MPCC: Matching Priors and Conditionals for Clustering

Nicols Astorga[1,4][0000−0001−6172−9362], Pablo Huijse[2,4][0000−0003−3541−1697],
Pavlos Protopapas[3], and Pablo Estvez[1,4][0000−0001−9164−4722]

[1] Department of Electrical Engineering, Universidad de Chile
[2] Informatics Institute, Faculty of Engineering Sciences, Universidad Austral de Chile
[3] Institute for Applied Computational Science, Harvard University
[4] Millennium Institute of Astrophysics, Chile
nicolas.astorga.r@ug.uchile.cl, phuijse@inf.uach.cl,
pavlos@seas.harvard.edu, pestevez@ing.uchile.cl

Abstract. Clustering is a fundamental task in unsupervised learning that depends heavily on the data representation that is used. Deep generative models have appeared as a promising tool to learn informative low-dimensional data representations. We propose Matching Priors and Conditionals for Clustering (MPCC), a GAN-based model with an encoder to infer latent variables and cluster categories from data, and a flexible decoder to generate samples from a conditional latent space. With MPCC we demonstrate that a deep generative model can be competitive/superior against discriminative methods in clustering tasks surpassing the state of the art over a diverse set of benchmark datasets. Our experiments show that adding a learnable prior and augmenting the number of encoder updates improve the quality of the generated samples, obtaining an inception score of 9.49 ± 0.15 and improving the Fréchet inception distance over the state of the art by a 46.9% in CIFAR10.

1 Introduction

Clustering is a fundamental unsupervised learning problem that aims to group the input data based on a similarity criterion. Traditionally, clustering models are trained on a transformed low-dimensional version of the original data obtained via feature-engineering or dimensionality reduction *e.g.* PCA. Hence the performance of clustering relies heavily on the quality of the feature space representation. In recent years deep generative models have been successful in learning low-dimensional representations from complex data distributions, and two particular models have gained wide attention: The Variational Autoencoder (VAE) [33], [56] and the Generative Adversarial Network (GAN) [17].

In VAE an encoder and a decoder network pair is trained to map the data to a low-dimensional latent space, and to reconstruct it back from the latent space, respectively. The encoder is used for inference while the decoder is used for generation. The main limitations of the standard VAE are the restrictive assumptions associated with the explicit distributions of the encoder and decoder

outputs. For the latter this translates empirically as loss of detail in the generator output. In GAN a generator network that samples from latent space is trained to mimic the underlying data distribution while a discriminator is trained to detect whether the generated samples are true or synthetic (fake). This adversarial training strategy avoids explicit assumptions on the distribution of the generator, allowing GANs to produce the most realistic synthetic outputs up to date [3], [30], [31]. The weaknesses of the standard GAN are the lack of inference capabilities and the difficulties associated with training (*e.g.* mode collapse).

One would like to combine the strengths of these two models, *i.e.* to be able to infer the latent variables directly from data and to have a flexible decoder that learns faithful data distributions. Additionally, we would like to train simultaneously for feature extraction and clustering as this performs better according to [71], [74], [73]. Extensions of the standard VAE that modify the prior distribution to make it suitable for clustering have been proposed in [29], [9], [28], although they still suffer from too restrictive generator models. On the other hand, the standard GAN has been extended to infer categories [64], [39], [7]. Other works have extended GAN to infer the posterior distribution of the latent variables reporting good results in both reconstruction and generation [5], [13], [10], [65], [40], [41]. These models do inference and have flexible generators but were not designed for clustering.

In this paper we propose a model able to learn good representations for clustering in latent space. The model is called Matching Priors and Conditionals for Clustering (MPCC). This is a GAN-based model with (a) a learnable mixture of distributions as prior for the generator, (b) an encoder to infer the latent variables from the data and (c) a clustering network to infer the cluster membership from the latent variables. Code are available at `github.com/jumpynitro/MPCC`.

2   Background

MPCC is based on a matching joint distribution optimization framework. Let us denote $q(x)$ as the true distribution and $p(z)$ the prior, where $x \in \mathcal{X}$ is the observed variable and $z \in \mathcal{Z}$ is the latent variable, respectively. $q(x)$ and $p(z)$ stand for the marginalization of the inference model $q(x, z)$ and generative model $p(x, z)$, respectively. If the joint distributions $q(x, z)$ and $p(x, z)$ match then it is guaranteed that all the conditionals and marginals also match. Intuitively this means that we can reach one domain starting from the other, *i.e.*, we have an encoder that allows us to reach the latent variables $p(z) \approx q(z) = \mathbb{E}_{q(x)}[q(z|x)]$ and a generator that approximates the real distribution $q(x) \approx p(x) = \mathbb{E}_{p(z)}[p(x|z)]$. Notice that the latter approximation corresponds to a GAN optimization problem. In the case of vanilla GAN the Jensen-Shannon divergence $D_{JS}(q(x)||p(x))$ is minimized, but other distances can be used [1], [19], [54], [48].

Although other classifications can be done [77], we recognize that the joint distribution matching problem can be divided in three general categories: i) matching the joints directly, ii) matching conditionals in $\mathcal{Z}$ and marginals in $\mathcal{X}$, and iii) matching conditionals in $\mathcal{X}$ and marginals in $\mathcal{Z}$. The straight forward

approach is to minimize the distance between the joint distributions using a fully adversarial optimization such as [13], [10], [11], which yields competitive results but still shows difficulties in reconstruction tasks likely affecting unsupervised representation learning. According to [40] these issues are related to the lack of an explicit optimization of the conditional distributions.

