Machine Learning
https://doi.org/10.1007/s10994-022-06220-w

GENs: generative encoding networks

Surojit Saha[1] · **Shireen Elhabian**[1] · **Ross Whitaker**[1]

Received: 18 October 2021 / Revised: 22 April 2022 / Accepted: 2 July 2022

Abstract
Mapping data from and/or onto a known family of distributions has become an important topic in machine learning and data analysis. Deep generative models (e.g., generative adversarial networks) have been used effectively to match known and unknown distributions. Nonetheless, when the form of the target distribution is known, analytical methods are advantageous in providing robust results with provable properties. In this paper, we propose and analyze the use of nonparametric density methods to estimate the Jensen-Shannon divergence for matching unknown data distributions to known target distributions, such Gaussian or mixtures of Gaussians, in latent spaces. This analytical method has several advantages: better behavior when training sample quantity is low, provable convergence properties, and relatively few parameters, which can be derived analytically. Using the proposed method, we enforce the latent representation of an autoencoder to match a target distribution in a learning framework that we call a generative encoding network. Here, we present the numerical methods for bandwidth estimation; derive the expected distribution of the data in the latent space; show the advantages over the adversarial counterpart; study the properties of the latent space such as entropy, sample generation, interpolation; and demonstrate the application of the method in the real world.

Keywords Nonparametric density estimation · Latent space regularization · Generative model

Editor: Krzysztof Dembczynski and Emilie Devijver.

✉ Surojit Saha
surojit@cs.utah.edu

Shireen Elhabian
shireen@sci.utah.edu

Ross Whitaker
whitaker@cs.utah.edu

[1] Scientific Computing and Imaging Institute, School of Computing, University of Utah, Salt Lake City 84112, Utah, USA

Published online: 19 September 2022

1 Introduction

Autoencoder-based generative models have proved their merit to be used as a generative model besides building a latent representation that can be used for statistical analysis and downstream tasks, such as classification and regression. In autoencoder-based generative models, the low-dimensional data manifold in the latent space of an autoencoder is regularized to match a target, defined *a priori*. A slew of such methods have been developed lately that primarily differ in terms of the regularization penalty over the reconstruction loss of the autoencoder. In this paper, we propose an autoencoder-based generative model that uses the Jensen-Shannon divergence (JSD) to match the data distribution in the latent space (aka encoded distribution), approximated by the kernel density estimate (KDE), to the target. In addition to the general properties of the autoencoder-based generative models, the proposed method has provable properties of the encoded distribution.

Research in statistical modeling and deep learning has made great strides in the discovery of latent spaces in the context of nonlinear, high-dimensional, and very general transformations, such as those developed by autoencoders that employ deep neural networks (Tschannen et al., 2018). While encoder technologies, such as deterministic denoising (Vincent et al., 2008) and contractive (Rifai et al., 2011) autoencoders, can produce latent representation for samples, the structure of the data population in that latent space is often underspecified or unconstrained, such that one cannot readily reason about that distribution or sample from them. Thus, such models often fail as generators of new samples.

Generators, on the other hand, such as those produced by the generative adversarial network (GAN) (Goodfellow et al., 2014; Radford et al., 2016), have demonstrated impressive capabilities to produce new, very realistic samples (both qualitatively and quantitatively) from complex distributions described by training data. This is achieved, typically, by learning a transformation from a known distribution in a latent space into the high-dimensional data space. However, it is difficult to achieve convergence (Kodali et al., 2017; Liu et al., 2017; Mescheder et al., 2018; Barnett, 2018), optimizing the objective function of the vanilla GAN. Different training strategies have been adopted to address the stability of training GANs (Salimans et al., 2016; Roth et al., 2017). Addressing stability in training GANs using different divergence measures, regularization penalties and training strategies has led to the development of many variants of the vanilla GAN, such as the fGAN (Nowozin et al., 2016), WGAN (Arjovsky & Bottou, 2017), WGAN-GP (Gulrajani et al., 2017), LS-GAN (Mao et al., 2015), and MMD-GAN (Li et al., 2017). The autoregressive model (Larochelle & Murray, 2011; Germain et al., 2015) is another class of generative model that, unlike the GAN, explicitly learns the data distribution as products of conditional distributions built using some hidden representations. These methods are, however, limited by the slow sampling process and are not as popular as GANs.

Inverse mapping (from data samples into the latent space) is another concern with GANs. Therefore it is often difficult to reason about the absolute or relative positions of generated sample in the latent space. Models that map training samples into latent spaces with well-defined properties are of significant value and interest. For instance, using such mappings, we can compute data densities in the latent space, which could be relevant for anomaly detection (Chalapathy & Chawla, 2019). We might also want to compare or manipulate data samples in the latent space, and thereby generate new, realistic samples in controlled ways (Zhu et al., 2016). These motivations, and others, have led to the development of autoencoder-based generative models that seek to build latent spaces that have two properties: (i) an approximate invertible mapping from the data space to the latent

space and (ii) data in the latent space with defined statistical characteristics that lead to probabilistic reasoning and sample generation. Autoencoder-based generative models learn a joint distribution of data and latent variables where the encoded distribution is mapped to a prior distribution that makes it a generative model. These methods are very stable in terms of training, and are not finicky about the architectural choices and hyperparamter settings, as experienced with GANs. The variational autoencoder (VAE) (Kingma & Welling, 2014) learns the data distribution by maximization of the evidence lower bound (ELBO) that matches the conditional posterior to a prior. Other alternatives, such as the adversarial autoencoder (AAE) (Makhzani et al., 2016), and the Wasserstein autoencoder (WAE) (Tolstikhin et al., 2018), replace the stochastic encoder and decoder with the deterministic version and match the aggregate posterior to the prior distribution, unlike VAEs. AAEs and WAE-GANs employ a discriminator in the latent space to match the encoded distribution to the known target distribution (e.g., standard normal). However, finding the optimum discriminator is not an easy task. In addition, the min-max optimization problem with competing networks is highly sensitive to the hyperparameter settings and training strategies.

In this paper, we take an alternative approach. Rather than using an adversary to compare distributions, we propose a nonadversarial framework, *generative encoding networks* or GENs, to enforce the latent representation of an autoencoder to a desired target distribution. The main contributions of this paper can be summarized as follows:

- Use of the KDE for approximation of the data distribution in the latent space, which is used in the computation of the JSD loss.
- A robust, automated method for the KDE bandwidth estimation that uses the known structure of the target distribution.
- An analytical proof that the proposed optimization, on convergence, coincides to, within a known scaling factor, the target distribution.
- Empirical results showing that the KDE-based JSD computation is as or more effective than the results from adversarial training, particularly with limited training data.
- Using different measures of efficacy on multiple data sets, it is demonstrated that the generative capability of the proposed method is as good or better than the state-of-the-art (SOTA) autoencoder-based generative models.
- A demonstration of the use of the latent space built with GENs for outlier detection with comparisons against SOTA methods.

2 Related work

Directly related to the GEN are deep, generative models that build low-dimensional latent spaces with known/quantifiable properties (e.g., known density such as multivariate normal) and learn a functional mapping from this latent space to the data space. VAEs (Kingma & Welling, 2014; Rezende et al., 2014) introduce a parametric prior distribution on the latent space and learn an inference network (i.e., encoder) jointly with the generative model (i.e., decoder) to maximize a variational lower bound of the otherwise intractable marginal log-likelihood of the training data. Main contributions of the VAE can be summarized as follows: (i) efficient posterior inference using the inference model, which is an approximation to the intractable true posterior, and (ii) efficient optimization of the variational lower bound by using the reparameterization strategy. Nonetheless, VAEs often fail to match the marginal (aka aggregate) posterior to the prior distribution

due to the presence of *pockets* or *holes* in the aggregate posterior, which indicates that regions strongly supported under the prior have low density under aggregate posterior. This adversely affects the quality of the generated samples in VAEs (Rosca et al., 2018; Bauer & Mnih, 2019). An analysis of the ELBO (Hoffman & Johnson, 2016) shows that with Gaussian as the prior may lead to over-regularization, and the prior should be learned. Following this idea, flexible and richer priors, VampPrior and LARS priors, were proposed by Tomczak and Welling (2018) and Bauer and Mnih (2019), respectively. To improve the sample quality produced by the VAE, a two-stage VAE was proposed by Dai and Wipf (2019), which uses two VAEs for addressing suboptimal solutions produced due to mismatch with the prior. Other VAE variants improve matching distributions in the latent space through learning a data-driven prior distribution using normalizing flows in the low-dimensional latent space (Bhalodia et al., 2020; Xiao et al., 2019). VAEs could produce degenerate solutions when the KL divergence between the conditional posterior and prior approaches zero. This phenomenon is known as *posterior collapse* (Lucasz et al., 2019; Dai et al., 2020), and in this scenario the information about data in the latent space is completely ignored. Use of powerful decoders in the VAE can also lead to this scenario. Different strategies, such as lower regularizing effect, use of discrete latent representation (Oord et al., 2017) and ensuring the KL divergence has a lower bound (Razavi et al., 2019), have been developed to circumvent this problem.

VAE variants, such as AAEs (Makhzani et al., 2016) and WAEs (Tolstikhin et al., 2018), match the aggregate posterior distribution to target distribution using the JSD or maximum mean discrepancy (MMD) regularizers in the latent space. The AAE or WAE-GAN uses a discriminator (neural network) to implicitly match distributions in the latent space. However, the correct choice of the discriminator, setting the hyperparameters, and figuring the training strategy is challenging in practice. The generative moment matching network (GMMN) (Swersky & Zemel, 2015) similar to the WAE-MMD uses the MMD to match the encoded and target distribution that matches all the moments of the corresponding distributions. Nevertheless, the GMMN uses a pretrained autoencoder, unlike the WAE-MMD, which jointly minimizes the reconstruction loss and divergence measure. Although moment matching technique is more stable (in terms of training) than an adversarial setup, the correct choice of kernels and related hyperparameters plays an important role in the performance of the model.

The regularized autoencoder (RAE) (Ghosh et al., 2020) is a variant of the VAE that replaces the variational framework in VAEs with a deterministic framework. This method is an extension of the constant-variance VAEs (CV-VAEs) that assumes the diagonal covariance entries to be a constant. Moreover, in the RAE, the noise added to the decoder input using the reparameterization strategy is replaced with explicit regularization over the decoder parameters, such as the L-2 regularization. Overall, the RAE resembles a vanilla autoencoder with different regularization techniques applied over the decoder parameters. However, to use this as a generative model, an ex post density estimation is required to be done over the latent representation after the model is trained.

GANs, in their basic form (Goodfellow et al., 2014), do not provide reliable mappings of the samples in the data space back into the latent space. Encoder-decoder GAN variants (e.g., BiGAN (Donahue et al., 2017), ALI (Dumoulin et al., 2017)) simultaneously learn inference and generator networks. However, reconstructed samples from their latent representation do not preserve sample identity since the data-latent correspondence is learned through a discriminator that approximates a density ratio between their joint distribution. A summary of the pros and cons of different methods discussed in this section is reported in Table 9 of "Appendix A".

3 Generative encoding networks

In autoencoder-based generative models, an encoded distribution in the latent space, p_e, is formed to resemble a target distribution, p_t (selected *a priori*), which is achieved by minimization of the following loss function:

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left\{\| \mathbf{x} - D_\theta(E_\phi(\mathbf{x})) \|_2^2 + \lambda \cdot D_\mathbf{z}(p_e(\mathbf{z}), p_t(\mathbf{z}))\right\}. \tag{1}$$

In Eq. (1), E_ϕ and D_θ are the encoder and decoder of the autoencoder with parameters ϕ and θ, respectively. Representation in the latent space is defined as $\mathbf{z} = E_\phi(\mathbf{x})$, where $\mathbf{z} \in \mathbf{R}^l$. The first term in Eq. (1) is the reconstruction loss of the conventional autoencoder, and the second is a regularization with a divergence measure, $D_\mathbf{z}$, which penalizes differences between the encoded distribution $p_e(\mathbf{z})$ and the target distribution, $p_t(\mathbf{z})$ with $\lambda > 0$ as a hyperparameter that mediates the compromise. The $D_\mathbf{z}$ can take different forms such as the Kullback-Leibler divergence (KLD) in matching the conditional distribution $p_e(\mathbf{z} \mid \mathbf{x})$ to the prior $p_t(\mathbf{z})$ as in the VAE, JSD for implicitly matching the aggregate posterior $p_e(\mathbf{z})$ to the target $p_t(\mathbf{z})$ in the AAE and WAE-GAN, or MMD in the WAE-MMD.

