Clustering using deep generative models

Yilun Li
bobliyilun.li@mail.utoronto.ca

Srinath Dama
srinath.dama@mail.utoronto.ca

Abstract

Clustering by definition is a way to organize unlabelled data. Conventional clustering methods are K-means and Gaussian Mixture Models (GMMs) for low dimensional data. With deep generative neural networks, learning the distribution of clusters for high dimensional datasets become possible along with additional benefit of generating better quality new data. This project explores the idea of using Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN) in clustering.

1 Introduction

Clustering using conventional GMM/K-means method would suffer from the curse of dimensionality with high dimensional input such as images; thus, the intuitive solution is to do dimensionality reduction (using PCA) pre-clustering [1]. Main drawback with this method is that the latent representations obtained in dimensionality reduction step are not adequate for meaningful clustering. Thus, we look into VAEs [2] and GANs [3], for robustness to the clustering as these models learns the distribution of latent variables. Clustering on VAE's latent dimension is natural as it contains an encoder/inference network; however, GANs do not have any. Therefore, most papers focus on VAE-based clustering, except ClusterGAN. This project builds on an existing implementation of ClusterGAN [4] and experimented on MNIST, Fashion-MNIST and CIFAR-10. Additionally, a standard VAE followed by GMM (VAE+GMM) was implemented for comparison.

2 Related Work

Tian et al. [5] trained an autoencoder network to learn the non-linear low dimensional feature vectors followed by applying K-means on the feature vectors for clutering. Song et al. [6] improves upon the two-step approach by integrating K-means loss on the latent representations so that training autoencoder would help learning cluster centric latent representations. In Deep Generative models, VAE network with Gaussian mixture as a prior on the latent variables is introduced in Variational Deep Embedding (VaDE) [7] for unsupervised clustering. Compared to a VAE, GAN has no inference network to use for clustering in latent representations. There are two components in solving this.

Discrete-Continuous Mixture and Relationship with DCGAN and InfoGAN: First, in DCGAN, the code vector/noise z can be generated from a uniform distribution[8]. Assuming one can recover the latent vector losslessly from images, these vectors might spread over the latent space uniformly if one adopts DCGAN's strategy[4]. The desired latent space should be interpretable and disentangled for clustering. InfoGAN introduced sampling from a discrete-continuous mixture[4][9]. More specifically, it uses a one-hot encoding vector to represent each cluster (e.g. digits 0-9 in MNIST) and a continuous vector as a source of noise for image generation[9]. However, InfoGAN only seeks to maximize the mutual information between code vectors and generated distributions, and is not optimized for clustering.

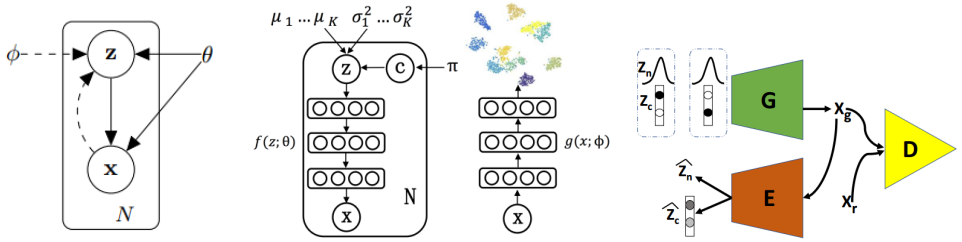
Inference Network/Encoder and Relationship with Bidirectional GAN(BiGAN): Now the second component, the answer to recover the latent vector from an image losslessly is attaching an encoder which maps from input space X to latent space Z[4]. Thus, ClusterGAN shares an almost identical architecture with BiGAN which tries to match the joint distribution between $(x, E(x))$ and

$(G(z), z)$ (x refers to a real image, z is a sampled noise vector). When BiGAN'S discriminator can no longer tell real tuples from fake ones, the joint distributions match and the encoder becomes an inverted generator[10]. ClusterGAN is aiming toward the same goal for its encoder. However, ClusterGAN's encoder takes in generated images from the generator, and tries to recover latent vectors from them directly, bypassing the discriminator which only takes in real and generated images instead of tuples of latent vectors and images.

3 Method

3.1 Baseline

Following the discussion in introduction, to overcome the issues with high dimensional inputs, PCA is used as a pre-processing step followed by applying GMM to the low dimensional representations. We refer to this method as PCA-GMM.



