DEEP CLUSTERING USING ADVERSARIAL NET BASED CLUSTERING LOSS

Kart Leong Lim
Institute of Microelectronics (IME),
Agency for Science, Technology and Research (A*STAR),
2 Fusionopolis Way, Innovis #08-02, Singapore 138634, Republic of Singapore
limkl@ime.a-star.edu.sg

ABSTRACT

Deep clustering is a recent deep learning technique which combines deep learning with traditional unsupervised clustering. At the heart of deep clustering is a loss function which penalizes samples for being an outlier from their ground truth cluster centers in the latent space. The probabilistic variant of deep clustering reformulates the loss using KL divergence. Often, the main constraint of deep clustering is the necessity of a closed form loss function to make backpropagation tractable. Inspired by deep clustering and adversarial net, we reformulate deep clustering as an adversarial net over traditional closed form KL divergence. Training deep clustering becomes a task of minimizing the encoder and maximizing the discriminator. At optimality, this method theoretically approaches the JS divergence between the distribution assumption of the encoder and the discriminator. We demonstrated the performance of our proposed method on several well cited datasets such as MNIST, REUTERS and CIFAR10, achieving on-par or better performance with some of the state-of-the-art deep clustering methods.

1 INTRODUCTION

Deep neural network such as the autoencoder is applied to many domains such as speech and audio processing Karamatli et al. (2019); Last et al. (2020), image clustering Huang et al. (2019); Xie et al. (2016) and medical data processing Abdelhameed & Bayoumi (2019); Han et al. (2020). While the latent space of autoencoder is well suited for dimension reduction through reconstruction loss Song et al. (2013); Yang et al. (2017), the latter is not optimized for clustering/ classification since class labels cannot be used in the reconstruction loss Min et al. (2018). A recent autoencoder technique known as the Euclidean distance based clustering (ABC) Song et al. (2013); Yang et al. (2017), utilize class label information in neural network by introducing a loss function known as the deep clustering loss The goal is to minimizes the Euclidean distance in the latent space between samples and partitioning learnt by clustering algorithms e.g. Kmeans or Gaussian mixture model (GMM). In other words, the latent space of an autoencoder learnt using reconstruction loss will look different when we further perform K-means on the latent space. A recent probabilistic approach to ABC uses KL divergence (KLD) Lim et al. (2020); Jiang et al. (2017) and assumes that both the variational autoencoder (VAE) Kingma & Welling (2014) latent space and clustering approach are Gaussian distributed. We can compare the similarities and differences between deep clustering and VAE to better understand the former (A.2, A.3). The neural network of VAE and deep clustering are optimized by backpropagating samples in the latent space to update the encoder weights. Also, both deep clustering and VAE share a similar goal of modeling each input image as a sample draw of the Gaussian distribution in the latent space. However, their Gaussian distributions are different. VAE enforce samples in the latent space to be closely distributed by a Gaussian distribution with continuous mean μ and variance σ i.e. $z \sim \mu + \sigma \cdot \mathcal{N}(0, 1)$. Whereas the latent space of deep clustering approaches a Gaussian mixture model (GMM) and each Gaussian has a set of fixed parameters i.e. $z \sim \mathcal{N}(B_k, \sigma_k)$, $\forall k \in K$. The goal of VAE is to enforce all class samples in the latent space to be distributed by continuous mean and variance. The goal of deep clustering is to enforce class samples in the latent space to be distributed by a discrete K set of mean and variance. Thus, only deep clustering is associated with clustering, making it more suitable for unsupervised classification.

Deep clustering is not without problem. Mainly, when backpropagating from a neural network, we require a closed form loss function for tractability. Most deep learning methods including VAE restrict to the assumption of a KLD between two Gaussians to obtain a closed form loss function for backpropagating. When generalizing to f-divergence between two Gaussians, closed form solution does not exist. A well known workaround to this problem is to approximate the JS divergence (JSD) between two Gaussians using adversarial net. In fact, adversarial net is quite versatile as shown in Table 1. For different methods, the discriminator can be seen as performing a specific task different from other adversarial nets. We distinguish the proposed work from other "**adversarial net based X**" in Table I where **X** can be any discriminator input. In GAN Goodfellow et al. (2014), **X** refers to image generation. Other instances of "**adversarial net based X**" includes AAE Makhzani et al. (2016) , DASC Zhou et al. (2018) and etc. In our approach, "Deep clustering using adversarial net based clustering loss" (DCAN), **X** refers to clustering. More specifically, in DCAN the JSD between latent space and clustering is approximated by adversarial net. More details are discussed in A.1.

