TRAINING DEEP GENERATIVE MODELS VIA AUXILIARY SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep generative modeling has long been viewed as a challenging unsupervised learning problem, partly due to the lack of labels and the high dimension of the data. Although various latent variable models have been proposed to tackle these difficulties, the latent variable only serves as a device to model the observed data, and is typically averaged out during training. In this article, we show that by introducing a properly pre-trained encoder, the latent variable can play a more important role, which decomposes a deep generative model into a supervised learning problem and a much simpler unsupervised learning task. With this new training method, which we call the auxiliary supervised learning (ASL) framework, deep generative models can benefit from the enormous success of deep supervised learning and representation learning techniques. By evaluating on various synthetic and real data sets, we demonstrate that ASL is a stable, efficient, and accurate training framework for deep generative models.

1 INTRODUCTION

In recent years deep generative models have received immense interest from deep learning researchers and practitioners, and have become an essential part of unsupervised learning techniques. Representative deep generative models include variational autoencoders (VAE, Kingma & Welling, 2014), generative adversarial networks (GAN, Goodfellow et al., 2014), normalizing flows (Rezende & Mohamed, 2015), autoregressive models (Larochelle & Murray, 2011; Gregor et al., 2014), diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020), and energy-based models (EBM), among many others. The fundamental goal of deep generative modeling is to train a statistical distribution $p_\theta(x)$ based on observed data points $X_1, \ldots, X_n \in \mathcal{X} \subset \mathbb{R}^p$. While this goal is shared by many classical statistical and machine learning models, deep generative models have gained great popularity due to their unique features and focuses. First, deep generative models vastly benefit from the advance of deep neural networks, leading to highly expressive probability models. Second, the distribution $p_\theta(x)$ in deep generative models does not need to be explicitly specified. In fact, it is more commonly expressed by the simulation and sampling mechanism. Third, the data space $\mathcal{X}$ is in general high-dimensional, but data are assumed to approximately lie on a d-dimensional manifold with $d \ll p$, so that the intrinsic dimension of the data is much smaller than the ambient dimension.

To reflect such properties, many deep generative models assume that each data point $X \in \mathcal{X} \subset \mathbb{R}^p$ is generated by a low-dimensional latent variable $Z \in \mathcal{Z} \subset \mathbb{R}^d$ via the model

$$p_\theta(x) = \int_{\mathcal{Z}} p(z) p_\theta(x|z) \mathrm{d}z, \tag{1}$$

where $p(z)$ is the marginal distribution of Z , and $p_\theta(x|z)$ is the conditional distribution of X given the latent variable Z. In most cases, $p(z)$ is taken to be a fixed distribution such as the standard normal, and $p_\theta(x|z)$ utilizes deep neural networks to enrich the class of distributions that $p_\theta(x)$ can express. One such example is VAE, which typically chooses $p_\theta(x|z) = N(f_\theta(z), \sigma^2 I)$ with a deep neural network $f_\theta(z)$ when X is continuous.

In the latent variable model (1), Z is unobserved without actual data points, so it is only used as a device for modeling X, and would typically be integrated out in the objective function. As a result, deep generative models are commonly viewed as challenging unsupervised learning problems, as

the only given data are X. On the other hand, a great part of the deep learning innovation was driven by various supervised learning problems including regression and classification. How to make full use of supervised learning techniques in deep generative models remains an open question.

In this article, we develop the auxiliary supervised learning (ASL) framework for training deep generative models. ASL augments the data with a carefully chosen auxiliary random vector, and converts the original unsupervised learning problem into a supervised model coupled with a smaller unsupervised problem. This seemingly trivial reformulation of the learning objective turns out to have a great potential to improve and generalize existing methods.

One core idea of ASL is to materialize the unobserved auxiliary variable via representation learning. We show that with suitable choice of the auxiliary variable, the observed data can be well predicted by their latent representations, which allows the use of advanced supervised learning techniques. To meet this goal we propose a new method to construct useful latent representations of high-dimensional data. As a whole, the highlights of this article are as follows:

- We develop an intuitive and powerful framework to train deep generative models, which benefits from advances in deep supervised learning and representation learning techniques.
- We propose a novel mutual-information-based method to capture the latent structure of the data, which shows promising results compared with conventional methods such as VAE.
- The resulting training method is easy to implement, avoids heavy manual tuning, and enjoys excellent stability.
- Numerical experiments demonstrate the superior performance of ASL on various synthetic and real data sets.

2 Auxiliary Supervised Learning for Deep Generative Models

2.1 Motivation

The ASL framework is motivated by a simple and familiar identity, the decomposition of joint distributions. For the data random vector X, we can always augment it by introducing an auxiliary random vector $U \in \mathbb{R}^r$ and define their joint distribution $p(x, u)$. Here U is arbitrary and does not need to be the latent variable Z in model (1). Clearly, $p(x, u)$ admits two ways of decomposition,

$$p(x, u) = p_x(x)p_{u|x}(u|x) = p_u(u)p_{x|u}(x|u). \tag{2}$$

If U is taken to be the latent variable Z in deep generative models, then (1) is equivalent to the second form of the decomposition. Whereas in this article, we show that the first form provides another view of the data generation process, and can play an important role in training deep generative models. Since the data distribution $p_x(x)$ is fixed, the joint distribution $p(x, u)$ is completely determined by the conditional distribution $p_{u|x}(u|x)$, which we call an *encoder* following the convention of the VAE literature.

The importance and usefulness of the encoder can be illustrated in the following way. Suppose that $p_{u|x}(u|x)$ is pre-trained and can be used to generate $U \sim p_{u|x}(u|x)$ for any x. Then for each data point X_i, we can simulate $U_i \sim p_{u|x}(u|X_i)$, resulting in an augmented data set $(X_i, U_i), i = 1, \ldots, n$. The data point pair (X_i, U_i) exactly follows the joint distribution given by (2), so the second form of decomposition also holds, *i.e.*, $(X_i, U_i) \sim p_u(u)p_{x|u}(x|u)$.

