MaskAAE: Latent space optimization for Adversarial Auto-Encoders

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

Sankalan Pal Chowdhury[*]
IIT Delhi
cs1160701@iitd.ac.in

Aravind Jayendran[*†]
Flipkart Internet Pvt. Ltd.
aravind.j@flipkart.com

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

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

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

Abstract

The field of neural generative models is dominated by the highly successful Generative Adversarial Networks (GANs) despite their challenges, such as training instability and mode collapse. Auto-Encoders (AE) with regularized latent space provide an alternative framework for generative models, albeit their performance levels have not reached that of GANs. In this work, we hypothesise that the dimensionality of the AE model's latent space has a critical effect on the quality of generated data. Under the assumption that nature generates data by sampling from a "true" generative latent space followed by a deterministic function, we show that the optimal performance is obtained when the dimensionality of the latent space of the AE-model matches with that of the "true" generative latent space. Further, we propose an algorithm called the Mask Adversarial Auto-Encoder (MaskAAE), in which the dimensionality of the latent space of an adversarial auto encoder is brought closer to that of the "true" generative latent space, via a procedure to mask the spurious latent dimensions. We demonstrate through experiments on synthetic and several real-world datasets that the proposed formulation yields betterment in the generation quality.

1 INTRODUCTION

The objective of a probabilistic generative model is to learn to sample new points from a distribution given a finite set of data points drawn from it. Deep generative models, especially the Generative Adversarial Networks (GANs) (Goodfellow et al. (2014)) have shown remarkable success in this task by generating high quality data (Brock et al. (2019)). GANs implicitly learn to sample from the data distribution by transforming a sample from a simplistic distribution (such as Gaussian) to the sample from the data distribution by optimising a min-max objective through an adversarial game between a pair of function approximators called the generator and the discriminator. Although GANs generate high-quality data, they are known to suffer from problems like instability of training (Arora et al. (2017); Salimans et al. (2016)), degenerative supports for the generated data (mode collapse) (Arjovsky and Bottou (2017); Srivastava et al. (2017)) and sensitivity to hyper-parameters (Brock et al. (2019)).

Auto-Encoder (AE) based generative models (Zhao et al. (2017); Kingma and Welling (2013); Makhzani et al. (2016); Tolstikhin et al. (2018)) provide an alternative to GAN based models. The fundamental idea is to learn a lower dimensional latent representation of data through a deterministic or stochastic encoder and learn to generate (decode) the data through a decoder. Typically, both the encoder and decoder are realised through learnable family of function approximators or deep neural networks. To facilitate the generation process, the distribution over the latent space is forced to follow a known distribution so that sampling from it is feasible. Despite resulting in higher data-likelihood and stable training, the quality of generated data of the AE-based models is known to be far away from state-of-the-art GAN models (Dai and Wipf (2019); Grover et al. (2018); Theis et al. (2015)).

While there have been several angles of looking at the shortcomings of the AE-based models (Dai and Wipf (2019); Hoshen et al. (2019); Kingma et al. (2016); Tomczak and Welling (2017); Klushyn et al. (2019); Bauer and Mnih (2019); van den Oord et al. (2017)), an important question seems to have remained unaddressed: How does the dimensionality of the latent space (bottle-neck

[*]Equal contribution
[†]Work partially done when at IIT Delhi

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

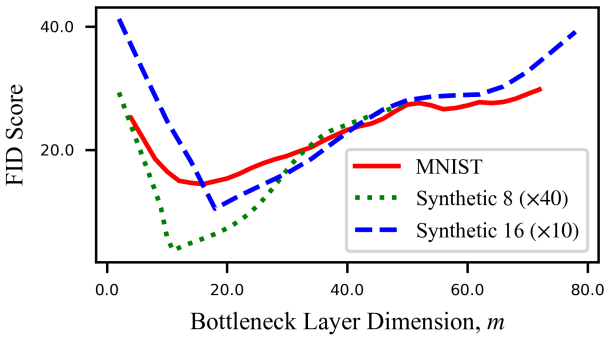


Figure 1: FID score for a Wasserstein Auto-Encoder with varying latent dimensionality m for 2 synthetic datasets of 'true' latent dimensions, $n = 8$ and $n = 16$ and MNIST. It is seen that the generation quality gets worse on both the sides of a certain latent dimensionality. FID scores have been scaled appropriately to bring them in the same range

layer) affect the generation quality in AE-based models?

It is a well-known fact that most of the naturally occurring data effectively lies in a manifold with dimension much lesser than its original dimensionality (Cayton (2005); Law and Jain (2006); Narayanan and Mitter (2010)). Intuitively, this suggests that with functions that Deep Neural Networks learn, there exists an optimal number of latent dimensions, since "lesser" or "extra" number of latent dimensions may result in loss of information and noisy generation, respectively. This observation is also corroborated by empirical evidence provided in Fig. 1 where a state-of-the-art AE-based generative model (Wasserstein Auto-Encoder Tolstikhin et al. (2018)) is constructed on two synthetic (detailed in Section 5) and MNIST datasets, with varying latent dimensionality (everything else kept the same). It is seen that the generation quality metric (FID) follows a U-shaped curve. Thus, to obtain optimal generation quality, a brute-force search over a large range of values of latent dimensionality may be required, which is practically infeasible. Motivated by the aforementioned observations, in this work, we explore the role of latent dimensionality in AE-based generative models, with the following contributions:

1. We model the data generation as a two-stage process comprising of sampling from a "true" latent space followed by a deterministic function.

2. We provide theoretical understanding on the role of the latent space dimensionality on the generation quality, by formalizing the requirements for a faithful generation in of AE-based generative models with deterministic encoder and decoder networks.

3. Owing to the obliviousness of the dimensionality of the "true" latent space in real-life data, we propose a method to algorithmically "mask" the spurious dimensions in AE-based models (and thus call our model the MaskAAE).

4. We demonstrate the efficacy of the proposed model on synthetic as well as large-scale image datasets by achieving better generation quality metrics compared to the state-of-the-art AE-based models.

2 RELATED WORK

Let $\boldsymbol{x}$ denote data points lying in the space $\mathcal{X}$ conforming to an underlying distribution $\Upsilon(\boldsymbol{x})$, from which a generative model desires to sample. An Auto-Encoder based model constructs a lower-dimensional latent space $\mathcal{Z}$ to which the data is projected through an (probabilistic or deterministic) Encoder function, E_κ. An inverse projection map is learned from $\mathcal{Z}$ to $\mathcal{X}$ through a Decoder function D_ψ, which can be subsequently used as a sampler for $\Upsilon(\boldsymbol{x})$. For this to happen, it is necessary that the distribution of points over the latent space $\mathcal{Z}$ is regularized (to some known distribution $\Pi(\boldsymbol{z})$) to facilitate explicit sampling from $\Pi(\boldsymbol{z})$, so that decoder can generate data taking samples from $\Pi(\boldsymbol{z})$ as input. Most of the AE-based models maximize the data likelihood (or a lower bound on it), which is shown (Kingma and Welling (2013); Hoffman and Johnson (2016)) to consist of the sum of two critical terms - (i) the likelihood of the Decoder generated data and, (ii) a divergence measure between the assumed latent distribution, $\Pi(\boldsymbol{z})$, and the distribution imposed on the latent space by the Encoder, $\Psi(\boldsymbol{z}) = \int \Psi(\boldsymbol{z}|\boldsymbol{x})\Upsilon(\boldsymbol{x})d\boldsymbol{x}$, (Hoffman and Johnson (2016); Makhzani et al. (2016)). This underlying commonality, suggests that the success of an AE-based generative model depends upon simultaneously optimising the aforementioned terms. The first criterion is fairly easily ensured in all AE models by minimizing a surrogate function such as the reconstruction error between the samples of the true data and output of the decoder, which can be made arbitrarily small (Burgess et al. (2017); Dai and Wipf (2019); Alain and Bengio (2014)) by increasing the network capacity. It is well recognized that the quality of the generated data relies heavily on achieving the second criteria of bringing the Encoder imposed latent distribution $\Psi(\boldsymbol{z})$ close to the assumed latent prior distribution $\Pi(\boldsymbol{z})$ (Dai and Wipf (2019); Hoffman and Johnson (2016); Burgess et al. (2017)). This can be achieved either by (i) assuming a pre-defined primitive distribution for $\Pi(\boldsymbol{z})$ and modifying the Encoder such that $\Psi(\boldsymbol{z})$ follows assumed $\Pi(\boldsymbol{z})$ (Kingma and Welling (2013); Makhzani et al. (2016); Tolstikhin et al. (2018); Chen et al. (2018); Higgins et al. (2017);

Kim and Mnih (2018); Kingma et al. (2016)) or by (ii) modifying the latent prior $\Pi(\boldsymbol{z})$ to follow whatever distribution $\big(\Psi(\boldsymbol{z})\big)$ Encoder imposes on the latent space (Tomczak and Welling (2017); Bauer and Mnih (2019); Klushyn et al. (2019); Hoshen et al. (2019); van den Oord et al. (2017)).

The seminal paper on VAE (Kingma and Welling (2013)) proposes a probabilistic Encoder which is tuned to output the parameters of the conditional posterior $\Psi(\boldsymbol{z}|\boldsymbol{x})$ which is forced to follow the Normal distribution prior assumed on $\Pi(\boldsymbol{z})$. However, the minimization of the divergence between the conditional latent distribution and the prior in the VAE leads to trade-off between the reconstruction quality and the latent matching, as this procedure also leads to the minimization of the mutual information between $\mathcal{X}$ and $\mathcal{Z}$, which in turn reduces Decoder's ability to render good reconstructions (Kim and Mnih (2018)). This issue is partially mitigated by altering the weights on the two terms of the ELBO during optimization (Higgins et al. (2017); Burgess et al. (2017)), or through introducing explicit penalty terms in the ELBO to strongly penalize the deviation of $\Psi(\boldsymbol{z})$ from assumed prior $\Pi(\boldsymbol{z})$ (Chen et al. (2018); Kim and Mnih (2018)). Adversarial Auto-Encoders (AAE) (Makhzani et al. (2016)) and Wasserstein Auto-Encoders (WAE) (Tolstikhin et al. (2018)) address this issue, by taking advantage of adversarial training to minimize the divergence between $\Psi(\boldsymbol{z})$ and $\Pi(\boldsymbol{z})$, via deterministic Encoder and Decoder networks. There also have been attempts in employing the idea of normalizing flow for distributional estimation for making $\Psi(\boldsymbol{z})$ close to $\Pi(\boldsymbol{z})$ (Kingma et al. (2016); Rezende and Mohamed (2015)). These methods, although improve the generation quality over vanilla VAE while providing additional properties such as disentanglement in the learned space, fail to match the generation quality of GAN and its variants.

