DIFFERENTIABLE GAUSSIANIZATION LAYERS FOR INVERSE PROBLEMS REGULARIZED BY DEEP GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep generative models such as GANs, normalizing flows, and diffusion models are powerful regularizers for inverse problems. They exhibit great potential for helping reduce ill-posedness and attain high-quality results. However, the latent tensors of such deep generative models can fall out of the desired high-dimensional standard Gaussian distribution during an inversion process, particularly in the presence of data noise and inaccurate forward models. In such cases, deep generative models are ineffective in attaining high-fidelity solutions. To address this issue, we propose to reparameterize and Gaussianize the latent tensors using novel differentiable data-dependent layers wherein custom operators are defined by solving optimization problems. These proposed layers constrain inverse problems to obtain high-fidelity in-distribution solutions. We tested and validated our technique on three inversion tasks: compressive-sensing MRI, image deblurring, and eikonal tomography (a nonlinear PDE-constrained inverse problem), using two representative deep generative models: StyleGAN2 and Glow, and achieved state-of-the-art performance in terms of accuracy and consistency.

1 INTRODUCTION

Inverse problems are ubiquitous and crucial to solve in scientific endeavors and our daily life. For example, astrophysicists use radio electromagnetic data to image galaxies and black holes (Högbom, 1974; Akiyama et al., 2019). Geoscientists use seismic recordings to reveal the internal structures of Earth (Tarantola, 1984; Tromp et al., 2005; Virieux & Operto, 2009). Biomedical engineers and doctors use X-ray projections, ultrasound measurements, and magnetic resonance data to reconstruct images of human tissues and organs (Lauterbur, 1973; Gemmeke & Ruiter, 2007; Lustig et al., 2007).

Solving an inverse problem starts with the definition of a forward mapping from parameters $\mathbf{m}$ to data $\mathbf{d}$, which we formally write as

$$\mathbf{d} = f(\mathbf{m}) + \boldsymbol{\epsilon}, \tag{1}$$

where f stands for a forward model that usually describes some physical process, $\boldsymbol{\epsilon}$ denotes noise, $\mathbf{d}$ the observed data, and $\mathbf{m}$ the parameters to be estimated. The forward model can be either linear or nonlinear and either explicit or implicitly defined by solving partial differential equations (PDEs). This study considers three representative inverse problems: **Compressive Sensing MRI**, **Deblurring**, and **Eikonal (traveltime) Tomography**, which have important applications in medical science, geoscience, and astronomy. The details of each problem and its forward model are in App. A.

The forward problem maps $\mathbf{m}$ to $\mathbf{d}$, while the inverse problem estimates $\mathbf{m}$ given $\mathbf{d}$. Unfortunately, inverse problems are generally under-determined with infinitely many compatible solutions and intrinsically ill-posed because of the nature of the physical system. Worse still, the observed data are usually noisy, and the assumed forward model might be inaccurate, exacerbating the ill-posedness. These challenges require using regularization to inject *a priori* knowledge into inversion processes to obtain plausible and high-fidelity results. Therefore, an inverse problem is usually posed as an optimization problem:

$$\arg\min_{\mathbf{m}} \ (1/2) \left\| \mathbf{d} - f\left(\mathbf{m}\right) \right\|_2^2 + \mathcal{R}(\mathbf{m}), \tag{2}$$

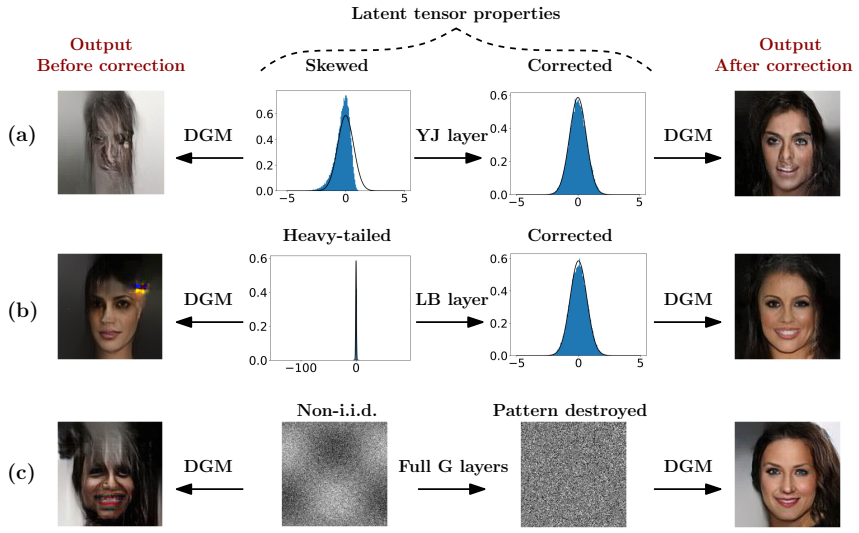

Figure 1: The comparison of images generated by a deep generative model (DGM) – Glow using latent tensors that deviate from a spherical Gaussian distribution and those after corresponding corrections. The visual effects highlight the necessity of keeping the latent tensor within such a distribution during inversion. The second column shows the characteristics of deviated latent tensors: (a) histogram: i.i.d. components but the distribution is skewed; (b) histogram: i.i.d. components but the distribution is heavy-tailed; (c) latent tensor image: non-i.i.d. entries. The first column shows the corresponding outputs of a Glow network. The third column shows latent tensors corrected by (a) the Yeo-Johnson layer (YJ), (b) the Lambert $W \times F_X$ layer (LB), and (c) the full set of our Gaussianization layers (G layers). Those corrected latent tensors map to realistic images shown in the fourth column. All latent tensors have a norm of $0.7\sqrt{\text{vec dim}}$ because of the Gaussian Annulus Theorem (App. I) and the fact that Glow works best with a temperature smaller than one (see Fig. 9). Additional examples for **StyleGAN2** and **Stable Diffusion** (Rombach et al., 2022) can be found in App. B.

where $\mathcal{R}(\mathbf{m})$ is the regularization term. Beyond traditional regularization methods such as the Tikhonov regularization and Total Variation (TV) regularization, deep generative models (DGM), such as VAEs (Kingma & Welling, 2013), GANs (Goodfellow et al., 2014), and normalizing flows (Dinh et al., 2014; 2016; Kingma et al., 2016; Papamakarios et al., 2017; Marinescu et al., 2020), have shown great potential for regularizing inverse problems (Bora et al., 2017; Van Veen et al., 2018; Hand et al., 2018; Ongie et al., 2020; Asim et al., 2020; Mosser et al., 2020; Li et al., 2021; Siahkoohi et al., 2021; Whang et al., 2021). Such deep generative models directly learn from training data distributions and are a powerful and versatile prior. They map latent vectors $\mathbf{z}$ to outputs $\mathbf{m}$ distributed according to an *a priori* distribution: $\mathbf{m} = g(\mathbf{z}) \sim p_{\texttt{target}}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, for example. The framework of DGM-regularized inversion (Bora et al., 2017) is

$$\underset{\mathbf{z}}{\arg\min}\ (1/2)\left\|\mathbf{d} - f \circ g\left(\mathbf{z}\right)\right\|_2^2 + \mathcal{R}'(\mathbf{z}), \tag{3}$$

where the deep generative model g, whose layers are frozen, reparameterizes the original variable $\mathbf{m}$, acting as a hard constraint. Instead of optimizing for $\mathbf{m}$, we now estimate the latent variable $\mathbf{z}$ and retrieve the inverted $\mathbf{m}$ by forward mappings. Since the latent distribution is usually a standard Gaussian, the new (optional) regularization term $\mathcal{R}'(\mathbf{z})$ can be chosen as $\beta\|\mathbf{z}\|_2^2$ for GANs and VAEs, where β is a weighting factor. See App. J.1 for more details on a similar formulation for normalizing flows. Since the optimal β depends on the problem and data, tuning β is highly subjective and costly.

However, this formulation of DGM-regularized inversion still leads to unsatisfactory results if the data are noisy or the forward model is inaccurate, as shown in Fig. 20, even if we fine-tune the weighting parameter β.

To analyze this problem, first recall that a well-trained DGM has a latent space (usually) defined on a standard Gaussian distribution. In other words, a DGM either only sees standard Gaussian latent

tensors during training (*e.g.*, GANs) or learns to establish 1-1 mappings between training examples and typical samples from the standard Gaussian distribution (*e.g.*, normalizing flows, App. J.2). As a result, the generator may map out-of-distribution latent vectors to unrealistic results. We show in Fig. 1 the visual effects of several types of deviations of latent vectors from a spherical Gaussian (with a temperature of 0.7). It can be seen that (1) the latent tensor should have i.i.d. entries, and (2) these entries should be distributed as a 1D standard Gaussian in order to generate plausible images. Since the traditional DGM-regularized inversion lacks such Gaussianity constraint, we conjecture[1] that the latent tensor deviates from the desired high-dimensional standard Gaussian distribution during inversion, leading to poor results. Our observations and reasoning motivate us to propose a set of differentiable Gaussianization layers to reparameterize and Gaussianize the latent vectors of deep generative models (*e.g.*, StyleGAN2 (Karras et al., 2020) and Glow (Kingma & Dhariwal, 2018)) for inverse problems.

2 METHOD

2.1 OVERVIEW - REPARAMETERIZATION USING GAUSSIANIZATION LAYERS

Our solution is based on a necessary condition of the standard Gaussian prior on latent tensors. Let us define a *partition* of a latent tensor $\mathbf{z} \in \mathbb{R}^n$ as the collection of non-overlapping patches $P(\mathbf{z}) = \{\mathbf{z}_i\}_{i=1,\cdots,N}$, where the patches $\mathbf{z}_i \in \mathbb{R}^D$ are of the same dimension and can be assembled as $\mathbf{z}$, *i.e.*, $n = N \times D$. If the latent tensor $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then for all $\mathbf{z}_i$ from any partition $P(\mathbf{z})$, we have $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Note that $\mathbf{z}$ is a symbolic representation of the latent tensor. In a specific DGM, $\mathbf{z}$ can be either a 2D/3D tensor or a list of such tensors corresponding to a multi-scale architecture (App. D). Even though there are numerous partition schemes, such as random grouping of components, we choose the simplest: partitioning $\mathbf{z}$ based on neighboring components, which works well in practice.

To constrain $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ during inversion, we reparameterize $\mathbf{z}_i$ by constructing a mapping $h : \mathbf{v}_i \to \mathbf{z}_i$, such that $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The new variables $\mathbf{v}_i$ are of the same dimension as $\mathbf{z}_i$. Suppose that we have constructed the patch-level mapping h, we can obtain a mapping $h^\dagger$ at the tensor level, such that $\mathbf{z} = h^\dagger(\mathbf{v})$, where $\mathbf{v}$ is assembled by $\mathbf{v}_i$ in the same way as $\mathbf{z}$ by $\mathbf{z}_i$. For example, $h^\dagger = \texttt{diag}(h, \cdots, h)$ if patches are extracted from neighboring components and are concatenated into a vectorized $\mathbf{v}$. Hence, the original DGM-regularized inversion 3 becomes

$$\arg\min_{\mathbf{v}} \; (1/2) \left\| \mathbf{d} - f \circ g \circ h^\dagger(\mathbf{v}) \right\|_2^2 . \tag{4}$$

Since we are imposing a constraint through reparameterization, there is no need to include the regularization term $\mathcal{R}'(\mathbf{z})$. The new formulation is still an unconstrained optimization problem, enabling us to use highly efficient unconstrained optimizers, such as L-BFGS (Nocedal & Wright, 2006) and ADAM (Kingma & Ba, 2015).

The remaining critical piece is to construct h, and it leads to our Gaussianization layers. First, we translate the constraint of $\mathbf{z}_i = h(\mathbf{v}_i) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ into the following optimization problem:

$$\arg\min_{h} D_{\mathrm{KL}} \left(p_Z \left(h \left(\mathbf{v}_i \right) \right) \| \mathcal{N}(\mathbf{0}, \mathbf{I}) \right) , \tag{5}$$

where p_Z is the probability density function (PDF) of $\mathbf{z}_i$. Second, we adopt the framework proposed by precursor works of normalizing flows on Gaussianization (Chen & Gopinath, 2000; Laparra et al., 2011; Meng et al., 2020) to solve this optimization problem. The KL-divergence can be decomposed as the sum of the multi-information $I(\mathbf{z}_i)$ and the marginal negentropy $J_m(\mathbf{z}_i)$ (Chen & Gopinath, 2000):

$$D_{\mathrm{KL}} \left(p_Z \left(\mathbf{z}_i \right) \| \mathcal{N} \left(\mathbf{0}, \mathbf{I} \right) \right) = I(\mathbf{z}_i) + J_m(\mathbf{z}_i), \tag{6}$$

where

$$I(\mathbf{z}_i) = D_{\mathrm{KL}} \left(p_Z \left(\mathbf{z}_i \right) \Big\| \prod_j^D p_j(z_i^{(j)}) \right) , \text{ and } J_m(\mathbf{z}_i) = \sum_{j=1}^D D_{\mathrm{KL}} \left(p_j(z_i^{(j)}) \Big\| \mathcal{N}(0,1) \right) . \tag{7}$$

[1] Out-of-distribution/typicality tests are very challenging for high-dimensional data (Rabanser et al., 2019; Nalisnick et al., 2019). Also, since latent tensors are Gaussian/Gaussian-like, it is hard to conduct dimension reduction for such tests.

Here $z_i^{(j)}$ denotes the j-th component of patch vector $\mathbf{z}_i$, and p_j stands for the marginal PDF for that component. The multi-information $I(\mathbf{z}_i)$ quantifies the independence of the components of $\mathbf{z}_i$, while the marginal negentropy $J_m(\mathbf{z}_i)$ describes how close each component is to a 1D standard Gaussian. With this decomposition, the optimization procedure depends on the facts: (1) the KL divergence and a standard Gaussian in Eq. 6 are invariant to an orthogonal transformation, and (2) the multi-information term is invariant to a component-wise invertible differentiable transformation (App. J.3). As a result, we perform Gaussianization in two steps:

1. Minimize the multi-information $I(\mathbf{z}_i)$. This is done by an orthogonal transformation that keeps the overall KL divergence the same but increases the negentropy $J_m(\mathbf{z}_i)$. We achieve this by using our independent component analysis (ICA) layer. ICA is the optimal choice since it maximizes non-Gaussianity so that the subsequent marginal Gaussianization step removes it and results in a large decrease in $D_{\mathrm{KL}}\left(p_Z\left(h\left(\mathbf{v}_i\right)\right) \| \mathcal{N}(\mathbf{0}, \mathbf{I})\right)$.

2. Minimize the marginal negentropy $J_m(\mathbf{z}_i)$ by component-wise operations that perform 1D Gaussianization of marginal distributions $p_{j,\, j=1,\cdots,D}$. The multi-information does not change under component-wise invertible operations. Therefore, the overall KL divergence between $\mathbf{z}_i$ and the Gaussian distribution decreases.

The Gaussianization steps are well-aligned with the motivating example (Fig. 1). To constrain DGM outputs to be plausible, one should make components within latent tensor patches independent (or destroy the patterns) (Fig. 1(c)) and shape the 1D distribution as close as possible to Gaussian (Fig. 1(a)(b)).

We will see that h is parameterized by an orthogonal matrix and two scalar parameters in 1D Gaussianization layers. Unlike conventional neural network layers, the input-data-dependent layers are not defined by learning from a dataset ahead of time but by solving certain optimization problems on the job (Fig. 18). Special care should be taken to implement the gradient computation correctly and ensure that they pass the finite-difference convergence test (App. G). As an overview, the composition of our proposed layers is:

$$\mathbf{v} \rightarrow \textbf{Whitening} \rightarrow \textbf{ICA} \rightarrow \textbf{Yeo-Johnson} \rightarrow \textbf{Lambert}\ W \times F_X \rightarrow \textbf{Standardization} \rightarrow \mathbf{z},$$

where whitening and ICA belong to the first step and the rest belong to the second step. We will discuss in App. K some possible simplifications of the layers in practice after our ablation studies.