The observation above comprises the core idea of ASL, which advocates a better use of the latent representation of the data. With *any* given encoder $p_{u|x}(u|x)$, we can form the augmented data set $\{(X_i, U_i)\}_{i=1}^n$, and our task is to estimate $p_u(u)$ and $p_{x|u}(x|u)$ based on (X_i, U_i). Once these two distributions are trained, a new data point of X can be easily generated by the following sampling process:

$$U^* \sim p_u(u), \quad X^* \sim p_{x|u}(x|U^*).$$

If the dimension of U is small, then $p_u(u)$ can be estimated from the marginal data $\{U_i\}_{i=1}^n$, which is a much simpler task than estimating $p_x(x)$ itself. Therefore, the main challenge in the ASL framework is to train the conditional distribution $p_{x|u}(x|u)$. But since we have full access to the paired data $\{(X_i, U_i)\}_{i=1}^n$, estimating $p_{x|u}(x|u)$ is essentially a ***supervised*** learning problem, which

is a well-studied area in machine learning. The proposed framework, auxiliary supervised learning, obtains its name exactly from this part: we convert the original challenging unsupervised learning problem into a simple low-dimensional density estimation task and a supervised learning problem, in which the "labels" are the original data X_i, and the "features" are the artificial and auxiliary variables U_i. A diagram of the ASL framework is shown in Figure 1.

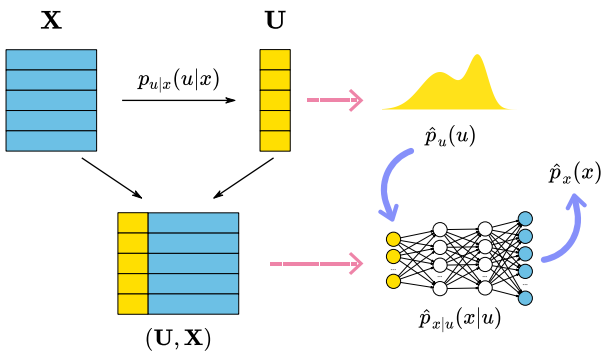


Figure 1: An illustration of the ASL framework. The auxiliary data points U_i are generated from X_i by the encoder $p_{x|u}(x|u)$, forming the paired data (X_i, U_i). Marginal distribution $p_u(u)$ is estimated from the U_i data, and generator distribution $p_{x|u}(x|u)$ is learned in a supervised fashion based on (X_i, U_i). New data points of X are generated by simulation of $U^* \sim p_u(u)$, followed by the generator $X^* \sim p_{x|u}(x|U^*)$.

2.2 Supervised Learning Can be Hard

Although it is commonly considered that supervised learning possesses certain advantages over unsupervised learning, we emphasize that the modeling and training of $p_{x|u}(x|u)$ is not necessarily an easy task. As an extreme example, if U is completely independent of X, then learning $p_{x|u}(x|u)$ is the same as learning $p_x(x)$ itself, which of course is a hard problem. Therefore, the choice of the encoder $p_{u|x}(u|x)$ has an impact on the difficulty in learning $p_{x|u}(x|u)$. We defer such discussions to Section 3, where we give conditions on which supervised learning is beneficial, and provide concrete methods to achieve this goal.

2.3 Outline of the Framework

Assuming that the encoder $p_{u|x}(u|x)$ is given or pre-trained, the ASL framework consists of the following three steps.

I. Simulation step In this step we generate the paired data points (X_i, U_i) by sampling from the encoder. Specifically, the encoder $p_{u|x}(u|x)$ can be parameterized by a deterministic mapping $f_\phi : \mathbb{R}^{p+r} \to \mathbb{R}^r$ such that $U = f_\phi(x, \varepsilon) \sim p_{u|x}(u|x)$ with $\varepsilon \sim N(0, I_r)$. This formulation is a generalization to the reparameterization trick commonly used by VAE (Kingma & Welling, 2014). For example, one can take $f_\phi(x, \varepsilon) = \mu(x) + \sigma(x) \odot \varepsilon$, where $\mu, \sigma : \mathbb{R}^p \to \mathbb{R}^r$ are deep neural networks and $\odot$ means elementwise multiplication. Of course, more general forms of f_ϕ exist. With this formulation, U_i is simulated as $U_i = f_\phi(X_i, \varepsilon_i)$, where ε_i are i.i.d. $N(0, I_r)$ random vectors independent of all X_i.

II. Density estimation step Since the dimension of U is typically much smaller than X, many existing methods can be used to efficiently estimate $p_u(u)$, including normalizing flows, EBM, Gaussian mixture models, etc. Dai & Wipf (2019) also shows that a vanilla VAE can be used to estimate a low-dimensional density accurately if the latent dimension equals the ambient dimension. Although theoretically adversarial training can also be used to learn $p_u(u)$, our numerical experiments show that likelihood-based methods such as affine coupling normalizing flows (Dinh et al., 2014) are good enough for this step, and have the huge advantage of stable training. For this reason, we only show the likelihood-based training method in Algorithm 1 with a generic density model $p_\eta(u)$ for $p_u(u)$.

III. Supervised learning step For now, we simply assume that $p_{x|u}(x|u)$ can be modeled by a regression model, $X = g_\theta(U) + \varepsilon$, where $g_\theta : \mathbb{R}^r \to \mathbb{R}^p$ is a deep neural network, and ε is a zero-mean error term. As a convention of supervised learning, we define a loss function $\ell(x, \hat{x})$ to measure the difference between the true values of X and its predicted values. Common choices of

ℓ include $\ell(x, \hat{x}) = \frac{1}{2}\|x - \hat{x}\|_2^2$ and $\ell(x, \hat{x}) = \|x - \hat{x}\|_1$, among many others. The neural network parameters θ are then learned via stochastic gradient methods. After g_θ is trained, we can use the residuals $X_i - g_\theta(U_i)$ to estimate the covariance of ε under suitable distribution assumptions.

In practice, the simulation step of ASL can be implemented within each iteration of step II and step III. In this way the U_i data points have more variations, making the two training steps II and III less prone to over-fitting. Also note that the density estimation and supervised learning steps are completely independent of each other, so $p_u(u)$ and $p_{x|u}(x|u)$ can be trained in parallel, saving visible computing time. As a whole, the outline of ASL is summarized in Algorithm 1.