In another class of methods, the latent prior $\Pi(\boldsymbol{z})$ is made learnable instead of being fixed to a primitive distribution so that it matches with Encoder imposed $\Psi(\boldsymbol{z})$. In VamPrior (Tomczak and Welling (2017)), the prior is taken as a mixture density whose components are learned using pseudo-inputs to the Encoder. Klushyn et al. (2019) introduces a graph-based interpolation method to learn the prior in a hierarchical way. In van den Oord et al. (2017); Kyatham et al. (2019), discrete latent space is employed, using vector quantization schemes where the prior is learned using a discrete auto-regressive model. While these prior matching methods provide various advantages, there is no mechanism to ward-off the 'spurious' latent dimensions that are known to degrade the generation quality. While there exists a possibility that the Decoder learns to ignore those spurious dimensions by making the corresponding weights zero there is no guarantee or empirical evidence of neglecting those dimensions. Another indirect approach to handle this issue is to add noise to the input data and prevent variance collapse in the latent space through explicit regularization (Rubenstein et al. (2018)). However, this approach avoids the problem instead of solving it. The closest work to ours is 2-stage VAE (Dai and Wipf (2019)), in which the authors show that VAEs struggle to match the latent distribution to an isotropic standard Gaussian when there is a mismatch between the original data manifold dimension and the latent space capacity. To resolve this, they propose two VAEs, where the first one maps the data to a latent code having the same dimension as the latent space capacity, and the second stage then maps the latent spaced mapped in the first stage to an isotropic Gaussian (see supplementary for a detailed discussion).

To summarize, it is observed that without additional modifications, in vanilla AE-based models, the existence of superfluous latent dimensions degrades the generation quality. We formally address this problem, presenting a novel theoretical analysis of the issues involved, and also provide a method to ameliorate this problem by explicitly masking the spurious dimensions in the latent space of AE based models.

3 EFFECT OF LATENT DIMENSIONALITY

3.1 PRELIMINARIES

In this section, we theoretically examine the effect of latent dimensionality on the quality of generated data in AE based generative models. We show that if dimensionality of the latent space $\mathcal{Z}$ is more than the *optimal* dimensionality (to be defined), $\Pi(\boldsymbol{z})$ and $\Psi(\boldsymbol{z})$ diverge too much whereas it being less leads to information loss.

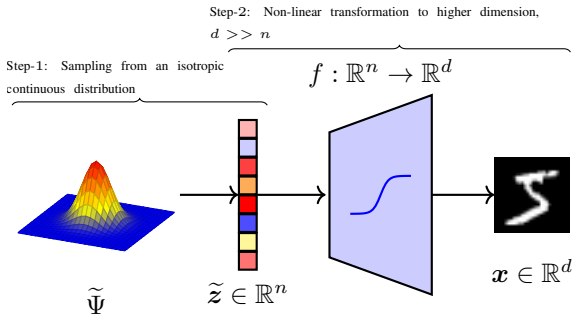


Figure 2: Depiction of the assumed data generation process. Samples drawn from a 'true' latent distribution $\widetilde{\Psi}(\widetilde{\mathbf{z}})$ are passed through a function f to obtain $\boldsymbol{x}$.

To begin with, we allow a certain inductive bias in assuming that nature generates data as described in Figure 2 using the following two-step process: First sam-

ple from some isotropic continuous latent distribution in n-dimensions (call this $\widetilde{\Psi}$ over $\widetilde{\mathcal{Z}}$), and then pass this through a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^d$, where d is the dataset dimensionality. Typically $d >> n$, thereby making data to lie on a low-dimensional manifold in R^d. Since $\widetilde{\mathcal{Z}}$ can intuitively be viewed as the latent space from which the nature is generating the data, we call n the *true latent dimension* and function f, as the *data-generating function*. Note that within this ambit, $\widetilde{\mathcal{Z}}$ forms the domain of f and it is unique only up to its range with following properties:

A1 f is *L-lipschitz*: $\exists$ some finite $L \in \mathbb{R}^+$ satisfying $||f(\widetilde{\boldsymbol{z}}_1) - f(\widetilde{\boldsymbol{z}}_2)|| \leq L||\widetilde{\boldsymbol{z}}_1 - \widetilde{\boldsymbol{z}}_2||,\ \forall \widetilde{\boldsymbol{z}}_1, \widetilde{\boldsymbol{z}}_2 \in \widetilde{\mathcal{Z}}$.

A2 There does not exist $f^* : \mathbb{R}^{n'} \rightarrow \mathbb{R}^d, n' < n$ satisfying A1 such that the range of f is a subset of the range of f^*.

The first property is satisfied by a large class of functions, including neural networks and the second simply states that n, the dimension of the domain (generative latent space) of f is minimal[1]. Hence, it is reasonable to impose these restrictions on data-generating functions. (An illustrative example for A2 is provided in the supp.)

3.2 CONDITIONS FOR GOOD GENERATION

In this section, we formulate the conditions required for faithful generation in latent variable generative models. Let $\Gamma(\boldsymbol{x}, \boldsymbol{z})$ and $\Gamma'(\boldsymbol{x}, \boldsymbol{z})$ denote the true and the (implicitly) inferred joint distribution of the observed and latent variables. The goal of the latent variable generative models is to minimize the negative log-likelihood of $\Gamma'(\boldsymbol{x}, \boldsymbol{z})$ under $\Gamma(\boldsymbol{x}, \boldsymbol{z})$:

$$\mathcal{L}(\Gamma, \Gamma') = - \mathop{\mathbb{E}}_{\boldsymbol{x}, \boldsymbol{z} \sim \Gamma} \big[\log(\Gamma'(\boldsymbol{x}, \boldsymbol{z})) \big] \tag{1}$$

An AE-based generative model would attempt to minimize Eq. 1 by learning two parametric functions, $E_\kappa \triangleq g : \mathbb{R}^d \rightarrow \mathbb{R}^m$ (m is hereafter referred to as *assumed latent dimension / model capacity*) and $D_\psi \triangleq g' : \mathbb{R}^m \rightarrow \mathbb{R}^d$, to approximate the distributions $\Psi(\boldsymbol{z}|\boldsymbol{x})$ and $\Gamma(\boldsymbol{x}|\boldsymbol{z})$, respectively. Further, Eq. 1 can be split into two terms, and the objective of any AE-model can be restated as:

$$\min \Bigg(\underbrace{\mathop{\mathbb{E}}_{\Gamma}[-\log(\Gamma'(\boldsymbol{x}|\boldsymbol{z}))]}_{\text{R1}} + \underbrace{\mathop{\mathbb{E}}_{\Gamma}[\log \frac{1}{\Gamma'(\boldsymbol{z})}]}_{\text{R2}} \Bigg) \tag{2}$$

If E_κ and D_ψ are deterministic (as in the case of AAE (Makhzani et al. (2016)) [2], WAE (Tolstikhin et al. (2018)) etc.), then the two terms in Eq. 2 can be cast as the following two requirements (see the supp. for the proof):

R1 $f(\widetilde{\boldsymbol{z}}) = g'(g(f(\widetilde{\boldsymbol{z}})))\ \forall\ \widetilde{\boldsymbol{z}}\ \in \mathbb{R}^n$. This condition states that the reconstruction error between the real and generated data should be minimal.

R2 The Cross Entropy $\mathcal{H}(\Psi, \Pi)$ between the chosen prior Ψ, and Π on $\mathcal{Z}$ is minimal.

With this, we state and prove the conditions required to ensure R1 and R2 are met with assumed data generation process.

Theorem 1. *With the assumption of data generating process mentioned in Sec.3.1, requirements R1 and R2 (Sec.3.2), can be satisfied iff assumed latent dimension m is equal to true latent dimension n.*

Proof: We prove by contradicting either R1 or R2, in assuming both the cases of $m < n$ or $m > n$.

Case A ($m < n$)*:* For R1 to hold, the range of f must be a subset of the range of g'. Further, since g' is a Neural Network, it satisfies A1. But, by A2, such a function cannot exist if $m < n$.

Case B ($m > n$): For the sake of simplicity, let us assume that $\widetilde{\mathcal{Z}}$ is a unit cube[3] in $\mathbb{R}^n$. We show in Lemma 1 and 2 that in this case, R2 will be contradicted if $m > n$. The idea is to first show that the range of $g \circ f$ will have Lebesgue measure 0 (Lemma 1) and this leads to arbitrarily large $\mathcal{H}$ (Lemma 2).

Lemma 1: Let $\Omega : [0, 1]^\alpha \rightarrow \mathbb{R}^\beta$ be an $L - lipschitz$ function. Then its range $R \in \mathbb{R}^\beta$ has Lebesgue measure 0 in $\mathbb{R}^\beta$ dimensions if $\beta > \alpha$.

Proof: For some $\epsilon \in \mathbb{N}$, consider the set of points:

$$\text{S}=\Big\{ \big(\tfrac{a_0 + 0.5}{\epsilon}, \ldots, \tfrac{a_{\alpha-1} + 0.5}{\epsilon} \big) \Big| a_i \in \{0, \ldots, \epsilon - 1\} \Big\}.$$

Construct closed balls around them having radius $\frac{\sqrt{\alpha}}{2\epsilon}$. It is easy to see that every point in the domain of Ω is contained in at least one of these balls. This is because,

[1] If there exists such an f^*, then that would become the generating function with n' being minimal.

[2] Makhzani et al. (2016), in their work, have observed that the performance of stochastic and deterministic networks are comparable. Thus, We consider only deterministic networks for theoretical analysis and experimentation in our work.

[3] One can easily obtain another function $\nu : [0, 1]^n \rightarrow \widetilde{\mathcal{Z}}$ that scales and translates the unit cube appropriately. Note that for such a ν to exist, we need $\widetilde{\mathcal{Z}}$ to be bounded, which may not be the case for certain distributions like the Gaussian distributions. Such distributions, however, can be approximated successively in the limiting sense by truncating at some large value Rudin et al. (1964)

for any given point, the nearest point in S can be at-most $\frac{1}{2\epsilon}$ units away along each dimension. Also, since Ω is L-lipschitz, we can conclude that the image set of a closed ball having radius r and centre $\boldsymbol{u} \in [0,1]^\alpha$ would be a subset of the closed ball having centre $\Omega(\boldsymbol{u})$ and radius $L \times r$.

The range of Ω is then a subset of the union of the image sets off all the closed balls defined around S. The volume of this set is upper bounded by the sum of the volumes of the individual image balls, each having volume $\frac{c}{\epsilon^\beta}$ where c is a constant having value $\frac{(L)^\beta(\alpha\pi)^{\frac{\beta}{2}}}{\Gamma(\frac{\beta}{2}+1)}$. Therefore,

$$\text{vol(R)} \leq |S| \times \frac{c}{\epsilon^\beta} = \frac{c}{\epsilon^{\beta-\alpha}}. \tag{3}$$

The final quantity of Eq. 3 can be made arbitrarily small by choosing ϵ appropriately. Since the Lebesgue measure of a closed ball is same as its volume, the range of Ω, R has measure 0 in $\mathbb{R}^\beta$. □

Since f, and g are Lipschitz, $g \circ f$ must have a range with Lebesgue measure 0 as a consequence of Lemma 2. Now we show that as a consequence of the range of $g \circ f$ (call it $\mathcal{R}$) having measure 0, the cross-entropy between Π and Ψ goes to infinity.

Lemma 2: If Π and Ψ are two distributions as defined in Sec.3.1 such that the support of the latter has a 0 Lebesgue measure, then $\mathcal{H}(\Pi, \Psi)$ grows to be arbitrarily large.