The overall proposed inversion process (one iteration) is illustrated in Fig. 2.

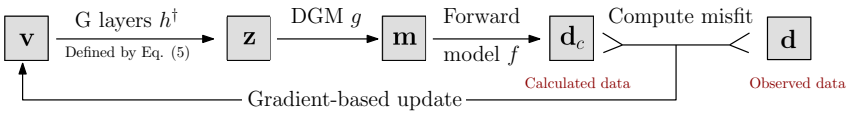

Figure 2: Illustration of the proposed inversion process.

2.2 Reducing multi-information – ICA layer

The orthogonal matrix $\boldsymbol{W}$ is constructed by the independent component analysis (ICA). The input is the patch vectors $\{\mathbf{v}_i|_{i=1,\cdots,N}\}$. The ICA algorithm first computes the input-dependent orthogonal matrix $\boldsymbol{W}$ and then computes $\mathbf{p}_i = \boldsymbol{W}^\top \mathbf{v}_{i,\, i=1,\cdots,N}$ as the output. The orthogonal matrix $\boldsymbol{W}$ makes the entries of each $\mathbf{p}_i$ independent random variables.

We use the FastICA algorithm (Hyvarinen, 1999; Hyvärinen & Oja, 2000), which employs a fixed-point algorithm to maximize a contrast function Φ (*e.g.*, the logcosh function) for our ICA layer to reduce multi-information.

The FastICA algorithm typically requires that the data are pre-whitened. We adopt the ZCA whitening method or the iterative whitening method introduced in Hyvarinen (1999) (App. E). With the whitened data, we compute $\boldsymbol{W}$ using a damped fixed-point iteration scheme:

$$\boldsymbol{W} = \frac{1}{N}\left[\alpha \boldsymbol{V} \phi\left(\boldsymbol{W}^\top \boldsymbol{V}\right)^\top - \boldsymbol{W}\,\mathtt{diag}\left(\phi'\left(\boldsymbol{W}^\top \boldsymbol{V}\right)\mathbf{1}\right)\right], \tag{8}$$

where $\mathbf{1}$ is an all-one vector, the column vectors of $\boldsymbol{V}$ are $\{\mathbf{v}_i|_{i=1,\cdots,N}\}$, $\phi(\cdot) = \Phi'(\cdot)$, $\alpha \in (0, 1)$, and we use $\alpha = 0.8$ throughout our experiments. To save computation time, we only perform a maximum of 10 iterations. The details of the whole algorithm can be found in App. E.

We set the initial $\boldsymbol{W}$ as an identity matrix. If the input vectors are already standard Gaussian (in-distribution), the computed $\boldsymbol{W}$ will still be an identity matrix, which maps the input to the same output. In practice, the empirical distribution from finite samples is not a standard Gaussian, so $\boldsymbol{W}$ is not an identity matrix but another orthogonal matrix, which still maps standard Gaussian input vectors to standard Gaussian vectors as output. For this reason, we sample starting $\{\mathbf{v}_i|_{i=1,\cdots,N}\}$ from the standard Gaussian distribution to start inversions with plausible outputs.

2.3 Reducing marginal negentropy

For 1D Gaussianization, we choose a combination of the Yeo-Johnson transformation that reduces skewness and the Lambert $W \times F_X$ transformation that reduces heavy-tailedness. Both are layers based on optimization problems with only one parameter, which is cheap to compute and is easy to back-propagate the gradient. Eq. 7 requires us to perform such 1D transformations for each component of the random vectors. In other words, we need to solve the same optimization problem for D times, which imposes a substantial computational burden. Instead, we empirically find it acceptable to share the same optimization-generated parameter across all components. In other words, we perform only a single 1D Gaussianization, treating all entry values in the latent vector as the data simultaneously.

Power transformation layer We propose to use the power transformation or Yeo-Johnson transformation (Yeo & Johnson, 2000) to reduce the skewness of distributions. As shown in Fig. 3(a), the form of the Yeo-Johnson activation function depends on parameter λ. If $\lambda = 1$, the mapping is an identity mapping. If $\lambda > 1$, the activation function is convex, compressing the left tail and extending the right tail, reducing the left-skewness. If $\lambda < 1$, the activation function is concave, which oppositely reduces the right-skewness. We refer the readers to App. E for details.

Lambert $W \times F_X$ layer Due to noise and inaccurate forward models, we observe that the distribution of latent vector values tends to be shaped as a heavy-tailed distribution during the inversion process. To reduce the heavy-tailedness, we adopt the Lambert $W \times F_X$ method detailed in Goerg (2015).

We use the parameterized Lambert $W \times F_X$ distribution family to approximate a heavy-tailed input and solve an optimization to estimate an optimal parameter δ (App. E), with which the inverse transformation maps the heavy-tailed distribution towards a Gaussian.

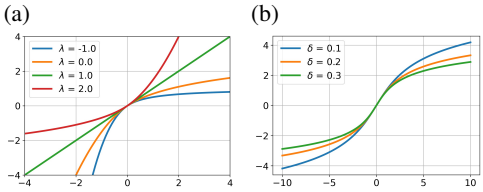

Figure 3: The nonlinear activation functions from (a) the power transformation (Yeo-Johnson) layer and (b) the Lambert $W \times F_X$ layer.

Fig. 3(b) shows that the Lambert $W \times F_X$ layer acts as a nonlinear squashing function. As δ increases, it compresses more the large values and reduces the heavy-tailedness. Intuitively, the Lambert $W \times F_X$ layer can also be interpreted as an intelligent way of imposing constraints on the range of values instead of a simple box constraint. We refer the readers to App. E for more details about the optimization problem and implementation.

Standardization with temperature Since the Lambert $W \times F_X$ layer output may not necessarily have a zero mean and a unit (or a prescribed) variance, we standardize the output using

$$\boldsymbol{z} = (\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])/\sqrt{\mathrm{Var}(\boldsymbol{x})} * \gamma, \tag{9}$$

where γ is the temperature parameter suggested in Kingma & Dhariwal (2018).

3 RELATED WORK

End-to-end NNs for inverse problems There are vast amounts of end-to-end neural networks designed for inverse problems, using CNNs (Chen et al., 2017; Jin et al., 2017; Sriram et al., 2020), GANs (Mardani et al., 2018; Lugmayr et al., 2020; Wang et al., 2018), invertible networks (Ardizzone et al., 2018), and diffusion models (Kawar et al., 2022). The general idea is simple: train a neural network that directly maps observed data to estimated parameters. Even though such methods seem effective in a few applications, DGM-regularized inversion with our Gaussianization layers is preferable for the following reasons. First, the forward modeling can be so expensive computationally that it is infeasible to collect a decent training datasets for some applications. For example, one large-scale fluid mechanics or wave propagation simulation can take hours, if not days. Second, the relationship between parameters and data can be highly nonlinear, and multiple solutions may exist. An end-to-end network may map data to interpolated solutions that are not realistic. In comparison, our method can start from different initializations or even employ sampling techniques to address this issue. Third, the configuration of data collection can change from experiment to experiment. It is unbearable, for example, to re-train the network each time we change the number and locations of sensors. In contrast, not only can our method deal with this situation, but it can even use the same DGM for different forward models, as we can see in the compressive sensing MRI and eikonal tomography examples. Almost all end-to-end methods are only applied to linear inverse problems, whereas our Gaussianization layers are also effective in nonlinear problems.

Other techniques to improve DGM-regularized inversion In high-dimensional space, the probability mass of a standard Gaussian distribution concentrates within the so-called Gaussian typical set (App. I). To be in the Gaussian typical set, one necessary but not sufficient condition is to be within an annulus area around a high-dimensional sphere (App. I). Utilizing this geometric property, DGM-regularized inversion methods like Bojanowski et al. (2017) and Liang et al. (2021) force updated latent vectors to stay on the sphere. This strategy is closely related to spherical interpolation (White, 2016). We call this strategy the **spherical constraint** for inversion. In the original StyleGAN2 paper, the authors also noticed that in image projection tasks, the noise maps tend to have leakage from signals (Karras et al., 2020) – the same phenomenon we discussed. They proposed a multi-scale noise regularization term (**NoiseRlg**) to penalize spatial correlation in noise maps. We extend the same technique to our inverse problems for comparison. Note that we use a whitening layer before the ICA layer. The **whitening layer** can be used alone, similar to Huang et al. (2018) and Siarohin et al. (2018), whose performance will be reported in ablation studies. Also, Wulff & Torralba (2020) observed that for StyleGAN2, the Leaky ReLU function can "Gaussianize" latent vectors in the W space. **CSGM-w** (Kelkar & Anastasio, 2021) utilizes this *a priori* knowledge to improve DGM-regularized compressive sensing problems. Recently, score-based generative models have been shown to be promising for solving inverse problems (Jalal et al., 2021; Song et al., 2021). Currently, they seem only applicable to linear inverse problems. In addition, we propose an alternative idea to ensure latent vectors are within the standard Gaussian distribution. We reparameterize latent vectors using learnable orthogonal matrices (Cayley parameterization) and keep initial latent vectors fixed, which is closely related to the work of orthogonal over-parameterized training (Liu et al., 2021). We refer readers to App. H for details on this technique.

4 EXPERIMENTS

We consider three representative inversion problems for testing: compressive sensing MRI, image deblurring, and eikonal traveltime tomography. For MRI and eikonal tomography, we used synthetic brain images as inversion targets and used the pre-trained StyleGAN2 weights from Kelkar & Anastasio (2021) (trained on data from the databases of fastMRI (Zbontar et al., 2018; Knoll et al., 2020), TCIA-GBM (Scarpace et al., 2016), and OASIS-3 (LaMontagne et al., 2019)) for regularization. We used the test split of the CelebA-HQ dataset (Karras et al., 2018) for deblurring, and the DGM is a Glow network trained on the training split. We refer readers to App. F for details on datasets and training.

We tested each parameter configuration in each inversion on 100 images (25 in the eikonal tomography due to its expensive forward modeling). Since the deep generative models are highly nonlinear, the results may get stuck in local minima. Thus, we started inversion using three different randomly

initialized latent tensors for each of the 100 or 25 images, picked the best value among the three for each metric, and reported the mean and standard deviation of those metrics, except for CSGM-w, TV, and NoiseRlg, where the initialization is fixed. The metrics we used were PSNR, SSIM (Wang et al., 2004), and an additional LPIPS (Zhang et al., 2018) for the CelebA-HQ data. We used the LBFGS (Nocedal & Wright, 2006) optimizer in all experiments except TV, noise regularization, and CSGM-w, which use FISTA (Beck & Teboulle, 2009) or ADAM (Kingma & Ba, 2015). The temperature was set to 1.0 for StyleGAN2 and 0.7 for Glow.

4.1 Compressive sensing MRI using StyleGAN2

The mathematical model of compressive sensing MRI is

$$\mathbf{d} = \boldsymbol{A}\mathbf{m} + \boldsymbol{\epsilon}, \tag{10}$$

where $\boldsymbol{A} \in \mathbb{C}^{M \times N}$ is the sensing matrix, which consists of FFT and subsampling in the k-space (frequency domain). Eq. 10 is an under-determined system, and we use Accl $= N/M$ to denote the acceleration ratio. We also added i.i.d. Gaussian noise with a signal-to-noise ratio (SNR) of 20 dB or 10 dB to the measured data. See App. A.1 for more background information.

Table 1 summarizes the comparison of results from total variation regularization (TV), noise regularization (NoiseRlg) (Karras et al., 2020), spherical constraint/reparameterization: $\boldsymbol{z} = \boldsymbol{v}/\|\boldsymbol{v}\|_2 * \sqrt{\texttt{dim}(\boldsymbol{v})}$, CSGM-w (Kelkar & Anastasio, 2021), our proposed orthogonal reparameterization (Orthogonal), and our proposed Gaussianization layers (G layers). Fig. 4 shows examples of inversion results. In the base case where Accl=8x and SNR=20 dB, the Gaussianization layers gives the best scores. Their advantage gets more significant when data SNR decreases to 10 dB. We also notice that scores from all methods are greatly improved if we make the system better determined (*i.e.*, Accl=2x). Also, the performance of TV, spherical constraint, and Gaussianization layers are similar in this scenario. We conclude that our proposed Gaussianization layers are effective and more robust than other methods.

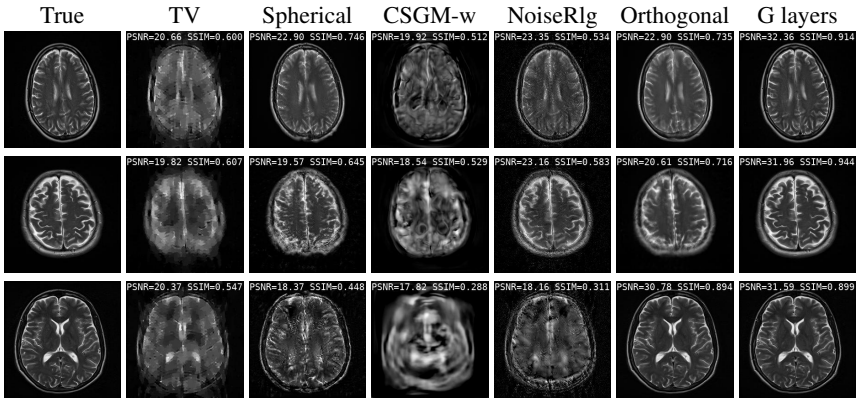

Figure 4: Comparison of compressive sensing MRI inversion results (Accl=8x, SNR=20 dB).

Table 1: Comparison of compressive sensing MRI results.

Method	Accl = 8x, SNR = 20 dB		Accl = 8x, SNR = 10 dB		Accl = 2x, SNR = 20 dB	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
TV	20.20±1.33	0.60±0.063	19.78±1.32	0.57±0.058	32.92±1.36	0.91±0.024
Spherical	26.93±6.57	0.79±0.159	22.78±5.38	0.61±0.246	32.90±3.50	0.90±0.061
CSGM-w	20.19±2.11	0.57±0.096	19.98±1.94	0.55±0.091	27.53±2.98	0.74±0.085
NoiseRgl	21.61±2.27	0.50±0.073	18.09±1.05	0.27±0.036	28.04±1.46	0.66±0.037
Orthogonal	26.14±5.79	0.78±0.134	25.10±5.08	0.77±0.132	29.29±5.32	0.84±0.108
G layers	27.99±5.70	0.83±0.128	25.48±4.76	0.78±0.149	32.41±4.60	0.90±0.079

Ablation study Table 2 summarizes the ablation study on the components of the Gaussianization layers. We kept the standardization layer on for all cases. The conclusions are as follows: 1. The ICA layer is the most significant part in improving result scores; 2. The whitening/ZCA layer alone is not effective; 3. The Yeo-Johnson (YJ) and the Lambert $W \times F_X$ (Lambt) layers are more effective when the noise level is higher (*e.g.*, SNR=10 dB vs. 20 dB). Their performance seems data-dependent: when SNR=20 dB, YJ seems more effective, while Lambt gives the best scores when SNR=10 dB. But the overall improvement from these two layers is marginal. In practice, one may only use ICA and one of the 1D Gaussianization layers. Additionally, we tested the effect of patch size on the z (style) vectors (App. C). We find that the largest possible patch size gives the best results.

Table 2: Ablation study of the Gaussianization layers.

