Regularized Autoencoders via Relaxed Injective Probability Flow

Abhishek Kumar
Google Research

Ben Poole
Google Research

Kevin Murphy
Google Research

Abstract

Invertible flow-based generative models are an effective method for learning to generate samples, while allowing for tractable likelihood computation and inference. However, the invertibility requirement restricts models to have the same latent dimensionality as the inputs. This imposes significant architectural, memory, and computational costs, making them more challenging to scale than other classes of generative models such as Variational Autoencoders (VAEs). We propose a generative model based on probability flows that does away with the bijectivity requirement on the model and only assumes injectivity. This also provides another perspective on regularized autoencoders (RAEs), with our final objectives resembling RAEs with specific regularizers that are derived by lower bounding the probability flow objective. We empirically demonstrate the promise of the proposed model, improving over VAEs and AEs in terms of sample quality.

1 Introduction

Invertible flow-based generative models (Dinh et al., 2016; Kingma & Dhariwal, 2018) have recently gained traction due to several desirable properties: (i) exact log-likelihood calculation (unlike VAEs that maximize a lower bound (Kingma & Welling, 2013; Rezende et al., 2014)), (ii) exact inference of latent variables, and (iii) good sample quality relative to VAEs.

However, a limitation of invertible flow models is that they require invertibility on the full ambient space, resulting in a latent space with the same dimensionality as the input data. This requirement leads to larger models with higher memory and computational costs that are more difficult to scale than VAE and GAN counterparts that have lower-dimensional latent spaces (Goodfellow et al., 2014; Kingma & Dhariwal, 2018). This lack of dimensionality reduction also makes it difficult to capture high-level generative factors directly in *individual* latent dimensions, a property that is often argued to be desirable for generative models (Higgins et al., 2017; Narayanaswamy et al., 2017; Kumar et al., 2018; Kim & Mnih, 2018; Chen et al., 2018).

In this work we propose a generative model of data based on probability flows that relaxes the bijectivity requirement. The model g maps low dimensional latents $z \in Z = \mathbb{R}^d$ to samples in the image of g, residing in the much higher-dimensional ambient space $\mathbb{R}^D$. A probability distribution in the latent space (*e.g.*, standard normal) is pushed forward by the mapping g to induce a distribution on the *image*, $g(Z)$. By taking the mapping g to be one-to-one or injective and differentiable, we can use a change of variables theorem to obtain a closed form for the distribution over $g(Z)$. While the resulting log-likelihood in ambient space is intractable, we can form tractable lower-bounds using stochastic approximations to obtain objectives amenable to stochastic first-order optimization. Relaxing the bijectivity requirement loses the ability to provide exact likelihoods for data points that lie off the image $g(Z)$. In this work, we limit ourselves to using the derived flow-based objective for learning a sampling mechanism that always generates samples from the image $g(Z)$. This is in contrast to VAEs where generated samples lie off the image $g(Z)$ due to the presence of an additional distribution at the output of the decoder.[1]

Our final objective, although derived from the probability flow perspective, resembles a regularized autoencoder with an additional prior log probability term and an annealing of the weight on the reconstruction loss that increases over time. This flow perspective

---

[1]One can augment the decoder with an ambient noise distribution (*e.g.*, Gaussian) either as part of the training objective or post-hoc after training (Wu et al., 2017) for estimating log-likelihoods, but we do not consider that here.
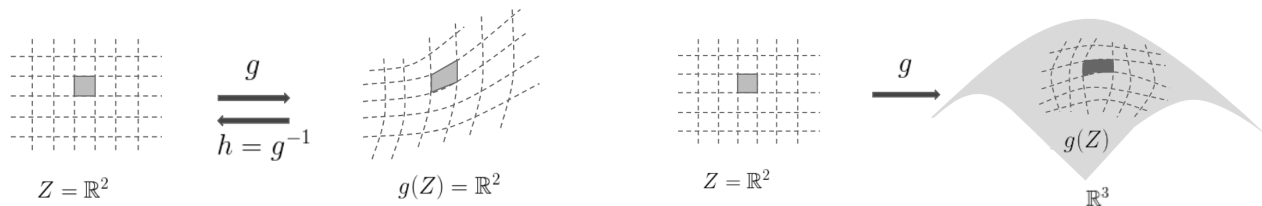
Figure 1: Schematic of invertible (left) vs. injective (right) mappings. Invertible flow models require that Z and X be the same dimensionality, and the mapping g to be invertible on the full domain. In contrast, injective mappings can have lower-dimensional Z but are invertible only on the image of g (shaded).

motivates several commonly used autoencoder regularization strategies, *e.g.*, those in Ghosh et al. (2019). We evaluate the relaxed injective flow models on MNIST, CIFAR-10, and CelebA, where we observe better FID scores compared to VAEs, and Ghosh et al. (2019). Our results demonstrate that these models provide an efficient and tractable mechanism for training neural samplers with compressed latent spaces.

