To Regularize or Not To Regularize? The Bias Variance Trade-off in Regularized AEs

Arnab Kumar Mondal
IIT Delhi
anz188380@cse.iitd.ac.in

Himanshu Asnani
TIFR Mumbai
himanshu.asnani@tifr.res.in

Parag Singla
IIT Delhi
parags@cse.iitd.ac.in

Prathosh AP
IIT Delhi
prathoshap@ee.iitd.ac.in

Abstract

Regularized Auto-Encoders (RAEs) form a rich class of neural generative models. They effectively model the joint-distribution between the data and the latent space using an Encoder-Decoder combination, with regularization imposed in terms of a prior over the latent space. Despite their advantages, such as stability in training, the performance of AE based models has not reached the superior standards as that of the other generative models such as Generative Adversarial Networks (GANs). Motivated by this, we examine the effect of the latent prior on the generation quality of deterministic AE models in this paper. Specifically, we consider the class of RAEs with deterministic Encoder-Decoder pairs, Wasserstein Auto-Encoders (WAE), and show that having a fixed prior distribution, *a priori*, oblivious to the dimensionality of the 'true' latent space, will lead to the infeasibility of the optimization problem considered. Further, we show that, in the finite data regime, despite knowing the correct latent dimensionality, there exists a bias-variance trade-off with any arbitrary prior imposition. As a remedy to both the issues mentioned above, we introduce an additional state space in the form of flexibly learnable latent priors, in the optimization objective of the WAEs. We implicitly learn the distribution of the latent prior jointly with the AE training, which not only makes the learning objective feasible but also facilitates operation on different points of the bias-variance curve. We show the efficacy of our model, called FlexAE, through several experiments on multiple datasets, and demonstrate that it is the new state-of-the-art for the AE based generative models.

Introduction

Regularized Auto-Encoder (AE) based latent variable models implicitly define a joint distribution over the input data and a lower-dimensional latent space, by approximating the true latent posterior, with a variational distribution. This variational distibution is parameterized using a neural network called the Encoder. The distribution induced by the Encoder is regularized to follow a pre-defined latent prior distribution. Subsequently, a Decoder network is trained to conditionally sample from the data distribution via optimizing a data-reconstruction metric. The parameters of the Encoder and the Decoder networks are learnt by optimizing either a bound on the data likelihood (Kingma and Welling 2013) or a divergence measure between the true and generated data distributions (Tolstikhin et al. 2018). The framework of AE-based generative models is attractive because of its ease and stability in training, efficiency in sampling, and flexibility in architectural choices. However, despite their advantages, AE-based models have failed to reach the performance of

other State-of-The-Art (SoTA) generative models (Dai and Wipf 2019; Mondal et al. 2020).

Several aspects such as the loss function used for optimization (Higgins et al. 2017; Larsen et al. 2016), presence of conflicting terms in the optimization objective (Hoffman and Johnson 2016; Kim and Mnih 2018), distributional choices (E.g., Gaussianity) imposed on the Encoder and Decoder (Zhao, Song, and Ermon 2019; Rezende and Viola 2018), dimensionality of the latent space used (Dai and Wipf 2019; Mondal et al. 2020), the mismatch between the learned and imposed prior (Shengjia Zhao and Ermon 2017; Tomczak and Welling 2018) have been identified as possible causes for the sub-optimal performance of the AE-based models. Many remedial measures, including the modification of the objective function (Zhao, Song, and Ermon 2019; Higgins et al. 2017; Kim and Mnih 2018), use of non-Gaussian Encoder/Decoder (Larsen et al. 2016; Nalisnick, Hertel, and Smyth 2016), masking of spurious latent dimensions (Mondal et al. 2020), incorporating a richer class of priors on the latent space (Tomczak and Welling 2018; Takahashi et al. 2019; Klushyn et al. 2019), have been proposed in the literature to address some of these issues. While these modifications have improved AE models' performance, they are still behind SoTA generative models (Dai and Wipf 2019; Mondal et al. 2020). In this work, we address one of these issues with the following contributions:

1. We theoretically establish that in a deterministic AE based generative model, choosing a latent prior distribution supported on the entire space, leads to infeasible optimization objective, when the 'true' latent space has dimensionality that is other than that of the model's latent space.

2. We argue that even with matched dimensionality, there exists a bias-variance trade off that arises from the choice of any assumed latent prior, whenever there is a finite data.

3. As a remedy, we propose a new model, which we call FlexAE, that can impose flexible learnable priors on RAEs that not only make the optimization problem feasible but also facilitate a better trade off between the bias and variance on-the-go, during AE-training.

4. We empirically demonstrate our claims through extensive experimentation on synthetic and real-world datasets by achieving significant improvement over the SoTA AE-based generative models.
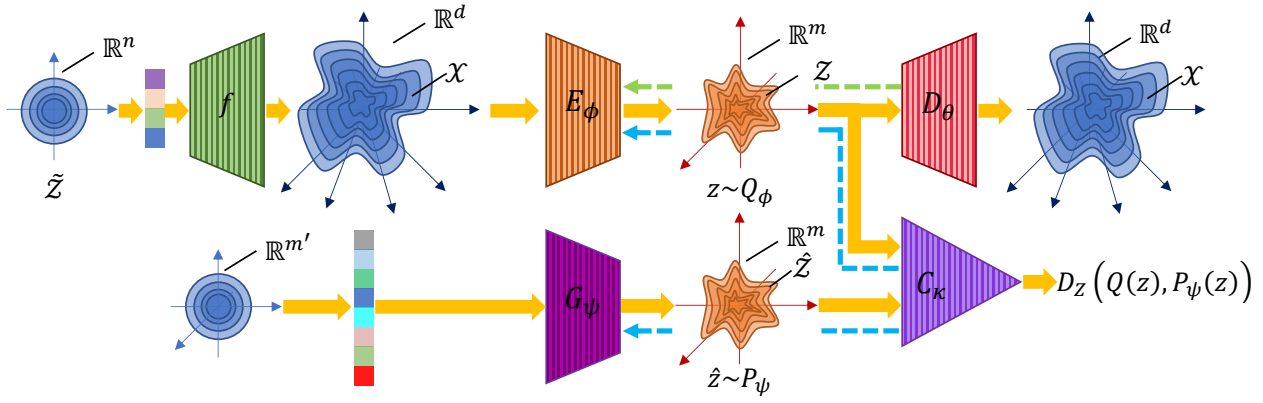
Figure 1: Our Model, FlexAE: Nature first samples an n-dimensional latent code from the true latent space, $\widetilde{\mathcal{Z}}$. Next, the latent code is mapped to an n-dimensional manifold, $\mathcal{X}$ in $\mathbb{R}^d$. The observed variables are encoded using a deterministic Encoder, E_ϕ. The m-dimensional encoded representations lie in an n-dimensional manifold $\mathcal{Z}$. The decoder network, D_θ, learns an inverse projection from the learnt latent space, $\mathcal{Z}$ to the dataspace, $\mathcal{X}$. The generator network, G_ψ parameterizes the learnable prior distribution, that maps an isotropic Gaussian distribution in $\mathbb{R}^{m'}$ to any arbitrary prior $P_\psi(\boldsymbol{z})$ in $\mathbb{R}^m$. The critic network, C_κ measures the distributional divergence between Q_ϕ and P_ψ. G_ψ and C_κ are jointly trained along with the Auto-Encoder.

Background and Related Work

The general theme in all RAEs is to implicitly learn the joint distribution between the observed data and a latent variable, via optimizing an objective function which consists of an auto-encoding (conditional likelihood) and latent regularization term (divergence measure). Variational Autoencoder (VAE) (Kingma and Welling 2013) is the pioneering member of this family, in which the variational latent posterior and conditional data likelihood are respectively parameterized by probabilistic (Gaussian) Encoder and Decoder networks, while the latent prior is assumed to be an isotropic Gaussian distribution. A related class of AE-models are the Adversarial Auto-Encoders (AAEs) (Makhzani et al. 2016) and Wasserstein Auto-Encoders (WAEs) (Tolstikhin et al. 2018) where a pair of deterministic Encoder-Decoder is used with Jenson-Shannon and Wasserstein distance respectively, between the aggregated latent posterior and the latent prior.

Even though VAEs/WAEs provide solid frameworks for AE-based generative models, several drawbacks are associated with it, which prevents them to compete with the other SoTA generative models. It is shown that there exists a conflict between the two terms of the objective, in the case of VAEs (Higgins et al. 2017; Shengjia Zhao and Ermon 2017; Rezende and Viola 2018). A few remedial measures such as introduction of a tunable parameter in the objective (Burgess et al. 2017), use of additional penalties such as mutual information (Zhao, Song, and Ermon 2019), total correlation (Kim and Mnih 2018), and generalised optimization objective (Rezende and Viola 2018) have been proposed. Another often discussed issue with AE-models with stochastic Encoder-Decoders is that they adopt a simple unimodal Gaussian distribution for parameterization (Rosca, Lakshminarayanan, and Mohamed 2018). To address this, (Nalisnick and Smyth 2017) implements a Bayesian nonparametric version of the variational autoencoder that has a latent representation with stochastic dimensionality and could represent richer class of distributions. Invertible flow-based generative models (Kingma et al. 2016; Rezende and Mohamed 2015) capitalize on the idea of normalizing flow for the Encoder and Decoder networks. VAE/GAN (Larsen et al. 2016), VGH/VGH++ (Rosca, Lakshminarayanan, and Mohamed 2018) incorporates an adversarial learning at the Decoder so that it can represent a rich class of distributions.