Proof: Ψ can be equivalently expressed as:

$$\Psi(\boldsymbol{z}) = \begin{cases} \widetilde{\Psi}(\widetilde{\boldsymbol{z}}) & \text{if } \exists\, \widetilde{\boldsymbol{z}}^4 \in \widetilde{\mathcal{Z}} \text{ s.t. } g(f(\widetilde{\boldsymbol{z}})) = \boldsymbol{z}, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Define $\mathbb{I}_\mathcal{R}$ as the indicator function of $\mathcal{R}$, i.e.

$$\mathbb{I}_\mathcal{R}(\boldsymbol{z}) = \begin{cases} 1 & \text{if } \exists\, \widetilde{\boldsymbol{z}} \in \widetilde{\mathcal{Z}} \text{ s.t. } g(f(\widetilde{\boldsymbol{z}})) = \boldsymbol{z}, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Since $\mathcal{R}$ has measure 0 (Lemma 2), we have

$$\int_{\mathbb{R}^m} \mathbb{I}_\mathcal{R}(\boldsymbol{z}) d\boldsymbol{z} = 0 \tag{6}$$

Further, since $\mathbb{I}_\mathcal{R}$ is identically 1 in the support of Ψ,

$$\Psi(\boldsymbol{z}) = \Psi(\boldsymbol{z})\mathbb{I}_\mathcal{R}(\boldsymbol{z}) \tag{7}$$

Next, consider the cross-entropy between Π and Ψ:

$$\begin{aligned} \mathcal{H}(\Pi, \Psi) &= \int_\mathcal{Z} \Pi(\boldsymbol{z})(-\log(\Psi(\boldsymbol{z})))d\boldsymbol{z} \\ &\geq \int_{\mathcal{Z}-\mathcal{R}} \Pi(\boldsymbol{z})(-\log(\Psi(\boldsymbol{z})\mathbb{I}_\mathcal{R}(\boldsymbol{z})))d\boldsymbol{z} \\ &\geq \varrho \int_{\mathcal{Z}-\mathcal{R}} \Pi(\boldsymbol{z}) d\boldsymbol{z} \end{aligned} \tag{8}$$

[4]Note that in general, $\widetilde{\boldsymbol{z}}$ is not unique, and if multiple such $\widetilde{\boldsymbol{z}}$ exist, we have to sum(or perhaps integrate) $\widetilde{\Psi}$ over all such $\widetilde{\boldsymbol{z}}$

for any arbitrarily large positive real ϱ. This holds true because $\mathbb{I}_\mathcal{R}$ is identically 0 over the domain of integration. Further,

$$\begin{aligned} \int_{\mathcal{Z}-\mathcal{R}} \Pi(\boldsymbol{z}) &\geq \int_\mathcal{Z} \Pi(\boldsymbol{z}) - \int_\mathcal{R} \Pi(\boldsymbol{z}) \\ &= 1 - \int_{\mathbb{R}^m} \Pi(\boldsymbol{z})\mathbb{I}_\mathcal{R}(\boldsymbol{z}) d\boldsymbol{z} \\ &\geq 1 - \max_{\mathbb{R}^m}(\Pi(\boldsymbol{z})) \int_{\mathbb{R}^m} \mathbb{I}_\mathcal{R}(\boldsymbol{z}) d\boldsymbol{z} \\ &= 1 \end{aligned} \tag{9}$$

Combining 8 and 9, the required cross-entropy is lower bounded by an arbitrarily large quantity ϱ. □

Thus Lemma 2 contradicts R2 required for good generation when $m > n$. Therefore, to ensure good generation neither $m > n$ nor $m < n$ can be true. Thus, the only possibility is $m = n$. This concludes Theorem 1. □

One can ensure good generation, by satisfying both R1 and R2 via a trivial solution in the form of $g' = f$ with an appropriate g and making $m = n$. However, since neither n nor f is known, one needs a practical method to ensure m approaches n which is described in the next section.

4 MaskAAE (MAAE)

4.1 MODEL DESCRIPTION

Our premise in section 3.2 demands a pair of deterministic Encoder and Decoder networks satisfying R1 and R2, to ensure good quality generation. AE-models with deterministic E_κ and D_ψ networks, such as Adversarial Auto-Encoder (AAE) (Makhzani et al. (2016)) and Wasserstein Auto-Encoder (WAE) (Tolstikhin et al. (2018)) implement R1 by approximating norm-based losses and R2 through an adversarial training mechanism under metrics such as JS-Divergence or Wasserstein distance. However, most of the time, the choice of the latent dimensionality is ad hoc and there is no mechanism to get rid of the excess latent dimensions that is critical for good-quality generation as demanded by Theorem 1. Therefore, in this section, we take the ideas presented in Section 3, and propose an architectural modification on models such as AAE/WAE, such that being initialized with a large enough estimated latent space dimension the model would learn a binary-mask automatically discovering the right number of latent dimensions required.

Specifically, we propose the following modifications in the AAE-like architecture (Makhzani et al. (2016); Tolstikhin et al. (2018)), which contain an additional component called Discriminator (H_ζ) that is used to match

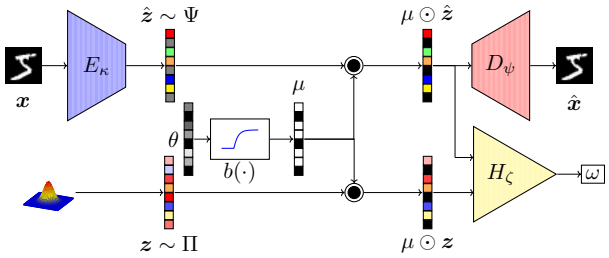


Figure 3: Block Diagram of MaskAAE. It consists of an encoder, E_κ, a decoder, D_ψ, and a discriminator H_ζ as in AAE. A new layer called mask, μ is introduced at the end of the encoder to suppress spurious latent dimensions. The prior also gets multiplied with the same mask vector before going into the Discriminator to ensure prior matching (R2).

$\Psi(\boldsymbol{z})$ and $\Pi(\boldsymbol{z})$ via adversarial learning. Our model, called the MaskAAE is detailed in figure 3.

1. We introduce a trainable mask layer, $\mu \in \{0,1\}^m$, just after the final layer of the Encoder network.

2. Before passing the encoded representation, $\hat{\boldsymbol{z}}$ of an input image $\boldsymbol{x}$ to the decoder network (D_ψ) and the Discriminator network (H_ζ) a Hadamard product is performed between $\hat{\boldsymbol{z}}$ and μ.

3. A Hadamard product is performed between the prior sample, $\boldsymbol{z} \sim \Pi(\boldsymbol{z})$ and the same mask μ as in item (1), before passing it as an input to the discriminator network H_ζ to ensure R2.

4. During inference, the prior samples are multiplied with the learned mask before giving as input to the Decoder (D_ψ) network which serves the generator.

Intuitively, masking of both the encoded latent vector and prior with a same binary mask allows us to work only with a subset of dimensions in the latent space. This means that even though m (the initial assumed latent dimensionality) may be greater than n, mask (if learned properly) reduces the encoded latent space to $\mathcal{R}^n$. This will in-turn facilitate better matching of $\Psi(\boldsymbol{z})$ and $\Pi(\boldsymbol{z})$ (R2) required for better generation.

4.2 TRAINING MaskAAE

MaskAAE is trained exactly similarly as one would train an AAE/WAE but with the addition of a loss term to train the mask layer. Here, we provide the details of the mask-loss only. For a complete description of other AAE/WAE based training loss terms refer to the supplementary.

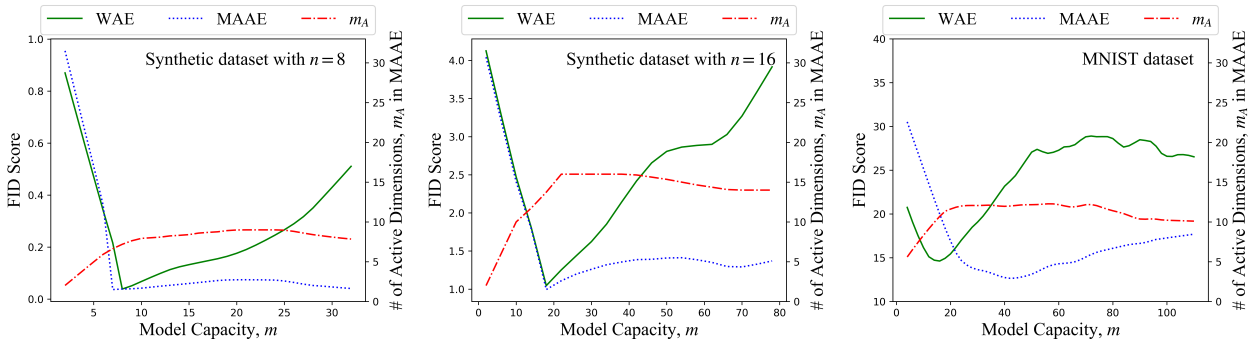
Although, the mask by definition is a binary-valued vector, to facilitate gradient flow during training, we relax it to be continuous valued while penalizing it for deviation from either 0 or 1. Specifically, we parameterize μ using a vector $\theta \in \mathbb{R}^m$ such that $\mu = b(\theta)$ where, $b(\theta) = \max(0, 1 - e^{-\theta})$. θ is initialized by drawing samples from $\mathcal{U}[0, a]$, where $a \in \mathbb{Z}^+$. Intuitively, this parameterization bounds μ in the range $[0, 1)$. Since the mask layer affects both the requirements R1 and R2, it is trained so as to minimize both the norm-based reconstruction error (first term in Eq. 10) and divergence metrics such as JS-divergence or Wasserstein's distance, between the masked prior distribution and the masked encoded latent distribution (second term in Eq. 10). Finally, a polynomial regularizer (third term in Eq. 10) is also added on μ so that any deviation from $\{0, 1\}$ is penalized. Therefore, the final objective function for the mask layer, L_{mask} consists of three terms as below.

$$L_{mask} = \frac{\lambda_1}{s} \sum_{i=1}^{s} ||\boldsymbol{x}^{(i)} - D_\psi(\mu \odot E_\kappa(\boldsymbol{x}^{(i)}))|| + \lambda_2(1+\omega)^2 + \lambda_3 \sum_{j=1}^{m} |\mu_j(\mu_j - 1)| \tag{10}$$

where, $\omega = \frac{1}{s}\sum_i H_\zeta(\mu \odot \boldsymbol{z}^{(i)}) - \frac{1}{s}\sum_i H_\zeta(\mu \odot E_\kappa(\boldsymbol{x}^{(i)}))$ is the Wasserstein's distance, s denotes batch size, and the weights $(\lambda_1, \lambda_2, \lambda_3)$ of different loss terms are hyperparameters. The training algorithm, and the architectures for E_κ , D_ψ and H_ζ are available in the supplementary.

5 EXPERIMENTS AND RESULTS

We divide our experiments into two parts: (a) Synthetic, and (b) Real. In synthetic experiments, we control the data generation process, with a known number of true latent dimensions. Hence, we can compare the performance of our proposed model for several true latent dimensions, and examine whether our method can discover the true number of latent dimensions. This also helps us validate some of the theoretical claims made in Section 3. On the other hand, the objective of the experiments with real datasets is to examine whether our masking based approach can result in a better generation quality as compared to the state-of-the-art AE-based models. We would also like to understand the behaviour of the number of dimensions which are masked in this case (though the true latent dimension may not be known). We also analysed linear and ternary search over a range on the size of the latent space as naïve alternatives. We found them to be computationally prohibitive, taking at least an order of magnitude more time compared to our approach. Refer to supplement Sec. 8 for details.



