GENERATIVE LATENT FLOW

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we propose the Generative Latent Flow (GLF), an algorithm for generative modeling of the data distribution. GLF uses an Auto-encoder (AE) to learn latent representations of the data, and a normalizing flow to map the distribution of the latent variables to that of simple i.i.d noise. In contrast to some other Auto-encoder based generative models, which use various regularizers that encourage the encoded latent distribution to match the prior distribution, our model explicitly constructs a mapping between these two distributions, leading to better density matching while avoiding over regularizing the latent variables. We compare our model with several related techniques, and show that it has many relative advantages including fast convergence, single stage training and minimal reconstruction trade-off. We also study the relationship between our model and its stochastic counterpart, and show that our model can be viewed as a vanishing noise limit of VAEs with flow prior. Quantitatively, under standardized evaluations, our method achieves state-of-the-art sample quality and diversity among AE based models on commonly used datasets, and is competitive with GANs' benchmarks.

1 INTRODUCTION

Deep generative models have recently attracted much attention in the literature on deep. These models are used to formulate the distribution of complex data as a function of random noise passed through a network, so that rendering samples from the distribution is particularly easy. Deep generative models can be roughly classified into explicit and implicit models. The former class assumes explicit parametric specification of the distribution, whereas the latter does not. Implicit models are dominated by Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Radford et al., 2015). GANs have exhibited impressive performance in generating high quality images (Brock et al., 2018) and other vision tasks (Zhu et al., 2017; Ledig et al., 2017). Despite their success, training GANs can be challenging, partly because they are trained by solving a saddle point optimization problem formulated as an adversarial game. It is well known that training GANs is unstable and extremely sensitive to hyper-parameter settings (Salimans et al., 2016; Arora et al., 2017), and sometimes training leads to mode collapse (Goodfellow, 2016), where most of the samples share some common properties. Although there have been multiple efforts to overcome the difficulties in training GANs, by modifying the objective functions or introducing normalization (Arjovsky et al., 2017; Metz Luke & Sohl-Dickstein, 2017; Srivastava et al., 2017; Miyato et al., 2018), researchers are also actively studying non-adversarial methods that are known to be less affected by these issues.

Some explicit methods directly model $p(x)$, the distribution of the data, and training is guided by maximizing the data likelihood. For example (Papamakarios et al., 2017; Oord et al., 2016) assume the data distribution can be expressed in an auto-regressive pattern, have a simple and stable training process, and currently give the best likelihood results; however, they cannot provide low-dimensional representations of images, and their sampling procedure is inefficient. Normalizing flows (Dinh et al., 2014; 2016; Kingma & Dhariwal, 2018) model $p(x)$ as an invertible transformation from a simple distribution through a change of variables. While being mathematically clear, normalizing flows have one major drawback: computational complexity. Flows have to keep the dimensionality of the original data in order to maintain bijectivity, and this makes training computationally expensive. Considering the prohibitively long training time and advanced hardware requirements in training large scale flow models such as (Kingma & Dhariwal, 2018), we believe that it is worth exploring the application of flows in the low dimensional representation spaces rather than in the original data.

Other explicit generative models often adopt low dimensional latent representations, which are usually obtained from auto-encoders, and generate samples by decoding z's sampled from a pre-defined prior distribution $p(z)$. We call this type of method AE based models. Variational Auto-encoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014) are perhaps the most influential AE based models. VAEs are trained to minimize a variational bound of the data log likelihood, which is composed of the reconstruction loss plus the KL divergence between $q(z|x)$, the approximate posterior distribution returned by the probabilistic encoder, and the prior $p(z)$. AE based models are easy to train, and they provide low dimensional codes for the data, but unfortunately, their generation quality still lies far below that of GANs, especially on large datasets. For example, it is observed that VAEs tend to generate blurry images, an effect that is usually attributed to the failure to match the marginal distribution in the latent space and the prior (Dai & Wipf, 2019; Rosca et al., 2018). Some modifications to VAEs (Burda et al., 2015; Rezende & Mohamed, 2015) improve the estimated test data likelihood. However, it is known that higher likelihood is not directly related to better sample quality (Theis et al., 2015; Grover et al., 2018). Some other modifications have recently been proposed to better match the distribution of latent variables and the prior distribution (Dai & Wipf, 2019; Tolstikhin et al., 2017; Ghosh et al., 2019; Hoshen et al., 2019), and they are shown to have the potential to generate high quality samples. These are discussed in greater detail in Section 2.

Our work pursues the same goal of improving the generation quality of AE based models. To this end, we propose Generative Latent Flow (GLF), which uses a deterministic auto-encoder to learn a mapping to and from a latent space, and a normalizing flow that serves as an invertible transformation between the latent space distribution and a simple noise distribution. Our contributions are summarized as follows: i) we propose Generative Latent Flow, which is an AE based generative model that can generate high quality samples. ii) through standardized evaluations, we show that our model achieves state-of-the-art sample quality among competing models, and can match the benchmarks of GANs. Moreover it has the advantage of one stage training and faster convergence. iii) we carefully study some variants of our method and show its relationship to other methods.

2 Motivation and related works

Consider an AE based generative model that can generate samples from the data space $\mathcal{X}$. Ideally, the auto-encoder defines a low dimensional latent space $\mathcal{Z}$, where each image $x_i \in \mathcal{X}$ is associated with a latent vector $z_i \in \mathcal{Z}$ through the decoder $x_i = G(z_i)$. The decoder G and observed data distribution detremine the marginal distribution over $\mathcal{Z}$, denoted by $\tilde{p}(z)$. This distribution is unknown and possibly The model also has a predefined prior distribution $p(z)$ on $\mathcal{Z}$. Since samples are generated by sampling $\epsilon \sim p(\epsilon)$ from the prior and feeding ϵ to the decoder, in order to generate high quality samples, AE based models need to have: (a) a good decoder G that can output realistic images given latent variables sampled from $\tilde{p}(z)$, and (b) a good match between $\tilde{p}(z)$ and $p(z)$.