Further, it is observed that there is a mismatch between the aggregated variational posterior and the latent prior, leading to sub-optimality of the divergence term in the objective and in turn poor generation (Tomczak and Welling 2018; Dai and Wipf 2019). Several methods try to alleviate this problem, broadly in two ways (i) using a richer class of parametric priors on the latent space (Tomczak and Welling 2018; Klushyn et al. 2019; Kumar, Poole, and Murphy 2020) and (ii) using a post-hoc technique to minimize the divergence or sample from the latent space without regularizing it (Bauer and Mnih 2019; Ghosh et al. 2020; Takahashi et al. 2019). Among the first category of methods, VampPrior (Tomczak and Welling 2018) assumes the prior to be a mixture of the conditional posteriors with a set of learnable pseudo-inputs. Authors in (Klushyn et al. 2019) adapt the constrained optimization setting in (Rezende and Viola 2018) and substitute the standard normal prior with a hierarchical prior and use an importance-weighted bound as the optimization objective. In (Huang et al. 2017; Kumar, Poole, and Murphy 2020; Kingma et al. 2016), latent priors are learned using normalizing flow based methods. Within the second category of methods, (Bauer and Mnih 2019) learns to sample from a rich class of priors by multiplying a simplistic prior distribution with a learned acceptance function. In (Takahashi et al. 2019), kernel density trick is used for matching the prior to the aggregated posterior. RAE-GMM (Ghosh et al. 2020), imposes an L2-norm penalty in the latent space and learns to sample from it using a Gaussian Mixture Model (GMM) on the latent space. While these methods report improve-

ment over the SoTA metrics, not many give a theoretical justification for using richer-class of latent priors. Further, post-hoc latent samplers such as RAE-GMM (Ghosh et al. 2020) do not have control over the amount of bias imposed (modulo a simple objective scaling factor), that might lead to over/under fitting as shown later.

However, it has been both theoretically and empirically observed that dimensionality of the latent space used has a critical impact on the performance of these models (Mondal et al. 2020; Dai and Wipf 2019; Rubenstein, Schoelkopf, and Tolstikhin 2018). Authors in (Dai and Wipf 2019) study the implication of the mismatch between the dimensionality of the data and the true latent space and the role of Decoder variance, in the case of AEs with stochastic Encoders. They argue a learnable variance in the Decoder would make the objective reach negative infinity even when the aggregated posterior would not match the standard Gaussian prior not because of simplistic modelling assumption but because of mismatch between data dimensionality and the true latent dimensionality. To resolve this issue they introduce a second-stage VAE, which is used on the latent space of the first stage (which is a usual VAE), where the data and the latent dimensions match. In MaskAAE (Mondal et al. 2020), the authors noted that the generation quality degrades when there is a mismatch between the dimensionality of the true and the assumed latent space of a deterministic AE. They develop a procedure to explicitly zero-out (mask) the spurious latent dimensions via a learnable masking layer. In this backdrop, however, ours is the first study to theoretically demonstrate the in-feasibility of the objective of a deterministic Generative AE model such as WAE (Tolstikhin et al. 2018), under a fixed prior in relation with the mismatched latent dimensionality.

Proposed Method

Optimality of the Latent Space of WAEs

We start by assuming that the true data is generated in nature via a two-step process. First, the true latent variables are sampled from an n-dimensional space, $\widetilde{\mathcal{Z}}$ according to some continuous distribution in $\mathbb{R}^n$. Next, a non-linear function, $f : \widetilde{\mathcal{Z}} \to \mathcal{X}$ maps the true latent space, $\widetilde{\mathcal{Z}}$ to the observed data space, $\mathcal{X} \subseteq \mathbb{R}^d$, with $d >> n$, in most practical cases. In other words, observed data $\boldsymbol{x}$ lies on $\mathcal{X}$, an n-dimensional manifold embedded in $\mathbb{R}^d$. We make a benign assumption on f that it can be represented using neural networks with sigmoidal (or hyperbolic tangent, ReLU, Leaky ReLU etc.) activations to arbitrary closeness. Under this model, the data could be seen as lying in an n-dimensional manifold within $\mathbb{R}^d$, with an underlying ground truth distribution $P_d(\boldsymbol{x})$. The objective of the WAE model is to estimate (or learn to sample from) the distribution $P_d(\boldsymbol{x})$, given some i.i.d. samples drawn from it. The distribution learned by an RAE, denoted by $P_\theta(\boldsymbol{x})$ is given by $P_\theta(\boldsymbol{x}) = \int_{\mathcal{Z}} P_\theta(\boldsymbol{x}|\boldsymbol{z})dP_z$, where $P_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the distribution parameterized by a deterministic Decoder neural network $D_\theta(\boldsymbol{z})$ and $P_Z(\boldsymbol{z})$ is the latent prior defined on an m-dimensional space, $\mathcal{Z}$. The distribution $P_\theta(\boldsymbol{x})$ is estimated by minimizing the Wasserstein distance between $P_\theta(\boldsymbol{x})$ and $P_d(\boldsymbol{x})$ which is obtained by

solving the following optimization problem (Tolstikhin et al. 2018):

$$\inf_{\phi,\theta} \left(\mathbb{E}_{P_d} \mathbb{E}_{Q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[c\big(\boldsymbol{x}, D_\theta(\boldsymbol{z})\big) \right] \right) \tag{1}$$
$$\text{such that } Q_\phi(\boldsymbol{z}) = P_Z(\boldsymbol{z})$$

Here $Q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is the variational conditional posterior, which is also parameterzied by a deterministic neural network called the Encoder, $E_\phi : \mathcal{X} \to \mathcal{Z}$.
$Q_\phi(z) = \int_{\mathbb{R}^d} Q_\phi(\boldsymbol{z}|\boldsymbol{x}) \, dP_d(\boldsymbol{x})$ is the aggregated posterior distribution imposed by the Encoder, $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is any measurable cost function (such as Mean Square Error, MSE) and $\phi \in \Phi$, $\theta \in \Theta$ are the learnable parameters of Encoder and Decoder, respectively. The constrained optimization problem in Eq. 1 translates to auto-encoding the input data with a constraint (regularizer) that the aggregated distribution imposed by the Encoder matches with a predefined latent prior distribution. It can be equivalently written as an unconstrained problem by introducing a Lagrangian:

$$D_{WAE}(P_d, P_{\theta^*}) = \inf_{\phi,\theta} \left(\underbrace{\mathbb{E}_{P_d} \mathbb{E}_{Q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[c\big(\boldsymbol{x}, D_\theta(\boldsymbol{z})\big) \right]}_{\text{a}} + \right.$$
$$\left. \lambda \cdot \underbrace{D_Z\big(Q_\phi(\boldsymbol{z}), P_Z(\boldsymbol{z})\big)}_{\text{b}} \right) \tag{2}$$

Where λ is the Lagrange multiplier[1], $D_Z(.)$ is any divergence measure such as Kullaback-Leibler, Jenson-Shannon or Wasserstein distance, between two distributions and θ^* represents the optimum decoder parameters. Note that objective in Eq. 1 becomes feasible only when $D_Z\big(Q_\phi(\boldsymbol{z}), P_Z(\boldsymbol{z})\big)$ becomes zero. Equipped with these, in Theorem 1, we show that when $m > n$ (most common practical case), the WAE objective (Eq. 1) does not have a feasible solution when the prior is fixed a priori to be any distribution which is supported outside of a set of countable union of all possible n-dimensional manifolds in an m-dimensional space, denoted by $\mathcal{Q}_m^n$. An example for such a prior is the an isotropic Gaussian distribution in $\mathbb{R}^m$, which is the usual choice in most models.

Theorem 1. *If $m > n$, then the regularization term in the objective function of a WAE/AAE (Eq. 2), $D_Z(Q_\phi(\boldsymbol{z}), P_Z(\boldsymbol{z})) > 0$, $\forall \phi$ and for any distributional divergence D_Z when the support of $P_Z(\boldsymbol{z}) \notin \mathcal{Q}_m^n$.*

Proof. Since $f : \widetilde{\mathcal{Z}} \to \mathcal{X}$ can be approximated arbitrarily closely using a neural network (the assumption we have made earlier) and the Encoder function $E_\phi : \mathcal{X} \to \mathcal{Z}$ is also a neural network, $E_\phi \circ f : \mathbb{R}^n \to \mathbb{R}^m$ belongs to the class of composition of affine transformations and point wise non-linearities (such as rectifiers, leaky rectifiers, or

---

[1]Theoretically, the objective should be optimized w.r.t. the Lagrange multiplier λ. However, in practical implementations (Tolstikhin et al. 2018) it is considered to be a hyper-parameter.

smooth strictly increasing functions like sigmoid, tanh, softplus, etc.). Consequently, $\mathcal{Z}$ is a always a countable union of n-dimensional manifolds in a m-dimensional ambient space (Lemma 1 in (Arjovsky and Bottou 2017)). Therefore, given that the Encoder is deterministic, by definition, $Q_\phi(\boldsymbol{z})$ has measure zero on $\mathbb{R}^m \backslash \mathcal{Z}$, whereas the support of $P_Z(\boldsymbol{z}) \notin \mathcal{Q}_m^n$ which implies that it has a non-zero measure outside $\mathcal{Z}$. Thus, any distributional divergence measure between $Q_\phi(\boldsymbol{z})$ and $P_Z(\boldsymbol{z})$ will assume a non-zero value, whenever $m > n$. $\square$

The above theorem asserts that it is impossible to match the aggregated latent posterior to the prior when the assumed latent dimension is more than the true latent dimension and the assumed prior has full-support, which consequently leads to bad generation quality. One possible solution for this problem is to make $m = n$ which is practically impossible because n is unknown. Another way of countering this issue is to use a stochastic Encoder and fill the 'extra' dimensions with external noise, however, it leads to other issues such as difficulty in Decoder training (Rezende and Viola 2018), conflict between the two terms in the objective (Burgess et al. 2017) and non-uniqueness of solutions (Dai and Wipf 2019). Hence, we restrict the scope of this paper to the case of deterministic Encoder-Decoder pair and reserve the case of stochastic Encoders for our future work.

