ClusterGAN: Latent Space Clustering in Generative Adversarial Networks

Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, Sreeram Kannan

{sudipm, asnani, lines, ksreeram}@uw.edu
Department of Electrical and Computer Engineering,
University of Washington, Seattle, WA-98195.

Abstract

Generative Adversarial networks (GANs) have obtained remarkable success in many unsupervised learning tasks and unarguably, clustering is an important unsupervised learning problem. While one can potentially exploit the latent-space back-projection in GANs to cluster, we demonstrate that the cluster structure is not retained in the GAN latent space. In this paper, we propose ClusterGAN as a new mechanism for clustering using GANs. By sampling latent variables from a mixture of one-hot encoded variables and continuous latent variables, coupled with an inverse network (which projects the data to the latent space) trained jointly with a clustering specific loss, we are able to achieve clustering in the latent space. Our results show a remarkable phenomenon that GANs can preserve latent space interpolation across categories, even though the discriminator is never exposed to such vectors. We compare our results with various clustering baselines and demonstrate superior performance on both synthetic and real datasets.

1 Introduction

1.1 Motivation

Representation learning enables machine learning models to decipher underlying semantics in data and disentangle hidden factors of variation. These powerful representations have made it possible to transfer knowledge across various tasks. But what makes one representation better than another ? (Bengio, Courville, and Vincent 2013) mentioned several general-purpose priors that are not dependent on the downstream task, but appear as commonalities in good representations. One of the general-purpose priors of representation learning that is ubiquitous across data intensive domains is clustering. Clustering has been extensively studied in unsupervised learning with multifarious approaches seeking efficient algorithms (Ng, Jordan, and Weiss 2002), problem specific distance metrics (Xiang, Nie, and Zhang 2008), validation (Halkidi, Batistakis, and Vazirgiannis 2001) and the like. Even though the main focus of clustering has been to separate out the original data into classes, it would be even nicer if such clustering was obtained along with dimensionality reduction where the real data actually seems to come from a lower dimensional manifold.

In recent times, much of unsupervised learning is driven by deep generative approaches, the two most prominent being Variational Autoencoder (VAE) (Kingma and Welling 2013) and Generative Adversarial Network (GAN) (Goodfellow et al. 2014). The popularity of generative models themselves is hinged upon the ability of these models to capture high dimensional probability distributions, imputation of missing data and dealing with multimodal outputs. Both GAN and VAE aim to match the real data distribution (VAE using an explicit approximation of maximum likelihood and GAN through implicit sampling), and simultaneously provide a mapping from a latent space $\mathcal{Z}$ to the input space $\mathcal{X}$. The latent space of GANs not only provides dimensionality reduction, but also gives rise to novel applications. Perturbations in the latent space could be used to determine adversarial examples that further help build robust classifiers (Ilyas et al. 2017). Compressed sensing using GANs (Bora et al. 2017) relies on finding a latent vector that minimizes the reconstruction error for the measurements. Generative compression is yet another application involving $\mathcal{Z}$ (Santurkar, Budden, and Shavit 2017). One of the most fascinating outcomes of the GAN training is the interpolation in the latent space. Simple vector arithmetic properties emerge which when manipulated lead to changes in the semantic qualities of the generated images (Radford, Metz, and Chintala 2015). This differentiates GANs from traditional dimensionality reduction techniques (Mika et al. 1999) (Maaten and Hinton 2008) which lack interpretability. One potential application that demands such a property is clustering of cell types in genomics. GANs provide a means to understand the change in high-dimensional gene expression as one traverses from one cell type (i.e., cluster) to another in the latent space. Here, it is critical to have both clustering as well as good interpretability and interpolation ability. This brings us to the principal motivation of this work: ***Can we design a GAN training methodology that clusters in the latent space?***

1.2 Related Works

Deep learning approaches have been used for dimensionality reduction starting with variants of the autoencoder such as the stacked denoising autoencoders (Vincent et al. 2010), sparse autoencoder (Coates, Ng, and Lee 2011) and deep CCA (Andrew et al. 2013). Architectures for deep unsupervised subspace clustering have also been built on the
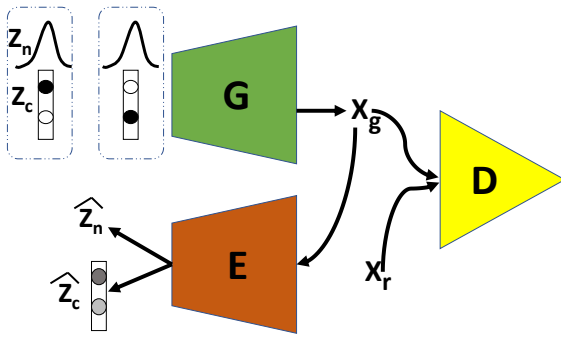
Figure 1: ClusterGAN Architecture

encoder-decoder framework (Ji et al. 2017). Recent works have addressed this problem of joint clustering and dimensionality reduction in autoencoders. (Xie, Girshick, and Farhadi 2016) solved this problem by initializing the cluster centroids and the embedding with a stacked autoencoder. Then they use alternating optimization to improve the clustering and report state-of-the-art results in both clustering accuracy and speed on real datasets. The clustering algorithm is referred to as DEC in their paper. Since K-means is often the most widely used algorithm for clustering, (Yang et al. 2017) improved upon DEC by introducing a modified cost function that incorporates the K-means loss. They optimized the non-convex objective using alternating SGD to obtain an embedding that is amenable to K-means clustering. Their algorithm DCN was shown to outperform all standard clustering methods on a range of datasets. It is interesting to note that the vanilla autoencoder by itself did not explicitly have any clustering objective. But it could be improved to achieve this end by careful algorithmic design. Since GANs have outperformed autoencoders in generating high fidelty samples, we had a strong intuition in favour of the powerful latent representations of GAN providing improved clustering performance also.