2 Formulation

2.1 Invertible Flows

Let $g : Z \to X$ be a generator mapping from latents to data, assumed to be differentiable everywhere. If g is a bijection with $h = g^{-1} : X \to Z$, then X and Z must have the same dimensionality and we can write the distribution induced over X in terms of the distribution over Z using change of variables formula:

$$\begin{aligned} \ln p_x(x) &= \ln p_z(h(x)) + \ln |\det J_h(x)| \\ &= \ln p_z(h(x)) - \ln |\det J_g(h(x))| \end{aligned} \quad (1)$$

where $J_h(x)$ and $J_g(z)$ are the Jacobians of h and g at x and z, respectively. Invertible flow models (Dinh et al., 2016; Papamakarios et al., 2017; Kingma & Dhariwal, 2018) optimize (1) to learn a generative model of the data. They provide tractable objectives by structuring the generator so that the inverse and the log-det-Jacobian terms are tractable. Recent work on invertible residual nets (Behrmann et al., 2018) makes use of certain approximations to get a tractable objective for invertible flows with ResNets having Lipschitz constrained residual blocks. More recently, Behrmann et al. (2020) studied the numerical stability of invertible flow models, finding that, in practice, numerical issues may prevent the models from being invertible in certain regions even slightly off the data manifold.

2.2 Injective Flows

We are interested in developing probability flow-based models for the setting when the dimensionality of the latent space Z is much lower than the data dimensionality, i.e., $Z = \mathbb{R}^d$ and $X = \mathbb{R}^D$, where $D \gg d$. We can obtain a change of variables formula for this setting by looking at how an infinitesimal volume element dz at $z \in Z$ is transformed by the mapping g. The theory of integration on manifolds tells us that the mapping g transforms an infinitesimal volume element dz at $z \in Z$ to a corresponding volume $[\det(J_g(z)^\top J_g(z))]^{1/2}dz$ on $g(Z)$ (Boothby, 1986), where $g(Z) = \{g(z) : z \in Z\}$ is the image of Z under g. If we assume g is an injective function and thus invertible when seen as a mapping $g : Z \to g(Z)$, we can write the *probability flow* from Z to $g(Z)$ as:

$$\ln p_x(x) = \ln p_z(z) - \frac{1}{2} \ln \left| \det J_g(z)^\top J_g(z) \right|,$$
$$\text{s.t. } x = g(z). \quad (2)$$

Note that $\ln |\det J_g(z)^\top J_g(z)| = \ln \det J_g(z)^\top J_g(z)$ as $J_g(z)^\top J_g(z)$ is a positive definite matrix. Figure 1 presents a schematic of invertible and injective functions transforming an infinitesimal volume element. The more familiar change of variables formula in (1) can be derived as the special case when J is square and thus $\det J^T J = (\det J)^2$. In order to avoid solving an inverse problem for every x in our data (*i.e.*, finding z for every x s.t. $x = g(z)$), we assume the existence of an encoder $h : X \to Z$ such that $g(h(x)) = x$ for every $x \in g(Z)$. This lets us write

$$\ln p_x(x) = \ln p_z(h(x)) - \frac{1}{2} \ln \left| \det J_g(h(x))^\top J_g(h(x)) \right|,$$
$$\text{s.t. } x = g(h(x)). \quad (3)$$

Optimizing the $\ln \left| \det J_g(h(x))^\top J_g(h(x)) \right|$ term exactly may be computationally challenging for large models as it requires computing a $D \times d$ Jacobian matrix for every data point (however, it could still be tractable for smaller models where the latent dimensionality d is small). We propose two ways to lower bound the log likelihood in (3) in order to obtain a tractable objective we can maximize. Let the singular values of $J_g(h(x))$ be given by $\{s_i\}_{i=1}^d$. Using the inequality