The Bias-Variance Trade-off

One way to visualize Eq. 2 is that there is an reconstruction objective (term a) and a regularizer in the form of a divergence metric (term b). Under this view, the issue mentioned in Theorem 1 could be seen to be originating because of the choice of a 'wrong' regularizer. Noting this, a few recent work (Ghosh et al. 2020; Kumar, Poole, and Murphy 2020) have suggested to completely get rid of the latent regularizer but use an ex-post latent code sampler such as GMM, Markov-Chain Monte-Carlo (MCMC) or a GAN on the latent space after a free-form AE (only term a of Eq. 1) is trained. While this technique will theoretically avoid the problem mentioned, we argue that it imposes another practical issue when there is finite data.

It is well-observed that with sufficient model capacity, a deterministic AE when trained on a finite amount of data will lead to an increased variance (over-fitting). This is because the Encoder can settle with a trivial solution for $q_{\phi^*}(\boldsymbol{z}|\boldsymbol{x}_i)$ which is Dirac-deltas at all input data points $\boldsymbol{x}_i$ (Rezende and Viola 2018). Subsequently, the post-hoc sampler (e.g. GAN) will learn to sample from finite set of Dirac-deltas (Sanjeev Arora and Zhang 2018) resulting in a non-smooth latent space and poor generalization. On the other hand, as seen in the previous section, a high bias or over-regularization will also impact the generation quality. This is the infamous bias-variance trade-off that warrants a flexible prior which could facilitate the operation of an AE-model at different points of the bias-variance points. Note that this problem may arise with models with stochastic Encoders too. In the subsequent sections, we propose a model that can effectively handle both the issues.

Flexibly Learning Prior: FlexAE

Based on the discussion so far, fixing a prior makes the optimization objective infeasible and no prior leads to overfitting. To alleviate these, we propose to flexibly learn the latent prior jointly with the AE-training by introducing an additional state-space in the objective of an WAE as follows:

$$D_{FlexAE}(P_X, P_{\theta^*}) = \inf_{\psi, \phi, \theta} \left(\underbrace{\underset{P(\boldsymbol{x})}{\mathbb{E}} \underset{Q(\boldsymbol{z}|\boldsymbol{x})}{\mathbb{E}} \left[c(\boldsymbol{x}, D_\theta(\boldsymbol{z})) \right]}_{a} + \right.$$

$$\left. \underbrace{\lambda \cdot D_Z(Q_\phi(\boldsymbol{z}) || P_\psi(\boldsymbol{z}))}_{b} \right) \quad (3)$$

where $P_\psi(\boldsymbol{z})$ is a learnable latent prior parmaterized using a neural network called the Prior-Generator (P-GEN), G_ψ, that takes an $m' \geq n$ dimensional isotropic Gaussian distribution as the input and generates sample from an m-dimensional $P_\psi(\boldsymbol{z})$ (refer Fig. 1). θ^* denotes the optimal decoder parameters. In our model, referred to as the Flexible AE or FlexAE, P-GEN is jointly trained with the AE to alternatively minimize the divergence measure and the reconstruction terms in Eq. 3. Upon convergence, the output of the P-GEN forms the prior that is imposed on the latent space. In the following we show that not only that FlexAE doesn't suffer from the infeasibility problem but also helps in more flexible bias-variance trade-offs. First, it is to be noted that that $D_{FlexAE} \leq D_{WAE}$ and thus the new formulation does not harm the optimization. Next, the Theorem below states that the divergence measure can be brought to zero with FlexAE.

Theorem 2. $\forall m' \geq n$, $D_Z(Q_\phi(\boldsymbol{z}) || P_\psi(\boldsymbol{z}))$ *(term (b) in FlexAE objective (Eq. 3) becomes zero for optimum set of parameters.*

Proof. Let, $\mathcal{P}_\psi$ denote the set of all possible manifolds on which the output of P-GEN network, G_ψ, may lie within $\mathbb{R}^m$. Given sufficiently large deep nets, sample size, and computation time, $\mathcal{P}_\psi = \cup_{\eta \leq m'} \mathcal{Q}_m^\eta$. As $m' \geq n$, this implies $\mathcal{Q}_m^n \subseteq \mathcal{P}_\psi$ which implies that G_ψ can learn $P_\psi(\boldsymbol{z})$ to match $Q_\phi(\boldsymbol{z})$ driving $D_Z(Q_\phi(\boldsymbol{z}) || P_\psi(\boldsymbol{z}))$ to zero. $\square$

For implementation, we use MSE for c in term (a) of Eq. 3. D_Z, in principle can be chosen to be any distributional divergence such as Kullback-Leibler divergence (KLD), JensenShannon divergence (JSD), Wasserstein Distance and so on. In this work, we propose to use Wasserstein distance and utilize the principle laid in (Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017), to optimize the divergence term (b) in Equation 3. The loss functions used for different blocks of FlexAE are as follows:

1. Likelihood Loss - Realization of Term a in Eq. 3:

$$L_{AE} = \frac{1}{s} \sum_{i=1}^{s} ||\boldsymbol{x}^{(i)} - D_\theta(E_\phi(\boldsymbol{x}^{(i)}))||^2 \quad (4)$$
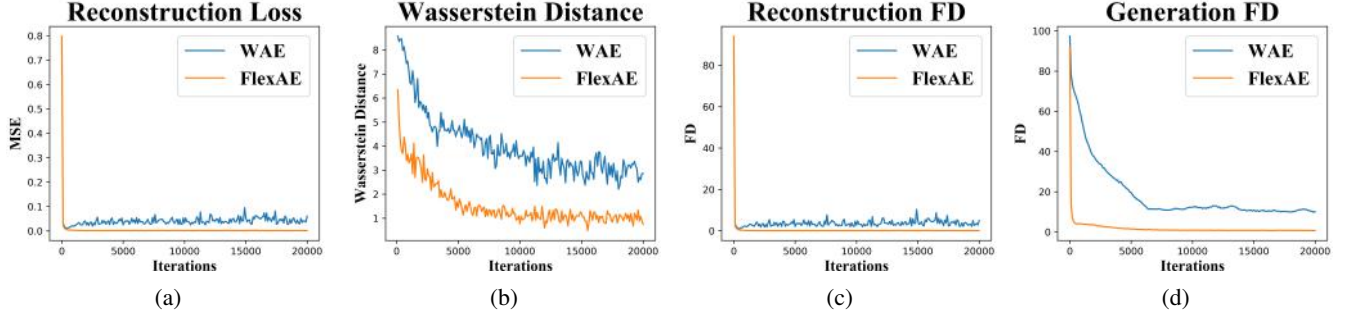
Figure 2: Comparison of RAEs with fixed and learnable latent priors on a synthetic dataset. It is seen that the Wasserstein distance between P_z and Q_z reduce faster in the case of FlexAE compared to a fixed prior WAE, leading to a better FD.
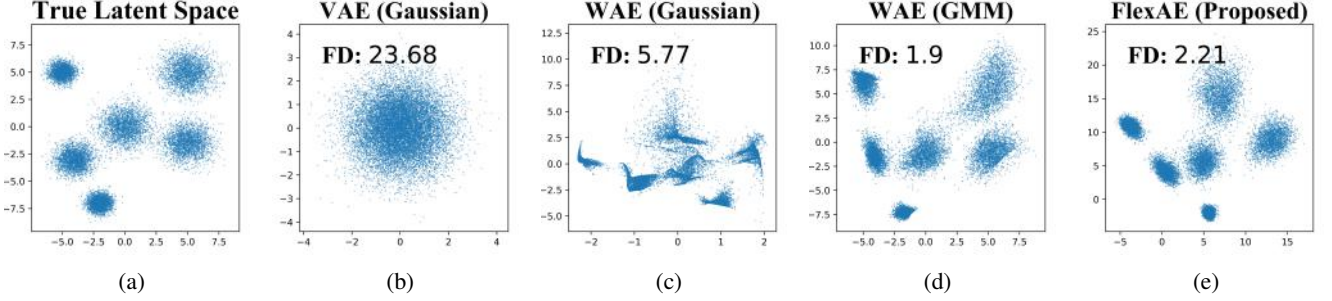

Figure 3: Visualization of (a) true latent space; (b) latent space learned by the VAE (Kingma and Welling 2013); (c) latent space learned by the WAE (Tolstikhin et al. 2018) with Normal prior; (d) latent space learned by the WAE (Tolstikhin et al. 2018) with GMM prior; and (e): latent space learned by the proposed FlexAE model, along with generation Frchet Distance (FD) in each case. For multimodal data, model with multimodal prior (WAE-GMM) and FlexAE perform better.