Interpretable representation learning in the latent space has been investigated for GANs in the seminal work of (Chen et al. 2016). The authors trained a GAN with an additional term in the loss that seeks to maximize the mutual information between a subset of the generator's noise variables and the generated output. The key goal of InfoGAN is to create interpretable and disentangled latent variables. While InfoGAN does employ discrete latent variables, it is not specifically designed for clustering. In this paper, we show that our proposed architecture is superior to InfoGAN for clustering. The other prominent family of generative models, VAE, has the additional advantage of having an inference network, the encoder, which is jointly learnt during training. This enables mapping from $\mathcal{X}$ to $\mathcal{Z}$ that could potentially preserve cluster structure by suitable algorithmic design. Unfortunately, no such inference mechanism exists in GANs, let alone the possibility of clustering in the latent space. To bridge the gap between VAE and GAN, various methods such as Adversarially Learned Inference (ALI) (Dumoulin et al. 2016), Bidirectional Generative Adversarial Networks (BiGAN) (Don-

ahue, Krähenbühl, and Darrell 2016) have introduced an inference network which is trained to match the joint distributions of (x, z) learnt by the encoder $\mathcal{E}$ and decoder $\mathcal{G}$ networks. Typically, the reconstruction in ALI/BiGAN is poor as there is no deterministic pointwise matching between x and $\mathcal{G}(\mathcal{E}(x))$ involved in the training. Architectures such as Wasserstein Autoencoder (Tolstikhin et al. 2017), Adversarial Autoencoder (Makhzani et al. 2015), which depart from the traditional GAN framework, also have an encoder as part of the network. So this led us to consider a formulation using an Encoder which could *both reduce the cycle loss as well as aid in clustering.*

1.3 Main Contributions

To the best of our knowledge, this is the first work that addresses the problem of clustering in the latent space of GAN. The main contributions of the paper can be summarized as follows:

- We show that even though the GAN latent variable preserves information about the observed data, the latent points are smoothly scattered based on the latent distribution leading to no observable clusters.

- We propose three main algorithmic ideas in ClusterGAN in order to remedy this situation.

  1. We utilize a **mixture of discrete and continuous** latent variables in order to create a non-smooth geometry in the latent space.
  2. We propose a **novel backpropagation algorithm** accommodating the discrete-continuous mixture, as well as an **explicit inverse-mapping network** to obtain the latent variables given the data points, since the problem is non-convex.
  3. We propose to jointly train the GAN along with the inverse-mapping network with **a clustering-specific loss** so that the distance geometry in the projected space reflects the distance-geometry of the variables.

- We compare ClusterGAN and other possible GAN based clustering algorithms, such as InfoGAN, along with multiple clustering baselines on varied datasets. This demonstrates the superior performance of ClusterGAN for the clustering task.

- We demonstrate that ClusterGAN surprisingly retains good interpolation across the different classes (encoded using one-hot latent variables), even though the discriminator is never exposed to such samples.

The formulation is general enough to provide a *meta* framework that incorporates the additional desirable property of clustering in GAN training.

2 Discrete-Continuous Prior

2.1 Background

Generative adversarial networks consist of two components, the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$. Both $\mathcal{G}$ and $\mathcal{D}$ are usually implemented as neural networks parameterized by Θ_G and Θ_D respectively. The generator can also be considered to be a mapping from latent space to the data space
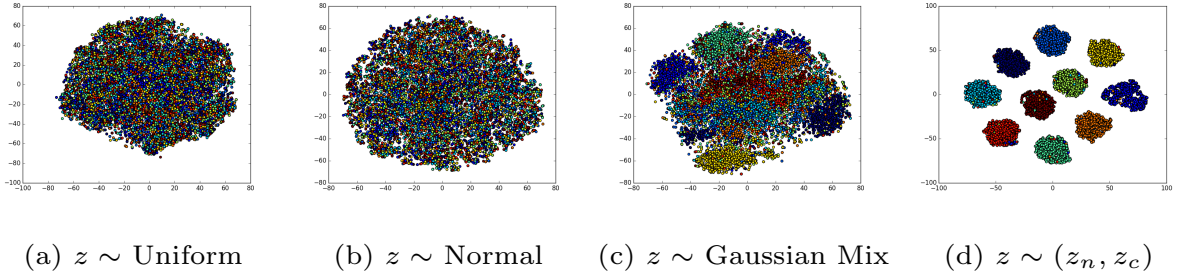
(a) $z \sim$ Uniform (b) $z \sim$ Normal (c) $z \sim$ Gaussian Mix (d) $z \sim (z_n, z_c)$

Figure 2: TSNE visualization of latent vectors for GANs trained with different priors on MNIST.

