TRUMPETS: Injective Flows for Inference and Inverse Problems

Konik Kothari  AmirEhsan Khorashadizadeh  Maarten de Hoop  Ivan Dokmanić

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 an order of magnitude faster than earlier 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 fast inverse yields a projection while thanks to injectivity we can compute 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 and Bouman, 2020] to work with generative priors and a likelihood. 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 and Bouman, 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 and 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 injective 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 injective 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 \( \theta \) 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

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

(1)

Given an iid training dataset \( \{\xi(i)\}_{i=1}^N \) from some ground truth distribution \( p_\Xi \), 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 \( \theta \) in order to learn a generative model \( f_\theta \). Equivalently, it entails minimizing the KL divergence between \( p_X \) and \( p_\Xi \). While invertibility ensures a non-singular \( J_{f_\theta} \) at all points, defining likelihoods only requires injectivity of \( f_\theta \).

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 \( p_\Xi \) is concentrated close to a \( d \)-dimensional manifold in \( \mathbb{R}^D \), with \( d \ll D \). We then aim to learn a generative model \( f_\theta \), now mapping from \( \mathbb{R}^d \) to \( \mathbb{R}^D \), such that the observed data lies in the range of \( f_\theta \). When \( f_\theta \) 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 \( f_\theta \) by modifying (1) as [Boothby, 1986]

\[
\log p_X(x) = \log p_Z(f_\theta^\dagger(x)) - \frac{1}{2} \log |\det[J_{f_\theta}(f_\theta^\dagger(x))]|, \tag{2}
\]

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

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 \( \xi \) to denote samples from the ground truth distribution \( p_\Xi \) to distinguish them from the samples \( x \) from \( p_X \), the distribution induced by our network \( f_\theta \).

2 With (Le)ReLU activations, Jacobians are defined “only” almost everywhere; this rarely (if ever) causes issues in practice.
We now generalize the second step to allow for an increase in dimension while retaining computational tractability.

Injective \(1 \times 1\) convolutions. We consider generalizations of the \(1 \times 1\) convolution layers \(\ell_w\) 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 \(1 \times 1\) convolutions:

**LINEAR**

Forward: \(y = w \ast x\),

Inverse: \(x := w^\dagger \ast y\), \(x := w^\dagger \ast ([I - I] y)\).

**ReLU**

Forward: \(y = \text{ReLU}(\begin{bmatrix} w \\ -w \end{bmatrix} \ast x)\),

Inverse: \(x := w^\dagger \ast y\), \(x := w^\dagger \ast ([I - I] y)\).

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

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

\[
\log \det J_{\ell_w}^\top J_{\ell_w} = \sum_{i=1}^c s_i(w)^2,
\]

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. \([k/2]c\)) 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 \(1 \times 1\) convolutions to increase dimensions, we can still utilize the revnet step as in Glow by replacing the invertible \(1 \times 1\) 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 \(\ell_w\) 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 \(h_\eta\) 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 \(g_\gamma\) is to match the shape of the manifold that supports the ground truth distribution \(p_{\mathbf{z}}\), while the role of the low-dimensional flow is to match the density on the manifold. As recently proposed by Brehmer and Cranmer [2020] and as we elaborate in Section 2.3 this separation enables training even though likelihood is not defined for samples outside the range of \(f_\theta\).

To build the injective map \(g_\gamma\) 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 $m \approx \log_2 \frac{D}{\pi^2}$ 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 $f_\theta$ is a $d$-dimensional submanifold in $\mathbb{R}^D$, likelihoods of the samples not on this manifold are not defined. To circumvent this difficulty we adopt a strategy recently proposed by Brehmer and Cranmer [2020]. We split the training into two phases: (i) the mean squared error (MSE) phase where we only optimize the injective part $g_{\eta}$, and (ii) the maximum likelihood (ML) training phase where we fit the parameters $\eta$ of the bijective part $h_\gamma$ to maximize the likelihood of the preimage of training data through $g_{\gamma}$; this step aims to match the density of $p_X$ to that of the ground truth $p_Z$.