Makhzani et al. (2016), in the context of GANs, note that an ideal discriminator or *adversary* classifying samples between the p_t and p_e with a cross entropy loss computes the ratio of densities required for the JSD, and thus the cross entropy is proportional to the JSD. This result is very powerful, but requires the design and training of a discriminator that can approximate the ideal one, with the associated hyperparameters, architectural choices, and training strategies. The proposed method is based on the following observation: *in scenarios where the p_t has a known regular form, which is true in the context of autoencoder-based generative models, we can estimate the JSD if we have a good approximation to the p_e through samples, without the need of an additional neural network or adversary.*

$$\begin{aligned}\mathrm{JSD}\left[p_e, p_t\right] = &\mathbb{E}_{\mathbf{z}'\sim p_e}\left\{\log\left[\frac{p_e(\mathbf{z}')}{p_t(\mathbf{z}') + p_e(\mathbf{z}')}\right]\right\} \\ &+ \mathbb{E}_{\mathbf{z}''\sim p_t}\left\{\log\left[\frac{p_t(\mathbf{z}'')}{p_t(\mathbf{z}'') + p_e(\mathbf{z}'')}\right]\right\}\end{aligned} \tag{2}$$

In this work, we use the $\mathrm{JSD}\left[p_t, p_e\right]$, given by Eq. (2), for standard normal as the target distribution (leaving out the constants), for penalizing the difference between the p_e and p_t. The JSD is an expectation over the ratio of the densities where the first term is the expectation over the encoded distribution, p_e, and the second term is the expectation over the target distribution, p_t. Relative to other choices of divergence, this penalty has the advantages of being a metric with respect to the two distributions and having values between 0 and 1 (in the analytical case), which controls the influence of the penalty under a wide range of circumstances. However, while the expectation can be approximated with a sample mean, it requires an estimate of the ratios of probabilities, which typically requires us to estimate the encoded density, p_e. Estimation of the encoded density, p_e, is a challenging task. In this work, the p_e is estimated using the KDE with m samples and is defined as follows:

$$p_e(\mathbf{z}') = \frac{1}{m}\sum_{i=1}^{m} K\left(\frac{||\mathbf{z}' - \mathbf{z}i||}{h}\right) \tag{3}$$

where $h \in \mathbf{R}^+$ denotes the kernel bandwidth and K is the chosen kernel at $\mathbf{z}i$. Without loss of generality, we consider isotropic Gaussian kernel, and denote it as $G_h(\mathbf{z}' - \mathbf{z}i)$. In principle, the target distribution, p_t, of GENs, can be *any distribution* from which we can sample, but in this paper we focus on the cases where the target distribution is Gaussian or mixtures of Gaussians.

This strategy leaves us with two important decisions regarding hyperparameters. The first is the choice of the number of KDE samples, m, to be used in the KDE-based estimate of the p_e and second is the bandwidth, h, used in the same estimate. Of course, bandwidth selection for the KDE is, in general, a challenging problem without a completely satisfying general solution. However, because we are quantifying differences with known target distributions, where accuracy is most important when distributions are similar, we can use the target distribution itself, the dimensionality of the latent space, l, and the size of the sample set used in the KDE (m above) to estimate the optimal bandwidth, h_{opt}. We seek the h_{opt} that best differentiates two sample sets from the target distribution, and therefore maximizes the JSD between two sample sets.

Given only the latent dimension, l, and number of KDE samples, m, we can estimate the optimal bandwidth by either root trapping or a fixed-point method on h, by evaluating the derivative of (2) with respect to h, that is $\partial \text{JSD}[p_e, p_t]/\partial h$. For the target distribution as the standard normal, Table 1 shows the optimal bandwidth estimates for a variety of latent dimensions, l, and number of KDE samples, m. As expected, the bandwidth increases with higher dimension and fewer samples. It is also observed that for certain latent dimensions, l, and sample size, m, the estimated bandwidth is greater than 1 (standard deviation of the target distribution). This will result in degenerate solutions. Other KDE estimates, such as the variable bandwidth KDE where h potentially varies with the distance from the mean of the target distribution is a possible workaround for this problem which will be studied in the future scope of this work Silverman (1986).

The novelty of this work lies in the computation of the JSD using the KDE of the p_e, and estimation of the optimum bandwidth, h_{opt}, for the p_e, leveraging the information of the target distribution, p_t, known *a priori*. This eliminates the onerous task of finding the optimum *adversary* as used in the computation of the JSD in AAEs and WAE-GANs. In addition, use of analytical method for the computation of density in the encoded distribution, p_e, offers additional benefits, such as deriving provable properties of the encoded distribution on convergence, as discussed in Sect. 4.

The loss function of GENs can thus be formulated as,

Table 1 Optimal bandwidths, h_{opt}, estimated to maximize the JSD between two standard normal samples sets

l/m	250	500	1000	2000	4000	8000
2	0.46 ± 0.08	0.41 ± 0.04	0.38 ± 0.03	0.34 ± 0.02	0.31 ± 0.03	0.28 ± 0.01
5	0.65 ± 0.03	0.59 ± 0.02	0.55 ± 0.02	0.51 ± 0.01	0.47 ± 0.01	0.44 ± 0.01
10	0.77 ± 0.02	0.73 ± 0.01	0.69 ± 0.01	0.66 ± 0.01	0.63 ± 0.01	0.6 ± 0.01
20	0.92 ± 0.01	0.89 ± 0.01	0.86 ± 0.00	0.82 ± 0.01	0.80 ± 0.00	0.77 ± 0.00
40	> 1.0	> 1.0	> 1.0	0.98 ± 0.00	0.95 ± 0.00	0.94 ± 0.00

Error bars are computed over 10 trials. Notice that the bandwidth increases with increasing dimensions (vertical) or decreasing sample size (horizontal)

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left\{\| \mathbf{x} - D_\theta(E_\phi(\mathbf{x})) \|_2^2 + \lambda \cdot \mathrm{JSD}\left[p_e, p_t\right]\right\} \tag{4}$$

where the KDE-based JSD estimate is used to match the *encoded* distribution to the desired or *target* distribution in the latent space.

In this work, the KDE-based estimate of the encoded distribution, p_e, represents the overall state of the encoder network E_ϕ, and we propose to let the p_e lag the updates of the autoencoder, much as we would do with adversarial networks where, for a single update of the discriminator (the discriminator lags), the generator is updated multiple times. This entails an update of the encoder parameters with stochastic gradient descent using the gradient from the first term of (2), which is an expectation over the samples $\mathbf{z}'$ that comprise the *minibatch* (i.e., sampled from the training data). In this scenario, the second term in (2) does not affect the update of the autoencoder.

Algorithm 1 : **GENs training.** Minibatch stochastic gradient descent of GENs loss 4.

Input: Training samples $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, Minibatch size n_b, Latent dimension l, Number of lagged samples m for the KDE, Number of minibatch updates the KDE estimate should lag k_b, Relative scaling factor for the JSD loss λ

Output: encoder parameters ϕ and decoder parameters θ

Estimate the optimal kernel bandwidth h_{opt} given (l, m)
Initialize ϕ and θ
Initialize lagging samples for the KDE
Initialize minibatch index $b \leftarrow 0$
for number of minibatch updates **do**
    Sample a minibatch of size n_b from $p(\mathbf{x})$, $\mathcal{X}_b = \{\mathbf{x}_1, \ldots \mathbf{x}_{n_b}\}$
    Encode minibatch samples in the latent space $\mathcal{Z}'_b = \{\mathbf{z}'_1, \ldots \mathbf{z}'_{n_b}\}$, where $\mathbf{z}'_i = E_\phi(\mathbf{x}_i)$
    Update the encoder parameters, ϕ using stochastic gradient descent:

$$\nabla_\phi \frac{1}{n_b} \sum_{i=1}^{n_b} \left\{\| \mathbf{x}_i - D_\theta(E_\phi(\mathbf{x}_i)) \|_2^2\right\} + \lambda \nabla_\phi \frac{1}{n_b} \sum_{i=1}^{n_b} \left\{\log \frac{p_e(\mathbf{z}'_i)}{p_t(\mathbf{z}'_i) + p_e(\mathbf{z}'_i)}\right\}$$

    Update the decoder parameters, θ using stochastic gradient descent:

$$\nabla_\theta \frac{1}{n_b} \sum_{i=1}^{n_b} \| \mathbf{x}_i - D_\theta(E_\phi(\mathbf{x}_i)) \|_2^2$$

    if $b \mod k_b$ **then**
        Update the KDE samples using the current state of the encoder
    end if
    $b \leftarrow b + 1$
end for

The GEN simultaneously updates the encoder E_ϕ, and the decoder D_θ to minimize the expected reconstruction loss in data space, $\mathbf{R}^d$, and JSD in the latent space, $\mathbf{R}^l$ (using

KDE of the encoded distribution). Algorithm 1 outlines the training of GENs. Here, we see the relationship with several other methods. The KDE estimate of JSD is similar to an adversary. Its single parameter, bandwidth (h) is chosen to maximize the discrimination (similar to an adversary) between samples from the same distributions (i.e., pessimistic assumptions). Furthermore, minimization of the JSD loss (by updating the encoder parameters) set up a min-max game as in the AAE—except that we know the optimal/final value of the single parameter, bandwidth, h, in the latent space, unlike the AAE, which optimizes parameters of a neural network to find this estimate. There is also a relationship with the VAE formulation. The aggregate latent distribution is a convolution of data samples with a kernel. The difference is that the width of that kernel in the GEN is chosen specifically to aid analysis in the latent space relative to a target distribution, rather than basing it on a hypothetical noise model. Furthermore, the JSD penalty directly operates on the aggregate latent distribution, similar to the AAE and WAE. As demonstrated in Sect. 5, this results in latent distributions that are better matched to the target.

4 Latent space distribution

The use of an analytical method for estimating density (of the encoded distribution, p_e) in the GEN facilitates us in systematic examination of the properties of the final empirical distribution of the data in the latent space, when the training has converged, i.e., when there is no gradient from the JSD loss and the reconstruction loss has saturated. In this section, we derive the closed form expression of the resultant distribution in the latent space of the GEN when the target distribution is standard normal. However, this can be generalized to any multivariate Gaussian distribution parameterized by (μ, Σ). When the training of the GEN converges, the encoder parameters, ϕ, are not updated and the gradient of the $\mathrm{JSD}[p_e, p_t]$, with respect to the encoded data sample, $\mathbf{z}'$ (where $\mathbf{z}' = E_\phi(\mathbf{x})$) is zero. Under this condition, we derive the expected distribution in the latent space of GENs, which holds true for any lagging set of size m, used to build the KDE of p_e.