(a) Non-linear generator with $z \sim \mathcal{N}(0, I)$ (b) Linear generator with $z \sim$ one-hot encoded normal.
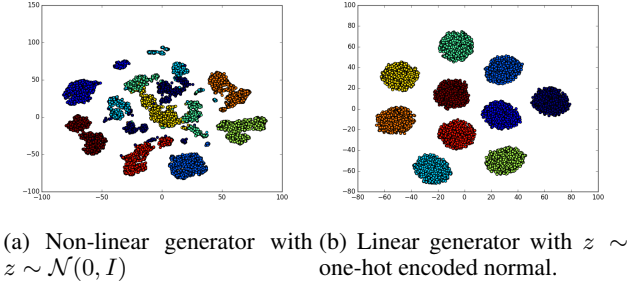
Figure 3: TSNE visualization of latent vectors. Linear Generator recovers clusters, suggesting that representation power is not a bottleneck.

which we denote as $\mathcal{G} : \mathcal{Z} \mapsto \mathcal{X}$. The discrimator defines a mapping from the data space to a real value which can correspond to the probability of the sample being real, $\mathcal{D} : \mathcal{X} \mapsto \mathbb{R}$. The GAN training sets up a two player game between $\mathcal{G}$ and $\mathcal{D}$, which is defined by the minimax objective : $\min_{\Theta_G} \max_{\Theta_D} \mathbf{E}_{x \sim \mathbb{P}_x^r} q(\mathcal{D}(x)) + \mathbf{E}_{z \sim \mathbb{P}_z} q(1 - \mathcal{D}(\mathcal{G}(z)))$, where $\mathbb{P}_x^r$ is the distribution of real data samples, $\mathbb{P}_z$ is the prior noise distribution on the latent space and $q(.)$ is the quality function. For vanilla GAN, $q(x) = \log x$, and for Wasserstein GAN (WGAN) $q(x) = x$. We also denote the distribution of generated samples x_g as $\mathbb{P}_x^g$. The discriminator and the generator are optimized alternatively so that at the end of training $\mathbb{P}_x^g$ matches $\mathbb{P}_x^r$.

2.2 Vanilla GAN does not cluster well in the latent space

One possible way to cluster using a GAN is to backpropagate the data into the latent space (using backpropagation decoding (Lipton and Tripathi 2017)) and cluster the latent space. However, this method usually leads to very bad results (see Fig. 2 for clustering results on MNIST). The key reason is that, if indeed, back-propagation succeeds, then the back-projected data distribution should look similar to the latent space distribution, which is typically chosen to be a Gaussian or uniform distribution, and we cannot expect to cluster in that space. Thus even though the latent space may contain full information about the data, the distance geometry in the latent space does not reflect the inherent clustering. In (Gurumurthy, Sarvadevabhatla, and Babu

2017), the authors sampled from a Gaussian mixture prior and obtained diverse samples even in limited data regimes. However, even GANs with a Gaussian mixture failed to cluster, as shown in 2(c). As observed by the authors of DeLiGAN, Gaussian components tend to 'crowd' and become redundant. Lifting the space using categorical variables could solve the problem effectively. But categorical variables might introduce discontinuity. Continuity in latent space has been traditionally viewed to be a pre-requisite for the objective of good interpolation. In other words, interpolation seems to be at loggerheads with the clustering objective. We demonstrate in this paper how ClusterGAN can obtain **good interpolation and good clustering** simultaneously.

2.3 Sampling from Discrete-Continuous Mixtures

In ClusterGAN, we sample from a prior that consists of normal random variables cascaded with one-hot encoded vectors. To be more precise $z = (z_n, z_c), z_n \sim \mathcal{N}(0, \sigma^2 I_{d_n}), z_c = e_k, k \sim \mathcal{U}\{1, 2, \ldots, K\}$, e_k is the kth elementary vector in $\mathbb{R}^K$ and K is the number of clusters in the data. In addition, we need to choose σ in such a way that the one-hot vector provides sufficient signal to the GAN training that leads to each mode only generating samples from a corresponding class in the original data. To be more precise, we chose $\sigma = 0.10$ in all our experiments so that each dimension of the normal latent variables, $z_{n,j} \in (-0.6, 0.6) << 1.0 \; \forall j$ with high probability. Small variances σ are chosen to ensure the clusters in $\mathcal{Z}$ space are separated. Hence this prior naturally enables us to design an algorithm that clusters in the latent space.