Criterion (a) is easily ensured by minimizing the reconstruction loss of the auto-encoders, and there are different ways to ensure criterion (b). Intuitively, criterion (b) can be achieved by either modifying the decoder so that $\tilde{p}(z)$ is close to $p(z)$, or conversely modifying $p(z)$ to match some observed distribution on the latent space. The classic VAE model adopts the first approach indirectly using an approximation $q(z)$ for $\tilde{p}(z)$. It assumes a simple prior, and the KL regularizer in the ELBO objective penalizes $D_{\mathrm{KL}}(q(z)\|p(z))$ plus a mutual information term as shown in (Alemi et al., 2017). The approximation $q(z) = \mathbb{E}_{x\sim p_{data}}[q(z|x)]$ is called the aggregated approximate posterior. Several modifications to VAE's objective (Alemi et al., 2017; Chen et al., 2018; Kim & Mnih, 2018), which are designed for the task of unsupervised disentanglement, further decompose the KL term in ELBO, put a stronger penalty specifically on the mismatch between $q(z)$ and $p(z)$. There are also attempts to use normalizing flows as VAEs' posterior distributions (Rezende & Mohamed, 2015; Kingma et al., 2016; Berg et al., 2018). Although similar in name, they are completely different from our models, as they aim to complicate the approximate posterior $q(z|x)$ thus modifying the distribution $q(z)$. As of yet, these modifications to VAEs have not been shown to improve generation quality. In particular, empirically VAEs with flow posterior have been shown to improve neither the matching of $q(z)$ and $p(z)$ (Rosca et al., 2018), nor the generation quality (Dai & Wipf, 2019). Adversarial auto-encoders (Makhzani et al., 2015) and Wasserstein auto-encoders (Tolstikhin et al., 2017) use adversarial regularizer or MMD regularizers (Gretton et al., 2012) to force the $q(z)$ to be close to $p(z)$. These regularizations can be applied to both deterministic and probabilistic auto-encoders, and are shown to improve generation quality, as they generate sharper images than VAEs do.

One problem with regularizing $q(z)$ is that it introduces a trade off with reconstruction, i.e. criterion (a). This motivates the use of learnable priors optimized to match $q(z)$. (Tomczak & Welling, 2017; Klushyn et al., 2019; Bauer & Mnih, 2018) propose different ways to approximate the aggregated posterior $q(z)$ during training, and use the approximated $q(z)$ as their VAEs' prior distributions. This is a natural way to modify the prior to match $q(z)$, however, these methods have not been shown to improve generation quality. Two-stage-VAE(Dai & Wipf, 2019) introduces another VAE on the latent space defined by the first VAE to learn the distribution of its latent variables. GLANN (Hoshen et al., 2019) learns a latent representation by GLO (Bojanowski et al., 2017) and matches the densities of the latent variables with an implicit maximum likelihood estimator (Li & Malik, 2018). VQ-VAE (Oord et al., 2017) first learns an AE with discrete latent variables that are stored in a code-book, and then fits an auto-regressive prior on the latent space. RAE+GMM(Ghosh et al., 2019) trains a regularized auto-encoder (Alain & Bengio, 2014) and fits a mixture of Gaussian distribution on the latent space. These methods significantly improve the generation quality, but they all involve two-stage training procedure, which adds a computational overhead.

Motivated by these papers, we wish to design an AE based generative model that enjoys the best of both worlds: it can be trained end-to-end in a single stage, and it can greatly improve the generation quality without over regularizing the latent variables. We accomplish these goals by using normalizing flow on the latent space of a deterministic AE. More details will be presented in Section 3. Note that our method is closely related VAEs with normalizing flow as a learnable prior and the connections are discussed in Section 3.4.

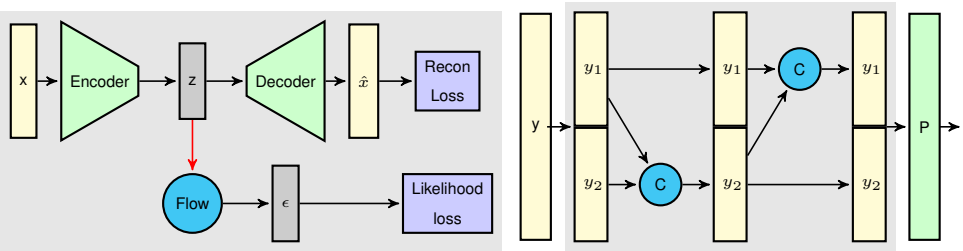


Figure 1: (a) Illustration of GLF model. The red arraw contains a stop gradient operation. See Section 3.3. (b) Structure of one flow block. It splits the input into two parts $y = (y_1, y_2)$, goes through the coupling layer C, and applies the random permutation P.

3 The Generative Latent Flow (GLF) Model

Our model uses a deterministic auto-encoder composed of an encoder $E_\eta : \mathcal{X} \to \mathcal{Z}$ and decoder (generator) $G_\phi : \mathcal{Z} \to \mathcal{X}$. In addition we have a transformation F_θ from the distribution on the noise space $\mathcal{E}$, which is assumed to be the standard Gaussian distribution, to the distribution on $\mathcal{Z}$. The transformation F_θ is defined in terms of a normalizing flow and all three components are learned simultaneously end to end using a loss that combines the reconstruction quality and the likelihood of the encoded data z_i with respect to the transformation F_θ.

3.1 Normalizing flows for the transformation F

The core of normalizing flows is carefully-designed invertible networks that map the training data to a simple distribution. Let $z \in \mathcal{Z}$ be an observation from an unknown target distribution $z \sim p(z)$ and p_ϵ be the unit Gaussian prior distribution on $\mathcal{E}$. Given a bijection $f_\theta : \mathcal{Z} \to \mathcal{E}$, we define a model $p_\theta(z)$ with parameters θ on $\mathcal{Z}$, and we can compute the negative log likelihood (NLL) of z by the change of variable formula:

$$-\log(p_\theta(z)) = \mathcal{L}_{\text{NLL}}(f_\theta(z)) = -\Big(\log p_\epsilon(f_\theta(z)) + \log \Big|\det\Big(\frac{\partial f_\theta(z)}{\partial z}\Big)\Big|\Big) \tag{1}$$

where $\frac{\partial f_\theta(z)}{\partial z}$ is the Jacobian matrix of f_θ. In order to learn the flow f_θ, the NLL objective of z is minimized, which is equivalent to maximize the likelihood of z. Since the mapping is a bijection,

sampling from the trained model $p_\theta(z)$ is trivial: simpy sample $\epsilon \sim p_\epsilon$ and compute $z = f_\theta^{-1}(\epsilon)$. In our method, we use the normalizing flow to model the transformation $F_\theta = f_\theta^{-1}$.

