BRIDGING THE ELBO AND MMD

Anonymous authors
Paper under double-blind review

ABSTRACT

One of the challenges in training generative models such as the variational auto encoder (VAE) is avoiding posterior collapse. When the generator has too much capacity, it is prone to ignoring latent code. This problem is exacerbated when the dataset is small, and the latent dimension is high. The root of the problem is the ELBO objective, specifically the Kullback–Leibler (KL) divergence term in objective function (Zhao et al., 2019). This paper proposes a new objective function to replace the KL term with one that emulates the maximum mean discrepancy (MMD) objective. It also introduces a new technique, named latent clipping, that is used to control distance between samples in latent space. A probabilistic autoencoder model, named μ-VAE, is designed and trained on MNIST and MNIST Fashion datasets, using the new objective function and is shown to outperform models trained with ELBO and β-VAE objective. The μ-VAE is less prone to posterior collapse, and can generate reconstructions and new samples in good quality. Latent representations learned by μ-VAE are shown to be good and can be used for downstream tasks such as classification.

1 INTRODUCTION

Autoencoders(AEs) are used to learn low-dimensional representation of data. They can be turned into generative models by using adversarial, or variational training. In the adversarial approach, one can directly shape the posterior distribution over the latent variables by either using an additional network called a Discriminator (Makhzani et al., 2015), or using the encoder itself as a discriminator (Huang et al., 2018). AEs trained with variational methods are called Variational Autoencoders (VAEs) (Kingma & Ba, 2014; Rezende et al., 2014). Their objective maximizes the variational lower bound (or evidence lower bound, ELBO) of $p_\theta(x)$. Similar to AEs, VAEs contain two networks:

Encoder - Approximate inference network: In the context of VAEs, the encoder is a recognition model $q_\phi(z|x)$[1], which is an approximation to the true posterior distribution over the latent variables, $p_\theta(z|x)$. The encoder tries to map high-level representations of the input x onto latent variables such that the salient features of x are encoded on z.

Decoder - Generative network: The decoder learns a conditional distribution $p_\theta(x|z)$ and has two tasks: i) For the task of reconstruction of input, it solves an inverse problem by taking mapped latent z computed using output of encoder and predicts what the original input is (i.e. reconstruction $x' \approx x$). ii) For generation of new data, it samples new data x', given the latent variables z.

During training, encoder learns to map the data distribution $p_d(x)$ to a simple distribution such as Gaussian while the decoder learns to map it back to data distribution $p(x)$ [2]. VAE's objective function has two terms: log-likelihood term (reconstruction term of AE objective function) and a prior regularization term [3]. Hence, VAEs add an extra term to AE objective function, and approximately maximizes the log-likelihood of the data, $\log p(x)$, by maximizing the evidence lower bound (ELBO):

[1] ϕ refers to parameters of encoder while θ is parameters of decoder.
[2] Note that output distribution of decoder is model distribution $p(x)$, not data distribution $p_d(x)$
[3] One such term would be Kullback-Leibler (KL) divergence, which is a measure of how similar two probability distributions are.

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{p_d(x)} \left[\mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right]\right] - \mathbb{E}_{p_d(x)} \left[\text{KL}(q_\phi(z|x) \| p(z))\right] \tag{1}$$

Maximizing ELBO does two things:

- Increase the probability of generating each observed data x.
- Decrease distance between estimated posterior $q(z|x)$ and prior distribution $p(z)$, pushing KL term to zero. Smaller KL term leads to less informative latent variable.

Pushing KL terms to zero encourages the model to ignore latent variable. This is especially true when the decoder has a high capacity. This leads to a phenomenon called posterior collapse in literature (Razavi et al., 2019; Chen et al., 2016; Dieng et al., 2018; Kim et al., 2018; van den Oord et al., 2017; Bowman et al., 2015; Kingma et al., 2016; Sønderby et al., 2016; Zhao et al., 2017).

This work proposes a new method to mitigate posterior collapse. The main idea is to modify the KL term of the ELBO such that it emulates the MMD objective (Gretton et al., 2007; Zhao et al., 2019). In ELBO objective, minimizing KL divergence term pushes mean and variance parameters of each sample at the output of encoder towards zero and one respectively. This , in turn, brings samples closer, making them indistinguishable. The proposed method replaces the KL term in the ELBO in order to encourage samples from latent variable to spread out while keeping the aggregate mean of samples close to zero. This enables the model to learn a latent representation that is amenable to clustering samples which are similar. As shown in later sections, the proposed method enables learning good generative models as well as good representations of data. The details of the proposal are discussed in Section 4.

2 Related Work

In the last few years, there have been multiple proposals on how to mitigate posterior collapse. These proposals are concentrated around i) modifying the ELBO objective, ii) imposing a constraint on the VAE architecture, iii) using complex priors, iv) changing distributions used for the prior and the posterior v) or some combinations of these approaches. Modifications of the ELBO objective can be done through annealing the KL term (Sønderby et al., 2016; Bowman et al., 2015), lower-bounding the KL term to prevent it from getting pushed to zero (Razavi et al., 2019), controlling KL capacity by upper bounding it to a pre-determined value (Burgess et al., 2018) or lower-bounding the mutual information by adding skip connections between the latent layer and the layers of the decoder (Dieng et al., 2018). Proposals that constrain the structure of the model do so by reducing the capacity of the decoder (Bowman et al., 2015; Yang et al., 2017; Gulrajani et al., 2016), by adding skip connections to each layer of the decoder (Dieng et al., 2018), or by imposing constraints on encoder structure (Razavi et al., 2019). Taking a different approach, Tomczak & Welling (2017) and van den Oord et al. (2017) replace simple Gaussian priors with more complex ones such as a mixture of Gaussians.