(a) VAE [2] (b) VaDE [7] (c) ClusterGAN [4]

Figure 1: Architectures

3.2 VAE+GMM

As discussed in introduction, VAE help to learn the latent variables distribution. This is achieved by addressing the intractability issue related to computing posterior $p(\mathbf{z}/\mathbf{x})$ by approximating it with a inference network $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. The VAE network, shown in Figure-1a, is trained by maximizing the ELBO loss [2] given as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \underbrace{\mathcal{L}(\theta, \boldsymbol{\phi}; \mathbf{x}^{(i)})}_{\text{ELBO}} = -\underbrace{\mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{KL term}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\Big]}_{\text{Reconstruction term}} \quad (1)$$

Where the reconstruction term helps maximizing the expectation of negative reconstruction error as in standard autoencoder network. Whereas, the KL term tries to make distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ close to the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$. We chose the prior as isotropic Gaussian as in VAE introduced by Kingma and Welling [2]. Final approximation of ELBO loss used in VAE training is given in Appendix-6.1. Once the VAE network is trained using the training data $\{\mathbf{x}^{(i)}\}_{i=1}^{N_{train}}$, we use the encoder to get the corresponding latent mean vectors $\{\boldsymbol{\mu}^{(i)}\}_{i=1}^{N_{train}}$ which in turn are used to fit a GMM model for clustering. As an extension, instead of using isotropic guassian as prior, GMM was used in VaDE as a prior distribution [7] to improve clustering. The architecture of VaDE is shown in Figure-1b. By using GMM as prior, encoder would be able to jointly learn categorical distribution as well as latent distribution corresponding to clusters. We find training VaDE network difficult as categorical probability for certain clusters is becoming negative after few epochs. Thus, our focus is shifted to ClusterGAN (Figure-1c).

3.3 Clustering using GAN

Let $z = (z_n, z_c)$ where z_n refers to a noise vector for image generation, z_c is a one-hot vector specifying a cluster. Let Encoder $E : X \rightarrow Z$, Generator $G : Z \rightarrow X$ where X is the image space and Z is the latent space, and let D be the discriminator. They are parameterized by neural networks

$\Theta_E, \Theta_G, \Theta_D$ respectively. The overall objective is

$$\min_{E,G} \max_{D} \mathbb{E}_{x\sim\mathbb{P}_x^r} D(x) - \mathbb{E}_{z\sim\mathbb{P}_z} D(G(z)) + \beta_n \mathbb{E}_{z\sim\mathbb{P}_z} \|z_n - E(G(z_n))\|_2^2 + \beta_c \mathbb{E}_{z\sim\mathbb{P}_z} \mathbb{H}(z_c, E(G(z_c))) \tag{2}$$

where $\mathbb{H}(z_c, E(G(z_c)))$ is cross-entropy loss, and β_n, β_c are coefficients that control the importance of the discrete and continuous portion of the latent space respectively. During training, encoder and generator are jointly trained and updated first. Let batchsize be m:

$$L(G, E) = \frac{1}{m}(\beta_n \sum_{i=1}^{m} \left\|z_n^i - E(G(z_n^i))\right\|_2^2 + \beta_c \sum_{i=1}^{m} \mathbb{H}(z_c^i, E(G(z_c^i)))) \tag{3}$$

$$L(D, G) = \frac{1}{m}(\sum_{i=1}^{m} D(x) - \sum_{i=1}^{m} D(G(z^i))) \tag{4}$$

At each time step t, sample $z^{(i)} = (z_n^i, z_c^i) \sim \mathbb{P}_z$. $z_n \sim \mathcal{N}(0, \sigma^2 I_{d_n})$ where $d_n = dim(z_n)$, and $z_c \sim K^{-1}$ where K is the number of clusters. Respectively, $\Theta_G, \Theta_E, \Theta_D$ are updated using Adam with $\nabla_{\Theta_G}[L(G, E) + L(D, G)], \nabla_{\Theta_E} L(G, E), \nabla_{\Theta_D} L(D, G)$. Using the encoder loss above, the objective is trying to recover the discrete-continuous latent vector that the generator started with. For the continuous z_n, mean squared error is used, and cross entropy is used for the discrete z_c. During this experiment, the loss function from DCGAN was employed as well as the loss function from ClusterGAN where $1 - \frac{1}{m}\sum_{i=1}^{m} D(G(z^i))$ was used in place of $-\frac{1}{m}\sum_{i=1}^{m} D(G(z^i))$ in generator and discriminator training. However, from their generated images, these models suffer from mode collapse and the clusters are entangled (Figure-6). Employing wasserstein loss resolves this issue as shown in Figure-4 [11].

4 Experiments

All clustering performances are evaluated using the metrics – clustering accuracy, Normalized Mutual Information (NMI) and Adjusted Rand index (ARI). Higher the metric value implies better quality of clustering.