The key to designing a tractable flow model is defining the transformation f_θ so that the inverse transformation and the determinant of the Jacobian matrix can be efficiently computed. Based on (Dinh et al., 2016), we adopt the following layers to form the flows used in our model.

Affine coupling layer: Given a D dimensional input data z and $d < D$, we partition the input into two vectors $z_1 = z_{1:d}$ and $z_2 = z_{d+1:D}$. The output of one affine coupling layer is given by $y_1 = z_1$, $y_2 = z_2 \odot \exp(s(z_1)) + t(z_1)$ where s and t are functions from $\mathbb{R}^d \to \mathbb{R}^{D-d}$ and $\odot$ is the element-wise product. The inverse of the transformation is explicitly given by $z_1 = y_1$, $z_2 = (y_2 - t(y_1)) \odot \exp(-s(y_1))$. The determinant of the Jacobian matrix of this transformation is simply $\det \frac{\partial \mathrm{y}}{\partial \mathrm{z}} = \prod_{\mathrm{j}=1}^{\mathrm{d}} (\exp[\mathrm{s}(\mathrm{z}_1)_\mathrm{j}])$. Since computing both the inverse and the Jacobian does not require computing the inverse and Jacobian of s and t, both functions can be arbitrarily complex.

Combining coupling layers with random permutation: Affine coupling layers leave some components of the input data unchanged. In order to transform all the components, two coupling layers are combined in an alternating pattern to form a coupling block, so the unchanged components in the first layer can be transformed in the second layer. In particular, we add a fixed random permutation of dimensions of the input data after each coupling layer. See Figure 1b for an illustration of a coupling block used in our model.

3.2 THE OBJECTIVE FUNCTION

Having defined the invertible flow $F_\theta : \mathcal{E} \to \mathcal{Z}$, we also need to train a deterministic auto-encoder composed of an encoder E_η and a decoder G_ϕ. The auto-encoder is trained to minimize the reconstruction loss, which we set to be the common MSE loss. The overall training objective is a combination of the reconstruction loss and the NLL loss for the flow transformation:

$$\mathcal{L}(\eta, \phi, \theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(\beta \mathcal{L}_{\text{recon}}\big(\mathbf{x}_i, G_\phi(E_\eta(\mathbf{x}_i))\big) + \mathcal{L}_{\text{NLL}}\big(f_\theta(\text{sg}\,[E_\eta(\mathbf{x}_i)])\big) \Big) \tag{2}$$

where η, ϕ are the parameters of the encoder and decoder respectively, θ is the parameter of the flow, $\text{sg}[\cdot]$ is the stop gradient operation, and β is a hyper-parameter that controls the relative weight of the reconstruction loss and the NLL loss in equation equation 1. Note that the MSE loss corresponds to the assumptio that the observed pixel distribution is independent Gaussian with variance γ^2, conditional on the output of the decoder. Thus β can be viewed as the inverse variance (or dispersion).

After training the model, the generating process is easy: first sample a noise $\epsilon \sim \mathcal{N}(0, I)$ and then obtain a generated sample $\tilde{x} = G_\phi(F(\epsilon))$, where $F = f_\theta^{-1}$. Since the highlight of our model is applying a flow on latent variables, we name it **Generative Latent Flow** (GLF). See Figure 1a for an illustration of the GLF model.

3.3 WHY STOP THE GRADIENTS?

The stop gradient operation in equation 2 is important. If we let gradients of the NLL loss back propagate into the latent variables, it can lead to degenerate z's. This is because f_θ has to transform the z's to unit Gaussian noise, so the smaller the scale of the z's, the more negative the log-determinant of the Jacobian becomes. Since there is no constraint on the scale of latent variables, the Jacobian term can dominate the entire objective, driving the NLL loss to negative infinity by shrinking z towards 0. While the latent variables cannot become exactly 0 because of the presence of reconstruction loss in the objective, the extremely small scale of z may cause numerical issues that cause severe fluctuations. Therefore, we propose to stop the gradient of the NLL loss at the latent variables so that it cannot modify the values of z or affect the parameters of the encoder. We experimentally verify the problems of latent regularization in Section 4.2.1.

We call our original model with stopped gradients **GLF** and without stopped gradients **regularized GLF**, since the flow acts as a regularizer on the auto-encoder. Note that for GLF, the value of β in equation 2 does not matter, since the reconstruction loss and the NLL loss are independent. Note also that GLF can be trained in two stages, namely an auto-encoder is trained first, and then the flow

is trained to map the distributions. Empirically, we find that the two-stage training strategy leads to similar performance, so we only focus on one-stage training.

3.4 CONNECTION TO VAEs WITH FLOW PRIOR

Our method is closely related to VAEs with normalizing flow priors. To see this, consider the ELBO objective of plain VAEs with Gaussian prior and posterior ($\boldsymbol{\eta}$, $\boldsymbol{\phi}$ denote the parameters of encoder and decoder, respectively):

$$\text{ELBO}(\eta, \phi) = \mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{q_\eta(\mathbf{z}|\mathbf{x})}\left[\log p_{\phi,\beta}(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\eta(\mathbf{z}|\mathbf{x})\right] \tag{3}$$

The first term is related to the reconstruction loss and depends on the precision β of the observation at each pixel, while the last two terms can be combined as $D_{\mathrm{KL}}(q(z|x)\|p(z))$.

If we introduce a normalizing flow f_θ for the prior distribution, then the prior p_θ becomes $p_\theta(\mathbf{z}) = p_\epsilon(f_\theta(\mathbf{z}))\left|\det\left(\frac{\partial f_\theta(\mathbf{z})}{\partial \mathbf{z}}\right)\right|$, where p_ϵ is the standard Gaussian density. Substituting this prior into equation 3, we obtain the ELBO(η, ϕ, θ) for VAEs with flow prior:

$$\mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{q_\eta(\mathbf{z}|\mathbf{x})}\left[\log p_{\phi,\beta}(\mathbf{x}|\mathbf{z}) + \log p_\epsilon\left(f_\theta(\mathbf{z})\right) + \log\left|\det\left(\frac{\partial f_\theta(\mathbf{z})}{\partial \mathbf{z}}\right)\right| - \log q_\eta(\mathbf{z}|\mathbf{x})\right] \tag{4}$$

Comparing equation 4 and equation 2, we observe that if the expectation over $q_\eta(z|x)$ is estimated by sampling, ELBO(η, ϕ, θ) is precisely the negative of GLF's objective (without stopping gradients) plus an additional entropy term that corresponds to the entropy of encoder distribution. As β increases two things occur as demonstrated empirically in Section 4.2.1. First the estimated variances from the encoder decrease, and second the contribution of the reconstruction loss to the gradient of the encoder parameters becomes larger than the contribution of the flow's likelihood loss. Thus as β increases the VAE+flow converges to GLF. Furthermore Gaussian VAEs with flow prior does not suffer from the degeneracy of regularized GLF because of the presence of the entropy term. Since the entropy term is expressed as the negative sum of the log variances of the latent variables, it encourages the encoder to output large posterior variance, preventing latent variables from collapsing to 0.