The most recent of these proposals are δ-VAE (Razavi et al., 2019) and SKIP-VAE (Dieng et al., 2018). δ-VAE imposes a lower bound on KL term to prevent it from getting pushed to zero. One of the drawbacks of this approach is the fact that it introduces yet another hyper-parameter to tune carefully.Also, the model uses dropout to regularize the decoder, reducing the effective capacity of the decoder during training. It is not clear how effective the proposed method is when training more powerful decoders without such regularization. Moreover, the proposal includes an additional constraint on encoder structure, named the anti-causal encoder.

SKIP-VAE , on the other hand, proposes to lower bound mutual information by adding skip connections from latent layers to each layer of decoder. One drawback of this approach is that it introduces additional non-linear layer per each hidden layer, resulting in more parameters to optimize. Moreover, its advantage is not clear in cases, where one can increase capacity of decoder by increasing number of units in each layer (or number of channels in CNN-based decoders) rather than adding more layers.

3 The problem statement:

When we train a VAE model, we ideally want to end up with a model that can reconstruct a given input well and can generate new samples in high quality. Good reconstruction requires extracting the most salient features of data and storing them on latent variable ('Encoder + Latent layer' part of the model). Generating good samples requires a generative model ('Latent layer + Decoder' part) with a model distribution that is a good approximation to actual data distribution.

However, there tends to be a trade-off between reconstruction quality of a given input, and quality of new samples. To understand why we have such a trade-off, we can start by looking at ELBO objective function[4]:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \text{KL}(q_\phi(z|x) \| p(z)) \tag{2}$$

Maximizing this objective function increases $p_\theta(x)$, the probability of generating each observed data x while decreasing distance between $q(z|x)$ and prior $p(z)$. Pushing $q(z|x)$ closer to $p(z)$ makes latent code less informative i.e. z is influenced less by input data x.

The reason why the KL term can be problematic becomes more clear when we look at the KL loss term typically modelled with log of variance during optimization:

$$KL_{loss} = \frac{1}{2}\sum_{d=1}^{D}\left[\mu_d^{(i)2} + \left[\exp(\log\sigma^2)\right]_d^{(i)} - (\log\sigma^2)_d^{(i)} - 1\right] \tag{3}$$

where D is the dimension of latent variable, and i refers to i^{th} sample. Noting that the mean is in L2 norm, minimizing the KL term leads to pushing the each dimension of the mean,$\mu_d^{(i)}$, to zero while pushing σ^2 towards 1. This makes estimated posterior less informative and less dependent on input data. The problem gets worse when dimension of latent variable, D, increases, or when the KL term is multiplied with a coefficient $\beta > 1$ (Higgins et al., 2017). Ideally, we want to be able to distinguish between different input samples. This can be achieved by having distinctive means and variances for clusters of samples. This is where MMD might have advantage over the KL divergence. Matching distributions using MMD can match their sample means although their variance might still differ.

4 PROPOSAL: μ-VAE

We can emulate behaviour of MMD by modifying the KL term. We do so by changing L2 norm of mean, $\sum_{i=1}^{D}\mu_d^{(i)2}$ to L1 norm, $|\sum_{i=1}^{D}\mu_d^{(i)}|$. Re-writing it for B samples, we have:

$$\frac{1}{B}|\sum_{i=1}^{B}\sum_{d=1}^{D}\mu_d^{(i)}| \tag{4}$$

It is important to note that we are taking absolute value of sum of sample means. This new formulation results in aggregate mean of samples to be zero (i.e. same mean as that of prior distribution) while allowing samples to spread out and enabling model to encode information about input data onto z. It should be noted that this new L1 norm of μ can push individual mean estimates to very high values if it is not constrained. To avoid that, L2 norm of means for each sample is clipped by a pre-determined value during optimization. Based on experiments, it is found that clipping L2 norm of sample means by three times square root of latent dimension works well in general although bigger values might help improve results in tasks such as classification:

$$\|\mu_{sample}\| \leq 3 * \sqrt{z_{dim}} \tag{5}$$

This method will be referred as latent clipping for the rest of this work. In addition, the remaining terms in the KL loss can be kept as is, i.e. $\left[\exp\left(\log\sigma^2\right) - \log\sigma^2 - 1\right]$, or we can just optimize

[4]Ignoring $\mathbb{E}_{p_d(x)}\left[.\right]$ term for clarity

for subset of it by using either "$\log\sigma^2$", or "$\left[\exp\left(\log\sigma^2\right)-1\right]$" term since each method will push $\log\sigma^2$ towards zero (i.e. variance towards one). $\log\sigma^2$ is chosen in this work since it is simpler.

Finally, the μ-VAE objective function can be formulated as follows:

$$\boxed{\mathcal{L}_{\mu\text{-VAE}} = \frac{1}{B}\left[\sum_{i=1}^{B}\sum_{j=1}^{J}\|x_j^{(i)} - x_j^{\prime(i)}\|^2 + |\sum_{i=1}^{B}\sum_{d=1}^{D}\mu_d^{(i)}| + \sum_{i=1}^{B}\sum_{d=1}^{D}\left[\log\sigma^2\right]_d^{(i)}\right]} \tag{6}$$

where first term is reconstruction loss, B refers to batch size since aggregated mean is computed over batch samples, J refers to dimension of data, D refers to dimension of latent variable, x is original input, and x' is reconstructions.