Theorem 1 *Let $G_h(\mathbf{z}' - \mathbf{z}i)$ be an isotropic Gaussian kernel with mean $\mathbf{z}i$ and standard deviation (bandwidth) h, and the encoder is trained to convergence with the KDE-based JSD loss as used in GENs, and a standard normal target distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The distribution of the training data is mapped (in the latent space) to an isotropic, Gaussian distribution with mean zero and standard deviation $(1 - h^2)^{1/2}$.*

Proof of Theorem 1 Setting the derivative of the JSD with respect to $\mathbf{z}'$, $\partial \mathrm{JSD}[p_e, p_t]/\partial \mathbf{z}'$, to 0 we get,

$$\mathbf{z}' = \frac{1}{(1-h^2)} \left[\frac{1}{\sum_{i=1}^{m} G_h(\mathbf{z}' - \mathbf{z}_i)} \right] \left[\sum_{i=1}^{m} G_h(\mathbf{z}' - \mathbf{z}_i) \mathbf{z}_i \right]. \tag{5}$$

The expression in (5) should hold $\forall \mathbf{z}' \in \mathcal{Z}'$. and rearranging the same we get,

$$\mathbf{z}' \sum_{i=1}^{m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right) = \left(\frac{1}{h^2} \left[\sum_{i=1}^{m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right)\left(\mathbf{z}' - \mathbf{z}_i\right) \right] \right). \tag{6}$$

The expression in (6) should hold for all configuration of $\mathbf{z}_i$'s and hence in the expected sense we can write (6) as follows:

$$\begin{aligned} &\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_m)} \left\{ \mathbf{z}' \sum_{i=1}^{m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right) \right\} \\ &= \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_m)} \left\{ \frac{1}{h^2} \left[\sum_{i=1}^{m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right)\left(\mathbf{z}' - \mathbf{z}_i\right) \right] \right\}. \end{aligned} \tag{7}$$

$$\begin{aligned} &\text{Let, } \mathbf{A} = \sum_{i=1}^{m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right) \text{ and } \mathbf{B} = \sum_{i=1}^{m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right)\left(\mathbf{z}' - \mathbf{z}_i\right). \\ &\mathbf{z}' \int_{\mathbf{z}_1} \int_{\mathbf{z}_2} .. \int_{\mathbf{z}_m} \mathbf{A} p(\mathbf{z}_1, \mathbf{z}_2, .., \mathbf{z}_m) d\mathbf{z}_1 d\mathbf{z}_2 .. d\mathbf{z}_m \\ &= \frac{1}{h^2} \int_{\mathbf{z}_1} \int_{\mathbf{z}_2} .. \int_{\mathbf{z}_m} \mathbf{B} p(\mathbf{z}_1, \mathbf{z}_2, .., \mathbf{z}_m) d\mathbf{z}_1 d\mathbf{z}_2 .. d\mathbf{z}_m. \end{aligned} \tag{8}$$

We know, that $\mathbf{z}_i$s' are independent and hence we can split the joint probability distribution into a product of marginals,

$$p(\mathbf{z}_1, \mathbf{z}_2, .., \mathbf{z}_m) = p(\mathbf{z}_1) p(\mathbf{z}_2) .... p(\mathbf{z}_m). \tag{9}$$

Applying the independence of $\mathbf{z}_i$s' in (8) and using the linearity property of summation in $\mathbf{A}$,

$$\begin{aligned} &\mathbf{z}' \int_{\mathbf{z}_1} .. \int_{\mathbf{z}_m} \mathbf{A} p(\mathbf{z}_1, .., \mathbf{z}_m) d\mathbf{z}_1 .. d\mathbf{z}_m \\ &= \mathbf{z}' \sum_{i=1}^{m} \int_{\mathbf{z}_1} .. \int_{\mathbf{z}_m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right) p(\mathbf{z}_1), .., p(\mathbf{z}_m) d\mathbf{z}_1 .. d\mathbf{z}_m. \end{aligned} \tag{10}$$

For the ith KDE sample, all other variables except i get marginalized from (10), which results in

$$\mathbf{z}' \int_{\mathbf{z}_i} G_h\left(\mathbf{z}' - \mathbf{z}_i\right) p(\mathbf{z}_i) d\mathbf{z}_i. \tag{11}$$

Considering all the KDE samples in (10), we get,

$$\mathbf{z}' \sum_{i=1}^{m} \int_{\mathbf{z}_1} .. \int_{\mathbf{z}_m} G_h\left(\mathbf{z}' - \mathbf{z}_i\right) p(\mathbf{z}_1), .., p(\mathbf{z}_m) d\mathbf{z}_1 .. d\mathbf{z}_m = \mathbf{z}' \sum_{i=1}^{m} \int_{\mathbf{z}_i} G_h\left(\mathbf{z}' - \mathbf{z}_i\right) p(\mathbf{z}_i) d\mathbf{z}_i. \tag{12}$$

Likewise, applying the independence condition and linearity of summation in the RHS of (7), we get,

$$\frac{1}{h^2}\int_{\mathbf{z}_1}..\int_{\mathbf{z}_m}\mathbf{B}p(\mathbf{z}_1,\mathbf{z}_2,..,\mathbf{z}_m)d\mathbf{z}_1..d\mathbf{z}_m=\frac{1}{h^2}\sum_{i=1}^{m}\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)(\mathbf{z}'-\mathbf{z}_i)p(\mathbf{z}_i)d\mathbf{z}_i. \quad (13)$$

Thus, from (7) we get,

$$\mathbf{z}'\sum_{i=1}^{m}\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)p(\mathbf{z}_i)d\mathbf{z}_i=\frac{1}{h^2}\sum_{i=1}^{m}\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)(\mathbf{z}'-\mathbf{z}_i)p(\mathbf{z}_i)d\mathbf{z}_i. \quad (14)$$

Considering the expected value on both sides, we get

$$m\left[\mathbf{z}'\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)p(\mathbf{z}_i)d\mathbf{z}_i\right]=m\left[\frac{1}{h^2}\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)(\mathbf{z}'-\mathbf{z}_i)p(\mathbf{z}_i)d\mathbf{z}_i\right], \quad (15)$$

$$\mathbf{z}'\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)p(\mathbf{z}_i)d\mathbf{z}_i=\frac{1}{(h^2)}\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)(\mathbf{z}'-\mathbf{z}_i)p(\mathbf{z}_i)d\mathbf{z}_i. \quad (16)$$

We know in the LHS of (16),

$$\int_{\mathbf{z}_i}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)p(\mathbf{z}_i)d\mathbf{z}_i=G_h\big(\mathbf{z}'\big)\circledast p(\mathbf{z}'). \quad (17)$$

In the RHS of (16),

$$\frac{(\mathbf{z}'-\mathbf{z}_i)}{h^2}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)=-\nabla_{\mathbf{z}'}G_h\big(\mathbf{z}'-\mathbf{z}_i\big), \quad (18)$$

$$\int_{\mathbf{z}_i}\nabla_{\mathbf{z}'}G_h\big(\mathbf{z}'-\mathbf{z}_i\big)p(\mathbf{z}_i)d\mathbf{z}_i=-\nabla_{\mathbf{z}'}\big[G_h\big(\mathbf{z}'\big)\circledast p(\mathbf{z}')\big]. \quad (19)$$

Plugging the results in (16) gives,

$$\mathbf{z}'\big[G_h\big(\mathbf{z}'\big)\circledast p(\mathbf{z}')\big]=-\nabla_{\mathbf{z}'}\big[G_h\big(\mathbf{z}'\big)\circledast p(\mathbf{z}')\big], \quad (20)$$

$$\nabla_{\mathbf{z}'}Q(\mathbf{z}')=-\mathbf{z}'Q(\mathbf{z}'), \quad (21)$$

$$\text{where, } Q(\mathbf{z}')=\big[G_h\big(\mathbf{z}'\big)\circledast p(\mathbf{z}')\big]. \quad (22)$$

This is a partial differential equation and the solution for $Q(\mathbf{z}')$ is as follows:

$$Q(\mathbf{z}')=k\times\exp\big(-||\mathbf{z}'||^2/2\big), \quad (23)$$

$$G_h\big(\mathbf{z}'\big)\circledast p(\mathbf{z}')=G_1\big(\mathbf{z}'\big), \quad (24)$$

$$\text{This proves that, } p(\mathbf{z}')=G_\sigma\big(\mathbf{z}'\big), \quad (25)$$

$$\text{where, } \sigma^2=1-h^2. \quad (26)$$

□

Using the KDE-based JSD computation we have proved that the expected latent distribution is an isotropic Gaussian with variance $(1 - h^2)$, when the target is standard normal distribution. This demonstrates the value of the analytical approach used in formulation of the GEN. Proving such properties of the latent distribution is challenging for many other autoencoder-based generative models, such as AAEs and VAEs. AAEs asymptotically minimizes the JSD (in an implicit manner) between the encoded and target distribution, given an ideal discriminator with optimal parameter settings (Goodfellow et al., 2014). In practice, designing and training such a discriminator (e.g., with a finite architecture) is quite challenging, and the problems around design and convergence (as in GANs (Barnett, 2018; Kodali et al., 2017; Liu et al., 2017; Mescheder et al., 2018) require additional strategies that undermine the claims of the formal result. The formulation of the VAE proposes to match, via KL divergence, the *conditional* posterior (approximated by a Gaussian with mean and diagonal covariance) to the prior. Thus, the VAE formulation *makes no claims* about the aggregate/marginalized posterior, which represents encoded distribution—the topic of concern in this work. Indeed, without additional precautions (e.g. regularization) VAEs are prone to posterior collapse (Lucasz et al., 2019; Dai et al., 2020), and the measures (Oord et al., 2017; Razavi et al., 2019) typically used to alleviate that tendency further interfere with any rigorous claims about the structure of the final distribution in the latent space.

5 Results

To demonstrate the efficacy of GENs in different scenarios, we consider the AAE, VAE, WAE, and RAE as potential methods for comparison. We have used the WAE-MMD in all our experiments due to stable training. Although the RAE is not similar in principle to the regularization strategy adopted in the GEN or other autoencoder-based generative model, we consider this method worth studying as it has manifested its strength on multiple public data sets. For the RAE, we have used the L−2 regularization for simplicity and as no significant benefit was observed with the use of other complex regularization techniques (Ghosh et al., 2020). We do not use the Gaussian mixture model (GMM) as an ex post density estimator over the latent representation (adopted by the RAE to address the mismatch between the aggregate posterior and target distribution) in any of our experiments because we want to compare the baseline methods using the encoded distribution produced by their basic formulation. Here, we present experimental results that demonstrate the power of the KDE-based approach, relative to a neural network adversary, in computing the JSD. We also evaluate the generative capability of the methods, both qualitatively and quantitatively, on benchmark data sets. In addition, we study the entropy of the latent space distribution of the competing methods, which can be interpreted as a measure of matching the target distribution. To illustrate that the performance of GENs is not limited to any fixed target distribution, such as the standard normal, we evaluate the GEN with mixtures of Gaussians as the target distribution. Finally, we showcase the potential use of the proposed method for outlier detection.

5.1 Correlation study

Here, we compare the proposed KDE-based JSD computation with an adversarial (i.e., discriminator) neural network (NN). We create a set of experiments that control the degree of divergence between two distributions, the target distribution and hypothetical latent distribution, and quantify the correlation between the JSD and the parameter that controls these differences.

In this experiment, the target distribution is chosen to be standard normal, and we construct sequences of hypothetical latent distributions, a mixture of two standard normal distributions, separated by distance, s, that controls the distributions' divergence. In an ideal scenario, the computed JSD should increase with the parameter, s, which ranges from 0 to 6 in our study. For this particular experiment, we whiten all data to avoid trivial differences. The top of Fig. 1 shows examples from the hypothetical latent distributions (in 2D) with $s = [0, 3, 6]$. This arrangement allows us to study the behavior of the competing methods in estimating the difference between distributions over multiple trials, for different latent dimensions, l, and different training sample sizes, m.

The single parameter, the bandwidth, needed for estimating the JSD in GENs is computed with the optimization technique described in Sect. 3. For the NN, we train a discriminator with labeled instances from the two distributions, target distribution (standard normal distribution) and hypothetical latent distribution (mixtures of Gaussians). For a given latent dimensions, l, and training sample size, m, the NN-discriminator architecture is chosen from a pool of 10 architectures, varying in capacity, with the minimum validation loss. The baseline model has three hidden layers, with the number of units at each layer proportional to l, and the first layer being the widest, with $10 \times l$ units, followed by $5 \times l$, and $2 \times l$ units. For variation in the model capacity, more hidden layers are added, the width of the existing hidden layers is increased or a combination of both is applied. Details of the NN architectures can be found in "Appendix B". In the search for the best architecture for a given l and m, all the models are trained for 50 epochs, and to avoid overfitting the learning rate of the Adam optimizer (initialized at $1e-03$) is reduced by 0.5, if the validation loss did not improve for 5 consecutive epochs. Similar hyperparameter settings are also used for training the best neural network architecture.