2. Wasserstein Loss - We use Wasserstein distance (Arjovsky, Chintala, and Bottou 2017) for D_Z (Term b Eq. 3):

$$L_{Critic} = \frac{1}{s}\sum_{i=1}^{s} C_\kappa(\hat{\boldsymbol{z}}^{(i)}) - \frac{1}{s}\sum_{i=1}^{s} C_\kappa(\boldsymbol{z}^{(i)})$$
$$+ \frac{\beta}{s}\sum_{i=1}^{s}\left(||\nabla_{\boldsymbol{z}_{avg}}^{(i)} C_\kappa(\boldsymbol{z}_{avg}^{(i)})||-1\right)^2 \quad (5)$$

$$L_{Gen} = -\frac{1}{s}\sum_{i=1}^{s} C_\kappa(\hat{\boldsymbol{z}}^{(i)}) \quad (6)$$

$$L_{Enc} = \frac{1}{s}\sum_{i=1}^{s} C_\kappa(\boldsymbol{z}^{(i)}) \quad (7)$$

Where, $\boldsymbol{z}^{(i)} = E_\phi(\boldsymbol{x}^{(i)})$, $\hat{\boldsymbol{z}}^{(i)} = G_\psi(\boldsymbol{n}^{(i)})$ and $\boldsymbol{n}^{(i)} \sim \mathcal{N}(0, I)$. $\boldsymbol{z}_{avg}^{(i)} = \alpha \boldsymbol{z}^{(i)} + (1-\alpha)\hat{\boldsymbol{z}}^{(i)}$, α, β are hyper parameters, with $\alpha \sim \mathcal{U}[0,1]$, and β as in (Gulrajani et al. 2017). E_ϕ, D_θ, G_ψ, and C_κ denote the encoder, decoder, latent generator and critic respectively. Also with the cost c chosen as MSE (Eq. 4), the LHS of the objective (Eq. 3) becomes 2-Wasserstein distance.

Figure 2 demonstrates the benefit of FlexAE over a WAE, where the performance of both the models is shown on a synthetic data: $\widetilde{\mathcal{Z}} = \mathbb{R}^5$ and $f: \mathbb{R}^5 \to \mathbb{R}^{128}$ is an arbitrary

multi-layer perceptron (details in the Tech. Appendix). It is seen that, when $m = 50$, Wasserstein distance between P_z and Q_z reduce faster and reaches much lower values in the case of FlexAE compared to a fixed prior WAE, leading to a better Fréshet Distance on the generated data.

Further, the P-GEN network allows FlexAE to better trade-off between over-fitting and under-fitting: former is addressed by having a regularizer in the form of finite capacity P-GEN, and latter is avoided by having a learnable P-GEN with sufficient capacity to represent the desired distribution (see Figure 5). Figure 3e demonstrates this effect where it is seen that the latent space learned by a FlexAE and a WAE with a GMM prior, on a synthetic data (details in the Tech. Appendix) results in better generation as compared to the models with fixed uni-modal Gaussian priors (Note that this figure is to show that a flexible prior helps in learning but not to show impossibility). Finally, the data generation in FlexAE happens as follows - (i) sample from a primitive (Gaussian) distribution and pass it through the P-GEN to sample a point from the latent space $p_\psi(\boldsymbol{z})$, (ii) input the latent sample through the Decoder to generate a data sample. Algorithm for training FlexAE can be found in the Tech. Appendix.

Experiments and Results

We consider three real-world datasets: MNIST (Lecun 2010), CIFAR-10 (Krizhevsky 2009), and CelebA (Liu et al.

Table 1: Comparison of FID scores (Heusel et al. 2017) on real datasets. Lower is better.

	MNIST		CIFAR10		CELEBA	
	Rec.	Gen.	Rec.	Gen.	Rec.	Gen.
VAE (Kingma and Welling 2013)	65.10	57.04	176.5	169.1	62.36	72.48
β-VAE (Higgins et al. 2017)	7.91	24.31	43.86	83.59	30.06	50.66
VAE-VampPrior (Tomczak and Welling 2018)	11.01	49.75	107.33	161.02	49.71	64.26
VAE-IOP (Takahashi et al. 2019)	8.01	32.61	92.17	141.92	41.52	57.30
WAE-GAN (Tolstikhin et al. 2018)	8.06	13.30	42.39	72.90	29.34	39.58
AE + GMM (L2) (Ghosh et al. 2020)	8.69	12.14	41.45	70.97	30.16	43.89
RAE + GMM (L2) (Ghosh et al. 2020)	6.15	7.30	40.48	69.24	29.05	35.30
VAE + FLOW (Kingma et al. 2016)	8.62	20.17	43.87	73.28	36.31	42.39
InjFlow^{ln} (Kumar, Poole, and Murphy 2020)	7.40	35.96	40.11	78.78	27.93	47.70
InjFlow^{ln} + GMM (Kumar, Poole, and Murphy 2020)	7.40	9.93	40.11	68.26	27.93	40.23
2-S VAE (Dai and Wipf 2019)	6.38	7.41	47.03	86.15	29.38	37.85
MaskAAE (Mondal et al. 2020)	8.46	10.52	58.40	71.90	35.75	40.49
FlexAE (Proposed)	**4.33**	**4.69**	**39.91**	**62.66**	**21.17**	**25.96**

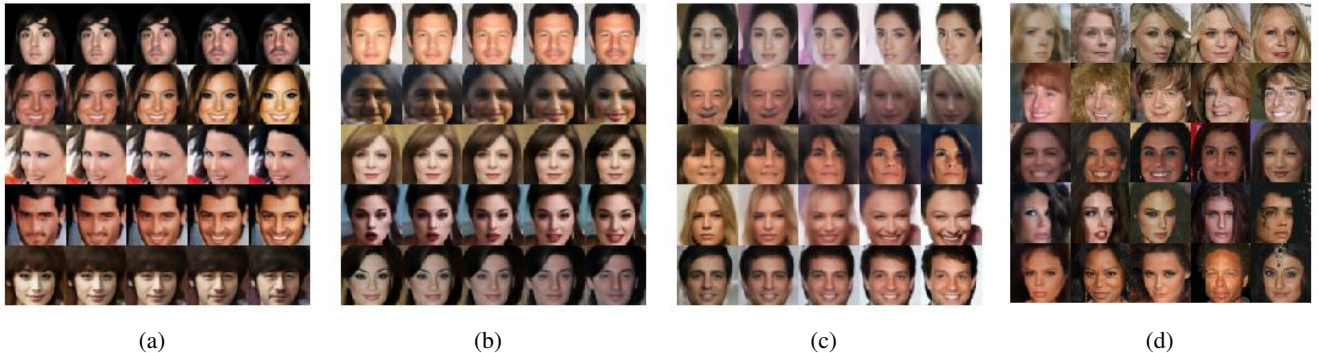(a)　　　　　　　(b)　　　　　　　(c)　　　　　　　(d)

Figure 4: Interpolations in the latent space of FlexAE on CelebA. Each row in (a) and (b) presents manipulation of a particular face attribute (Big Nose, Heavy Makeup, Black Hair, Smiling, Male). The central image of each row of (a) and (b) is a true image from the train and test split with and without the attribute respectively. Each row in (c) represents linear interpolation in the latent space between two randomly selected test samples in the first and the last entry. The first image in each row in (d) shows randomly generated samples using FlexAE and the next four entries are the four nearest neighbours from training data.

Table 2: Comparison of Precision/Recall scores (Sajjadi et al. 2018) on real datasets. Higher is better.

	MNIST	CIFAR10	CELEBA
VAE (Kingma and Welling 2013)	0.69/0.76	0.23/0.47	0.47/0.58
2S-VAE (Dai and Wipf 2019)	0.97/0.98	0.47/0.76	0.75/0.72
RAE + GMM (L2) (Ghosh et al. 2020)	0.98/0.98	0.61/0.87	0.74/0.75
MaskAAE (Mondal et al. 2020)	0.94/0.96	0.58/0.83	0.59/0.68
FlexAE (Proposed)	**0.99/0.99**	**0.68/0.85**	**0.89/0.88**

2015) for our four set of experiments.

