TRUMPETS: Injective Flows for Inference and Inverse Problems

Konik Kothari[1] **AmirEhsan Khorashadizadeh**[2] **Maarten de Hoop**[3] **Ivan Dokmanić**[2]

[1]Coordinated Science Laboratory, University of Illinois at Urbana-Champaign
[2]Department of Mathematics and Computer Science, University of Basel
[3] Computational and Applied Mathematics, Rice University

Abstract

We propose injective generative models called TRUMPETs that generalize invertible normalizing flows. The proposed generators progressively increase dimension from a low-dimensional latent space. We demonstrate that TRUMPETs can be trained orders of magnitudes faster than standard flows while yielding samples of comparable or better quality. They retain many of the advantages of the standard flows such as training based on maximum likelihood and a fast, exact inverse of the generator. Since TRUMPETs are injective and have fast inverses, they can be effectively used for downstream Bayesian inference. To wit, we use TRUMPET priors for maximum a posteriori estimation in the context of image reconstruction from compressive measurements, outperforming competitive baselines in terms of reconstruction quality and speed. We then propose an efficient method for posterior characterization and uncertainty quantification with TRUMPETs by taking advantage of the low-dimensional latent space. Our code is publicly available at `https://github.com/swing-research/trumpets`.

1 INTRODUCTION

Modeling a high-dimensional distribution from samples is a fundamental task in unsupervised learning. An ideal model would efficiently generate new samples and assign likelihoods to existing samples. Some deep generative models such as generative adversarial networks (GANs) [Goodfellow et al., 2014] can produce samples of exceedingly high quality, but they do not give access to the underlying data distribution. Moreover, GANs are often hard to train, suffering from pathologies such as mode collapse [Thanh-Tung and Tran, 2020, Arjovsky and Bottou, 2017]. Since they are generally not invertible, or computing the inverse is slow, they are not well-suited for downstream inference tasks such as image reconstruction from compressive measurements or uncertainty quantification.

Normalizing flows alleviate many of the drawbacks of GANs: they approximate high-dimensional probability distributions as invertible transformations of a simple, tractable base distribution. They allow both efficient sampling and likelihood evaluation. They can be trained using maximum likelihood, and at inference time they provide direct access to likelihoods. These desirable features are a consequence of clever architectural components known as coupling layers [Dinh et al., 2014].

Normalizing flows, however, are extremely compute-intensive. As a case in point, training a Glow model [Kingma and Dhariwal, 2018] for the 5-bit 256×256 CelebA dataset takes a week on 40 GPUs. This is in part because the dimension of the "latent" space in normalizing flows equals that of the generated images. Since signals of interest are often concentrated close to low-dimensional structures embedded in high-dimensional spaces, this is a waste of resources. Beyond reducing computational cost, a low-dimensional latent space acts as a natural regularizer when solving ill-posed inverse problems [Bora et al., 2017].

In this paper we propose a new generative model termed TRUMPET—an injective flow based on convolutional layers that are *injective* by construction. Similarly to traditional coupling layers our proposed layers have fast, simple inverses and tractable Jacobians; however, they map to a space of higher dimension. Since they are injective, they can be inverted on their range. Our design combines standard coupling layers with recent results on injective neural networks [Puthawala et al., 2020]. Further, our models can be trained via exact maximum likelihood by separating the training of the injective part from that of the bijective part [Brehmer and Cranmer, 2020].

TRUMPETs can be trained orders of magnitude faster than previous injective models based on traditional normalizing

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

flows [Brehmer and Cranmer, 2020] while producing samples of comparable (or better) quality. Moreover, thanks to their fast inverse, they can be used to design fast inference algorithms based on generative priors. We apply TRUMPETs to Bayesian inference problems in compressive sensing and limited-angle tomography. In particular, we devise an algorithm for efficient computation of a MAP estimator using a variant of projected gradient descent. The projection is computed via the fast inverse while thanks to injectivity we can access the likelihoods. We then adapt recent work on uncertainty quantification for inverse problems with normalizing flows [Sun and Bouman, 2020] to work with generative priors and a low-dimensional latent space of TRUMPETs. We anticipate that neural-network-based uncertainty quantification can be naturally integrated in a rigorous analysis in the context of inverse problems [Mosegaard and Tarantola, 1995, Monard et al., 2020].

Our **main contributions** can be summarized as follows:

- We propose *injective* coupling layers with fast inverses and tractable Jacobians.
- We use these layers to construct TRUMPETs—injective flow generative models. The proposed generative models train orders of magnitude faster than the usual flow models while producing samples of comparable or better quality and giving access to likelihoods.
- We apply the proposed models to Bayesian inference problems and uncertainty quantification, showing remarkable gains in efficiency as well as reconstruction quality over established methods. In particular, we show how the low-dimensional latent space of TRUMPETs leads to an efficient variational approximation of the posterior distribution.

In the following section we describe the construction of TRUMPETs; an overview of related work is given in Section 5.

2 TRUMPETS: INJECTIVE FLOWS

Flow-based generative models [Dinh et al., 2014, 2016] approximate the target distribution via a series of bijective transformations of a simple latent distribution. Unlike GANs [Goodfellow et al., 2014] or VAEs [Kingma and Welling, 2013] they allow for efficient *exact* likelihood evaluation. Crucial to the design of flow-based models are tractable inverses and Jacobians of all the constituent bijective transformations [Kingma and Dhariwal, 2018, Grathwohl et al., 2018], based on special coupling layers such as NICE [Dinh et al., 2014] or Real-NVP [Dinh et al., 2016]. A generative model $f_\theta : \mathbb{R}^D \to \mathbb{R}^D$ parameterized by the weights θ maps latent variables Z to data X. Note that we use uppercase letters for random vectors and corresponding lowercase letters for their realizations. Log-likelihoods of the generated samples $x = f_\theta(z)$ can be evaluated as

$$\log p_X(x) = \log p_Z(f_\theta^{-1}(x)) - \log|\det J_{f_\theta}(f_\theta^{-1}(x))|. \tag{1}$$

Given an iid training dataset $\{\xi^{(i)}\}_{i=1}^n$ from some ground truth distribution[1] p_Ξ, training a normalizing flow entails maximizing the log-likelihood of the training data given as $\sum_{i=1}^N \log p_X(\xi^{(i)})$ over the weights θ in order to learn a generative model f_θ. Equivalently, it entails minimizing the KL divergence between p_X and p_Ξ. While invertibility ensures a non-singular J_{f_θ} at all points, defining likelihoods only requires injectivity of f_θ upto a measure-zero set of non-diferentiable kinks due to non-smooth activation functions.

2.1 MAKING FLOWS INJECTIVE

Machine learning for high-dimensional signals such as images relies on the fact that these signals concentrate around low-dimensional structures. We adopt the common assumption that p_Ξ is concentrated close to a d-dimensional manifold in $\mathbb{R}^D$, with $d \ll D$. We then aim to learn a generative model f_θ, now mapping from $\mathbb{R}^d$ to $\mathbb{R}^D$, such that the observed data lies in the range of f_θ. When f_θ is an injective map its Jacobian $J_{f_\theta} \in \mathbb{R}^{D\times d}$ has full column rank for all input points. Thus one can still have access to likelihoods of samples generated by f_θ by modifying (1) as [Boothby, 1986]

$$\begin{aligned}\log p_X(x) = &\log p_Z(f_\theta^\dagger(x)) \\ &- \frac{1}{2}\log|\det[J_{f_\theta}(f_\theta^\dagger(x))^\mathsf{T} J_{f_\theta}(f_\theta^\dagger(x))]|\end{aligned} \tag{2}$$

which is valid for $x \in \mathrm{Range}(f_\theta)$. We use $f_\theta^\dagger$ to denote an inverse of f_θ on its range, that is $f_\theta^\dagger(f_\theta(z)) = z$. As described later, due to the way we construct $f_\theta^\dagger$, Equation (2) corresponds to the likelihood of a projection of x on the range of f_θ for $x \notin \mathrm{Range}(f_\theta)$.