Correlation of the JSD with divergence, s, for the NN and KDE-based method is determined by computing the JSD over 10 trials for a given s. Using the correlation study, we can quantify the slope for each method as a function of dimension and sample size. The Pearson correlations are reported in Table 2 over a range of dimensions and sample sizes. In each cell, the NN | KDE are on the left | right, respectively. Correlations that are statistically significant, i.e., a correlation coefficient with p-value less than 0.05, are represented in regular font with the higher number indicated in bold, and numbers in italics have either no statistical significance (considering p-value) or have negative correlation or both. For cells with single significant correlation value, no comparison is done. For lower dimensions, $l = [2, 5]$, the behaviors of both methods are consistent, even with a low sample count, but the response of the NN is observed to be highly unstable for dimensions $l >= 10$. This is also manifested in the line graph of the estimated JSD as a function of s for $l = 20$ and $m = 8000$, with the error bars produced with 10 trials, in the bottom of Fig. 1. Here, the correlation for the proposed KDE-based approach is way higher relative to the NN, which is almost zero. Consistent increase in the JSD with increase in separation s between the mixtures (positive slope) for the KDE (orange curve) explains the high correlation coefficient of 0.83 of the KDE-based JSD estimation for $l = 20$ and $m = 8000$

in Table 2. Similarly, the overall flat curve for the NN (in blue) justifies a low correlation coefficient of 0.05, where there is almost no change in the JSD with distance between the mixtures. These slopes and the corresponding error bars are important, because the derivatives of the computed JSD drive the structure of the latent space in this autoencoder context. These results also demonstrate that the estimation of the JSD using the NN is strongly influenced by the sample size and data dimension, unlike the robust behavior of the KDE, which can produce good results even with a low sample size. In most of the scenarios reported in Table 2, the KDE has a high positive correlation coefficient. The correlation study helped us in choosing the number of KDE samples to be used in GENs when we fix the latent dimension of the autoencoder. This experiment illustrates that, in general, the KDE-based JSD computation gives us a better estimate of the difference between distributions, relative to the NN counterpart.

5.2 Results on bench-mark data

We examine qualitative results for several benchmark data sets: MNIST (LeCun et al., 2010), SVHN (Netzer et al., 2011), and CelebA (Liu et al., 2015) with the standard normal as the target distribution in the latent space. As the proposed method is an autoencoder-based generative model, it is important to evaluate the generative capability. In this experiment, we have used the widely known Fréchet Inception Distance (FID) (Heusel et al., 2017) for estimating the quality of the generated samples. The lower the FID score, the better the modelling of the data distribution. The FID score is known to be robust to noises and rewards variability in the generated examples (Heusel et al., 2017; Lucic et al., 2017), unlike the Inception Score (IS) (Salimans et al., 2016), an alternate technique to evaluate generated samples. Performance of the GEN is compared with the AAE, VAE, WAE, and RAE. The numbers of the KDE samples used in GENs for the MNIST, SVHN and CelebA data sets are $10K$, $10K$, and $20K$, respectively. The dimension of the bottleneck layer, l, used for the MNIST, and SVHN data sets are 8 (Makhzani et al., 2016), and 35 (Makhzani et al., 2016), respectively, for all the methods. The latent dimension for the SVHN data set is increased to 35 from 20, as used in Makhzani et al. (2016), because of better reconstruction. The latent dimension l, used for the CelebA is 64 (Ghosh et al., 2020) for all the methods except the GEN. For

Table 2 The Pearson correlation between the JSD estimates and varying degrees of separation, s, between the distributions computed with data gathered over multiple trials for the NN (left of bars) and KDE for different settings of latent dimension, l and sample size, m

$l \backslash m$	500	1000	2000	4000	8000	10000
2	0.64 | **0.91**	0.84 | **0.91**	0.90 | **0.93**	0.85 | **0.92**	0.91 | **0.93**	0.80 | **0.93**
5	0.42 | **0.88**	0.72 | **0.90**	0.87 | **0.93**	0.91 | **0.93**	0.91 | **0.93**	0.91 | **0.93**
10	*0.03* | 0.54	0.28 | **0.82**	0.56 | **0.88**	0.84 | **0.91**	0.87 | **0.92**	**0.93** | *0.91*
20	*−0.02* | *0.06*	*−0.27* | *0.18*	*0.01* | *0.28*	*0.05* | 0.48	*0.05* | 0.83	*−0.27* | 0.84
30	*0.06* | *0.17*	*−0.26* | *−0.11*	*0.02* | *0.12*	0.32 | *−0.09*	*−0.45* | *0.06*	*−0.09* | 0.40
40	*0.13* | *−0.08*	*−0.31* | *−0.08*	*−0.45* | *−0.16*	*0.19* | *−0.46*	*−0.09* | *0.05*	*0.22* | *−0.16*

The rows and columns of the table span across different latent dimensions and sample sizes, respectively. Bold indicates significant correlations that are higher than the alternative model

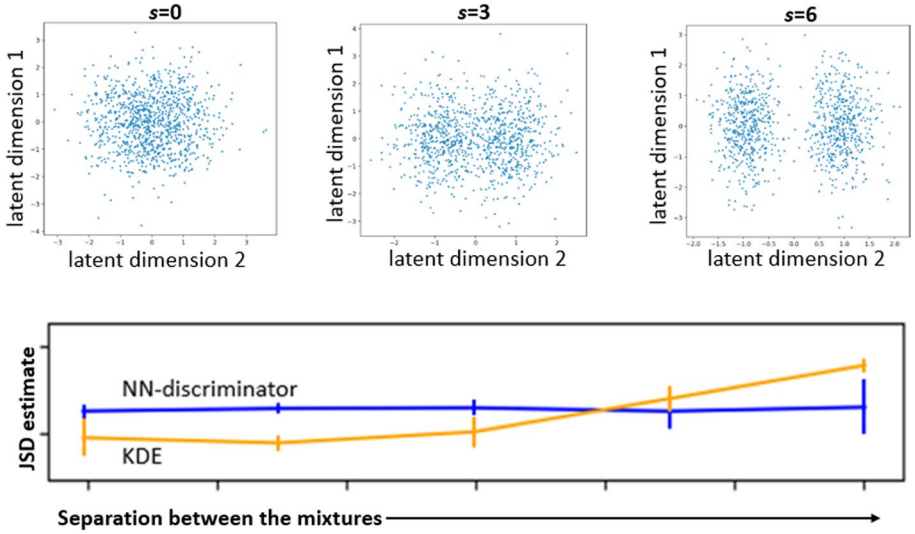

Fig. 1 Top: Sequences of hypothetical latent distributions (in 2D) with $s = [0, 3, 6]$, all with the same variance, that differ in controlled ways from the target distribution (standard normal) allow us to evaluate a method's ability to quantify these differences. Bottom: A line graph with error bars shows how the JSD estimated using a neural network (NN) and using a KDE-based approach varies with different degrees of separation in the hypothetical latent distributions for $l = 20$ dimensions and $m = 8000$ samples. The JSD estimated using the KDE increases consistently with the separation between the mixtures, unlike the NN

latent dimensions $l >= 50$, the bandwidth estimation method produces $h > 1$, which will degenerate solutions in GENs. This represents a limitation in using a single fixed bandwidth for the KDE in very high dimensions—plausible solutions are beyond the scope of this paper, but are discussed in Sect. 6.

In all these methods except the RAE, the latent distribution is matched to a target distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$. In autoencoder-based generative methods, it is well known that there is a trade-off between the reconstruction quality and the regularizing effect in the latent space. Therefore, to compare the performance of different methods, it is important that the reconstruction quality is more or less similar for all the methods. To this end, we have reported the FID score of the reconstructed held-out data for all the benchmark data sets. It is also important to understand that how smooth is the structure in the latent space with different regularization techniques. To address this issue, we have reported the FID score of the interpolated samples generated by spherical interpolation between the encoding of random pairs of test data. FID scores of the reconstructed, interpolated and generated samples are computed using $10K$ samples for all the competing methods.

For fair comparison, the architecture of the autoencoder (similar to Ghosh et al. (2020)) is same for a particular data set for all the competing methods. Network architectures for all the data sets and details related to training are reported in "Appendix C". We know that the latent distribution for the GEN on convergence is $\mathcal{N}\big(\mathbf{0}, \mathbf{I}(1 - h^2)\big)$ from Theorem 1, where h is the bandwidth used in training GENs. So, the generative capability of the GEN is evaluated using samples drawn from $\mathcal{N}\big(\mathbf{0}, \mathbf{I}(1 - h^2)\big)$. Since the latent representation of the RAE is not matched to any target distribution, samples are generated from a normal distribution whose parameters are estimated using the embedding of the training data. Quantitative analysis of different methods is reported in Table 3. For generated and interpolated

examples, the FID score is the average over 5 different sets. The FID score (in Table 3) of the best performing method for all the data sets is highlighted in bold. Scenarios, when GENs is the second best method, FID scores are reported in italics. As expected, we can see that the FID score of the reconstructed held-out data is consistently lower than the FID score of the generated samples for all the methods across different data sets. For the MNIST data set, the FID score of the reconstructed samples is the minimum for the GEN and is the next best (shown in italics), both for samples from the target distribution and interpolated examples, following the AAE. For the SVHN, it closely follows the best performing method, RAE, for reconstructed and generated samples. For the CelebA, GENs has the best FID score for the generated samples, even with $l = 40$ dimension latent space, and the second best for the reconstructed examples. With the use of variable bandwidth KDE, to be studied in the future scope of this work, we hope to have better results when the latent space of GENs can be scaled even higher. Overall, FID scores of the GEN for multiple data sets are consistently close to the best performing method. Important thing to observe here is that the performance of GENs is achieved without extensive search of the hyperparameter settings. For qualitative evaluation, the reconstructed, generated, and interpolated images produced by different methods for the CelebA data set are shown in Fig. 2.

5.3 Evaluating structure in learned latent spaces

Here we quantify how well-formed the Gaussian structure is in the latent space after the autoencoder is optimized. We rely on the fact that for a given covariance, $\mathbf{\Sigma}$, the Gaussian has the maximum entropy among all distributions with the same covariance. Thus, we can compute the entropy, $\mathrm{H}\big[p_e\big]$, of the whitened latent distributions from various learned models to evaluate the *normality* of the encoded distribution for different methods. Evaluating entropy relies on estimates of the density. For this, we also use the KDE, as defined in (3). For a certain latent dimension l, and sample size m, the optimal bandwidth, h_{opt}, can be determined by minimizing the entropy of the KDE estimate of a distribution given samples from it (maximum log likelihood). This is in contrast to the estimation of the bandwidth, using the JSD for GENs, which best differentiates samples from two distributions, encoded and target.

To find h_{opt} that minimizes the entropy of a set of latent space samples, we differentiate $\mathrm{H}\big[p_e\big]$ with respect to h, set it equal to zero, and solve for h_{opt} with a fixed point strategy. We have,

$$\mathrm{H}\big[p_e(\mathbf{z})\big] \approx -\frac{1}{m}\sum_i \log \frac{1}{m-1}\sum_{j\neq i} K\left(\frac{||\mathbf{z}_j - \mathbf{z}_i||}{h}\right), \tag{27}$$

which gives the fixed point update:

$$h^2 \leftarrow \frac{1}{lm}\sum_i \frac{\sum_{j\neq i} ||\mathbf{z}_j - \mathbf{z}_i||_2^2 G_h\big(\mathbf{z}_j - \mathbf{z}_i\big)}{\sum_{j\neq i} G_h\big(\mathbf{z}_j - \mathbf{z}_i\big)}. \tag{28}$$

We evaluate the entropy of the latent distributions of all the competing methods for varying training sample sizes, ranging from 2000 through 20000, so as to study the effect of the data size in matching distributions. If the latent representation closely matches the target, standard normal distribution, then its entropy will be similar to that of the standard normal for that dimension, which has an analytical expression. This experiment is conducted on the MNIST data set with

Table 3 FID scores of the competing methods for different data sets (lower is better)

	MNIST			SVHN			CelebA		
	Rec.	Samples		Rec.	Samples		Rec.	Samples	
		$\mathcal{N}$	Int.		$\mathcal{N}$	Int.		$\mathcal{N}$	Int.
AAE	16.69	**17.36 ± 0.13**	**17.46 ± 0.14**	41.92	47.00 ± 0.54	46.65 ± 0.23	43.56	49.08 ± 0.2	47.32 ± 0.14
VAE	19.81	23.86 ± 0.20	21.52 ± 0.17	39.21	52.65 ± 0.37	46.39 ± 0.10	**38.57**	48.79 ± 0.08	43.22 ± 0.13
WAE	16.98	20.98 ± 0.24	20.33 ± 0.13	44.57	56.63 ± 0.32	59.79 ± 0.27	40.58	48.39 ± 0.27	39.77 ± 0.15
RAE	16.15	24.32 ± 0.11	23.44 ± 0.11	**33.82**	**37.40 ± 0.12**	**44.5 ± 0.23**	42.05	49.67 ± 0.26	**39.30 ± 0.05**
GEN	**15.76**	*18.16 ± 0.10*	*19.12 ± 0.16*	*37.02*	*42.00 ± 0.36*	47.86 ± 0.25	*40.32*	**45.37 ± 0.29**	44.52 ± 0.03

The FID score of the best performing method is highlighted in bold. Scenarios, where GENs is the second best method, FID scores are reported in italics

Fig. 2 **a** Reconstructed **b** generated and **c** interpolated images for the CelebA data set for different methods. For interpolation results, images in the first and last column represent ground truth (GT)

latent dimension of $l = 8$. For fair comparison, the architecture of the E_ϕ and D_θ used is same for all methods, defined in "Appendix C". A part of the training budget is used for the KDE estimate in GENs. For this experiment, 50% of the training data is used as KDE samples for GENs. Say for training data size of 2000, the GEN uses 1000 samples for the KDE estimate and the remaining 1000 samples to update model parameters. Whereas, all other methods use the complete training data (2000 examples in this case) to update model parameters. To rule out the confounding factor of the reconstruction loss in the study of the entropy, we have ensured that the reconstruction loss is similar for all the methods for different training data sizes by determining the correct scaling factor of the regularizer. However, the reconstruction loss of the RAE is less than other methods and close to that of an unconstrained autoencoder, as the regularizations used in this method never attempts to impair the performance of an unconstrained autoencoder, unlike other methods, which match the latent distribution to the target distribution. For the RAE, we have reported results for regularization scalars that gave the best entropy.