$\ln x \leq \frac{x}{\lambda} + \ln \lambda - 1$, for all $x > 0$ and $\lambda > 0$ (based on concavity of the log), we have

$$
\begin{aligned}
\ln p(x) &= \ln p(h(x)) - \frac{1}{2} \ln \det[J_g(h(x))^\top J_g(h(x))] \\
&= \ln p(h(x)) - \frac{1}{2} \sum \ln s_i^2 \\
&\geq \ln p(h(x)) - \frac{1}{2} \sum_i \left(\frac{s_i^2}{\lambda} + \ln \lambda - 1 \right) \\
&= \ln p(h(x)) - \frac{1}{2\lambda} \|J_g(h(x))\|_F^2 - \frac{d}{2} \ln \lambda + \frac{d}{2}, \quad (4) \\
&\qquad \text{s.t. } x = g(h(x)).
\end{aligned}
$$

This lower bound is maximized for $\lambda = \frac{1}{d} \sum_i s_i^2$. Substituting it into (4), we get

$$
\begin{aligned}
\ln p(x) &\geq \ln p(h(x)) - \frac{d}{2} \ln \left(\frac{1}{d} \sum_i s_i^2 \right) \\
&= \ln p(h(x)) - \frac{d}{2} \ln \left(\frac{1}{d} \|J_g(h(x))\|_F^2 \right), \quad (5) \\
&\qquad \text{s.t. } x = g(h(x)).
\end{aligned}
$$

The bounds in (4) and (5) are also computationally expensive but we will show how to form efficient stochastic approximations in the next section.

Tightness of the bounds. The inequality in (4) is tight when all singular values are equal to λ. Note that it is also possible to use separate λ_i corresponding to each s_i and tune these as hyperparameters to improve upon the tightness of the bound, however we do not explore this for the sake of simplicity. We will still tune the hyperparameter λ in (4) to see how it performs against the objective in (5).

The objectives in (5) and (4) are constrained optimization problems that can be solved with a variety of approaches. Recent work on VAEs has used the augmented Lagrangian method to enforce reconstruction constraints (Rezende & Viola, 2018), but here we use the penalty method for its simplicity (Bertsekas, 2016). Applying penalty method to (5), we get:

$$
\ln p_z(h(x)) - \frac{d}{2} \ln \left[\left[\frac{1}{d} \|J_g(h(x))\|_F^2 \right] \right] - \mu \|x - g(h(x))\|_2^2, \tag{6}
$$

where μ is a positive real that is increased as the optimization progresses (Bertsekas, 2016). Optimizing (6) can still be computationally demanding as it involves computing the full Jacobian of the generator. We can use Hutchinson's trace estimator (Hutchinson, 1990) to avoid explicitly materializing the full Jacobian. Hutchinson's trace estimator is based on the fact that

$\operatorname{tr}(A) = \mathbb{E}_v \operatorname{tr}(Avv^\top) = \mathbb{E}_v v^\top A v$ for any random vector v s.t. $\mathbb{E}_v vv^\top = I$. We write the Frobenius norms of the Jacobian as $\|J_g(h(x))\|_F^2 = \operatorname{tr}(J_g(h(x))^\top J_g(h(x))) = \mathbb{E}_v v^\top J_g(h(x))^\top J_g(h(x)) v = \mathbb{E}_v \|J_g(h(x))v\|_2^2$ for $v \sim N(0, I_d)$. We further employ the unbiased Monte-Carlo estimation $\mathbb{E}_v \|J_g(h(x))v\|_2^2 \approx \frac{1}{k} \sum_{i=1}^k \|J_g(h(x))v_i\|_2^2$, and use one Monte-Carlo sample per example ($k = 1$) in a minibatch. This leads to an unbiased estimator of the bound when used with objective (4), and the expectation of the Monte-Carlo approximation *remains a lower bound* on the log likelihood.

When used with the objective in (5), Hutchinson's estimator leads to a biased estimator of $\ln\|J_g(h(x))\|_F^2$, as the expectation of this estimator is smaller than $\ln \mathbb{E}_v \|J_g(h(x))v\|_2^2$. This results in an estimate whose expectation may *no longer be a bound on the log likelihood*. Similar issue arises in earlier works that try to do Monte Carlo estimation for $\ln \mathbb{E}_x f(x)$ (Li & Turner, 2016; Rhodes & Gutmann, 2018). In spite of no longer bounding log-likelihood, we find that this approximation is still effective in practice to train neural samplers. Using this Monte-Carlo approximation yields

$$
\ln p_z(h(x)) - \frac{d}{2} \ln\|J_g(h(x))v\|_2^2 - \mu\|x - g(h(x))\|_2^2, \tag{7}
$$

with $v \sim N(0, I_d)$ and ignoring the constant terms (the factor of $\frac{1}{d}$ can also be absorbed in v). We use automatic differentiation[2] to optimize the term containing the Jacobian-vector product. However, we observe numerical instabilities while training models for some configurations (CIFAR-10 with $p_z = N(0, I)$). In these cases, we use the finite difference approximation:

$$
\|J_g(h(x))v\|_2^2 \approx \frac{\|g(z + \epsilon v) - g(z)\|^2}{\epsilon^2}, \tag{8}
$$

with small $\epsilon \ (= 0.01)$, and $v \sim N(0, I_d)$.