Method			Accl = 8x, SNR = 20 dB		Accl = 8x, SNR = 10 dB	
			PSNR↑	SSIM↑	PSNR↑	SSIM↑
ICA (✗),	YJ (✗),	Lambt (✗)	26.93±6.40	0.787±0.153	22.98±5.54	0.604±0.252
ICA (✗),	YJ (✓),	Lambt (✗)	26.92±6.42	0.787±0.156	23.14±5.44	0.622±0.245
ICA (✗),	YJ (✗),	Lambt (✓)	25.33±5.89	0.743±0.163	23.58±5.19	0.695±0.189
ZCA,	YJ (✗),	Lambt (✗)	27.08±6.52	0.786±0.157	22.94±5.50	0.623±0.235
ICA (✓),	YJ (✗),	Lambt (✗)	27.91±5.77	0.824±0.129	25.22±4.76	0.770±0.154
ICA (✓),	YJ (✓),	Lambt (✗)	27.99±5.70	0.831±0.128	25.48±4.76	0.779±0.149
ICA (✓),	YJ (✗),	Lambt (✓)	27.21±5.74	0.816±0.125	25.57±4.98	0.779±0.148
ICA (✓),	YJ (✓),	Lambt (✓)	27.37±5.90	0.819±0.135	25.09±4.91	0.771±0.144

4.2 Image deblurring using Glow

The mathematical model for deblurring is

$$\mathbf{d} = \boldsymbol{H} * \mathbf{m} + \boldsymbol{\epsilon}, \tag{11}$$

where $\boldsymbol{H}$ is a smoothing filter and $*$ denotes convolution. We used a Gaussian smoothing filter with a standard deviation of 3, and added noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, 50^2\mathbf{I})$ to the observed data. Although the system may not be under-determined, the high-frequency information is lost due to low-pass filtering; hence this is also an ill-posed problem. Though we only tested on facial images, deblurring has wide applications in astronomy and geophysics. See App. A.2 for more background information.

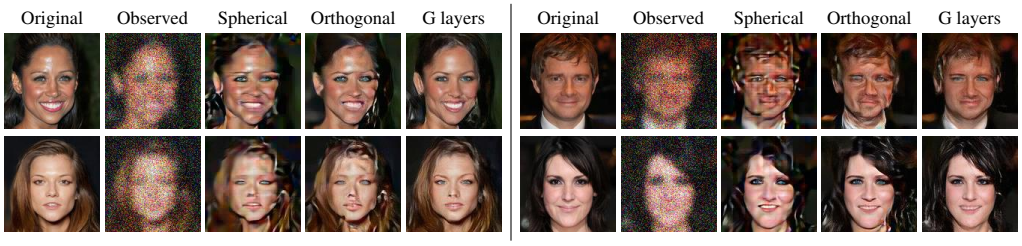

Figure 5: Comparison of deblurring results from different methods.

Table 3 and Fig. 5 show that the Gaussianization layers are also effective in Glow, better than using the spherical constraint or the orthogonal reparameterization. We also demonstrate the efficacy of Gaussianization layers when the forward model is inaccurate, in which case the induced error in data is not Gaussian (App. C).

Ablation study Table 3 also shows that the two parameterization schemes (App. D) for Glow using the Gaussianization layers have similar performance. Besides, we report the ablation study on the components of the Gaussianizaiton layers for Glow (App. C).

4.3 EIKONAL TOMOGRAPHY USING STYLEGAN2

In acoustic wave imaging (e.g., ultrasound tomography), we excite waves using sparsely distributed sources one at a time at the boundary of the object. Then we reconstruct its internal structures (the spatial distribution of wave speed) given the first-arrival travel time recorded on the boundary. The following eikonal equation approximately describes the shortest travel time $T(\boldsymbol{x}; \boldsymbol{x}_s)$ that the acoustic wave emerging from the source location $\boldsymbol{x}_s$ takes to reach location $\boldsymbol{x}$ inside the target object (Yilmaz, 2001):

$$|\nabla T(\boldsymbol{x}; \boldsymbol{x}_s)| = 1/c(\boldsymbol{x}), \quad T(\boldsymbol{x}_s; \boldsymbol{x}_s) = 0, \tag{12}$$

where $c(\boldsymbol{x})$ is the wave propagation speed at each location. Both this eikonal PDE and the implicitly defined forward mapping $c(\boldsymbol{x}) \to T(\boldsymbol{x})$ are nonlinear, and there has been little research on DGM-regularized inverse problems with such nonlinear characteristics. The inverse problem is severely ill-posed, which is equivalent to a curved-ray tomography problem. See App. A.3 for more background information. We added noise to the recorded travel time using the following formula: $T_{\text{noisy}}(\boldsymbol{x}_r; \boldsymbol{x}_s) = T(\boldsymbol{x}_r; \boldsymbol{x}_s)(1 + \epsilon)$, where $\epsilon \sim \mathcal{N}(0, 0.001^2)$ and $\boldsymbol{x}_r$ denotes any receiver location. In other words, longer traveltime corresponds to larger uncertainties.

We show that the Gaussianization layers outperformed other methods in this tomography task in Table 4 and Fig. 6. Note that this is a statistical conclusion. We also report an example where the spherical constraint works better than the Gaussianization layers (bottom right).

Table 3: Deblurring results using Glow.

Method	LPIPS↓	PSNR↑	SSIM↑
Spherical	0.17±0.06	21.78±1.14	0.580±0.066
Orthogonal	0.16±0.06	21.91±1.31	0.583±0.063
G layers P1	0.13±0.05	22.40±1.34	0.583±0.069
G layers P2	0.13±0.05	22.47±1.27	0.590±0.064

Table 4: Eikonal tomography using StyleGAN2

Method	PSNR↑	SSIM↑
TV	20.66±1.21	0.543±0.073
Spherical	22.19±4.05	0.686±0.144
G layers	24.80±2.55	0.803±0.093

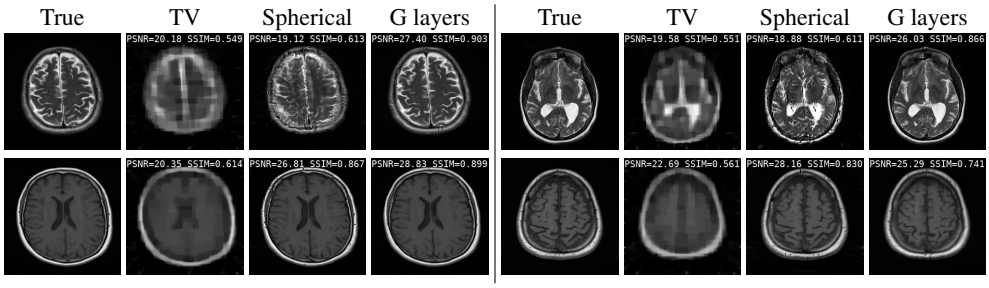

Figure 6: Comparison of eikonal tomography results from different methods.

5 DISCUSSION AND CONCLUSIONS

We provide insights on the experiment results and further discuss this work's limitations, computational cost, and broader impact in App. K.

In summary, we have identified a critical problem in DGM-regularized inversion: the latent tensor can deviate from a typical example from the desired high-dimensional standard Gaussian distribution, leading to unsatisfactory inversion results. To address the problem, we have introduced the differentiable Gaussianization layers that reparameterize and Gaussianize latent tensors so that the inversion results remain plausible. In general, our method has achieved the best scores in terms of mean values and standard deviations compared with other methods, demonstrating our method's advantages and high performance in terms of accuracy and consistency. (Regarding the standard deviation of scores, we only compare with other methods with competitive mean scores, such as orthogonal reparameterization and spherical constraint.) Our proposed layers are plug-and-play, require minimal parameter tuning, and can be applied to various deep generative models and inverse problems.

REFERENCES

Kazunori Akiyama, Antxon Alberdi, Walter Alef, Keiichi Asada, Rebecca Azulay, Anne-Kathrin Baczko, David Ball, Mislav Baloković, John Barrett, Dan Bintley, et al. First m87 event horizon telescope results. iv. imaging the central supermassive black hole. *The Astrophysical Journal Letters*, 875(1):L4, 2019.

Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pp. 399–409. PMLR, 2020.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of data science*. Cambridge University Press, 2020.

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pp. 537–546. PMLR, 2017.

Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

Hu Chen, Yi Zhang, Mannudeep K Kalra, Feng Lin, Yang Chen, Peixi Liao, Jiliu Zhou, and Ge Wang. Low-dose ct with a residual encoder-decoder convolutional neural network. *IEEE transactions on medical imaging*, 36(12):2524–2535, 2017.

Scott Chen and Ramesh Gopinath. Gaussianization. *Advances in neural information processing systems*, 13:423–429, 2000.

Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. On the lambertw function. *Advances in Computational mathematics*, 5(1):329–359, 1996.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. Wiley, 2012. ISBN 9781118585771. URL `https://books.google.com/books?id=VWq5GG6ycxMC`.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

H Gemmeke and NV Ruiter. 3d ultrasound computer tomography for medical imaging. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 580(2):1057–1065, 2007.

Georg M Goerg. The lambert way to gaussianize heavy-tailed data with the inverse of tukey'sh transformation as a special case. *The Scientific World Journal*, 2015, 2015.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3012–3021, 2020.

Paul Hand, Oscar Leong, and Vladislav Voroninski. Phase retrieval under a generative prior. *arXiv preprint arXiv:1807.04261*, 2018.

JA Högbom. Aperture synthesis with a non-regular distribution of interferometer baselines. *Astronomy and Astrophysics Supplement Series*, 15:417, 1974.

Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 791–800, 2018.

Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, 10(3):626–634, 1999.

Aapo Hyvärinen. The fixed-point algorithm and maximum likelihood estimation for independent component analysis. *Neural Processing Letters*, 10(1):1–5, 1999.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

Ajil Jalal, Marius Arvinte, Giannis Daras, Eric Price, Alexandros G Dimakis, and Jon Tamir. Robust compressed sensing mri with deep generative priors. *Advances in Neural Information Processing Systems*, 34:14938–14954, 2021.

Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9): 4509–4522, 2017.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. 2018.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *arXiv preprint arXiv:2201.11793*, 2022.

Varun A Kelkar and Mark Anastasio. Prior image-constrained reconstruction using style-based generative models. In *International Conference on Machine Learning*, pp. 5367–5377. PMLR, 2021.

Diederik Kingma and Jimmy Ba. Adam, a method for stochastic optimization. 2015.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Florian Knoll, Jure Zbontar, Anuroop Sriram, Matthew J Muckley, Mary Bruno, Aaron Defazio, Marc Parente, Krzysztof J Geras, Joe Katsnelson, Hersh Chandarana, et al. fastmri: A publicly available raw k-space and dicom dataset of knee images for accelerated mr image reconstruction using machine learning. *Radiology: Artificial Intelligence*, 2(1):e190007, 2020.

Pamela J LaMontagne, Tammie LS Benzinger, John C Morris, Sarah Keefe, Russ Hornbeck, Chengjie Xiong, Elizabeth Grant, Jason Hassenstab, Krista Moulder, Andrei G Vlassenko, et al. Oasis-3: longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and alzheimer disease. *MedRxiv*, 2019.

Valero Laparra, Gustavo Camps-Valls, and Jesús Malo. Iterative gaussianization: from ica to random rotations. *IEEE transactions on neural networks*, 22(4):537–549, 2011.

Paul C Lauterbur. Image formation by induced local interactions: examples employing nuclear magnetic resonance. *nature*, 242(5394):190–191, 1973.

Dongzhuo Li, Kailai Xu, Jerry M Harris, and Eric Darve. Coupled time-lapse full-waveform inversion for subsurface flow problems using intrusive automatic differentiation. *Water Resources Research*, 56(8):e2019WR027032, 2020.

Dongzhuo Li, Huseyin Denli, Cody MacDonald, Kyle Basler-Reeder, Anatoly Baumstein, and Jacquelyn Daves. Multiparameter geophysical reservoir characterization augmented by generative networks. In *First International Meeting for Applied Geoscience & Energy*, pp. 1364–1368. Society of Exploration Geophysicists, 2021.

Jingyun Liang, Kai Zhang, Shuhang Gu, Luc Van Gool, and Radu Timofte. Flow-based kernel prior with application to blind super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10601–10610, 2021.

LR Lines and S Treitel. A review of least-squares inversion and its application to geophysical problems. *Geophysical prospecting*, 32(2):159–186, 1984.

Weiyang Liu, Rongmei Lin, Zhen Liu, James M Rehg, Liam Paull, Li Xiong, Le Song, and Adrian Weller. Orthogonal over-parameterized training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7251–7260, 2021.

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.

Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *European Conference on Computer Vision*, pp. 715–732. Springer, 2020.

Michael Lustig, David Donoho, and John M Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

Morteza Mardani, Enhao Gong, Joseph Y Cheng, Shreyas S Vasanawala, Greg Zaharchuk, Lei Xing, and John M Pauly. Deep generative adversarial neural networks for compressive sensing mri. *IEEE transactions on medical imaging*, 38(1):167–179, 2018.

Razvan V Marinescu, Daniel Moyer, and Polina Golland. Bayesian image reconstruction using deep generative models. *arXiv preprint arXiv:2012.04567*, 2020.

Chenlin Meng, Yang Song, Jiaming Song, and Stefano Ermon. Gaussianization flows. In *International Conference on Artificial Intelligence and Statistics*, pp. 4336–4345. PMLR, 2020.

Lukas Mosser, Olivier Dubrule, and Martin J Blunt. Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. *Mathematical Geosciences*, 52(1):53–79, 2020.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *ICLR*, 2019. URL `https://openreview.net/forum?id=H1xwNhCcYm`.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Erkki Oja and Zhijian Yuan. The fastica algorithm revisited: Convergence analysis. *IEEE transactions on Neural Networks*, 17(6):1370–1381, 2006.

Gregory Ongie, Ajil Jalal, Christopher A Metzler, Richard G Baraniuk, Alexandros G Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

Lisa Scarpace, L Mikkelsen, T Cha, Sujaya Rao, Sangeeta Tekchandani, S Gutman, and D Pierce. Radiology data from the cancer genome atlas glioblastoma multiforme [tcga-gbm] collection. *The Cancer Imaging Archive*, 11(4):1, 2016.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9243–9252, 2020.

Ali Siahkoohi, Gabrio Rizzuti, Mathias Louboutin, Philipp A Witte, and Felix J Herrmann. Preconditioned training of normalizing flows for variational inference in inverse problems. *arXiv preprint arXiv:2101.03709*, 2021.

Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and coloring batch transform for gans. In *International Conference on Learning Representations*, 2018.

Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. *arXiv preprint arXiv:2111.08005*, 2021.

Anuroop Sriram, Jure Zbontar, Tullie Murrell, Aaron Defazio, C Lawrence Zitnick, Nafissa Yakubova, Florian Knoll, and Patricia Johnson. End-to-end variational networks for accelerated mri reconstruction. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 64–73. Springer, 2020.

Jean-Luc Starck, Eric Pantin, and Fionn Murtagh. Deconvolution in astronomy: A review. *Publications of the Astronomical Society of the Pacific*, 114(800):1051, 2002.

Albert Tarantola. Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, 49 (8):1259–1266, 1984.

Jeroen Tromp, Carl Tape, and Qinya Liu. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels. *Geophysical Journal International*, 160(1):195–216, 2005.

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

Jean Virieux and Stéphane Operto. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26, 2009.

Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Jay Whang, Qi Lei, and Alex Dimakis. Solving inverse problems with a flow-based noise model. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11146–11157. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/whang21a.html`.

Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Jonas Wulff and Antonio Torralba. Improving inversion and generation diversity in stylegan using a gaussianized latent space. *arXiv preprint arXiv:2009.06529*, 2020.