(a) (b) (c)

Figure 4: (a) and (b) shows FID score for WAE and MAAE and active dimension in a trained MAAE model with varying model capacity, m for synthetic dataset of true latent dimensions, $n = 8$ and $n = 16$, m_A represents the number of unmasked latent dimensions in the trained model and (c) shows the same plots for MNIST dataset.

5.1 SYNTHETIC EXPERIMENTS

In the following description, we will use n to denote the true latent dimension, and m to denote the assumed latent dimension (or model capacity) in line with the notation used earlier in the paper. Assuming that data is generated according to the generation process described in Section 3, we are interested in answering the following questions: (a) Given sufficient model capacity (i.e, $m \geq n$ and sufficiently powerful E_κ, D_ψ and H_ζ), can MAAE discover the true number of latent dimensions? (b) How is the quality of the data generated by MAAE for different values of m?

Ideally, we would expect that whenever $m \geq n$, MAAE masks $(m-n)$ number of dimensions. Further, we would expect that the performance of MAAE is independent of the value of m, whenever $m \geq n$. For each value of m that we experimented with, we trained an equivalent WAE model with exactly same architecture for E_κ , D_ψ and H_ζ as in MAAE without the mask layer. We would expect the performance of the WAE model to deteriorate whenever $m \neq n$ if our theory were to hold correct.

In line with our assumed data generation process, the data for our synthetic experiments is generated as below:

- Sample $\tilde{\boldsymbol{z}} \sim \mathcal{N}(\mu_s, \Sigma_s)$, where the mean $\mu_s \in \mathbb{R}^n$ was fixed to be zero and $\Sigma_s \in \mathbb{R}^{n \times n}$ represents the diagonal co-variance matrix (isotropic Gaussian).
- Compute $\boldsymbol{x} = f(\tilde{\boldsymbol{z}})$, where f is a non-linear function computed using a two-layer fully connected neural network with k units in each layer, $d >> n$ output units, and using leaky ReLU as the non-linearity (refer to the supplement for more details). The weights of these networks are randomly fixed and k was taken as 128.

We set $n = 8$ and 16, and varied m in the range of $[2, 32]$ and $[2, 78]$ with step size 2, for $n = 8$ and $n = 16$ respectively. We use the standard Fréchet Inception Distance (FID) [5] (Heusel et al. (2017)) score between generated and real images to validate the quality of the generated data, because FID has been shown to correlate well the human visual perception and also sensitive to artifacts such as mode collapse (Lucic et al. (2018); Sajjadi et al. (2018)). Figure 4 (a) and (b) presents our results on synthetic data. On X-axis, we plot m and Y-axis (left) plots the FID score comparing MAAE and WAE for different values of m. Y-axis (right) plots the number of active dimensions discovered by our algorithm. It is seen that both MAAE and WAE, achieve the best FID score when $m = n$. But whereas the performance for WAE deteriorates with increasing m, MAAE retains the optimal FID score independent of the value of m. Further, in each case, we get very close to the true number of latent dimensions, even with different values of m (as long as $m > 8$ or 16, respectively). Table 3 of the supplementary material presents the variation of log-likelihood scores for generated data with model capacity (m) for WAE and MaskAEE for synthetic dataset $(n = 16)$; this exhibits a similar behaviour. These results clearly validate our theoretical claims, and also the fact that MAAE is capable of offering good quality generation in practice.

5.2 REAL EXPERIMENTS

Next, we examine the behavior of MAAE on real-world datasets. The true latent data dimensions (n) is unknown for real datasets. However, the behaviour can still be analyzed as the estimated latent dimension (m) is varied. We experiment with the following four image datasets: (a) MNIST (Lecun (2010)) (b) Fashion MNIST (Xiao et al.

[5] We compute the Fréchet Distance between the real and the generated data directly for synthetic experiments. As synthetic data is low-dimensional, computation of Inception Net embedding is not required.

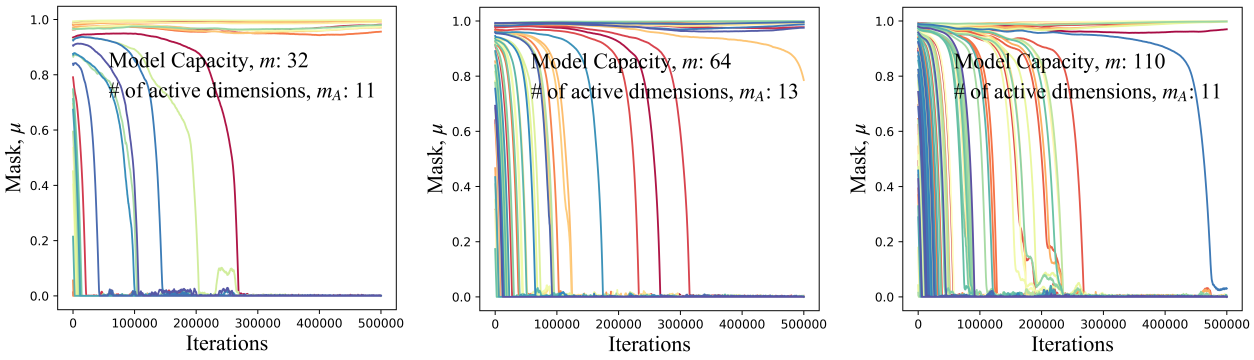


Figure 5: Behaviour of mask in MAAE with different model capacities, m for MNIST dataset. m, in figure (a), (b), and (c) are 32, 64, and 110, respectively. Dimensions active after training are m_A are 11, 13, and 11 respectively.

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

Figure 6: Randomly generated images of (a) MNIST, (b) Fashion MNIST, (c) CelebA, and (d) CIFAR-10 datasets.

(2017)) (c) CIFAR-10 (Krizhevsky (2009)) (d) CelebA (Liu et al. (2015)) with standard test/train splits.

In our first set of experiments, we perform an analysis similar to the one done in the case of synthetic data, for the MNIST dataset. Specifically, we varied the estimated latent dimension (model capacity m) for MNIST from 10 to 110, and analyzed the FID score, as well as the true dimensionality as discovered by the model. For comparison, we also did the same experiment using the WAE model. Figure 4 (c) shows the results. As in the case of synthetic data, we observe a U-shape behavior for the WAE model, with the lowest value achieved at $m = 13$. This validates our thesis that the best performance is achieved at a specific value of latent dimension, which is around 13 in this case. Further, looking at MAAE curve, we notice that the performance (FID score) more or less stabilizes for $m \geq 16$. In addition, the true latent dimension discovered also stabilizes around $10 - 13$ irrespective of m, without compromising much on the generation quality. Note that the same network architecture was used at all points of Figure 4. These observations are in line with the expected behavior of MAAE, and the fact that it can indeed mask the spurious dimensions to achieve good generation quality.

Figure 5 shows the behaviour of mask for model capacity $m = 32, 64$ and 110 on MNIST dataset. Interestingly, in each case, we are able to discover almost the same number of unmasked dimensions, independent of the starting point. It is also observed that the Wasserstein distance is minimized at the point where the mask reaches the optimal point (we refer to the supplementary material for the plots).

Table 1: FID scores for generated images from different AE-based generative models (Lower is better).

	MNIST	Fashion	CIFAR-10	CelebA
VAE (cross-entr.)	16.6	43.6	106.0	53.3
VAE (fixed variance)	52.0	84.6	160.5	55.9
VAE (learned variance)	54.5	60.0	76.7	60.5
VAE + Flow	54.8	62.1	81.2	65.7
WAE-MMD	115.0	101.7	80.9	62.9
WAE-GAN	12.4	31.5	93.1	66.5
2-Stage VAE	12.6	29.3	72.9	44.4
MAAE	**10.5**	**28.4**	**71.9**	**40.5**

Finally, to measure generation quality, we present the FID scores of MAAE in Table 1 along with several state-of-the-art AE-based models mentioned in Sec. 2. Our approach achieves the best FID score on all the datasets compared to the state-of-the-art AE based generative models. Performance of MAAE is also comparable to that of GANs listed in Lucic et al. (2018), despite using a simple reconstruction loss and an isotropic unimodal Gaussian prior. Figure 6 presents 100 randomly selected MAAE-generated samples for each dataset. The better FID scores of MAAE can be attributed to better distribution matching in the latent space between $\Psi(\boldsymbol{z})$

and $\Pi(\boldsymbol{z})$. But a quantative comparison of distributional match is not straight forward as MAAE might mask out some of the latent dimensions resulting in a dimensionality mismatch among the latent space in different models, thus rendering the usual metrics unsuitable. We therefore, calculate the averaged off-diagonal normalized absolute co-variance[6] (NAC) of the encoded latent vectors and report it in Table 2 (Refer supp. for full co-variance matrix). Since $\Pi(\boldsymbol{z})$ is assumed to be an isotropic Gaussian, ideally NAC should be zero and any deviation from zero indicates a mismatch. We use only the unmasked latent dimensions of MAAE for NAC computation, to avoid underestimation by considering the unused dimension. Note that for the same model capacity, MAAE has lesser NAC than the corresponding WAE indicating better distribution matching in the latent space. These results clearly demonstrate that not only MAAE can achieve the best FID scores on various datasets, it also serves as a first step in discovering the underlying latent structure for a given dataset.

Table 2: Average off-diagonal covariance NAC for both WAE and MAAE. m_A represents the number of unmasked latent dimensions in the trained model. It is seen that MAAE has lower NAC values indicating lesser deviation of $\Psi(\boldsymbol{z})$ from $\Pi(\boldsymbol{z})$ as compared to a WAE.

Dataset	Model Capacity	WAE		MAAE	
		m_A	NAC	m_A	NAC
Synthetic_8	16	16	0.040	9	**0.030**
Synthetic_{16}	32	32	0.031	16	**0.013**
MNIST	64	64	0.027	13	**0.020**
FMNIST	128	128	0.025	40	**0.019**
CIFAR-10	256	256	0.017	120	**0.013**
CelebA	256	256	0.046	77	**0.039**

6 DISCUSSION AND CONCLUSION

Despite demonstrating its pragmatic success, we critically analyze the possible deviations of the practical cases from the presented analysis. More often than not, the naturally occurring data contains some noise superimposed onto the actual image. Thus, theoretically one can argue that this noise can be utilized to minimize the divergence between the distributions. Practically, however, this noise has a very low amplitude, so it can only work for a few extra dimensions, giving a slight overestimate of n. Further, in practice, not all latent dimensions contribute equally to the data generation. Since the objective of our model is to ignore noise dimensions, it may at times end up throwing away meaningful data dimensions which do not contribute significantly. This can lead to a slight underestimate of n (which is occasionally observed during experimentation). Finally, neural networks, however deep, can represent only a certain level of complexity in a function which is simultaneous advantageous and otherwise. It is good because while we have shown that certain losses cannot be made zero for $m \neq n$, universal approximators can bring them arbitrarily close to zero, which is practically the same thing. Due to their limitation, however, we end up getting a U-curve. It is a disadvantageous because even when $m \geq n$, the encoder and decoder networks might be unable to learn the appropriate functions, and for $m \leq n$, the Discriminator fails to make distributions apart. This implies that instead of discovering the exact same number of dimensions every time, we might get a range of values near the true latent dimension. Also, the severity of this problem is likely to increase with the complexity of the dataset (again corroborated by the experiments).