4.1 Latent Clipping:

To visualize the implications of the latent clipping, a toy VAE model shown in Table 2 in Appendix A is used. Figure 1 compares three cases, in which a VAE model is trained on MNIST dataset using ReLu, Tanh, and Leaky ReLu activation functions for each case, and the latent layer is visualized to observe clustering of digits. Those three cases are: i) Standard VAE objective with the KL divergence, ii) Standard VAE objective with the KL divergence + latent clipping, and iii) μ-VAE objective function + latent clipping. Two observations can be made:

1. Latent clipping might help improve smoothness of latent space, even in the case of standard VAE objective, ELBO.
2. μ-VAE objective function seems to work well.

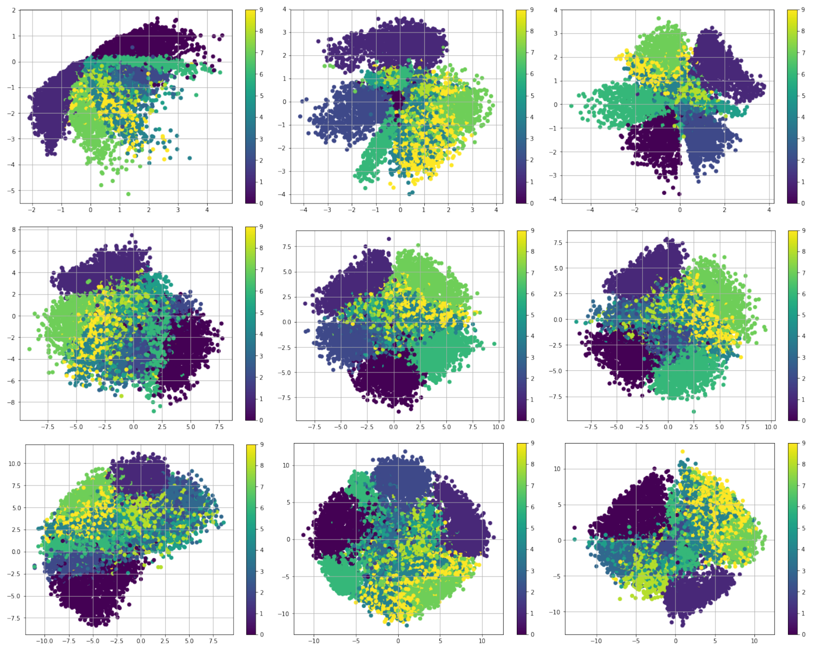

Figure 1: **Clustering of MNIST:** From left to right: Tanh, ReLu, Leaky ReLu. **Top row:** Standard VAE training with ELBO objective, **Middle row:** Standard VAE with ELBO objective and latent clipping, **Bottom row:** μ-VAE objective function and latent clipping.

5 Experiments

To test the effectiveness of μ-VAE objective, a CNN-based VAE model is designed and trained on MNIST and MNIST Fashion using same hyper-parameters for both datasets. Centered isotropic

Gaussian prior, $p(z) \sim \mathcal{N}(0,\, 1.0)$, is used and the true posterior is approximated as Gaussian with an approximately diagonal co-variance. No regularization methods such as dropout, or techniques such as batch-normalization is used to avoid having any extra influence on the performance, and to show the advantages of the new objective function.

The model is trained with four different objective functions: i) VAE (ELBO objective), ii) β-VAE with $\beta = 4$, iii) μ-VAE#1 s.t. $\|\mu_{sample}\| \leq 3 * \sqrt{z_{dim}}$ and iv) μ-VAE#2 s.t. $\|\mu_{sample}\| \leq 6 * \sqrt{z_{dim}}$, where $z_{dim} = 10$. Details of architecture, objective functions, hyper-parameters, and training are described in Appendix B.

During training of the models, a simple three layer fully connected classifier is also trained over 10 dimensional latent variable to learn to classify data using features encoded on latent variable. Classifier parameters are updated when encoder and decoder parameters are frozen and vice versa so that classifier has no impact on how information is encoded on the latent variable.

5.1 Evaluation

Evaluation of the generative model is done qualitatively in the form of inspecting quality, and diversity of samples. Posterior collapse is assessed by comparing reconstructions of input data to observe whether the decoder ignores latent code encoded by input data and by comparing the KL divergences obtained for each model. For all three objective functions, the KL divergence is measured using standard KL formula in Equation 3. Moreover, the accuracy of the classifier trained on latent variable is used as a measure of how well the latent variable represents data (Dieng et al., 2018). Higher classification accuracy reflects a better representation of data and opens doors to use latent representation for downstream tasks such as classification.

5.2 Results

Figure 2 shows training curves for MNIST Fashion dataset (MNIST results can be seen in Appendix C). The new objective function results in lower reconstruction loss, higher KL divergence, and higher classification accuracy. Higher KL divergence and classification accuracy can be interpreted as a sign of learning a more informative latent code. β-VAE performs the worst across all metrics as expected. The reason is that β factor encourages latent code to be less informative, and is known to result in worse reconstruction quality (Higgins et al., 2017).