Algorithm 1 Outline of the ASL framework for training deep generative models

Input: Data $\{X_i\}_{i=1}^n$, pre-trained encoder f_ϕ, density model p_η, regression model g_θ

Output: Estimated neural network parameters $\hat{\eta}$, $\hat{\theta}$

for $k = 1, 2, \ldots, K$ **do**
  Sample mini-batch indices $I_1, \ldots, I_M$
  Sample $U_{I_b} \leftarrow f_\phi(X_{I_b}, \varepsilon_b)$, $\varepsilon_b \overset{iid}{\sim} N(0, I_r)$, $b = 1, \ldots, M$
  Define $L_1(\eta) = -M^{-1} \sum_{b=1}^M \log p_\eta(U_{I_b})$
  Update $\eta_k \leftarrow \eta_{k-1} - \alpha_k \nabla L_1(\eta_{k-1})$
end for

Density estimation

↓

In parallel

↑

Supervised learning

for $k = 1, 2, \ldots, K$ **do**
  Sample mini-batch indices $I_1, \ldots, I_M$
  Sample $U_{I_b} \leftarrow f_\phi(X_{I_b}, \varepsilon_b)$, $\varepsilon_b \overset{iid}{\sim} N(0, I_r)$, $b = 1, \ldots, M$
  Define $L_2(\theta) = M^{-1} \sum_{b=1}^M \ell(X_{I_b}, g_\theta(U_{I_b}))$
  Update $\theta_k \leftarrow \theta_{k-1} - \alpha_k \nabla L_2(\theta_{k-1})$
end for

return $\hat{\eta} = \eta_K$, $\hat{\theta} = \theta_K$

3 Encoders That Make Supervised Learning Easy

As mentioned in Section 2.2, the supervised learning of $p_{x|u}(x|u)$ is not necessarily easy. In general, if the conditional distribution $p_{x|u}(x|u)$ has a complicated form other than simple distributions such as normal, then the training of $p_{x|u}(x|u)$ is also difficult, even when the paired data (X_i, U_i) are fully observed. However, $p_{x|u}(x|u)$ is determined by the encoder $p_{u|x}(u|x)$, which is free to choose in the ASL framework. In Sections 3.1 and 3.2 below, we provide two schemes for pre-training encoders that make the supervised learning part easy.

3.1 VAE-based Method

The first case is when the observed data are *exactly* restricted to a d-dimensional manifold. Dai & Wipf (2019) proves that an optimally trained normal VAE with latent dimension $r \geq d$ can perfectly reconstruct X by sampling $U^* \sim q(u)$ and $X \sim p_{x|u}(x|U^*)$, where $q(u) = \int p_{u|x}(u|x)p(x)\mathrm{d}x$, and $p_{u|x}(u|x)$, $p_{x|u}(x|u)$ are the learned encoder and decoder from VAE, respectively. In other words, a simple normal model for $p_{x|u}(x|u)$ suffices.

As a result, under the assumption above, the learned encoder in VAE is also a valid encoder in the ASL framework, and step II and step III of ASL can be viewed as learning the prior $p_u(u)$ and fine-tuning the decoder $p_{x|u}(x|u)$ after training a VAE model. We use the name VAE-ASL to stand for ASL training methods based on a VAE encoder.

3.2 A New MI-based Method

More generally, X only *approximately* lies on a manifold, so we cannot solely rely on VAE to obtain the encoder. The second case to make $p_{x|u}(x|u)$ easy is when this conditional distribution has a small uncertainty given $U = u$. For example, suppose that $p_{x|u}(x|u)$ is primarily determined

by its conditional mean $f(u)$, then $p_{x|u}(x|u)$ can be well approximated by a degenerate distribution concentrated at $f(u)$, and the estimation of $f(\cdot)$ is basically a regression problem.

To achieve this goal, the auxiliary variable U should capture most of the information contained in X, so pre-training the encoder is essentially a representation learning problem. One formal criterion to measure the information overlap between two random vectors is mutual information (MI), defined as

$$\mathcal{I}(X,U) = \int p(x,u)\log\frac{p(x,u)}{p_x(x)p_u(u)}\mathrm{d}x\mathrm{d}u = \mathcal{D}_{\mathrm{KL}}[p(x,u)\|p_x(x)p_u(u)],$$

where $\mathcal{D}_{\mathrm{KL}}(\cdot\|\cdot)$ is the Kullback–Leibler divergence. An interpretation of MI is given by the identity $\mathcal{I}(X,U) = \mathcal{H}(X) - \mathcal{H}(X|U)$, where $\mathcal{H}(X)$ and $\mathcal{H}(X|U)$ are the differential entropy and conditional differential entropy, respectively. Since $\mathcal{H}(X)$ is fixed, a large $\mathcal{I}(X,U)$ implies that $\mathcal{H}(X|U)$ is small, meaning that the uncertainty about X after observing U is small.

Naturally, one would hope to seek the encoder $p_{u|x}(u|x)$ such that $\mathcal{I}(X,U)$ is maximized. However, directly estimating MI is intractable, so a number of lower bounds of MI have been developed Poole et al. (2019). But still, it was pointed out that these estimators are typically biased and exhibit high variance (Tschannen et al., 2020). To overcome such difficulties, in Theorem 1 we propose a new easy-to-compute lower bound for MI.

Theorem 1. *Let $(X,U) \in \mathbb{R}^p \times \mathbb{R}^r$ be a pair of continuous random vectors with joint distribution $p(x,u)$. Then for any mapping $g : \mathbb{R}^r \to \mathbb{R}^p$ and any vector $a = (a_1,\ldots,a_p)^{\mathrm{T}}$ with $a_i > 0$, $i = 1,\ldots,p$, denote $\xi = g(U)$, and we have*

$$\mathcal{I}(X,U) \geq \mathcal{I}_{\mathrm{ASL}}(X,U;g,a) \coloneqq \mathcal{H}(X) - \frac{1}{2}\sum_{i=1}^{p}\left[\frac{\mathbb{E}_{p(x,u)}(X_i-\xi_i)^2}{a_i} + \log(a_i)\right] - \frac{p\log(2\pi)}{2}.$$

Since g and a are arbitrary, one can use a deep neural network to represent g, and optimize g and a jointly to tighten the bound.