Table 1: Clustering metrics

Dataset	Method	ACC	NMI	ARI
MNIST	Baseline (PCA-GMM)	0.67	0.66	0.53
	VAE+GMM	0.88	0.79	0.75
	ClusterGAN	**0.96**	**0.91**	**0.91**
F-MNIST	Baseline (PCA-GMM)	0.55	0.55	0.37
	VAE+GMM	0.60	0.58	0.42
	ClusterGAN	**0.63**	**0.63**	**0.49**
CIFAR-10	Baseline (PCA-GMM)	0.28*	0.13	0.09
	VAE+GMM	0.28*	0.14	0.08
	ClusterGAN	0.108	0.009	0.001

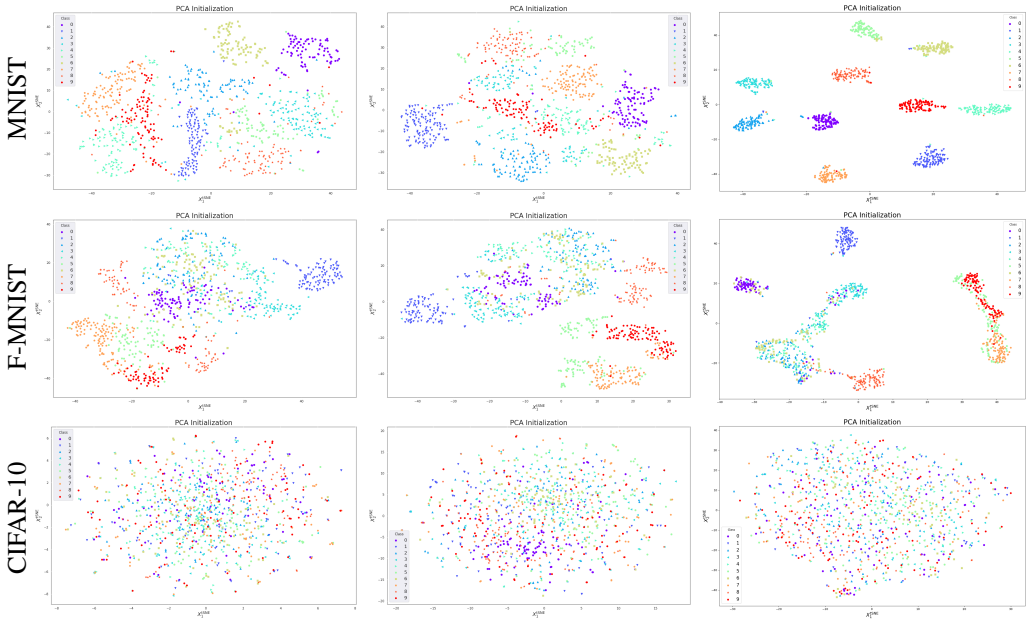
* - For CIFAR-10, cluster metrics for ClusterGAN are much stricter than other two methods. Appendix-6.3

Sensitivity analysis on MNIST: For MNIST and F-MNIST datasets, the encoder and generator convolutional networks used in VAE are same as those used for ClusterGAN and are given in Appendix-6.2. For these experiments we set the default batch size to 64 and learning rate to $1e^{-4}$. For different batch size, learning rate is multiplied by a factor of $batchsize/64$. During each hyper-parameter experiment, other hyperparameters are kept the same. For VAE+GMM sensitivity analysis, we choose latent dimension size and batch size as the hyper-parameters. Whereas, for ClusterGAN the hyper-parameters are the number of clusters, $dim(z_n)$, variance σ^2 in z_n's distribution and batch size. Results and analysis of these experiments are presented in Appendix-6.3.

Comparison between VAE and ClusterGAN's Performances: Overall best performance results across the datasets are shown in Table-1. We can notice that ClusterGAN outperforms VAE+GMM

method on MNIST and FASHION-MNIST datasets. Whereas, VAE+GMM outperforms baseline method on these two datasets. We can also see this trend reflecting in t-SNE plots in Figure-2, where ClusterGAN has more concentrated clusters (the sample sizes are all 1000). This meets our expectation. Here, we compare VAE and ClusterGAN. The KL term in VAE loss is functionally analogous to $L(G, E)$ in ClusterGAN. Both algorithms are trying to close the gap between the output distributions of their encoders (e.g. $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ in VAE, the distribution for $E(G(z_n, z_c))$ in ClusterGAN) and the input/prior distribution to their generators (e.g. $p_{\boldsymbol{\theta}}(\mathbf{z})$ in VAE, $\mathbb{P}_z$ in ClusterGAN). However, VAE assumes Gaussian or Bernoulli observation model for $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ which might not always be the case. GANs make no such assumptions. Instead, it has a discriminator. Its main objective's $L(D, G)$ loss term aims to match generated images' distribution with real images' distribution. This contributes massively to GAN's superior performance with additional advantage of discrete-continuous mixture ClusterGAN samples from.

Limitation on CIFAR-10 dataset: For CIFAR-10 dataset, larger models with more convolution layers were implemented for both algorithms(Table-3). Authors of ClusterGAN show that the model is clustering images based on their background colors; however, they did not provide any accuracy scores [4]. We were unable to observe the background-based clustering, possibly due to our limit in compute (Figure-7).From Figure-2, for Fashion-MNIST and MNIST, using baseline creates some resemblance of clustering for both datasets; furthermore, such clustering only becomes more and more concentrated for VAE and GAN. However, for CIFAR-10, all 3 methods have a uniformly spread out latent space. Our explanation is that due to CIFAR-10's intra-class variability and common features(e.g. blue background for ships and planes) among each class, it is difficult for these models to find class dicriminative features needed to create clustering that matches the given categories.