VAEs with flow prior have attracted very little attention (Huang et al., 2017), and they have only focused on improvement of the data likelihood. Our work is differs in two ways: 1. we are the first to evaluate the effects of normalizing flow prior on generation quality; 2. we use deterministic AEs rather than VAEs, and this modification leads to better performances and avoids the need to choose β.

4 EXPERIMENTS

To demonstrate the performance of our method, we present both quantitative and qualitative evaluations on four commonly used datasets for generative models: MNIST (Lecun, 2010), Fashion MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009) and CelebA (Liu et al., 2015). Throughout the experiments, we use 20-dimensional latent variables for MNIST and Fashion MNIST, and 64-dimensional latent variables for CIFAR-10 and CelebA.

(Lucic et al., 2018) adopted a common network architecture based on InfoGAN (Chen et al., 2016) to evaluate GANs. In order to make fair comparisons without designing arbitrarily large networks to achieve better performance, we use the generator architecture of InfoGAN as our decoder's architecture, and we make the encoder to be symmetric to the decoder. For details of the AE network structures, see Appendix A. For the flow applied on latent variables, we use 4 affine coupling blocks defined as in Figure 1b, where each block contains 3 fully connected layers each with k hidden units. For MNIST and Fashion MNIST, $k = 64$, while for CIFAR-10 and CelebA, $k = 256$. Note that the flow only adds a small parameter overhead on the auto-encoder (less than 3%).

4.1 METRICS

We use the Fréchet Inception Distance (FID) (Heusel et al., 2017) as a metric for image generation quality. FID is computed by first extracting features of a set of real images x and a set of generated images g from an intermediate layer of the Inception network (Szegedy et al., 2015). Each set of

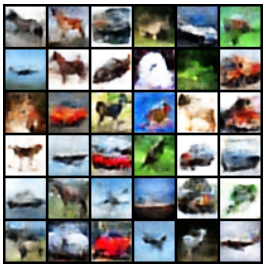
features is fitted with a Gaussian distribution, yielding means μ_x, μ_g and co-variances matrices Σ_x, Σ_g. The FID score is defined to be the Frchet distance between these two Gaussians:

$$\text{FID}(x, g) = \|\mu_x - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_x + \Sigma_g - 2\left(\Sigma_x\Sigma_g\right)^{\frac{1}{2}}\right)$$

It is claimed that the FID score is sensitive to mode collapse and correlates well with human perception of generator quality (Lucic et al., 2018). Recently, (Sajjadi et al., 2018) proposed using Precision and Recall for Distributions (PRD) which can assess both the quality and diversity of generated samples. We also include PRD in our studies.

(a) MNIST

(b) Fashion MNIST



(c) CIFAR-10

(d) CelebA

(e) CelebA-HQ

(f) Noise interpolation

Figure 2: (a)-(e): Randomly generated samples from our method trained on different datasets. (f): Random noise interpolation on CelebA.

Table 1: FID scores obtained from different models. For our reported results, we executed 10 independent trials and report the mean and standard deviation of the FID scores. Each trail is computing the FID between 10k generated images and 10k real images.

	MNIST	Fashion	CIFAR-10	CelebA
VAE	28.2 ± 0.3	57.5 ± 0.4	142.5 ± 0.6	71.0 ± 0.5
WAE-GAN	12.4 ± 0.2	31.5 ± 0.4	93.1 ± 0.5	66.5 ± 0.7
Two-Stage VAE	10.9 ± 0.7	26.1 ± 0.9	96.1 ± 0.9[1]	65.2 ± 0.8
RAE + GMM	10.8 ± 0.1	25.1 ± 0.2	91.6 ± 0.6	57.8 ± 0.4
VAE+flow prior	28.3 ± 0.2	51.8 ± 0.3	110.4 ± 0.5	54.3 ± 0.3
VAE+flow posterior	26.7 ± 0.3	55.1 ± 0.3	143.6 ± 0.8	67.9 ± 0.3
GLF (ours)	**8.2** ± 0.1	**21.3** ± 0.2	**88.3** ± 0.4	**53.2** ± 0.2
GLANN with perceptual loss	8.6 ± 0.1	13.0 ± 0.1	46.5 ± 0.2	46.3 ± 0.1
GLF+perceptual loss (ours)	**5.8** ± 0.1	**10.3** ± 0.1	**44.6** ± 0.3	**41.8** ± 0.2

[1]Note that there is a large discrepancy between this and the result reoprted in the original paper. See Appendix B.4 for explanation

4.2 RESULTS

Table 1 summarizes the main results of this work. We compare the FID scores obtained by our method with the scores of the VAE baseline and several existing AE based models that are claimed to produce high quality samples. Instead of directly citing their reported results, we re-ran the experiments because we want to evaluate them under standardized settings so that all models adopt the same AE architectures, latent dimensions and image pre-processing. We use GLF and VAE+flow prior/posterior with $\beta = 1$ to report the results in the table. For other methods, we largely follow their proposed experimental settings. Details of each experiment are presented in Appendix B.

Note that the authors of WAE propose two variants, namely WAE-GAN and WAE-MMD. We only report the results of WAE-GAN, as we found it consistently outperforms WAE-MMD. Note also that, GLANN (Hoshen et al., 2019) obtains impressive FID scores, but it uses perceptual loss (Johnson et al., 2016) as the reconstruction loss. The perceptual loss is obtained by feeding both training images and reconstructed images into a pre-trained network such as VGG (Simonyan & Zisserman, 2014), and computing the L_1 distance between some of the intermediate layers' activation. We also train our method with perceptual loss and compare with GLANN in the last two rows of Table 1.