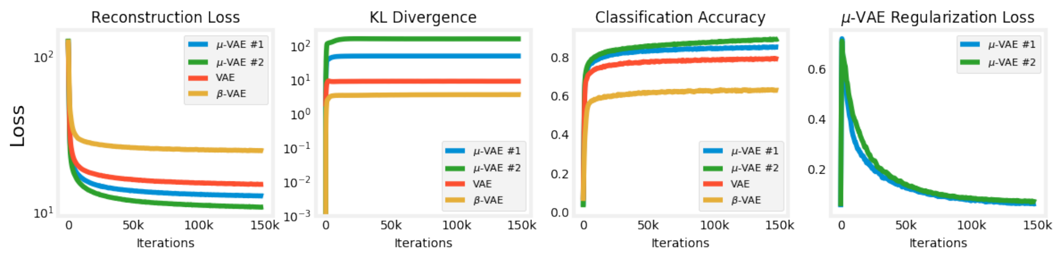

Figure 2: Training curves of the model trained on MNIST Fashion using each of four objectives: VAE (ELBO), β-VAE ($\beta = 4$), μ-VAE#1 and μ-VAE#2. Plots from left to right: reconstruction loss, KL divergence, classification accuracy and regularization loss of μ-VAE ($|\sum_{n=1}^{B} \mu_n| + \sum_{n=1}^{B} \left[\log \sigma^2\right]_n$) i.e. the term that replaces KL.

Figure 3 shows results of t-SNE (Maaten & Hinton, 2008) of samples obtained using test data for both datasets. VAE seems to able to distinguish all ten digits, but performs worse in MNIST Fashion. β-VAE pushes samples closer as expected, which explains why its performance is low in classification task. μ-VAE, on the other hand, is able to cluster similar samples together in both datasets. Moreover, when upper-bound on $\|\mu_{sample}\|$ is increased, it spreads out clusters of samples, making them easier to distinguish. Hence, upper-bound used in latent clipping can be a knob to control distance between samples. Also, we should note that we can achieve similar clustering results to the

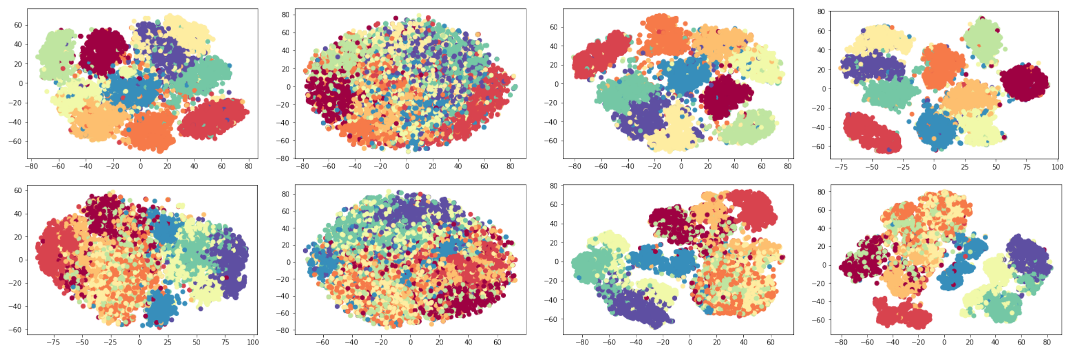
Figure 3: Clustering of samples from latent layer obtained using t-SNE and test datasets of MNIST (top row) and MNIST Fashion (bottom row). From left to right: VAE, β-VAE, μ-VAE#1, and μ-VAE#2.

one obtained by VAE through reducing the upper-bound. Smaller upper-bound closes gaps between the clusters.

Table 1 lists classification accuracy obtained using test datasets. μ-VAE performs the best as expected since it is able to push the clusters apart. Higher upper bound on $\|\mu_{sample}\|$ results in a higher classification accuracy. Also, it should be noted that reported accuracy numbers can be improved, but the purpose of this test was to show that new objective function can reliably be used in downstream tasks such as classification.

Table 1: Comparing test accuracy of classifiers trained on latent variables of each model.

Model	**MNIST**	**MNIST Fashion**
VAE	93.422	78.01
β**-VAE**	62.45	62.35
μ**-VAE #1**	**95.28**	**82.71**
μ**-VAE #2**	**96.44**	**84.260**

Figure 4 compares sample distributions obtained at each dimension of latent variable using test dataset of MNIST Fashion for each objective function. β-VAE samples follow $N(0, 1)$ prior very closely, and hence resulting in the smallest KL divergence term. Sample distributions from the most dimensions of VAE are also close to prior distribution although some of them show a multi-modal behavior. Sample distributions from both μ-VAE#1 & #2 result in zero mean, but they are more spread out as expected. Spread is controlled by upper-bound on $\|\mu_{sample}\|$. Similar to VAE, some sample distributions show a multi-modal behavior.

Figure 5 shows reconstruction of input using test dataset. β-VAE reconstructions are either blurrier, or wrong, the latter of which is a sign of posterior collapse. VAE performs better, and both versions of μ-VAE gives the best reconstruction quality.

Figure 6 shows images generated using random samples drawn from multivariate Gaussian, N(0, σ), where $\sigma = 1$ is for VAE and β-VAE while it is 3 for μ-VAE since their samples are more spread out (MNIST results can be seen in Appendix C). We can observe that some samples generated from μ-VAE models have dark spots. This is because the model is trying to generate texture on these samples. This can also be observed in samples of VAE model, but it is less pronounced. However, samples from β-VAE do not show any such phenomena since the model perhaps learns global structure of shapes while ignoring local features. Failing to capture local structures is a known problem in latent variable models (Larsen et al., 2015; Razavi et al., 2019).