The steps involved in this experiment are summarized as follows: train models using the training data; generate the latent encoding of the evaluation data; determine the optimal bandwidth (h_{opt}) using the latent encoding of the evaluation data (post whitening); compute the entropy of the whitened latent encoding using the optimal bandwidth. The mean entropy with the error estimate and the reconstruction loss (in parentheses) is reported in Table 4. Results are computed using the test data (10K samples) of the MNIST data set. For statistical evaluation, we have trained 5 different models of each method for a given sample size. Additional experimental details are available in the "Appendix C". Numerical estimates of the entropy of standard-normal distribution reported in Table 4 are very close to the true entropy of the standard normal distribution (11.35 for $l = 8$), which validates our implementation. In this experiment, the performance of the RAE is surpassed by majority of the competing methods, in most of the scenarios. This is due to the lack of regularizer, imposing distribution matching in the

Table 4 Mean entropy of the latent representations built by the AAE , VAE , WAE, RAE, and GEN for the MNIST data set (using latent dimension of $l = 8$) along with standard deviation and reconstruction loss (in parentheses)

*Method**Samples*	2000	4000	10000	20000
AAE	$9.741 \pm 0.210(0.141)$	$9.803 \pm 0.145(0.114)$	$10.239 \pm 0.107(0.124)$	$10.439 \pm 0.061(0.101)$
VAE	$9.066 \pm 0.073(0.141)$	$9.710 \pm 0.075(0.114)$	$9.897 \pm 0.071(0.105)$	$10.080 \pm 0.133(0.096)$
WAE	$10.40 \pm 0.047(0.130)$	$10.503 \pm 0.032(0.115)$	$10.610 \pm 0.032(0.103)$	$10.687 \pm 0.062(0.096)$
RAE	$9.743 \pm 0.031(0.093)$	$9.781 \pm 0.069(0.079)$	$9.621 \pm 0.029(0.071)$	$9.250 \pm 0.045(0.063)$
GEN	$\mathbf{10.500 \pm 0.027(0.139)}$	$\mathbf{10.840 \pm 0.031(0.123)}$	$\mathbf{11.046 \pm 0.020(0.109)}$	$\mathbf{11.097 \pm 0.022(0.099)}$
Standard Normal	11.668	11.621	11.575	11.538

latent space, in the loss function of the RAE. Poor performance of the VAE in matching the aggregate posterior to the prior can be attributed to *holes* or *pockets* in the encoded distribution Rosca et al. (2018). The performance of the AAE and WAE scales up with the training sample sizes and both these methods are close to the performance of GENs. However, the strategy of matching moments of the distributions, as used in the WAE-MMD, is found to be more effective than the adversarial training in the AAE. Among all the competing methods, the performance of GENs in matching the encoded distribution to the standard normal is the best and very close to the entropy of the standard normal for the training sample size >= 10000.

5.4 Mixtures of Gaussians

So far in all our experiments, the standard normal distribution was considered as the target distribution. Nevertheless, formulation of GENs can be extended to other target distributions, such as mixtures of Gaussians (MoG). In this experiment, we consider a MoG with 10 modes as the target distribution in two-dimensional latent space, $l = 2$. Similar to previous experiments, we need to compute the KDE bandwidth by maximizing the JSD between two sample sets from the MoG. Number of KDE samples used in this experiment is $m = 10K$. Two experiments are devised over the MNIST data set, one in unsupervised framework and the other in semi-supervised framework. In the unsupervised setup, GENs match the latent distribution to MoG as shown in Fig. 3a. Here we see the projection of held out data ($10K$ samples) in the latent space. We can observe that though GENs could successfully match the distributions in the latent space, different classes in the MNIST data set are not matched distinctly to specific modes in the MoG, which is obvious. However, if we consider some labeled data for training the GEN, input data can be mapped to specific modes of the distribution (depending on its category) shown in Fig. 3b. We call this the semi-supervised setup. In the semi-supervised experiment, out of the total training data we use labels for $10K$ samples. This experiment clearly demonstrates that GENs can be used for matching any target distribution. Use of the MoG as the target distribution for the GEN in higher latent dimension, l, for richer data representation will be considered in the future scope of this work. The autoencoder architecture used in this experiment is the one used in Sect. 5.2 for the MNIST data, defined in "Appendix C".

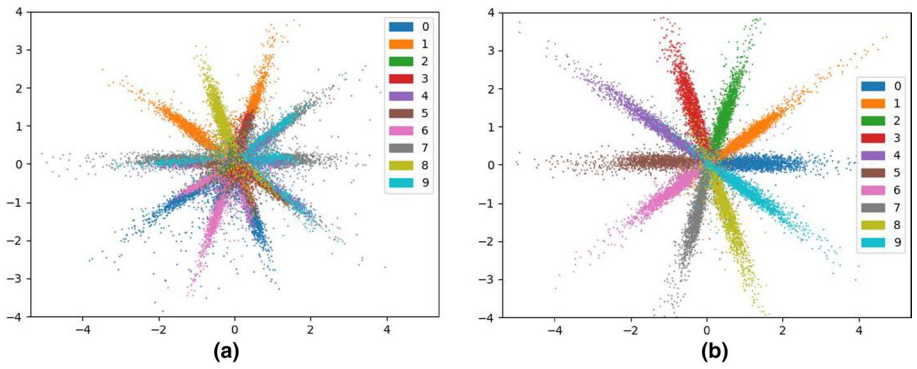

Fig. 3 Mapping of the MNIST held-out data to 10 2D Gaussians in the latent space **a** without any supervision **b** with labels for $10K$ training examples

5.5 Anomaly detection

For the autoencoder-based generative model, a distribution is imposed in the latent space to add generative capability to the model. So, for all the methods studied in this work it would be interesting to see whether the latent distribution can be used for some discriminative task, for example detecting unusual or rare events (aka outliers). This is based on the assumption that the prevalent samples in the data distribution are mapped to the high probability region of the latent distribution whereas, outliers representing rare examples from the data distribution or examples from a different data distribution are mapped to the low probability region. Provided the latent distribution closely matches the target distribution, this approach of detecting outliers in the unsupervised framework is very effective as estimation of distribution in data space is usually not easy. Most of the outlier detection methods, using the encoder-decoder architecture, rely on the reconstruction loss (Xia et al., 2015; Xu et al., 2021) or employs a one-class classifier (Sabokrou et al., 2018) where the classifier is trained with samples from the inlier class which subsequently helps in detecting less frequent outlier examples. Detection of outliers using the density in the latent space was proposed in the deep autoencoding Gaussian mixture model (DAGMM) due to Zong et al. (2018). In this method, density is estimated as mixtures of Gaussians and samples with low probability are identified as outliers. Likewise, anomaly detection with density estimated by the energy based model using a denoising autoencoder, deep structured energy based models for anomaly detection (DSEBM) (Zhai et al., 2016), is another density-based outlier detection method. Generative probabilistic novelty detection with adversarial autoencoders (GPND) (Pidhorskyi et al., 2018) is an outlier detection method which relies on the probability of samples on the data manifold learned by autoencoder in observed space, where the score is computed using in-manifold and out-of-manifold probability.

Comparative study of the use of the latent space of the autoencoder-based generative models for outlier detection can be considered another way of interpreting how well-formed the latent structure is. For our study, Mahalanobis distance is used for computing the outlier score where the mean and covariance of the distribution in the latent space is computed from the training data. Performance metric used in this experiment are the F_1 score and area under the ROC curve (AUC). The AUC score summarizes the model performance for different classification thresholds. For both the F_1 and AUC score, higher is better. This study is focussed on two public data sets, the MNIST (LeCun et al., 2010), and NSL-KDD (Xu et al., 2021).

5.5.1 MNIST

For this data set, we leverage the labeling of the data to identify inlier and outlier examples. In this setup, a class of digit is selected as inlier and examples from that class in the training data are used to train the model. For outlier detection evaluation, a test set is constructed with samples from the inlier class in test data mixed with random samples from other classes, representing outliers, with anomaly proportion, ρ, ranging from 0.1 through 1.0 with a step size of 0.1, $\rho = [0.1 : 1.0;0.1]$. In this experiment no noise is considered in the training data, such as adding examples from a class other than the inlier class. After the models are trained, threshold scores to detect outliers are determined by applying the percentile rule on the outlier scores of the training examples. For this experiment, 90^{th} percentile of the outlier scores of the training examples is used as the threshold for computing the F_1 scores reported in Table 5.AUC scores reported in Table 6 aggregate

the model performance for the complete range of thresholds (based on percentile). Since the test set contains outlier samples chosen randomly from the MNIST classes leaving out the inlier class, an average of the F_1 and AUC score for 10 different evaluation sets is reported in Tables 5 and 6, respectively. For uncluttered representation, F_1 and AUC scores of all methods are reported for $\rho = 0.2$ and $\rho = 1.0$ on the left | right, respectively. The F_1 and AUC score of the best performing method for each MNIST class in Tables 5 and 6 is reported in bold and scenarios where the GEN is the second best method are indicated in italics. To evaluate the performance of the AAE, VAE, WAE, RAE, and GEN as outlier detector we have reported the results for the DSEBM, a specially designed outlier detection algorithm. Out of the two proposed metrics, DSEBM-e and DSEBM-r, we have considered the DSEBM-e in our experiment due to better accuracy in its predictions. The encoder architecture used in the DSEBM-e is built following instructions in Zhai et al. (2016). The autoencoder architecture used for all methods in this experiment is defined in "Appendix C".

Both in Tables 5 and 6, it is observed that for most of the MNIST classes, the GEN outperforms other methods or closely follows the best performing method, both for low and high anomaly proportions. Overall, the performance of GENs is better than other autoencoder-based generataive methods, and is better or comparable to the specially built outlier detection algorithm, DSEBM-e. This is also observed in Fig. 4a, which plots mean F_1 scores marginalized over all MNIST classes for different anomaly proportions, ρ. F_1 scores of the RAE and VAE closely follow GENs. Likewise, the F_1 score for different MNIST classes, representing average over different anomaly proportions, ρ, can be found in Fig. 4b. The GEN is observed to be among the top performers for all the MNIST classes. The bar plot manifests that the DSEBM-e is doing consistently good for all the MNIST classes except digit 0, which is also observed in Table 5. This experiment of classifying inliers from outliers clearly demonstrates that the latent space built by the GEN, in an unsupervised setup, contains meaningful information that can be used for downstream tasks.