As shown in Table 1, our method obtains significantly lower FID scores than competing AE based models across all four datasets. In particular, GLF greatly outperforms VAE+flow prior in the default setting. A more detailed analysis and comparison between the two methods will be done in Section 4.2.1. We also confirm that VAE+flow posterior cannot improve generation quality. Perhaps the competing model with the closest performances to ours is RAE+GMM, which shares some similarity with GLF in that both methods fit the density of the latent variables of an AE explicitly. To compare our method with GANs, we also include the results from (Lucic et al., 2018) in Appendix D. In (Lucic et al., 2018), the authors conduct standardized and comprehensive evaluations of representative GAN models with large-scale hyper-parameter searches, and therefore, their results can serve as a strong baseline. The results indicate that our method's generation quality is competitive with that of carefully tuned GANs.

In Table 3, Appendix C, we present the Precision and Recall scores of our method and several competing methods. As shown in the table, GLF obtains state-of-the-art Precision and Recall scores across all datasets, indicating that our method outperforms competing methods in terms of both sample quality and diversity.

Some qualitative results are shown in Figure 2. Besides samples of the datasets used for quantitative evaluation, samples of CelebA-HQ (Karras et al., 2017) with the larger size of 256×256 are also included in Figure 2e to show our method's potential of scaling up to images with higher resolution. Qualitative results show that our model can generate sharp and diverse samples in each dataset. In Figure 2f, we show CelebA images generated by linearly interpolating two samples of random noise. The smooth interpolation indicates that our method fits the distribution of latent variables well. For more qualitative results, including samples from the models trained with perceptual loss, see Appendix G. We see that samples from models trained with perceptual loss have higher quality.

4.2.1 COMPARISONS: GLF VS. REGULARIZED GLF AND VAE+FLOW PRIOR.

As discussed in Section 3.3, regularized GLF is unstable because of the degeneracy of latent variables created by the NLL loss. We empirically study the effect of latent regularization as a function of β on CIFAR-10. Settings for the experiments in this subsection can be found in Appendix B.6. For low values of $\beta = 1$ and 10, the NLL loss completely dominates the learning signal and the reconstruction loss quickly diverges. Even for larger values of $\beta = 50, 100, 400$ the NLL loss decreases to a very small negative value, and although overall performance is reasonable it oscillates quite strongly as training proceeds. The relevant plots are shown in Figure 4 in Appendix E. In contrast, for GLF, where the flow does not modify z, the NLL loss does not degenerate, resulting in stable improvements of FID scores as training progress.

We also trained VAEs+flow prior with different choices of decoder variances (equivalently, different choices of β), plus one with learnable decoder variance as done in (Dai & Wipf, 2019). We record the progression of FID scores of these models on CIFAR-10 in Figure 3a. In Figure 3b, we plot the entropy loss, which is one term in VAE+flow prior's minimization objective. The entropy loss is

expressed as $-\sum_{j=1}^{d} \log(\sigma_j)/2$, where σ_j is the standard deviation of the j^{th} latent variable. Higher entropy loss means that the latent variables have lower variances. In Figure 3c, we plot the NLL loss.

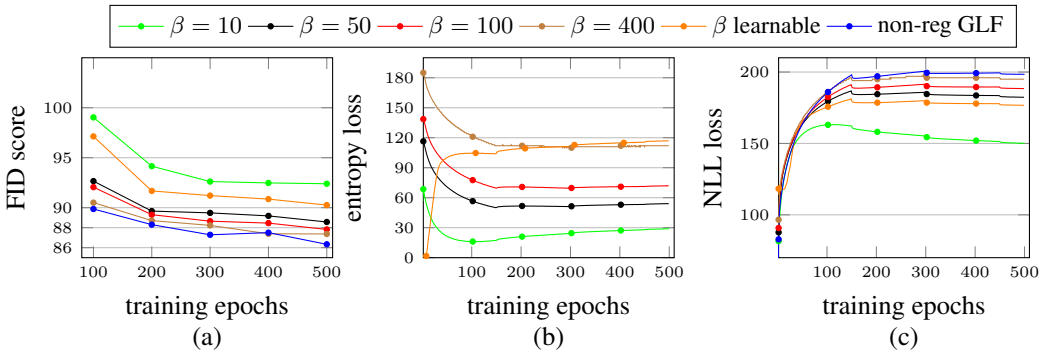


Figure 3: (a) Record of FID scores on CIFAR-10 for VAEs+flow prior with different values of β and GLF. (b) Record of entropy losses for corresponding models. (c) Record of NLL losses for corresponding models.

.

From Figure 3a, we see that GLF converges faster and obtains lower FID score than VAEs+flow prior. The performance gap closes as β increases, however, even with large β, GLF still slightly outperforms VAE+flow prior. We also find that using learnable β is not effective, probably due to relatively small values of β in early time. When β is large, as indicated before, the posterior variances become very small, so that effectively we are training an AE. For example, as shown in Figure 3b, when $\beta = 400$, the corresponding average posterior variance is around 10^{-4}.

In contrast to regularized GLF, there is no degeneracy of latent variables observed thanks to the noise introduced by VAEs and the corresponding entropy term. Indeed, Figure 3c shows that the training of VAE+flow prior does not over-fit the NLL loss, as opposed to regularized GLF where severe over-fitting to NLL loss occurs as shown in Figure 4c. Comparing Figure 3a and 4a, we observe that unlike regularized GLF, VAE+flow prior does not suffer from divergence or fluctuations in FID scores, even with relatively small β. In general, the results of FID scores show that regularized GLF is unstable, while as β increases, the performance of VAE+flow prior converges to that of GLF, which outperforms them all.

4.3 Training time

Besides better performances, our method also has the advantage of faster convergence among competing methods such as GLANN and Two-stage VAE. In Table 5, Appendix F, we compare the number of training epochs to obtain the FID scores in Table 1. We also compare the per epoch training clock time in Table 6, Appendix F. The combined results indicate that GLF requires much less training time while generate samples with higher quality.

5 Conclusion

In this paper, we introduce Generative Latent Flow, a novel generative model which uses an auto-encoder to learn a latent space from training data and a normalizing flow to match the distribution of the latent variables with the prior. Under standardized evaluations, our model achieves state-of-the-art results in image generation quality and diversity among several recently proposed Auto-encoder based models. Besides higher generation quality, our method also enjoys advantages such as faster training time and end-to-end single stage training. While we are not claiming that our GLF model is superior to GANs, we do believe that it opens the door to realize the potential of AE based models to produce high quality samples just as GANs do. The comparison with its stochastic counterparts seems to indicate that our method corresponds to the vanishing noise limit of the data likelihood. This is a topic that deserves further study.

