{1}{n} \sum_{i=1}^n [\ln D_\phi(x_{tr,i}, y_{tr,i}) + \ln (1 - D_\phi(G_\theta(z_i, y_{tr,i}), y_{tr,i}))]$  

Output: the trained parameters $\theta^*$ of $G$

**Test phase**

Input: $y_{te} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times d}$, $G_\theta^* : \mathbb{R}^v \mapsto \mathbb{R}^d$, $\tau \in \mathbb{R}_+$, $(L, T) \in \mathbb{N}^2$ where $L < T$

Initialize: sample $z_0$ from $p_z(z)$, $\nu_0 = 0$

for $t = 0$ to $T - 1$

if $t < L$ then

$z_{t+1} = z_t - \tau \frac{\partial}{\partial z} \left( \| A(G_\theta^*(z, y_{te}) + \nu_t) - y_{te} \|^2 + \lambda \| B\nu_t \|_1 \right)_{\|z\|=z_t}$

$\nu_{t+1} = \nu_t$

else

$T = (z, \nu)$

$(z_{t+1}, \nu_{t+1}) = (z_t, \nu_t) - \tau \frac{\partial}{\partial T} \left( \| A(G_\theta^*(z, y_{te}) + \nu) - y_{te} \|^2 + \lambda \| B\nu \|_1 \right)_{T=(z_t, \nu_t)}$

end if

end for

Output: the signal estimate $\hat{x} = G_\theta^*(z_T, y_{te}) + \nu_T$
D Experimental Details

When adding the measurement vector $y$ into the DCGAN architecture, we emulate a conventional architecture suggested in [55] since $y$ can be considered as the text description embedding. In the generator $G_\theta$, $y$ is concatenated to the noise vector $z$. In the discriminator $D_\phi$, $y$ is duplicated spatially and concatenated to the $4 \times 4$ sized image feature maps in a channel-wise manner. To reduce the number of channels to the original number of ones, the concatenated feature maps pass through a $1 \times 1$ convolution followed by a batch normalization and the rectified linear unit function.

When putting $y$ in the BEGAN architecture, $y$ is solely concatenated to latent variables in the generator and the decoder of the discriminator, which is far simpler than the DCGAN architecture supplemented with $y$.

In Figure 1(a), we employ CSGM and CSGM-IM ($m = 1000$) using DCGAN trained on CelebA dataset. Red/blue/green-colored ‘x’ indicates the first/second/third sample in the test set of CelebA. We sample 50 samples from $G_\theta^*(z, y_{te})$ (CSGM-IM) per test sample and 150 samples from $G_\theta^*(z)$ (CSGM). In Figure 1(b), each curve represents the average reconstruction error over 64 random test samples per iteration, where the experimental setting of Figure 1(b) is the same as that of Figure 1(a).

For experiments on PGDGAN and PGDGAN-IM in Section 6, we follow the same experimental setting as [26] such as $\alpha = 0.5$, $\tau = 0.1$, $T = 10$, and $K = 100$.

When conducting experiments on DCS and DCS-IM in Section 6, a sensing matrix $A$ is not learned and the learning rate in the latent optimization ($\tau$ in Algorithm 4) is fixed as 0.01 for brevity. When implementing DCS, $\lambda$ in Algorithm 4 is set to 1.0 like [10] for $m \leq 1000$, but 0.001 for $m > 1000$ because a large value of $R(\theta)$ in Algorithm 4 hinders DCS from performing well. Furthermore, $T = 5$ for $m = 5000$ in DCS, whereas we follow the same hyperparameters ($\lambda = 1.0$, $T = 3$) as [10] for all $m$ in DCS-IM.

When running experiments on SparseGen and SparseGen-IM in Section 6, $\lambda$ in (19) and (20) (or Algorithm 5) is chosen among $\{0.1, 0.5, 1.0\}$ for each $m$. Other than $\lambda$, we make use of the identical hyperparameters ($L = 250$, $T = 500$ in Algorithm 5) as introduced in [18]. Note that we solely consider a transform matrix $B$ in Algorithm 5 as the wavelet basis given that [18] recommend the wavelet basis rather than the discrete cosine transform.

The fastMRI dataset consists of 34,732 training slices, 7,135 validation slices, and 3,903 test slices. As the values of the original slices are complex, we preprocess the data as follows: (i) apply two-dimensional fast Fourier transform to each slice, (ii) crop it at the center to the size $256 \times 256$, (iii) take the absolute value of a complex-valued slice, (iv) subtract 0.5 and divide by 0.5, and (v) downsample it to the size $128 \times 128$. 