(a) Baseline (b) VAE (c) ClusterGAN

Figure 2: t-SNE plots for the best performing experiments

5 Summary

This project has demonstrated and explained the superior performance of ClusterGAN over VAE+GMM method. We also have shown VAE+GMM and ClusterGAN's sensitivity to hyper-parameters and ClusterGAN's proclivity to mode collapse which can be fixed by Wasserstein Loss. We've also shown and analyzed both model's weakness on natural image sets like CIFAR-10. For future work, it would be interesting to combine discrete-continuous mixture sampling with BiGAN introduced in related work as well as VAE networks with similar mixture of discrete-continuous latent variable such as GMM as prior distribution [12].

References

[1] C. Bouveyron, S. Girard, and C. Schmid. "High-dimensional data clustering". In: *Comput. Stat. Data Anal.* 52.1 (2007), pp. 502–519.

[2] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]* (May 2014). arXiv: 1312.6114.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Networks". In: *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv: 1406.2661.

[4] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan. "ClusterGAN : Latent Space Clustering in Generative Adversarial Networks". In: *CoRR* abs/1809.03627 (2018).

[5] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. "Learning Deep Representations for Graph Clustering". In: (), p. 7.

[6] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan. "Auto-encoder based data clustering". In: *Iberoamerican congress on pattern recognition*. Springer. 2013, pp. 117–124.

[7] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. *Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering*. 2017.

[8] A. Radford, L. Metz, and S. Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016.

[9] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[10] J. Donahue, P. Krähenbühl, and T. Darrell. "Adversarial Feature Learning". In: *CoRR* abs/1605.09782 (2016).

[11] M. Arjovsky, S. Chintala, and L. Bottou. *Wasserstein GAN*. 2017.

[12] V. Prasad, D. Das, and B. Bhowmick. "Variational Clustering: Leveraging Variational Autoencoders for Image Clustering". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–10.

Contributions

- **Yilun Li** contributed majorly with reprodcuing ClusterGAN results and running experiments on ClusterGAN for sensitivity analysis. **Yilun Li** also implemented the ClusterGAN model for Cifar-10.
- **Srinath Dama** contributed majorly with setting up Baseline method, VAE+GMM method and doing sensitivity analysis on VAE+GMM method.
- Both **Yilun** and **Srinath** equally contributed in the following areas – project proposal, discussions on related papers, setting up experiments and debugging issues, setting up and experimenting with VaDE network and final project report.
- **Muneeb Ansari** used to attend team meeting and gave some inputs during project proposal. However, he didn't write any part of the proposal. He claimed that he conducted initial research; however, he did not provide any feedback on what paper he specifically read or his reflection on these research. All literature found in reference were scoured by **Yilun** and **Srinath**. He did try to modify one tsne visualization script from ClusterGAN for VAE; however, it was not tested and was ultimately completed by **Srinath**.

6 Appendix

6.1 VAE loss equation

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \underbrace{\mathcal{L}(\theta, \boldsymbol{\phi}; \mathbf{x}^{(i)})}_{\text{ELBO}} = -\underbrace{\mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{KL term}} + \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})\Big]}_{\text{Reconstruction term}} \tag{5}$$

For univariate Gaussian prior case, $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ and posterior approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})$ also being Gaussian. -KL term can be expanded as follows [2]

$$-\mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) \tag{6}$$

where J is the dimensionality of latent variable $\mathbf{z}$, μ_j, σ_j are the j-th elements of variational mean and standard deviation vectors evaluated for $\mathbf{x}^{(i)}$.

When $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is assumed to Gaussian, reconstruction term would be negative mean square error between generator output image ($\tilde{\mathbf{z}}$) and input image ($\mathbf{x}$). For training VAE, we assumed the image data (MNIST, F-MNIST and CIFAR-10) falls under Gaussian distribution. Whereas in the case of $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ being Bernoulli distribution, reconstruction term is binary cross entropy between between generator output image ($\tilde{\mathbf{z}}$) and input image ($\mathbf{x}$).

6.2 Network architecture

ClusterGAN network architecture given in ClusterGAN[4] is used as it is. Whereas, for VAE, we modified the first layer in Generator and last layer in Encoder as mentioned in the Tables-2, 3. Network architecture for MNIST and Fashion-MNIST are given in below Table-2. For CIFAR-10, a slightly large capacity network given in Table-3 is used.