Building on the general change of variable formula (2), we propose TRUMPET—a network architecture that is injective by construction. The network architecture (Figure 1) consists of a "flat" invertible part which maps $\mathbb{R}^d$ to $\mathbb{R}^d$ and an expanding injective part which maps $\mathbb{R}^d$ to $\mathbb{R}^D$, resembling its namesake in shape. Crucially, expansion is enabled via injective revnet steps [Jacobsen et al., 2018] generalizing the recently proposed Glow [Kingma and Dhariwal, 2018] layers.

We begin by reviewing the revnet step. A forward (F) revnet step has 3 operations, each having a simple inverse (I):

[1] We use ξ to denote samples from the ground truth distribution p_Ξ to distinguish them from the samples x from p_X, the distribution induced by our network f_θ.

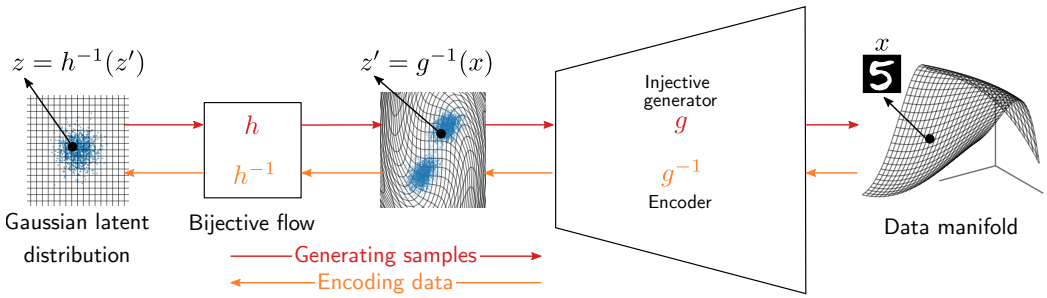


Figure 1: TRUMPET—A reversible injective flow-based generator

1. activation normalization,

$$\text{F: } y = \frac{x - \mu}{\sigma} \qquad \text{I: } x = \sigma y + \mu$$

2. 1×1 convolution with a kernel w,

$$\text{F: } y = \ell_w(x) = w * x \qquad \text{I: } x = w^{-1} * y$$

3. affine coupling layer

$$\begin{array}{lllll} \text{F:} & y_1 = x_1 & y_2 = s(x_1) \circ x_2 + b(x_2) \\ \text{I:} & x_1 = y_1 & x_2 = s(y_1)^{-1} \circ (y_2 - b(y_1)), \end{array}$$

where $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ and $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Here s and b are the scale and bias functions respectively that are implemented by neural networks. The coupling layers have triangular Jacobians making their log determinants easy to compute.

We now generalize the second step to allow for an increase in dimension while retaining computational tractability.

Injective 1×1 convolutions. We consider generalizations of the 1×1 convolution layers (ℓ_w) that (1) are injective, (2) have fast (pseudo)inverse and (3) a fast Jacobian independent of x. These requirements yield two layer variants—linear and ReLU 1×1 convolutions:

	LINEAR	ReLU
FORWARD	$y = w * x$	$y = \text{ReLU}\left(\begin{bmatrix} w \\ -w \end{bmatrix} * x\right)$
INVERSE	$x := w^\dagger * y$	$x := w^\dagger * \left(\begin{bmatrix} I & -I \end{bmatrix} y\right)$.

Here $w^\dagger$ is the left pseudoinverse of w. Since w is a 1×1 convolution, we write it as a matrix of size $c_{\text{out}} \times c$, where c, c_{out} are the number of input and output channels respectively; taking the pseudoinverse of this matrix yields $w^\dagger$.

In Appendix B, we show that for both types of layers,

$$\log\det J_{\ell_w}^\intercal J_{\ell_w} = \sum_{i=1}^{c} s_i(w)^2,$$

where the $s_i(w)$ are the singular values of w. We choose the size of w such that the number of output channels is kc (resp. $\lfloor \frac{k}{2} \rfloor c$) for the linear (resp. ReLU) layer. While $k \geq 1$ is enough for the linear variant to be injective, $k \geq 2$ is necessary and sufficient for the ReLU variant [Puthawala et al., 2020].

Injective revnet step. By generalizing the 1×1 convolutions to increase dimensions, we can still utilize the revnet step as in Glow by replacing the invertible 1×1 convolutions by their injective counterparts.

Therefore, if the input tensor is of size $N \times N \times C$, the output after an injective revnet step is of size $N \times N \times kC$, where the expansion by a factor k occurs in the injective convolution (ℓ_w) step.

2.2 ARCHITECTURE OF TRUMPETS

Injective coupling layers introduced in the previous section allow us to build an architecture that trains at a fraction of the time and memory cost of regular flows. As shown in Figure 1, a TRUMPET model $f_\theta(z) = g_\gamma(h_\eta(z))$ with weights $\theta = (\gamma, \eta)$ has two components: an injective map $g_\gamma(z') = g_1 \circ g_2 \ldots \circ g_K(z')$ which maps from $\mathbb{R}^d$ to $\mathbb{R}^D$, and a bijective part h_η implemented as a flow $z' = h_\eta(z) = h_1 \circ h_2 \ldots \circ h_L(z)$ in the low-dimensional latent space. Unlike normalizing flows such an architecture allows us to progressively increase dimension and markedly reduce the number of parameters.

The role of the injective part g_γ is to match the shape of the manifold that supports the ground truth distribution p_Ξ, while the role of the low-dimensional flow is to match the density on the manifold. As we elaborate in Section 2.3, and as was also recently noted by Brehmer and Cranmer [2020], this separation enables training even when likelihood is not defined for samples outside the range of f_θ.

To build the injective map g_γ we compose the proposed injective revnet layers, progressively increasing dimension from that of the latent space to that of the image space. To improve expressivity, at each resolution, we interleave a small number of bijective revnet layers. Each injective

layer increases feature dimension by a factor of 2 in a single step in the forward direction (and decreases it by a factor of 2 in the reverse direction). Since our latent space is d-dimensional we need $m \approx \log_2 \frac{D}{d}$ injective layers interspersed with a few bijective layers. Following Dinh et al. [2016] we employ upsqueezing to increase resolution. Our network architecture results in significantly fewer parameters and faster training than the recently proposed variant of injective flows [Brehmer and Cranmer, 2020].

Finally, performance of revnets in generative modeling of images can be improved [Dinh et al., 2016] by introducing multiscale implementations of the scale (s) and bias (b) functions. For these implementations, we propose to use U-Nets [Ronneberger et al., 2015] in affine coupling layers as opposed to regular convolutional stacks used in previous normalizing flows [Dinh et al., 2016, Kingma and Dhariwal, 2018]. We find that integrating U-Nets greatly improves the performance of our network.

2.3 TRAINING OF TRUMPETS

An advantage of injective architectures such as TRUMPETs is that they can be trained using maximum likelihood. However, since the range of f_θ is a d-dimensional submanifold in $\mathbb{R}^D$, likelihoods of the samples not on this manifold are not defined. We circumvent this difficulty by splitting the training procedure into two phases—(i) mean squared error (MSE) training phase where we only optimize over the injective part (g_γ) of the network, and (ii) maximum likelihood (ML) training phase where we fit the parameters η of the bijective part h_η so as to maximize the likelihood of the preimage of training data through g_γ; this step matches the density of p_X to that of the ground truth p_Ξ.

The loss function that we minimize to find the parameters of g_γ is given as

$$\mathcal{L}_{\text{MSE}}(\gamma) = \frac{1}{N}\sum_{i=1}^{N} \|\xi^{(i)} - g_\gamma(g_\gamma^\dagger(\xi^{(i)}))\|_2^2 \qquad (3)$$

where $\xi^{(i)}$-s are the training samples. We find that only a few epochs of training are sufficient to train g_γ. Note that $P_{g_\gamma}(x) := g_\gamma(g_\gamma^\dagger(x))$ is an idempotent projection operator on the range of g_γ. The low-dimensional range of g_γ acts as a regularizer in the context of inverse problems. Injectivity implies that the range of f_θ is a true manifold unlike in the case of GANs where it may be an arbitrary low-dimensional structure [Puthawala et al., 2020]. This allows us to define likelihoods as in (2).