Baseline Experiments

Methodology: The first task is to evaluate the FlexAE as a generative model. We use Fréchet Inception Distance, (FID) (Heusel et al. 2017), one of the most commonly used evaluation methods as it correlates well with human visual perception (Lucic et al. 2018). However, as observed in (Sajjadi et al. 2018), FID, being uni-dimensional, fails to distinguish between different cases of failure (poor sample quality and limited variation in the samples). Thus, we also report the precision and recall metrics described in (Sajjadi et al. 2018) along with FID, both of which are computed between the generated and the real test images. We compare FlexAE with

a number of SoTA AE-based generative models that cover a broad class namely, VAE (Kingma and Welling 2013), β-VAE (Higgins et al. 2017), VAE-VampPrior (Tomczak and Welling 2018), VAE-IOP (Takahashi et al. 2019), WAE (Tolstikhin et al. 2018), a plain with AE post-hoc GMM, RAE+GMM (Ghosh et al. 2020), VAE+Flow (Kingma et al. 2016), InjFlow (Kumar, Poole, and Murphy 2020), 2-stage VAE (Dai and Wipf 2019) and MaskAAE (Mondal et al. 2020), with same architectures (see Tech. Appendix).

Results: Table 1 compares the average reconstruction and generation FID scores (lower is better) of FlexAE over three executions (variance ± 0.59) with other AE-based generative models. It is seen that while models with parametric learnable priors (VampPrior, IOP, Flow) offer some improvement over the naive VAE, they are non optimum. It is also seen that complex prior models tend to over fit more (gap between the generation and reconstruction FIDs). Further, having the "right" dimensional latent space seems to have significant impact (2SVAE, MaskAAE). A relatively better performance of RAE+GMM, InjFlow shows that while absence of prior imposition will reduce the bias, it might lead to over fitting. Finally, FlexAE offers the best performance on

all three datasets as compared to other AE based generative models and its performance on MNIST and CelebA are comparable to that of the GANs. A similar trend is observed with the Precision/Recall numbers in Table 2 (We only use better SoTA models for comparison). It is seen that FlexAE offers significantly better numbers in terms of both Precision and Recall confirming its effectiveness in generating samples that are of both high quality and variety.

Effect of Latent Space Dimensionality

Methodology: To study how the latent space dimensionality affects the generation quality of the RAE, we train FlexAE and WAE models with varying m.

Results: As presented in Table 3, with increasing m, the reconstruction FID decreases for both WAE and FlexAE models. However, the generation FID of WAE models increases with m. While generation FID of FlexAE remains almost constant. This shows that FlexAE can achieve better optimum irrespective of the chosen model dimensionality.

Table 3: Variation of FID w.r.t. bottleneck layer dimension, m. For MNIST, $m_b = 20$ and for CELEBA $m_b = 64$.

m	MNIST				CELEBA			
	Rec.		Gen.		Rec.		Gen.	
	WAE	FlexAE	WAE	FlexAE	WAE	FlexAE	WAE	FlexAE
m_b	7.16	5.59	14.32	5.99	30.12	24.45	40.23	26.09
$2m_b$	5.17	3.22	23.11	4.22	29.34	21.17	39.58	25.96
$4m_b$	3.12	1.42	35.20	5.92	28.21	21.13	49.34	28.36

Bias-Variance Trade-off

Methodology: To evaluate our claims on the Bias-Variance trade-off, we repeat the generation experiments by varying the capacity of the prior generator (P-GEN) from very low capacity to very high capacity (details of models in the Tech. Appendix), on a small subset of training data (5000 samples). Sub-sampling is to ensure that effect of bias-variance is apparent. Models of huge capacity are needed to observe similar effects of the entire dataset.

Results: Figure 5 shows that there is a performance drop at either sides of moderate capacity models (Model 3 or Model 4). As the capacity of the P-GEN increases, the reconstruction FID decreases while generation FID increases, signalling over fitting. A reverse observation could be made about the high-bias low capacity models. This confirms our hypothesis of existence of a Bias-Variance curve. Please note, in Experiment 1 and 2, the architecture of the P-GEN was kept fixed across all datasets. Therefore, even though the mere architectural choice for the P-GEN imposes a bias, the flexibility (needed for trade off) is ensured in terms of the parameters of P-GEN.

Smoothness of the Latent Space

Methodology: To ascertain the smoothness of the learned latent space and that FlexAE doesn't over fit, we conduct a few qualitative experiments on the CelebA dataset: (i) Generation by transitions in the latent space along the direction of a particular attribute, (ii) transitions in the latent vectors between two generated samples and (iii) plot of the Nearest neighbour samples for a given generated image, from the training set, with interpolations done in the latent space.
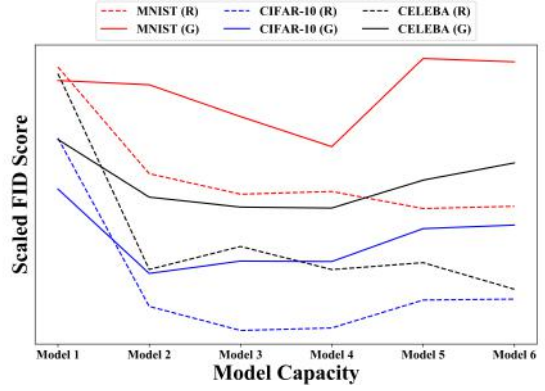

Figure 5: Variation of reconstruction and generation FID scores on limited training datasets with varying P-GEN capacity, demonstrating bias-variance trade-off. Models (1-6) are presented in increasing order of capacity.

Results: The outcome of these experiments are shown in Figure 4a, 4b, 4c and 4d. Each row in (a) and (b) presents manipulation of a particular face attribute (Big Nose, Heavy Makeup, Black Hair, Smiling, Male). The middle image in each row of (a) corresponds to a training sample with the attribute present and the middle image of a row in (b) represents a sample without the attribute. Each row in (c) represents linear interpolation in the latent space between two randomly selected test samples in the first and the last column. The interpolation results presented in (a), (b), and (c) clearly depicts the smoothness of the learnt latent space of FlexAE as it provides smooth transition between any two random images. The first image in each row in (d) shows a randomly generated sample using FlexAE and the next four entries are the four nearest neighbours from the training split. Visual dissimilarity between any generated image and its nearest neighbours from the training split confirms that FlexAE has not merely memorized the training set. (cf. Tech. Appendix for more qualitative results).

Conclusion

In this paper, we systematically studied the effect of the latent prior on the AE-based generative models. We demonstrated that fixing any kind of prior in a data-agnostic way is detrimental to the performance. We also showed that with finite data, there exists a bias-variance trade-off with imposition of any prior on the latent space. We proposed a model called the FlexAE that can potentially operate at different points of the bias-variance curve, and empirically demonstrated its efficacy.

Ethical Impact

Our work falls broadly in the area of techniques dealing with automatic data generation. We have touched both the theoretical as well as experimental aspects of this problem in our work. We believe our results/findings should be available for all scientific community for furthering research and development in this area, independent of their background (e.g., race, caste, creed, gender, nationality etc.). Datasets used in our experiments are pretty standard, and we do not think our work poses any specific ethical questions or creates potential biases against any particular groups.

To Regularize or Not To Regularize? The Bias Variance Trade-off in Regularized AEs

Technical Appendix

Details of Datasets

In this section, we describe the steps involved in synthetic dataset creation and provide relevant details (such as dimension, number of training and test examples and so on) of the synthetic datasets and the real datasets used in our work to experimentally validate our theoretical claims.

Synthetic Datasets

Synthetic data has been generated using a two step process. The steps involved in creating the dataset (corresponding to Figure 3 in the main paper) where the true latent space is GMM are listed below.

1. **Step 1:** Six two-dimensional Gaussian distributions are used to generate true latent space of the synthetic dataset. $z_{k1}^{(i)}$ and $z_{k2}^{(i)}$ denotes the 1st and the 2nd dimensions of the ith sample from the kth distribution respectively. The distributions are as mentioned below:

$$\begin{bmatrix} z_{11}^{(i)} \\ z_{12}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{21}^{(i)} \\ z_{22}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ 5 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{31}^{(i)} \\ z_{32}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} -5 \\ 5 \end{bmatrix}, \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{41}^{(i)} \\ z_{42}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ -1.5 \end{bmatrix}, \begin{bmatrix} 0.95 & 0 \\ 0 & 0.95 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{51}^{(i)} \\ z_{52}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} -2 \\ -7 \end{bmatrix}, \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{61}^{(i)} \\ z_{62}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} -4 \\ -3 \end{bmatrix}, \begin{bmatrix} 0.75 & 0 \\ 0 & 0.75 \end{bmatrix} \right)$$

2. **Step 2:** Next, a three layer MLP is used to map the two-dimensional points obtained from Step 1 to 128-dimensional data points. Each layer consists of 128 neurons and non-linearity used in each layer is $\tanh, \exp, \tanh$ respectively. Weight and bias parameters of each layer is drawn randomly from the following three distributions respectively: $\mathcal{N}(0, 0.05), \mathcal{N}(0, 0.2), \mathcal{N}(0, 0.1)$.