2.4 Modified Backpropagation Based Decoding

Previous works (Creswell and Bharath 2018) (Lipton and Tripathi 2017) have explored solving an optimization problem in z to recover the latent vectors, $z^* = \arg \min_z \mathcal{L}(\mathcal{G}(z), x) + \lambda \|z\|_p$, where $\mathcal{L}$ is some suitable loss function and $\|\cdot\|_p$ denotes the norm. This approach is insufficient for clustering with traditional latent priors even if backpropagation was lossless and recovered accurate latent vectors. To make the situation worse, the optimization problem above is non-convex in z ($\mathcal{G}$ being implemented as a neural network) and can obtain different embeddings in the $\mathcal{Z}$ space based on initialization. Some of the approaches to address this issue could be multiple restarts with different initializa-
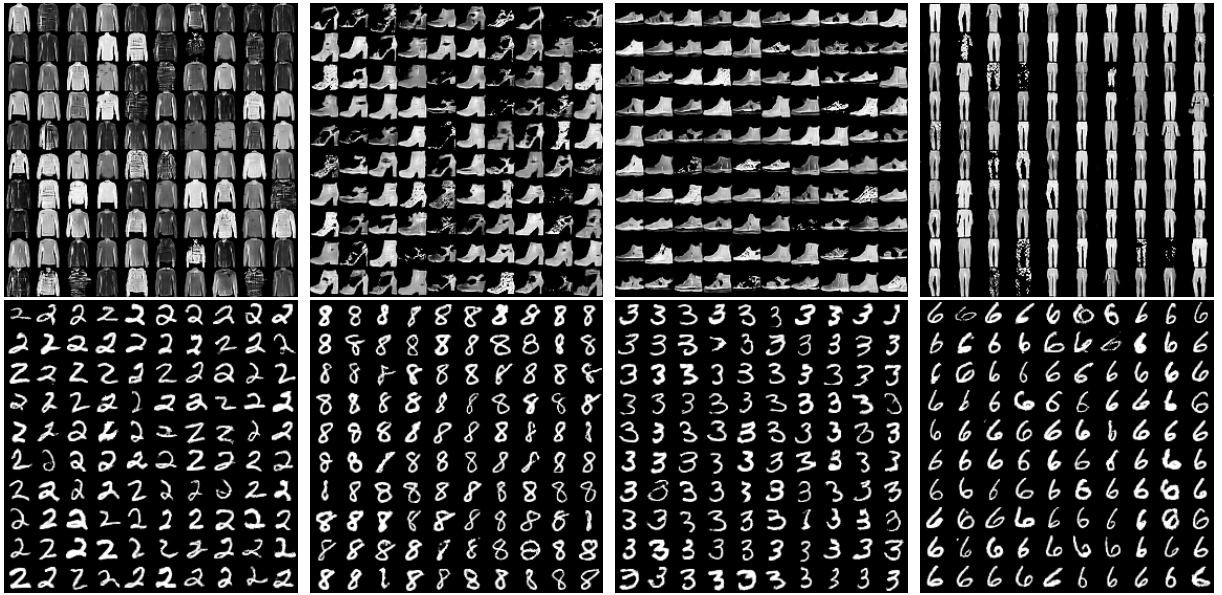
Figure 4: Fashion Items (Fashion-MNIST) and Digits (MNIST) generated from distinct modes of the GAN trained with one-hot encoded normal vectors in an unsupervised manner without additional loss terms.
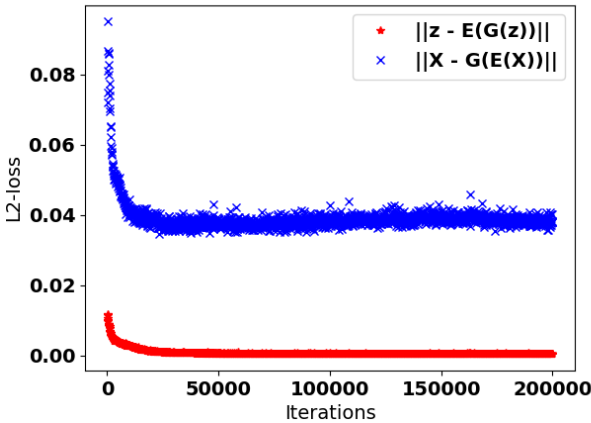

Figure 5: Decrease of Cycle Loss with iterations in MNIST. The mean square L2-distance $||x - \mathcal{G}(\mathcal{E}(x))||$ was 0.038 and $||z - \mathcal{E}(\mathcal{G}(z))||$ was 0.0004. Mean distances were computed on a test batch not used in training.

tions to obtain z^*, or stochastic clipping of z at each iteration step. None of these lead to clustering, since they do not address the root problem of sampling from separated manifolds in $\mathcal{Z}$. But our sampling procedure naturally gives way to such an algorithm. We use $\mathcal{L}(\mathcal{G}(z), x) = ||\mathcal{G}(z) - x||_1$. Since we sample from a normal distrubution, we use the regularizer $||z_n||_2^2$, penalizing only the normal variables. We use K restarts, each sampling z_c from a different one-hot component and optimize with respect to only the normal variables, keeping z_c fixed. Adam (Kingma and Ba 2014) is used for the updates during Backprop decoding.

2.5 Linear Generator clusters perfectly

The following lemma suggests that with discrete-continuous mixtures, we need only linear generation to generate mixture of Gaussians in the generated space.

Lemma 1. *Clustering with only z_n cannot recover a mixture of gaussian data in the linearly generated space. Further $\exists$ a linear $G(\cdot)$ mapping discrete-continuous mixtures to a mixture of Gaussians.*

Proof. If latent space has only the continuous part, $z_n \sim \mathcal{N}(0, \sigma^2 I_{d_n})$, then by the linearity property, any linear generation can only produce Gaussian in the generated space. Now we show there exists a $G(\cdot)$ mapping discrete-continuous mixtures to the generate data $X \sim \mathcal{N}(\mu_\omega, \sigma^2 I_{d_n})$, where $\omega \sim \mathcal{U}\{1, 2, \ldots, K\}$ (K is the number of mixtures). This is possible if we let $z_n \sim \mathcal{N}(0, \sigma^2 I_{d_n}), z_c = e_k, k \sim \mathcal{U}\{1, 2, \ldots, K\}$ and $G(z_n, z_c) = z_n + Az_c$, $A = diag[\mu_1, \cdots, \mu_K]$ being a $K \times K$ diagonal matrix with diagonal entries as the means μ_i. $\square$