Table 1: Different strategies of using "adversarial net based X"

Method	**Adversarial net**	**Task X**
GAN	Is x from the ground truth or randomly generated? $\begin{cases} \text{if } x \sim p(\text{data}),\ T=1 \\ \text{elseif } x \sim p(g),\ T=0 \end{cases}$	$x \sim p(data)$
AAE	Is z from a Gaussian prior? $\begin{cases} \text{if } z \sim \mu + \sigma \cdot \mathcal{N}(0,1),\ T=1 \\ \text{elseif } z \sim q(z \mid x),\ T=0 \end{cases}$	$z \sim p(z)$
DASC	Does z fall along the assigned subspace? $\begin{cases} \text{if } z \sim p(\psi^*),\ T=1 \\ \text{elseif } z \sim p(\hat{\psi}),\ T=0 \end{cases}$	$z \sim p(\psi^*)$
DCAN	Does z belong to the assigned cluster? $\begin{cases} \text{if } z \sim p(z \mid \theta^*),\ T=1 \\ \text{elseif } z \sim q(z \mid x),\ T=0 \end{cases}$	$z \sim p(z \mid \theta^*)$

2 PROPOSED METHOD

One motivation for using a JSD approach is that it does not suffer from asymmetry unlike KLD Nielsen (2019); Goodfellow et al. (2014). However, there is no closed form solution available for the JSD between two data distributions. GAN Goodfellow et al. (2014) overcome the need for a closed form solution by employing an adversarial training procedure. GAN approximates JSD or $D_{JS}\left(p_{data} \parallel p_g\right) = E_{x \sim p_{data}}\left[\log D_G(x)\right] + E_{x \sim p_g}\left[\log\left(1 - D_G\left\{x\right\}\right)\right]$ at optimality, where p_{data} is the real data distribution and p_g is the generated data distribution. In the JSD between two Gaussian distributions for deep clustering, we seek $D_{JS}\left(\, p(z \mid \theta^*) \parallel q(z \mid x)\,\right) = E_{z \sim p(z|\theta^*)}\left[\ln D(z)\right] + E_{z \sim q(z|x)}\left[\ln\left(1 - D\left\{z\right\}\right)\right]$, whereby $z \sim p(z \mid \theta^*)$ refers to the sample distributed from clustering and $z \sim q(z \mid x)$ refers to the sample distributed from the encoder. Unlike the discriminator of GAN in the original space, the discriminator of DCAN works in the latent space. We can visualize $D_{JS}\left(\, p(z \mid \theta^*) \parallel q(z \mid x)\,\right)$ using Fig 1. The discriminator tries to distinguish samples from clustering, $z \sim p(z \mid \theta^*)$ versus samples generated from the encoder, $z \sim q(z \mid x)$.

2.1 TRAINING DCAN

We train DCAN by feeding it with $1...N$ positive and negative samples. In Fig 1, we refer to each n^{th} sample of x as $x^{(n)}$. During the forward pass, samples enter the discriminator through the encoder and clustering. Samples generated from clustering are positive i.e. $T = 1$ and samples generated from encoder are negative i.e. $T = 0$. We define the neural network output and target label as $y = \{0, 1\}$ and $T = \{0, 1\}$ respectively. We update the discriminator weight using DCAN loss (A.4)

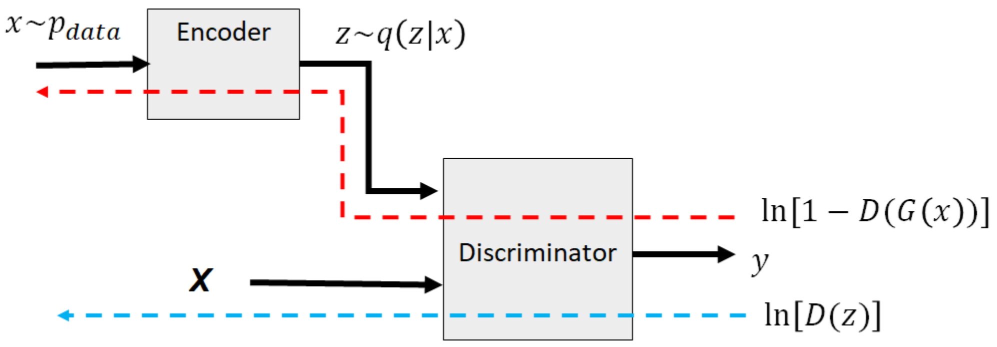

Figure 1: Architecture of "**adversarial net based X**" for discriminator task in the latent space, (e.g. AAE). Task **X** can be any discriminator input as shown in Table 1. Replace x with z (and vice-versa) for discriminator task in the image space, (e.g. GAN).