5.5.2 NSL-KDD

The KDDCup99 data set (Cup 1999) is a widely used data set used to build computer network intrusion detection systems which identifies the abnormal (aka outlier) samples from the normal (or inlier) data. However, this data set has several redundant entries both in the

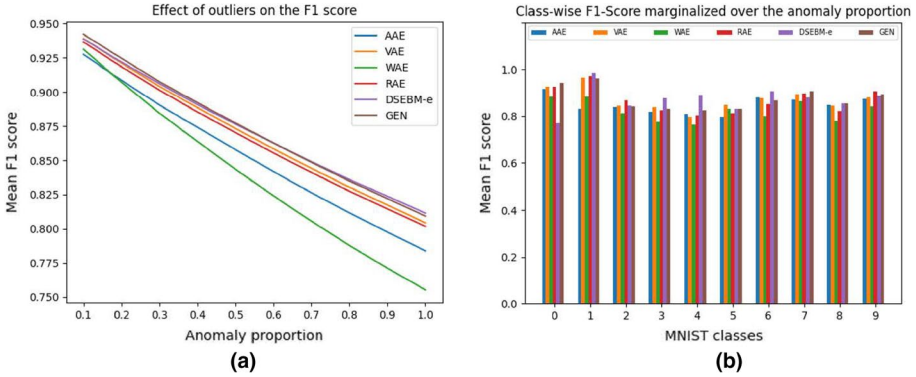

Fig. 4 Mean F_1 scores **a** for different anomaly proportions marginalized over all the MNIST classes **b** for the MNIST classes marginalized over the anomaly proportions, ρ

Table 5 Mean F_1 scores of the competing methods for different data classes in the MNIST data set for $\rho = 0.2$ and $\rho = 1.0$ on the left right, respectively, evaluated over 10 different test sets

	AAE	VAE	WAE	RAE	DSEBM-e	GEN
Zero	0.929 | 0.875	0.934 | 0.884	0.919 | 0.815	0.939 | 0.890	0.891 | 0.714	**0.950** | **0.912**
One	0.904 | 0.763	0.955 | 0.943	0.926 | 0.833	0.949 | 0.940	0.950 | **0.950**	**0.956** | *0.943*
Two	0.902 | 0.765	0.911 | 0.754	0.901 | 0.733	**0.919** | **0.790**	0.907 | 0.762	0.904 | 0.736
Three	0.909 | 0.748	0.914 | 0.752	0.900 | 0.707	0.912 | 0.742	**0.934** | **0.820**	*0.917* | *0.753*
Four	0.902 | 0.747	0.896 | 0.713	0.892 | 0.703	0.901 | 0.726	**0.930** | **0.833**	*0.914* | *0.749*
Five	0.894 | 0.725	**0.914** | **0.775**	0.905 | 0.759	0.898 | 0.737	0.906 | 0.744	*0.913* | *0.762*
Six	0.902 | 0.815	0.917 | 0.804	0.896 | 0.717	0.901 | 0.772	**0.932** | **0.853**	*0.918* | 0.801
Seven	0.916 | 0.816	0.927 | 0.844	0.921 | 0.796	**0.928** | 0.846	0.914 | 0.826	**0.928** | **0.858**
Eight	0.914 | 0.772	0.917 | 0.758	0.902 | 0.709	0.901 | 0.720	**0.924** | **0.783**	0.916 | **0.746**
Nine	0.917 | 0.814	0.927 | 0.818	0.913 | 0.779	**0.938** | **0.856**	0.933 | 0.831	0.931 | *0.834*

The F_1 score of the best performing method for each MNIST digit is reported in bold, and scenarios where the GEN is the second best method, F_1 scores are indicated in italics

Table 6 Mean AUC scores of the competing methods for different data classes in the MNIST data set for $\rho = 0.2$ and $\rho = 1.0$ on the left right, respectively, evaluated over 10 different test sets

	AAE	VAE	WAE	RAE	DSEBM-e	GEN
Zero	0.943 | 0.942	0.951 | 0.955	0.901 | 0.901	0.953 | 0.956	0.655 | 0.657	**0.968** | **0.970**
One	0.803 | 0.806	0.987 | 0.987	0.870 | 0.876	0.989 | 0.989	**0.997** | **0.997**	0.987 | 0.986
Two	0.827 | 0.827	0.840 | 0.837	0.780 | 0.781	**0.874** | **0.872**	0.824 | 0.829	*0.844* | *0.844*
Three	0.805 | 0.805	0.842 | 0.840	0.708 | 0.707	0.801 | 0.801	**0.900** | **0.901**	0.827 | 0.826
Four	0.748 | 0.757	0.741 | 0.748	0.669 | 0.673	0.754 | 0.760	**0.908** | **0.913**	*0.803* | *0.808*
Five	0.730 | 0.728	**0.832** | **0.832**	0.793 | 0.796	0.743 | 0.746	0.822 | 0.825	0.820 | 0.820
Six	0.886 | 0.887	0.883 | 0.887	0.743 | 0.749	0.830 | 0.840	**0.931** | **0.934**	0.874 | 0.881
Seven	0.890 | 0.885	0.917 | 0.918	0.868 | 0.866	**0.929** | **0.928**	0.893 | 0.895	*0.925* | *0.925*
Eight	0.843 | 0.846	0.830 | 0.829	0.712 | 0.713	0.796 | 0.795	**0.852** | **0.851**	*0.848* | *0.848*
Nine	0.881 | 0.878	0.890 | 0.887	0.830 | 0.826	**0.915** | **0.913**	0.901 | 0.900	*0.912* | **0.913**

The AUC score of the best performing method for each MNIST digit is reported in bold, and scenarios where the GEN is the second best method, AUC scores are indicated in italics

train and test set which makes the outlier detection algorithm biased towards learning the frequent class. Moreover, owing to the large size of the training data, different methods split the training data into training and evaluation set which has strong ramifications on fair comparison of different learning algorithms. These limitation were first studied in details by (Tavallaee et al., 2009) and they proposed a new data set, NSL-KDD (Tavallaee et al., 2009; Xu et al., 2021) by pruning the redundant entries from the train and test set of the original KDDCup99 data set (Cup 1999). Reasonable size of the new train and test set encourages the use of the complete set for training and evaluation of the outlier detection algorithms, ensuring a fair and consistent comparison.

The length of each entry in the NSL-KDD data set is 41 and has both continuous and categorical variables. One-hot vector representation of the categorical data along with the continuous variables produces a vector of 122 dimensions. It should be noted that extreme value of any feature in the training set has negative impact on the performance of the outlier detection algorithms. Hence, it is reasonable to pre-process the training data by removing such entries. Out of different data pre-processing techniques, such as z-score, inter-quartile range, median absolute deviation (MAD) (Leys et al., 2013), and percentile rule (Xu et al., 2021), we have used the percentile rule in this experiment, typical set to 98^{th} percentile. Using this pre-processing scheme, any entry in the training data is dropped if any one of the feature value is greater than the 98^{th} percentile of that feature. The training data is then normalized to the range [0, 1] using the minimum and maximum value of individual features.

From the train data, only normal samples are used for training the network parameters and samples from the test data, containing both normal and attack examples, are used for model evaluation using the F_1 and AUC score. The autoencoder architecture used for the AAE, VAE, WAE, RAE and GEN is defined in "Appendix C" along with other experimental details. The compression network used in the DAGMM (Zong et al., 2018) is the same as in GENs, and the estimation network follows the configuration for the KDDCUP99 10 percent data set (Lichman 2013) defined in (Zong et al., 2018). The encoder used in the DSEBM (Zhai et al., 2016) is the same as used in the GEN. In this experiment, we have studied the performance of the models for different latent dimensions, $l = 2, 5$, and 10. After the models are trained, threshold scores to detect outliers are determined by applying the percentile rule on the outlier scores of the training examples. For this experiment, typically 90^{th} percentile of the outlier scores of the training examples is used as the threshold for computing F_1 scores reported in Table 7. AUC scores of the competing methods for different latent dimensions are reported in Table 8.

For F_1 and AUC scores reported in Tables 7 and 8, respectively, it is observed that the performance of the autoencoder-based generative models improves with the size of the latent dimension, l. However, the specially built methods for outlier detection, DAGMM (Zong et al., 2018) and DSEBM (Zhai et al., 2016), are indifferent to the size of the latent dimension, l. Based on the F_1 scores reported in Table 7, GENs outperform all other methods for latent dimension, $l = 10$, and for lower latent dimensions, $l = 2$, and 5, the GEN is observed to be better than all other autoencoder-based generative models. Considering the

Table 7 F_1 scores of the competing methods for different latent dimensions, l

l	AAE	VAE	WAE	RAE	DSEBM-e	DAGMM	GEN
2	0.680	0.624	0.661	0.601	**0.916**	0.915	0.805
5	0.850	0.819	0.760	0.860	0.912	**0.913**	0.870
10	0.835	0.909	0.810	0.917	0.914	0.915	**0.924**

Table 8 AUC scores of the competing methods for different latent dimensions, l

l	AAE	VAE	WAE	RAE	DSEBM-e	DAGMM	GEN
2	0.777	0.722	0.756	0.754	0.908	**0.933**	0.856
5	0.874	0.897	0.846	0.911	0.904	**0.922**	**0.922**
10	0.872	0.936	0.849	0.922	0.915	0.920	**0.939**

AUC scores reported in Table 8, which is invariant to the choice of thresholds, GENs is the best performing outlier detection method for the latent dimension, $l = 10$. GENs jointly share the best AUC score for $l = 5$ with DAGMM. This experiment manifests the potential use of the latent representation produced by GENs, without any supervision, for solving complex real-world challenges, such as intrusion detection.

6 Discussion

In this paper, we propose an autoencoder-based generative model, the GEN, where, a KDE-based JSD estimate is used to match the encoded distribution to the target, defined *a priori*. An important property of GENs is that, for the standard normal as the target distribution, the expected encoded distribution, on convergence, is an isotropic Gaussian distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I}(1 - h^2))$. This property can be generalized to any multivariate Gaussian distribution parameterized by (μ, Σ), as the target. Use of the KDE in estimating the encoded distribution facilitates us in deriving such provable property of the latent structure, unlike other autoencoder-based generative model.

Through extensive experimental evaluation in the study of the correlation of the JSD with divergence, we could show that the proposed KDE-based JSD estimate increases with the increase in difference between the distributions. Proper estimate of the JSD is important for structuring the latent space such that the encoded distribution closely matches the target. Overall, the performance of the GEN was observed to be as or more accurate than the NN counterpart. This is also validated by the results of the entropy study. In this experiment, the entropy of GENs is observed to be consistently higher than all other competing methods in different scenarios and close to the entropy of the standard normal distribution (computed analytically). This indicates close matching of the encoded distribution to the target, which could be attributed to the better estimate of the JSD in GENs. Evaluation of the generative capability of GENs using the FID score, a way of assessing the quality of the generated samples, on multiple public data sets, clearly showcases the strength of GENs to be used as a generative model. Performance of the GEN is found to be better than or comparable to the SOTA autoencoder-based generative models.

In this paper, we have also demonstrated the practical use of the latent representation produced by GENs for unsupervised anomaly detection on mutiple public data sets. In the outlier detection experiment on the MNIST data set, GENs is observed to be superior to other autoencoder-based generative models and comparable to the method designed specifically for outlier detection. However, for the NSL-KDD data set, a challenging data set for modeling intrusion detection systems, GENs outperformed the specially built methods for outlier detection. This illustrates that the feature space produced by the GEN is effective for other downstream discriminative tasks. All these experimental results, combined with the relatively few design choices of the method (number of training samples used for the KDE estimate and the weight on the regularizer) bodes well for the practicality of the proposed method. So far, we

have discussed experimental results considering the standard normal distribution as the target. Nevertheless, we show that the capability of GENs is not limited to any fixed target distribution and can be extended to other complex distributions, such as mixtures of Gaussians.

One important challenge for GENs is the bandwidth estimation for higher latent dimensions, $l >= 50$. This limits the scalability of GENs to higher latent dimensions. However, there are methods in the statistics literature to address this issue (Silverman 1986), but their practicality and efficacy in this context remains a topic for future development. Formulation of the GEN for any target distribution offer a promising next step for development. Use of the latent space constructed by GENs, on convergence, for more downstream discriminative tasks, is an interesting direction for future research.

Appendix A: Summary of the pros and cons of the generative models

Table 9 summarizes the pros and cons of different autoencoder-based generative models related to the GEN.

Appendix B: Architecture for correlation study

Different neural network architectures used in the correlation study, discussed in Sect. 5.1, are defined in Table 10.