The loss function that we minimize to find the parameters of $g_{\gamma}$ is given as

$$L_{\text{MSE}}(\gamma) = \frac{1}{N} \sum_{i=1}^{N} \| \xi^{(i)} - g_{\gamma}(g_{\eta}^{\dagger}(\xi^{(i)})) \|^2_2$$

(3)

where $\xi^{(i)}$-s are the training samples. We find that only a few epochs of training are sufficient to train $g_{\gamma}$. Note that $P_{g_{\gamma}}(x) := g_{\gamma}(g_{\eta}^{\dagger}(x))$ is an idempotent projection operator on the range of $g_{\eta}$. The low-dimensional range of $g_{\eta}$ acts as a regularizer in the context of inverse problems. Injectivity implies that the range of $f_\theta$ 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 $p_Z$ by maximizing the likelihood of the preimages of the training samples $\{g_{\eta}^{\dagger}(\xi^{(i)})\}$ over $\eta$. This gives us the loss function for the ML training phase as

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

(4)

where $z^{(i)} = h_{\eta}^{-1}(g_{\eta}^{\dagger}(\xi^{(i)}))$ and $J_{h_{\eta,l}}$ are evaluated at appropriate intermediate inputs. Together with the gradually-expanding architecture of TRUMPETS this two-step procedure yields much faster training than previous work which concatenates standard invertible flows.

Stability of layerwise inversions. To minimize $L_{\text{MSE}}$, we need to calculate the left inverse $g_{\eta}^{\dagger}$ for points that do not lie in the range of $g_{\eta}$. This entails computing the pseudo-inverses of injective convolutional layers $\ell_w$. We study the stability of inversion for out-of-range points under the assumption that $y' = \ell_w(x) + \epsilon$, $\epsilon \sim N(0, \sigma^2 I)$. 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

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

where $\sigma_i(w)$-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 $\sigma$, 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

$$\sum_{i=1}^{c} \frac{1}{\sigma_i(w)^2}$$

to

$$\sum_{i=1}^{c} \frac{s_i(w)^2}{\lambda^2},$$

which is upper bounded by $\frac{1}{\lambda^2 c^2}$, thus effectively stabilizing training. Here, $\lambda$ is the regularization parameter.

3 INFEERENCE 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 $y = Ax + \epsilon$, where $\epsilon$ 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) and 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

\[
x_{\text{MAP}} = \arg\max_x p_{\mathcal{X}|\mathcal{Y}}(y|x);
\]

Bayes theorem yields

\[
x_{\text{MAP}} = \arg\min_x \log p_{\mathcal{Y}|\mathcal{X}}(y|x) - \log p_X(x) = \arg\min_x \frac{1}{2} \| y - Ax \|^2_2 - \sigma^2 \log p_X(x),
\]

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

### 3.1 MAP ESTIMATION WITH TRUMPET PRIOR

We now address two inference tasks where TRUMPETS are particularly effective. Recall that since \( g_{\eta} \) is injective one can build a fast projector \( P_{g_{\eta}}(x) = g_{\eta}(l^T x) \) on the range of \( g_{\eta} \), 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 \([\text{W}h\text{a}n\text{g} \text{e}t \text{a}l. \text{2}0\text{2}0]\). 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 the \( -\log p_X \) term.

#### Algorithm 1: iFlow

Input: loss function \( L, y, A, g_{\eta} \);

Parameter: step size \( \eta \) and \( \lambda \propto \sigma^2 \);

\[
x^{[0]} = A^T y;
\]

for \( i \leftarrow 0 \) to \( T - 1 \) do

\[
\begin{align*}
& v \leftarrow P_{g_{\eta}}(x^{[i]}); \\
& x^{[i+1]} \leftarrow \text{GradientStep}(L(v));
\end{align*}
\]

end

\[
x^{[T]} \leftarrow P_{g_{\eta}}(x^{[T]});
\]

One caveat with computing \( -\log p_X(x) \) is that it requires \( \log |\det J_{f_\theta} f_\theta(x)| \), according to \((2)\). While we have layer-wise tractable Jacobians, \( \log |\det J_{f_\theta} f_\theta| \) 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 \( g : \mathbb{R}^K \rightarrow \mathbb{R}^D \), where \( g := g_1 \circ g_2 \circ \ldots \circ g_K \), we have \( \log |\det J^T_{f_\theta} J_{f_\theta}| \leq \sum_{i=1}^K \log |\det J^T_{g_i} J_{g_i}| \). Thus one gets an upper bound

\[
-\log p_X(x) \leq -\log p_Z(f^T(x)) + \frac{1}{2} \sum_{k=1}^K \log |\det J^T_{g_{\eta,k}} J_{g_{\eta,k}}| + \sum_{i=1}^K \log |\det J_{n_{\eta,i}}|,
\]

where the layer Jacobians are evaluated at the appropriate intermediate layer outputs. Since all our layers including the injective layers have \( \log \det J \) 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

\[
L(x) := \frac{1}{2} \| y - Ax \|^2_2 + \sigma^2 R(x).
\]