in eqn (1). Since the samples in the latent space displaces each time the encoder weight changes, clustering parameters θ has to be re-computed each time the discriminator weight is updated. The encoder $q(z \mid x)$ maps x to z in the latent space. However, we seek to minimize between $q(z \mid x)$ and $p(z \mid \theta^*)$. Meaning that we desire to produce a sample z from $q(z \mid x)$ that is as close as possible to the sampling from $p(z \mid \theta^*)$. Lastly, we backpropagate from the discriminator to update the encoder using DCAN loss. (A.5)

$$\begin{aligned} L_{DCAN} = & \left\{ \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \underset{x^{(n)} \sim p_{data}}{E} \left[\ln \left(1 - D\left(G(x^{(n)}) \right) \right) \right] \right\}_{T=0} \\ & + \left\{ \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \underset{z^{(n)} \sim p(z|\theta^*)}{E} \left[\ln D\left(z^{(n)} \right) \right] \right\}_{T=1} \end{aligned} \tag{1}$$

We model $p(z \mid \theta^*)$ using Gaussian mixture model (GMM) Bishop (2006) in eqn (2). We denote GMM variables, $\theta = \{\eta, \tau, \psi\}$ as the mean η, precision τ and mixture component ψ. The significance of ψ is that each cluster is represented as a probability ψ_k as opposed to hard assignment in Kmeans. (A.6)

$$\ln p(z|\eta, \tau, \psi) = \sum_{n=1}^{N} \sum_{k=1}^{K} \{\ln \psi_k + \ln \mathcal{N}(z_n|\eta_k, \tau_k, \psi_k)\} \zeta_{nk} \tag{2}$$

3 EXPERIMENTS

We selected some benchmarked datasets used by deep clustering for our experiments in Tables 2. There are several factors affecting deep clustering. i) The clustering algorithm used in the latent space e.g. Kmeans or GMM. ii) The number of hidden layers and nodes of the neural network. iii) The gradient ascent method used for training the loss functions. iv) The activation functions used. For DCAN, we use $tanh$ for the hidden layers and $sigmoid$ for the output. We set the clustering iteration to one in Table 2 and a sufficient statistics of at least 600 points in the raw latent space to estimate the cluster parameters. The number of cluster is the same as the number of classes. We use a minibatch size of $N = 16$ samples, 500 iterations and we use gradient ascent with momentum with a learning rate between 10^{-3} and 10^{-5}. We use accuracy (ACC) Cai et al. (2005) to evaluate the performance of our clustering.

MNIST: A well cited 10 digit classes dataset with no background but it has more samples than USPS. The train and test sample size is at 50K and 10K. Our settings are $196 - 384 - 256$ for the encoder and $256 - 16 - 1$ for the discriminator. The input dimension is originally at 28×28 but we have downsampled it to 14×14 and flattened to 196. In Table 2, DCAN using raw pixel information obtained 0.8565 for ACC which is on par or better than deep clustering methods including DEC, ABC and DC-GMM. However on MNIST, most deep clustering methods could not obtain the performance of recent deep learning methods such as KINGDRA-LADDER-1M (a.k.a. KINGDRA)

Table 2: ACC Benchmark

Approach	MNIST	Reuters10k	CIFAR10
ABC Song et al. (2013)	0.760	0.7019	0.435
DEC Xie et al. (2016)	0.843	0.7217	0.469
DC-GMM Tian et al. (2017)	0.8555	0.6906	-
AAE Makhzani et al. (2016)	0.8348	0.6982	-
IMSAT-RPT Hu et al. (2017)	0.896	0.719	0.455
KINGDRA Gupta et al. (2020)	**0.985**	0.705	0.546
DCAN (proposed)	0.8565	**0.7867**	**0.5844**

and IMSAT as their goals are much more ACC result oriented. For KINGDRA, it combines several ideas together including ladder network Rasmus et al. (2015) and psuedo-label learning Lee (2013) for the neural network and using ensemble clustering. In DAC, their approach is based on pairwise classification. In IMSAT-RPT, the neural network was trained by augmenting the training dataset. In our case, we simplified our focused on deep clustering using adversarial net. DCAN penalizes samples for being an outlier from their cluster centers and does so uniquely by using the discriminator to penalize the encoder. The original objective of deep clustering is specifically aimed at clustering. In some datasets, the latent space of this objective do not necessarily co-serve as the most important factor for raw pixel feature extraction. Thus, we also used ResNet18 pretrained model as a feature extractor for MNIST, prior to deep clustering. This simple step allows the ACC for DCAN to improved to 0.995 (not shown in Table 2) and in fact outperforms KINGDRA.