Figure 7 shows latent traverse in each dimension of latent variable for MNIST Fashion (MNIST results can be seen in Appendix C). Each dimension of VAE and β-VAE is swept in $[-2, 2]$ range linearly while other dimensions are kept at zero. For μ-VAE models, ranges of $[-10, 10]$ and $[-20, 20]$ are used since samples are more spread out. β-VAE gives mostly similar classes of objects, a sign

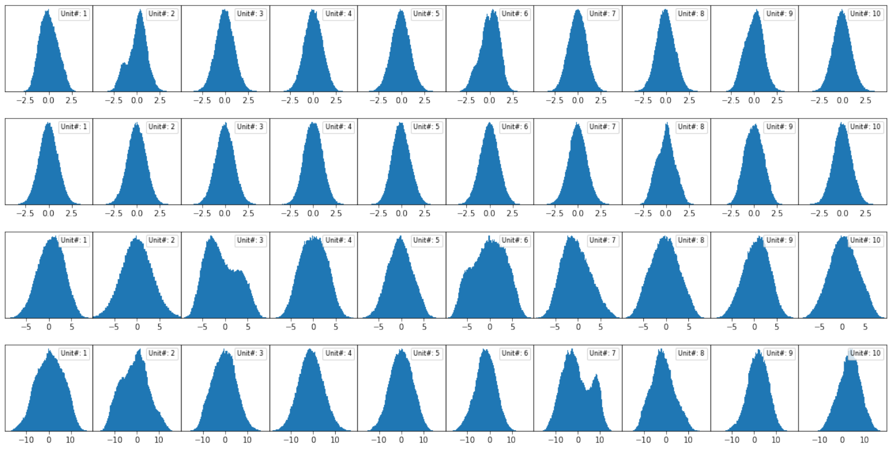

Figure 4: Sample distribution obtained at each dimension of latent variable using test dataset of MNIST Fashion. From top to bottom: VAE, β-VAE, μ-VAE#1, and μ-VAE#2.

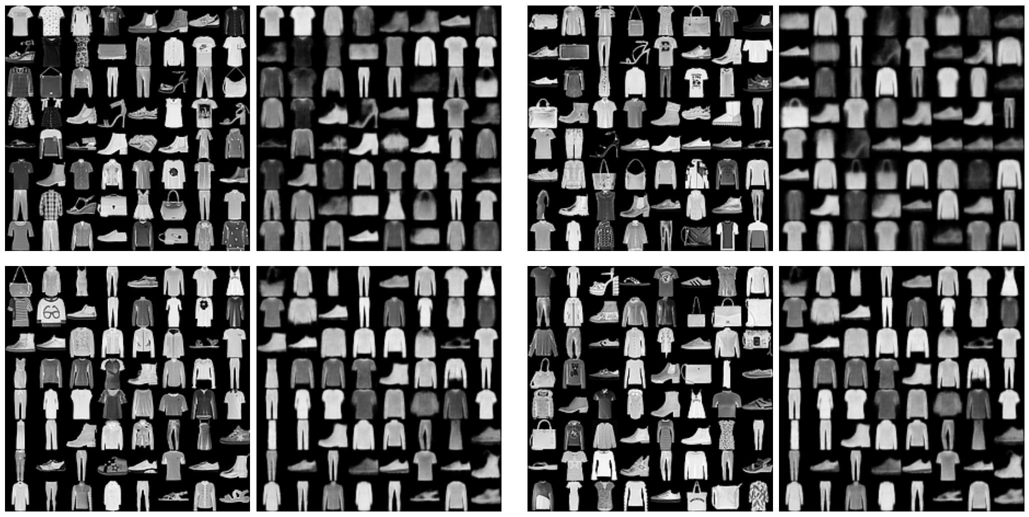

Figure 5: Reconstructions of input data obtained using test dataset of MNIST Fashion. Top row: VAE (left), β-VAE (right). Bottom row: μ-VAE#1 (left), μ-VAE#2 (right).

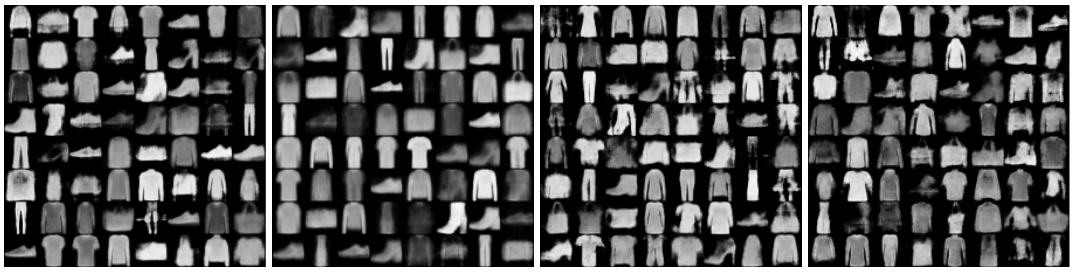

Figure 6: Random samples drawn from multi-variate Gaussian, N(0, σ).From left to right, model (σ): VAE (σ=1), β-VAE (σ=1), μ-VAE#1 (σ=3), and μ-VAE#2 (σ=3). Higger σ is used for μ-VAE models since their samples are more spread out.

that most dimensions of latent variable are not very informative. VAE is slightly better. However, both μ-VAE models learn diverse classes of objects across different dimensions. Moreover, they learn different classes on opposite sides of same dimension. This is encouraging since it shows its power to learn rich representations.