To conclude, in this work, we have taken a step towards constructing an optimal latent space for improving the generation quality of Auto-Encoder based neural generative model. We have argued that, under the assumption two-step generative process, the optimal latent space for the AE-model is one where its dimensionality matches with that of the latent space of the generative process. Further, we have proposed a practical method to arrive at this optimal dimensionality from an arbitrary point by masking the 'spurious' dimensions in AE-based generative models. Finally, we have shown the effectiveness of our method in improving the generation quality using several experiments on synthetic and real datasets.

[6]Refer supplementary material for mathematical formula.

Acknowledgement

We thank IIT Delhi HPC facility[7] for computational resources. We also express our gratitude to Prof. Sivananthan Sampath (Dept. of Maths., IIT Delhi) for his help in formalising the proof of Lemma 1. Parag Singla is supported by the DARPA Explainable Artificial Intelligence (XAI) Program with number N66001-17-2-4032, the Visvesvaraya Young Faculty Fellowships by Govt. of India and IBM SUR awards. Himanshu Asnani acknowledges the support of Department of Atomic Energy, Government of India, under project no. 12-R&D-TFR-5.01-0500. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of the funding agencies.

[7]http://supercomputing.iitd.ac.in

References

G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *The Journal of Machine Learning Research*, vol. 15, no. 1, 2014.

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

M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. of ICML*, 2017.

S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang, "Generalization and equilibrium in generative adversarial nets (gans)," in *Proc. of ICML*, 2017.

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

A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *Proc. of ICLR*, 2019.

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

L. Cayton, "Algorithms for manifold learning," *Univ. of California at San Diego Tech. Rep*, vol. 12, no. 1, 2005.

T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Proc. of NeuRIPS*, 2018.

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

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NeuRIPS*, 2014.

A. Grover, M. Dhar, and S. Ermon, "Flow-gan: Bridging implicit and prescribed learning in generative models," in *Proc. of AAAI*, 2018.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Proc. of NeuRIPS*, 2017.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. of NeuRIPS*, 2017.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. of ICLR*, 2017.

M. D. Hoffman and M. J. Johnson, "Elbo surgery: yet another way to carve up the variational evidence lower bound," in *Workshop in Advances in Approximate Bayesian Inference, NIPS*, vol. 1, 2016.

Y. Hoshen, K. Li, and J. Malik, "Non-adversarial image synthesis with generative latent nearest neighbors," in *Proc. of CVPR*, 2019.

H. Kim and A. Mnih, "Disentangling by factorising," in *Proc. of ICML*, 2018.

D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013.

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

A. Klushyn, N. Chen, R. Kurle, B. Cseke, and P. van der Smagt, "Learning hierarchical priors in VAEs," in *Proc. of NeuRIPS*, 2019.

A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

V. Kyatham, D. Mishra, T. K. Yadav, D. Mundhra *et al.*, "Variational inference with latent space quantization for adversarial resilience," *arXiv preprint arXiv:1903.09940*, 2019.

M. H. Law and A. K. Jain, "Incremental nonlinear dimensionality reduction by manifold learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 3, 2006.

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

Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. of ICCV*, 2015.

M. Lucic, K. Kurach, M. Michalski, O. Bousquet, and S. Gelly, "Are gans created equal? a large-scale study," in *Proc. of NeuRIPS*, 2018.

A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," in *Proc. of ICLR*, 2016.

H. Narayanan and S. Mitter, "Sample complexity of testing the manifold hypothesis," in *Proc. of NeuRIPS*, 2010.

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

P. K. Rubenstein, B. Schoelkopf, and I. Tolstikhin, "Wasserstein auto-encoders: Latent dimensionality and random encoders," in *ICLR Workshop*, 2018.

W. Rudin *et al.*, *Principles of mathematical analysis*. McGraw-hill New York, 1964, vol. 3.

M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," in *Proc. of NeuRIPS*, 2018.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proc. of NeuRIPS*, 2016.

A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, "Veegan: Reducing mode collapse in gans using implicit variational learning," in *Proc. of NeurIPS*, 2017.

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

I. Tolstikhin, O. Bousquet, S. Gelly, and B. Scholkopf, "Wasserstein auto-encoders," in *Proc. of ICLR*, 2018.

J. M. Tomczak and M. Welling, "VAE with a vampprior," *arXiv preprint arXiv:1705.07120*, 2017.

A. van den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," in *Proc. of NeuRIPS*, 2017.

H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

S. Zhao, J. Song, and S. Ermon, "Infovae: Information maximizing variational autoencoders," 2017.

7 THEORY

7.1 JUSTIFICATION FOR A2

In the section 3 of the main paper, we present two assumptions that we make on our data generating function f, one of which is: *There does not exist $f^* : \mathbb{R}^{n'} \rightarrow \mathbb{R}^d, n' < n$ satisfying A1 such that the range of f is a subset of the range of f^*.* Here we offer some intuition on why this assumption is essential and how things would be incomplete in its absence.

Since our hypothesis relies on the fairly intuitive fact that there is a true manifold dimension n, it is not hard to see why we need this to be uniquely defined, given the data manifold of a data set (note that this is equivalent to having an infinite number of data points, a fair upper-bound to any real data set). Now, to appreciate the importance of the second assumption, consider the following functions

$$\begin{aligned} f_1(x, y, z) &= (x + y, x - y, y - x) \\ f_2(x, y) &= (x + y, x - y, y - x) \end{aligned} \tag{11}$$

Further, let $\widetilde{\Psi}$ be the uniform distribution over the unit cube and square respectively. It is easy to see that the data manifold will be exactly identical in both these cases. Since both f_1 and f_2 satisfy A1, in the absence of A2, both qualify as f meaning that n could be both 2 and 3, which is undesirable. To mitigate this, we introduce A2, whereby the existence of f_2 disqualifies f_1 and n becomes fixed at 2 (note that we also need the fact that there is no function from the real line to a superset of the data manifold, and in this case, it happens to be true by lemma 1)

7.2 DERIVATIONS FOR R1 AND R2

In the main paper, we have stated the following conditions as requirements for optimal generation:

R1 $f(\widetilde{\boldsymbol{z}}) = g'(g(f(\widetilde{\boldsymbol{z}}))) \; \forall \; \widetilde{\boldsymbol{z}} \; \in \mathbb{R}^n$.

R2 $\mathcal{H}(\Psi, \Pi)$ on $\mathcal{Z}$ is minimal.

In this section, we shall show that these are indeed necessary and sufficient to minimise the cross-entropy between the true data distribution Γ and the generated data distribution Γ'.
Since auto-encoder based frameworks work through a latent space $\mathcal{Z}$, their objective is to minimise the cross entropy between the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{z}$. We define these joint distributions as:

$$\begin{aligned} \Gamma(\boldsymbol{x}, \boldsymbol{z}) &= \Upsilon(\boldsymbol{x})\delta(g(\boldsymbol{x}) = \boldsymbol{z}) \\ \Gamma'(\boldsymbol{x}, \boldsymbol{z}) &= \Pi(\boldsymbol{z})\delta(\boldsymbol{x} = g'(\boldsymbol{z})) \end{aligned} \tag{12}$$

where δ is the Dirac delta function. The cross entropy between these two distributions can be broken down as follows:

$$\begin{aligned} \mathcal{L}(\Gamma, \Gamma') &= \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{z})\sim\Gamma}(-\log(\Gamma'(\boldsymbol{x}, \boldsymbol{z}))) \\ &= \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{z})\sim\Gamma}(-\log(\Gamma'(\boldsymbol{x}|\boldsymbol{z}))) \\ &\quad + \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{z})\sim\Gamma}(\log(\frac{1}{\Pi(\boldsymbol{z})})) \end{aligned} \tag{13}$$

The first term in the final expression can further be expressed as:

$$\begin{aligned} &\mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{z})\sim\Gamma}(-\log(\Gamma'(\boldsymbol{x}|\boldsymbol{z}))) \\ &= -\int_{\boldsymbol{x}}\int_{\boldsymbol{z}} \Upsilon(\boldsymbol{x})\delta(g(\boldsymbol{x}) = \boldsymbol{z}) \log(\delta(\boldsymbol{x} = g'(\boldsymbol{z})))d\boldsymbol{z}d\boldsymbol{x} \\ &= -\int_{\boldsymbol{x}} \Upsilon(\boldsymbol{x}) \log(\delta(\boldsymbol{x} = g'(g(\boldsymbol{x}))))d\boldsymbol{x} \end{aligned} \tag{14}$$

If the equality inside the delta function does not hold at any point, it will push the logarithm to negative infinity, and in turn the entire quantity will become very high. To prevent this, we need $\boldsymbol{x} = g'(g(\boldsymbol{x}))$ at all points. Since $\boldsymbol{x}$ varies over the range of f, this reduces to R1.
In the second term, the expectation is over a joint distribution, but the variable $\boldsymbol{x}$ never appears inside the expectation, so it is safe to take the expectation over the marginal of $\boldsymbol{z}$. However, this marginal, $\Gamma(\boldsymbol{z})$ is exactly the distribution imposed on the latent space by the encoder and the data distribution, which we previously called Ψ. Making this change turns this term into $\mathcal{H}(\Psi, \Pi)$ and since we need this to be minimal, we recover the R2.

7.3 DISCUSSION

Our generative process assumes that n, the dimension of the input space of f is minimal. Intuitively this means that each of the latent dimension contributes in generation of some (possibly small) region in the domain of the observed data but it is not necessary that every latent dimension (independently) affects every observed data point.

For example, consider the case where a leaf is being photographed. A young leaf in broad daylight has colour roughly (120,100,50) in the HSL system. As the age of the leaf increases, the lightness starts to fall, but a similar fall in lightness will also be observed with fading daylight. At this point, lighting conditions and age of leaf have identical effect on the appearance of the leaf. However, after a point, age will start reducing the hue of the

leaf, while lighting conditions will continue decreasing its lightness. Since at this point these two factors influence the outcome differently, they can be separate factors in our input space. On the other hand, the distance from which the photo was taken and the optical zoom of the lens will always have similar effect, and therefore, only one of these is allowed as a factor. Note that this example is presented for illustrative purposes only, and in real cases, the input factors are unlikely to directly map to real-world causes.

7.4 INTUITION FOR LEMMA 1

In the main paper, we have claimed that given a set S, defined as: $S=\Big\{(\frac{a_0+0.5}{\epsilon}, \ldots, \frac{a_{n-1}+0.5}{\epsilon}) \big| a_i \in \{0, \ldots, \epsilon - 1\}\Big\}$ if we build closed balls around each point in S, then every point in $[0,1]^n$ lies in atleast one of these balls. To further the intuition behind this, we present an illustration for the case where $n = 2$ and $\epsilon = 4$.