Reuters-10k: The Reuters-10k dataset according to Tian et al. (2017); Xie et al. (2016) contains 10K samples with 4 classes and the feature dimension is 2000. End-to-end learning is performed on this dataset. We compared DCAN to other unsupervised deep learning approaches. DCAN uses $2000 - 100 - 500$ and $500 - 100 - 1$ respectively for the encoder and discriminator. In Table 2, we observed that Kmeans alone can achieve an ACC of 0.6018 on the raw feature dimension. Most deep learning methods can achieve between 0.69 to 0.73 on this dataset. DCAN was able to obtain the best ACC at 0.7867.

CIFAR10: A 10 classes object categorization dataset with 50K training and 10K testing of real images. On CIFAR10, most methods have difficulty performing end-to-end learning. As compared to deep ConvNet which utilizes convolutional neural network and supervised learning for a specific task (i.e. feature extraction), deep clustering typically rely on fewer hidden layer in the encoder for end-to-end learning (i.e. both feature extraction and clustering). Thus, we followed the experiment setup in KINGDRA Gupta et al. (2020) by using the “avg pool” layer of ResNet50 He et al. (2016) (ImageNet pretrained) as a feature extractor with a dimension of 2048. In Table 2, DCAN uses $2048 - 1024 - 512 - 128$ and $128 - 16 - 1$ respectively for the encoder and discriminator. Despite the stronger performance of DAC, IMSAT and KINGDRA on MNIST, we outperformed these comparisons on CIFAR10 at 0.5844. This is despite the fact that both KINGDRA and IMSAT also use ResNet50 for CIFAR10.

4 Conclusion

The objective of deep clustering is to optimize the autoencoder latent space with clustering information. The key to this technique lies in a loss function that minimizes both clustering and encoder in the latent space. Despite the recent success of deep clustering, there are some concerns: i) How to use probabilistic approach for the loss function? ii) Can we backpropagate without closed form solution for the probabilistic function? A JS divergence approach does not suffer from asymmetry like KL divergence loss. But there is no closed form solution to the JS divergence when backpropagating the network. This paper propose a novel deep clustering approach based on adversarial net to estimate JS divergence for deep clustering.

Acknowledgements

This work is supported by ”Predictive Package Integrity and Reliability for Multi Chiplet Heterogeneous Integration Products” Xilinx ICP Grant project WBS SC24/24 - 813700 Grant no I2301E0028 and AMAT IME P3.0 Grant project WBS SC24/21 - 803100 Grant No I231E0008.

References

A. M. Abdelhameed and M. Bayoumi. Semi-supervised eeg signals classification system for epileptic seizure detection. *IEEE Signal Processing Letters*, 26(12):1922–1926, 2019.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Deng Cai, Xiaofei He, and Jiawei Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637, December 2005.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Divam Gupta, Ramachandran Ramjee, Nipun Kwatra, and Muthian Sivathanu. Unsupervised clustering using pseudo-semi-supervised learning. In *International Conference on Learning Representations*, 2020.

M. Han, O. Ozdenizci, Y. Wang, T. Koike-Akino, and D. Erdogmus. Disentangled adversarial autoencoder for subject-invariant physiological feature extraction. *IEEE Signal Processing Letters*, 27:1565–1569, 2020. doi: 10.1109/LSP.2020.3020215.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1558–1567. JMLR. org, 2017.

Weitian Huang, Ming Yin, Jianzhong Li, and Shengli Xie. Deep clustering via weighted k-subspace network. *IEEE Signal Processing Letters*, 26(11):1628–1632, 2019.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1965–1972. AAAI Press, 2017.

E. Karamatli, A. T. Cemgil, and S. Kirbiz. Audio source separation using variational autoencoders and weak class supervision. *IEEE Signal Processing Letters*, pp. 1–1, 2019. ISSN 1070-9908. doi: 10.1109/LSP.2019.2929440.

Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, 2014.

P. Last, H. A. Engelbrecht, and H. Kamper. Unsupervised feature learning for speech using correspondence and siamese networks. *IEEE Signal Processing Letters*, 27:421–425, 2020. doi: 10.1109/LSP.2020.2973798.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

Kart-Leong Lim, Xudong Jiang, and Chenyu Yi. Deep clustering with variational autoencoder. *IEEE Signal Processing Letters*, 27:231–235, 2020.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.

Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

Frank Nielsen. On the jensen–shannon symmetrization of distances relying on abstract means. *Entropy*, 21(5):485, 2019.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pp. 3546–3554, 2015.

Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican Congress on Pattern Recognition*, pp. 117–124. Springer, 2013.

Kai Tian, Shuigeng Zhou, and Jihong Guan. Deepcluster: A general clustering framework based on deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 809–825. Springer, 2017.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487, 2016.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3861–3870. JMLR. org, 2017.

Pan Zhou, Yunqing Hou, and Jiashi Feng. Deep adversarial subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1596–1604, 2018.