After the MSE training phase, we have a manifold that near-interpolates the data samples. In the ML training phase, we match the density (or measure) on the manifold to p_Ξ by maximizing the likelihood of the preimages of the training samples $\{g_\gamma^\dagger(\xi^{(i)})\}$ over η. This gives us the loss function for the ML training phase as

$$\mathcal{L}_{\text{ML}}(\eta) = \frac{1}{N}\sum_{i=1}^{N}\left(-\log p_Z(z^{(i)}) + \sum_{l=1}^{L}\log|\det J_{h_{\eta,l}}|\right), \qquad (4)$$

where $z^{(i)} = h_\eta^{-1}(g_\gamma^\dagger(\xi^{(i)}))$ and $J_{h_{\eta,l}}$ are evaluated at appropriate intermediate inputs. Such a stratified training strategy was proposed recently by Brehmer and Cranmer [2020]. They, however, concatenate regular bijective normalizing flows and pad zeros to the low-dimensional latent codes. This makes their method almost as compute intensive as regular flows.

Stability of layerwise inversions. To minimize $\mathcal{L}_{\text{MSE}}$ (3), we need to calculate the left inverse $g_\gamma^\dagger$ for points that do not lie in the range of g_γ. This entails computing the pseudoinverses of injective convolutional layers ℓ_w. We study the stability of inversion for out-of-range points under the assumption that $y' = \ell_w(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$. In particular, we are interested in estimating the inverse error $E_{\text{Inv}}(y') = \|\ell_w^\dagger(y') - x\|_2^2$ and the re-projection error $E_{\text{Proj}}(y') = \|\ell_w(\ell_w^\dagger(y')) - y'\|_2^2$.

We show in Appendix B that for both linear and ReLU injective convolutions the average errors are

$$\mathbb{E}_\epsilon E_{\text{Inv}}(y) \propto \sigma_\epsilon^2 \sum_{i=1}^{c}\frac{1}{s_i(w)^2}, \qquad \mathbb{E}_\epsilon E_{\text{Proj}}(y) \propto \sigma_\epsilon^2,$$

where $s_i(w)$-s are the singular values of w and c is the number of input channels in the forward direction.

The reconstruction error thus behaves gracefully in σ_ϵ, but could blow up for poorly conditioned w. In order to stabilize inversions and training, we regularize the inverse via Tikhonov regularization. This changes the error terms from $\sum_{i=1}^{c} 1/s_i(w)^2$ to $\sum_{i=1}^{c}\frac{s_i(w)}{s_i(w)^2+\lambda}$ which is upper bounded by $\frac{c}{2\sqrt{\lambda}}$, thus effectively stabilizing training. Here, λ is the regularization parameter.

3 INFERENCE AND UNCERTAINTY QUANTIFICATION WITH TRUMPET

We consider reconstructing an object $x \in \mathbb{R}^D$ from measurements $y \in \mathbb{R}^n$. We assume that x and y are realizations of jointly distributed random vectors X, Y, with the joint distribution $p_{X,Y}(x, y)$. In inference, we are mainly interested in characterizing the posterior $p_{X|Y}(x|y)$. We note that this setting generalizes point estimation of x given y common in inverse problems where the task is to recover x from measurements $y = Ax + \epsilon$. Here ϵ is additive noise and $A \in \mathbb{R}^{n\times D}$ is the forward operator. Examples of forward operators include the subsampled Fourier transform in magnetic resonance imaging (MRI) or a random matrix in

compressed sensing. In many practical problems the number of measurements n is much smaller than the number of unknowns to recover D. In such applications one often computes the maximum a posteriori (MAP) estimate $x_{\text{MAP}} = \text{argmax}_x\, p_{X|Y}(x|y)$; Bayes theorem yields

$$\begin{aligned} x_{\text{MAP}} &= \text{argmin}_x - \log p_{Y|X}(y|x) - \log p_X(x) \\ &= \text{argmin}_x \tfrac{1}{2}\|y - Ax\|_2^2 - \sigma_\epsilon^2 \log p_X(x), \end{aligned} \quad (5)$$

where we assume that $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$.

3.1 MAP ESTIMATION WITH TRUMPET PRIOR

We now address two inference tasks where TRUMPETs are particularly effective. Recall that since g_γ is injective one can build a fast projector $P_{g_\gamma}(x) = g_\gamma(g_\gamma^\dagger(x))$ on the range of g_γ, i.e., the range of our generator.

Beyond simply projecting on the range, injectivity and Bayes theorem enable us to maximize the likelihood of the reconstruction under the posterior induced by the TRUMPET prior [Whang et al., 2020]. The injective flow (iFlow) algorithm described below in Algorithm 1 then alternates projections on the range with gradient steps on the data fidelity term and the prior density. We study two variants—iFlow and iFlow-L that correspond to running Algorithm 1 without and with $-\log p_X$ terms.

Algorithm 1: iFlow

Input: loss function L, y, A, g_γ
Parameter: step size η and $\lambda(\propto \sigma^2)$;

$x^{[0]} = A^\dagger y$;
for $i \leftarrow 0$ **to** $T-1$ **do**
  $v \leftarrow P_g\,(x^{[i]})$;
  $x^{[i+1]} \leftarrow \texttt{GradientStep}\,(L(v))$;
end
$x^{[T]} \leftarrow P_g(x^{[T]})$;

One caveat with computing $-\log p_X(x)$ is that it requires $\log|\det[J_{f_\theta}^\mathsf{T} J_{f_\theta}](f_\theta^\dagger(x))|$ according to (2). While we have layer-wise tractable Jacobians, $\log|\det J_{f_\theta}^\mathsf{T} J_{f_\theta}|$ cannot be split into layerwise $\log\det$ terms due to the change of dimension. Fortunately, the literature is abundant with efficient stochastic estimators. We describe one in Section 3.3 that we use to compare and report likelihoods. In order to implement the iFlow-L, however, we propose a much faster scheme based on a bound.

We show in Appendix B that for an injective function $g : \mathbb{R}^d \mapsto \mathbb{R}^D$, where $g := g_1 \circ g_2 \ldots \circ g_K$, $\log|\det J_g^\mathsf{T} J_g| \leq \sum_{i=1}^K \log|\det J_{g_i}^\mathsf{T} J_{g_i}|$. Thus one gets an upper bound

$$\begin{aligned} &-\log p_X(x) \leq -\log p_Z(f^\dagger(x)) \\ &+ \frac{1}{2}\sum_{k=1}^K \log|\det J_{g_{\gamma,k}}^\mathsf{T} J_{g_{\gamma,k}}| + \sum_{l=1}^L \log|\det J_{h_{\eta,l}}|, \end{aligned} \quad (6)$$

where the layer Jacobians are evaluated at the appropriate intermediate layer outputs. Since all our layers including the injective layers have $\log\det$ Jacobians readily available we use (6) as a proxy for $-\log p_X(x)$. Denoting the right-hand side of (6) by $R(x)$ yields the proposed iFlow-L algorithm (Algorithm 1) for solving (5). The objective function is

$$L(x) := \tfrac{1}{2}\|y - Ax\|_2^2 + \sigma^2 R(x). \quad (7)$$

Note that when solving inverse problems we constrain the final solution x to be in the range of f, that is, $x = f_\theta(z)$ for some $z \in \mathbb{R}^d$.

3.2 POSTERIOR MODELING AND UNCERTAINTY QUANTIFICATION

The second application enabled by TRUMPETs is efficient uncertainty quantification for inverse problems in imaging. We build on a method recently proposed by Sun and Bouman [2020] which computes a variational approximation to the posterior $p_{X|Y}(x|y)$ corresponding to the measurement y and a "classical" regularizer. They train a normalizing flow which produces samples from the posterior, with the prior and the noise model given implicitly by the regularized misfit functional.

The injectivity of the TRUMPET generator f_θ and the assumption that the modeled data concentrates close to the range of f_θ allows us to write the posterior on X, $p_{X|Y}$, in terms of $p_{Z|Y}$, with $X = f_\theta(Z)$. That is,

$$p_{X|Y}(f_\theta(z)|y) = p_{Z|Y}(z|y) \cdot |\det J_{f_\theta}^T J_{f_\theta}|^{-1/2}. \quad (8)$$

We can thus derive a computationally efficient version of the algorithm proposed by Sun and Bouman [2020] by only training a low-dimensional flow.