REFERENCES

Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

Alexander A Alemi, Ben Poole, Ian Fischer, Joshua V Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. *arXiv preprint arXiv:1711.00464*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 224–232. JMLR. org, 2017.

Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. *arXiv preprint arXiv:1810.11428*, 2018.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. *arXiv preprint arXiv:1803.05649*, 2018.

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

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Bin Dai and David Wipf. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

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.

Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

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

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 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. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Yedid Hoshen and Lior Wolf. Nam: Non-adversarial unsupervised domain mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 436–451, 2018.

Yedid Hoshen, Ke Li, and Jitendra Malik. Non-adversarial image synthesis with generative latent nearest neighbors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5811–5819, 2019.

Chin-Wei Huang, Ahmed Touati, Laurent Dinh, Michal Drozdzal, Mohammad Havaei, Laurent Charlin, and Aaron Courville. Learnable explicit density for continuous latent space and variational inference. *arXiv preprint arXiv:1710.02248*, 2017.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 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.

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

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Alexej Klushyn, Nutan Chen, Richard Kurle, Botond Cseke, and Patrick van der Smagt. Learning hierarchical priors in vaes. *arXiv preprint arXiv:1905.04982*, 2019.

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

Y. Lecun. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 2010.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018.

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*, pp. 3730–3738, 2015.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems 31*, pp. 700–709, 2018.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Pfau David Metz Luke, Poole Ben and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

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

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

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

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference. *arXiv preprint arXiv:1802.06847*, 2018.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems 31*, pp. 5228–5237, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pp. 3308–3318, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

Table 2: Network structure for auto-encoder based on InfoGAN

Encoder	Decoder
Input x	Input z
4×4 Conv_{64}, ReLU	FC nz $\rightarrow 1024$, BN, ReLU
4×4 Conv_{128}, BN, ReLU	FC $1024 \rightarrow 128 \times M \times M$, BN, ReLU
Flatten, FC $128 \times M \times M \rightarrow 1024$, BN, ReLU	4×4 Deconv_{64}, BN, ReLU
FC $1024 \rightarrow$ nz	4×4 Deconv_{128}, Sigmoid

A Network Architectures

In this section we provide Table 2 that summarizes the auto-encoder network structure. The network structure is adopted from InfoGAN(Chen et al., 2016), and the difference between the networks we used for each dataset is the size of the fully connected layers, which depends on the size of the image. All convolution and deconvolution layers have stride $= 2$ and padding $= 1$ to ensure the spatial dimension decreases/increases by a factor of 2. M is simply the size of an input image divided by 4. Specifically, for MNIST and Fashion MNIST, $M = 7$; for CIFAR-10, $M = 8$; for CelebA, $M = 16$. BN stands for batch normalization.

For VAEs, the final FC layer of the encoder will have doubled output size to return both the mean and standard deviation of latent variables.

B Experiment Settings

In this section, we present the details of our experimental settings for results in Table 1. Since the settings for MNIST and Fashion MNIST are the same, we only mention MNIST for simplicity. For GLANN, we directly cite the results from (Hoshen et al., 2019), as their experimental settings is very similar to ours.

We use the original images in the training sets for MNIST, Fashion MNIST and CIFAR-10. For CelebA, we follow the same pre-processing as in (Lucic et al., 2018): center crop to 160×160 and then resize to 64×64.

B.1 Settings for training GLF

For all datasets (except CelebA-HQ), we use batch size 256 and Adam (Kingma & Ba, 2014) optimizer with initial learning rate 10^{-3} for the parameters of both the AE and the flow. We add a weight decay 2×10^{-5} to the optimizer for the flow. For MNIST, we train our model for 100 epochs, with learning rate decaying by a factor of 2 after 50 epochs. For CIFAR-10, we train our model for 200 epochs, with the learning rate decaying by a factor of 2 every 50 epochs. For CelebA, we train our model for 40 epochs with no learning rate decay.

For GLF with perceptual loss, we compute the perceptual loss as suggested in (Hoshen & Wolf, 2018). See `https://github.com/facebookresearch/NAM/blob/master/code/perceptual_loss.py` for their implementation. Other settings are the same.

For CelebA-HQ dataset, we adopt our AE network structure based on DCGAN (Radford et al., 2015). Note that this is a relatively simple network for high resolution imgaes. We use batch size 64, with initial learning rate 10^{-3} for both the AE and the flow. We train our model for 60 epochs, with learning rate decaying by a factor of 2 after 40 epochs.

B.2 Settings for training VAEs and VAE variants

We adopt common settings for our reported results of VAE, VAE+flow prior and VAE+flow posterior. We use $\beta = 1$ for all three VAE variants. We still use batch size 256, and Adam optimizer with initial learning rate 10^{-3} for both the VAE and the flow, if applicable. We find VAEs need longer time to converge, so we double the training epochs. We train MNIST for 200 epochs, with learning rate decaying by a factor of 2 after 100 epochs. We train CIFAR-10 for 400 epochs, with the learning

rate decaying by a factor of 2 every 100 epochs. We train CelebA for 80 epochs with learning rate decaying by a factor of 2 after 40 epochs.

B.3 SETTINGS FOR TRAINING WAE-GAN

We follow the settings introduced in the original WAE paper(Tolstikhin et al., 2017). The adversary in WAE-GAN has the following architecture:

$$\begin{aligned} z \in \mathcal{R}^d &\to \mathrm{FC}_{512} \to \mathrm{ReLU} \\ &\to \mathrm{FC}_{512} \to \mathrm{ReLU} \\ &\to \mathrm{FC}_{512} \to \mathrm{ReLU} \\ &\to \mathrm{FC}_{512} \to \mathrm{ReLU} \to \mathrm{FC}_1 \end{aligned}$$

where d is the dimension of the latent variables.