For g to be injective, a necessary condition is to constrain all singular values of $J_g(h(x))$ to be bounded away from zero. Instead of directly enforcing this which can be computationally challenging, we simply enforce that $\|J_g(h(x))v\|_2$ is greater than a threshold η for all v with $\|v\| = 1$. A similar approach was used by Odena et al. (2018). There are scenarios where positivity of singular values does not ensure global injectivity, *i.e.*, there may exist z_1, z_2 s.t. $x = g(z_1) = g(z_2)$ (see self-intersections in Lagrange et al. (2007)). Suppose $h(x) = z_1$ in this case, then the lower bounds in (5) and (4) are still valid since $\ln p(x) \geq \ln p(h(x)) - \frac{1}{2} \ln \det[J_g(h(x))^\top J_g(h(x))]$.

---

[2] If the automatic differentiation framework only allows for reverse mode AD, one can use $\mathbb{E}_v \|J_g(h(x))^\top v\|_2^2$ with $v \sim N(0, I_D)$ for $\|J_g(h(x))\|_F^2$, instead of $\mathbb{E}_v \|J_g(h(x))v\|_2^2$ with $v \sim N(0, I_d)$.

While training, we take the latent space distribution p_z to be an isotropic Gaussian distribution $N(0, \sigma^2 I_d)$, which reduces the first term in bounds (4) and (5) to be $\frac{1}{2\sigma^2}\|h(x)\|_2^2$. Our final minimization objective corresponding to the lower bound of (5) is given by

$$\min_{h,g} \ \mathbb{E}_{x,v} \Bigg(\frac{1}{2\sigma^2}\|h(x)\|_2^2 + \mu\|x - g(h(x))\|_2^2 +$$
$$\frac{d}{2}\ln \llbracket \max(\|J_g(h(x))v\|_2^2, \eta^2\|v\|^2) \rrbracket \ + \ \ (9)$$
$$\mu_{in}\left[\frac{1}{\|v\|}\|J_g(h(x))v\|_2 - \eta\right]_{-}^2 \Bigg)$$

where $v \sim N(0, I_d)$, $[a]_{-} = \min(a, 0)$, and μ_{in} is a positive penalty on the constraint enforcing the local injectivity of the generator. Both μ and μ_{in} are increased over the course of optimization (Bertsekas, 2016).

Following similar steps of forming an unconstrained objective using the penalty method and using Monte-Carlo estimation for $\|J_g(h(x))\|_F^2$, we obtain the following minimization objective corresponding to the lower bound of (4):

$$\min_{h,g} \ \mathbb{E}_{x,v} \Bigg(\frac{1}{2\sigma^2}\|h(x)\|_2^2 + \mu\|x - g(h(x))\|_2^2 +$$
$$\frac{1}{2\lambda}\llbracket \max(\|J_g(h(x))v\|_2^2, \eta^2\|v\|^2) \rrbracket \ + \ \ \ \ \ (10)$$
$$\mu_{in}\left[\frac{1}{\|v\|}\|J_g(h(x))v\|_2 - \eta\right]_{-}^2 \Bigg),$$

where λ is a fixed hyperparameter (which is not optimized over but can be tuned as discussed earlier). We optimize the objectives (9) and (10) with respect to parameters of both the generator g and the encoder h.

2.2.1   Sampling from the model

Although the injective flow model transforms an isotropic Gaussian prior to the data distribution, in practice we observe that the distribution of encoded data points ("aggregate posterior") deviates from the prior distribution, which is also reflected in poor quality of generated samples. Note that this is not linked to invertibility and can happen even when the network is perfectly invertible. This issue is not specific to our model and is present even in VAEs and bijective flow based models. For VAEs, recent work highlighted this issue in the case of modeling a data distribution that lies along a low-dimensional manifold (Sec. 4 in Dai & Wipf (2019)) and proposed fitting another distribution on the encoded latents after training the VAE. For invertible-flow models the Euclidean norm of the latent codes is often different from the typical set of the prior, indicating a systematic aggregate posterior-prior

mismatch (see Choi et al. (2018), and Fig. 8 in Kingma & Dhariwal (2018)).