In-Kwon Yeo and Richard A Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000.

Özdoğan Yilmaz. *Seismic data analysis*, volume 1. Society of exploration geophysicists Tulsa, 2001.

Jure Zbontar, Florian Knoll, Anuroop Sriram, Tullie Murrell, Zhengnan Huang, Matthew J Muckley, Aaron Defazio, Ruben Stern, Patricia Johnson, Mary Bruno, et al. fastmri: An open dataset and benchmarks for accelerated mri. *arXiv preprint arXiv:1811.08839*, 2018.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. 2018.

Rui Zhang and John Castagna. Seismic sparse-layer reflectivity inversion using basis pursuit decomposition. *Geophysics*, 76(6):R147–R158, 2011.

Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250): 603–627, 2005.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

A BACKGROUND OF FORWARD MODELS

A.1 COMPRESSIVE SENSING MRI

The MRI process essentially samples the spatial frequency components of a target, following some trajectories in the spatial frequency space (k-space) according to the design of the physical system. For example, a system may sample the k-space line-by-line horizontally/vertically or in radial directions. If the k-space has been fully sampled on a Cartesian grid, one can directly use inverse FFT to reconstruct the image. For various practical reasons, however, it is necessary to speed up the data collection process, usually by skipping data points in the k-space, which can be mathematically represented by a masking operation. In addition, there can be multiple coils with different sensitivity maps collecting data simultaneously. The mathematical formulation reads

$$\mathbf{d}_i = \boldsymbol{P}\boldsymbol{F}\boldsymbol{S}_i\mathbf{m} + \boldsymbol{\epsilon}, \quad i = 1, \cdots, N_{\text{coils}} \tag{13}$$

where $\mathbf{d}_i \in \mathbb{C}^N$ is the k-space data corresponding to the i-th coil, $\mathbf{m} \in \mathbb{R}^N$ is the target object, $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ is the mask, $\boldsymbol{F} \in \mathbb{C}^{N\times N}$ is the Fourier transform operator, $\boldsymbol{S}_i \in \mathbb{C}^{N\times N}$ is the point-wise sensitivity map (a diagonal matrix) corresponding to the i-th coil, and $\boldsymbol{\epsilon}$ denotes noise.

To ensure a fair comparison with prior work and reproducibility, we used the same masks from the repository of Kelkar & Anastasio (2021) (Fig. 7). In addition, we also used the same single-coil setup as in Kelkar & Anastasio (2021), where the sensitivity matrix is an identity matrix.

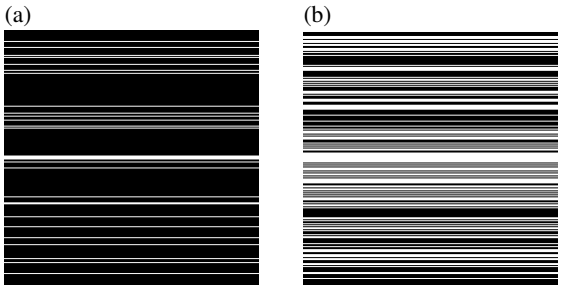

Figure 7: Masks for compressive sensing MRI (Kelkar & Anastasio, 2021). White: 1, black: 0. (a) $8\times$ acceleration; (b) $2\times$ acceleration.

We condense the effects of all operators into a linear operator $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ and arrive at the underdetermined system

$$\mathbf{d} = \boldsymbol{A}\mathbf{m} + \boldsymbol{\epsilon} \tag{10}$$

from the main text, where we use Accl $= N/M$ to denote the acceleration ratio.

A.2 DEBLURRING

The mathematical model behind deblurring is

$$\mathbf{d} = \boldsymbol{H} * \mathbf{m} + \boldsymbol{\epsilon}, \tag{14}$$

where $\boldsymbol{H}$ is a smoothing filter, $*$ denotes convolution, and $\boldsymbol{\epsilon}$ is noise.

The purpose of deburring is to recover the original sharp image $\mathbf{m}$ given a noisy blurred observation $\mathbf{d}$. In this study, we showed deblurring examples for natural images. In scientific applications, deblurring or deconvolution is also a powerful tool. For example, in astronomy, $\mathbf{d}$ is a blurred image from a telescope, $\boldsymbol{H}$ is a point-spread function (PSF) constructed from the physics model of the telescope, and we want to obtain a sharper image from the observation (Starck et al., 2002). In geophysics, $\mathbf{d}$ can be the seismic data, $\boldsymbol{H}$ is a calibrated wavelet, and we want to obtain sharp images of reflectivities defining the boundaries of subsurface strata (Lines & Treitel, 1984; Zhang & Castagna, 2011). In general, $\boldsymbol{H}$ is a low-pass or band-pass filter, so certain frequency contents are lost in the forward process. The deblurring or deconvolution process needs to recover such missing information. In addition, the noise makes the inversion process unstable. The deblurring or deconvolution problem is thus ill-posed.

A.3 EIKONAL TOMOGRAPHY

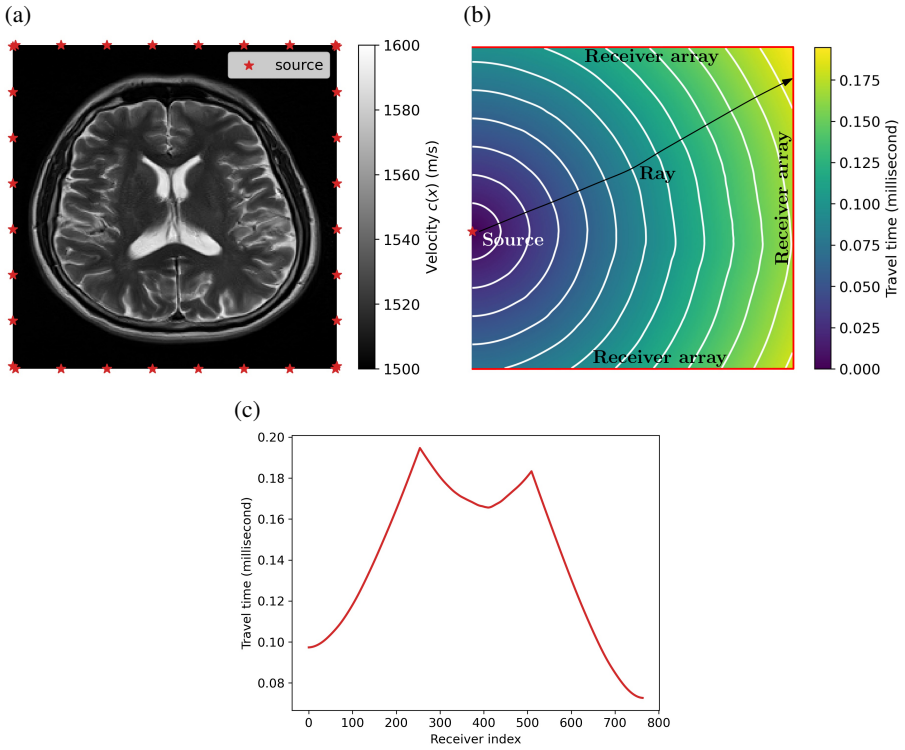

Figure 8: Experimental setup for eikonal tomography. (a) The target image and source locations; (b) Travel time field and the receiver array corresponding to the source. The contours are wavefronts. The propagation of waves can be viewed as curved rays traveling in directions perpendicular to such wavefronts; (c) The profile of the shortest wave travel time recorded at the receiver array. The receiver index starts from 0 in the top-left corner in subfigure (b) and increases in the clockwise direction.

In acoustic wave imaging (e.g., ultrasound tomography), we excite waves using sparsely distributed sources one at a time at the boundary of the object. Then we reconstruct its internal structures (the spatial distribution of wave speed) given the first-arrival travel time recorded on the boundary. The following eikonal equation describes the shortest travel time $T(\boldsymbol{x}; \boldsymbol{x}_s)$ that the acoustic wave emerging from the source location $\boldsymbol{x}_s$ takes to reach location $\boldsymbol{x}$ inside the target object in the high-frequency

limit (Yilmaz, 2001):

$$|\nabla T(\boldsymbol{x};\boldsymbol{x}_s)| = 1/c(\boldsymbol{x}), \quad T(\boldsymbol{x}_s;\boldsymbol{x}_s) = 0, \tag{15}$$

where $c(\boldsymbol{x})$ is the wave propagation speed at each location. Both this eikonal PDE and the implicitly defined forward mapping $c(\boldsymbol{x}) \to T(\boldsymbol{x})$ are nonlinear.

To be more specific, we show the setup of our experiment in Fig. 8(a). For the convenience of numerical testing, we put sources and receivers on the boundaries of a square box that contains the object, suggesting that the object is immersed in a square box filled with water. In reality, one can put sources and receivers directly on the target. The dimension of the square area is 25.6 cm $\times$ 25.6 cm with a grid interval of 0.001 m. There are eight sources located on each side, and receivers are located at each grid point on the boundary. Fig. 8(b) shows the shortest travel time field of the generated wave from the indicated source location. For each source, we only use receivers on the three other sides, indicated by the red lines, excluding the one on which the source is located. Fig. 8(c) shows the profile of the shortest wave travel time recorded at the receiver array. The receiver index starts from 0 in the top-left corner and increases in the clockwise direction.

We can interpret the nonlinearity and ill-posedness of eikonal tomography from another perspective. As shown in Fig. 8(b), we plot contours $T(\boldsymbol{x}) = \texttt{const}$ representing the wavefronts. Under the high-frequency approximation, the propagation of waves can be viewed as curved rays traveling in directions perpendicular to such wavefronts. Since the wavefronts depend on velocity field $c(\boldsymbol{x})$, the rays are also functionals of the parameter $c(\boldsymbol{x})$ to be estimated, contrary to straight-ray tomography such as CT. The eikonal inversion problem is thus nonlinear. Besides, the rays carry information about the medium averaged along their paths. One property of curved-ray tomography is that the ray coverage is uneven inside the object. In fact, curved rays tend to avoid low-velocity areas, giving us little information about such regions, making the inverse problem intrinsically ill-posed.

We solved the eikonal equation using the fast sweeping method (Zhao, 2005) and computed the gradient using the discrete adjoint-state method, using the code from the repository of Li et al. (2020).

Our eikonoal tomography uses the same StyleGAN2 network as the compressive sensing MRI experiments. The value range of StyleGAN2 is $[-1, 1]$. In the forward model, we map the StyleGAN2 output to $c(\boldsymbol{x})$ in two steps. First, we convert its values to the range of $[0, 1]$ by using $\mathbf{m} \leftarrow (\mathbf{m}+1)/2$. Second, we convert the values to the range of acoustic wave velocity using $\mathbf{m} \leftarrow 100 \times \mathbf{m} + 1500$. This relationship is purely manufactured for our synthetic tests. One should use a more realistic relationship in practice.

B ADDITIONAL MOTIVATING EXAMPLES

Glow We varied the ℓ_2-norm of the latent tensors for Glow and reported the outputs in Fig. 9. The images are getting increasingly unrealistic as we increase the norm from $1.0\sqrt{d}$, where d is the latent tensor dimension. This phenomenon demonstrates why we need to use a temperature < 1 for Glow.

StyleGAN2 StyleGAN2 is a well-designed generator with a built-in spherical transformation for the style vector. It seems very robust to random vectors drawn from the distributions similar to those in Fig. 1. However, we were able to find challenging examples for it from inversion tasks. In Fig. 10, The first example is the direct output only using the pathologic style vector. The second example is the generator output using the style vector after whitening. As we can see, whitening alone is ineffective in improving the image quality, which is confirmed by our ablation studies. On the contrary, if we use the ICA layer, we see a huge improvement in visual quality, and our interpretation is that getting rid of higher-order dependencies is very important. Note that there are eyeglasses in the third figure, meaning that if we relax the 1D Gaussian requirement, we can sample some rare examples. Finally, we use the full G layers to get another plausible image.

Stable diffusion Stable Diffusion is a state-of-the-art deep generative model with text-to-image synthesis capability, which maps a Gaussian latent tensor to a high-resolution image.

We repeated the tests for Glow and StyleGAN2 on Stable Diffusion and reported the results in Fig. 11. The distribution for skewed case was the exponential-gamma distribution $\log \Gamma(1, 1)$ (or the "log-gamma distribution" in scipy), the distribution for the heavy-tailed case was a Lambert $W \times F_X$ distribution with parameter $\delta = 0.5$ based on a standard Gaussian (Goerg, 2015). To generate the

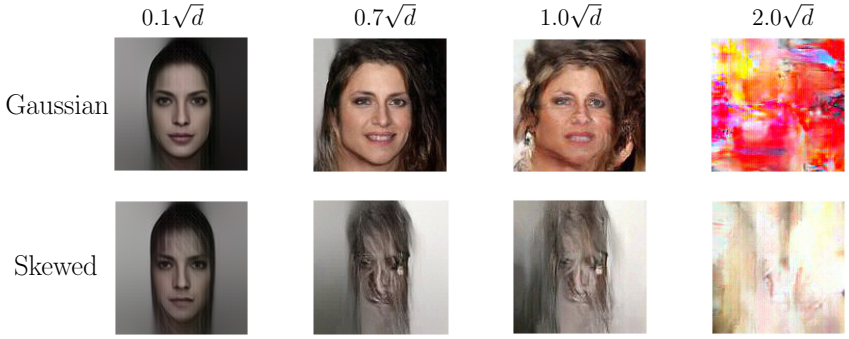

Figure 9: The visual effects of the ℓ_2 norm of latent tensors on Glow outputs. The top and bottom panels correspond to latent tensors drawn from a spherical Gaussian and a spherical skewed distribution, respectively. The two tensors are scaled to have various norms: 0.1, 0.7, 1.0, and 2.0 of $\sqrt{d}$ (the square root of the latent tensor dimension). The images are getting increasingly unrealistic as we increase the norm from $1.0\sqrt{d}$. This phenomenon demonstrates why we need to use a temperature < 1 for Glow.

Figure 10: The visual effects of various components of Gaussianization layers on a pathologic style vector for StyleGAN2.

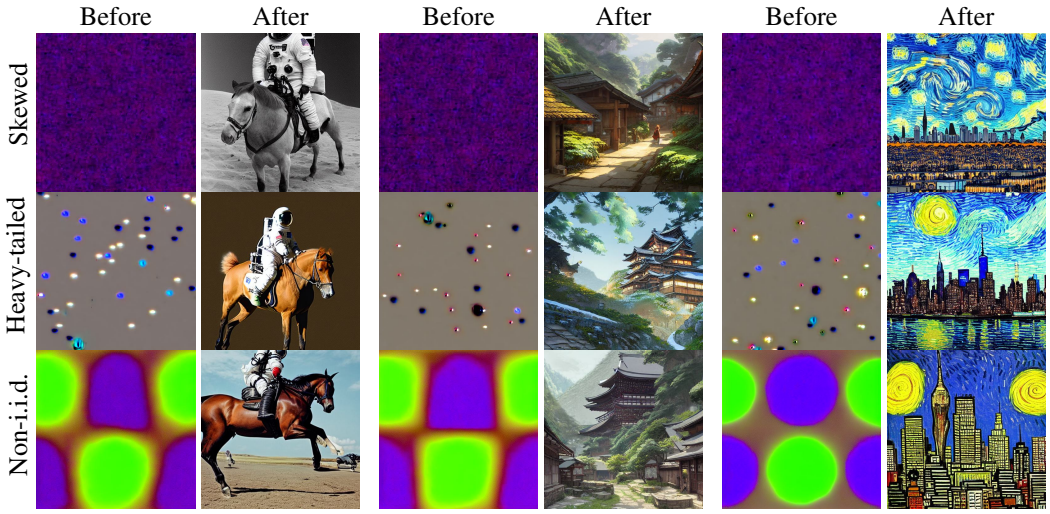

Figure 11: Comparisons of Stable Diffusion outputs before and after applying the Gaussianization layers to latent tensors. The components of these latent tensors are i.i.d. skewed, i.i.d. heavy-tailed, and non-i.i.d., respectively. We scaled all latent tensors to be on the sphere with a radius of $\sqrt{\text{tensor dimension}}$. Stable Diffusion picks up and amplifies the sparse large-amplitude points in the heavy-tailed case and the 2D sinusoidal pattern in the non-i.i.d. case.