24
### E Additional Experimental Results

**Table 4:** Reconstruction error per pixel with 95% confidence interval of 5 trials for Section 4.1 using DCGAN.  
**Table 5:** Reconstruction error per pixel with 95% confidence interval of 5 trials for Section 4.1 using BEGAN.

| m   | CSGM [22]               | CSGM-IM (Ours)         | m   | CSGM [22]               | CSGM-IM (Ours)         |
|-----|-------------------------|------------------------|-----|-------------------------|------------------------|
| 20  | 0.3038 ± 0.0678         | 0.2088 ± 0.0106        | 20  | 0.2125 ± 0.0314         | 0.1855 ± 0.0166        |
| 50  | 0.1859 ± 0.0096         | 0.1131 ± 0.0077        | 50  | 0.1309 ± 0.0218         | 0.1008 ± 0.0114        |
| 100 | 0.1043 ± 0.0116         | 0.0720 ± 0.0069        | 100 | 0.0990 ± 0.0127         | 0.0580 ± 0.0067        |
| 200 | 0.0624 ± 0.0069         | 0.0464 ± 0.0039        | 200 | 0.0805 ± 0.0096         | 0.0331 ± 0.0038        |
| 500 | 0.0392 ± 0.0039         | 0.0286 ± 0.0031        | 500 | 0.0711 ± 0.0111         | 0.0147 ± 0.0022        |
| 1000| 0.0332 ± 0.0031         | 0.0217 ± 0.0025        | 1000| 0.0680 ± 0.0095         | 0.0075 ± 0.0012        |
| 2500| 0.0298 ± 0.0036         | 0.0185 ± 0.0017        | 2500| 0.0669 ± 0.0095         | 0.0031 ± 0.0005        |
| 5000| 0.0285 ± 0.0031         | 0.0183 ± 0.0018        | 5000| 0.0660 ± 0.0109         | 0.0020 ± 0.0003        |

**Table 6:** Reconstruction error per pixel with 95% confidence interval of 5 trials for Section 4.2 using DCGAN.

| m   | PGDGAN [26]             | PGDGAN-IM (Ours)       | m   | DCS [10]                | DCS-IM (Ours)           |
|-----|-------------------------|------------------------|-----|-------------------------|------------------------|
| 20  | 0.6298 ± 0.0757         | 0.4415 ± 0.0205        | 20  | 0.2460 ± 0.0052         | 0.1095 ± 0.0081        |
| 50  | 0.2730 ± 0.0353         | 0.1626 ± 0.0105        | 50  | 0.2190 ± 0.0175         | 0.0694 ± 0.0067        |
| 100 | 0.1281 ± 0.0101         | 0.0942 ± 0.0067        | 100 | 0.1587 ± 0.0072         | 0.0489 ± 0.0053        |
| 200 | 0.0793 ± 0.0065         | 0.0539 ± 0.0046        | 200 | 0.1789 ± 0.0077         | 0.0329 ± 0.0036        |
| 500 | 0.0489 ± 0.0063         | 0.0311 ± 0.0030        | 500 | 0.1136 ± 0.0071         | 0.0173 ± 0.0023        |
| 1000| 0.0383 ± 0.0037         | 0.0234 ± 0.0025        | 1000| 0.0874 ± 0.0032         | 0.0099 ± 0.0014        |
| 2500| 0.0337 ± 0.0035         | 0.0195 ± 0.0017        | 2500| 0.0854 ± 0.0055         | 0.0036 ± 0.0005        |
| 5000| 0.0322 ± 0.0035         | 0.0194 ± 0.0018        | 5000| 0.0817 ± 0.0024         | 0.0024 ± 0.0003        |

**Table 7:** Reconstruction error per pixel with 95% confidence interval of 5 trials for Section A.1 only using the generator of DCGAN.

**Table 8:** Reconstruction error per pixel with 95% confidence interval of 5 trials for Section A.2 using DCGAN.

| m   | SparseGen [18]          | SparseGen-IM (Ours)    | m   | DCS [10]                | DCS-IM (Ours)           |
|-----|-------------------------|------------------------|-----|-------------------------|------------------------|
| 20  | 0.3743 ± 0.0598         | 0.2242 ± 0.0218        | 20  | 0.2460 ± 0.0052         | 0.1095 ± 0.0081        |
| 50  | 0.2125 ± 0.0390         | 0.1121 ± 0.0103        | 50  | 0.2190 ± 0.0175         | 0.0694 ± 0.0067        |
| 100 | 0.1184 ± 0.0180         | 0.0720 ± 0.0054        | 100 | 0.1587 ± 0.0072         | 0.0489 ± 0.0053        |
| 200 | 0.0691 ± 0.0158         | 0.0470 ± 0.0037        | 200 | 0.1789 ± 0.0077         | 0.0329 ± 0.0036        |
| 500 | 0.0378 ± 0.0050         | 0.0283 ± 0.0031        | 500 | 0.1136 ± 0.0071         | 0.0173 ± 0.0023        |
| 1000| 0.0302 ± 0.0040         | 0.0213 ± 0.0024        | 1000| 0.0874 ± 0.0032         | 0.0099 ± 0.0014        |
| 2500| 0.0248 ± 0.0028         | 0.0178 ± 0.0016        | 2500| 0.0854 ± 0.0055         | 0.0036 ± 0.0005        |
| 5000| 0.0238 ± 0.0025         | 0.0174 ± 0.0017        | 5000| 0.0817 ± 0.0024         | 0.0024 ± 0.0003        |