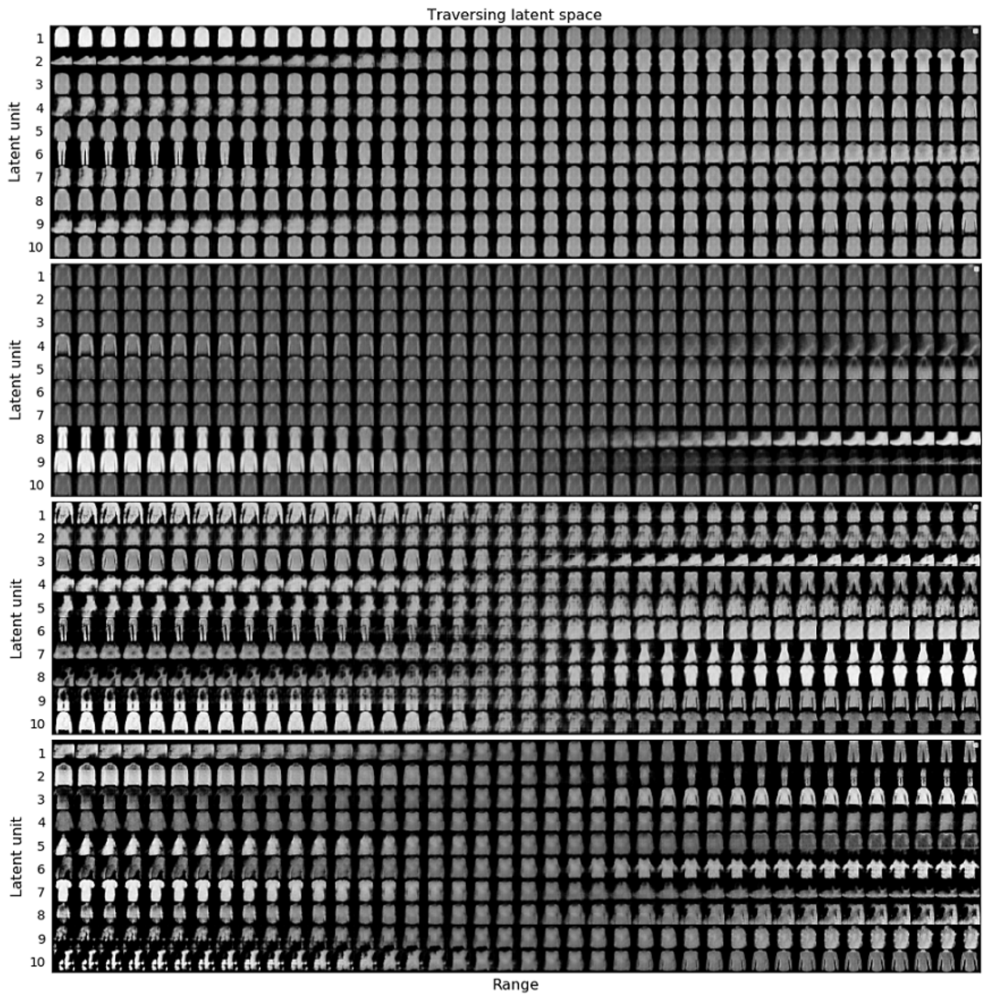

Figure 7: Traversing each dimension of latent variable in 40 steps. From top to bottom, the model and sweeping range used for the model: VAE [-2,2], β-VAE [-2,2], μ-VAE#1 [-10,10], μ-VAE#2 [-20,20].

6 Summary

In this work, a new objective function is proposed to mitigate posterior collapse observed in VAEs. It is shown to give better reconstruction quality and to learn good representations of data in the form of more informative latent codes. A method, named latent clipping, is introduced as a knob to control distance between samples. Samples can be pushed apart for tasks such as classification, or brought closer for smoother transition between different clusters of samples. Unlike prior work, the proposed method is robust to parameter tuning, and does not constraint encoder, or decoder structure. It can be used as a direct replacement for ELBO objective. Moreover, the proposed method is demonstrated to learn representations of data that can work well in downstream tasks such as classification. Applications of μ-VAE objective with more powerful decoders in various settings can be considered as a future work.

REFERENCES

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pp. 513–520, 2007.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. Introvae: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing Systems*, pp. 52–63, 2018.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

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

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.

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

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. In *33rd International Conference on Machine Learning (ICML 2016)*, 2016.

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

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

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3881–3890. JMLR. org, 2017.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Balancing learning and inference in variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5885–5892, 2019.

A Model architecture used for toy example of latent clipping

Table 2: Toy VAE Model for visualization of experiments

Function	**Layer**
Encoder	Fully Connected Layer 784x512 + Activation
	Fully Connected Layer 512x512 + Activation
	Fully Connected Layer 512x512 + Activation
Sampling	Linear layer with 2 units
Decoder	Fully Connected Layer 2x512 + Activation
	Fully Connected Layer 512x512 + Activation
	Fully Connected Layer 512x512 + Activation
	Fully Connected Layer 512x784 + Sigmoid
	*Activation functions are all either Leaky ReLu (alpha=0.2), ReLu, or Tanh as part of the experimentation.

B Model architecture, objective functions and details of training.

Optimization: In all experiments, learning rate of 1e-4 and batch size of 64 are used. Adam algorithm with high momentum ($\beta 1 = 0.9, \beta 2 = 0.999$) is used as optimizer. High momentum is chosen mainly to let most of previous training samples influence the current update step.

For reconstruction loss, mean square error, $\|x - x'\|^2$, is used for all cases. As for initialization, since the model consists of convolutional layers with Leaky ReLu in both encoder and decoder, Xavier initialization is used Glorot & Bengio (2010). Thus, initial weights are drawn from a Gaussian distribution with standard deviation (stdev) of $\sqrt{2/N}$, where N is number of nodes from previous layer. For example, for a kernel size of 3x3 with 32 channels, N = 288, which results in stdev of 0.083.