Hence, for sampling from the model, we fit a distribution over the encoded training data in the latent space after the model has been trained, an approach taken by several recent works (van den Oord et al., 2017; Dai & Wipf, 2019; Ghosh et al., 2019). Dai & Wipf (2019) train another VAE on the encoded training data to get a complex post-fit prior. However, in this paper we experiment with fitting a Gaussian prior and a mixture of 10 Gaussians similar to Ghosh et al. (2019).

3   Related Work

Our work is similar in spirit to the recent work of Ghosh et al. (2019); van den Oord et al. (2017); Dai & Wipf (2019), that find regularized autoencoders paired with a learned prior produces high-quality samples. Our work provides another perspective on the regularized autoencoder (RAE) objective in (Ghosh et al., 2019), wherein the regularization terms arise naturally from approximating the log-likelihood objective of the injective probability flow. Ghosh et al. (2019) motivate the RAE objective by considering constant posterior-variance VAEs and connecting stochasticity at the decoder's input (arising by sampling from $q(z|x)$) to *smoothness* of the decoder. Recently, Kumar & Poole (2020) analyzed the implicit regularization in β-VAEs, deriving a regularizer that also depends on the Jacobian of the deocder but has a different form.

Regularized autoencoders have been widely studied in earlier works as well (Rifai et al., 2011b; Alain & Bengio, 2014; Poole et al., 2014). Contractive autoencoders (Rifai et al., 2011b) also penalize the Frobenius norm of the Jacobian, however the penalty is on the *encoder* Jacobian, which is different from our penalty on the *decoder* Jacobian. Most of these prior works on RAEs has been on improving the quality of the encoder for downstream tasks, wheres we are primarily interested in the quality of the generator for producing samples. Recent work has turned to regularizing autoencoders for sample quality as well, for example improving interpolation quality using an adversarial training objective (Berthelot et al., 2018).

Krusinga et al. (2019) recently used Eq. (2) to get density estimates for trained GANs. However as we noted earlier, these density estimates are by nature undefined for unseen real examples which may lie off the manifold.

Several earlier works have also used spectral regularizers in training generative models. Miyato et al. (2018) encourage Lipschitz smoothness of the GAN discriminator by normalizing the spectral norm of each layer.

Odena et al. (2018) study the spectral properties of the Jacobian of the generator and its correlation with the quality of generated samples. They empirically observe that regularizing the condition number of the Jacobian leads to more stable training and improved generative model.

4 Experiments

Datasets. Our experimental framework is based on Ghosh et al. (2019). We evaluate our proposed model and baselines on three publicly available datasets: CelebA (Liu et al., 2015), CIFAR-10 (Krizhevsky & Hinton, 2009), and MNIST (Lecun et al., 1998). We use 64×64 cropped images for CelebA faces as used in several prior works. Image size for CIFAR-10 and MNIST is 32×32 and 28×28, respectively.

Baseline models. Our final objective, although obtained by developing an injective flow and lower bounding its log likelihood, has resemblance with recently proposed regularized autoencoders (Ghosh et al., 2019) which arise as natural models for comparison.

We consider several smoothness regularizers in our evaluations, some of which have also been used by Ghosh et al. (2019):

(i) AE: Vanilla autoencoder trained with ℓ_2 reconstruction loss. **(ii) AE+L2**: Autoencoder with an additional ℓ_2-norm penalty on the decoder parameters (weight decay). **(iii) AE+SN**: Autoencoder with an additional spectral normalization on each individual layer of the decoder (*i.e.* normalizing the top singular value to be 1), motivated by (Miyato et al., 2018). **(iv) CAE**: We also use contractive autoencoder (Rifai et al., 2011a) as a baseline which penalizes the Frobenius norm of the *encoder's Jacobians*. We use a similar Hutchinson trace stochastic approximation (as used for our objectives) to optimize the Frobenius norm term in the CAE objective. Ghosh et al. (2019) also consider a *gradient penalty regularized AE* which penalizes the Frobenius norm of the decoder's Jacobian, a term which is also present in our objective (4). We also compare with **(v) VAE** (Kingma & Welling, 2013) and **(vi) β-VAE** (Higgins et al., 2017), both with a Gaussian observation model at the decoder's output $N(0, \sigma^2 I_D)$. For VAE, σ^2 is taken to be 1, while for β-VAE, varying σ directly controls β, with $\beta = \sigma^2$. We do not report a comparison with Wasserstein Autoencoders (WAE) as Ghosh et al. (2019) have shown that the tractable WAE-MMD version is outperformed by regularized autoencoders.