One issue of MI is that it is invariant to invertible transformations of the variables, so the marginal distribution of U can be arbitrary. To stabilize training and enhance identifiability, we need a regularizer to force U close to a fixed distribution. Define

$$k(u,u') = \frac{4\beta(\beta+1)\|u-u'\|^2}{(1+\|u-u'\|^2)^{\beta+2}} + 2\beta\cdot\frac{r-\|u-u'\|^2}{(1+\|u-u'\|^2)^{\beta+1}} + \frac{u^{\mathrm{T}}u'}{(1+\|u-u'\|^2)^{\beta}}$$

for some $\beta > 0$, and then $\mathcal{D}_{\mathrm{Stein}}(p_u) = \mathbb{E}_{u,u'\sim p_u}[k(u,u')]$ is the kernelized Stein discrepancy between $p_u(u)$ and $N(0,I_r)$ using an inverse multi-quadric kernel (Liu et al., 2016; Hu et al., 2018). In general, a small value of $\mathcal{D}_{\mathrm{Stein}}(p_u)$ indicates that p_u is close to a standard normal distribution.

Note that both $\mathcal{I}_{\mathrm{ASL}}(X,U;g,a)$ and $\mathcal{D}_{\mathrm{Stein}}(p_u)$ can be unbiasedly estimated from samples of (X_i,U_i), which indicates that we do not require the encoder $p_{u|x}(u|x)$ to have a tractable density function. This greatly frees us from the VAE-based encoders that assume $p_{u|x}(u|x)$ is normal. Instead, we represent the encoder using a deterministic deep neural network f_ϕ by generating U as $U = f_\phi(X,\varepsilon), \varepsilon \sim N(0,I_r)$.

As a whole, we propose to estimate f_ϕ, g, and a jointly by minimizing the following objective function:

$$\min_{f_\phi,g,a} -\mathcal{I}_{\mathrm{ASL}}(X,U;g,a) + \lambda\mathcal{D}_{\mathrm{Stein}}(p_u), \tag{3}$$

where λ is a weight parameter. It should be noted that the main purpose of (3) is to obtain the encoder f_ϕ, whereas g and a are merely auxiliary quantities to tighten the MI bound. However, g has the same structure as the regression function g_θ in step III of ASL, and in practice we find that g provides an excellent initial value for g_θ in the supervised learning step. We use the name MI-ASL to stand for ASL training based on (3).

3.3 Decoder-free Methods

Both VAE-ASL and MI-ASL simultaneously pre-train the encoder f_ϕ and the regression function g_θ. There are also methods that directly optimize f_ϕ without including g_θ, such as the deep InfoMax

(Hjelm et al., 2019) and the augmented multiscale deep InfoMax (Bachman et al., 2019). More generally, methods that can learn latent representations of high-dimensional data can all be viewed as candidates for the encoder in our framework.

4 Related Work

Both VAE and ASL put an emphasis on the latent representations of the data. However, ASL has some fundamental differences with VAE. As pointed out by Zhao et al. (2018), the objective function of VAE is equivalent to *matching* the two forms of decomposition by minimizing $\mathcal{D}_{\mathrm{KL}}[p_x(x)p_{u|x}(u|x)\|p_u(u)p_{x|u}(x|u)] \equiv -\mathrm{ELBO} - \mathcal{H}(X)$. However, in most cases p_u, $p_{u|x}$, and $p_{x|u}$ are all taken to be Gaussian, so there is almost always some discrepancy between the two forms of decomposition. Even though some articles propose to use more complicated distributions for the prior p_u and the posterior $p_{u|x}$ (Rezende & Mohamed, 2015; Salimans et al., 2015; Kingma et al., 2016; Tomczak & Welling, 2018; Bauer & Mnih, 2019), and there are various works on improving the optimization process of VAE (Kim et al., 2018; He et al., 2019; Fu et al., 2019), we shall point out that VAE does not necessarily generate good latent representations. This is because matching $p_x p_{u|x}$ with $p_u p_{x|u}$ does not mean U and X share information. In this sense, the VAE objective function has an intrinsic limitation on representation learning.

ASL, on the other hand, *utilizes* the identity between the two decompositions, so it mitigates the distribution matching problem of VAE and is less restricted by the model capacity. Moreover, in MI-ASL we explicitly forces the latent U to have large information overlap with X, thus ensuring the quality of the latent representation.

Another line of works have focused on refining existing pre-trained models. For example, Che et al. (2020) proposes to improve the latent space of GAN via rejection sampling, which uses the discriminator from an existing GAN to implement an EBM on latent variables. Dai & Wipf (2019) proposes the two-stage VAE, which enhances a VAE model by using another VAE to estimate the aggregated posterior. While these works emphasize on the resampling of priors, the ASL framework points out that the decoder also needs further training.

5 Numerical Experiments

5.1 Low-dimensional Synthetic Data

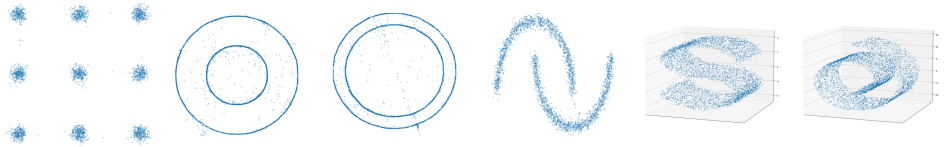
Although deep generative models are most useful for modeling high-dimensional data, it is helpful to first examine their performances on some low-dimensional yet challenging synthetic data sets. Figure 2(a) shows the true data points of six synthetic data sets, which exhibit properties such as multi-modality, nearly singular densities, manifold embedding, etc. We emphasize that although these data only have two or three dimensions, correctly estimating their distributions is a highly difficult task. To illustrate this point, we use VAE and normalizing flows to fit the data, with results shown in Figure 8 in the appendix. It shows that even for such low-dimensional data, standard methods can have surprisingly bad output. In contrast, Figure 2(b) gives the generated samples by ASL with latent dimension $r = 2$, implying that ASL successfully recovers the true distributions with only minimal deviations.