A Appendix

A.1 Adversarial net based X

The optimization of adversarial net in general centers around minimizing the encoder/generator and maximizing the discriminator. However, there are several methods using "**adversarial net based X**" in Table 1 and we highlight the similarities and differences as follows: In GAN the generator is responsible for generating an image when given a sample from the latent space. The discriminator of GAN checks whether the generated image belongs to the distribution of the original dataset. Similarly, in DCAN we minimize the encoder and maximize the discriminator but for different purpose. In DCAN, the encoder receives an image and outputs a sample in the latent space. GAN trains the generator to be sensitive to the variance of the image class i.e. small displacement of the sample in the latent space can result in huge image variance. DCAN trains the encoder to be insensitive to the class variance i.e. different images from the same class populate tightly close to the cluster center in the latent space. Both DCAN and AAE utilize the adversarial net in totally different ways despite having similar architecture (encoder acting as generator). The goal of AAE is identical to VAE i.e. the AAE encoder models the latent sample as a sum and product of $\mathcal{N}(0, 1)$. On the other hand, the goal of DCAN is not identical to VAE. DCAN models the latent sample, as the sample draw from the nearest cluster (using e.g. Kmeans or Gaussian mixture model). The goal of DASC is subspace clustering, and it is different from deep clustering. Like principal component analysis, DASC projects the dataset onto different number of low dimensional subspaces. Each cluster using Ncut is a low dimensional subspace. Thus task **X** in the "**adversarial net based X**" of all four methods, GAN (models the dataset), AAE (models VAE loss), DASC (models subspace clustering) and DCAN (models GMM clustering) are optimized in totally different ways as summarized in Table I.

A.2 Deep clustering

The simplest form of deep clustering uses the Euclidean distance loss in the Autoencoder based clustering (ABC) Song et al. (2013) approach. In ABC, the deep clustering objective optimizes the neural network to be affected by Kmeans clustering or $\mathcal{L}_2$ loss function in eqn (3). On the contrary, traditional reconstruction loss or $\mathcal{L}_1$ does not utilize any information from Kmeans.

$$\begin{aligned} &\underset{w,b}{min}\ \{\mathcal{L}_1 + \lambda \cdot \mathcal{L}_2\} \\ &= \underset{w,b}{max}\ -\tfrac{1}{2}(T-y)^2 - \lambda\left\{-\tfrac{1}{2}(h_z - \eta^*)^2\right\} \end{aligned} \tag{3}$$

A.3 Probabilistic variant of deep clustering

Training a deep clustering algorithm such as ABC in eqn (3) requires a closed-form equation for the clustering loss function. In higher dimensional latent space, the Euclidean function will become less robust due to the curse of dimensionality. The probabilistic approach of ABC is based on the "KLD between two Gaussians" in Lim et al. (2020); Jiang et al. (2017). When the probabilistic approach is no longer a "KLD between two Gaussians", the main problem is that:

a) the loss function can become non-trival to differentiate for backprogation.

Instead, we propose a new way to train deep clustering without facing such constrains i.e. we discard the use of probabilistic approach in deep clustering. Instead, we approximate deep clustering using adversarial net. There exist a relationship between deep clustering and adversarial net in two simple steps:

a) **From ABC to KLD to JSD:** We show that under certain condition (i.e. unit variance assumption), ABC is identical to KLD and thus related to JSD. Both KLD and JSD are also under the family of the f-divergence.

b) **Adversarial net as JSD:** We approximate JSD using adversarial net. This is possible since adversarial net approaches JSD when the training becomes optimal.

Our main goal is to find a relationship between adversarial net and deep clustering. First, we establish a relationship between KLD and ABC Lim et al. (2020). KLD measures the probabilistic distance between two distributions. The distributions of the latent space and deep clustering space are denoted $q(z \mid x)$ and $p(z \mid \theta)$ respectively. Deep clustering space is the latent space partitioned using GMM or Kmeans. For brevity, we denote GMM parameters with $\theta = \{\eta, \tau, \zeta\}$ as the mean η, precision τ and assignment $\boldsymbol{\zeta}$ for K number of clusters and N number of samples. Sample in the latent space is denoted $z = \left\{z^{(n)}\right\}_{n=1}^{N} \in \mathbb{R}^Z$, where Z is the dimension of the latent space. Taking $q(z \mid x)$ and $p(z \mid \theta)$, the KLD based clustering loss is defined as Lim et al. (2020):

$$\begin{gathered} D_{KL}\left(\, p(z \mid \theta) \parallel q(z \mid x)\,\right), \\ p(z \mid \theta) = \prod_{k=1}^{K} \mathcal{N}(z \mid \eta_k, (\tau_k)^{-1})^{\zeta_k}, \\ q(z \mid x) = \mu + \sigma \cdot \mathcal{N}(0,1) \end{gathered} \tag{4}$$