Architectures. We use convolutional neural net based architecture for both encoder and decoder, each having five layers of convolutions or transposed convolutions, respectively. Strides and kernel-size in the convolutional filters differ across datasets, but stay same for all the models for a given dataset. We use a slightly larger network than (Ghosh et al., 2019) (5 vs. 4 layers), and thus rerun all baseline methods so that the results are comparable. This also results in improved scores for baselines over those reported in (Ghosh et al., 2019). We use *elu* activation in both encoder and decoder, and also use batch normalization. Latent dimensionality is taken to be 128 for CIFAR-10 and CelebA, and 32 for MNIST. More details on the architectures used are provided in the supplementary material.

Hyperparameters and training. Our log-Frobenius norm objective (9) (referred as InjFlow$^{\ell n}$) has four hyperparameters: (i) variance σ^2 of the isotropic Gaussian distribution on latent space, which determines the weight on the term penalizing the norm of the encodings $\|h(x)\|_2^2$, (ii) penalty coefficient μ on the reconstruction loss, (iii) penalty coefficient μ_{in} on the injectivity loss term, and (iv) singular value threshold used in the injectivity term η. We use $\eta = 0.1$ in all our experiments. Both penalty coefficients μ and μ_{in} are initialized to be 1 at the beginning of optimization and are increased with each minibatch iteration i as $1 + \frac{i^\nu}{1000}$, where ν is searched over $\{1, 1.3\}$. The weight on the prior term $\|h(x)\|^2$ is searched over $\{0, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Our objective (10) (referred as InjFlow) has an additional hyperparameter λ that determines the weight on Frobenius norm regularization term, which we fix to 1 in all our experiments. As discussed earlier, this will result in a tight bound only for the case when all Jacobian singular values are one.

For AE+L2, the hyperparameter for the ℓ_2 regularization term is searched over the set $\{0.001, 0.01, 0.1, 0.5, 1\}$. For β-VAE, we search over the standard deviation σ of the decoder's distribution (which is related to β as $\beta = \sigma^2$ for the Gaussian observation model) over the set $\{0.001, 0.01, 0.05, 0.1, 0.5\}$ (for VAE, $\sigma^2 = \beta = 1$). For CAE (Rifai et al., 2011b), the hyperparameter penalizing the encoder's Jacobian norm is searched over $\{0.001, 0.01, 0.1, 0.5, 1., 10.\}$. We train all models using Adam optimizer (Kingma & Ba, 2014) with batch size of 128 and a fixed learning rate of 0.001. All models are trained for 100k minibatch iterations.

Evaluation. Evaluation of sample quality is a challenging task (Theis et al., 2015), and several metrics have been proposed in literature for this (Salimans et al., 2016; Heusel et al., 2017; Sajjadi et al., 2018). We use the FID score (Heusel et al., 2017) as the quantitative metric for our evaluations, which is one of the most popular metrics and has been used in several

Table 1: FID scores (lower is better). *Rec:* score for reconstructed test data, $\mathcal{N}$: score for decoded samples from a Gaussian prior with full covariance fit to encoded training samples, *GMM:* score for decoded samples from a GMM fit to encoded training samples. InjFlow and InjFlow$^{\ell n}$ are the models obtained from the objectives (10) and (9), respectively (with superscript ℓn denoting the presence of log with the Frobenius term in (5) and (9)).