The dataset related to Figure 2 in the main paper is also generated using a 6 component GMM latent space. However, the true latent has dimension 5 and synthetic data has dimension 128 as before. The mean and variance of the Gaussian components of the true latent space are listed below.

$$\begin{bmatrix} z_{11}^{(i)} \\ z_{12}^{(i)} \\ z_{13}^{(i)} \\ z_{14}^{(i)} \\ z_{15}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{21}^{(i)} \\ z_{22}^{(i)} \\ z_{23}^{(i)} \\ z_{24}^{(i)} \\ z_{25}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{31}^{(i)} \\ z_{32}^{(i)} \\ z_{33}^{(i)} \\ z_{34}^{(i)} \\ z_{35}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} -5 \\ 5 \\ 3 \\ 4.5 \\ -6 \end{bmatrix}, 0.5 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{41}^{(i)} \\ z_{42}^{(i)} \\ z_{43}^{(i)} \\ z_{44}^{(i)} \\ z_{45}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 5 \\ -1.5 \\ -6.5 \\ 3 \\ 1 \end{bmatrix}, 0.95 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{51}^{(i)} \\ z_{52}^{(i)} \\ z_{53}^{(i)} \\ z_{54}^{(i)} \\ z_{55}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} -2 \\ -7 \\ 9 \\ -1.5 \\ -4.5 \end{bmatrix}, 0.5 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_{61}^{(i)} \\ z_{62}^{(i)} \\ z_{63}^{(i)} \\ z_{64}^{(i)} \\ z_{65}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} -4 \\ -3 \\ -5.5 \\ 2 \\ 4 \end{bmatrix}, 0.75 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right)$$

We have generated 15k training examples and 10k test examples for both of the synthetic datasets.

Real Datasets

The MNIST (Lecun 2010) database of gray scale handwritten digits consists of 60000 training examples and 10000 test samples. The CIFAR-10 (Krizhevsky 2009) dataset consists of 60000 tiny RGB images from 10 classes, with 6000 images per class. The standard split of this dataset consists of 50000 training images and 10000 test images. For experiments with MNIST and CIFAR-10, we use datasets as provided by Tensorflow API. CelebFaces Attributes Dataset (CelebA) (Liu et al. 2015) is a large-scale face attributes dataset with 202599 celebrity images, each with 40 attribute annotations. For experiments with CELEBA, we resize the images to 64×64 following many prior works (Kumar, Poole, and Murphy 2020; Mondal et al. 2020; Dai and Wipf 2019; Ghosh et al. 2020) in generative model. Table 4 summarizes the important information about the real datasets used in this paper. Although, the test split of CELEBA dataset contains more than 10k examples, we use 10k randomly selected samples for FID and precision/recall score computation for all the datasets.

Table 4: Details of Real Datasets

	Dimension ($h \times w \times c$)	Train Split Size	Test Split Size
MNIST (Lecun 2010)	$28 \times 28 \times 1$	60000	10000
CIFAR-10 (Krizhevsky 2009)	$32 \times 32 \times 3$	50000	10000
CELEBA (Liu et al. 2015)	$64 \times 64 \times 3$	162770	19962

Network Architectures

Like any other AE based generative model, FlexAE has a reconstruction pipeline consisting of an encoder (E_ϕ) and a decoder (D_θ) network. We have introduced a P-GEN network consisting of a generator network (G_ψ) and a critic network (C_κ) to facilitate sampling from the latent space of the reconstruction pipeline. The generation pipeline involves the latent generator, G_ψ and the image generator, D_θ, meaning generation is a two-step process. First, we sample from the latent space using the latent generator, G_ψ. Next, the image generator, D_θ samples from the image space using the generated latent code.

Next, we describe the architectures of each of the components of FlexAE used for the synthetic and the real experiments.

Synthetic Experiments

Table 5 presents architectures of different networks used in conducting the synthetic experiment. VAE (Kingma and Welling 2013) consists of only encoder and decoder. WAE (Tolstikhin et al. 2018) consists of encoder, decoder and critic. FlexAE involves all the networks.

Real Experiments

For real experiments, the encoder, E_ϕ and the decoder, D_θ architectures are adopted from prior work (Kumar, Poole, and Murphy 2020). The architecture of the encoder and the decoder networks

vary from one dataset to another as presented in Table 6. However, the architectures of the generator, G_ψ, the critic, C_κ and the regression network are fixed across all datasets as mentioned in Table 7. The capacity (no. of trainable parameters) of G_ψ and C_κ is fairly small as compared to the AE to ensure that the adversarial training does not overfit the latent space. However, if the capacity of G_ψ is too small then the bias in the latent space will increase, which will ultimately lead to a strong regularization. Therefore, we choose a moderate capacity generator and critic network. To study the effect to latent space dimensionality, m on the generation quality, we train different FlexAE models with varying m while everything else is kept fixed.

Table 8 lists the architectures of different capacity generators used in the bias-variance experiment. Please note that the number of parameters of the latent generator model increases with model number in Table 8. Thus, the capacity of the Model-1 is the least and the capacity of the Model-6 is the highest.

Training Algorithm, Hyper-parameters, Computing Resources and Average Runtime

As mentioned in the main paper, the auto-encoder is required to be optimized jointly with the P-GEN to ensure regularization in the AE latent space. This regularization effectively enforces smoothness in the learnt latent space and prevents the AE from overfitting on the training examples. In order to be able to satisfy the above requirement in practice, we optimize each of the four losses specified in the main paper in every training iteration. Specifically, in each learning loop, we optimize the L_{AE}, L_{Critic}, L_{Gen}, and L_{Enc} in that order using a learning schedule. We use Adam optimizer for our optimization. The training algorithm is described in Algorithm 1. For real experiments we have trained our models for 130000 iterations on each dataset with a batch size of 128. We have used a machine with Intel® Xeon® Gold 6142 CPU, 376GiB RAM, and Zotac GeForce® GTX 1080 Ti 11GB Graphic Card for all of our experiments. The average runtime for experiments on MNIST, CIFAR-10, and CELEBA is approximately 20 hours, 40 hours and 100 hours respectively.

Algorithm 1 Pseudo code for the training loop of FlexAE

Hyper-parameters: $\eta_{AE} = 0.001$, $\eta_{Critic} = 0.0001$, $\eta_{Gen} = 0.0005$, $\eta_{Enc} = 0.00001$, AE_OPT = Adam(lr $= \eta_{AE}, \beta_1 = 0.9, \beta_2 = 0.999$), CRITIC_OPT = Adam(lr $= \eta_{Critic}, \beta_1 = 0.0, \beta_2 = 0.9$), GEN_OPT = Adam(lr $= \eta_{Gen}, \beta_1 = 0.0, \beta_2 = 0.9$), ENC_OPT = Adam(lr $= \eta_{Enc}, \beta_1 = 0.0, \beta_2 = 0.9$), $disc\_training\_ratio = 5$.

1: **function** TRAIN
2:     **for** $i \leftarrow 1$ to $training\_steps$ **do**
3:         Minimize L_{AE} and Update ϕ, θ
4:         **for** $j \leftarrow 1$ to $disc\_training\_ratio$ **do**
5:             Minimize L_{Critic} and Update κ
6:         **end for**
7:         Minimize L_{Gen} and Update ψ
8:         Minimize L_{Enc} and Update ϕ
9:     **end for**
10: **end function**

Table 5: Network Architectures for Synthetic Experiment

Encoder	Decoder	Generator	Critic
$\boldsymbol{x} \in \mathbb{R}^{128}$	$\boldsymbol{z} \in \mathbb{R}^2$	$\boldsymbol{n} \in \mathbb{R}^2$	$\boldsymbol{z} \in \mathbb{R}^2$
$\to \mathrm{FC}_{128} \to \mathrm{ReLU}$	$\to \mathrm{FC}_{128} \to \mathrm{Tanh}$	$\to \mathrm{FC}_{128} \to \mathrm{ReLU}$	$\to \mathrm{FC}_{128} \to \mathrm{ReLU}$
$\to \mathrm{FC}_m$		$\to \mathrm{FC}_m$	$\to \mathrm{FC}_{128} \to \mathrm{ReLU}$
			$\to \mathrm{FC}_1$

$m = 50$ for the first synthetic experiment (Figure 2 in the main paper)
and $m = 2$ for the second synthetic experiment (Figure 3 in the main paper).

Table 6: Encoder and Decoder Architectures for Real Datasets

MNIST	CIFAR10	CELEBA
Encoder		
$\boldsymbol{x} \in \mathbb{R}^{28 \times 28}$	$\boldsymbol{x} \in \mathbb{R}^{32 \times 32 \times 3}$	$\boldsymbol{x} \in \mathbb{R}^{64 \times 64 \times 3}$
$\to \mathrm{Conv}_{64,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{128,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{128,5,1} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{Conv}_{128,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{256,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{256,5,2} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{Conv}_{256,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{512,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{512,5,2} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{Conv}_{512,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{1024,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{1024,5,2} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{Conv}_{512,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{1024,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{Conv}_{1024,5,2} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{Flatten} \to \mathrm{FC}_{32}$	$\to \mathrm{Flatten} \to \mathrm{FC}_{128}$	$\to \mathrm{Flatten} \to \mathrm{FC}_{128}$
Decoder		
$\boldsymbol{z} \in \mathbb{R}^{32}$	$\boldsymbol{z} \in \mathbb{R}^{128}$	$\boldsymbol{z} \in \mathbb{R}^{128}$
$\to \mathrm{FC}_{7 \times 7 \times 256} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{FC}_{8 \times 8 \times 512} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{FC}_{16 \times 16 \times 512} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{Reshape}_{7 \times 7 \times 256}$	$\to \mathrm{Reshape}_{8 \times 8 \times 512}$	$\to \mathrm{Reshape}_{16 \times 16 \times 512}$
$\to \mathrm{TCONV}_{512,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{1024,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{1024,5,1} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{TCONV}_{256,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{512,4,1} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{512,5,1} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{TCONV}_{128,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{256,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{256,5,2} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{TCONV}_{64,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{128,4,2} \to \mathrm{BN} \to \mathrm{ELU}$	$\to \mathrm{TCONV}_{128,5,2} \to \mathrm{BN} \to \mathrm{ELU}$
$\to \mathrm{CONV}_{1,4,1} \to \mathrm{Sigmoid}$	$\to \mathrm{CONV}_{3,4,1} \to \mathrm{Sigmoid}$	$\to \mathrm{CONV}_{3,5,1} \to \mathrm{Sigmoid}$