Appendix C: Architecture for bench-mark data

The autoencoder architectures used in all methods for the MNIST, SVHN, CelebA, and NSL-KDD data set are defined in Table 11, 12, 13, and 14, respectively. In the neural network architectures, Conv n and TransConv n defines convolution and transpose convolution operation, respectively, with n filters in the output. We have used 5×5 filters for all data sets. Stride size of 2 is used for transpose convolution expect the last layer of the decoder. Fully connected layers are represented as $\texttt{FC}_{\texttt{n}}$ with n nodes. Activation functions used in the networks are LReLU, represents Leaky ReLU (rectified linear units), and Tanh, hyperbolic tangent function. Architecture of the discriminator used in the AAE for the MNIST, SVHN, CelebA, and NSL-KDD data sets are defined in Tables 15, 16, 17, and 18, respectively.

Input is mapped to the range $[-1, 1]$ for the MNIST, SVHN and CelebA data sets. For the NSL-KDD data set, the input is mapped to the range [0, 1]. Adam optimizer is used in all experiments with learning rate of $2e-04$. All methods are trained for a maximum of 100, 100, 70, and 200 epochs for the MNIST, SVHN, CelebA, and NSL-KDD data set, respectively, with a minibatch size of 64. However, for the entropy study of the latent space minibatch size is 25 for 2000 samples and 50 for other sample sizes. Batch normalization is not used in our implementations.

For the CelebA data set, the latent dimension size is $l = 64$ (Ghosh et al. 2020) for all the methods, except the GEN. The dimension of latent space is set to $l = 40$ for the GEN in this work.

Table 9 Pros and cons of different autoencoder-based generative models

Methods	Pros	Cons
VAE (Kingma and Welling 2014)	• Efficient optimization of the variational lower bound by using the reparameterization strategy Efficient posterior inference using the recognition model which is an approximation to the intractable true posterior	• Pockets or holes in the latent distribution Prone to the collapse of the approximate posterior to prior
AAE (Makhzani et al. 2016)	• Implicitly matches the aggregate posterior to the prior using a discriminator in the latent space	• Sensitive to the choice of the discriminator, hyperparameter settings, and training strategy for optimization of the min-max objective function. Stopping criteria is unknown
RAE (Ghosh et al. 2020)	• Trains a deterministic autoencoder with simple regularization policies	• Ex-post density estimation over the latent representation hyperparameter tuning for the ex-post density estimation
WAE (Tolstikhin et al. 2018)	• Matches marginal distributions in the latent space using optimal transport theorem	• Problems related to adversarial training in the WAE-JSD (as observed in the AAE) Choice of kernels and related hyperparameters for the WAE-MMD
BiGAN (Donahue et al. 2017)	• Simultaneously learn the inference and generator networks	• Reconstructed samples from their latent representation do not preserve sample identity

Table 10 Neural network architectures of discriminators used in the correlation study for the latent dimension, l

Arch1	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 1, Sigmoid)$
Arch2	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 1, Sigmoid)$
Arch3	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 1, Sigmoid)$
Arch4	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 1, Sigmoid)$
Arch5	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 4\times l, ReLU) \rightarrow FC(4\times l, 1, Sigmoid)$
Arch6	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 10\times l, ReLU) \rightarrow FC(10\times l, 4\times l, ReLU) \rightarrow FC(4\times l, 1, Sigmoid)$
Arch7	$FC(l, 20\times l, ReLU) \rightarrow FC(20\times l, 10\times l, ReLU) \rightarrow FC(10\times l, 4\times l, ReLU) \rightarrow FC(4\times l, 1, Sigmoid)$
Arch8	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 4\times l, ReLU) \rightarrow FC(4\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 1, Sigmoid)$
Arch9	$FC(l, 10\times l, ReLU) \rightarrow FC(10\times l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 4\times l, ReLU) \rightarrow FC(4\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 1, Sigmoid)$
Arch10	$FC(l, 20\times l, ReLU) \rightarrow FC(20\times l, 10\times l, ReLU) \rightarrow FC(10\times l, 10\times l, ReLU) \rightarrow FC(10\times l, 5\times l, ReLU) \rightarrow FC(5\times l, 4\times l, ReLU) \rightarrow FC(4\times l, 2\times l, ReLU) \rightarrow FC(2\times l, 1, Sigmoid)$

Table 11 Neural network architecture of the Encoder and Decoder used for the MNIST data set for $l = 8$ in all the methods

Encoder	Decoder
$\mathbf{x} \in \mathbf{R}^{28\times 28\times 1}$	$\mathbf{z} \in \mathbf{R}^{8}$
$\downarrow$	$\downarrow$
$\texttt{Conv64} \rightarrow \texttt{LReLU}$	$\texttt{FC}_{1024} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Conv128} \rightarrow \texttt{LReLU}$	$\texttt{FC}_{1024} \rightarrow \texttt{FC}_{7\times 7\times 128} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Flatten}_{7\times 7\times 128} \rightarrow \texttt{FC}_{1024} \rightarrow \texttt{LReLU}$	$\texttt{TransConv64} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{FC}_{1024} \rightarrow \texttt{FC}_{8} \rightarrow \texttt{None}$	$\texttt{TransConv1} \rightarrow \texttt{Tanh}$

Table 12 Neural network architecture of the Encoder and Decoder used for the SVHN data set for $l = 35$ in all the methods

Encoder	Decoder
$\mathbf{x} \in \mathbf{R}^{32\times 32\times 3}$	$\mathbf{z} \in \mathbf{R}^{35}$
$\downarrow$	$\downarrow$
$\texttt{Conv64} \rightarrow \texttt{LReLU}$	$\texttt{FC}_{4\times 4\times 256} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Conv128} \rightarrow \texttt{LReLU}$	$\texttt{TransConv128} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Conv256} \rightarrow \texttt{LReLU}$	$\texttt{TransConv64} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Flatten}_{4\times 4\times 256} \rightarrow \texttt{FC}_{35} \rightarrow \texttt{None}$	$\texttt{TransConv3} \rightarrow \texttt{Tanh}$

Table 13 Neural network architecture of the Encoder and Decoder used for the CelebA data set in all the methods

Encoder	Decoder
$\mathbf{x} \in \mathbf{R}^{64\times64\times3}$	$\mathbf{z} \in \mathbf{R}^{l}$
$\downarrow$	$\downarrow$
$\texttt{Conv64} \rightarrow \texttt{LReLU}$	$\texttt{FC}_{4\times4\times512} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Conv128} \rightarrow \texttt{LReLU}$	$\texttt{TransConv256} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Conv256} \rightarrow \texttt{LReLU}$	$\texttt{TransConv128} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Conv512} \rightarrow \texttt{LReLU}$	$\texttt{TransConv64} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{Flatten}_{4\times4\times512} \rightarrow \texttt{FC}_{40} \rightarrow \texttt{None}$	$\texttt{TransConv3} \rightarrow \texttt{Tanh}$

Table 14 Neural network architecture of the Encoder and Decoder used for the NSL-KDD data set for $l = 2, 5$, and 10 in all the methods

Encoder	Decoder
$\mathbf{x} \in \mathbf{R}^{122}$	$\mathbf{z} \in \mathbf{R}^{l}$
$\downarrow$	$\downarrow$
$\texttt{FC}_{32} \rightarrow \texttt{LReLU}$	$\texttt{FC}_{32} \rightarrow \texttt{LReLU}$
$\downarrow$	$\downarrow$
$\texttt{FC}_{1} \rightarrow \texttt{None}$	$\texttt{FC}_{122} \rightarrow \texttt{None}$

Table 15 Neural network architecture of the discriminator used for the MNIST data set in the AAE

Discriminator
$\mathbf{z} \in \mathbf{R}^{8}$
$\downarrow$
$\texttt{FC}_{100} \rightarrow \texttt{LReLU}$
$\downarrow$
$\texttt{FC}_{100} \rightarrow \texttt{LReLU}$
$\downarrow$
$\texttt{FC}_{100} \rightarrow \texttt{LReLU}$
$\downarrow$
$\texttt{FC}_{1} \rightarrow \texttt{None}$

Appendix D: Anomaly detection

MNIST

For this data set, we leverage the labeling of the data to identify inlier and outlier examples. In this setup, a class of digit is selected as inlier and examples from that class in the training data are used to train the model. For outlier detection evaluation, a set is constructed with samples from the inlier class in test data mixed with random samples from other classes, representing

Table 16 Neural network architecture of the discriminator used for the SVHN data set in the AAE

Discriminator
$\mathbf{z} \in \mathbf{R}^{35}$
↓
$\texttt{FC}_{1024} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{1024} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{1024} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{1} \rightarrow \texttt{None}$

Table 17 Neural network architecture of the discriminator used for the CelebA data set in the AAE

Discriminator
$\mathbf{z} \in \mathbf{R}^{l}$
↓
$\texttt{FC}_{1024} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{4096} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{1024} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{1} \rightarrow \texttt{None}$

Table 18 Neural network architecture of the discriminator used for the NSL-KDD data set in the AAE for $l = 2, 5$, and 10

Discriminator
$\mathbf{z} \in \mathbf{R}^{l}$
↓
$\texttt{FC}_{5} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{5} \rightarrow \texttt{LReLU}$
↓
$\texttt{FC}_{1} \rightarrow \texttt{None}$

outliers, with anomaly proportion, ρ, ranging from 0.1 through 1.0 with a step size of 0.1, $\rho = [0.1 : 1.0;0.1]$. In this experiment no noise is considered in the training data, such as adding examples from a different class. The threshold for detecting anomalous samples in an evaluation set is determined from the anomaly proportion, ρ. Say for $\rho = 0.4$, top 40% of the sorted scores will be marked as outliers. For uncluttered representation, only F_1 scores of all methods are reported in Table 19 for all MNIST classes for $\rho = 0.2$ and $\rho = 1.0$ on the left |

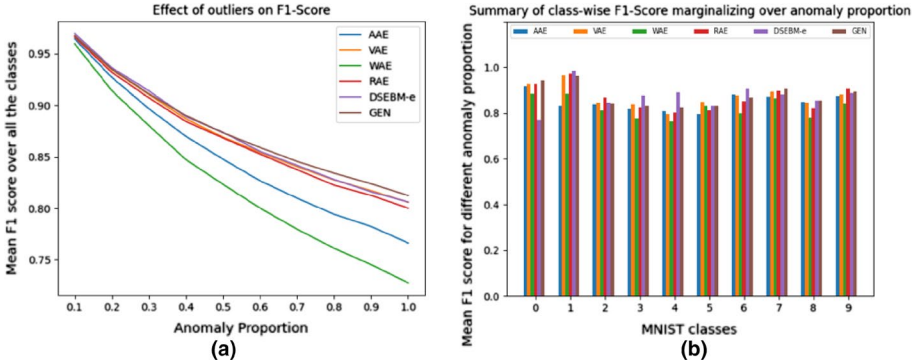

Fig. 5 Average F_1 scores **a** for different anomaly proportions marginalized over all MNIST classes **b** for the MNIST classes averaged over different anomaly proportions, ρ

right, respectively. Since the test set contains outlier samples chosen randomly from the MNIST classes leaving out the inlier class, an average of the F_1 scores for 10 different evaluation sets is reported in Table 19. It is observed that the GEN outperforms other methods for many classes both for low and high outlier percentages or it closely follows the best performing method. To evaluate the performance of the AAE, VAE, WAE, RAE, and GEN as outlier detector we have reported the results for the DSEBM, a specially designed outlier detection algorithm. Out of the two proposed metric, DSEBM-e and DSEBM-r, we have considered the DSEBM-e in our experiment due to better accuracy in its predictions. The encoder architecture used in the DSEBM-e is built following instructions in the paper (Zhai et al., 2016). The autoencoder architecture used for all the methods in this experiment is defined in "Appendix C". Mean F_1 scores marginalized over all MNIST classes for different anomaly proportions are shown in Fig. 5a. It can be observed that the GEN is better than all competing methods for different scenarios. The GEN is even better than the DSEBM-e in many occasions. Performance of the RAE and VAE closely follow GENs. Likewise, F_1 scores for different MNIST classes, representing average over different anomaly proportions, ρ, can be found in Fig. 5b. It can be inferred from the bar plot that the DSEBM-e is doing consistently better for all the MNIST classes except 0 which is also observed in Table 19. The GEN is observed to be among the top performers for all the digits.