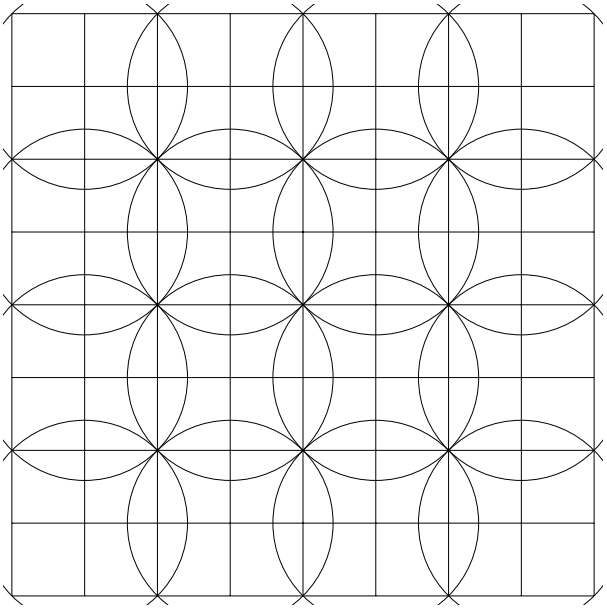


Figure 7: For $n = 2$, $\epsilon = 4$ radius of each ball, $r = \frac{\sqrt{(2)}}{8}$.

In figure 7 the big square represents the unit square. By the nature of Cartesian space, this can be tiled completely by 64 smaller squares having side $\frac{1}{8}$ units. The 16 circles represent the closed balls in $\mathbb{R}^2$. It is easy to see that each of the smaller squares lies completely in some circle. Since each point in the bigger square must lie in one of the smaller squares, they also lie in one of the circles.

7.5 COMPARISON TO 2-STAGE VAE

Our theoretical contributions share some of its motivations with Dai and Wipf (2019). In this section, we try to put our contributions in perspective with Dai and Wipf (2019).

To begin with, it is important to note that the most important difference between the two works lies in the base model used. Dai and Wipf (2019) intend to improve VAE (Kingma and Welling (2013)) while we aim at enhancing WAE (Tolstikhin et al. (2018)). Because of the above mentioned reason, the issues faced and solved by the two papers are in fact, complementary to each other.

Throughout our paper, we point out the fact that it is detrimental to have $m \neq n$ in a WAE. While having $m < n$ is still problematic for a VAE, the $m > n$ scenario is naturally dealt with using the encoder variance. In the words of Dai and Wipf (2019), *"For superfluous dimensions that are unnecessary for representing $\boldsymbol{x}$, the associated encoder variance in these directions can be pushed to one. This will optimize $KL[q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})]$ along these directions, and the decoder can selectively block the residual randomness to avoid influencing the reconstructions."* However, in order to have this stochasticity VAEs also need to introduce a decoder variance γ, which in turn creates several superfluous minima if the dimension of the manifold being learnt is strictly less than that of the space in which it is embedded, and the main purpose of the 2-Stage VAE construction is to avoid these.

Since, we have used deterministic WAEs, that doesn't need an additional parameter γ, and therefore a similar issue is unlikely to exist with WAE. However, this also means that $m > n$ becomes a major problem, which we attempt to solve by introducing a Mask layer.

8 NAIVE ALTERNATIVES TO MAAE

While MaskAAE offers a principled methodology to automatically suppress the spurious dimensions in the latent code, one naïve approach to find out the optimal latent dimensionality might be linear search or ternary search. However, these approaches are computationally prohibitive as they require an order of magnitude more time as compared to the proposed algorithm.

It is natural to assume that we have in mind a range of possible values for n (MaskAAE needs this to set the model capacity). The simplest way to find optimal m is to do a *linear search* over this range. This will involve training one WAE for each possible value, and then select the one with the best FID. A somewhat more sophisticated approach is to do a *ternary search* over the range, as described in Algorithm 1. Here we set tol to 1, and $func$ to be a function that trains a WAE with the said latent dimensions and returns the FID score obtained. Although this method makes $O(\log n)$ calls to $func$, it has

significantly large constants than that of binary search (approximately 3.42 times larger). It returns the minima as long as $func$ is uni-modal. While this is the true for FID as a function of model capacity, the variance is very high because we obtain a noisy estimate which varies between different executions. This might create severe problems for the ternary search.

Both these methods, however, are significantly more resource intensive than MaskAAE. Consider the case where we want to find n for MNIST, and assume that it lies somewhere between 2 and 100. In our experiments we found that, it takes approximately 900 minutes to train a WAE with model capacity, $m = 2$ and $m = 100$. So training any intermediate sized WAE will consume similar time. A linear search would require 99 WAEs to be trained, while in the ideal scenario, ternary search will need to train 18 models. Compared to this, we need to train MaskAAE only once, which takes about 1000 minutes for model capacity, $m = 100$.

Algorithm 1 Pseudo code for ternary search

function FINDMIN($low, high, tol, func$)
  while $high - low \geq tol$ **do**
    $mid_1 \leftarrow \lfloor \frac{2 \times high + low}{3} \rfloor$
    $mid_2 \leftarrow \lceil \frac{high + 2 \times low}{3} \rceil$
    if $func(mid_1) < func(mid_2)$ **then**
      $high \leftarrow mid_1$
    else
      $low \leftarrow mid_2$
    end if
  end while
  return $high$
end function

9 OBJECTIVE, TRAINING AND ARCHITECTURE OF MAAE

In this section we describe the MAAE model and the training algorithm in detail.

1. **Re-construction Pipeline**: This is the standard pipeline in any given AE based model, which tries to minimize the reconstruction loss (R1). An input sample $\boldsymbol{x}$ is passed through the encoder E_κ results in $\hat{\boldsymbol{z}}$, the corresponding representation in the latent space. The new addition here is the Hadamard product with the mask μ (explained next), resulting in the masked latent space representation $\mu \odot \hat{\boldsymbol{z}}$. The masked representation is then fed to the decoder D_ψ to obtain the re-constructed output $\hat{\boldsymbol{x}}$. The goal here is to minimize the norm of the difference between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$.

2. **Masking Pipeline**: Introduction of a mask is one of the novel contributions of our work, and this is the second part of our architecture presented in the middle of the Figure 3. Our mask is represented as μ and is a binary vector of size m (model capacity). Ideally, the mask would be a binary vector, but in order to make it learnable, we relax it to be continuous valued, while imposing certain regularizers so that it does not deviate too much from 0 or 1 during learning.

3. **Distribution-Matching Pipeline:** This is the third part of our architecture presented at the bottom of Figure 3. Objective of this pipeline is to minimize the distribution loss between a prior distribution, Π, and the distribution Ψ imposed on the latent space by the encoder. $\boldsymbol{z}$ is a random vector sampled from the prior distribution, whose Hadamard product is taken with the mask μ (similar to in the case of encoder), resulting in a masked vector $\mu \odot \boldsymbol{z}$. This masked vector is then passed through the network H_ζ , where the goal is to separate out the samples coming from prior distribution ($\boldsymbol{z}$) from those coming from the encoded space ($\hat{\boldsymbol{z}}$) using some divergence metric. We use the principles detailed in Arjovsky et al. (2017) using the Wasserstein's distance to measure the distributional divergence. Note that H_ζ has two inputs namely, samples of $\Pi(\boldsymbol{z})$ and output of E_κ.

9.1 OBJECTIVE FUNCTIONS OF MAAE

Next, corresponding to each of the components above, we present a loss function where s represents the batch size.

1. **Auto-Encoder Loss:** This is the standard loss to capture the quality of re-construction as used earlier in the AE literature. In addition, we have a term corresponding to minimization of the variance over the masked dimensions in the encoded output in a batch. The intuition is that encoder should not inject information into the dimensions which are going to be masked anyway. The loss is specified as:

$$L_{ae} = \frac{\alpha_1}{s} \sum_{i=1}^{s} ||\boldsymbol{x}^{(i)} - D_\psi(\mu \odot E_\kappa(\boldsymbol{x}^{(i)}))|| + \alpha_2(\delta^T Diag(A)) \tag{15}$$

A represents the co-variance matrix for the encoding matrix $E_\kappa(X)$, X being the data matrix for the

Algorithm 2 Pseudo code for the training loop of MAAE

Hyper-parameters: $\alpha_1 = 1$, $\alpha_2 = 100$, $\gamma = 10$, $\lambda_1 = 1000$, $\lambda_2 = 1$ and $\lambda_3 = \frac{2}{m}$

function TRAIN
  $\lambda_3 \leftarrow \frac{2}{m}$
  for $i \leftarrow 1$ to $training\_steps$ **do**
    for $j \leftarrow 1$ to $ae\_training\_ratio$ **do**
      $t \leftarrow ij + j$
      $g_\kappa(t) \leftarrow \nabla_\kappa L_{ae}$
      $\kappa_t \leftarrow \kappa_{t-1} - \frac{\eta_{ae} g_\kappa(t)}{\sqrt{\rho g_\kappa(t) + (1-\rho) g_\kappa^2(t-1) + \epsilon}}$
      $g_\Psi(t) \leftarrow \nabla_\Psi L_{ae}$
      $\Psi_t \leftarrow \Psi_{t-1} - \frac{\eta_{ae} g_\Psi(t)}{\sqrt{\rho g_\Psi(t) + (1-\rho) g_\Psi^2(t-1) + \epsilon}}$
    end for
    for $j \leftarrow 1$ to $disc\_training\_ratio$ **do**
      $t \leftarrow ij + j$
      $g_\zeta(t) \leftarrow \nabla_\zeta L_{disc}$
      $\zeta_t \leftarrow \zeta_{t-1} - \frac{\eta_{disc} g_\zeta(t)}{\sqrt{\rho g_\zeta(t) + (1-\rho) g_\zeta^2(t-1) + \epsilon}}$
    end for
    $g_\kappa(i) \leftarrow \nabla_\kappa L_{gen}$
    $\kappa_i \leftarrow \kappa_{i-1} - \frac{\eta_{gen} g_\kappa(i)}{\sqrt{\rho g_\kappa(i) + (1-\rho) g_\kappa^2(i-1) + \epsilon}}$
    if $i\%reg\_schedule\_interval == 0$ **then**
      $\lambda_3 \leftarrow \lambda_3 \times 2$
    end if
    $g_M(i) \leftarrow \nabla_M L_{mask}$
    $M_i \leftarrow M_{i-1} - \frac{\eta_{mask} g_M(i)}{\sqrt{\rho g_M(i) + (1-\rho) g_M^2(i-1) + \epsilon}}$
  end for
end function

current batch. δ is the vector obtained by applying the function $a(u) = e^{-\gamma \times u}$ point-wise to μ. α_1, α_2 and γ are hyperparameters.