Instead of using TRUMPETs to simply reduce computational complexity, we showcase another interesting possibility: approximating the posterior with respect to the learned prior given by the TRUMPET. To do this we train another network u_υ which is a low-dimensional flow, so that the distribution of $f_\theta \circ u_\upsilon(T)$ approximates the posterior $p_{X|Y}$ when T is an iid Gaussian vector. The generative process for (approximate) samples from $p_{X|Y}$ is then

$$T \xrightarrow{u_\upsilon} Z \underbrace{\xrightarrow{h_\eta} Z' \xrightarrow{g_\gamma}}_{f_\theta} X.$$

We thus require that $u_\upsilon(T) \sim p_{Z|Y}$ with $T \sim \mathcal{N}(0, I)$ and $X = f_\theta(Z)$. Letting q_υ be the distribution of $u_\upsilon(T)$, the

parameters υ are adjusted by minimizing the KL divergence between q_υ and $p_{Z|Y}$,

$$\begin{aligned}
&\upsilon^* = \operatorname{argmin}_\upsilon \ \mathrm{D}_{\mathrm{KL}}\left(q_\upsilon \,\|\, p_{Z|Y}\right) \\
&= \operatorname{argmin}_\upsilon \ \mathbb{E}_{Z\sim q_\upsilon}[-\log p_{Y|Z}(y|Z) - \log p_Z(Z) + \log q_\upsilon(Z)] \\
&= \operatorname{argmin}_\upsilon \mathbb{E}_{T\sim\mathcal{N}(0,I)}[-\log p_{Y|Z}(y|u_\upsilon(T)) - \log p_Z(u_\upsilon(T)) \\
&\qquad + \log p_T(T) - \log|\det J_{u_\upsilon}(T)|].
\end{aligned} \tag{9}$$

We revisit the inverse problem associated with $y = Ax + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. In this setting we have

$$\begin{aligned}
\upsilon^* = \operatorname{argmin}_\upsilon \mathbb{E}_{T\sim\mathcal{N}(0,I)} \big[&\tfrac{1}{2}\|y - Af_\theta(u_\upsilon(T))\|_2^2 \\
&- \sigma^2 \log p_Z(u_\upsilon(T)) - \sigma^2 \log|\det J_{u_\upsilon}(T)|\big].
\end{aligned} \tag{10}$$

We evaluate (10) by drawing k iid samples $\{t_i\}_{i=1}^k$ from the base Gaussian, yielding the following loss to train u_υ,

$$\begin{aligned}
\mathcal{L}(\upsilon) := \frac{1}{k}\sum_{i=1}^{k} &(\|y - Af_\theta(u_\upsilon(t_k))\|_2^2 \\
&- \sigma^2 \log p_Z(u_\upsilon(t_k)) - \beta\sigma^2 \log|\det J_{u_\upsilon}(t_k)|),
\end{aligned} \tag{11}$$

where we added β as a hyper-parameter to control the diversity of samples we generate from the posterior [Sun and Bouman, 2020].

3.3 ESTIMATING LOG-LIKELIHOODS

The training of TRUMPETs only requires the log det of the Jacobian of h_η. Some applications call for the log det of the Jacobian of the full network, typically evaluated a small number of times. Here, we provide a stochastic estimate via the truncation of a Neumann series.

As $J_{f_\theta}^\mathsf{T} J_{f_\theta}$ is a square matrix, we find that

$$\begin{aligned}
\log|\det J_{f_\theta}^\mathsf{T} J_{f_\theta}| &= \operatorname{Tr}(\log J_{f_\theta}^\mathsf{T} J_{f_\theta}) \\
&= \operatorname{Tr}\left(\log\frac{1}{\alpha}(I - (I - \alpha J_{f_\theta}^\mathsf{T} J_{f_\theta}))\right) \\
&= -\operatorname{Tr}\left(\sum_{k=1}^{\infty}\frac{(I - \alpha J_{f_\theta}^\mathsf{T} J_{f_\theta})^k}{k}\right) - d\log\alpha \\
&\approx -\mathbb{E}_v \sum_{k=1}^{n}\frac{1}{k} v^\mathsf{T}(I - \alpha J_f^\mathsf{T} J_f)^k v - d\log\alpha
\end{aligned}$$

where we choose α such that the maximal singular value of $I - \alpha J_{f_\theta}^\mathsf{T} J_{f_\theta}$ is about 0.1. This ensures that the series converges fast and we can truncate the expansion to about 10 terms. We estimate the largest singular value of $J_{f_\theta}^\mathsf{T} J_{f_\theta}$ using power iteration. In the last step we use the Hutchinson trace estimator [Hutchinson, 1989] to evaluate the trace. Here, vs are sampled from $\mathcal{N}(0, I)$. The terms of the power series can be efficiently implemented by vector-Jacobian and Jacobian-vector products using automatic differentiation as described in Algorithm 2 Chen et al. [2019].

Algorithm 2: Stochastic $\log\det$ Jacobian estimator

Input: f, n
Output: $\log|\det J_f^\mathsf{T} J_f|$

$\log\det = 0$
$\beta = 0.9\,(\texttt{MaxSingularValue}(J_f))^{-1}$;
Draw v from $\mathcal{N}(0, I)$;
$w^\mathsf{T} = v^\mathsf{T}$;
for *k=1* **to** *n* **do**
  $u_1^\mathsf{T} = \texttt{jvp}(w)$;
  $u_2^\mathsf{T} = \texttt{vjp}(u_1)$;
  $w = w - \beta u_2$;
  $\log\det \mathrel{-}= \dfrac{w^\mathsf{T} v}{k}$;
end
$\log\det \mathrel{-}= d\log\beta$

4 COMPUTATIONAL EXPERIMENTS WITH IMAGING PROBLEMS

We begin by evaluating the generative performance of TRUMPETs. Next, we test TRUMPETs on two inference tasks in imaging: maximum a posteriori estimation and uncertainty quantification. .

4.1 GENERATIVE MODELING

We train TRUMPETs on the MNIST [LeCun et al., 1998], CIFAR10 [Krizhevsky et al., 2009], CelebA [Liu et al., 2015] and Chest X-ray [Wang et al., 2017] datasets with image sizes $32 \times 32 \times 1$, $32 \times 32 \times 3$, $64 \times 64 \times 3$ and $128 \times 128 \times 1$ respectively.

We find that our networks train much faster than invertible flows and their recent injective generalizations [Brehmer and Cranmer, 2020]. As a point of comparison, training the models of Brehmer and Cranmer [2020] takes over 10 days on the CelebA dataset. The corresponding TRUMPET trains in 38 hours while yielding better samples in terms of the Fréchet inception distance (FID) [Heusel et al., 2017] (see Table 1). [2]

Since the range of a TRUMPET is a manifold, a relevant metric is the reconstruction error, $\frac{\|\xi - f_\theta(f_\theta^\dagger(\xi))\|}{\|\xi\|}$, which we report for ξs in the test set in Table 2. We share generated samples and reconstructions on test sets from trained TRUMPETs in Figures 6b, 7b, 8 and 9 in Appendix C.

We note that the variants with the linear and ReLU 1×1 convolutions perform similarly (see Figures 6a,6b, 7a, 7b);

[2] Our FID scores are reported at sampling temperature $T = 1$, that is, we use the same prior distribution statistics for training and sampling. We show the variation of the FID metric with the temperature in Figure 5 in Appendix C

Table 1: FID scores on 8-bit 64×64 celebA dataset.

Model	FID
Kumar et al. [2020]	40.23
Brehmer and Cranmer [2020]	37.4
TRUMPET (Ours)	**34.3**

Table 2: Training times in hours for TRUMPET: all models were trained on a single V100 GPU

	Training time (hours)	$\frac{\|x - f_\theta(f_\theta^\dagger(x))\|}{\|x\|}$	Trainable params
MNIST	11	0.04	9M
CIFAR10	11	0.22	9M
CelebA	38	0.15	16M
Chest X-ray	25	0.13	11M

hence, for the subsequent datasets and experiments we only report results with the linear variant.