To illustrate this lemma, and hence the drawback of traditional priors $\mathbb{P}_z$ for clustering, we performed a simple experiment. The real samples are drawn from a mixture of 10 Gaussians in $\mathbb{R}^{100}$. The means of the Gaussians are sampled from $\mathcal{U}(-0.3, 0.3)^{100}$ and the variance of each component is fixed at $\sigma = 0.12$. We trained a GAN with $z \sim \mathcal{N}(0, I_{10})$ where the generator is a multi-layer perceptron with two hidden layers of 256 units each. For comparison, we also trained a GAN with z sampled from one-hot encoded normal vectors, the dimension of categorical variable being 10. The generator for this GAN consisted of a linear mapping $W \in \mathbb{R}^{100 \times 10}$, such that $x = Wz$. After training, the latent vectors are recovered using backpropagation with 10
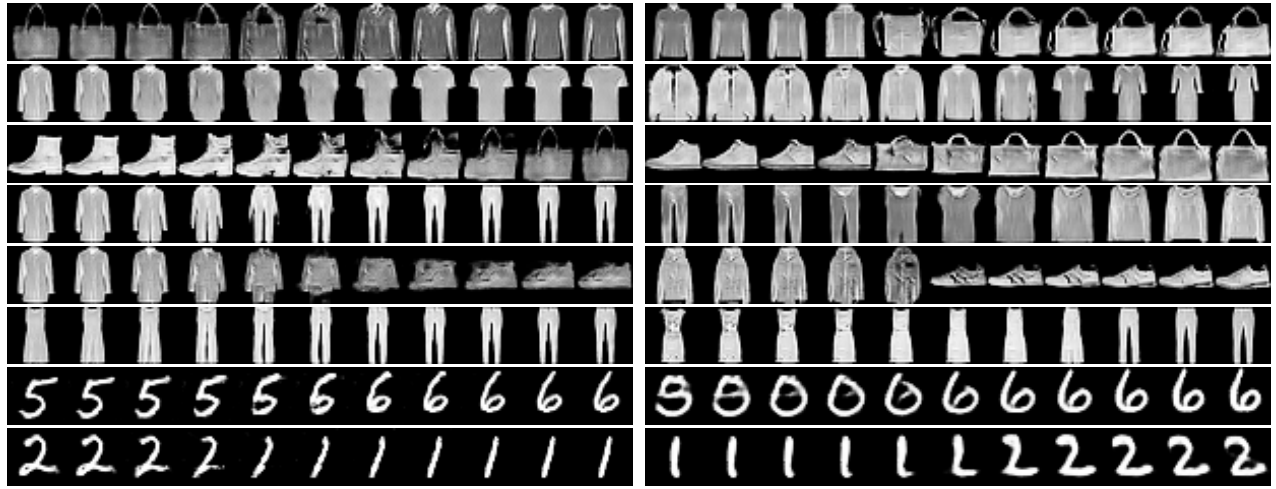
Figure 6: Comparison of Latent Space Interpolation : ClusterGAN (left) and vanilla WGAN (right)

restarts from random initializations both for the linear generator and non-linear generator. For linear generator, we used the modified backpropagation. Even for this toy setup, the linear generator perfectly clustered the latent vectors (Acc. = 1.0, NMI = 1.0, ARI = 1.0), but the non-linear generator performed poorly (Acc. = 0.73, NMI = 0.75, ARI = 0.60) (Figure 3). The situation becomes worse for real datasets such as MNIST when we trained a GAN using latent vectors drawn from uniform, normal or a mixture of Gaussians. None of these configurations succeeded in clustering in the latent space as shown in Figure 2.

2.6 Separate Modes for distinct classes in the data

It was surprising to find that trained in a purely unsupervised manner without additional loss terms, each one-hot encoded component generated points from a specific class in the original data. For instance, $z = (z_n, e_k)$ generated a particular digit $\pi(k)$ in MNIST, for multiple samplings of $z_n \sim \mathcal{N}(0, \sigma^2 I_{d_n})$ (π denotes a permutation). This was a necessary first step for the success of modified backpropagation. We also quantitatively evaluated the modes learnt by the GAN by using a supervised classifier for MNIST. The supervised classifier had a test accuracy of 99.2%, so it had high reliability of distinguishing the digits. We sample from a mode k and generate a digit x_g. It is then classified by the classifier as $\hat{y}$. From this pair $(k, \hat{y})$, we can map each mode to a digit and compute the accuracy of digit $\hat{y}$ being generated from mode k. This is denoted as Mode Accuracy. Each digit sample x_r with label y can be decoded in the latent space to obtain z. Now z can be used to generate x_g, which when passed through the classifier gives the label $\hat{y}$. The pair $(y, \hat{y})$ must be equal in the ideal case and this accuracy is denoted as Reconstruction Accuracy. Finally, all the mappings of points in the same class in $\mathcal{X}$ space should have the same one-hot encoding when embedded in $\mathcal{Z}$ space. This defines the Cluster Accuracy. This methodolgy can be extended to quantitatively evaluate mode generation for other datasets also, provided there is a reliable classifier. For MNIST, we obtained Mode Accuracy of 0.97, Reconstruction Accuracy

of 0.96 and Cluster Accuracy of 0.95. Some of the modes in Fashion-MNIST and MNIST are shown in Figures 4.