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 \([\text{S}u\text{n} \text{a}nd \text{B}o\text{u}m\text{a}n \text{2}0\text{2}0]\) which computes a variational approximation to the posterior \( p_{\mathcal{X}|\mathcal{Y}}(x|y) \) 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 \( f_{\theta} \) and the assumption that the modeled data concentrates close to the range of \( f_{\theta} \) allows us to write the posterior on \( X, p_{\mathcal{X}|\mathcal{Y}} \), in terms of \( p_{\mathcal{Z}|\mathcal{Y}} \) with \( X = f_{\theta}(Z) \). That is,

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

We can thus derive a computationally efficient version of the algorithm proposed by \([\text{S}u\text{n} \text{a}nd \text{B}o\text{u}m\text{a}n \text{2}0\text{2}0]\) 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 \( u_v \), which is a low-dimensional flow, so that the distribution of \( f_{\theta} \circ u_v(T) \) approximates the posterior \( p_{\mathcal{X}|\mathcal{Y}} \) when \( T \) is an iid Gaussian vector. The generative process for (approximate) samples from \( p_{\mathcal{X}|\mathcal{Y}} \) is then

\[
T \xrightarrow{u_v} Z \xrightarrow{h_\theta, G_\theta} X.
\]

We thus require that \( u_v(T) \sim p_{\mathcal{Z}|\mathcal{Y}} \) with \( T \sim \mathcal{N}(0, I) \) and \( X = f_{\theta}(Z) \). Letting \( q_{\eta} \) be the distribution of \( u_v(T) \), the parameters \( \nu \) are adjusted by minimizing the KL divergence.
We revisit the inverse problem associated with
\[ J \epsilon_p \] with
\[ \alpha \] such that the maximal singular value of \( I - \alpha J_{f_a}^T J_{f_a} \) 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 \( J_{f_a}^T J_{f_a} \) using power iteration. In the last step we use the Hutchinson trace estimator [Hutchinson, 1989] to evaluate the trace; the \( v \)-s are sampled iid from \( N(0, I) \). 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].
Table 1: FID scores on 8-bit 64 × 64 celebA dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kumar et al. [2020]</td>
<td>40.23</td>
</tr>
<tr>
<td>Brehmer and Cranmer [2020]</td>
<td>37.4</td>
</tr>
<tr>
<td>TRUMPET (Ours)</td>
<td>34.3</td>
</tr>
</tbody>
</table>

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

| Dataset       | Training time (hours) | ||x − f_θ(f_θ^t(x))|| / ||x|| | Trainable params |
|---------------|-----------------------|-----------------|-------------------------------|
| MNIST         | 11                    | 0.04            | 9M                            |
| CIFAR 10      | 11                    | 0.22            | 9M                            |
| CelebA        | 38                    | 0.15            | 16M                           |
| Chest X-ray   | 25                    | 0.13            | 11M                           |

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 issue is unfortunately not resolved by simply dividing by dimension. We thus caution the reader that such a comparison may 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 ∈ R^{m×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 × 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 \( \hat{x} = f(\arg\min_z ||y - Af(z)||^2) \) while DIP solves \( \hat{x} = f_\lambda(\arg\min_z ||y - Af(z)||^2) \) 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.

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× slower than iFlow. It was thus infeasible to finalize this comparison. On average we found that DIP converged the fastest followed by our method which was about 2× slower. Finally, while each iteration of CSGM was as fast as each of DIP, CSGM requires several restarts which made the method about 4× 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. To keep the comparisons fair we use the same generative model \( f_\theta \) 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 2 includes the performance of Algorithm 1 on this problem.)

Here we provide a pixel-wise uncertainty estimate of the form \( \mathbb{E}_{X \sim p_{X|Y=y}} [X - \langle X \rangle | Y] \) 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] have considerably improved performance of normalizing flows. Glow [Kingma and Dhariwal 2018] uses invertible \( 1 \times 1 \) 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 \( 256 \times 256 \) 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 several basic ideas in an intuitive way that yields gains in computational efficiency and modeling...
quality. It is worth noting that recent results on univer-
sality of globally injective neural networks [Puthawala et al.,
2020] and universality of flows [Teshima et al., 2020] sug-
uggest that TRUMPETS are universal approximators of prob-
ability measures concentrated on Lipschitz manifolds; a
rigorous proof is left to future work.

ACKNOWLEDGEMENTS

ID and AK were supported by the European Research Coun-
cil Starting Grant 852821—SWING. MVdH gratefully ac-
knowledges 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.

References

Martin Arjovsky and Léon Bottou. Towards principled meth-
ods for training generative adversarial networks. arXiv

William M Boothby. An introduction to differentiable mani-

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Di-

Johann Brehmer and Kyle Cranmer. Flows for simultaneous
manifold learning and density estimation. arXiv preprint

Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-
Henrik Jacobsen. Residual flows for invertible generative