The negative log-likelihood values estimated for trained TRUMPET models using Algorithm 2 on the $[-1, 1]$ normalized MNIST and CelebA dataset are 114.82 ± 5.8 and 294 ± 7.4 nats respectively. Note that these represent likelihoods over measures supported in a d-dimensional latent space whereas the previous literature [Kingma and Dhariwal, 2018, Dinh et al., 2016] reports D-dimensional likelihoods. This discrepancy is unfortunately not easily resolved by simply dividing by dimension. We note that in fact, such a comparison may at best be ambiguous and at worst be misleading.

4.2 MAP ESTIMATION

We test TRUMPETs on image reconstruction from compressive measurements. We work with four different forward operators / corruption models: (i) **RandGauss (m)**: we sample an entrywise iid Gaussian matrix $A \in \mathbb{R}^{n \times D}$, where $n = 250$ and D is the dimension of the vectorized image; (ii) **RandMask (p)**: we mask pixels (that is, replace a pixel with zero) with probability $p = 0.15$; (iii) **Super-resolution (x4)**: we downsample the image by a factor of 4 along each dimension; and (iv) **Mask (s)**: we mask (replace with zero) an $s \times s$-size portion of the image.

d

Since TRUMPETs have a readily available inverse we focus on the benefits this brings in imaging. Specifically, we use Algorithm 1 to compute an estimate using a trained TRUMPET prior. We test the algorithm on the MNIST and CelebA datasets and use the same TRUMPET prior for all problems. We compare our approach to two deep learning baselines—compressed sensing with generative models (CSGM) [Bora et al., 2017] and deep image prior (DIP) [Ulyanov et al., 2018]. CSGM solves $\hat{x} = f(\text{argmin}_z \|y - Af(z)\|_2^2)$ while DIP solves $\hat{x} = f_\theta(\text{argmin}_\theta \|y - Af_\theta(z)\|_2^2)$ given a randomly chosen fixed z and regularized by early stopping. Figure 2 compares all methods for the superresolution and random masking problems on the CelebA dataset while Table 3 gives a comprehensive evaluation for all inverse problems.

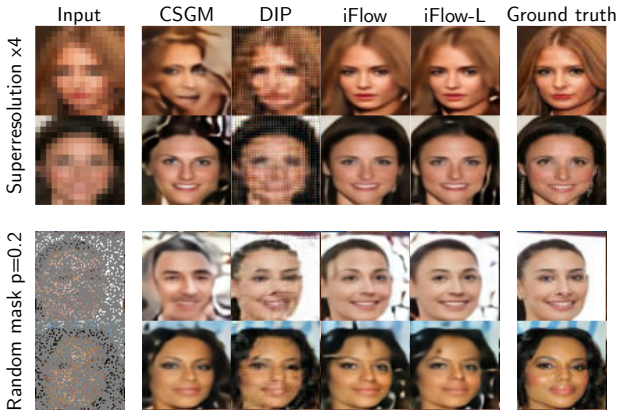


Figure 2: Comparison of various reconstruction schemes. The iFlow-L and iFlow methods refer to Algorithm 1 respectively with and without the likelihood term.

Table 3: Performance on inverse problems measured in reconstruction SNR (dB)

	Dataset	CSGM	DIP	iFlow	iFlow-L
RandGauss ($m = 250$)	MNIST	11.32	12.72	21.34	**21.81**
	CelebA	8.98	**11.25**	8.90	8.91
RandMask ($p = 0.15$)	MNIST	3.85	4.94	4.76	**10.10**
	CelebA	12.63	**17.26**	13.89	14.43
Super-resolution ($\times 4$)	MNIST	5.943	1.0	9.851	**12.75**
	CelebA	11.08	14.12	17.36	**20.07**
Mask ($s = 15$ px)	MNIST	3.34	4.38	3.90	**9.54**
	CelebA	13.42	21.31	21.74	**21.79**
Limited-view CT	Chest	11.58	13.76	20.93	**21.23**

We also perform an ablation study to assess the influence of including the prior likelihood as opposed to simply doing a gradient descent with manifold projections [Raj et al., 2019]. The latter corresponds to setting $\lambda = 0$ in Algorithm 1. Table 3 clearly shows that accounting for the prior density and not only support—that is, computing the MAP estimate—performs better in almost all settings.

We mention that we attempted to compare with a method involving projections proposed by Shah and Hegde [2018] but found it to be $50-100\times$ slower than iFlow. It was thus infeasible to finalize this comparison. On average we found that DIP converged the fastest followed by our method followed which was about $2\times$ slower. Finally, while each iteration of

CSGM was as fast as each of DIP, CSGM requires several restarts which made the method about $4\times$ slower than ours. We report the best results from CSGM with 10 restarts.

Note that the baselines [Bora et al., 2017, Ulyanov et al., 2018, Shah and Hegde, 2018] were developed without injectivity as a constraint. As a result they typically use off-the-shelf GAN architectures inspired by [Radford et al., 2015], but they are by design agnostic to architectural details. Therefore, in order to keep the comparisons fair, we use the same generative model f_θ for all methods. This allows us to test the importance of tractable inverses and likelihoods for the design of image reconstruction algorithms based on generative priors.

4.3 POSTERIOR MODELING AND UNCERTAINTY QUANTIFICATION

Next, we use TRUMPET priors for uncertainty quantification in computed tomography. We work with a chest X-ray dataset and use the limited-angle CT operator as the forward operator, A. We choose a sparse set of $n_{\text{angles}} = 30$ view angles from 30° to 150°, with a 60° missing cone. We add 30dB noise to the measurements. The resulting inverse problem is severely ill-posed and solving it requires regularization. (Note that Table 3 includes the performance of Algorithm 1 on this problem.)

Here we provide a pixel-wise uncertainty estimate of the form $\mathbb{E}_{X\sim p_{X|Y=y}}|X - \langle X\rangle|^p$, with $p = 1, 2$, $|\cdot|$ the pixel-wise absolute value, and $\langle X\rangle$ the posterior mean. In Figure 3, we show the MAP estimate obtained from the iFlow-L algorithm (Algorithm 1). We also show the Fourier spectrum of the mean absolute deviation calculated in the Fourier domain where the mean was calculated over the Fourier transform of all samples from the posterior. We observe a cone of increased uncertainty in the Fourier spectrum that corresponds to the missing angles in the limited-view CT operator. Furthermore, we observe a thick vertical bright line that corresponds to uncertainty in predicting the location of the ribs (which have a strong horizontal periodic component) as shown in the middle plot of Figure 3.

Reassuringly, both the spatial- and the frequency-domain representations of uncertainty correlate well with our intuitive expectations for this problem. Positions of the ribs in space and the missing cone in the spectrum exhibit higher uncertainty.

5 RELATED WORK

Normalizing flows have been introduced in [Dinh et al., 2014]. The key to their success are invertible coupling layers with triangular Jacobians. Different variants of the coupling layer along with multiscale architectures [Dinh et al., 2016, Kingma and Dhariwal, 2018, Grathwohl et al., 2018] have considerably improved performance of normalizing flows. Glow [Kingma and Dhariwal, 2018] uses invertible 1×1 convolutions to improve expressivity, producing better samples than NICE and Real-NVP. Alas, training a Glow model is extremely compute intensive—1 week on 40 GPUs for the 5-bit 256×256 CelebA dataset. A crucial drawback of the mentioned models is that they are bijective so the dimension of the latent and data spaces coincide. This results in a large number of parameters and slow training: since the ground data lies close to low-dimensional subset of $\mathbb{R}^D$, training should encourage the model to become "almost non-invertible" which makes the optimization more difficult.

Kumar et al. [2020] propose approximate injective flows by using spectral regularization in auto-encoders. However they lack access to likelihoods. Further, their training strategy is only a proxy for injectivity. Very recently, Brehmer and Cranmer [2020] proposed injective flows to learn a data distribution on a manifold very similar to our work, including a two-stage training scheme we use. However, they use regular normalizing flow architectures with zero padding in the latent space which results in architectures that are very expensive to train. Cunningham et al. [2020] build injective flows by adding noise to the range; this requires stochastic inversion whereas ours is deterministic.

In a parallel development, autoregressive flows were shown to have favorable expressivity compared to normalizing flows. We refer to Papamakarios et al. [2017], Kingma et al. [2016], Oord et al. [2016] and the references therein for a more extensive account.