Objective functions are shown in Table 3, where μ-VAE objective is written explicitly to avoid any ambiguity in terms of how batch statistics are computed.

Table 4 shows model architecture as well as classifier used for all experiments. It consists of CNN-based encoder and decoder while classifier is three layer fully connected neural network. They all use Leaky Relu activation and learning rate of 1e-4.

Table 3: Objective functions[5]

Objective type	**Objective**
ELBO	$\mathrm{L}_e = \mathbb{E}_{q_\phi(z\|x)}\left[\log p_\theta(x\|z)\right] - \mathrm{KL}(q_\phi(z\|x)\\|p(z))$
β**-VAE**	$\mathrm{L}_\beta = \mathbb{E}_{q_\phi(z\|x)}\left[\log p_\theta(x\|z)\right] - \beta * \mathrm{KL}(q_\phi(z\|x)\\|p(z))$
μ**-VAE**	$\mathrm{L}_\mu = \mathbb{E}_{q_\phi(z\|x)}\left[\log p_\theta(x\|z)\right] - \frac{1}{B}\left[\lvert\sum_{i=1}^{B}\sum_{d=1}^{D}\mu_d^{(i)}\rvert + \sum_{i=1}^{B}\sum_{d=1}^{D}\left[\log \sigma^2\right]_d^{(i)}\right]$

[5]Note that each $\mathbb{E}_{q_\phi(z|x)}[.]$ and KL[.] is computed on expectation, $\mathbb{E}_{p_{data}(x)}[.]$, but it is not shown explicitly in the formulas above to make them more readable. Also, regularization term in μ-VAE is shown explicitly to emphasize how sample means are computed over batches.

Table 4: Model used for comparing VAE, β-VAE and μ-VAE

Function	**Layer**
Encoder	2D Conv 28x28x64 + Leaky Relu
	2D Conv 14x14x64 + Leaky Relu
	2D Conv 7x7x64 + Leaky Relu
	Linear layer with (7x7x64)x10
Sampling	10 Latent layer units
Decoder	Linear Layer 10x(7x7x64) + Leaky Relu
	2D DeConv 7x7x64 + Leaky Relu
	2D DeConv 14x14x32 + Leaky Relu
	2D DeConv 28x28x1 + Sigmoid
Classifier	Dense 10x1024 + Leaky Relu
	Dense 1024x1024 + Leaky Relu
	Dense 1024x1024 + Leaky Relu
	Dense 1024x10 + Softmax
	*All Leaky ReLu layers use alpha=0.2.

C MNIST RESULTS

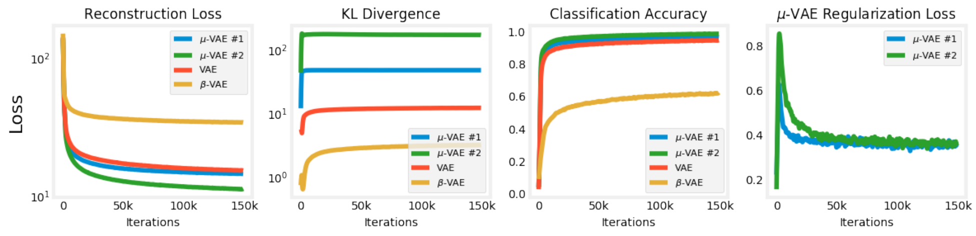

Figure 8: Training curves of the model trained on MNIST. Regularization loss of μ-VAE defined as $|\sum_{n=1}^{B} \mu_n| + \sum_{n=1}^{B} \left[\log \sigma^2\right]_n$, i.e. term that replaces KL.