Table 2: VAE and ClusterGAN Network architectures for MNIST and Fashion-MNIST

Generator	Encoder	Discriminator
GAN: Input $z = (z_n; z_c) \in \mathbb{R}^{n+10}$ (VAE: Input $z \in \mathbb{R}^n$)	Input $X \in \mathbb{R}^{28\times 28}$	Input $X \in \mathbb{R}^{28\times 28}$
FC 1024, ReLU, BN	4×4 conv, 64, stride 2, LReLU	4×4 conv, 64, stride 2, LReLU
FC $7 \times 7 \times 128$, ReLU, BN	4×4 conv, 128, stride 2, LReLU	4×4 conv, 128, stride 2, LReLU
4×4 upconv, 64, stride 2, ReLU, BN	FC 1024, LReLU	FC 1024, LReLU
4×4 upconv, 1, stride 2, Sigmoid	FC $n + 10$ linear on $\tilde{z}$ Softmax on last 10 to obtain $\tilde{z}_c$ (VAE: FC $2n$ linear softplus on last n to obtain σ)	FC 1, linear

where $n = 30$ for MNIST and $n = 40$ for Fashion-MNIST dataset

Table 3: VAE and ClusterGAN Network architectures for CIFAR-10

Generator	Encoder	Discriminator
GAN: Input $z = (z_n; z_c) \in \mathbb{R}^{n+10}$ (VAE: Input $z \in \mathbb{R}^n$)	Input $X \in \mathbb{R}^{32\times32}$	Input $X \in \mathbb{R}^{32\times32}$
FC $2 \times 2 \times 448$, ReLU, BN	4×4 conv, 64, stride 2, LReLU	4×4 conv, 64, stride 2, LReLU
4×4 upconv, 256, stride 2, ReLU	4×4 conv, 128, stride 2, LReLU, BN	4×4 conv, 128, stride 2, LReLU, BN
4×4 upconv, 128, stride 2, ReLU	4×4 conv, 256, stride 2, LReLU, BN	4×4 conv, 256, stride 2, LReLU, BN
4×4 upconv, 64, stride 2, ReLU	4×4 conv, 512, stride 2, LReLU, BN	4×4 conv, 512, stride 2, LReLU, BN
4×4 upconv, 3, stride 2, Sigmoid	FC $n + 10$ linear on $\tilde{z}$ Softmax on last 10 to obtain $\tilde{z}_c$ (VAE: FC $2n$ linear softplus on last n to obtain σ)	FC 1, linear

where $n = 50$ for CIFAR dataset

6.3 Sensitivity analysis

VAE+GMM Sensitivity analysis

In baseline and VAE+GMM, most frequent truth label of a mode/cluster is used for evaluating cluster accuracy.

Table 4: VAE+GMM Sensitivity to latent dimension size on MNIST

latent dimension size	Acc	NMI	ARI
5	0.78	0.75	0.67
10	0.80	0.75	0.64
20	0.78	0.70	0.69
30	0.74	0.71	0.62
50	0.76	0.72	0.60

Batch size is 64, Learning rate is 0.0001

Table 5: VAE+GMM Sensitivity to batch size on MNIST

latent dimension size	Acc	NMI	ARI
32	0.73	0.71	0.61
64	0.72	0.58	0.70
128	0.85	0.77	0.70
256	0.85	0.79	0.74
1024	0.88	0.79	0.75

Latent dimension size is set to 10, Learning rate is $0.0001 * (batchsize)/64$

The hyperparameters chosen for sensitivity analysis are the dimension of latent variable (z) and batch size. Table-4 shows that clustering performance is slightly effected by latent dimension and maximum clustering performance is achieved with dimension 10. Whereas, from Table-5, we can see that clustering performance is highly effected by batch size and learning rate. Best performance is achieved using batch size of 1024.

ClusterGAN Sensitivity analysis

Table 6: Cluster GAN Sensitivity to z_n dimension size on MNIST

z_n dimension size	Mode ACC	Reconstruction ACC	Cluster ACC	NMI	ARI
10	0.9067	0.8822	0.8606	0.8330	0.7912
20	0.9829	0.9258	0.9284	0.8629	0.8506
30	0.9833	0.9388	0.9354	0.8719	0.8643
50	0.9041	0.5702	0.5768	0.5098	0.3356

Variance for z_n is 0.75, Number of Clusters is 10, Batch size is 64

Table 7: Cluster GAN Sensitivity to z_n distribution variance on MNIST

z_n distribution variance	Mode ACC	Reconstruction ACC	Cluster ACC	NMI	ARI
0.1	0.9954	0.9564	0.9508	0.8976	0.8956
0.5	0.9099	0.9010	0.8868	0.8261	0.7924
0.75	0.9833	0.9388	0.9354	0.8719	0.8643
1	0.8016	0.7146	0.5978	0.5995	0.4352

Dimension for z_n is 30, Number of Clusters is 10, Batch size is 64

Table 8: Cluster GAN Sensitivity to batch size on MNIST

Batch size	Mode ACC	Reconstruction ACC	Cluster ACC	NMI	ARI
32	0.9947	0.9622	0.9596	0.9128	0.9137
64	0.9954	0.9564	0.9508	0.8976	0.8956
128	0.8772	0.9130	0.8118	0.7868	0.7225
256	0.8348	0.8956	0.7108	0.7258	0.6231
1024	0.6408	0.8196	0.3842	0.2881	0.208