KDDCup99

The KDDCup99 10 percent data set (Lichman 2013) from the UCI repository is a computer network intrusion data having continuous and categorical variables. Using continuous variable and one-hot vector representation of the categorical data, the length of each entry is 121 dimension. This data set has 20% normal data and 80% anomalous data labeled as "attack". Since, the percentage of "attack" data is significantly more than the normal data, we consider anomalous data as normal class in our experiments. In the basic set up, the data is split into equal proportion as train and test data. From the train data, only normal samples are used for training the network parameters and samples from the test data, containing both normal and attack examples, are used for model evaluation using the F_1 score. Subsequently, the training data is corrupted by including examples from abnormal class in the training set, ranging from 1% through 5% of the normal samples. For this experiment, we use the same autoencoder architecture as used in the

Table 19 Mean F1 scores of the methods for different data classes in the MNIST data set for $\rho = 0.2$ and $\rho = 1.0$ on the left right, respectively, averaged over 10 different evaluation sets

	AAE	VAE	WAE	RAE	DSEBM-e	GEN
Zero	0.954 | 0.873	0.963 | 0.888	0.931 | 0.835	0.958 | 0.893	0.892 | 0.630	**0.964** | **0.913**
One	0.910 | 0.744	0.979 | 0.947	0.941 | 0.817	0.982 | 0.959	**0.999** | **0.981**	0.973 | 0.949
Two	0.915 | 0.750	0.911 | 0.763	0.900 | 0.709	**0.923** | **0.802**	0.915 | 0.757	0.905 | 0.767
Three	0.923 | 0.722	0.926 | 0.760	0.908 | 0.644	0.919 | 0.728	**0.946** | **0.818**	0.924 | 0.740
Four	0.908 | 0.701	0.895 | 0.687	0.889 | 0.637	0.904 | 0.692	**0.939** | **0.835**	0.912 | 0.732
Five	0.912 | 0.683	**0.931** | **0.771**	0.922 | 0.740	0.916 | 0.713	0.921 | 0.749	0.927 | 0.743
Six	**0.950** | 0.817	0.941 | 0.823	0.913 | 0.690	0.926 | 0.779	**0.950** | **0.868**	0.939 | 0.801
Seven	0.935 | 0.802	0.938 | 0.843	0.924 | 0.792	0.941 | 0.846	0.936 | 0.821	**0.949** | **0.857**
Eight	0.918 | 0.764	0.916 | 0.756	0.884 | 0.658	0.898 | 0.729	**0.923** | 0.775	0.912 | **0.776**
Nine	0.944 | 0.806	0.944 | 0.819	0.929 | 0.756	**0.950** | **0.858**	**0.950** | 0.828	0.945 | 0.845

Table 20 Mean F1 scores of the methods for different noise % considered in the training data for KDDCup99 10 percent data set averaged over 10 different evaluation sets

noise%	AAE	VAE	WAE	RAE	DSEBM-e	DAGMM	GEN
0%	0.904	0.901	0.914	0.914	0.909	0.930	**0.960**
1%	0.862	**0.943**	0.860	0.849	0.938	0.918	0.942
2%	0.885	0.892	0.866	0.876	0.880	**0.959**	0.956
3%	0.905	0.926	0.875	0.872	0.863	0.955	**0.964**
4%	0.921	0.909	0.874	0.868	0.866	**0.932**	**0.932**
5%	0.919	0.920	0.903	0.866	0.850	0.952	**0.971**

DAGMM Zong et al. (2018) for the AAE, VAE, WAE, RAE, and GEN, except that the latent dimension is increased to 2 from 1 (used in the compression network of DAGMM). This is a reasonable choice as the DAGMM augments two more latent features (relative euclidean distance and cosine similarity) to the encoded vector for estimating the density in the latent space. The encoder used in the DSEBM is the same as used in the DAGMM. In addition, we followed other experimental details like optimizer, learning rate, batch size, epochs, as used in the training of the DAGMM. The percentage of abnormal samples ($\sim 20\%$) in the test data is used to determine the threshold for detecting outliers. Since the test set is created by a random subset of the original data, 10 different sets of train and test data are used for training and evaluation of all the methods. From the average F_1 score scores reported in Table 20, it can be inferred that the F_1 score of almost all methods are immune to the presence of noisy samples in the training data, except the RAE where, we observe a downward trend in the F_1 score with the increase in amount of noisy samples. The performance of GENs is better than other methods in almost all scenarios except a couple where it is the next best method.

Author Contributions All authors have contributed to the formulation of the proposed method. Mr. Surojit Saha conducted experiments, analysis of the results, and preparation of manuscript under the supervision of Prof. (Dr.) Shireen Elhabian and Prof. (Dr.) Ross Whitaker.

Funding This work is funded by ExxonMobil Corporation.

Availability of data and materials Experimental results are mostly produced using publicly available data sets. Following data sets are used for the experimental results: MNIST (LeCun et al., 2010), SVHN (Netzer et al., 2011), CelebA (Liu et al., 2015), NSL-KDD (Xu et al., 2021), and KDDCup99 10 percent (Lichman 2013). The source code used for generating data for *correlation study*, discussed in Sect. 5.1, is available from the corresponding author on request.

Code availability Codes developed/used by the authors for results reported in this submission can be produced by the corresponding author on request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval This article does not contain any studies with human participants or animals performed by any of the authors. So, formal consent is not required for this submission.

References

Arjovsky, M., Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In: International conference on learning representations.

A. Barnett, S. (2018). Convergence problems with generative adversarial networks (GANs). Preprint at https://arxiv.org/abs/1806.11382.

Bauer, M., Mnih, A. (2019). Resampled priors for variational autoencoders. In International conference on artificial intelligence and statistics.

Bhalodia, R., Lee, I., Elhabian, S. (2020). dpvaes: Fixing sample generation for regularized vaes. In: Asian conference on computer vision.

Chalapathy, R., Chawla, S. (2019). Deep learning for anomaly detection: A survey. Preprint at https://arxiv.org/abs/1901.03407.

Cup, K. D.D. (1999). Machine learning repository Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Dai, B., Wang, Z., Wipf, D. (2020). The usual suspects? reassessing blame for vae posterior collapse. In: International conference on machine learning.
Dai, B., Wipf, D. (2019). Diagnosing and enhancing vae models. In International conference on learning representations.
Donahue, J., Krähenbühl, P., Darrell, T. (2017). Adversarial feature learning. In: International conference on learning representations.
Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., & Courville, A. (2017). Adversarially learned inference. In: International conference on learning representations.
Germain, M., Gregor, K., Murray, I., Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In: International conference on machine learning.
Ghosh, P., Sajjadi, M.S.M., Vergari, A., Black, M., Scholköpf, B. (2020). From variational to deterministic autoencoders. In: International conference on learning representations.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In: Conference on neural information processing systems.
Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A. (2017) Improved training of wasserstein gans. In: Conference on neural information processing systems.
Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Conference on neural information processing systems.
Hoffman, M.D., Johnson, M.J. (2016) Elbo surgery: yet another way to carve up the variational evidence lower bound. In: NIPS workshop: Advances in approximate bayesian inference.
Kingma, D.P., Welling, M. (2014). Auto-encoding variational bayes. In International conference on learning representations.
Kodali, N., Abernethy, J., Hays, J., & Kira, Z. (2017) On convergence and stability of GANs. Preprint at https://arxiv.org/abs/1705.07215.
Larochelle, H., Murray, I. (2011). The neural autoregressive distribution estimator. In: International conference on artificial intelligence and statistics.
LeCun, Y., Cortes, C., & Burges, C. (2010). MNIST handwritten digit database. Available on: http://yann.lecun.com/exdb/mnist.
Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology, 49*(4), 764–766.
Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., & Póczos, B. (2017). Mmd gan: Towards deeper understanding of moment matching network. In: Conference on neural information processing systems.
Li, Y., Swersky, K., Zemel, R. (2015). Generative moment matching networks. In: International conference on machine learning.
Lichman, M. (2013). UCI machine learning repository. Available on: http://archive.ics.uci.edu/ml.
Liu, S., Bousquet, O., & Chaudhuri, K. (2017). Approximation and convergence properties of generative adversarial learning. In: Conference on neural information processing systems.
Liu, Z., Luo, P., Wang, X., Tang, X. (2015). Deep learning face attributes in the wild. In: International conference on computer vision. http://dblp.uni-trier.de/db/conf/iccv/iccv2015.html#LiuLWT15.
Lucasz, J., Tuckery, G., Grossez, R., & Norouziy, M. (2019). Understanding posterior collapse in generative latent variable models. In: International conference on learning representations.
Lucic, M., Kurach, K., Michalski, M., Bousquet, O., Gelly, S. (2017). Are gans created equal? a large-scale study. In: Conference on neural information processing systems.
Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B. (2016). Adversarial autoencoders. In: International conference on learning representations.
Mao, X., Li, Q., Xie, H., Y.K. Lau, R., Wang, Z., Paul Smolley, S. (2015). Least squares generative adversarial networks. In: International conference on computer vision.
Mescheder, L., Geiger, A., & Nowozin, S. (2018). Which training methods for gans do actually converge? In: International conference on machine learning.
Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A.Y. (2011). Reading digits in natural images with unsupervised feature learning. In: Conference on neural information processing systems.
Nowozin, S., Cseke, B., & Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In: Conference on neural information processing systems.
Oord, A.v.d., Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning. In: Conference on neural information processing systems.
Pidhorskyi, S., Almohsen, R., Adjeroh, D.A., & Doretto, G. (2018). Generative probabilistic novelty detection with adversarial autoencoders. In: Conference on neural information processing systems.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In: International conference on learning representations.
Razavi, A., Oord, A.v.d., Poole, B., & Vinyals, O. (2019). Preventing posterior collapse with δ-vaes. In: International conference on learning representations.
Rezende, D.J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In: International conference on machine learning, pp. 1278–1286.
Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In: International Conference on Machine Learning.
Rosca, M., Lakshminarayanan, B., & Mohamed, S. (2018) Distribution matching in variational inference. Preprint at https://arxiv.org/abs/1802.06847.
Roth, K., Lucchi, A., Nowozin, S., Hofmann, T. (2017). Stabilizing training of generative adversarial networks through regularization. In: Conference on neural information processing systems.
Sabokrou, M., Khalooei, M., Fathy, M., Adeli, E. (2018). Adversarially learned one-class classifier for novelty detection. In: IEEE conference on computer vision and pattern recognition.
Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. (2016) Improved techniques for training gans. In: Conference on neural information processing systems.
Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. London: Chapman and Hall.
Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A. (2009). A detailed analysis of the kdd cup 99 data set. In: IEEE symposium on computational intelligence in security and defense applications.
Tolstikhin, I., Bousquet, O., Gelly, S., Schoelköpf, B. (2018). Wasserstein auto-encoders. In: International conference on learning representations.
Tomczak, J.M., Welling, M. (2018). Vae with a vampprior. In: International conference on artificial intelligence and statistics.
Tschannen, M., Bachem, O., & Lucic, M. (2018). Recent advances in autoencoder-based representation learning. In: Workshop on Bayesian Deep Learning NeurIPS).
Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008) Extracting and composing robust features with denoising autoencoders. In: International Conference on Machine Learning, pp. 1096–1103.
Xia, Y., Cao, X., Wen, F., Hua, G., Sun, J. (2015). Learning discriminative reconstructions for unsupervised outlier removal. In: International conference on computer vision.
Xiao, Z., Yan, Q., Chen, Y., Amit, Y. (2019). Generative latent flow: A framework for non-adversarial image generation. Preprint at https://arxiv.org/abs/1905.10485.
Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., & Sabrina, F. NSL-KDD. Available on: https://www.unb.ca/cic/datasets/nsl.html.
Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., & Sabrina, F. (2021). Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. *IEEE Access, 9,* 140136–140146.
Zhai, S., Cheng, Y., Lu, W., & Zhang, Z.M. (2016). Deep structured energy based models for anomaly detection. In: International conference on machine learning.
Zhu, J.-Y., Krähenbühl, P., Shechtman, E., & Efros, A.A. (2016). Generative visual manipulation on the natural image manifold. In: European conference on computer vision.
Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: International conference on learning representations.