2. **Generator Loss:** This is the loss capturing the quality of generation in terms of how far the generated distribution is from the prior distribution. This loss measures the ability of the encoder to generate the samples such that they are coming from $\Pi(z)$ which is ensured using the generator loss mentioned in Arjovsky et al. (2017):

$$L_{gen} = -\frac{1}{s}\sum_{i=1}^{s} H_\zeta(\mu \odot E_\kappa(\boldsymbol{x}^{(i)})) \tag{16}$$

3. **Distribution-Matching Loss:** This is the loss incurred by the Distribution-matching network, H_ζ in matching the distributions. We use Wasserstein's distance (Arjovsky et al. (2017)) to measure the distributional closeness with the following loss:

$$L_{dm} = -\frac{1}{s}\sum_{i=1}^{s} H_\zeta(\mu \odot \boldsymbol{z}^{(i)}) + \frac{1}{s}\sum_{i=1}^{s} H_\zeta(\mu \odot \hat{\boldsymbol{z}}^{(i)}) + \frac{\beta_2}{s}\sum_{i=1}^{s}\left(||\nabla^{(i)}_{\boldsymbol{z}_{avg}} H_\zeta(\mu \odot \boldsymbol{z}^{(i)}_{avg})|| - 1\right)^2 \tag{17}$$

Recall that $\hat{\boldsymbol{z}}^{(i)} = E_\kappa(\boldsymbol{x}^{(i)})$. Further, we have used $\boldsymbol{z}^{(i)}_{avg} = \beta_1 \boldsymbol{z}^{(i)} + (1-\beta_1)\hat{\boldsymbol{z}}^{(i)}$. β_1, β_2 are hyper parameters, with $\beta_1 \sim \mathcal{U}[0,1]$, and β_2 set as in (Gulrajani et al. (2017)).

4. **Masking Loss:** This is the loss capturing the quality of the current mask. The loss is a function of three terms (1) Auto-encoder loss (2) distribution matching loss (3) a regularizer to ensure that μ parameters stay close to 0 or 1. This can be specified as:

$$L_{mask} = \frac{\lambda_1}{s}\sum_{i=1}^{s} ||\boldsymbol{x}^{(i)} - D_\psi(\mu \odot E_\kappa(\boldsymbol{x}^{(i)}))|| + \lambda_2(1+\omega)^2 + \lambda_3 \sum_{j=1}^{m} |\mu_j(\mu_j - 1)| \tag{18}$$

where $\omega = \frac{1}{s}\sum_i H_\zeta(\mu \odot \boldsymbol{z}^{(i)}) - \frac{1}{s}\sum_i H_\zeta(\mu \odot E_\kappa(\boldsymbol{x}^{(i)}))$ is the Wasserstein's distance. Here λ_1 and λ_2 and λ_3 are hyper-parameters (Supp. material).

9.2 TRAINING ALGORITHM

During training, we optimize each of the four losses specified above in turn. Specifically, in each learning loop, we optimize the L_{ae}, L_{dm}, L_{gen} and L_{mask}, in that order using a learning schedule. We use RMSProp for our optimization as described in algorithm 2.

9.3 ARCHITECTURE FOR SYNTHETIC DATASET

Here we provide the detailed architecture of E_κ D_ψ and H_ζ for different experiments performed in this work.

E_κ

$\boldsymbol{x} \in \mathbb{R}^{128}$
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_m$

D_ψ

$\boldsymbol{z} \in \mathbb{R}^m$
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_m$

H_ζ

$\boldsymbol{z} \in \mathbb{R}^m$
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_{1000}$ $\to$ ReLU
$\to$ FC$_1$

9.4 ARCHITECTURE FOR REAL DATASET

9.4.1 MNIST

E_κ

$\boldsymbol{x} \in \mathbb{R}^{28\times28\times1}$
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_m$

D_ψ

$\boldsymbol{z} \in \mathbb{R}^m$
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{28\times28}$ $\to$ Sigmoid
$\to$ Reshape$_{28\times28}$

H_ζ

$\boldsymbol{z} \in \mathbb{R}^m$
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ ReLU
$\to$ FC$_1$

9.4.2 Fashion MNIST

E_κ

$\boldsymbol{x} \in \mathbb{R}^{28\times28\times1}$
$\to$ CONV$_{64;k=(4,4);s=(2,2)}$ $\to$ BN $\to$ ReLU
$\to$ CONV$_{128;k=(4,4);s=(2,2)}$ $\to$ BN $\to$ ReLU
$\to$ FC$_{1024}$ $\to$ BN $\to$ ReLU
$\to$ FC$_m$

D_ψ

$\boldsymbol{z} \in \mathbb{R}^m$
$\to$ FC$_{1024}$ $\to$ BN $\to$ ReLU
$\to$ FC$_{7\times7\times128}$ $\to$ BN $\to$ ReLU
$\to$ Reshape$_{7\times7\times128}$
$\to$ TCONV$_{128;k=(4,4);s=(2,2)}$ $\to$ BN $\to$ ReLU
$\to$ TCONV$_{128;k=(4,4);s=(2,2)}$ $\to$ BN $\to$ ReLU
$\to$ CONV$_{1;k=(3,3);s=(1,1)}$ $\to$ Sigmoid

H_ζ Same as in 9.4.1

9.4.3 CIFAR-10

E_κ

$$
\begin{aligned}
&\boldsymbol{x} \in \mathbb{R}^{32\times32\times3}\\
&\to \text{CONV}_{128;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV}_{128;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV}_{256;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{FC}_{1024} \to \text{BN} \to \text{ReLU}\\
&\to \text{FC}_{1024} \to \text{BN} \to \text{ReLU}\\
&\to \text{FC}_{1024} \to \text{BN} \to \text{ReLU}\\
&\to \text{FC}_{1024} \to \text{BN} \to \text{ReLU}\\
&\to \text{FC}_{m}
\end{aligned}
$$

D_ψ

$$
\begin{aligned}
&\boldsymbol{z} \in \mathbb{R}^{m}\\
&\to \text{FC}_{2\times2\times512} \to \text{BN} \to \text{ReLU}\\
&\to \text{Reshape}_{2\times2\times512}\\
&\to \text{TCONV}_{256;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{256} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_TCONV}_{256;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{256} \to \text{BN} \to \text{ReLU}\\
&\to \text{TCONV}_{256;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{256} \to \text{BN} \to \text{ReLU}\\
&\to \text{TCONV}_{256;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{256} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV}_{3;k=(3,3);s=(1,1)} \to \text{Sigmoid}
\end{aligned}
$$

H_ζ Same as in 9.4.1

9.4.4 CelebA

E_κ

$$
\begin{aligned}
&\boldsymbol{x} \in \mathbb{R}^{64\times64\times3}\\
&\to \text{CONV}_{16;k=(3,3);s=(1,1)} \to \text{BN}\\
&\to \text{CONV\_RES\_BLOCK}_{16} \to \text{CONV}_{32;k=(4,4);s=(2,2)}\\
&\to \text{CONV\_RES\_BLOCK}_{32} \to \text{CONV}_{64;k=(4,4);s=(2,2)}\\
&\to \text{CONV\_RES\_BLOCK}_{64} \to \text{CONV}_{64;k=(4,4);s=(2,2)}\\
&\to \text{CONV\_RES\_BLOCK}_{64}\\
&\to \text{FC\_RES\_BLOCK}_{512}\\
&\to \text{FC}_{m}
\end{aligned}
$$

D_ψ

$$
\begin{aligned}
&\boldsymbol{z} \in \mathbb{R}^{m}\\
&\to \text{FC}_{2\times2\times16} \to \text{BN} \to \text{ReLU}\\
&\to \text{Reshape}_{2\times2\times16}\\
&\to \text{TCONV}_{32;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{32} \to \text{BN} \to \text{ReLU}\\
&\to \text{TCONV}_{64;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{64} \to \text{BN} \to \text{ReLU}\\
&\to \text{TCONV}_{128;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{128} \to \text{BN} \to \text{ReLU}\\
&\to \text{TCONV}_{256;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{256} \to \text{BN} \to \text{ReLU}\\
&\to \text{TCONV}_{512;k=(4,4);s=(2,2)} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV\_RES\_BLOCK}_{512} \to \text{BN} \to \text{ReLU}\\
&\to \text{CONV}_{3;k=(3,3);s=(1,1)} \to \text{Sigmoid}
\end{aligned}
$$

H_ζ Same as in 9.4.1

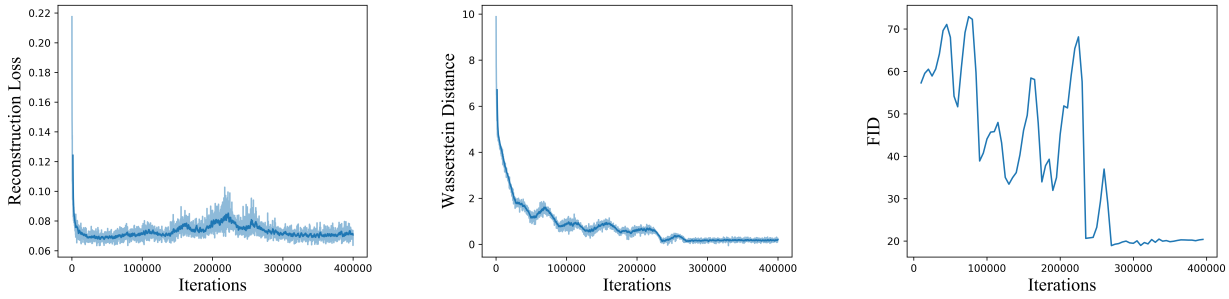
10 EXPERIMENTAL RESULTS

10.1 DETAILS OF THE SYNTHETIC DATASET

Figure 9, shows the architecture for synthetic data generation. The input layer has n nodes, representing n dimensions of the samples from a Gaussian distribution. The hidden layer and output layer introduce two layers of non-linearity and blow up the dimension from n to 128 of the synthetic dataset. In our experiments we have chosen $n = 8$, and $n = 16$.

10.2 LOG-LIKELIHOOD SCORE ON SYNTHETIC DATASET

As discussed in section 5.1 of the main paper, for evaluating the model performance on the synthetic dataset, we compute the Frechet Distance directly between the real and generated data as the data is already low dimensional and calculating Inception features does not make sense for the synthetic data. Here, we evaluate the models by computing the log likelihood of the generated data. We do so by fitting a kernel density estimator on the real data and then computing likelihood of the generated data under that distribution. We use a kernel density estimator with a Gaussian kernel and we find the optimal value for the kernel bandwidth by doing a line search on a held out validation set of the real data. As seen in section 5.1, we again observe that MAAE performs better as compared to a WAE when there is a dimension mismatch.



(a) (b) (c)

Figure 8: (a) Reconstruction loss, (b) Wasserstein distance, and (c) FID plot w.r.t. training iterations for MAAE model on MNIST for model capacity $m = 110$.

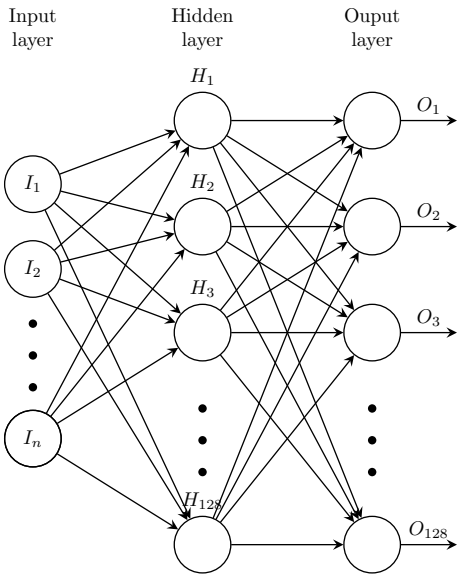


Figure 9: Architecture for Synthetic Dataset generation.