Dimension for z_n is 30, Number of Clusters is 10, Variance for z_n is 0.1, Learning rate is $1e^{-4} * (batchsize)/64$

Table 9: Cluster GAN Sensitivity to number of clusters(z_c's dimension) on MNIST

number of clusters	Mode ACC	Reconstruction ACC	Cluster ACC	NMI	ARI
7	0.6757	0.9076	0.6318	0.7120	0.5635
9	0.8614	0.9474	0.846	0.8358	0.7866
10	0.9954	0.9564	0.9508	0.8976	0.8956
11	0.8949	0.9432	0.8882	0.8365	0.7699
13	0.8457	0.9398	0.8552	0.7903	0.7158

Dimension for z_n is 30, Variance for z_n is 0.1, batch size is 64

Since learning is unsupervised, each z_c vector may correspond to a class that is not the same as the mode/index of the one hot vector. Thus, we generate images from each mode k specified by z_c, then pass the generated images to a classifier to get a label $\tilde{y}$. The most frequent $\tilde{y}$ computed from each mode k defines **mode accuracy** as well as the mapping from mode k to that class. Technically, this is more akin to the cluster accuracy for VAE.

A classifier was trained and used on MNIST and Fashion-MNIST with 99.2% and 93% accuracy respectively. A pretrained Resnet20 was also used as classifier on CIFAR 10 with 92.6% top-1

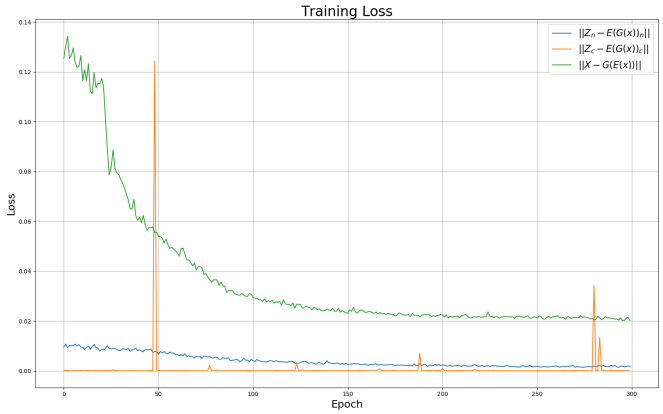
accuracy. Then we try to reconstruct images from real ones by passing them to encoder then generator like an autoencoder. We apply the classifier on the reconstructed images to get labels for comparison with the ground truths. This defines **reconstruction accuracy**. Class labels of real images should have their corresponding z_c mode coming from the mapping mentioned in mode accuracy. **Cluster accuracy**, NMI, and ARI are to measure the differences between actual z_c vectors when real images are classified by the encoder and the expected mode of these images [4]. This is a much stricter metric as a high cluster accuracy also needs an effective (almost bijective) mapping between modes and classes.

Table-7 shows the model is also not robust against z_n's distribution variance with $\sigma^2 = 0.1$ or 0.75 being the better performing models. Ideally, z_n vector is intended as a way to specify the quality of the generated image(e.g stroke thickness and so on), and it should not interfere clustering performance since z_c specifies the clusters unless each class in MNIST contains certain unique qualities which have to be captured by z_n and cannot be sufficiently represented by the one-hot encoding z_c. This also partially contributes to why models with higher cluster purity are has better reconstruction accuracy.

Table-8 shows that ClusterGAN is fairly sensitive to batchsize and learning rate. One can see that for batchsize 32 and 64, the model is performing well, but it starts to decrease for larger batchsize. It may be caused by ill conditioned loss landscape. It's worth noting that batchsize 32 yields better results than the paper's result. Table-9 shows that clustering is most accurate when the number of clusters matches the number of classes in MNIST. The model is more sensitive to forcing less clusters (7 and 9) than forcing more clusters (11 and 13).

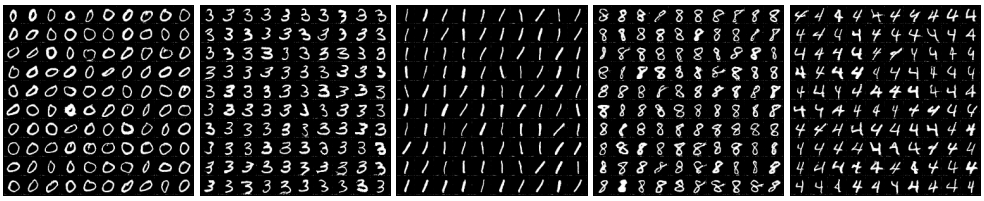
6.4 Loss Plots and Generated images by ClusterGAN

(a) Model loss plot



(b) Recovery loss plot between latent vector $\tilde{z} = (\tilde{z_n}, \tilde{z_c})$ recovered by encoder and input $z = (z_n, z_c)$ of the generator. As one can see, recovery loss is small.