The problem is how do we define KLD in terms of two Gaussians, instead of a Gaussian and a GMM in eqn (4), which is intractable. To overcome this, we re-express the GMM term $p(z \mid \theta)$ as follows

$$\begin{aligned} p(z \mid \theta^*) &= \arg\max_k p(z \mid \theta_k) \\ &= \mathcal{N}(z \mid \eta^*, (\tau^*)^{-1}) \end{aligned} \tag{5}$$

We define $p(z \mid \theta^*)$ as the k^{th} Gaussian component of Kmeans or GMM that draws sample z in the latent space. i.e. we use ζ (assignment) to compute η^* (mean) and τ^* (precision). Thus, the KLD of GMM and latent space in eqn (4) can now become a KLD between two Gaussian distributions in eqn (6). More importantly, a closed formed equation is available for the latter.

$$D_{KL}\left(\mathcal{N}(z \mid \eta^*, (\tau^*)^{-1}) \parallel \mathcal{N}(z \mid \mu, \sigma^2)\right) \tag{6}$$

The relationship between KLD and ABC can be explained in Lemma 1: If we discard the second order terms i.e. σ^2 and τ in eqn (6), the KLD reverts back to the original ABC loss Song et al. (2013); Yang et al. (2017) in eqn (3) as follows:

$$D_{KL}\left(\mathcal{N}(z \mid \eta^*) \parallel \mathcal{N}(z \mid \mu)\right) = \frac{1}{2}(\eta^* - \mu)^2 \tag{7}$$

To relate KLD to JSD, we simply recall JSD as the averaging between two KLDs in eqn (8). We next discuss how to relate adversarial net to JSD.

$$\begin{aligned} &D_{JS}\left(\, p(z \mid \theta^*) \parallel q(z \mid x) \, \right) \\ &= \tfrac{1}{2} D_{KL}\left(p \parallel \tfrac{p+q}{2}\right) + \tfrac{1}{2} D_{KL}\left(q \parallel \tfrac{p+q}{2}\right) \end{aligned} \tag{8}$$

Lemma 1 *Relating deep clustering in eqn (3) to probabilistic deep clustering in eqn (6): We show that under assumption of "unit variance", the KLD between the encoder latent space and GMM reverts back to the ABC loss.*

Proof: For a D_{KL} between two Gaussian distributions, there is a unique closed-form expression available. When we assume unit variance i.e. $\tau = \sigma = 1$, D_{KL} reverts back to the original Euclidean distance based clustering loss in eqn (3) as follows

$$\begin{aligned} &D_{KL}\left(p(z \mid \theta) \parallel q(z \mid x)\right) \quad s.t. \ \{\tau = \sigma = 1\} \\ &= D_{KL}\left(\mathcal{N}(z_n \mid \eta^*, (\tau^*)^{-1}) \parallel \mathcal{N}(z_n \mid \mu, \sigma)\right) \\ &= \ln \tau^* + \ln \sigma + \frac{(\tau^*)^{-1} + (\eta^* - \mu)^2}{2\sigma^2} - \frac{1}{2} \\ &= \frac{1}{2}\left(\eta^* - \mu\right)^2 \end{aligned} \tag{9}$$

□

A.4 Deep clustering using Adversarial net approaches JSD

The problem with using JSD for deep clustering is that there is rarely a closed form solution available for a "JSD between two Gaussian distributions". GAN overcome this by employing an adversarial training procedure that approximates $D_{JS}\left(p_{data} \parallel p_g\right)$ at optimality, where p_{data} is the real data distribution and p_g is the generated data distribution as quoted in Goodfellow et al. (2014):

$$\begin{aligned} &D_{JS}\left(p_{data} \parallel p_g\right) \\ &= \underset{x \sim p_{data}}{E}\left[\log D_G(x)\right] + \underset{z \sim p(z)}{E}\left[\log\left(1 - D_G\left\{G(z)\right\}\right)\right] \\ &= \underset{x \sim p_{data}}{E}\left[\log D_G(x)\right] + \underset{x \sim p_g}{E}\left[\log\left(1 - D_G\left\{x\right\}\right)\right] \end{aligned} \tag{10}$$