To quantitatively measure the quality of generated samples, we compute the 1-Wasserstein distance between the true and generate samples for each data set and each model. We repeat each experiment 30 times, and summarize the distributions of these distances in Figure 3. It is clear that ASL demonstrates satisfactory performance and stability.

5.2 Evaluation on Latent Representation

As discussed in Section 3, the encoder can play a critical role in extracting useful information from high-dimensional data. We then use the Fashion-MNIST data set (Xiao et al., 2017) to evaluate the performance of two encoder pre-training schemes: VAE-ASL and MI-ASL. Specifically, we train the two encoders with latent dimension $r = 32$, and then individually simulate U_i based on the same set of X_i data points. To visualize the structure of the generated auxiliary data points, we use t-SNE (Van der Maaten & Hinton, 2008) to project the 32-dimensional U_i onto a two-dimensional plane, as

(a) True samples



(b) Samples generated by ASL

Figure 2: True samples of six synthetic data sets and randomly generated data points by ASL.

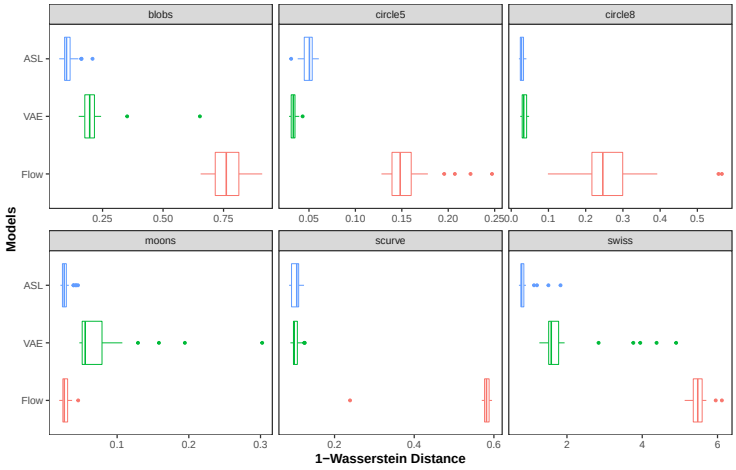


Figure 3: Quantitative evaluation of three generative models on the synthetic data sets. Each boxplot shows the distribution of the 1-Wasserstein distance between the true and generated data points.

is demonstrated in Figure 4. Surprisingly, the VAE-based encoder fails to capture the latent structure of the original data, as it is known that Fashion-MNIST consists of ten classes. In contrast, the new MI-based encoder clearly reveals the clusters that are well aligned with the true labels. This finding highlights the advantage of the MI-based encoder, and suggests that ASL would benefit from new advances in representation learning techniques.

5.3 Evaluation on Generation

After the encoders are pre-trained, we then use ASL to build generative models for the Fashion-MNIST data. We first examine the reconstructed images formed by passing the original data points to the encoder f_ϕ and the generator g_θ, which shed light on the best quality the generative models can achieve. Figure 5 shows the true and reconstructed images side by side, from which we can see that images reconstructed by MI-ASL have sharper edges than those by VAE-ASL, and preserve more details on the apparel. This finding is consistent with Figure 4, as we have shown that the MI-based encoder is more powerful in learning the latent structure. On the other hand, although the VAE encoder has poor performance in the latent space, ASL turns out to be very robust to this choice, which has acceptable quality on the reconstructed images.

To further test the generalization ability, we randomly generate images from various learned models, and compare ASL with the two-stage VAE model proposed by Dai & Wipf (2019). Two-stage VAE is an improved VAE model that has been shown to achieve good generation quality. Figure 6(a)

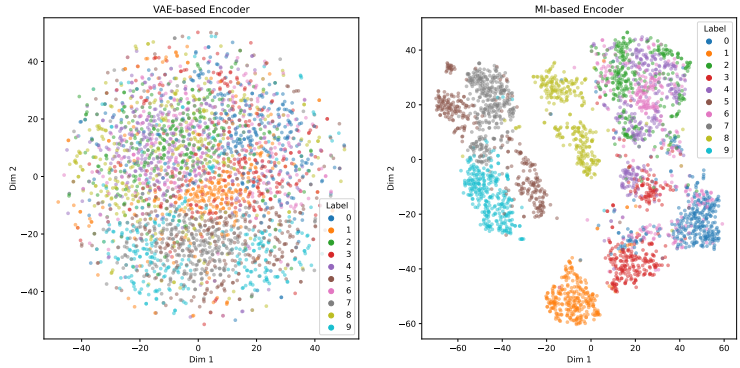
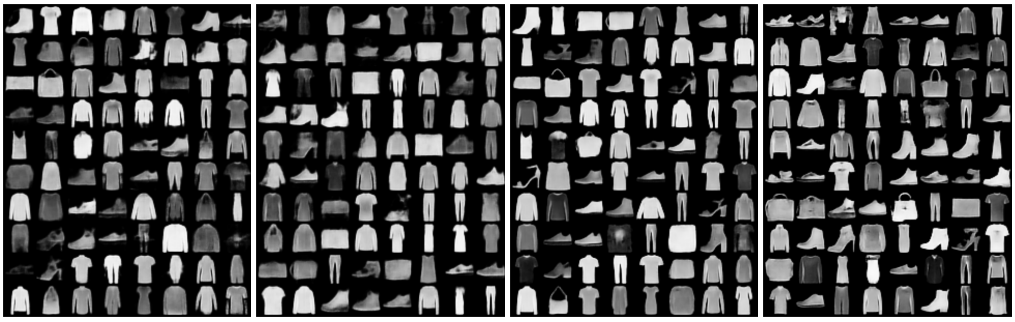


Figure 4: t-SNE visualization of the auxiliary data points generated by the VAE-based and MI-based encoders, respectively.

Figure 5: True and reconstructed images from the Fashion-MNIST data. Left: true images from the testing set. Middle: images reconstructed by VAE-ASL. Right: images reconstructed by MI-ASL.

shows the results by vanilla VAE, which obviously contain many blurry images. The two-stage VAE has significant improvement over VAE, but it still tends to over-simplify the patterns on the clothes. Finally, the two ASL models in Figures 6(c) and (d) are able to generate visible marks on shoes and handbags, which is non-trivial for Fashion-MNIST data. Overall, ASL has desirable performance in both reconstruction and generation tasks.