Figure 3: ClusterGAN loss plots on MNIST using the best performing configuration above: 32 for batchsize, 0.1 for variance of z_n's distribution, $dim(z_n) = 30$, 10 clusters.

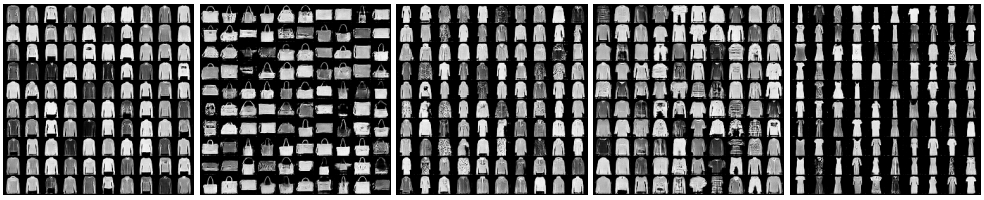


(a) mode 0 (b) mode 1 (c) mode 2 (d) mode 3 (e) mode 4

(f) mode 5 (g) mode 6 (h) mode 7 (i) mode 8 (j) mode 9

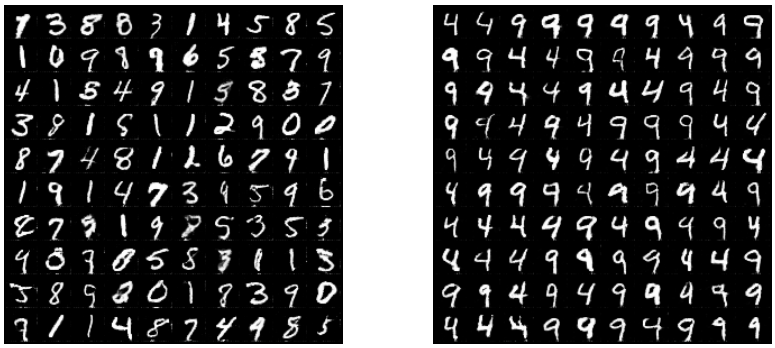
Figure 4: Distinct MNIST digits Generated by different modes(z_c latent vectors) in ClusterGAN. Using Wasserstein loss, a bijective mapping between modes and actual classes in the dataset has been achieved. No more mode collapses and entanglements.

(a) mode 0 (b) mode 1 (c) mode 2 (d) mode 3 (e) mode 4



(f) mode 5 (g) mode 6 (h) mode 7 (i) mode 8 (j) mode 9

Figure 5: Fashion MNIST categories generated by different modes(z_c latent vectors) in ClusterGAN



(a) Mode collapse example with default ClusterGAN loss function

(b) Mode collapse example with DCGAN loss function

Figure 6: Examples of mode collapses and class entanglement in ClusterGAN due to choices of loss functions.

(a) mode 0 (b) mode 1 (c) mode 2

(d) mode 3 (e) mode 4 (f) mode 5

(g) mode 6 (h) mode 7 (i) mode 8

(j) mode 9

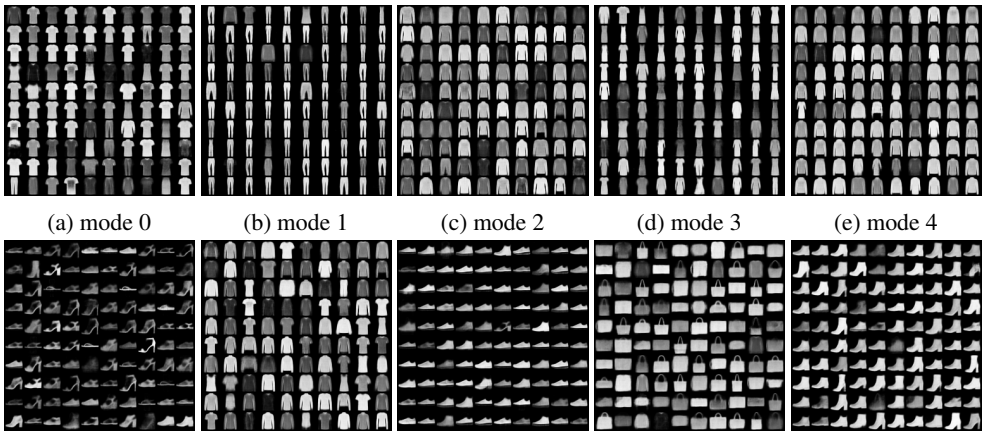
Figure 7: Images generated from each mode in ClusterGAN model on CIFAR-10