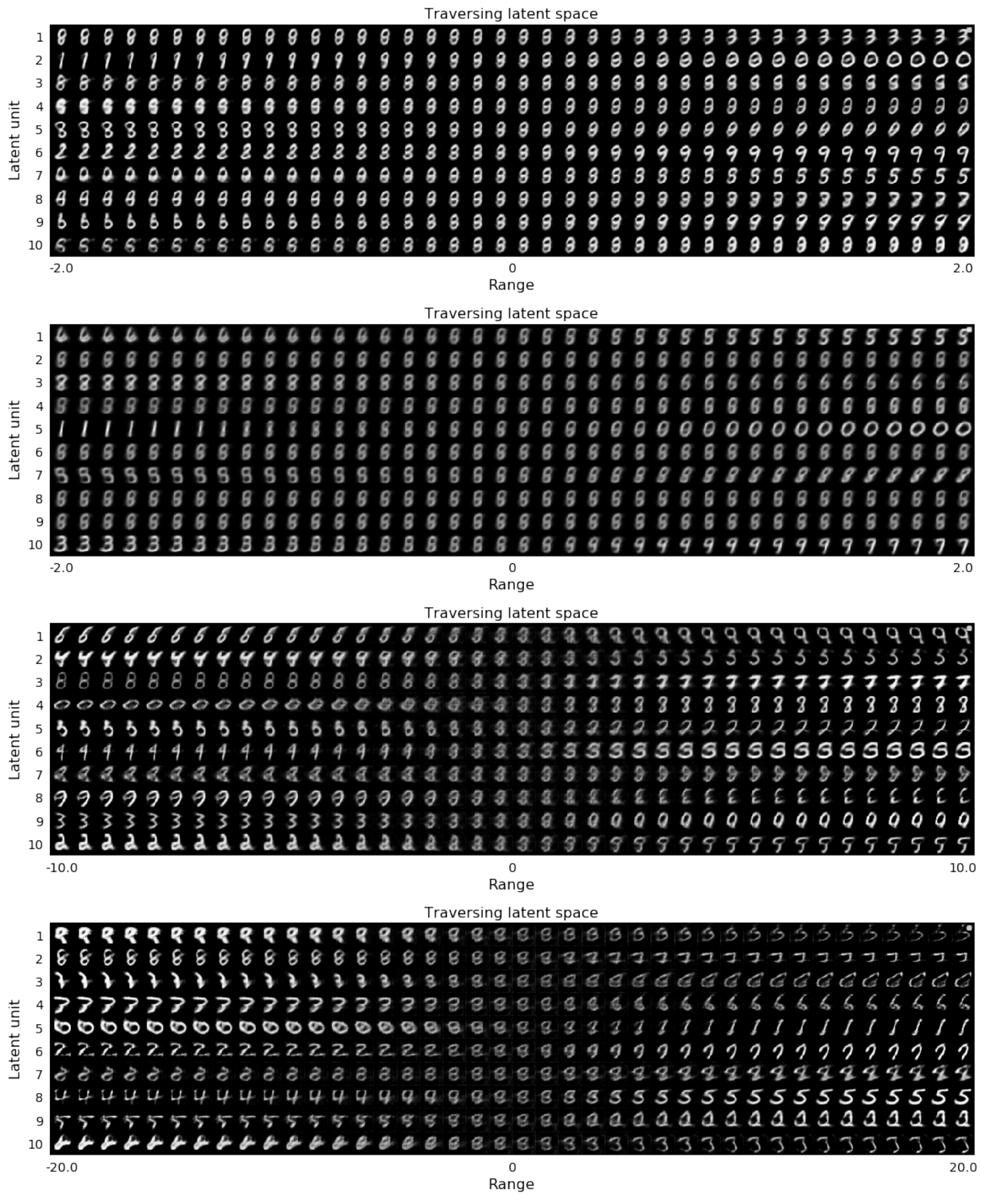

Figure 9: Traversing each dimension of latent variable in 40 steps. From top to bottom, model [range]: VAE [-2,2], β-VAE [-2,2], μ-VAE#1 [-10,10], and μ-VAE#2 [-20,20].

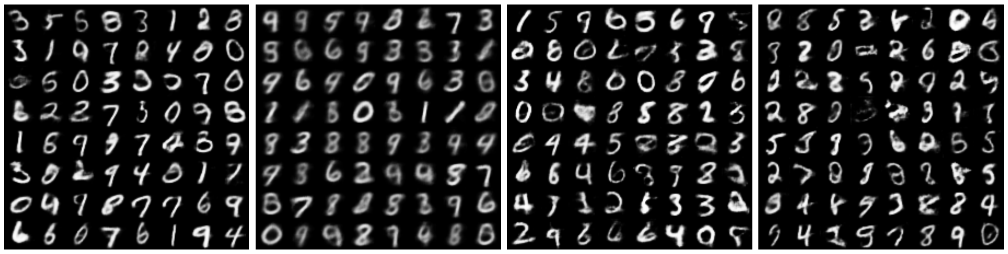

Figure 10: Random samples drawn from multi-variate Gaussian, N(0, σ).From left to right, model (σ): VAE (σ=1), β-VAE (σ=1), μ-VAE#1 (σ=3), and μ-VAE#2 (σ=3). Higger σ is used for μ-VAE models since their samples are more spread out.

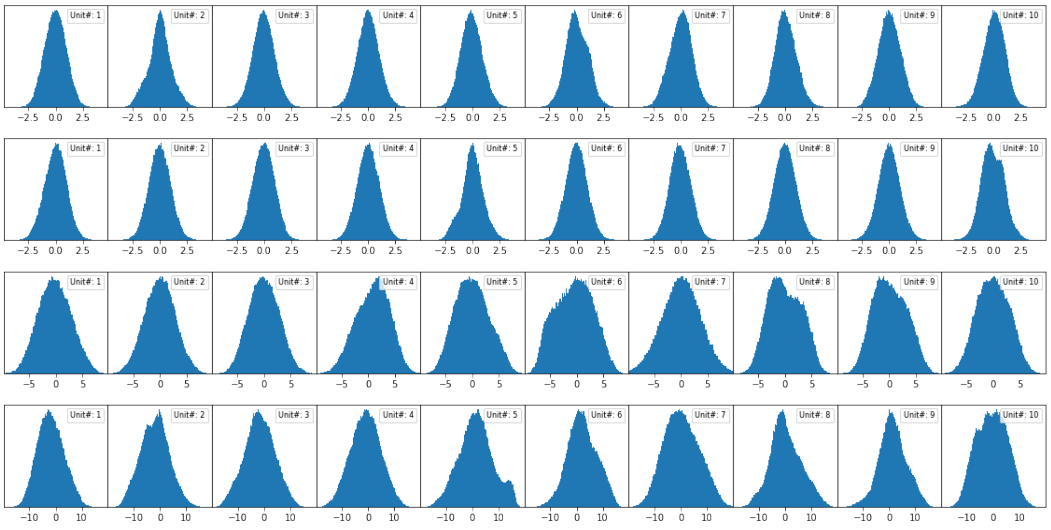

Figure 11: Sample distribution obtained at each dimension of latent variable using test dataset of MNIST. From top to bottom: VAE, β-VAE, μ-VAE#1, and μ-VAE#2.