WAE has two major hyper-parameters: λ that controls the weight coefficient of the adversarial regularizer, and σ^2 which is the variance of the prior. Batch size is 100 for all datasets. For MNIST, $\lambda = 10$ and $\sigma^2 = 1$, and the model is trained for 100 epochs. The initial learning rate is 10^{-3} for the AE and 5×10^{-4} for the adversary. After 30 epochs both learning rates decreased both by factor of 2, and after first 50 epochs further by factor of 5. For CIFAR, $\lambda = 10$ and $\sigma^2 = 1$ and the model is trained for 200 epochs. The initial learning rates are the same as training MNIST, and the learning rate decays by a factor of 2 after first 60 epochs, and further by a factor of 5 after 120 epochs. For CelebA, $\lambda = 1$ and $\sigma^2 = 2$. The model is trained for 55 epochs. The initial learning rate is 3×10^{-4} for the AE and 10^{-3} for the adversary. Both learning rates decays by factor of 2 after 30 epochs, further by factor of 5 after 50 first epochs.

B.4 SETTINGS FOR TRAINING TWO STAGE VAE

We adopt the settings in the original paper (Dai & Wipf, 2019). For all datasets, the batch size is set to be 64, and the initial learning rate for both the first and the second is 10^{-4}. For MNIST, the first VAE is trained for 400 epochs, with learning rate halved every 150 epochs; the second VAE is trained for 800 epochs with learning rate halved every 300 epochs. For CIFAR-10, 1000 and 2000 epochs are trained for the two VAEs respectively, and the learning rates are halved every 300 and 600 epochs for the two stages. For CelebA, 120 and 300 epochs are trained for the two VAEs respectively, and the learning rates are halved every 48 and 120 epochs for the two stages.

Explaining the discrepancy between our reported results and the results in the original paper: The original Two stage VAE paper adopts similar settings with our experiments, but we observe large discrepancies on the results of CIFAR-10 and CelebA. After carefully reviewing their published codes, we find that there is an issue in their FID score computation particularly for CIFAR-10 dataset. Specifically, the true images used for computing the FID on CIFAR-10 is obtained from saving the original data file in .jpg format and reading them back, and the saving will cause some errors in pixel values. After fixing this issue, we re-ran their published codes and obtained similar results as we reported. We also run through their original FID computation protocol using samples from our models, and we obtain scores around 65. For CelebA, one particular detail worth noting is that, (Dai & Wipf, 2019) applies 128×128 center-crop before re-sizing on CelebA, while 160×160 center-crop is used in our evaluations. With smaller center-crops the human faces occupy a larger portion of the image with less background, making the generative modeling easier.

B.5 SETTINGS FOR TRAINING RAE+GMM

The settings of batch size, learning rate scheduling and number of epochs for training RAE are the same as those of GLF. The objective of the RAE is reconstruction loss plus a penalty on the norm of the latent variable. Since the author does not report their choices for the penalty coefficient γ, we search over $\gamma \in 0.1, 0.5, 1, 2$, and we find that $\beta = 0.5$ leads to the best overall performances, and therefore we let $\gamma = 0.5$. After training the RAE, we fit a 10-component Gaussian mixture distribution on the latent variables.

B.6 SETTINGS FOR EXPERIMENTS IN SECTION 4.2.1

For all experiments in Section 4.2.1, we use batch size 256 and initial learning rate 10^{-3} for both AE and flow. We train all models for 500 epochs with learning rates decaying by a factor of 2 every 150 epochs.

C PRECISION AND RECALL

In this section, we report the precision and recall (PRD) evaluation of samples on each dataset in Table 3. We include WAE-GAN, Two-stage VAE, RAE+GMM and GLANN (with perceptual loss) for comparisons. As the case of FID scores, for GLANN, we directly cite their reported results, and we compute the results ofr other models. We report the PRD of models trained under the settings introduced in Appendix B.

The two numbers in each entry are $F_8, F_{\frac{1}{8}}$ that capture recall and precision, respectively. See (Sajjadi et al., 2018) for more details. Higher numbers are better.

Table 3: Evaluation of sample quality by precision/recall.

	MNIST	Fashion	CIFAR-10	CelebA
WAE-GAN	(0.978, 0.956)	(0.901, 0.837)	(0.414, 0.723)	(0.501, 0.512)
Two-stage VAE	(0.982, 0.977)	(**0.937**, 0.845)	(0.382, 0.669)	(0.452, 0.558)
RAE+GMM	(**0.988**, 0.971)	(0.922, 0.924)	(0.370, 0.733)	(0.333, 0.445)
GLF (ours)	(0.982, **0.985**)	(0.932, **0.926**)	(**0.485**, **0.767**)	(**0.542**, **0.618**)
GLANN+perceptual loss	(0.971, 0.979)	(0.985, 0.963)	(**0.860**, 0.825)	(0.574, 0.681)
GLF+perceptual loss (ours)	(**0.990**, **0.992**)	(**0.987**, **0.980**)	(0.765, **0.845**)	(**0.760**, **0.778**)

D COMPARISON WITH GANS

In Table 4 we combine our reported results of AE based models and the FID scores of GANs cited from (Lucic et al., 2018).

Table 4: FID score comparisons of GANs and various AE based models

	MNIST	Fashion	CIFAR-10	CelebA
MM GAN	9.8 ± 0.9	29.6 ± 1.6	72.7 ± 3.6	65.6 ± 4.2
NS GAN	6.8 ± 0.5	26.5 ± 1.6	58.5 ± 1.9	55.0 ± 3.3
LSGAN	7.8 ± 0.6	30.7 ± 2.2	87.1 ± 47.5	53.9 ± 2.8
WGAN	6.7 ± 0.4	21.5 ± 1.6	55.2 ± 2.3	41.3 ± 2.0
WGAN GP	20.3 ± 5.0	24.5 ± 2.1	55.8 ± 0.9	30.3 ± 1.0
DRAGAN	7.6 ± 0.4	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0
BEGAN	13.1 ± 1.0	22.9 ± 0.9	71.4 ± 1.6	38.9 ± 0.9
VAE	28.2 ± 0.3	57.5 ± 0.4	142.5 ± 0.6	71.0 ± 0.5
WAE-GAN	12.4 ± 0.2	31.5 ± 0.4	93.1 ± 0.5	66.5 ± 0.7
Two-Stage VAE	10.9 ± 0.7	26.1 ± 0.9	96.1 ± 0.9	65.2 ± 0.8
RAE + GMM	10.8 ± 0.1	25.1 ± 0.2	91.6 ± 0.6	57.8 ± 0.4
GLANN (with perceptual loss)	8.6 ± 0.1	13.0 ± 0.1	46.5 ± 0.2	46.3 ± 0.1
VAE+flow prior	28.3 ± 0.2	51.8 ± 0.3	110.4 ± 0.5	54.3 ± 0.3
VAE+flow posterior	26.7 ± 0.3	55.1 ± 0.3	143.6 ± 0.8	67.9 ± 0.3
GLF (ours)	8.2 ± 0.1	21.3 ± 0.2	88.3 ± 0.4	53.2 ± 0.2
GLF+perceptual loss (ours)	5.8 ± 0.1	10.3 ± 0.1	44.6 ± 0.3	41.8 ± 0.2

E ISSUES WITH LATENT REGULARIZATION

In this appendix section, we present the plots of FID scores, reconstruction loss and NLL loss of regularized GLF. Related results are discussed in Section 4.2.1.