6.5 Image Reconstructed By VAE

(a) mode 0 (b) mode 1 (c) mode 2 (d) mode 3 (e) mode 4

(f) mode 5 (g) mode 6 (h) mode 7 (i) mode 8 (j) mode 9

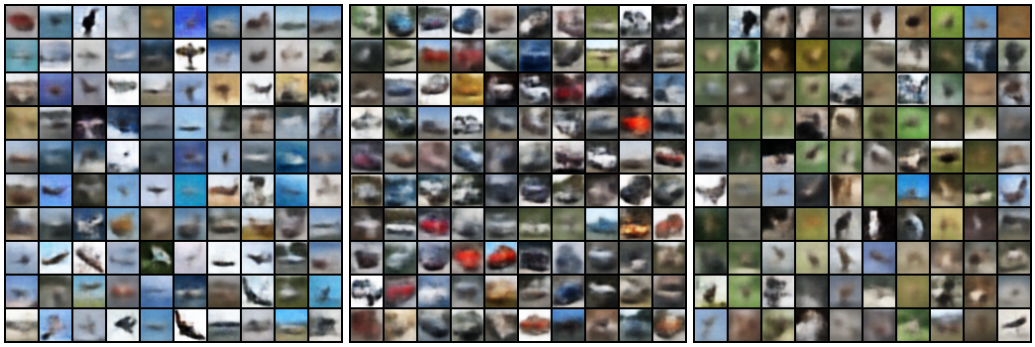
Figure 8: Distinct MNIST digits reconstructed by VAE network, Batch size 1024, latent dimension is 10



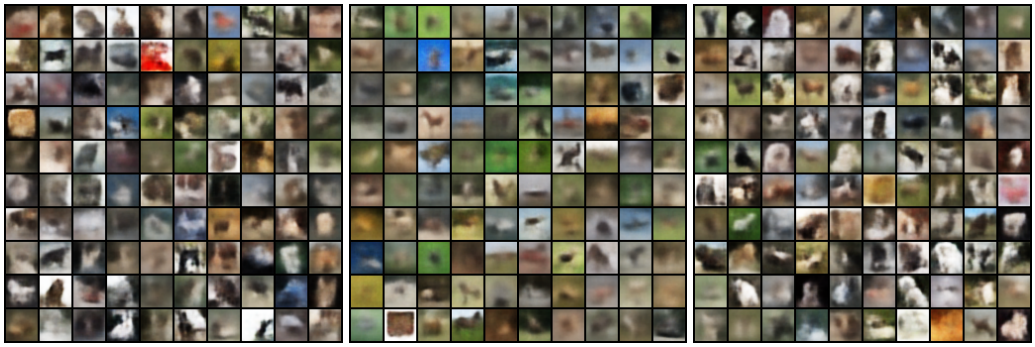
(a) mode 0 (b) mode 1 (c) mode 2 (d) mode 3 (e) mode 4

(f) mode 5 (g) mode 6 (h) mode 7 (i) mode 8 (j) mode 9

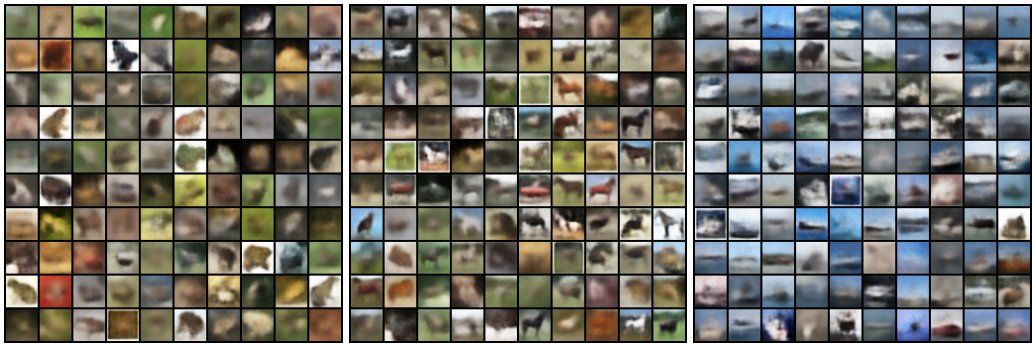
Figure 9: Distinct Fashion-MNIST categories reconstructed by VAE network, Batch size 64, latent dimension is 40



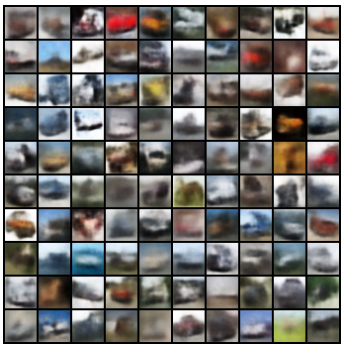
(a) class 0 (b) class 1 (c) class 2



(d) class 3 (e) class 4 (f) class 5



(g) class 6 (h) class 7 (i) class 8



(j) class 9

Figure 10: Samples images from each class reconstructed by VAE model on CIFAR-10