|	MNIST			CIFAR10			CelebA		
		Samples			Samples			Samples	
	Rec.	$\mathcal{N}$	GMM	Rec.	$\mathcal{N}$	GMM	Rec.	$\mathcal{N}$	GMM
VAE	65.10	57.04	62.08	176.5	169.1	184.3	62.36	72.48	67.82
β-VAE	7.91	**24.31**	**8.12**	43.86	83.59	71.56	30.06	50.66	42.77
AE	8.69	43.40	12.14	41.45	81.13	70.97	30.16	51.48	43.49
CAE	10.51	45.18	12.90	41.13	81.53	70.11	31.12	48.13	40.67
AE+L2	7.76	34.27	9.69	43.02	81.28	70.13	29.97	50.02	42.09
AE+SN	8.07	37.19	11.84	41.34	81.35	70.94	31.21	51.13	43.33
InjFlow$^{\ell n}$	7.40	35.96	9.93	**40.11**	**78.78**	**68.26**	**27.93**	**47.70**	**40.23**
InjFlow	**6.0**	42.65	11.43	40.86	79.67	68.37	28.51	49.01	40.57
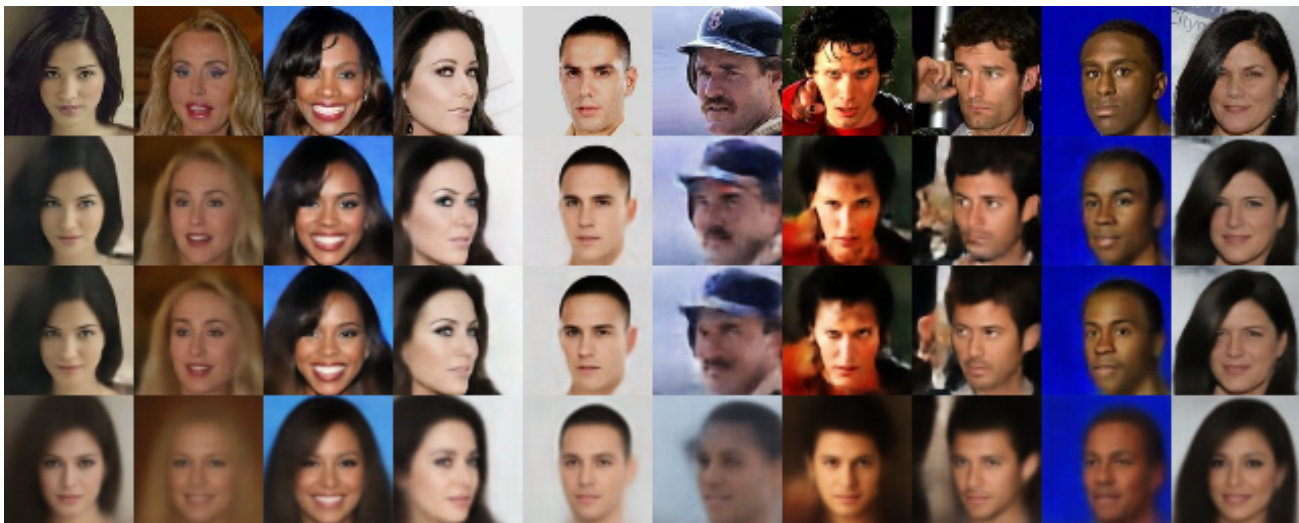
Figure 2: CelebA reconstructions: *Top to bottom:* Randomly sampled test examples, InjFlow reconstructions, Autoencoder reconstructions, VAE reconstructions.

recent works (Tolstikhin et al., 2017; Dai & Wipf, 2019; Ghosh et al., 2019) for evaluating generative models. As discussed earlier, we fit a Gaussian and a mixture of 10 Gaussians on the encoded training data, and use these as prior latent distributions to sample from the model. The covariance matrices for the Gaussian as well as for all mixture components in the GMM are taken to be full matrices. We also report FID scores on the test reconstructions apart from qualitative visualization of reconstructions and samples.

For all models, we report the best FID score obtained using decoder sampling from a post-fit GMM in the latent space. We then report all the other scores (i.e., scores for samples from post-fit Gaussian and test reconstructions) for the same model that gives the best

GMM samples FID score. This enables us to assess models in term of their best possible sample generation ability. The proposed injective flow models yield better FID scores than all the baseline models for CelebA and CIFAR10, and are competitive on MNIST where they are outperformed by β-VAE. For most cases, FID scores for samples with post-fit GMM are better than samples with post-fit Gaussian, except for VAE which we suspect could be due to a convergence issue with GMM fitting. In most cases, InjFlow$^{\ell n}$ yields better FID scores than InjFlow, which is expected as InjFlow$^{\ell n}$ uses optimal value of $\lambda = \frac{1}{d} \sum_i s_i^2$ (inequality (5)) as opposed to fixed and likely suboptimal value of $\lambda = 1$ used for InjFlow in our experiments.

Randomly generated CelebA samples for autoencoder,

Figure 3: Random CelebA Samples: *Top row:* InjFlow, *Middle row:* Autoencoder, *Bottom row:* VAE. *Left:* samples from post-fit GMM prior with 10 components, *Right:* samples from post-fit Gaussian prior.

VAE, and the proposed model (InjFlow) are visualized in Fig. 3. While VAE samples are blurry and tend to lose fine details, they retain global coherence. On the other hand, samples from InjFlow and autoencoder are sharper with more fine details but also have undesired visual artifacts in some cases. Fig. 2 shows reconstructions of randomly sampled test examples using InjFlow, autoencoder and VAE. InjFlow reconstructions preserve more fine details than both autoencoder and VAE (*e.g.*, hair strand for image in the third column), as also reflected by improved FID scores. More generated samples are shown in the supplementary material. It should be noted that better sample quality can be achieved by fitting a more expressive prior such as a GMM with more components, a VAE or a flow prior (Dinh et al., 2016; Papamakarios et al., 2017), however care must be taken to not overfit the latent encodings of the training points. In principle, a model that can produce good quality test reconstructions has the ability to generate good quality novel samples and the challenge lies in fitting a prior distribution that generalizes well.