(a) VAE (b) Two-stage VAE (c) VAE-ASL (d) MI-ASL

Figure 6: Randomly generated samples from various deep generative models.

To show the quantitative performance of ASL, we also fit the CIFAR-10 data (Krizhevsky et al., 2009) with generated images shown in Figure 9, and use the Fréchet inception distance (FID, Heusel et al., 2017) as a metric for image generation quality. Table 1 gives the means and standard deviations of the FID scores of the compared methods. Note that we primarily show the relative performance of

Table 1: FID scores for different generative models. The numbers in the parentheses are the standard deviations across ten independent runs.

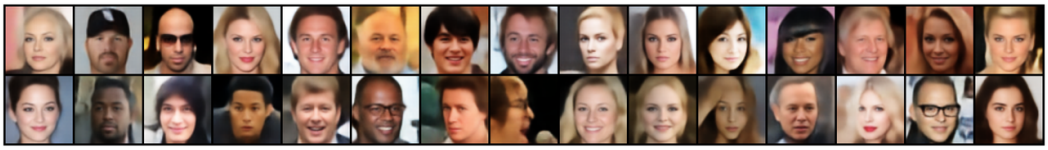
	VAE	Two-stage VAE	VAE-ASL	MI-ASL
Fashion-MNIST	47.19 (0.35)	43.88 (0.35)	38.39 (0.50)	**26.00** (0.28)
CIFAR-10	99.34 (0.25)	89.91 (0.26)	**86.51** (0.21)	87.19 (0.28)

ASL and two-stage VAE under a fixed network architecture, and do not pursue the lowest possible scores. The comparison of two-stage VAE and many other generative models can be found in Dai & Wipf (2019) and Xiao et al. (2019). Overall, the results indicate that ASL is competitive with modern deep generative models.

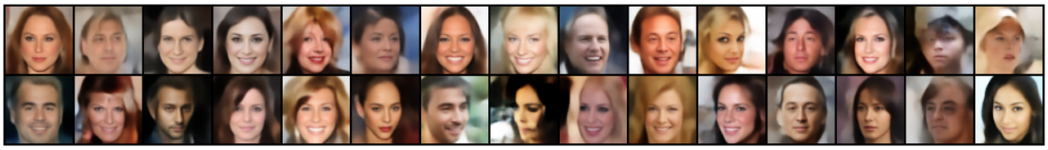
5.4 CelebA Human Face Data

Finally, we show qualitative results of ASL on human face image generation based on the large scale CelebA data (Liu et al., 2015). Figure 7 illustrates the true, reconstructed, and randomly generated images by ASL based on the MI encoder. A larger collection of generated samples are given in Figure 10 in the appendix, which also includes the results of VAE-ASL. Both variants of ASL are able to generate high-quality images, and more importantly, they do not require heavy tuning of hyperparameters. To show that ASL does not simply memorize data, we interpolate between images in the latent space, and demonstrate the results in Figure 11. It shows that the intermediate images are also meaningful. Furthermore, we plot various loss function values of ASL in Figure 12, which validates the stability of training in ASL. Overall, ASL is shown to be a promising framework that enjoys high generation quality, requires little manual tuning, and is extremely stable during training.

(a) True images



(b) Reconstructed images



(c) Randomly generated images

Figure 7: True images from the CelebA data and model-generated images based on ASL.

6 Discussion

In this article we propose the ASL framework for training deep generative models. ASL is conceptually simple, but exhibits great potential to improve the training of complicated generative models. At the core of the framework is the use of representation learning to construct informative auxiliary variables, and the application of supervised learning to train deep neural networks. Various numerical experiments demonstrate the accuracy and stability of ASL. Most importantly, ASL provides

a general framework that is not limited to a specific method. The architecture of ASL is highly modularized, and one has great flexibility in choosing the models for each component of ASL, including the encoder, density estimator, supervised learning algorithm, etc. We anticipate that under this framework deep generative models would greatly benefit from the advances of other subfields of deep learning.

Reproducibility Statement To enhance the reproducibility of this work, we provide the implementation details for the numerical experiments presented in this article. Source code, pre-trained model files, and output plots can be obtained at `https://drive.google.com/drive/folders/1emu6X4Rkxu_ft1HRVPGZAxKfYmPnPGt4`.

REFERENCES

Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems 32*, pp. 15509–15519, 2019. 3.3

Matthias Bauer and Andriy Mnih. Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 66–75, 2019. 4

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. In *Advances in Neural Information Processing Systems 33*, 2020. 4

Thomas M Cover and Joy A Thomas. *Elements of information theory, 2nd Edition*. 2006. 1

Bin Dai and David P. Wipf. Diagnosing and enhancing VAE models. In *The 7th International Conference on Learning Representations*, 2019. 2.3, 3.1, 4, 5.3, 5.3

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 2.3

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019. 4

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014. 1

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *The 31st International Conference on Machine Learning*, pp. 1242–1250, 2014. 1

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *7th International Conference on Learning Representations*, 2019. 4

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 30*, 2017. 5.3

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *The 7th International Conference on Learning Representations*, 2019. 3.3

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33*, pp. 6840–6851, 2020. 1

Tianyang Hu, Zixiang Chen, Hanxi Sun, Jincheng Bai, Mao Ye, and Guang Cheng. Stein neural sampler. *arXiv preprint arXiv:1810.03545*, 2018. 3.2

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pp. 2678–2687. PMLR, 2018. 4

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *The 2nd International Conference on Learning Representations*, 2014. 1, 2.3

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 29*, 2016. 4

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5.3

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *The Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011. 1

Qiang Liu, Jason D. Lee, and Michael I. Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *The 33rd International Conference on Machine Learning*, pp. 276–284, 2016. 3.2

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *The IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015. 5.4

Ben Poole, Sherjil Ozair, Aäron van den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *The 36th International Conference on Machine Learning*, pp. 5171–5180, 2019. 3.2

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *The 32nd International Conference on Machine Learning*, pp. 1530–1538, 2015. 1, 4

Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *The 32nd International Conference on Machine Learning*, pp. 1218–1226, 2015. 4

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *The 32nd International Conference on Machine Learning*, pp. 2256–2265, 2015. 1

Jakub M. Tomczak and Max Welling. VAE with a vampprior. In *The 21st International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223, 2018. 4

Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *8th International Conference on Learning Representations*, 2020. 3.2

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 5.2

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 5.2

Zhisheng Xiao, Qing Yan, and Yali Amit. Generative latent flow. *arXiv preprint arXiv:1905.10485*, 2019. 5.3

Shengjia Zhao, Jiaming Song, and Stefano Ermon. The information autoencoding family: A lagrangian perspective on latent variable generative models. *arXiv preprint arXiv:1806.06514*, 2018. 4

A APPENDIX

A.1 PROOF OF THEOREM 1

Let $|A|$ denote the determinant of a square matrix A. We first introduce the well-known result that for a given covariance matrix, the differential entropy is maximized by the normal distribution.

Lemma 1 (Theorem 8.6.5, Cover & Thomas, 2006)**.** *Let the random vector $X \in \mathbb{R}^p$ have zero mean and covariance matrix Σ. Then*

$$\mathcal{H}(X) \leq \frac{p}{2} + \frac{p\log(2\pi)}{2} + \frac{1}{2}\log|\Sigma|,$$

with equality if and only if $X \sim N(0, \Sigma)$.

Then we are ready to prove Theorem 1. For any univariate random variable $Z \in \mathbb{R}$ whose second moment exists, it is known that $\mathbb{E}(Z-c)^2$ is minimized by $c = \mathbb{E}(Z)$. Therefore, given $U = u$, each $\mathbb{E}_{X|U=u}(X_i - \hat{X}_i)^2$ is minimized by $\hat{X}^* = \mathbb{E}(X|U=u)$. Let $g : \mathbb{R}^r \to \mathbb{R}^p$ be any mapping and denote $\xi = g(U)$, and then

$$\mathbb{E}_{X|U=u}(X_i - \xi_i)^2 \geq \mathbb{E}_{X|U=u}(X_i - \hat{X}_i^*)^2 = \mathrm{Var}(X_i|U=u),$$

where $\mathrm{Var}(X_i|U=u)$ is the conditional variance of X_i given $U = u$.

Let $\Sigma(u)$ be the conditional covariance matrix of X given $U = u$, and define

$$\mathcal{H}(X|U=u) = -\int p_{x|u}(x|u)\log p_{x|u}(x|u)\mathrm{d}x.$$

Then by Lemma 1, we have

$$\mathcal{H}(X|U=u) \leq \frac{p}{2} + \frac{p\log(2\pi)}{2} + \frac{1}{2}\log|\Sigma(u)|.$$

By Hadamard's inequality,

$$|\Sigma(u)| \leq \prod_{i=1}^{p}\sigma(u)_{ii}^2,$$

where $\sigma(u)_{ii}^2 = \mathrm{Var}(X_i|U=u)$ are the diagonal elements of $\Sigma(u)$. Therefore,

$$\log|\Sigma(u)| \leq \sum_{i=1}^{p}\log\left[\mathrm{Var}(X_i|U=u)\right] \leq \sum_{i=1}^{p}\log\left[\mathbb{E}_{X|U=u}(X_i-\xi_i)^2\right]. \tag{4}$$

For any $x, a > 0$, $\log(x) \leq x/a + \log(a) - 1$, and the equality holds if and only if $a = x$. As a result, (4) indicates that

$$\log|\Sigma(u)| \leq \sum_{i=1}^{p}\left[\frac{\mathbb{E}_{X|U=u}(X_i-\xi_i)^2}{a_i} + \log(a_i) - 1\right] \tag{5}$$

for any vector $a = (a_1, \ldots, a_p)^{\mathrm{T}}$ with $a_i > 0$. Taking the expectation of U on both sides of 5, we get

$$\int p_u(u)\log|\Sigma(u)|\mathrm{d}u \leq \sum_{i=1}^{p}\left[\frac{\mathbb{E}_{p(x,u)}(X_i-\xi_i)^2}{a_i} + \log(a_i) - 1\right].$$

Finally,

$$\mathcal{H}(X|U) = \int p_u(u)\mathcal{H}(X|U=u)\mathrm{d}u \leq \frac{p\log(2\pi)}{2} + \frac{1}{2}\sum_{i=1}^{p}\left[\frac{\mathbb{E}_{p(x,u)}(X_i-\xi_i)^2}{a_i} + \log(a_i)\right],$$

and the required result is obtained.