3 ClusterGAN

Even though the above approach enables the GAN to cluster in the latent space, it may be able to perform even better if we had a clustering specific loss term in the minimax objective. For MNIST, digit strokes correspond well to the category in the data. But for more complicated datasets, we need to enforce structure in the GAN training. One way to ensure that is to enforce precise recovery of the latent vector. We therefore introduce an encoder $\mathcal{E} : \mathcal{X} \mapsto \mathcal{Z}$, a neural network parameterized by Θ_E. The GAN objective now takes the following form:

$$\min_{\Theta_G, \Theta_E} \max_{\Theta_D} \mathop{\mathbb{E}}_{x \sim \mathbb{P}_x^r} q(\mathcal{D}(x)) + \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} q(1 - \mathcal{D}(\mathcal{G}(z)))$$
$$+ \beta_n \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} \|z_n - \mathcal{E}(\mathcal{G}(z_n))\|_2^2 + \beta_c \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} \mathcal{H}(z_c, \mathcal{E}(\mathcal{G}(z_c)))$$

where $\mathcal{H}(.,.)$ is the cross-entropy loss. The relative magnitudes of the regularization coeficients β_n and β_c enable a flexible choice to vary the importance of preserving the discrete and continuous portions of the latent code. One could imagine other variations of the regularization that map $\mathcal{E}(\mathcal{G}(z))$ to be close to the centroid of the respective cluster, for instance $\|\mathcal{E}(\mathcal{G}(z^{(i)})) - \mu^{c(i)}\|_2^2$, in similar spirit as K-Means. The GAN training in this approach involves jointly updating the parameters of Θ_G and Θ_E.

As shown in Figure 5, in our architecture, both x is close to $\mathcal{G}(\mathcal{E}(x))$ and z is close to $\mathcal{E}(\mathcal{G}(z))$. Even though our architecture is optimizing for one type of cycle loss, both losses are small. The loss optimized for is even smaller.

Interpolation in latent space is preserved

The latent space in a traditional GAN with Gaussian latent distribution enforces that different classes are continuously scattered in the latent space, allowing nice inter-class interpolation, which is a key strength of GANs. In ClusterGAN,

Dataset	Algorithm	ACC	NMI	ARI
Synthetic	ClusterGAN	**0.99**	**0.99**	**0.99**
	Info-GAN	0.88	0.75	0.74
	GAN with bp	0.95	0.85	0.88
	GAN with Disc. ϕ	0.99	0.98	0.98
	AGGLO.	0.99	0.99	0.99
	NMF	0.98	0.96	0.97
	SC	0.99	0.98	0.98
MNIST	ClusterGAN	**0.95**	0.89	**0.89**
	Info-GAN	0.89	0.86	0.82
	GAN with bp	0.95	**0.90**	0.89
	GAN with Disc. ϕ	0.70	0.62	0.52
	DCN	0.83	0.81	0.75
	AGGLO.	0.64	0.65	0.46
	NMF	0.56	0.45	0.36
Fashion-10	ClusterGAN	**0.63**	**0.64**	**0.50**
	Info-GAN	0.61	0.59	0.44
	GAN with bp	0.56	0.53	0.37
	GAN with Disc. ϕ	0.43	0.37	0.23
	AGGLO.	0.55	0.57	0.37
	NMF	0.50	0.51	0.34
Fashion-5	ClusterGAN	**0.73**	**0.59**	**0.48**
	Info-GAN	0.71	0.56	0.45
	GAN with bp	0.73	0.54	0.45
	GAN with Disc. ϕ	0.67	0.49	0.40
	AGGLO.	0.66	0.52	0.36
	NMF	0.67	0.48	0.40
10x_73k	ClusterGAN	**0.81**	**0.73**	**0.67**
	Info-GAN	0.62	0.58	0.43
	GAN with bp	0.65	0.59	0.45
	GAN with Disc. ϕ	0.33	0.17	0.07
	AGGLO.	0.63	0.58	0.40
	NMF	0.71	0.69	0.53
	SC	0.40	0.29	0.18
Pendigits	ClusterGAN	**0.77**	**0.73**	**0.65**
	Info-GAN	0.72	0.73	0.61
	GAN with bp	0.76	0.71	0.63
	GAN with Disc. ϕ	0.65	0.57	0.45
	DCN	0.72	0.69	0.56
	AGGLO.	0.70	0.69	0.52
	NMF	0.67	0.58	0.45
	SC	0.70	0.69	0.52

Table 1: Comparison of clustering metrics across datasets

Dataset	Algorithm			
	Cluster GAN	WGAN (Normal)	WGAN (One-Hot)	Info GAN
MNIST	**0.81**	0.88	0.94	1.88
Fashion	**0.91**	0.95	6.14	11.04

Table 2: Comparison of Frechet Inception Distance (FID) (Lower distance is better)

the latent vector z_c is sampled with a one-hot distribution and in order to interpolate across the classes, we will have to sample from a convex combination on the one-hot vector. While these vectors have never been sampled during

Dataset : MNIST, Algorithm : ClusterGAN				
ACC				
K = 7	**K = 9**	**K = 10**	**K = 11**	**K = 13**
0.72	0.80	**0.95**	0.90	0.84

Table 3: Robustness to Cluster Number K

the training process, we surprisingly observed very smooth inter-class interpolation in ClusterGAN. To demonstrate interpolation, we fixed the z_n in two latent vectors with different z_c components, say $z_c^{(1)}$ and $z_c^{(2)}$ and interpolated with the one-hot encoded part to give rise to new latent vectors $z = (z_n, \mu z_c^{(1)} + (1 - \mu)z_c^{(2)}), \mu \in [0, 1]$. As Figure 6 illustrates, we observed a nice transition from one digit to another as well as across different classes in FashionMNIST. This demonstrates that ClusterGAN learns a very smooth manifold even on the untrained directions of the discrete-continuous distribution. We also show interpolations from a vanilla GAN trained with Gaussian prior as reference.