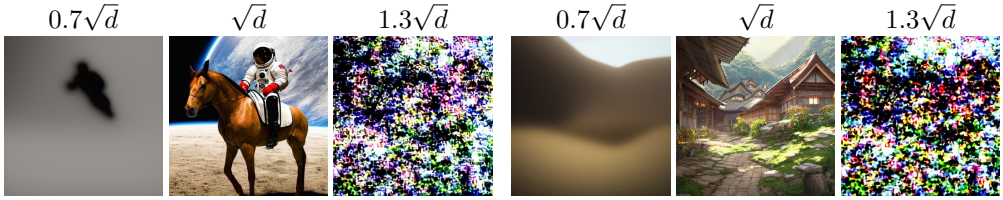

Figure 12: Comparison of Stable Diffusion outputs generated using spherical Gaussian latent tensors of different norms, where $d = \sqrt{\text{tensor dimension}}$.

latent tensor with non-i.i.d. entries, we first sampled $\mathbf{z}_0 \in \mathbb{R}^{4\times64\times64} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then we added $0.5 * \sin(2\pi/64\, x) \otimes \cos(2\pi/64\, y)$ to each channel of $\mathbf{z}_0$ for the latent tensor $\mathbf{z}$, where $\otimes$ stands for outer product. The number of denoising steps, classifier-free guidance, and the output dimensions for Stable Diffusion are 50, 7.5, and 512×512, respectively.

The outputs of Stable Diffusion are completely out-of-range if the latent tensors deviate from standard Gaussian, with much poorer quality than those from Glow and StyleGAN2. For example, Stable Diffusion picks up and amplifies the sparse large-amplitude points in the heavy-tailed case and the 2D sinusoidal pattern in the non-i.i.d. case. However, with our Gaussianization layers, we are still able to constrain the output within the plausible range. Interestingly, we show in Fig. 12 that Stable Diffusion is much more sensitive to the norm of latent tensors than the other DGMs. Therefore, we anticipate that our Gaussianization layers are critical for potential inversion applications using Stable Diffusion such as text-guided inversion.

C ADDITIONAL EXPERIMENTS

First, we report in Table 5 an ablation study on the effects of various combinations of regularization techniques on style vectors and noise maps in StyleGAN2. The observations are (1) If we turn off the update of noise maps, the Gaussianization layers lead to the best result; (2) If we turn on the update of noise maps, applying the Gaussianization layers to both the style vectors and noise maps gives the best results.

Table 5: Ablation study on different combinations of regularization on the style vectors and noise maps in StyleGAN2 for compressive sensing MRI. In the parentheses, "u" stands for unconstrained, "s" stands for spherical constraint, "g" stands for reparameterization with the Gaussianization layers, "o" stands for orthogonal reparameterization, and "✗" means that the update is turned off. Note that StyleGAN2 has a built-in spherical transformation layer for the style vectors.

Method	Accl = 8x, SNR = 20 dB		Accl = 8x, SNR = 10 dB	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Style (s), Noise (✗)	26.58±5.90	0.80±0.137	25.57±5.24	0.78±0.137
Style (g), Noise (✗)	27.39±5.26	0.83±0.124	27.02±4.57	0.82±0.122
Style (o), Noise (✗)	25.88±5.74	0.78±0.134	25.39±5.09	0.77±0.131
Style (s), Noise (u)	26.51±6.51	0.77±0.163	22.49±5.62	0.57±0.266
Style (g), Noise (o)	27.43±6.04	0.81±0.134	24.93±4.27	0.76±0.128
Style (o), Noise (o)	26.14±5.79	0.78±0.134	25.10±5.08	0.77±0.132
Style (o), Noise (g)	26.04±5.70	0.78±0.134	25.22±5.12	0.77±0.133
Style (g), Noise (g)	27.99±5.70	0.83±0.128	25.48±4.76	0.78±0.149

Second, we studied the effect of patch size on the performance of Gaussianization layers. To be more specific, we tested the effects of various patch sizes on 1D style vectors. The largest patch size is 64 since the number of extracted patches should not be smaller than the dimension of the patches. We only turned on the ICA layer and the standardization layer in this experiment. One can observe that both the PSNR and the SSIM increase as the patch size increases. We would advise using the

largest possible patch size in Gaussianization layers, although we only used a patch size of 32 for style vectors in all other experiments.

In all experiments, we fixed the patch size for Glow as $3 \times 8 \times 8$ and the patch size for the noise maps in StyleGAN2 as $1 \times 8 \times 8$. In the experiments with Glow, the image dimension was $3 \times 128 \times 128$. If we chose a larger patch size, we could not obtain enough patch vectors. As for StyleGAN2, if we look at the parameterization illustrated in Fig. 14, there is already a 4×4 noise map unconstrained if we choose the patch size as 8×8. If we increase the patch size to $1 \times 16 \times 16$, there will be two additional 8×8 noise maps unconstrained. To minimize the influence of unconstrained parameters, we chose the patch size for noise maps as $1 \times 8 \times 8$.

Table 6: Ablation study on patch size of the style vectors in StyleGAN2.

Method	PSNR↑	SSIM↑
Patch size = 8	25.33$\pm$6.42	0.76$\pm$0.155
Patch size = 16	26.83$\pm$6.57	0.79$\pm$0.144
Patch size = 32	27.91$\pm$5.77	0.82$\pm$0.129
Patch size = 64	28.03$\pm$5.44	0.83$\pm$0.130

Third, we did a parameter sweep on the weighting parameter β in the Glow-regularized deblurring problem (formulation 3 or 44). Consistent with Fig. 20, the conventional Glow-based regularization is not as effective as our Gaussianization layers. If β is too large (*e.g.*, 10 or 100), the inversion results become unrealistic with huge errors.

Table 7: Glow-regularized deblurring results using different βs.

Method	LPIPS↓	PSNR↑	SSIM↑
$\beta = 0.0$	0.172$\pm$0.06	21.74$\pm$1.27	0.58$\pm$0.072
$\beta = 0.01$	0.172$\pm$0.06	21.75$\pm$1.15	0.58$\pm$0.067
$\beta = 0.1$	0.174$\pm$0.06	21.72$\pm$1.15	0.57$\pm$0.069
$\beta = 1.0$	0.174$\pm$0.06	21.95$\pm$1.21	0.59$\pm$0.067
$\beta = 10.0$	0.247$\pm$0.08	22.55$\pm$1.10	0.58$\pm$0.086
$\beta = 100.0$	0.602$\pm$0.15	12.56$\pm$2.63	0.12$\pm$0.101

Fourth, we investigated the performance of Gaussianization layers when the forward model is inaccurate (Table 8). Our Gaussianization layers still outperform the conventional Glow-regularized inversion.

Table 8: Glow-regularized deblurring with an inaccurate filter. The ground-truth standard deviation of the Gaussian filter is 3, which is used to generate the observed data, but we used 5 for inversion.

Method	LPIPS↓	PSNR↑	SSIM↑
Conventional ($\beta = 1.0$)	0.21$\pm$0.06	17.72$\pm$1.41	0.49$\pm$0.059
G layers	0.17$\pm$0.06	18.70$\pm$1.40	0.50$\pm$0.060

In addition, we did an ablation study on the components of Gaussianization layers in Glow-regularized deblurring (Table 9). We adopted the second parameterization scheme (App. D). The observations are similar to those from Table 2: 1. The ICA layer is the most significant part in improving result scores, especially LPIPS – the score that matches human perception the best; 2. The whitening/ZCA layer is not effective when used alone; 3. There is no clear winner among combinations of 1D Gaussianization layers, and their difference in performance is marginal. So we picked the Lambert $W \times F_X$ layer, one of the best-performing options, in all other deblurring experiments as the 1D Gaussianization layer.

Table 9: Ablation study of the components of Gaussianization layers applied to Glow-regularized deblurring.

Method			LPIPS↓	PSNR↑	SSIM↑
ICA (✗),	YJ (✗),	Lambt (✗)	$0.17_{\pm 0.06}$	$21.79_{\pm 1.11}$	$0.583_{\pm 0.065}$
ICA (✗),	YJ (✗),	Lambt (✓)	$0.16_{\pm 0.06}$	$21.96_{\pm 1.31}$	$0.588_{\pm 0.065}$
ICA (✗),	YJ (✓),	Lambt (✗)	$0.17_{\pm 0.06}$	$21.75_{\pm 1.25}$	$0.579_{\pm 0.067}$
ICA (✗),	YJ (✓),	Lambt (✓)	$0.16_{\pm 0.06}$	$21.99_{\pm 1.20}$	$0.587_{\pm 0.066}$
ZCA,	YJ (✗),	Lambt (✗)	$0.16_{\pm 0.05}$	$21.66_{\pm 1.28}$	$0.579_{\pm 0.070}$
ICA (✓),	YJ (✗),	Lambt (✗)	$0.14_{\pm 0.06}$	$22.52_{\pm 1.30}$	$0.586_{\pm 0.070}$
ICA (✓),	YJ (✗),	Lambt (✓)	$0.13_{\pm 0.05}$	$22.47_{\pm 1.27}$	$0.590_{\pm 0.064}$
ICA (✓),	YJ (✓),	Lambt (✗)	$0.14_{\pm 0.05}$	$22.49_{\pm 1.35}$	$0.586_{\pm 0.070}$
ICA (✓),	YJ (✓),	Lambt (✓)	$0.13_{\pm 0.06}$	$22.40_{\pm 1.27}$	$0.586_{\pm 0.066}$

Inversion using out-of-distribution images Finally, we tested the limit of DGM-regularized inversion by using real-world out-of-distribution target images. We randomly sampled 25 MRI images from the fastMRI DICOM dataset (Zbontar et al., 2018; Knoll et al., 2020). The RSNA clinical trial processor was used to anonymize the whole dataset, and each image has been manually inspected to make sure that there is no protected information. According to the fastMRI paper (Zbontar et al., 2018), "This dataset represents a larger variety of machines and settings than are present in the raw data." Also, the image values are represented in `uint16` rather than float numbers, and their value range can be very large (0 to several thousand). So we normalized the image values to be between range $[-1, 1]$ using `img = (img - img.min())/(img.max()-img.min())*2 - 1`. However, the trained StyleGAN2 outputs are always slightly beyond $[-1, 1]$, and we clipped the values in both test image generation and inversion. Therefore, the new examples are out-of-distribution.

We kept the same experimental setup of $8\times$ acceleration and 20 dB SNR and tested inversion using the spherical constraint and Gaussianization layers on both only using the style vectors and using the additional noise maps. We report the inversion results in Table 10 and Fig. 13. To save experiment time, we only use one 1D Gaussianization layer (either Yeo-Johnson or Lambert $W \times F_X$).

Table 10: MRI inversion using real-world out-of-distribution images.

Method	PSNR↑	SSIM↑
(a) Style only, spherical	$26.14_{\pm 3.59}$	$0.784_{\pm 0.062}$
(b) Style+noise, spherical	$25.26_{\pm 3.37}$	$0.755_{\pm 0.065}$
(c) Style only, G layers, YJ for 1D	$25.23_{\pm 3.30}$	$0.765_{\pm 0.050}$
(d) Style + noise, G layers, YJ for 1D	$25.39_{\pm 3.28}$	$0.769_{\pm 0.045}$
(e) Style only, G layers, Lambert for 1D	$25.39_{\pm 3.38}$	$0.762_{\pm 0.064}$
(f) Style + noise, G layers, Lambert for 1D	$25.45_{\pm 3.39}$	$0.765_{\pm 0.064}$

First, none of the inversion results shown here are visually comparable to the ground truth, which shows the necessity of having a large training dataset that covers the target distribution. Also, it is vital to ensure that the pre-processing and value ranges of training and target images are the same. Second, in terms of metrics, the results are comparable to but slightly worse than those on synthetic test images reported previously. This is expected because these images are out of distribution. Besides, in cases where we applied Gaussianization layers, using both noise maps and style vectors gives better results than using style vectors only. In addition, Gaussianization layers improved results when we also optimized the noise maps in addition to the style vectors. However, we found that only using the style vector and the spherical constraint gives the best result on this test set, which contradicts our results reported in Table 5. Our interpretation is that the dimension of style vectors is still relatively lower than noise maps or the latent tensor used in Glow, so our Gaussianization layers may be more effective in dealing with the latter two types of parameters.

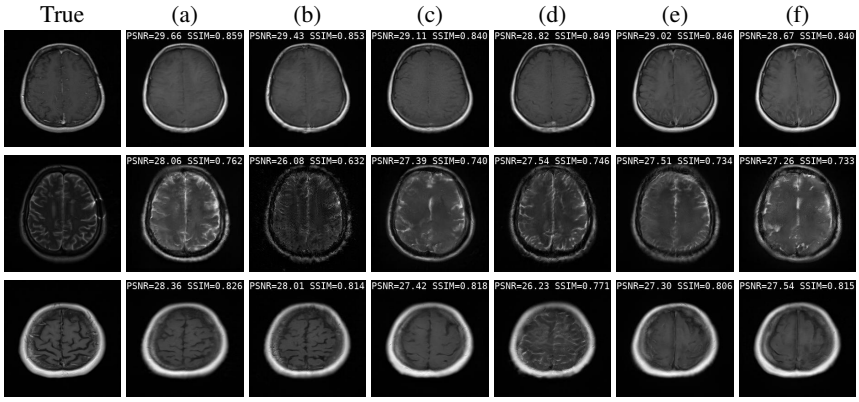

Figure 13: Examples of MRI inversion using out-of-distribution images. (Accl=8x, SNR=20 dB). (a) Style vector only + spherical constraint; (b) style vector + noise maps + spherical constraint; (c) style vector only + Gaussianization (only the Yeo-Johnson layer for 1D Gaussianization); (d) style vector + noise maps + Gaussianization (only the Yeo-Johnson layer for 1D); (e) style vector only + Gaussianization (only the Lambert layer for 1D); (f) style vector + noise maps + Gaussianization (only the Lambert layer for 1D).

D REPARAMETERIZATION SCHEMES FOR STYLEGAN2 AND GLOW

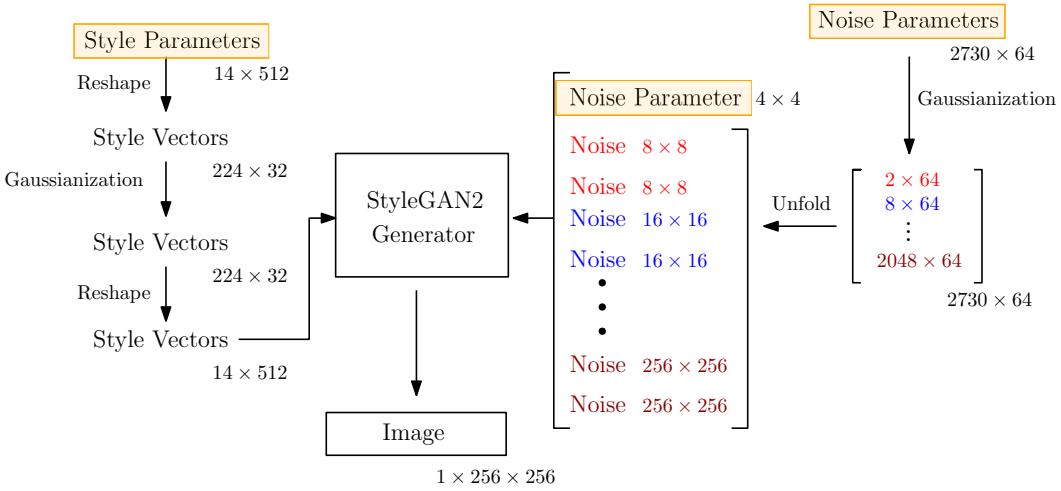

Figure 14: Reparameterization scheme for StyleGAN2. The dimensons of the latent parameters are [the number of vectors $\times$ vector dimension], except the 4×4 one. Note that we transpose latent tensors in corresponding equations and algorithms. The style and noise parameters before the Gaussianization layers are $\{\mathbf{v}_i|_{i=1,\cdots,N}\}$, and the style and noise vectors after the Gaussianization layers are $\{\mathbf{z}_i|_{i=1,\cdots,N}\}$.