Table 3: Effect of model capacity, m on generation LL scores of WAE and MAAE (for Synthetic$_{16}$ dataset).

	$m = 8$	$m = 16$	$m = 24$	$m = 40$	$m = 64$	$m = 80$
WAE	-196.05	-163.334	-159.31	-163.09	-167.19	-177.08
MAAE	-196.85	-160.08	-156.71	-156.89	-159.08	-154.97

10.3 ANALYSIS OF TRAINING OF MAAE ON MNIST

As discussed in section 5.2 of the main paper, we can see in figure 8 (b) that for $m = 110$, the Wasserstein distance becomes zero when number of active dimensions, $m_A = 11$ (Refer to Figure 5(c) in main paper). Also in Figure 8 (a), and (c) we see the reconstruction error stabilizes at that point and the FID score becomes minimum.

10.4 NORMALISED ABSOLUTE CO-VARIANCE MATRIX: WAVE VS MAAE

The following formula is used to compute the co-variance matrix over a batch size $b_s = 5000$.

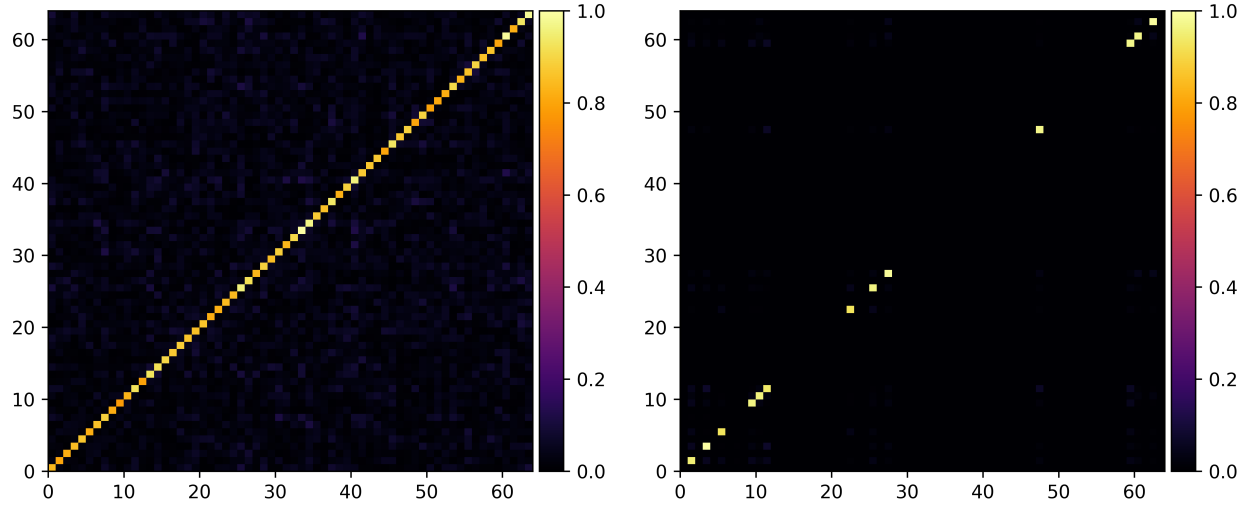
$$\Sigma = |\sum_{i=1}^{i=b_s} (\hat{\boldsymbol{z}}^{(i)} - \mu_{\hat{\boldsymbol{z}}})(\hat{\boldsymbol{z}}^{(i)} - \mu_{\hat{\boldsymbol{z}}})^T| \tag{19}$$

In figure 10, 11, 12, and 13 we have plotted the normalized co-variance matrix $\frac{\Sigma - min(\Sigma)}{max(\Sigma) - min(\Sigma)}$.

Also from figure 10, 11, 12, and 13, we see that the off-diagonal entries in the co-variance matrix corresponding to a masked dimension in MAAE model are very close to zero. Therefore, for a fair comparison of the average off-diagonal value in table 2 of the main paper, we neglect those dimensions in the MAAE matrix.

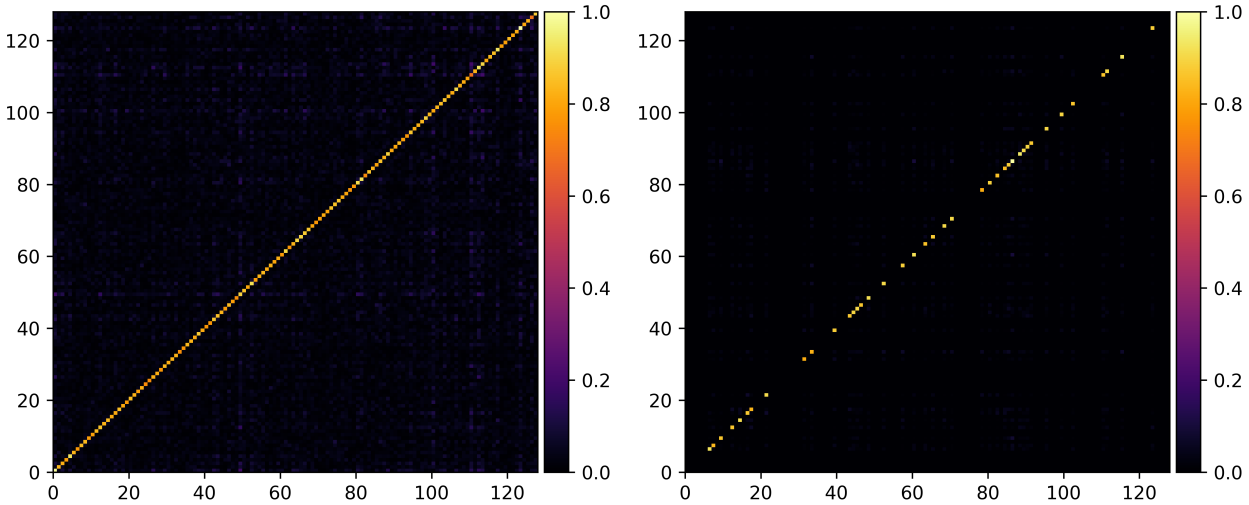
10.5 Computing Resources and Average Runtime

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, Fashion MNIST, CIFAR-10, and CELEBA is approximately 17 hours, 17 hours, 40 hours and 100 hours respectively.



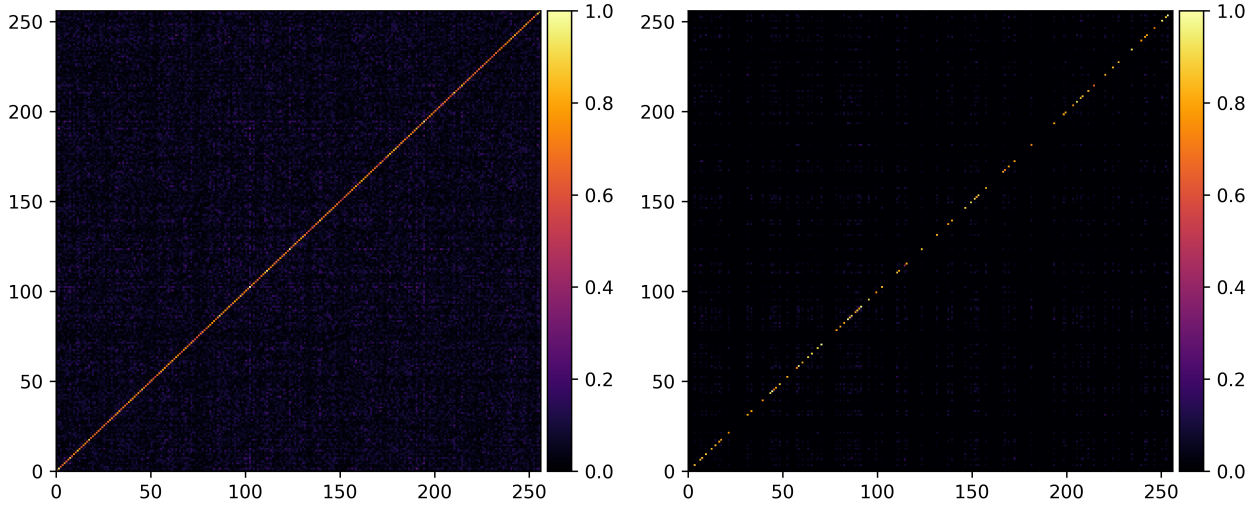
(a) (b)

Figure 10: Co-variance Matrix of (a) WAE (b) MAAE latent representation for MNIST dataset.



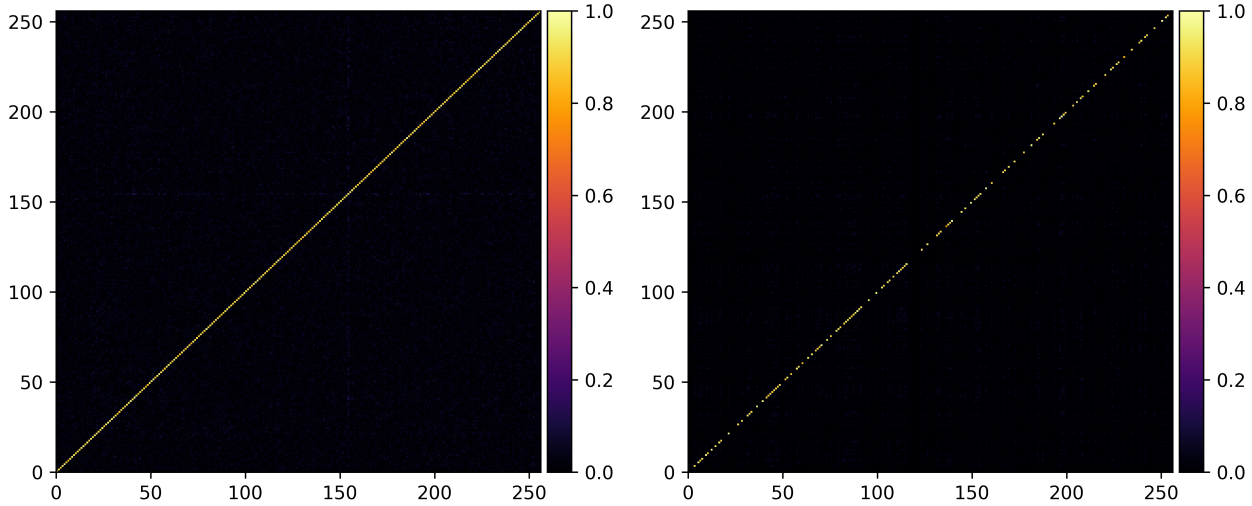
(a) (b)

Figure 11: Co-variance Matrix of (a) WAE (b) MAAE latent representation for Fashion MNIST dataset.



(a) (b)

Figure 12: Co-variance Matrix of (a) WAE (b) MAAE latent representation for CelebA dataset.



(a) (b)

Figure 13: Co-variance Matrix of (a) WAE (b) MAAE latent representation for CIFAR-10 dataset.