6 DISCUSSION AND CONCLUSION

We proposed TRUMPETs—a flow-based generative model that is injective by construction. TRUMPETs alleviate the main drawback of invertible normalizing flows which is that they are very expensive to train. We showed that TRUMPETs are competitive in terms of generative modeling performance and that the fast inverse on the range markedly improves reconstructions in ill-posed inverse problems. We also showed how to use TRUMPETs to model posteriors and perform uncertainty quantification directly in the low-dimensional latent space. Currently our reconstructions on data lack high frequency features. This is common in normalizing flow models [Dinh et al., 2016]. Strategies such as adding the adversarial loss in the MSE phase of training may help alleviate this drawback. Furthermore, using a richer class of coupling layers may help— Durkan et al. [2019] show that flows based on rational quadratic splines are more expressive. Integrating such layers also holds promise for improving the expressivity of TRUMPETs.

Our work combines a number of basic ideas in an intuitive way that yields gains in efficiency and accuracy. Additionally, recent results on universality of globally injective neu-

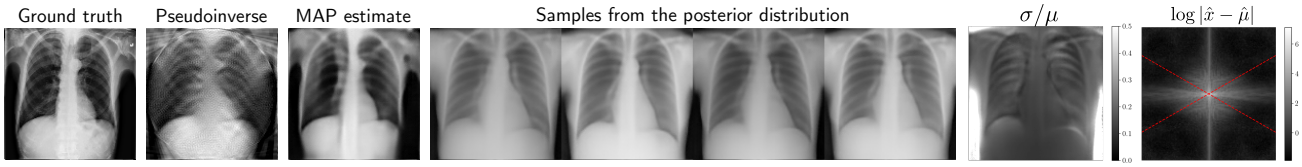


Figure 3: Uncertainty quantification for limited view CT.

ral networks [Puthawala et al., 2020] and universality of flows [Teshima et al., 2020] suggest that TRUMPETs are universal approximators of measures concentrated on Lipschitz manifolds; a rigorous proof is left to future work.

ACKNOWLEDGEMENTS

MVdH gratefully acknowledges support from the Department of Energy under grant DE-SC0020345, the Simons Foundation under the MATH + X program, and the corporate members of the Geo-Mathematical Imaging Group at Rice University. ID and AK were supported by the European Research Council Starting Grant 852821—SWING.

References

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

William M Boothby. *An introduction to differentiable manifolds and Riemannian geometry*. Academic press, 1986.

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

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913*, 2020.

Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019.

Edmond Cunningham, Renos Zabounidis, Abhinav Agrawal, Ina Fiterau, and Daniel Sheldon. Normalizing flows across dimensions. *arXiv preprint arXiv:2006.13070*, 2020.

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.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *arXiv preprint arXiv:1906.04032*, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27: 2672–2680, 2014.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

Alfred Horn. On the singular values of a product of completely continuous operators. *Proceedings of the National Academy of Sciences of the United States of America*, 36 (7):374, 1950.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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. *URL http://arxiv. org/abs/1606.04934*, 2016.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Abhishek Kumar, Ben Poole, and Kevin Murphy. Regularized autoencoders via relaxed injective probability flow. *arXiv preprint arXiv:2002.08927*, 2020.

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

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

François Monard, Richard Nickl, and Gabriel P Paternain. Consistent inversion of noisy non-abelian x-ray transforms. *Communications on Pure and Applied Mathematics*, 2020.

Klaus Mosegaard and Albert Tarantola. Monte carlo sampling of solutions to inverse problems. *Journal of Geophysical Research: Solid Earth*, 100(B7):12431–12447, 1995.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/oord16.html`.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Michael Puthawala, Konik Kothari, Matti Lassas, Ivan Dokmanić, and Maarten de Hoop. Globally injective relu networks. *arXiv preprint arXiv:2006.08464*, 2020.

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

Ankit Raj, Yuqi Li, and Yoram Bresler. Gan-based projector for faster recovery with convergence guarantees in linear inverse problems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5602–5611, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Viraj Shah and Chinmay Hegde. Solving linear inverse problems using gan priors: An algorithm with provable guarantees. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4609–4613. IEEE, 2018.

He Sun and Katherine L Bouman. Deep probabilistic imaging: Uncertainty quantification and multi-modal solution characterization for computational imaging. *arXiv preprint arXiv:2010.14462*, 2020.

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. *arXiv preprint arXiv:2006.11469*, 2020.

Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2020.

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

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.

Jay Whang, Qi Lei, and Alexandros G Dimakis. Compressed sensing with invertible generative models and dependent noise. *arXiv preprint arXiv:2003.08089*, 2020.

A NETWORK ARCHITECTURE AND TRAINING DETAILS

We describe the injective portion of our network architecture that was used to train a CelebA dataset in Figure 4. The bijective revnet block has 3 bijective revnet steps in each block while the injective revnet block has just one injective revnet step which is explained in details in Section 2.1. The bijective part of our network is not shown in Figure 4 but it has 32 bijective revenet steps.

For the scale and bias terms of the coupling layer we used the U-Net architecture with 2 downsampling blocks and 2 corresponding upsampling blocks. Each resolution change is preceded by 2 convolution layers with 32 and 64 output channels. We choose the latent space dimension as 64 for MNIST, 256 for Chest X-ray dataset and 192 for all other datasets. We normalize the data to lie in $[-1, 1]$.

The number of training samples for CelebA, Chest X-ray, MNIST and CIFAR10 are 80000, 80000, 60000, and 50000 respectively. We trained all models for about 300 epochs with a batch size of 64.

All models are trained with Adam optimizer [Kingma and Ba, 2014] with learning rate 10^{-4}. $\gamma = 10^{-6}$ was used as the Tikhonov regularizer parameter for computing pseudoinverse of injective convolutional layers.

B DERIVATIONS OF ERROR AND LIKELIHOOD BOUNDS

B.1 BOUNDING LOG-LIKELIHOOD FOR INJECTIVE FUNCTIONS

Claim 1. *For an injective function $f = f_1 \circ f_2 \circ \dots f_k(z)$ that maps $z \in \mathbb{R}^d$ to $x \in \mathbb{R}^D$,*

$$\log |\det J_f^\mathsf{T} J_f| \leq \sum_{i=1}^{K} \log |\det J_{f_k}^\mathsf{T} J_{f_k}|$$

Proof. We demonstrate the claim for 3 layers; the general statement follows by induction. Consider $x = f(z) = f_1 \circ f_2 \circ f_3(z)$, where $x \in \mathbb{R}^D$ and $z \in \mathbb{R}^d$, $d < D$. Assume that $f_1 : \mathbb{R}^D \mapsto \mathbb{R}^D$, $f_3 : \mathbb{R}^d \mapsto \mathbb{R}^d$ are bijective and $f_2 : \mathbb{R}^d \mapsto \mathbb{R}^D$ is injective. Then

$$J_f = \underbrace{\frac{\partial f_1}{\partial f_2}}_{J_1} \underbrace{\frac{\partial f_2}{\partial f_3}}_{J_2} \underbrace{\frac{\partial f_3}{\partial z}}_{J_3}$$

and we have

$$\begin{aligned} \log\det|J_f^\mathsf{T} J_f| &= \log\det|J_3^\mathsf{T} J_2^\mathsf{T} J_1^\mathsf{T} J_1 J_2 J_3| \\ &= 2\log\det|J_3| + \log\det|J_2^\mathsf{T} J_1^\mathsf{T} J_1 J_2|. \end{aligned} \tag{12}$$