Fig. 14 illustrates the reparameterization scheme for StylgeGAN2 using the Gaussianization layers. The patch size for the style vectors and noise maps are 32 and $1 \times 8 \times 8$, respectively. Note that we transpose the latent tensors in Fig. 14 compared to the notations in corresponding equations and algorithms. Also, as pointed out in Gu et al. (2020), we also find that a single latent code is insufficient for image reconstruction. Thus, we use multiple style vectors that were fed into different intermediate layers.

For Glow, we came up with two reparameterization schemes (Fig. 15). The first one (P1) reparameterizes patches from a latent tensor with the same dimension as the output image. The dimensions of the patches and the image are $3 \times 8 \times 8$ and $3 \times 128 \times 128$, respectively. Then a multi-scale squeezing operation maps the latent tensor into a list of tensors corresponding to different scales of

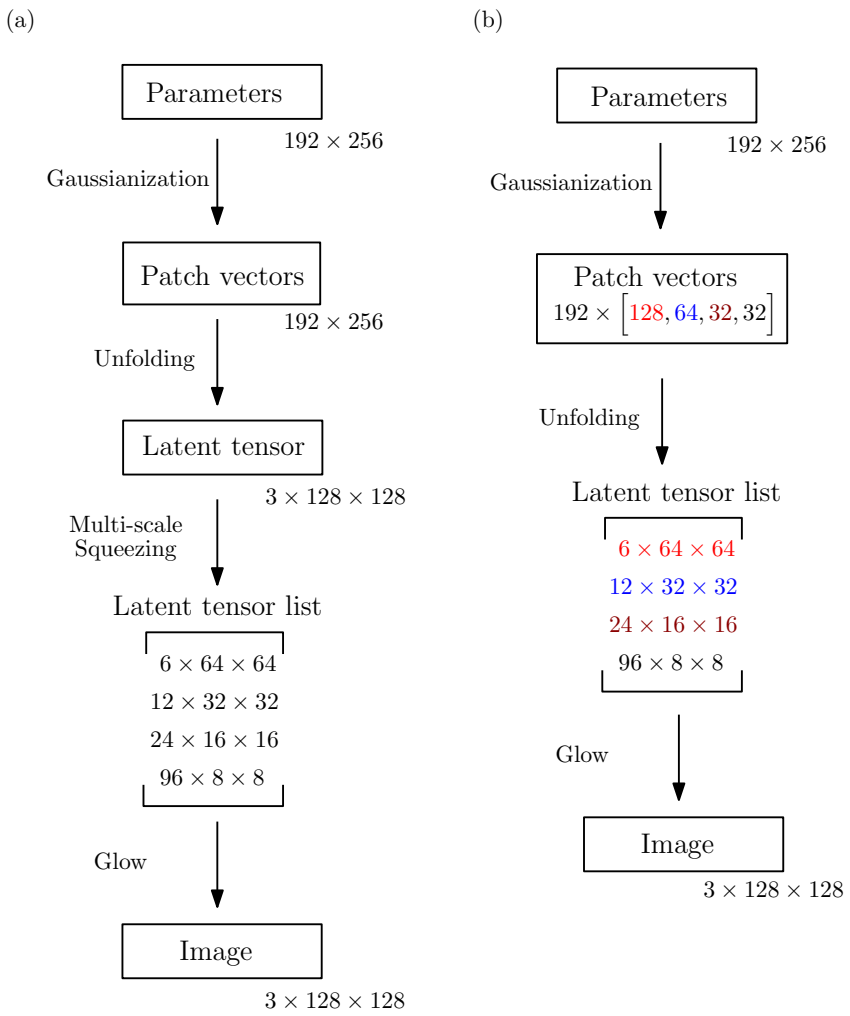

Figure 15: Two reparameterization schemes for Glow. (a) P1; (b) P2. The dimensions of the latent parameters are [vector dimension × the number of vectors]. The parameters before the Gaussianization layers are $\{\mathbf{v}_i|_{i=1,\cdots,N}\}$, and the patch vectors after the Gaussianization layers are $\{\mathbf{z}_i|_{i=1,\cdots,N}\}$.

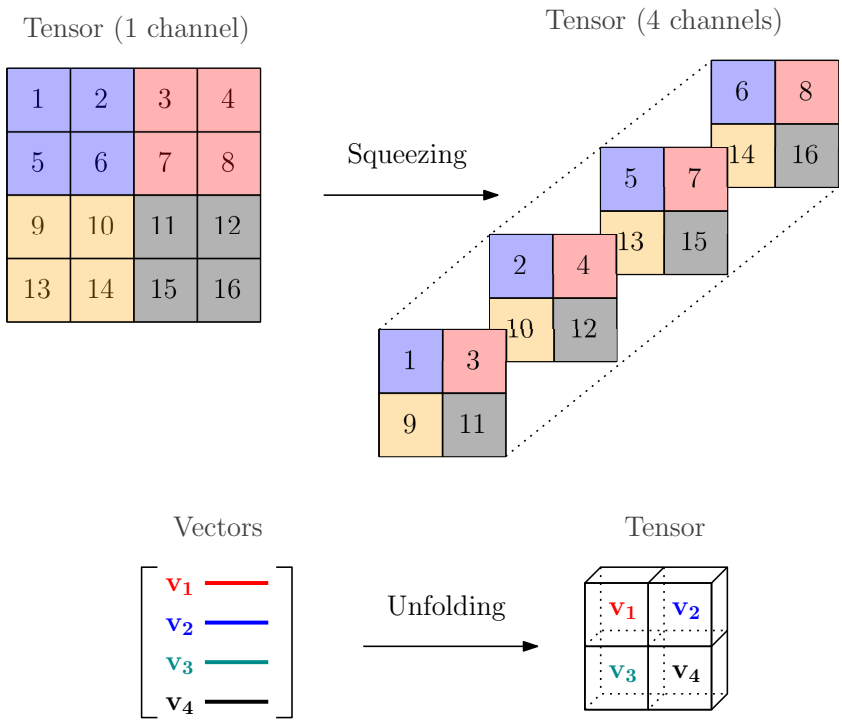

Figure 16: Illustration of the squeezeing and unfolding operations.

Glow. Glow uses the list of tensors as the input. The second scheme (P2) reparameterizes patches extracted directly from tensors in the above list. We also partition each tensor into $3 \times 8 \times 8$ patches. The unfolding and squeezing operations are illustrated in Fig. 16. In the multi-scale architecture of Glow, the squeezing operation is applied recursively on half of the output tensors cut in the channel direction (Dinh et al., 2016; Kingma & Dhariwal, 2018).

E More details of the Gaussianization layers

E.1 ICA layer

The overall ICA layer is summarized in algorithm 1. We set a maximum number of the fixed-point iterations to reduce computational cost and ensure accurate gradient computation that can pass the finite-difference convergence test. The gradients are backpropagated through the loops without using the trick introduced in App. G.

Whitening The FastICA algorithm typically requires that the data are pre-whitened. Let $\boldsymbol{V}$ be the data matrix whose columns are data vectors. We first subtract the mean from the data vector using $\boldsymbol{V} \leftarrow \boldsymbol{V} - \texttt{mean}(\boldsymbol{V}, \texttt{dim} = 1)$. Then we compute the data covariance matrix using $\boldsymbol{C} = (1-\eta)\frac{1}{N-1}\boldsymbol{V}\boldsymbol{V}^\top + \eta\boldsymbol{I}$, where we use a small constant (e.g., $\eta = 0.001$) to blend the empirical covariance matrix and an identity matrix to avoid ill-conditioning.

After these data preparation steps, the ZCA whitening used throughout the study first computes the eigenvalues $\boldsymbol{\Lambda}$ and eigenvectors $\boldsymbol{D}$, and then output the whitened data using $\boldsymbol{V} \leftarrow \boldsymbol{D}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{D}^\top\boldsymbol{V}$.

Alternatively, we can use the following steps to whiten the data, which are also used later in ICA iterations to decorrelate column vectors in the orthogonal matrix (Hyvarinen, 1999):

1. Initialize $\boldsymbol{W} \leftarrow \boldsymbol{I}$;
2. Compute

$$\boldsymbol{W} \leftarrow \boldsymbol{W} / \sqrt{\|\boldsymbol{W}^\top \boldsymbol{C} \boldsymbol{W}\|_2} \tag{16}$$

3. Repeat until convergence

$$\boldsymbol{W} \leftarrow \frac{3}{2}\boldsymbol{W} - \frac{1}{2}\boldsymbol{W}\boldsymbol{W}^\top\boldsymbol{C}\boldsymbol{W}, \tag{17}$$

4. Output $\boldsymbol{V} \leftarrow \boldsymbol{W}^\top\boldsymbol{V}$

Algorithm 1: ICA Layer

Input : Data matrix $\boldsymbol{V} \in \mathbb{R}^{D\times N}$; error tolerance ϵ; damping parameters: $\eta = 10^{-4}$ and $\alpha = 0.8$; maximum iteration numbers $J = 10$ and $K = 100$.
Output : Matrix $\boldsymbol{P} \in \mathbb{R}^{D\times N}$ with i.i.d. entries

```
// Whitening stage
```
$\boldsymbol{V} = \boldsymbol{V} - \texttt{mean}(\boldsymbol{V}, \texttt{dim} = 1)$, $\boldsymbol{C} = (1-\eta)\frac{1}{N-1}\boldsymbol{V}\boldsymbol{V}^\top + \eta\boldsymbol{I}$;
$\boldsymbol{V} \leftarrow \texttt{ZCA-whitening}(\boldsymbol{V})$;
```
// ICA stage
```
$\boldsymbol{W} = \boldsymbol{W}^* \leftarrow \mathbf{I}, \quad j \leftarrow 1$;
while $j \leq J$ **do**
  $\boldsymbol{W} \leftarrow \frac{1}{N}\left[\alpha\boldsymbol{V}\phi\left(\boldsymbol{W}^\top\boldsymbol{V}\right)^\top - \boldsymbol{W}\texttt{diag}\left(\phi'\left(\boldsymbol{W}^\top\boldsymbol{V}\right)\mathbf{1}\right)\right]$ Eq. 8 ;
  $\boldsymbol{W} = \boldsymbol{W}_0 \leftarrow \boldsymbol{W}/\sqrt{\|\boldsymbol{W}^\top\boldsymbol{W}\|_2}, \quad k \leftarrow 1$;
  while $k < K$ **do**
    $\boldsymbol{W} \leftarrow \frac{3}{2}\boldsymbol{W} - \frac{1}{2}\boldsymbol{W}\boldsymbol{W}^\top\boldsymbol{W}$;
    if $\|\boldsymbol{W} - \boldsymbol{W}_0\| < \epsilon$ **then**
      break;
    $\boldsymbol{W}_0 \leftarrow \boldsymbol{W}, \quad k \leftarrow k+1$;
  if $\|\boldsymbol{W} - \boldsymbol{W}^*\| < \epsilon$ **then**
    break;
  $\boldsymbol{W}^* \leftarrow \boldsymbol{W}, \quad j \leftarrow j+1$;
return $\boldsymbol{P} \leftarrow \boldsymbol{W}^\top\boldsymbol{V}$.

The modified FastICA iterations. As stated in Hyvärinen (1999), the objective function for one neural unit of the weight vector $\boldsymbol{w}_i$ and input $\mathbf{v}$ is

$$\arg\max_{\boldsymbol{w}_i} \mathbb{E}\left[\Phi\left(\boldsymbol{w}_i^\top\mathbf{v}\right)\right], \text{ s.t., } \mathbb{E}\left[\left(\boldsymbol{w}_i^\top\mathbf{v}\right)^2\right] = 1, \tag{18}$$

where Φ is the contrast function (*e.g.*, logcosh). The original derivations convert this constrained optimization to an unconstrained one using Lagrange multipliers. However, this procedure is unnecessary since the matrix $\boldsymbol{W}$ is orthogonalized after each iteration, and the input vectors have been pre-whitened. Therefore, we only need to solve the following equation

$$\mathbb{E}\big[\mathbf{v}\phi\left(\boldsymbol{w}_i^\top\mathbf{v}\right)\big] = 0, \tag{19}$$

whose Jacobian is

$$\begin{aligned} J &= \mathbb{E}\big[\mathbf{v}\mathbf{v}^\top\phi'(\boldsymbol{w}_i^\top\mathbf{v})\big] \\ &\approx \mathbb{E}\big[\mathbf{v}\mathbf{v}^\top\big]\mathbb{E}\big[\phi'(\boldsymbol{w}_i^\top\mathbf{v})\big] = \mathbb{E}\big[\phi'(\boldsymbol{w}_i^\top\mathbf{v})\big], \end{aligned} \tag{20}$$

where ϕ is the derivative of Φ. The Newton iteration scheme is thus

$$\boldsymbol{w}_i = \boldsymbol{w}_i - \mathbb{E}\big[\mathbf{v}\phi\left(\boldsymbol{w}_i^\top\mathbf{v}\right)\big]/\mathbb{E}\big[\phi'(\boldsymbol{w}_i^\top\mathbf{v})\big]. \tag{21}$$

To improve the convergence, we damp the iterations by a parameter $\alpha \in (0, 1)$. Also, using the same technique to convert the Newton iterations to fixed-point iterations in Hyvärinen (1999); Hyvarinen (1999), we arrive at the modified fixed-point iteration scheme:

$$\boldsymbol{w}_i = \alpha\mathbb{E}\left[\mathbf{v}\phi\left(\mathbf{w}_i^\top\mathbf{v}\right)\right] - \mathbb{E}\left[\phi'\left(\boldsymbol{w}_i^\top\mathbf{v}\right)\right]\boldsymbol{w}_i, \tag{22}$$

followed by the aforementioned decorrelation procedure after each step. The convergence of the modified FastICA iterations can be proved similarly as in Oja & Yuan (2006).

Algorithm 2: Power Transformation Layer

Input : Data vector $\boldsymbol{p}$
Output : Vector $\boldsymbol{s}$ whose values are 1D Gaussianized.
1 Estimate λ from $\boldsymbol{p}$ using Eq. 24 ;
2 Compute $\boldsymbol{s}$ with the estimated λ and data $\boldsymbol{p}$ using Eq. 23 ;
3 **return** $\boldsymbol{s}$.