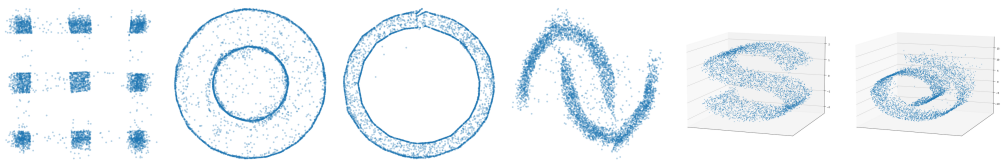
	Fashion MNIST	CelebA
Encoder-VAE	$x \in \mathbb{R}^{28\times 28}$ $\to$ Conv64 $\to$ ReLU $\to$ Conv64 $\to$ ReLU $\to$ Conv128 $\to$ ReLU $\to$Flatten$\to$FC$_{32\times 2}$	$x \in \mathbb{R}^{64\times 64}$ $\to$ Conv64 $\to$ BN $\to$ LeakyReLU $\to$ Conv128 $\to$ BN $\to$ LeakyReLU $\to$ Conv256 $\to$ BN $\to$ LeakyReLU $\to$ Conv512 $\to$ BN $\to$ LeakyReLU $\to$ Conv256$\to$Flatten$\to$FC$_{128\times 2}$
Encoder-MI	$x \in \mathbb{R}^{64\times 64}$ $\to$ Conv64 $\to$ ReLU $\to$ Conv64 $\to$ ReLU $\to$ Conv128 $\to$ ReLU $\to$Flatten$\to 2\times$(FC$_{256}\to$ReLU) $\to$ FC$_{32}$	$x \in \mathbb{R}^{64\times 64}$ $\to$ Conv64 $\to$ BN $\to$ LeakyReLU $\to$ Conv128 $\to$ BN $\to$ LeakyReLU $\to$ Conv256 $\to$ BN $\to$ LeakyReLU $\to$ Conv512 $\to$ BN $\to$ LeakyReLU $\to$ Conv1024$\to$Flatten $\to$Dense$_{256,512,256}$ $\to$ FC$_{128}$
Generator	$u \in \mathbb{R}^{32} \to$ ReLU $\to$FC $\to$ Conv$_{64}$ $\to$ ReLU $\to$ Conv$_{64}$ $\to$ ReLU $\to$ Conv$_1$ $\to$ Flatten $\to$ Sigmoid	$u \in \mathbb{R}^{128}$ $\to$ Conv512 $\to$ BN $\to$ ReLU $\to$ Conv256 $\to$ BN $\to$ LeakyReLU $\to$ Conv256 $\to$ BN $\to$ ReLU $\to$ Conv128 $\to$ BN $\to$ ReLU $\to$ Conv3$\to$Flatten$\to$ Sigmoid
Flow	$u\in \mathbb{R}^{32} \to$[FC$_{256}$ $\to$ReLU6 $\to$ FC$_{256}$ $\to$ReLU6 $\to$ MaskedAC]$_{\times 16}$	$u\in \mathbb{R}^{128} \to$[FC$_{128}$ $\to$ReLU6 $\to$ FC$_{128}$ $\to$ReLU6 $\to$ FC$_{128}$ $\to$ReLU6 $\to$ MaskedAC]$_{\times 20}$

Table 2: Network architectures for the numerical experiments in Section 5.

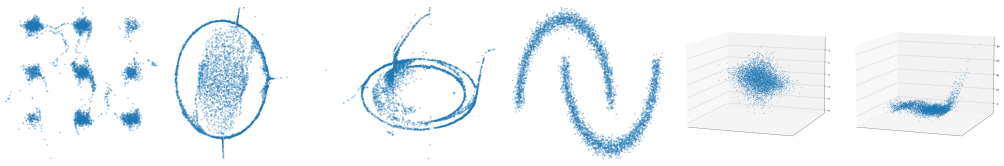
A.2 Neural Network Architectures

The network architectures for the numerical experiments in Section 5 are displayed in Table 2. Here *Dense* is a feed-forward neural network with subscripts indicating the hidden layers' dimensions and ReLU as the activation function. For the low-dimensional synthetic data, all of the networks are *Dense* networks with three hidden layers.

A.3 Additional Visualization Results



(a) Samples generated by VAE



(b) Samples generated by normalizing flow

Figure 8: Randomly generated samples of synthetic data by VAE and normalizing flow.

(a) VAE (b) Two-stage VAE

(c) VAE-ASL (d) MI-ASL

Figure 9: Randomly generated images trained from the CIFAR-10 data set.

(a) MI-ASL (b) VAE-ASL

Figure 10: Randomly generated face images by ASL using two different encoders.

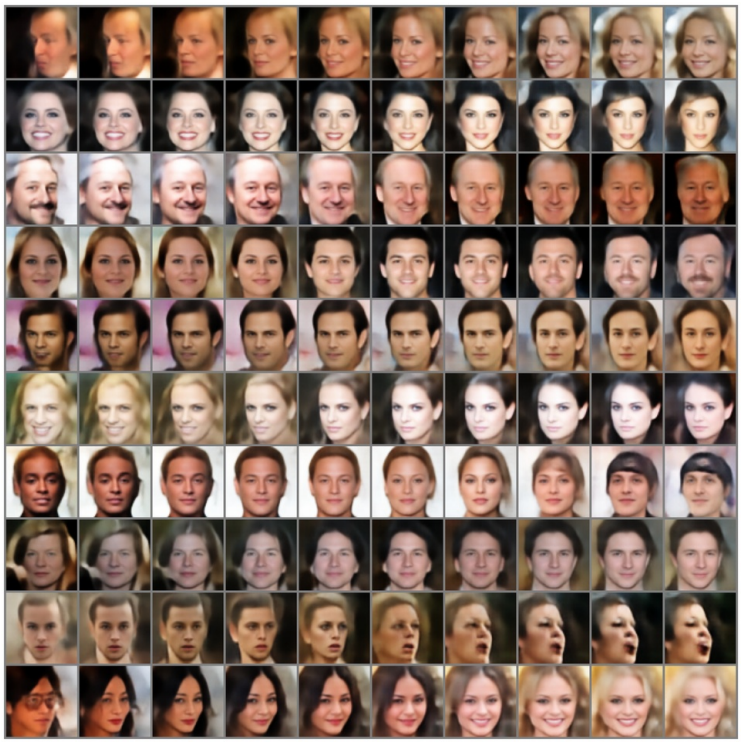


Figure 11: Interpolation of generated images in the latent space.

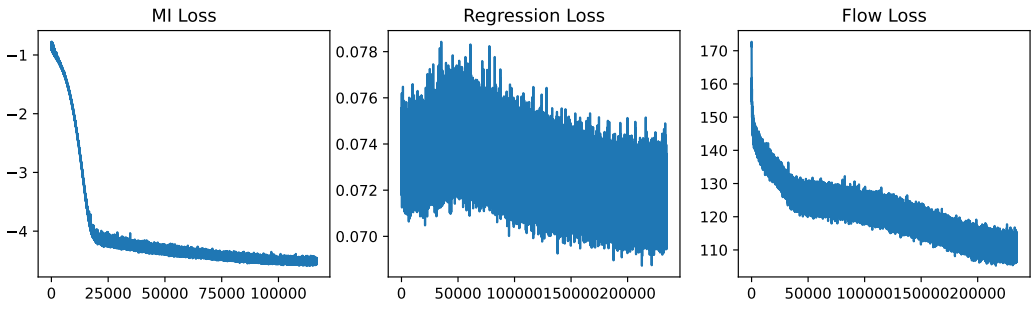


Figure 12: Loss function values of ASL in the training of encoder, regression function, and density estimation, respectively.