Table 7: Generator and Critic Architectures for Real Datasets

Generator	Critic
$\boldsymbol{n} \in \mathbb{R}^m$	$\boldsymbol{z} \in \mathbb{R}^m$
$\to \mathrm{FC}_{1024} \to \mathrm{ReLU}$	$\to \mathrm{FC}_{512} \to \mathrm{ReLU}$
$\to \mathrm{FC}_{512} \to \mathrm{ReLU}$	$\to \mathrm{FC}_{256} \to \mathrm{ReLU}$
$\to \mathrm{FC}_m$	$\to \mathrm{FC}_{128} \to \mathrm{ReLU}$
	$\to \mathrm{FC}_{128} \to \mathrm{ReLU}$
	$\to \mathrm{FC}_1$

$m = 32$ for MNIST and $m = 128$ for CIFAR10, CELEBA.

Table 8: Generator Architectures for Bias-Variance Experiment

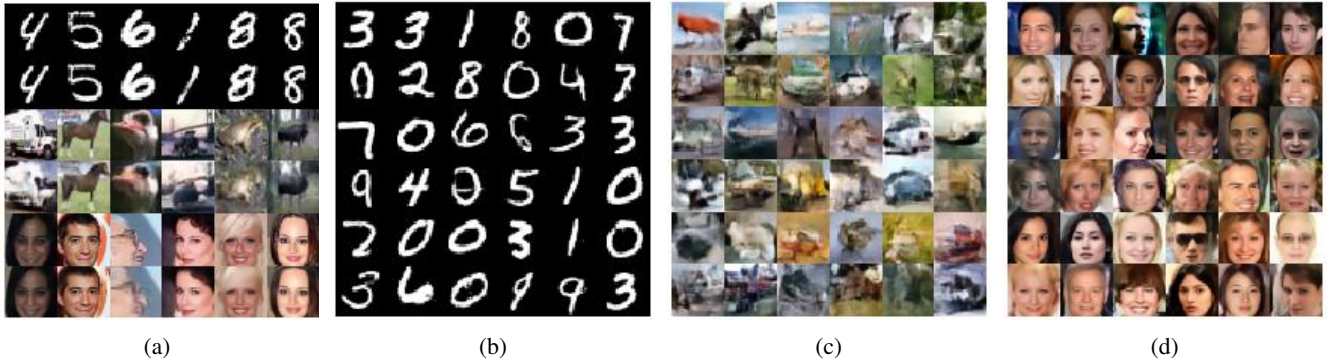| Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
$\boldsymbol{n} \in \mathbb{R}^m$	$\boldsymbol{n} \in \mathbb{R}^m$	$\boldsymbol{n} \in \mathbb{R}^m$	$\boldsymbol{n} \in \mathbb{R}^m$	$\boldsymbol{n} \in \mathbb{R}^m$	$\boldsymbol{n} \in \mathbb{R}^m$
$\rightarrow \text{FC}_{16} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{64} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{256} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{1024} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{1024} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{2048} \rightarrow \text{ReLU}$
$\rightarrow \text{FC}_{16} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{64} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{256} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{1024} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{1024} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{2048} \rightarrow \text{ReLU}$
$\rightarrow \text{FC}_m$	$\rightarrow \text{FC}_m$	$\rightarrow \text{FC}_m$	$\rightarrow \text{FC}_m$	$\rightarrow \text{FC}_{1024} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{2048} \rightarrow \text{ReLU}$
				$\rightarrow \text{FC}_{1024} \rightarrow \text{ReLU}$	$\rightarrow \text{FC}_{2048} \rightarrow \text{ReLU}$
				$\rightarrow \text{FC}_m$	$\rightarrow \text{FC}_m$
			$m = 32$ for MNIST and $m = 128$ for CIFAR10, CELEBA.		

(a) (b) (c) (d)

Figure 6: (a) Visualization of reconstruction quality of FlexAE model on randomly selected data from the test split of MNIST (first and second rows), CIFAR-10 (third and fourth rows) and CELEBA (fifth and sixth rows). The odd rows represent the real data and the even rows represent reconstructed data. Randomly generated samples from (b) MNIST, (c) CIFAR-10, and (d) CELEBA datasets using FlexAE model.

Experimental Results

In the main paper, the performance of FlexAE is evaluated mainly quantitatively, using standard metrics: FID (Heusel et al. 2017) and precision/recall (Sajjadi et al. 2018) score. We have used 10000 reconstructed and 10000 generated samples against 10000 test examples for computation of FID and precision/recall score for all datasets. It has been observed that FlexAE outperforms all other current state-of-the-art AE based generative models as measured using those metrics. In this section, we present more qualitative results (reconstruction on test examples, generated samples and resulting images due to interpolation in the latent space) for visual evaluation of the proposed generative framework, FlexAE.

Figure 6a represents reconstruction of 6 randomly chosen samples from test test split of MNIST (row 1 and 2), CIFAR-10 (row 3 and 4), and CELEBA (row 5 and 6) dataset. The odd rows represent true data and the even rows represents reconstructed data. Figure 6b, 6c, 6d presents 36 randomly generated samples of MNIST, CIFAR-10 and CELEBA datasets respectively.

Next, we present more attribute based interpolation results from the CELEBA test split in Figure 7, Figure 8, Figure 9, Figure 10, and Figure 11 for the attributes "Big Nose", "Heavy Makeup", "Black Hair", "Smiling", and "Male" respectively. The central image of the grid in the sub-figures (a) and (b) in every figure presents a negative test example from the CELEBA dataset i.e. a test sample without the corresponding attribute. Whereas, the central image in the grid of the sub-figures (c) and (d) presents a positive test example i.e. a test sample with the particular attribute. For latent space traversal along a particular attribute direction, we calculate the average representation, z_{pos} with respect to all the positive training samples and the average representation, z_{neg} with respect to all the negative training samples. Finally, we use the direction

$(z_{pos} - z_{neg})$ to traverse the latent space for attribute manipulation. Please note, this supervised traversal is performed post training in order to understand if the trained model could learn the meaning of different face attributes without supervision. The training was completely unsupervised without using any label information. As can be seen from the Figures 7 - 11, FlexAE could successfully learn the concept of different attributes without any kind of supervision. Otherwise, the interpolated figures would not be so smooth.

Finally, Figure 12 presents a 15×15 grid, where, the first column plots some randomly generated face images and the remaining entries in each row are the 14 nearest neighbours (in terms of Euclidean distance) from the training split. The generated images are visually significantly different as compared to the nearest training examples. This confirms that FlexAE has not memorised the training examples and generates unique, unseen images.

Figure 7: Interpolations in the latent space of FlexAE on CelebA. Each row in (a) and (b) presents manipulation of the attribute "Big Nose". The central image of each grid in (a), and (b) is a true image from the test split without the attribute. Whereas, the central image of each grid in (c) and (d) is a true image from the test split with the attribute.
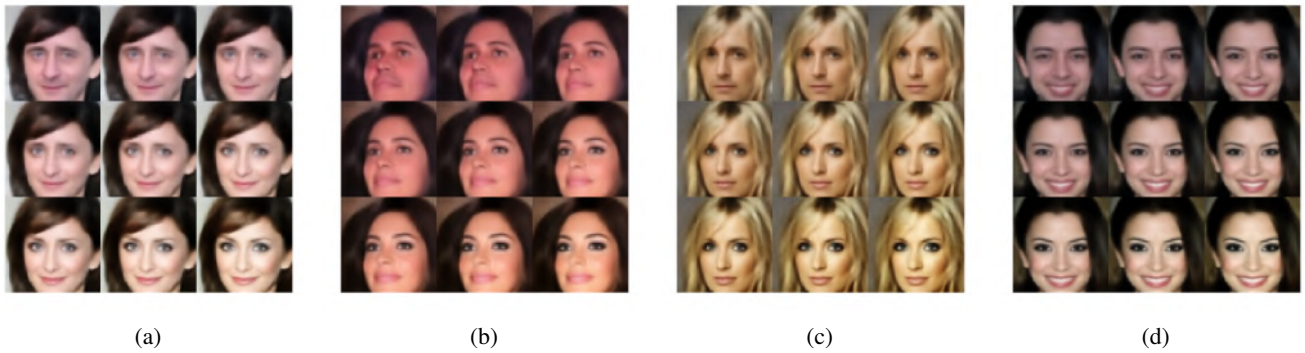