E.2 POWER TRANSFORMATION LAYER

We propose to use the power transformation or Yeo-Johnson transformation (Yeo & Johnson, 2000) to reduce the skewness of distributions:

$$s(\lambda, p) = \begin{cases} \left((p+1)^{\lambda} - 1\right)/\lambda, & p \geq 0, \lambda \neq 0, \\ \log(p+1), & p \geq 0, \lambda = 0, \\ -([-p+1]^{2-\lambda} - 1)/[(2-\lambda)], & p < 0, \lambda \neq 2, \\ -\log(-p+1), & p < 0, \lambda = 2, \end{cases} \tag{23}$$

where p is an input value, s is an output value, and λ is the parameter to be estimated. As shown in Fig. 3(a), the form of the Yeo-Johnson activation function depends on parameter λ. If $\lambda = 1$, the mapping is an identity mapping. If $\lambda \geq 1$, the activation function is convex, compressing the left tail and extending the right tail, reducing the left-skewness. If $\lambda \leq 1$, the activation function is concave, which oppositely reduces the right-skewness. The only parameter λ is determined by solving an optimization problem that minimizes the negentropy:

$$\lambda = \arg\max_{\lambda} l(\lambda|\boldsymbol{p}) = \arg\max_{\lambda} -\frac{n}{2}\log \mathrm{Var}(s(\lambda, p_i)) + (\lambda - 1)\sum_{i=1}^{n} \mathrm{sign}(p_i)\log(|p_i| + 1), \tag{24}$$

where $\boldsymbol{p}$ is the input data vector with entries $p_{i,\, i=1,\cdots,n}$.

We use a custom operator based on SciPy's implementation using Brent's algorithm (Brent, 2013) to find an approximate minimum of Problem 24. Continuing from the approximate minimum, we use Brent's root finding algorithm (Brent, 2013) to find the minimum where the gradient vanishes.

Since the parameter λ depends on input data, we need to back-propagate the gradient through the optimization process.

The power transformation layer is summarized in algorithm 2.

E.3 LAMBERT $W \times F_X$ LAYER

Due to noise and inaccurate forward models, we observe that the distribution of latent vector values tends to be shaped as a heavy-tailed distribution during the inversion process. To reduce the heavy-tailedness, we adopt the Lambert $W \times F_X$ method detailed in Goerg (2015).

Let X be a random variable whose CDF is F_X, with mean μ_X and standard deviation σ_X. The following transformation with a heavy-tail parameter $\delta \geq 0$:

$$S = \left(U \exp\left(\frac{\delta}{2}U^2\right)\right)\sigma_X + \mu_X, \tag{25}$$

where $U = (X - \mu_X)/\sigma_X$, is a bijection and maps X to another random variable S with heavier tails.

The transformation Eq. 25 is bijective if $\delta \geq 0$, and we can use the Lambert W function to find its inverse. The Lambert W function W is defined as the inverse of $q = W^{-1}(t) = t\exp(t)$, where t and q are scalars. Given q, Halley's method can be used to find $t = W(q)$ (Corless et al., 1996). Hence, the inverse of Eq. 25 is

$$X = W_{\delta}\left(\frac{S - \mu_X}{\sigma_X}\right)\sigma_X + \mu_X, \tag{26}$$

where

$$W_\delta(u) = \operatorname{sign}(u)\sqrt{\frac{W(\delta u^2)}{\delta}}. \tag{27}$$

We use the parameterized Lambert $W \times F_X$ distribution family to approximate a heavy-tailed input distribution and use Eq. 26 to recover a distribution with lighter tails. In order to make the recovered distribution close to a Gaussian distribution, we compute the optimal heavy-tail parameter δ by minimizing the difference between the kurtosis of the output distribution and 3 (Kurtosis is a common surrogate measure of negentropy (Hyvärinen & Oja, 2000)):

$$\hat{\delta}_{\text{GMM}} = \arg\min_{\delta>0} \left|\text{Kurt}\left(W_\delta\left(\frac{\boldsymbol{s}-\mu_X}{\sigma_X}\right)\right) - 3\right|^2, \tag{28}$$

where $\boldsymbol{s}$ is the data vector, and Kurt is the kurtosis. We constrain $\delta > 0$, and solve Eq. 28 using the L-BFGS-B optimizer (Zhu et al., 1997).

In addition, we estimate the mean μ_X and standard deviation σ_X along with δ using the Iterative Generalized Method of Moments (IGMM) (Goerg, 2015), which embeds an optimization problem for δ in an outer loop of iterations to estimate σ_X and μ_X (see algorithm 3). If the kurtosis of input data vector is not greater than 3, we skip the whole Lambert $W \times F_X$ layer by directly outputting the data vector.

Algorithm 3: Lambert $W \times F_X$ Layer with the Iterative Generalized Method of Moments (IGMM) (Goerg, 2015)

Input : Data vector $\boldsymbol{s}$, error tolerance $\epsilon = 10^{-5}$, maximum iteration number $K = 100$.
Output : vector $\boldsymbol{x}$ whose empirical distribution is less heavy-tailed than $\boldsymbol{s}$, and its kurtosis ≈ 3)

Initialize: $\xi^{(0)} \leftarrow \left(\hat{\mu}_X^{(0)}, \hat{\sigma}_X^{(0)}, \hat{\delta}^{(0)}\right)$, $k = 0$;
Compute initial kurtosis $\beta_2 = \text{Kurt}(\boldsymbol{s})$;
if $\beta_2 \leq 3$ **then**
  return $\boldsymbol{s}$;
end
while $k < K$ *and* $\|\xi^{(k)} - \xi^{(k-1)}\| \geq \epsilon$ **do**
  $\boldsymbol{u}^{(k)} \leftarrow \left(\boldsymbol{s} - \mu_X^{(k)}\right)/\sigma_X^{(k)}$;
  Compute $\delta^{(k+1)}$ using Eq. 28; $\boldsymbol{u}^{(k+1)} \leftarrow \boldsymbol{W}_{\delta^{(k+1)}}\left(\boldsymbol{u}^{(k)}\right)$; $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{u}^{(k+1)}\sigma_X^{(k)} + \mu_X^{(k)}$;
  Update $\mu_X^{(k+1)} \leftarrow \mathbb{E}\left[\boldsymbol{x}^{(k+1)}\right]$, and $\sigma_X^{(k+1)} \leftarrow \sqrt{\text{Var}\left(\boldsymbol{x}^{(k+1)}\right)}$;
  $\xi^{(k+1)} \leftarrow \left(\hat{\mu}_X^{(k+1)}, \hat{\sigma}_X^{(k+1)}, \hat{\delta}^{(k+1)}\right)$;
  $k \leftarrow k + 1$;
end
return $\boldsymbol{x} = W_\delta\left(\frac{\boldsymbol{s}-\mu_X}{\sigma_X}\right)\sigma_X + \mu_X$,

F Details of datasets and training

For MRI and eikonal tomography, we generate synthetic brain images as inversion targets using the pre-trained StyleGAN2 weights from Kelkar & Anastasio (2021), which are trained on data from the databases of fastMRI (Zbontar et al., 2018; Knoll et al., 2020), TCIA-GBM (Scarpace et al., 2016), and OASIS-3 (LaMontagne et al., 2019). As a result, there is no sensitive personal information in our target images. In the data generation process, we only used one style vector of a dimension of 512. We picked 100 images that are visually plausible, whose random seeds can be found in our code. These random seeds were never used to initialize inversion. In inversion experiments, we used 14 such style parameters: one for the lowest resolution of 4×4, one for the tRGB layer, and two for each resolution from 8×8 to 256×256, to avoid inversion crime. This choice of expanding latent space dimension is also justified by our observation that only one style parameter vector is generally insufficient for inversion tasks with data from datasets such as CelebA-HQ (Karras et al., 2018).

We used the CelebA-HQ dataset (Karras et al., 2018) (under the Creative Commons CC BY-NC 4.0 license) for the deblurring experiments. All images were downsampled to the resolution of 128×128. We split the 30000 images from CelebA-HQ into the subsets of training (24183 images), validation (2993 images), and testing (2824 images) following the original splits from CelebA (Liu et al., 2015). For the inversion tests, we randomly selected 100 images from the test set.

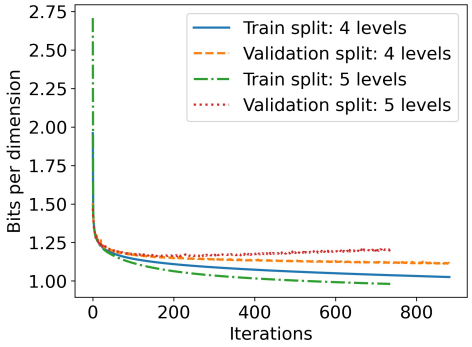

Figure 17: The negative log-likelihood or NLL (reported in bits per dimension) on the training and validation splits of the CelebA-HQ dataset with different numbers of multi-scale levels.

For the hyper-parameters of the Glow networks, we used 4 multi-scale levels and 32 flow-steps, and we only used additive coupling layers. Fig. 17 reports the training process. For each epoch, we computed the training negative log-likelihood (NLL) averaged throughout the epoch, and the validation NLL at the end of the epoch. The validation curves suggest that it is better to use 4 multi-scale levels. We chose the network weights from the epoch before the validation NLL stopped to decrease: 850 for the CelebA-HQ dataset. All training was conducted using 8×32 GB Nvidia V100 GPUs with a batch size of 64. We used the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-4}, as well as $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

G Gradient computation of the optimization-based differentiable layers

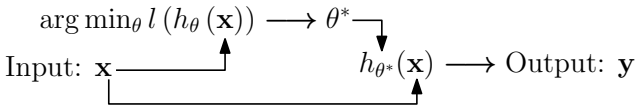

Figure 18: Illustration of the forward computation of optimization-based ICA, Yeo-Johnson, and Lambert $W \times F_X$ layers. We denote the input and output by $\mathbf{x}$ and $\mathbf{y}$, respectively. The layer is represented by h_θ, and the layer is defined by solving an optimization problem whose objective function is l. The back-propagation of gradients in components defined by Eq. 24 and Eq. 28 is enabled by the implicit function theorem in automatic differentiation detailed in App. G.

As shown in Fig. 18, in the power transformation and Lambert $W \times F_X$ layers, there are operators whose outputs are obtained by solving optimization problems formally described as

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} l(\boldsymbol{x}, \boldsymbol{\theta}), \tag{29}$$

where l denotes the objective function that defines the operator (combining the objective function and layer $h_{\boldsymbol{\theta}}$ in Fig. 18), symbol $\boldsymbol{\theta}$ stands for the optimal output, a scalar in our cases but can also be a vector in general situations. The optimal condition is

$$l_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{\theta}) \coloneqq L(\boldsymbol{x}, \boldsymbol{\theta}) = 0, \tag{30}$$

where the subscript denotes partial derivatives.

The optimal condition implicitly defines a forward operator of the following form:

$$\boldsymbol{\theta}^* = \mathtt{op}_{\text{forward}}\left(\boldsymbol{x}\right). \tag{31}$$

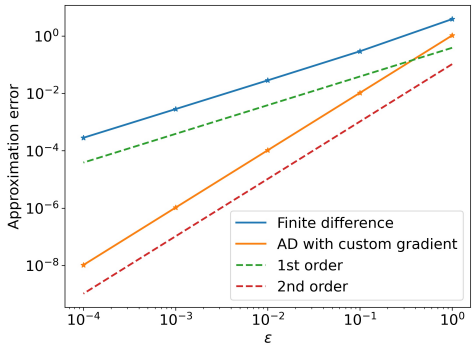

Figure 19: Gradient accuracy test for custom operators. An accurate gradient should make the error converge at a speed of the second order following the red dashed line.

The backward operator is

$$\frac{\partial \chi}{\partial \boldsymbol{x}} = \mathsf{op}_{\text{backward}}\left(\frac{\partial \chi}{\partial \boldsymbol{\theta}}, \boldsymbol{\theta}^*, \boldsymbol{x}\right), \tag{32}$$

where χ is the objective function of an inverse problem.

Differentiating Eq. 30 with respect to $\boldsymbol{x}$, we have

$$L_{\boldsymbol{x}} + L_{\boldsymbol{\theta}}\boldsymbol{\theta}_{\boldsymbol{x}} = 0 \implies \boldsymbol{\theta}_{\boldsymbol{x}} = -L_{\boldsymbol{\theta}}^{-1}L_{\boldsymbol{x}}, \tag{33}$$

using the implicit function theorem. Then, to back-propagate the gradient from $\frac{\partial \chi}{\partial \boldsymbol{\theta}}$ to $\frac{\partial \chi}{\partial \boldsymbol{x}}$, we use

$$\frac{\partial \chi}{\partial \boldsymbol{x}} = \frac{\partial \chi}{\partial \boldsymbol{\theta}}\boldsymbol{\theta}_{\boldsymbol{x}} = -\frac{\partial \chi}{\partial \boldsymbol{\theta}}L_{\boldsymbol{\theta}}^{-1}L_{\boldsymbol{x}} = -\frac{\partial \chi}{\partial \boldsymbol{\theta}}H_{\boldsymbol{\theta}}^{-1}L_{\boldsymbol{x}}, \tag{34}$$

where $H_{\boldsymbol{\theta}}$ is the Hessian matrix of χ with respect to $\boldsymbol{\theta}$.

In our problems, the output $\boldsymbol{\theta}$ is a scalar, so it is easy to use automatic differentiation to compute $L_{\boldsymbol{\theta}}$ directly and hence $L_{\boldsymbol{\theta}}^{-1}$. Otherwise, if $\boldsymbol{\theta}$ has many parameters, we can first solve the following linear system with an auxiliary vector $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda}H_{\boldsymbol{\theta}} = -\frac{\partial \chi}{\partial \boldsymbol{\theta}}, \tag{35}$$

and then compute the gradient using

$$\frac{\partial \chi}{\partial \boldsymbol{x}} = \boldsymbol{\lambda}L_{\boldsymbol{x}}, \tag{36}$$

a technique also known as the adjoint-state method. Note that there is no need to compute the Hessian explicitly, but one can use automatic differentiation to compute the vector-Hessian product $\boldsymbol{\lambda}H_{\boldsymbol{\theta}}$ and utilize iterative linear solvers like GMRES (Saad & Schultz, 1986) to solve the linear system.

As a final note, we check the accuracy of our gradient computation using the finite-difference convergence test based on Taylor expansion:

$$f(\boldsymbol{x} + \epsilon\delta\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon\nabla f(\boldsymbol{x})^\top\delta\boldsymbol{x} + \mathcal{O}(\epsilon^2), \tag{37}$$

where $\delta\boldsymbol{x}$ is a random vector with a unit ℓ_2 norm, and $\nabla f(\boldsymbol{x})$ is computed using our custom gradient. We here define f as a composite function that maps the (vector) output of an forward operator to a scalar, *e.g.*, $f(\boldsymbol{x}) = \|\mathsf{op}_{\text{forward}}(\boldsymbol{x})\|_2^2$. Once we decrease ϵ, we should see that the error term $f(\boldsymbol{x} + \epsilon\delta\boldsymbol{x}) - f(\boldsymbol{x}) - \epsilon\nabla f(\boldsymbol{x})^\top\delta\boldsymbol{x}$ decreases at a speed of the second order. All our layers passed this test, as the example shown in Fig. 19. This test should be conducted in double precision.

H ORTHOGONAL REPARAMETERIZATION

We also propose to reparameterize the latent vector $\mathbf{z}$ using an orthogonal matrix $\boldsymbol{R}$:

$$\mathbf{z} = \boldsymbol{R}\mathbf{v},\ \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{38}$$

where $\mathbf{v}$ is fixed during an inversion, and we treat $\boldsymbol{R}$ as the parameter instead. There are various ways to parameterized an orthogonal matrix. We choose the Cayley parameterization:

$$\boldsymbol{R} = (\boldsymbol{I} + \boldsymbol{W})(\boldsymbol{I} - \boldsymbol{W})^{-1}, \text{ where } \boldsymbol{W} = -\boldsymbol{W}^{\top}, \tag{39}$$

one of the best reported in Liu et al. (2021). Therefore, the DGM-regularized inversion using orthogonal reparameterization is

$$\underset{\boldsymbol{W}}{\arg\min}\ (1/2) \left\| \mathbf{d} - f \circ g \left(\boldsymbol{R}\mathbf{v} \right) \right\|_2^2 . \tag{40}$$

The specific reparameterization schemes for StyleGAN2 and Glow are the same as in App. D. For style vectors, we use the full dimension of 512 because of the observation described in Table 6. For Glow and noise maps in StyleGAN2, we use patch sizes $3 \times 8 \times 8$ and $1 \times 8 \times 8$, respectively, to save computation time and memory.