5 Discussion

We proposed a probability flow based generative model that leverages an injective generator mapping, relaxing the bijectivity requirement. We use a change of variables formula to derive an optimization objective for learning the generator and encoder, where a smoothness regularizer on the generator naturally arises from the probability flow, along with some additional penalty terms. This nicely motivates several autoencoder regularizers that have been used in the past, such as in Ghosh et al. (2019). The proposed model also improves over several regularizers studied in Ghosh et al. (2019) in terms of FID scores.

Relaxing the bijectivity constraint loses many nice properties of invertible flow based generative models, such as tractable likelihood and inference for unseen data. A possible approach to recover these aspects could be to do define a background probability model over the full ambient space X and work with a mixture of foreground distribution over $g(Z)$ coming from probability flow and the background distribution. Investigation of this will be an interesting future direction.

To enable tractable and efficient training of Injective Flow models, we relied on lower bounds and stochastic approximation for the Jacobian term, and an amortized encoder trained with penalty method. Future work should investigate the degree to which these approximations are accurate, and whether there are better and more efficient approaches for ensuring invertibility ($g(h(x)) = x$) on training points (*e.g.*, augmented Lagrangian methods (Bertsekas, 2016) which have been

used in Rezende & Viola (2018)). A benefit of Injective Flow models is the ability to scale to larger input dimensions. In future work, we plan to improve sample quality and cater to higher-resolution images by scaling the models and fitting more expressive priors.

References

Alain, G. and Bengio, Y. What regularized autoencoders learn from the data-generating distribution. *Journal of Machine Learning Research*, 15(1): 3563–3593, 2014.

Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks, 2018.

Behrmann, J., Vicol, P., Wang, K.-C., Grosse, R. B., and Jacobsen, J.-H. On the invertibility of invertible neural networks, 2020. URL https://openreview.net/forum?id=BJlVeyHFwH.

Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018.

Bertsekas, D. P. Nonlinear programming: 3rd edition. 2016.

Boothby, W. M. *An introduction to differentiable manifolds and Riemannian geometry*, volume 120. Academic press, 1986.

Chen, T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Choi et al. WAIC, but Why? Generative Ensembles for Robust Anomaly Detection. *arXiv e-prints*, art. arXiv:1810.01392, 2018.

Dai, B. and Wipf, D. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Ghosh, P., Sajjadi, M. S. M., Vergari, A., Black, M., and Schölkopf, B. From variational to deterministic autoencoders, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Kim, H. and Mnih, A. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

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

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krusinga, R., Shah, S., Zwicker, M., Goldstein, T., and Jacobs, D. Understanding the (un) interpretability of natural image distributions using generative models. *arXiv preprint arXiv:1901.01499*, 2019.

Kumar, A. and Poole, B. On implicit regularization in β-vaes. *arXiv preprint arXiv:2002.00041*, 2020.

Kumar, A., Sattigeri, P., and Balakrishnan, A. Variational inference of disentangled latent concepts from unlabeled observations. In *ICLR*, 2018.

Lagrange, S., Delanoue, N., and Jaulin, L. On sufficient conditions of the injectivity: development of a numerical test algorithm via interval analysis. *Reliable computing*, 13(5):409–421, 2007.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Li, Y. and Turner, R. E. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems*, pp. 1073–1081, 2016.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Narayanaswamy, S., Paige, T. B., Van de Meent, J.-W., Desmaison, A., Goodman, N., Kohli, P., Wood, F., and Torr, P. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pp. 5925–5935, 2017.

Odena, A., Buckman, J., Olsson, C., Brown, T. B., Olah, C., Raffel, C., and Goodfellow, I. Is generator conditioning causally related to gan performance? *arXiv preprint arXiv:1802.08768*, 2018.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Poole, B., Sohl-Dickstein, J., and Ganguli, S. Analyzing noise in autoencoders and deep networks, 2014.

Rezende, D. J. and Viola, F. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Rhodes, B. and Gutmann, M. Variational noise-contrastive estimation. *arXiv preprint arXiv:1810.08010*, 2018.

Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. The manifold tangent classifier. In *NIPS*, 2011a.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 833–840. Omnipress, 2011b.

Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems*, pp. 5228–5237, 2018.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

Theis, L., Oord, A. v. d., and Bethge, M. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. On the quantitative analysis of decoder-based generative models. In *ICLR*, 2017.