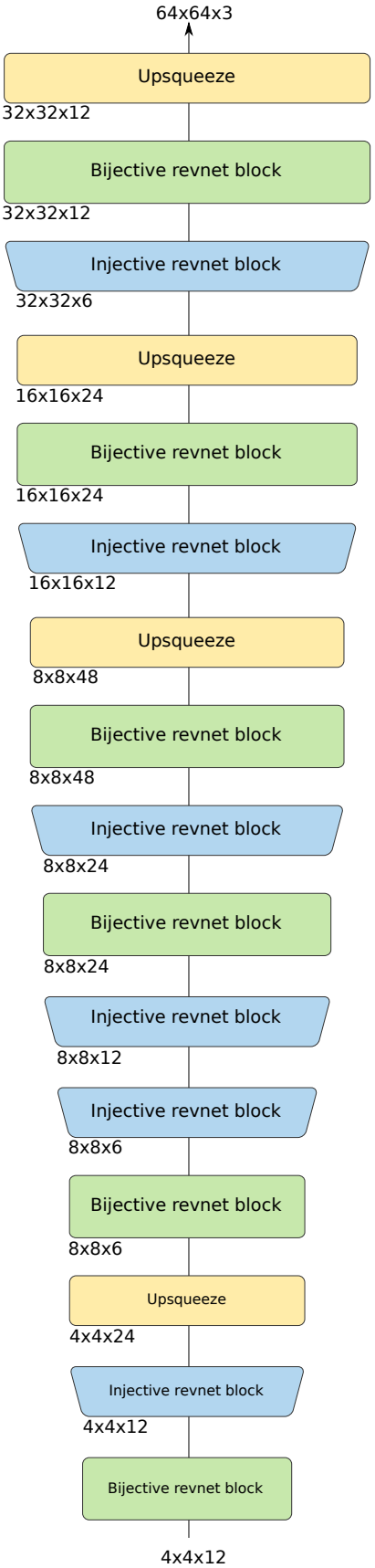


Figure 4: CelebA architecture for the injective portion g of TRUMPET. The input size to each layer is written below it.

Let now $J_1 = U_1\Sigma_1 V_1^\mathsf{T}$ and $J_2 = U_2\Sigma_2 V_2^\mathsf{T}$. We can compute as

$$\begin{aligned}\log|\det J_2^\mathsf{T} J_1^\mathsf{T} J_1 J_2| &= \log|\det V_2\Sigma_2 U_2^\mathsf{T} V_1\Sigma_1 U_1^\mathsf{T} U_1\Sigma_1 V_1^\mathsf{T} U_2\Sigma_2 V_2^\mathsf{T}| \\ &= \log|V_2\Sigma_2 U_2^\mathsf{T} V_1\Sigma_1^2 V_1^\mathsf{T} U_2\Sigma_2 V_2^\mathsf{T}| \\ &= 2\log|\det\Sigma_2| + \log|\det V_2^\mathsf{T} U_1\Sigma_1^2 U_1^\mathsf{T} V_2| \\ &\le 2\log|\det\Sigma_2| + 2\log|\det\Sigma_1| \\ &= \log|\det J_2^\mathsf{T} J_2| + \log|\det J_1^\mathsf{T} J_1| \end{aligned} \tag{13}$$

where we used that $\Pi_{i=1}^n \lambda_i(UHU^\mathsf{T}) \le \Pi_{i=1}^n \lambda_i(H)$ for any symmetric matrix H and unitary matrix U (Horn [1950]). Here $\lambda_i(M)$ is the ith eigenvalue of matrix M.

Substituting (13) in (12) we obtain,

$$\log|\det J_f^\mathsf{T} J_f| \le \sum_{i=1}^{3} \log|\det J_{f_k}^\mathsf{T} J_{f_k}|,$$

which establishes the claim. □

B.2 MEASURING ERROR DUE TO DEVIATIONS FROM RANGE

Claim 2. *Consider $y' = y + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I)$, $y = \ell_w(x)$ and let $E_{\text{Inv}}(y') := \|\ell_w^\dagger(y') - x\|_2^2$ and the re-projection error $E_{\text{Proj}}(y') := \|\ell_w(\ell_w^\dagger(y')) - y'\|_2^2$. Then for both* ReLU *and linear variants of ℓ_w we have*

$$\mathbb{E}_\epsilon E_{\text{Inv}}(y') \propto \sigma_\epsilon^2 \sum_{i=1}^{c} \frac{1}{s_i(w)^2}, \qquad \mathbb{E}_\epsilon E_{\text{Proj}}(y') \propto \sigma_\epsilon^2,$$

where $s_i(w)$'s are the singular values of w and c is the number of input channels in the forward direction.

Proof. Consider $y' = y + \epsilon$, where $y = \ell_w(x)$ and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2 I_{2n})$. We consider a vectorized x and write the 1×1 convolution as a matrix-vector product, Wx say. For a ReLU injective convolution one could write the inverse as

$$x' = W^\dagger \begin{bmatrix} I_n & -I_n \end{bmatrix} y'. \tag{14}$$

We calculate $\mathbb{E}_\epsilon\|x' - x\|_2^2$. Let $M := \begin{bmatrix} I_n & -I_n \end{bmatrix}$ and $B := W^\dagger$, then

$$\begin{aligned} x' &= BM(y + \epsilon) \\ x' - x &= BM\epsilon, \end{aligned}$$

whence

$$\begin{aligned} \|x' - x\|_2^2 &= (BM\epsilon)^\mathsf{T} BM\epsilon \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(BM\epsilon(BM\epsilon)^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(BM\epsilon\epsilon^\mathsf{T} M^\mathsf{T} B^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(M^\mathsf{T} B^\mathsf{T} BM\epsilon\epsilon^\mathsf{T}\right) \end{aligned}$$

so that

$$\begin{aligned} \mathbb{E}_\epsilon\|x' - x\|_2^2 &= \mathbb{E}_\epsilon \operatorname{Tr}\left(M^\mathsf{T} B^\mathsf{T} BM\epsilon\epsilon^\mathsf{T}\right) \\ \mathbb{E}_\epsilon\|x' - x\|_2^2 &= \operatorname{Tr}\left(M^\mathsf{T} B^\mathsf{T} BM\right)\sigma_\epsilon^2 \\ \mathbb{E}_\epsilon\|x' - x\|_2^2 &= 2\operatorname{Tr}\left(B^\mathsf{T} B\right)\sigma_\epsilon^2 \\ \mathbb{E}_\epsilon\|x' - x\|_2^2 &= 2\sum_{i=1}^{c} s_i(w)^{-2}\sigma_\epsilon^2. \end{aligned}$$ □

Similarly for a linear layer the inverse is given as $x' = By'$. Therefore,

$$\begin{aligned} x' &= B(y + \epsilon) \\ x' - x &= B\epsilon \end{aligned}$$

whence

$$\begin{aligned} \|x' - x\|_2^2 &= (B\epsilon)^\mathsf{T} B\epsilon \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(B\epsilon(B\epsilon)^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(B\epsilon\epsilon^\mathsf{T} B^\mathsf{T}\right) \\ \|x' - x\|_2^2 &= \operatorname{Tr}\left(B^\mathsf{T} B\epsilon\epsilon^\mathsf{T}\right) \end{aligned}$$

so that

$$\mathbb{E}_\epsilon\|x' - x\|_2^2 = \sum_{i=1}^{c} s_i(w)^{-2}\sigma_\epsilon^2. \qquad \square$$

The re-projection error for a ReLU layer is given as

$$\begin{aligned} \sqrt{E_{\text{Proj}}(y')} &= \left\|\text{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix} x'\right) - y\right\| \\ &= \left\|\text{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix} x'\right) - \text{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix} x\right) - \epsilon\right\| \\ &\le \left\|\begin{bmatrix} W \\ -W \end{bmatrix} x' - \begin{bmatrix} W \\ -W \end{bmatrix} x\right\| + \|\epsilon\| \\ &= \left\|\begin{bmatrix} W \\ -W \end{bmatrix} (x + BM\epsilon) - \begin{bmatrix} W \\ -W \end{bmatrix} x\right\| + \|\epsilon\| \\ &= \left\|\begin{bmatrix} W \\ -W \end{bmatrix} BM\epsilon\right\| + \|\epsilon\| \\ &\le (2\|WW^\dagger\|_F + 1)\|\epsilon\| \end{aligned}$$

Squaring both sides, we get

$$E_{\text{Proj}}(y') = (2\sqrt{c} + 1)^2\|\epsilon\|^2. \qquad \square$$

Similarly, for a linear layer we have

$$\begin{aligned} E_{\text{Proj}}(y') &= \|Wx' - Wx - \epsilon\|^2 \\ &= \left\|WW^\dagger\epsilon - \epsilon\right\|^2 \\ &= (c + 1)\left\|\epsilon\right\|^2. \end{aligned}$$

□

B.3 LOG-DETERMINANTS OF JACOBIANS FOR RELU INJECTIVE CONVOLUTIONS

We vectorize x and, again, write the 1×1 convolution as a matrix-vector product Wx. Then, for a ReLU 1×1 convolution, we have

$$y = \mathrm{ReLU}\left(\begin{bmatrix} W \\ -W \end{bmatrix}\right) x.$$

This could be trivially rewritten as $y = W'x$, where the rows of W' are $w'_i = w_i$ if $\langle w_i, x \rangle > 0$ and $w'_i = -w_i$ otherwise. We note that changing the row signs does not change $|\det W|$. Hence, for such a ReLU injective convolutional layer, $\ell_w \log |\det J_{\ell_w}^T J_{\ell_w}| = \sum_{i=1}^{c} s_i^2(w)$, where $s_i(w)$'s are the singular values of w, where w is the 1×1 kernel corresponding to the convolution matrix W.