I GAUSSIAN TYPICAL SET

The Gaussian annulus theorem states that a high-dimensional standard Gaussian distribution has most of its probability mass concentrated within an annulus area around a high-dimensional sphere:

Theorem 1 (Gaussian Annulus Theorem (Blum et al., 2020)) *For an n-dimensional standard Gaussian, for any $\beta \leq \sqrt{n}$, all but at most $3e^{-c\beta^2}$ of the probability mass lies within the annulus $\sqrt{n} - \beta \leq |x| \leq \sqrt{n} + \beta$, where c is a fixed positive constant.*

This theorem gives a necessary geometric condition of typical samples from a high-dimensional standard Gaussian. However, the converse is not true: not all vectors whose ℓ_2 norm is $\sqrt{n} - \beta \leq |x| \leq \sqrt{n} + \beta$ are typical examples from the standard Gaussian distribution. As a result, a DGM does not necessarily map a latent vector staying within the Gaussian annulus geometrically to a plausible image, which is demonstrated in Fig. 1.

The formal definition of a typical set is as follows.

Definition 1 (Cover & Thomas (2012)) *Let $p_X(x)$ be a distribution whose support is $\mathcal{X}$. The typical set $A_\epsilon^{(n)}$ is defined as the set of sequences $(x_1, x_2, \cdots, x_n) \in \mathcal{X}^n$, $x_i \sim p_X$, that satisfy*

$$\left| H\left[X\right] + \frac{1}{n} \log p_X(x_1, \cdots, x_n) \right| \leq \epsilon, \tag{41}$$

where $H\left[X\right]$ is the entropy of random variable X.

Now a random vector $\mathbf{x} \in \mathbb{R}^n \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ can be factorized as i.i.d. random variables that are distributed as $\mathcal{N}(0, \sigma^2)$. Therefore, we can regard $\mathbf{x}$ as an i.i.d. sequence and give the following definition:

Definition 2 (Gaussian Typical Set) *A Gaussian typical set is the typical set $A_\epsilon^{(n)}$ of $\mathbf{x} \in \mathbb{R}^n \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.*

The following theorem guarantees that a typical sample from $\mathbf{x} \in \mathbb{R}^n \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ resides in the Gaussian typical set with very high probability.

Theorem 2 (Cover & Thomas (2012)) *For every $\epsilon > 0$, the typical set has probability $P\left(A_\epsilon^{(n)}\right) > 1 - \epsilon$ with a sufficiently large dimension n.*

This theorem is a direct application of the asymptotic equipartition property (AEP), which is based on the i.i.d. assumption of sequence entries and the weak law of large numbers. Similar to the Gaussian annulus theorem, Theorem 2 depicts a geometric property of the Gaussian typical set: it is concentrated in an annulus near a shell of radius $\sigma\sqrt{n}$, which can be directly verified by definition. Of course, equivalently, if a vector whose ℓ_2 norm significantly deviates from $\sqrt{n}$ cannot be a typical sample from a standard Gaussian. However, one cannot assert that a vector is sampled from the Gaussian typical set by only checking its norm.

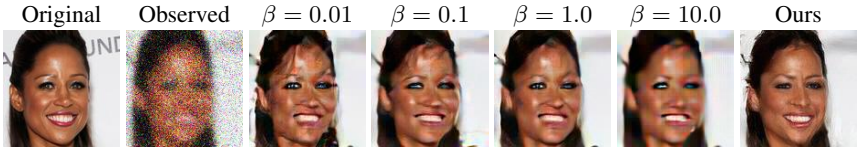

Figure 20: Comparison of conventional DGM (Glow)-regularized deblurring results with different weighting factor βs and results using our Gaussianization layers.

J MISCELLANEOUS TOPICS

J.1 INVERSION USING GLOW

In the Glow-regularization framework, we solve the inverse problem by finding the maximum *a posteriori* (MAP) estimate from

$$p_M(\mathbf{m}|\mathbf{d}) \propto p\left(\mathbf{d}|\mathbf{m}\right) p_M(\mathbf{m}), \tag{42}$$

where M denotes the parameter space. The probability density p_M introduces our *a priori* knowledge and is represented by a normalizing flow (Glow) $g_{\boldsymbol{\theta}}$, which is a differentiable invertible mapping between two distributions, parameterized by neural network parameters $\boldsymbol{\theta}$: $\mathbf{m} = g_{\boldsymbol{\theta}}(\mathbf{z})$, where $\mathbf{z}$ is the latent vector. After training, the log probability density of a given model $\mathbf{m}$ is

$$\log p_M(\mathbf{m};\boldsymbol{\theta}) = \log p_Z\left(g_{\boldsymbol{\theta}}^{-1}\left(\mathbf{m}\right)\right) + \log\left|\det J_{g_{\boldsymbol{\theta}}^{-1}}(\mathbf{m})\right| = \log p_Z\left(\mathbf{z}\right) - \log\left|\det J_{g_{\boldsymbol{\theta}}}(\mathbf{z})\right|, \tag{43}$$

where Z stands for the latent space. When we use the trained network in inversion, we freeze the network weights; hence, we drop $\boldsymbol{\theta}$ in $g_{\boldsymbol{\theta}}$ hereafter in our notation. Therefore, Eq. 3 for Glow-regularized inversion is

$$\arg\min_{\mathbf{z}} \ (1/2)\left\|\mathbf{d} - f \circ g\left(\mathbf{z}\right)\right\|_2^2 - \beta\left(\log p_Z\left(\mathbf{z}\right) - \log\left|\det J_g(\mathbf{z})\right|\right). \tag{44}$$

The new regularization term in Eq. 3 is $\mathcal{R}'(\mathbf{z}) = -\beta\left(\log p_Z\left(\mathbf{z}\right) - \log\left|\det J_g(\mathbf{z})\right|\right)$, and we have shown the results with different βs in Fig. 20 and Table 7.

J.2 DUALITY OF KL DIVERGENCE

As also shown in Papamakarios et al. (2017), the KL-divergence between two distributions does not change under a differentiable invertible transformation, so

$$D_{\mathrm{KL}}\left[p_M^*(\mathbf{m})\|p_M(\mathbf{m};\boldsymbol{\theta})\right] = D_{\mathrm{KL}}\left[p_Z^*(\mathbf{z};\boldsymbol{\theta})\|p_Z(\mathbf{z})\right], \tag{45}$$

where p_M^* is the target distribution in the physical parameter space, and p_Z^* is the corresponding latent-space distribution under the normalizing flow. This means that minimizing the forward KL divergence in the M domain or physical parameter space is equivalent to minimizing the reverse KL-divergence in the Z domain or the latent space.

This fact and Theorem 2 imply that a well-trained normalizing flow maps samples from the target distribution into the Gaussian typical set with very high probability and vice versa.

J.3 INVARIANCE OF KL-DIVERGENCE AND MULTI-INFORMATION

For the completeness of this paper, we briefly prove the key properties of KL-divergence and multi-information used in the Gaussianization framework.

First, we prove that the KL-divergence is invariant under differentiable bijections. We write the definition of the KL-divergence between distribution $p_{\mathbf{x}}$ and $q_{\mathbf{x}}$ of random vector $\mathbf{x}$ as

$$D_{\mathrm{KL}}(p_{\mathbf{x}}\|q_{\mathbf{x}}) = \int p_{\mathbf{x}}(\boldsymbol{x}) \log \frac{p_{\mathbf{x}}(\boldsymbol{x})}{q_{\mathbf{x}}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}. \tag{46}$$

Suppose there is a differentiable and invertible transformation T that maps $\mathbf{x}$ to $\mathbf{u}$: $\mathbf{u} = T(\mathbf{x})$. Then, we know the PDF under change of variable is

$$p_{\mathbf{x}}(\boldsymbol{x}) = p_{\mathbf{u}}(\boldsymbol{u})|\det J_T(\boldsymbol{x})|, \quad q_{\mathbf{x}}(\boldsymbol{x}) = q_{\mathbf{u}}(\boldsymbol{u})|\det J_T(\boldsymbol{x})|, \tag{47}$$

where we define the Jacobian matrix $J_T(\boldsymbol{x}) = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}(\boldsymbol{x})$. Therefore, we have

$$\begin{aligned} D_{\mathrm{KL}}(p_{\mathbf{x}} \| q_{\mathbf{x}}) &= \int p_{\mathbf{x}}(\boldsymbol{x}) \log \frac{p_{\mathbf{x}}(\boldsymbol{x})}{q_{\mathbf{x}}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \\ &= \int p_{\mathbf{u}}(\boldsymbol{u}) |\det J_T(\boldsymbol{x})| \log \frac{p_{\mathbf{u}}(\boldsymbol{u}) |\det J_T(\boldsymbol{x})|}{q_{\mathbf{u}}(\boldsymbol{u}) |\det J_T(\boldsymbol{x})|} \mathrm{d}\boldsymbol{x} \\ &= D_{\mathrm{KL}}(p_{\mathbf{u}} \| q_{\mathbf{u}}) \end{aligned} \tag{48}$$

Then, we show that the multi-information does not change under component-wise invertible differentiable transformations. The multi-information is defined by the KL-divergence between the joint distribution $p(\mathbf{x})$ and the marginals:

$$I(\mathbf{x}) = I(\mathrm{x}_1, \cdots, \mathrm{x}_D) = D_{\mathrm{KL}} \left(p(\mathbf{x}) \left\| \prod_j^D p_{\mathrm{x}}^j(\mathrm{x}_j) \right. \right). \tag{49}$$

We define the bijection $\mathbf{u} = T(\mathbf{x})$ specifically with invertible differentiable transformations for each component $T_i(\mathrm{x}_i) = \mathrm{u}_i$. Then, using the KL-divergence invariance we just proved and that $\prod_j^D p_{\mathrm{x}}^j(\mathrm{x}_j) = \prod_j^D p_{\mathrm{u}}^j(\mathrm{u}_j) |\det J_T(\boldsymbol{x})|$, we get $I(\mathbf{x}) = I(\mathbf{u})$.

J.4 The rolling operation

After applying one set of the Gaussianization layers to the latent tensor, it might be desirable to apply the layers to a different set of non-overlapping patches from the latent tensor again. The rolling operation shifts the latent tensor in the horizontal and vertical directions by half of the patch size w before patch extractions. The values at the boundaries are wrapped around to the opposite sides. One can, for example, use the `torch.roll` command to implement this functionality.

K Additional discussions

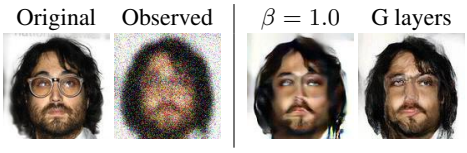

Figure 21: A failure example where inversion is unable to restore the eyeglasses because of the strong constraint from traversing within the Gaussian typical set. The results are dominated by the bias of the training dataset.

We point out the limitations and potential improvements of this work. First, requiring latent vectors to traverse within the Gaussian distribution/typical set means that the statistics of the training dataset will dominate the results. Fig. 21 shows that our method cannot restore the eyeglasses because it is not a significant feature within our training data. One needs to pay particular attention to training dataset construction to ensure that it represents the typical features to be investigated. One should also interpret their results with this caveat in mind. On the other hand, one may combine Gaussianization layers and latent space manipulation techniques (Shen et al., 2020) for a guided inversion, *e.g.*, requiring that the inverted images should have eyeglasses. Second, we only use one set of Gaussianization layers. One can further improve the performance by using additional sets of layers, with possible different partition schemes, such as using the rolling operation (App. J.4) to shift the patch extraction locations. Third, since the deep generative models are highly nonlinear, the results may get stuck in local minima. We mitigate this problem by starting inversion from multiple randomly initialized latent vectors. However, this technique is computationally expensive. It is an important research direction to improve the initialization process.

The Gaussianization layers are based on optimization and fixed-point iterations, which can be computationally expensive. To study possible simplifications, we estimated the forward and backward time for various combinations of sub-layers using 100 repeated experiments (Table 11). The computation

was performed using one Nvidia V100 GPU and an Intel Xeon Platinum 8276 CPU. We always turn on the ICA layer (together with whitening) since the ablation studies show that it contributes the most to performance increase (Table 2 and Table 9). Combining results from those ablation studies, we can see that the option ICA+YJ strikes a good balance between performance and computation time. It has a shorter runtime than ICA+Lambert and ICA+YJ+Lambert. At the same time, it gives comparable and sometimes the best results. On the other hand, the additional computation time from Gaussianization layers is well justified by the performance increase, especially if the physics simulation is much more time-consuming. For example, the eikonal solver for the experimental setup in our study has a forward runtime of 0.293 ± 0.013 seconds and a backward runtime of 5.933 ± 0.151 seconds, respectively. In addition, our implementation can also be greatly improved by more efficient utilization of GPUs and rewriting bottleneck components using high-performance codes.

Table 11: Forward and backward computation time of various combinations of Gaussianization sublayers. The times were computed using tensors of the same shape as latent tensors for Glow and StyleGAN2, respectively. Each computation time was estimated by 100 repeated experiments. We always turn on the whitening layer and the standardization layer.

(Note: the time is measured in seconds)	ICA	ICA + YJ	ICA + Lambert	ICA + YJ + Lambert
Forward (Glow tensor)	$0.030_{\pm 0.038}$	$0.286_{\pm 0.052}$	$1.911_{\pm 0.338}$	$2.022_{\pm 0.304}$
Backward (Glow tensor)	$0.017_{\pm 3.8\text{e-}4}$	$0.022_{\pm 3.3\text{e-}4}$	$0.751_{\pm 0.110}$	$0.693_{\pm 0.123}$
Forward (StyleGAN2 tensor)	$0.033_{\pm 3.1\text{e-}4}$	$0.280_{\pm 0.047}$	$1.068_{\pm 0.141}$	$1.212_{\pm 0.097}$
Backward (StyleGAN2 tensor)	$0.027_{\pm 2.6\text{e-}4}$	$0.037_{\pm 5.4\text{e-}4}$	$1.077_{\pm 0.100}$	$0.964_{\pm 0.065}$

We also provide insights into why Gaussianization layers perform better than other methods. First, the spherical constraint only relies on the geometric property of the Gaussian typical set (App. I). However, as shown in Fig. 1, it is insufficient to constrain the norm of latent vectors in inversion only. Gaussianization is necessary to ensure inverted results are in the range of DGMs. Second, the orthogonal reparameterization generally underperforms the Gaussianization layers. Our interpretation is as follows: rather than actively destroying latent tensor patterns like the Gaussianization layers do, orthogonal reparameterization cannot prevent permuting the fixed typical Gaussian latent tensors into ones with spatial patterns. In addition, we have shown in Fig. 10 that the whitening transformation alone is ineffective in keeping the DGM outputs in the range. We thus interpret that reducing higher-order dependencies is essential. Therefore, the noise regularization and the whitening layers yield less satisfying results than the Gaussianization layers.

We believe this study can benefit both the broader scientific community and the general public. However, we point out that the MRI and eikonal tomography experiments are purely synthetic and numerical. They do not fully reflect realistic medical imaging configurations. One should be cautious about applying our techniques to real data and interpreting the results.