Unlike the discriminator of GAN in the original space, the discriminator of DCAN works in the latent space. Despite that, we can express DCAN using the encoder-discriminator network in Fig 1:

$$\begin{aligned} &D_{JS}\left(p(z \mid \theta^*) \parallel q(z \mid x)\right) \\ &= \underset{z \sim p(z|\theta^*)}{E}\left[\ln D(z)\right] + \underset{x \sim p_{data}}{E}\left[\ln\left(1 - D\left\{G(x)\right\}\right)\right] \\ &= \underset{z \sim p(z|\theta^*)}{E}\left[\ln D(z)\right] + \underset{z \sim q(z|x)}{E}\left[\ln\left(1 - D\left\{z\right\}\right)\right] \end{aligned} \tag{11}$$

$E_{z \sim p(z|\theta^*)}\left[\cdot\right]$ refers to the expectation function where the sample is drawn from deep clustering space and $E_{z \sim q(z|x)}\left[\cdot\right]$ refers to the expectation function where the sample is drawn from the latent space. We refer to Lemma 2 for the claim on eqn (11).

Lemma 2 *The adversarial net based deep clustering loss by DCAN can be shown to approach JSD at optimum:*

Proof: For brevity, we refer to $p(z \mid \theta^*)$ and $q(z \mid x)$ as p and q respectively. Optimal discriminator occurs when G is fixed, i.e. $D = \frac{p}{p+q}$. Substituting D, we define LHS and RHS in eqn (12) below. When we subject $p = q = 1$ on both sides, we can validate the claim on eqn (11).

$$\begin{aligned} &\boldsymbol{LHS} : E_{z \sim p}\left[\ln D(z)\right] + E_{x \sim p_{data}}\left[\ln\left(1 - D\left\{G(x)\right\}\right)\right] \quad s.t. \ \{D = \tfrac{p}{p+q}\} \\ &= E_{z \sim p}\left[\log D(z)\right] + E_{z \sim q}\left[\log\left(1 - D(z)\right)\right] \quad s.t. \ \{D = \tfrac{p}{p+q}\} \\ &= E_{z \sim p}\left[\ln \tfrac{p}{p+q}\right] + E_{z \sim q}\left[\ln \tfrac{q}{p+q}\right] \\ &= \int_z p \ln \tfrac{p}{p+q} + q \ln \tfrac{q}{p+q} \, dz \end{aligned} \tag{12}$$

$$\begin{aligned} &\boldsymbol{RHS} : D_{JS}\left(p \parallel q\right) = \tfrac{1}{2} \int \, p \ln \tfrac{2 \cdot p}{p+q} + q \ln \tfrac{2 \cdot p}{p+q} \, dz \\ &= \tfrac{1}{2} \int_z \, p \left\{\ln \tfrac{p}{p+q} + \ln 2\right\} + q \left\{\ln \tfrac{q}{p+q} + \ln 2\right\} \, dz \end{aligned}$$

Thus, $LHS \leq 2RHS - 2\log 2$.

□

A.5 Implementation of DCAN

We can visualize $D_{JS}\left(\, p(z \mid \theta^*) \parallel q(z \mid x) \,\right)$ using Fig. 1. The discriminator tries to distinguish samples from deep clustering space, $z \sim p(z \mid \theta^*)$ versus samples from the encoder output, $z \sim q(z \mid x)$. We train DCAN by using the two loss functions L_1 and L_2.

A.5.1 Training the discriminator

Assuming MLP structure $x_i - h_j - z_\varsigma$ for the encoder and $z_\varsigma - h_f - y_d$ for the discriminator, the hidden layers are defined as h_f, h_j and z_ς. We train the discriminator by feeding it with $1...N$ positive and negative samples. We refer to each n^{th} sample of x as $x^{(n)}$. In the forward pass in Fig 1, latent samples go to the discriminator through the two paths. Samples generated from deep clustering space are positive (i.e. $T = 1$) and samples generated from the encoder are negative (i.e. $T = 0$). We define the neural network output and target label as $y = \{0, 1\}$ and $T = \{0, 1\}$ respectively. Both y and T have the same dimension d. Cross-entropy loss is defined for L_2 and L_1 using the following:

$$\begin{gathered} D = y^T (1-y)^{1-T} \\ L_1 = \ln D(z) \\ L_2 = \ln\left[1 - D(G(x))\right] \end{gathered} \tag{13}$$

Discriminator training, $T = 0$**:** We run forward pass using eqn (14). The activation function for each layer is represented as $\sigma(\cdot)$. As we seek sampling from $p(z \mid \theta^*)$ instead of $q(z \mid x)$, we set $T = 0$ for D. We update the discriminator weight using $\triangle w_\phi$ for weight $w_\phi = \{w_{fd}, w_{\varsigma f}\}$ in eqn (15).