4 Experiments

4.1 Datasets

Synthetic Data The data is generated from a mixture of Gaussians with 4 components in 2D, which constitutes the $\mathcal{Z}$ space. We generated 2500 points from each Gaussian. The $\mathcal{X}$ space is obtained by a non-linear transformation : $x = f(\mathbf{U} \cdot f(\mathbf{W}z))$, where $\mathbf{W} \in \mathbb{R}^{10 \times 2}, \mathbf{U} \in \mathbb{R}^{100 \times 10}$ with $W_{i,j} \sim \mathcal{N}(0,1), U_{i,j} \sim \mathcal{N}(0,1)$. $f(\cdot)$ is the sigmoid function to introduce non-linearity.

MNIST It consists of 70k images of digits ranging from 0 to 9. Each data sample is a 28×28 greyscale image.

Fashion-MNIST (10 and 5 classes) This dataset has the same number of images with the same image size as MNIST, but it is fairly more complicated. Instead of digits, it consists of various types of fashion products. We also merged some categories which were similar to form a separate 5-class dataset. The five groups were as follows : {Tshirt/Top, Dress}, {Trouser}, {Pullover, Coat, Shirt}, {Bag}, {Sandal, Sneaker, Ankle Boot}.

10x_73k Even though GANs have achieved unprecedented success in generating realistic images, it is not clear whether they can be equally effective for other types of data. Moreover, computer vision might have ample supply of labelled images, obtaining labels for some fields, for instance biology, is extremely costly and laborious. Thus, unsupervised clustering of data is truly a necessity for this domain. In this experiment, we trained a GAN to cluster cell types from a single cell RNA-seq counts matrix. The dataset consists of RNA-transcript counts of 73233 data points belonging to 8 different cell types (Zheng et al. 2017). To reduce the dimension of the data, we selected 720 highest variance genes across the cells. The entries of the counts matrix $\mathbf{C}$ are first tranformed as $\log_2(1 + C_{ij})$ and then divided by the maximum entry of the transformation to obtain values in the range of $[0, 1]$. One of the major challenges in this data is sparsity. Even after subselection of genes based on variance, the data matrix had close to 40% zero entries.
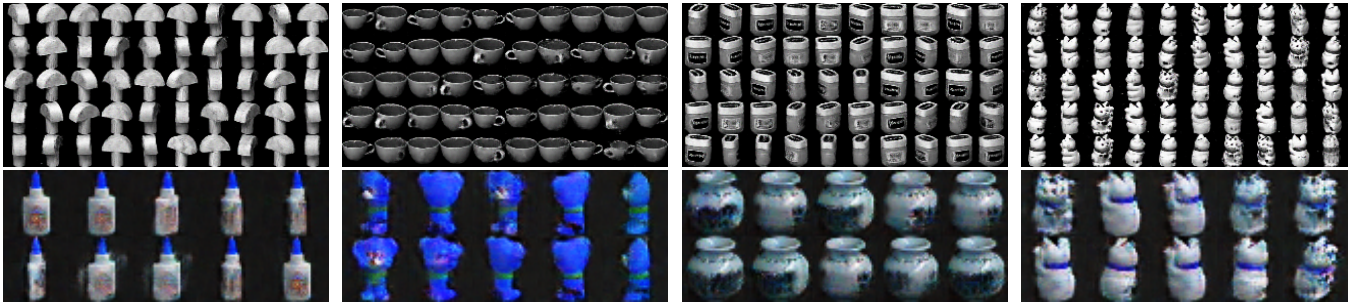
Figure 7: Scalability of ClusterGAN to large number of clusters : Modes of Coil-20 (above) and Coil-100 (below)

Pendigits It is a very different dataset that consists of a time series of $\{x_t, y_t\}_{t=1}^T$ coordinates. The points are sampled as writers write digits on a pressure sensitive tablet. The total number of datapoints is 10992, and consists of 10 classes, each for a digit. It provided a unique challenge of training GANs for point cloud data.

In all our experiments in this paper, we used an improved variant (WGAN-GP) which includes a gradient penalty (Gulrajani et al. 2017). Using cross-validation for selecting hyperparameters is not an option in purely unsupervised problems due to absence of labels. We adapted standard architectures for the datasets (Chen et al. 2016) and avoided data specific tuning as much as possible. Some choices of regularization parameters $\lambda = 10$, $\beta_n = 10$, $\beta_c = 10$ worked well across all datasets.

4.2 Evaluation

Since clustering is an unsupervised problem, we ensured that all the algorithms are oblivious to the true labels unlike a supervised framework like conditional GAN (Mirza and Osindero 2014). We compared ClusterGAN with other possible GAN based clustering approaches we could conceive.