Figure 8: Interpolations in the latent space of FlexAE on CelebA. Each row in (a) and (b) presents manipulation of the attribute "Heavy Makeup". The central image of each grid in (a), and (b) is a true image from the test split without the attribute. Whereas, the central image of each grid in (c) and (d) is a true image from the test split with the attribute.
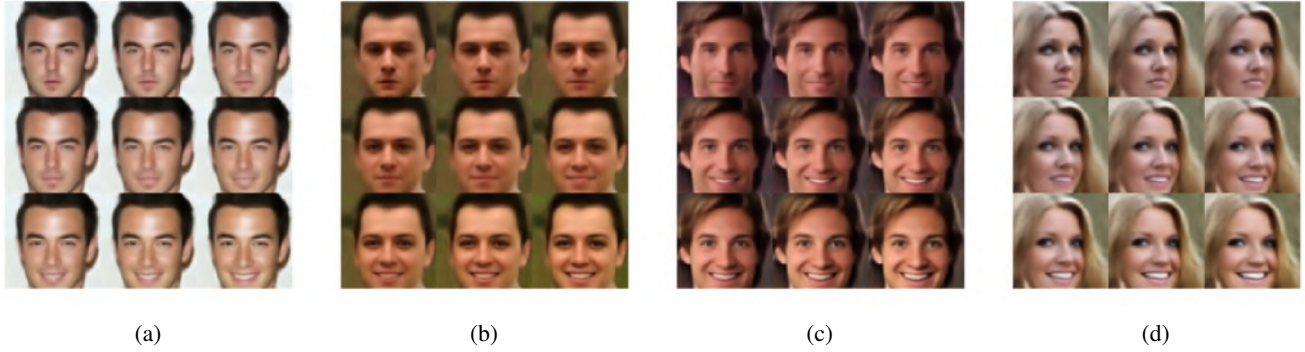

Figure 9: Interpolations in the latent space of FlexAE on CelebA. Each row in (a) and (b) presents manipulation of the attribute "Black Hair". The central image of each grid in (a), and (b) is a true image from the test split without the attribute. Whereas, the central image of each grid in (c) and (d) is a true image from the test split with the attribute.

Figure 10: Interpolations in the latent space of FlexAE on CelebA. Each row in (a) and (b) presents manipulation of the attribute "Smiling". The central image of each grid in (a), and (b) is a true image from the test split without the attribute. Whereas, the central image of each grid in (c) and (d) is a true image from the test split with the attribute.
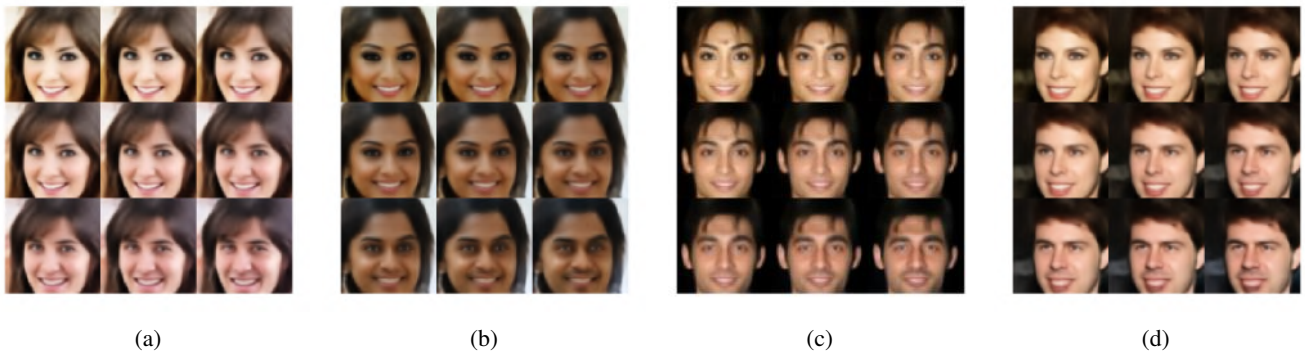

Figure 11: Interpolations in the latent space of FlexAE on CelebA. Each row in (a) and (b) presents manipulation of the attribute "Male". The central image of each grid in (a), and (b) is a true image from the test split without the attribute. Whereas, the central image of each grid in (c) and (d) is a true image from the test split with the attribute.

Figure 12: The first entry in each row represents a randomly generated face using FlexAE. The remaining entries in each row represents 14 nearest neighbours (in terms of Euclidean distance) from the train split of CELEBA dataset. It is seen that the generated images using FlexAE are very different as compared to the training examples. This confirms that the state of the art FID score and precision recall score obtained using FlexAE is not due to mere overfitting on the training split.

References

Arjovsky, M.; and Bottou, L. 2017. Towards principled methods for training generative adversarial networks. In *Proc. of ICLR*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *Proc. of ICML*, 214–223.

Bauer, M.; and Mnih, A. 2019. Resampled priors for variational autoencoders. In *Proc. of AISTATS*.

Burgess, C. P.; Higgins, I.; Pal, A.; Matthey, L.; Watters, N.; Desjardins, G.; and Lerchner, A. 2017. Understanding disentangling in β-VAE. In *NeuRIPS Workshop*.

Dai, B.; and Wipf, D. 2019. Diagnosing and enhancing vae models. In *Proc. of ICLR*.

Ghosh, P.; Sajjadi, M. S. M.; Vergari, A.; Black, M.; and Scholkopf, B. 2020. From Variational to Deterministic Autoencoders. In *Proc. of ICLR*.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved Training of Wasserstein GANs. In *Proc. of NeuRIPS*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Proc. of NeuRIPS*.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *Proc. of ICLR*.

Hoffman, M. D.; and Johnson, M. J. 2016. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *NeuRIPS Workshop*.

Huang, C.-W.; Touati, A.; Dinh, L.; Drozdzal, M.; Havaei, M.; Charlin, L.; and Courville, A. 2017. Learnable explicit density for continuous latent space and variational inference. *arXiv preprint arXiv:1710.02248* .

Kim, H.; and Mnih, A. 2018. Disentangling by Factorising. In *Proc. of ICML*.

Kingma, D. P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; and Welling, M. 2016. Improved variational inference with inverse autoregressive flow. In *Proc. of NeuRIPS*.

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

Klushyn, A.; Chen, N.; Kurle, R.; Cseke, B.; and van der Smagt, P. 2019. Learning Hierarchical Priors in VAEs. In *Proc. of NeuRIPS*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report.

Kumar, A.; Poole, B.; and Murphy, K. 2020. Regularized Autoencoders via Relaxed Injective Probability Flow. *arXiv preprint arXiv:2002.08927* .

Larsen, A. B. L.; Sønderby, S. K.; Larochelle, H.; and Winther, O. 2016. Autoencoding beyond pixels using a learned similarity metric. In *Proc. of ICML*.

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

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2015. Deep Learning Face Attributes in the Wild. In *Proc. of ICCV*.

Lucic, M.; Kurach, K.; Michalski, M.; Bousquet, O.; and Gelly, S. 2018. Are GANs Created Equal? A Large-scale Study. In *Proc. of NeuRIPS*.

Makhzani, A.; Shlens, J.; Jaitly, N.; and Goodfellow, I. 2016. Adversarial Autoencoders. In *Proc. of ICLR*.

Mondal, A. K.; Chowdhury, S. P.; Jayendran, A.; Singla, P.; Asnani, H.; and Prathosh, A. 2020. MaskAAE: Latent space optimization for Adversarial Auto-Encoders. In *Proc. of UAI*.

Nalisnick, E.; Hertel, L.; and Smyth, P. 2016. Approximate inference for deep latent gaussian mixtures. In *NeurIPS Workshop*.

Nalisnick, E.; and Smyth, P. 2017. Stick-Breaking Variational Autoencoders. In *Proc. of ICLR*.

Rezende, D. J.; and Mohamed, S. 2015. Variational inference with normalizing flows. In *Proc. of ICML*.

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

Rosca, M.; Lakshminarayanan, B.; and Mohamed, S. 2018. Distribution matching in variational inference. *arXiv preprint arXiv:1802.06847* .

Rubenstein, P. K.; Schoelkopf, B.; and Tolstikhin, I. 2018. Wasserstein auto-encoders: latent dimensionality and random encoders. In *ICLR Workshop*.

Sajjadi, M. S. M.; Bachem, O.; Lucic, M.; Bousquet, O.; and Gelly, S. 2018. Assessing Generative Models via Precision and Recall. In *Proc. of NeuRIPS*.

Sanjeev Arora, A. R.; and Zhang, Y. 2018. Do gans actually learn the distribution? an empirical study. In *Proc. of ICLR*.

Shengjia Zhao, J. S.; and Ermon, S. 2017. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658* .

Takahashi, H.; Iwata, T.; Yamanaka, Y.; Yamada, M.; and Yagi, S. 2019. Variational autoencoder with implicit optimal priors. In *Proc. of AAAI*.

Tolstikhin, I.; Bousquet, O.; Gelly, S.; and Scholkopf, B. 2018. Wasserstein Auto-Encoders. In *Proc. of ICLR*.

Tomczak, J. M.; and Welling, M. 2018. VAE with a VampPrior. In *Proc. of AISTATS*.

Zhao, S.; Song, J.; and Ermon, S. 2019. InfoVAE: Balancing Learning and Inference in Variational Autoencoders. In *Proc. of AAAI*.