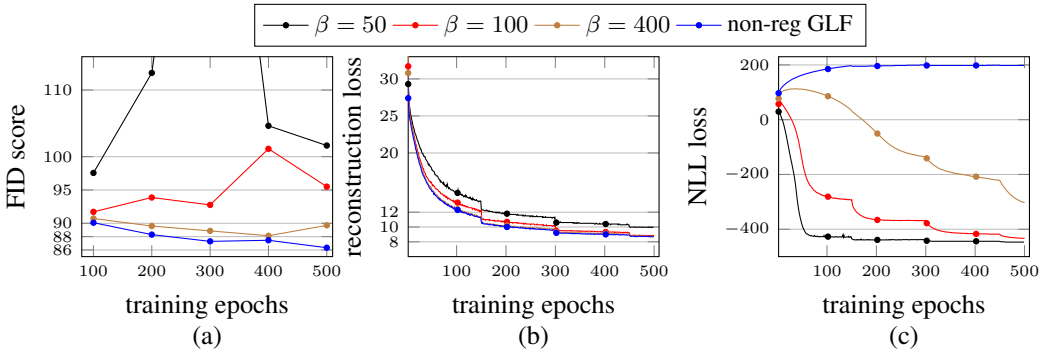


Figure 4: (a) Record of FID scores on CIFAR-10 for regularized GLF with different values of β and GLF. $\beta = 1$ and 10 are omitted because they leads to divergence in reconstruction loss. (b) Record of reconstruction losses for corresponding models. (c) Record of NLL losses for corresponding models.

F TRAINING TIME COMPARISONS

In Table 5, we report the number of training epochs of our method, two-stage VAE and GLANN. In Table 6, we report the clock training time per epoch of these methods. Note that for methods using perceptual loss, the per epoch training time is longer because VGG activations need to be computed. These two tables show that GLF needs much shorter training time than the two competing methods. In GLF, training the flow does not add much computational time due to the low dimensionality.

Table 5: Number of training epochs for Two-stage VAE, GLANN, and GLF

	MNIST/Fashion	CIFAR-10	CelebA
Two-stage VAE First/Second	400/800	1000/2000	120/300
GLANN First/Second	500/50	500/50	500/50
GLF	100	200	40

Table 6: Per-epoch training time in seconds

	MNIST/Fashion	CIFAR-10	CelebA
2-stage VAE 1st/2nd	5/2	6/2	60/28
GLF	10	13	108
GLANN with perceptual loss	14	16	292
GLF with perceptual loss	16	19	343

G MORE QUALITATIVE RESULTS

In Figure 5, we show more samples of each dataset generated by GLF, using either MSE or perceptual loss as reconstruction loss. In Figure 6, we show samples of CelebA-HQ datasets from GLF trained with perceptual loss. In Figure 7, we show examples of interpolations between two randomly sampled noises on CelebA from GLF trained with perceptual loss.

(a) MNIST (b) Fashion MNIST (c) CIFAR-10 (d) CelebA

(e) MNIST (f) Fashion MNIST (g) CIFAR-10 (h) CelebA

Figure 5: (a)-(d) Randomly generated samples from our method with MSE loss. (e)-(h) Randomly generated samples from our method with perceptual loss.

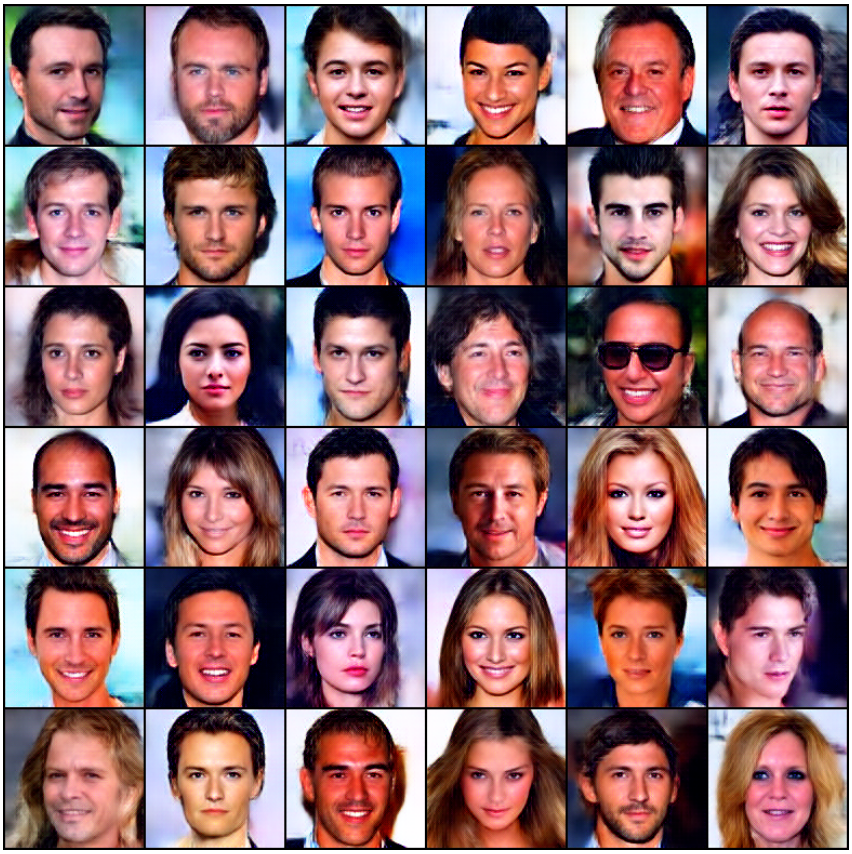


Figure 6: Randomly generated samples from our method with perceptual loss on CelebA-HQ dataset

Figure 7: Noise interpolation on CelebA