C SAMPLES

In Figures 6a, 6b and Figures 7a, 7b we compare the performance of TRUMPETs trained with ReLU and linear injective convolutions on the MNIST and 64×64 CelebA datsets. Both variants offer similar performance hence we choose to use linear convolutions for the rest of our results regarding inverse problems and uncertainty quantification. In Figures 9 and 8 we show generated samples from TRUMPET and a few reconstructions of original samples, x given as $f(f^{\dagger}(x))$ on the CIFAR10 and Chest X-ray datasets respectively. For the CIFAR10 dataset, we do see a low frequency bias in the generated samples. We hope to rectify this as per our discussions in Section 6. For other datasets the low-frequency bias seems to be less of a problem. In fact, on these datasets TRUMPETs outperform previous injective variants of flows [Brehmer and Cranmer, 2020, Kumar et al., 2020].

The temperature of sampling has a significant effect on the FID scores as shown in Figure 5. While samples in Figures 7a, 7b are for $T = 1$ we share some samples in Figure 10 for $T = 0.85$.

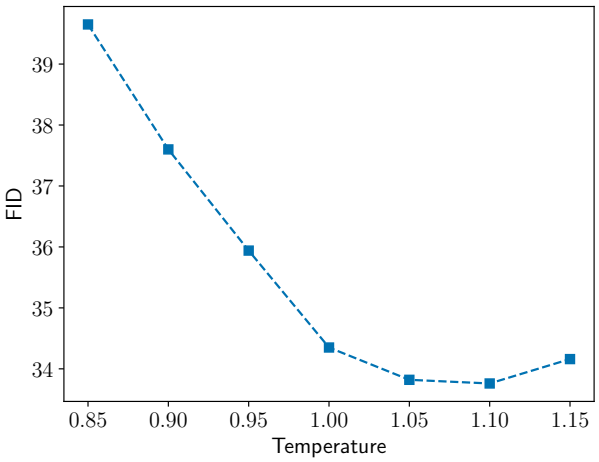
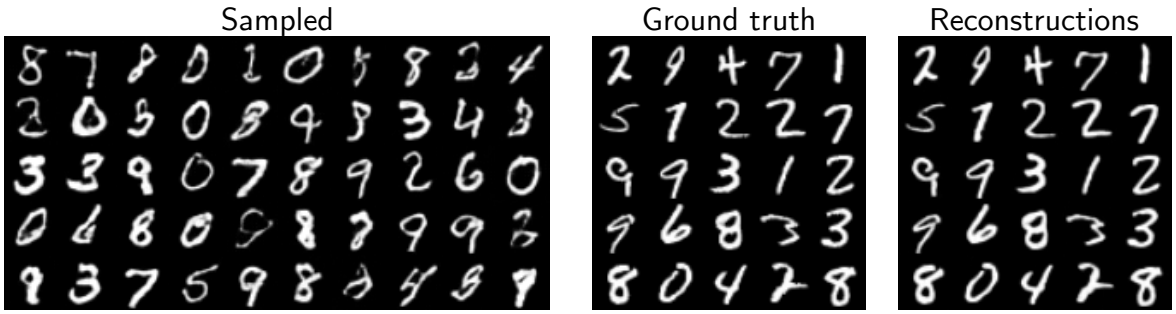
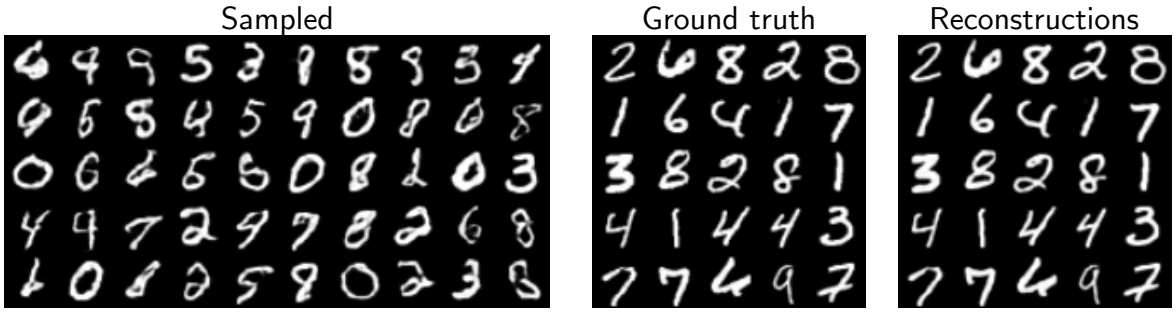


Figure 5: FID score of TRUMPET with sampling temperature.



(a) ReLU 1×1 convolutions



(b) Linear 1×1 convolutions

Figure 6: TRUMPETs trained with (a) ReLU and (b) linear 1×1 convolutions give similar sample quality.

(a) ReLU 1×1 convolutions

(b) Linear 1×1 convolutions

Figure 7: TRUMPETs trained with (a) ReLU and (b) linear 1×1 convolutions give similar sample quality. On the right, we showcase the reconstruction performance—the left column is ground truth and the right is our reconstruction (see Table 2 for quantitative assessment)

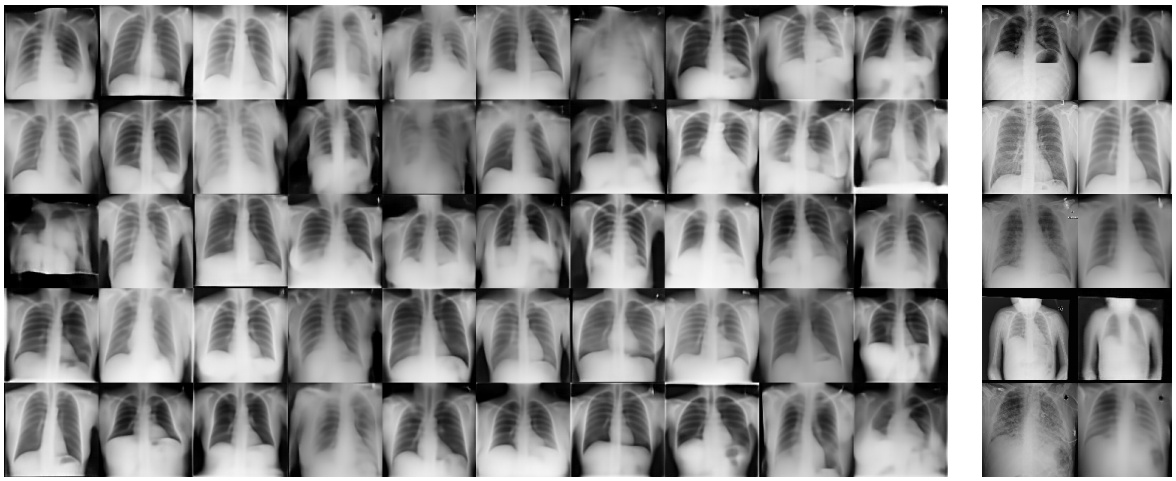


Figure 8: Generated samples on the Chest X-ray. On the right, we showcase the reconstruction performance—the left column is ground truth and the right is our reconstruction (see Table 2 for quantitative assessment)

Figure 9: Generated samples and reconstructions of original data on the CIFAR-10 dataset.

Figure 10: Generated samples on the celeba dataset with linear 1×1 convolution and $T = 0.85$.