$$\begin{aligned} h_j = \sigma_j\left(\textstyle\sum_{i=1}^{I} w_{ij} \cdot x_i\right), &\quad h_f = \sigma_f\left(\textstyle\sum_{\varsigma=1}^{Z} w_{\varsigma f} \cdot z_\varsigma\right) \\ z_\varsigma = \sigma_\varsigma\left(\textstyle\sum_{j=1}^{J} w_{j\varsigma} \cdot h_j\right), &\quad y_d = \sigma_d\left(\textstyle\sum_{f=1}^{F} w_{fd} \cdot h_f\right) \end{aligned} \tag{14}$$

$$\triangle w_\phi = \tfrac{\delta}{\delta w_\phi}\left\{ \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \underset{x^{(n)} \sim p_{data}}{E} \left[\ln\left(1 - D\left(G(x^{(n)})\right)\right)\right] \right\} \tag{15}$$

Discriminator training, $T = 1$**:** The samples in the latent space displaces each time the encoder weight changes. Thus, GMM has to be re-computed each time the weight is updated. After sampling from GMM i.e. $z \sim p(z \mid \theta^*)$ and setting $T = 1$ to D, we perform forward pass using eqn (16) and update the discriminator weight w_ϕ using eqn (17).

$$\begin{gathered} h_f = \sigma_f\left(\textstyle\sum_{\varsigma=1}^{Z} w_{\varsigma f} \cdot \{z \sim p(z \mid \theta^*)\}\right) \\ y_d = \sigma_d\left(\textstyle\sum_{f=1}^{F} w_{fd} \cdot h_f\right) \end{gathered} \tag{16}$$

$$\triangle w_\phi = \tfrac{\delta}{\delta w_\phi}\left\{ \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \underset{z^{(n)} \sim p(z|\theta^*)}{E} \left[\ln D\left(z^{(n)}\right)\right] \right\} \tag{17}$$

A.5.2 Training the encoder

Encoder training, $T = 0$**:** The encoder $q(z \mid x)$ maps x to z in the latent space. However, we seek to minimize between $q(z \mid x)$ and $p(z \mid \theta^*)$. Meaning that we desire to produce a sample z from $q(z \mid x)$ that is as close as possible to the sampling from $p(z \mid \theta^*)$. The discriminator path updates the encoder weight $w_\rho = \{w_{ij}, w_{j\varsigma}\}$ in Fig 1. Given N samples, eqn (14) compute the forward pass. Then, eqn (18) computes the weight update for the encoder for $T = 0$.

$$\triangle w_\rho = \tfrac{\delta}{\delta w_\rho}\left\{ \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \underset{x^{(n)} \sim p_{data}}{E} \left[\ln\left(1 - D\left(G(x^{(n)})\right)\right)\right] \right\} \tag{18}$$

A.6 Implementation of Gaussian mixture model

We define Gaussian mixture model (GMM) Bishop (2006) as an K finite mixture of Gaussians in eqn (19)

$$\ln p(z|\eta, \tau, \psi) = \sum_{n=1}^{N} \sum_{k=1}^{K} \left\{ \ln \psi_k + \ln \mathcal{N}(z_n|\eta_k, \tau_k, \psi_k) \right\} \zeta_{nk} \tag{19}$$

We denote cluster assignment for the n sample as ζ_n where ζ_n is a $1 - of - K$ vector, mixture component as $\psi = \{\psi_k\}_{k=1}^{K} \in \mathbb{R}^Z$, cluster mean $B = \{B_k\}_{k=1}^{K} \in \mathbb{R}^Z$ and cluster precision as $\tau = \{\tau k\}_{k=1}^{K} \in \mathbb{R}^Z$. In standard GMM, the hidden variables are updated by their point estimates, $\hat{\tau_k}, \hat{\eta_k}, \hat{\zeta_{nk}}, \hat{\psi_k}$. The expectation-maximization algorithm allows us to obtain point estimates using the closed form equations Bishop (2006) below

$$\hat{\tau_k} = \left\{ \tfrac{1}{2} \textstyle\sum_{n=1}^{N} \zeta_{nk} \right\} / \left\{ \tfrac{1}{2} \left(\textstyle\sum_{n=1}^{N} (z_n - \eta_k)^2 \zeta_{nk} \right) \right\} \tag{20}$$

$$\hat{\eta_k} = \frac{\sum_{n=1}^{N} \hat{\zeta_{nk}} z_n}{\sum_{n=1}^{N} \hat{\zeta_{nk}}} \tag{21}$$

$$\hat{\zeta_{nk}} = \arg\max_{k} \left\{ \ln \psi_k - \tfrac{1}{2} \left(z_n - \hat{\eta_k} \right)^2 \right\} \zeta_{nk} \tag{22}$$

$$\hat{\psi_k} = \frac{\sum_{n=1}^{N} \zeta_{nk}}{\sum_{k=1}^{K} \sum_{n=1}^{N} \zeta_{nk}} \tag{23}$$