Modified backpropagation + K-Means is denoted as "GAN with bp". For InfoGAN, we used $\arg\max_c \mathbb{P}(c \mid x)$ as an inferred cluster label for x. Further, the features $\phi(x)$ in the last layer of the Discriminator could contain some class-specific discriminating features for clustering. So we used K-Means on $\phi(x)$ to cluster, denoted as "GAN with Disc. ϕ". We also included clustering results from Non-negative matrix Factorization (NMF) (Lee and Seung 1999), Aggolomerative Clustering (AGGLO) (Zhang et al. 2012) and Spectral Clustering (SC). AGGLO with Euclidean affinity score and ward linkage gave best results. NMF had both l-1 and l-2 regularization, initialized with Non-negative Double SVD and used KL-divergence loss. SC had rbf kernel for affinity. We reported normalized mutual information (NMI), adjusted Rand index (ARI), and clustering purity (ACC). Since DCN has been shown to outperform various deep-learning based clustering algorithms, we reported its metrics from the paper (Yang et al. 2017) for MNIST and Pendigits. We found DCN to be very sensitive to hyperparameter choice, architecture and learning rates and could not obtain reasonable results from it on the other datasets. But we outperformed DCN results on MNIST and Pendig-

its dataset [1]. The code for ClusterGAN is available at https://github.com/sudiptodip15/ClusterGAN.

Since clustering metrics do not reveal the quality of generated samples from a GAN, we report the Frechet Inception Distance (FID) (Heusel et al. 2017) for the image datasets. We found that ClusterGAN achives good clustering without compromising sample quality as shown in Table 2.

In all datasets, we provided the true number of clusters to all algorithms. In addition, for MNIST, Table 3 provides the clustering performance of ClusterGAN as number of clusters is varied. Overestimates do not severely hurt Cluster-GAN; but underestimate does.

4.3 Scalability to Large Number of Clusters

We ran ClusterGAN on Coil-20 ($N = 1440$, $K = 20$) and Coil-100 ($N = 7200$, $K = 100$) datasets, where N is the number of Data points. ClusterGAN could obtain good clusters even with such high value of K. These data sets were particularly difficult for GAN training with only a few thousand data points. Yet, we found similar behavior as MNIST / Fashion-MNIST emerging here as well. Distinct modes generated distinct 3D-objects along with rotations as shown in Figure 7.

5 Discussion and Future Work

In this work, we discussed the drawback of training a GAN with traditional prior latent distributions for clustering and considered discrete-continuous mixtures for sampling noise variables. We proposed ClusterGAN, an architecture that enables clustering in the latent space. Comparison with clustering baselines on varied datasets using ClusterGAN illustrates that GANs can be suitably adapted for clustering. Future directions can explore better data-driven priors for the latent space.

6 Acknowledgments

References

Andrew, G.; Arora, R.; Bilmes, J.; and Livescu, K. 2013. Deep canonical correlation analysis. In *International Conference on Machine Learning*, 1247–1255.

---

[1] For all baselines and GAN variants, Table 1 reports metrics for the model with best validation purity from 5 runs.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.

Bora, A.; Jalal, A.; Price, E.; and Dimakis, A. G. 2017. Compressed sensing using generative models. In *ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 537–546.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2172–2180.

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 215–223.

Creswell, A., and Bharath, A. A. 2018. Inverting the generator of a generative adversarial network (ii). *arXiv preprint arXiv:1802.05701*.

Donahue, J.; Krähenbühl, P.; and Darrell, T. 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Dumoulin, V.; Belghazi, I.; Poole, B.; Mastropietro, O.; Lamb, A.; Arjovsky, M.; and Courville, A. 2016. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, 5769–5779.

Gurumurthy, S.; Sarvadevabhatla, R. K.; and Babu, R. V. 2017. Deligan: Generative adversarial networks for diverse and limited data. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4941–4949. IEEE.

Halkidi, M.; Batistakis, Y.; and Vazirgiannis, M. 2001. On clustering validation techniques. *Journal of intelligent information systems* 17(2-3):107–145.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 6626–6637.

Ilyas, A.; Jalal, A.; Asteri, E.; Daskalakis, C.; and Dimakis, A. G. 2017. The robust manifold defense: Adversarial training using generative models. *arXiv preprint arXiv:1712.09196*.

Ji, P.; Zhang, T.; Li, H.; Salzmann, M.; and Reid, I. 2017. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, 23–32.

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

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

Lee, D. D., and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788.

Lipton, Z. C., and Tripathi, S. 2017. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Makhzani, A.; Shlens, J.; Jaitly, N.; Goodfellow, I.; and Frey, B. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Mika, S.; Schölkopf, B.; Smola, A. J.; Müller, K.-R.; Scholz, M.; and Rätsch, G. 1999. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems*, 536–542.

Mirza, M., and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Ng, A. Y.; Jordan, M. I.; and Weiss, Y. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, 849–856.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Santurkar, S.; Budden, D.; and Shavit, N. 2017. Generative compression. *arXiv preprint arXiv:1703.01467*.

Tolstikhin, I.; Bousquet, O.; Gelly, S.; and Schoelkopf, B. 2017. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

Xiang, S.; Nie, F.; and Zhang, C. 2008. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition* 41(12):3600–3612.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487.

Yang, B.; Fu, X.; Sidiropoulos, N. D.; and Hong, M. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 3861–3870.

Zhang, W.; Wang, X.; Zhao, D.; and Tang, X. 2012. Graph degree linkage: Agglomerative clustering on a directed graph. In *European Conference on Computer Vision*, 428–441. Springer.

Zheng, G. X.; Terry, J. M.; Belgrader, P.; Ryvkin, P.; Bent, Z. W.; Wilson, R.; Ziraldo, S. B.; Wheeler, T. D.; McDermott, G. P.; Zhu, J.; et al. 2017. Massively parallel digital transcriptional profiling of single cells. *Nature communications* 8:14049.