Recent works [58], [41], [77] have shown that the VAE [33] loss function (ELBO) is related to matching the inference and generative joint distributions. This can be demonstrated for the KullbackLeibler (KL) divergence of p from q, which we refer as forward KL, as follows:

$$
\begin{aligned}
& D_{KL}(q(z,x)||p(z,x)) \\
= \ & \mathbb{E}_{q(x)}[D_{KL}(q(z|x)||p(z|x))] + D_{KL}(q(x)||p(x)) \\
= \ & \mathbb{E}_{q(x)}\mathbb{E}_{q(z|x)}[-\log p(x|z)] + \mathbb{E}_{q(x)}[D_{KL}(q(z|x)||p(z))] + \mathbb{E}_{q(x)}[\log q(x)] \\
= \ & \mathbb{E}_{q(x)}[-\text{ELBO}] + \mathbb{E}_{q(x)}[\log q(x)], \quad\quad\quad\quad\quad\quad\quad\quad (1)
\end{aligned}
$$

hence maximizing the ELBO can be seen as matching the conditionals in latent space $\mathcal{Z}$ and the marginals in data space $\mathcal{X}$ (see the second line in Eq. 1). The proof for the first equivalence in Eq. 1 can be found in the Appendix A.

In order to avoid latent collapse and the parametric assumptions of VAE, AIM [41] proposed the opposite, *i.e.* to match the conditionals in data space and the marginals in latent space. Starting from the KL divergence of q from p, which we refer as reverse KL, they obtained the following:

$$
\begin{aligned}
& D_{KL}(p(z,x)||q(z,x)) \\
= \ & \mathbb{E}_{p(z)}[D_{KL}(p(x|z)||q(x|z))] + D_{KL}(p(z)||q(z)) \\
= \ & \mathbb{E}_{p(z)}\mathbb{E}_{p(x|z)}[-\log q(z|x)] + \mathbb{E}_{p(z)}[D_{KL}(p(x|z)||q(x))] + \mathbb{E}_{p(z)}[\log p(z)], \quad (2)
\end{aligned}
$$

where $p(z)$ is a fixed parametric distribution hence $\mathbb{E}_{p(z)}[\log p(z)]$ is constant. Therefore [41] achieves the matching of joint distributions by minimizing $D_{KL}(p(x|z)||q(x))$ to learn the real domain, and maximizing the likelihood of the encoder $\mathbb{E}_{p(x|z)}[\log q(z|x)]$. This allows obtaining an overall better performance than [13], [10], [40] in terms of reconstruction and generation scores. This method matches the conditional distribution explicitly, uses a flexible generator [17] and avoids latent collapse problems [44].

Lot of research has been done in unsupervised and semi supervised learning using straight forward joint distribution optimization [10], [13], [11], [26], and even more for conditional in latent space decomposition [33], [34], [46], [45]. In this work we explore the representation capabilities of the decomposition proposed in [41]. Our main contributions are:

- A mathematical derivation that allows us to have a varied mixture of distributions in latent space enforcing its clustering capabilities. Based on this derivation we developed a new generative model for clustering called MPCC, trained by matching prior and conditional distributions jointly.
- A comparison with the state-of-the-art showing that MPCC outperforms generative and discriminative models in terms of clustering accuracy and generation quality.

- An ablation study of the most relevant parameters of MPCC and a comparison with the AIM baseline [41] using state of the art architectures [11].

3   Method

3.1   Model definition

MPCC extends the usual joint distribution of variables $x \in \mathcal{X}$ and $z \in \mathcal{Z}$ incorporating an additional latent variable, $y \in \mathcal{Y}$, which represents a given cluster. We specify the graphical models for *generation* and *inference* as

- $p(x, z, y) = p(y)p(z|y)p(x|z, y),$
- $q(x, z, y) = q(y|z)q(z|x)q(x),$

respectively. The only assumption in the graphical model is $q(y|z) = q(y|z, x)$, *i.e.* z contains all the information from x that is necessary to estimate y.

For generation, we seek to match the decoder $p(x|z, y)$ to the real data distribution $q(x)$. The latent variable is defined by the conditional distributions $p(z|y)$ which in general can be any distribution under certain restrictions (Section 3.3). The marginal distribution $p(y)$ is defined as multinomial with weight probabilities ϕ. Note that under this graphical model the latent space becomes multimodal defined by a mixture of distributions $p(z) = \sum_y p(y)p(z|y)$.

In the inference procedure the latent variables are obtained by the conditional posterior $q(z|x)$ using the empirical data distribution $q(x)$. The distribution $q(y|z)$ is a posterior approximation of the cluster membership of the data.

We call our model Matching Priors and Conditionals for Clustering (MPCC) and we optimize it by minimizing the reverse Kullback-Leibler divergence of the conditionals and priors between the inference and generative networks as follows:

$$
\begin{aligned}
&D_{KL}\left(p(x, z, y)||q(x, z, y)\right) \\
=\ &\mathbb{E}_{p(z,y)}[D_{KL}(p(x|z, y)||q(x|z, y))] \\
&+ \mathbb{E}_{p(y)}[D_{KL}(p(z|y)||q(z|y))] + D_{KL}(p(y)||q(y)).
\end{aligned} \tag{3}
$$

The proof for Eq. (3) can be found in Appendix A. In the following sections we derive a tractable expression for Eq. (3) and present the MPCC algorithm.

3.2   Loss function

Because $q(y)$, $q(z|y)$ and $q(x|z, y)$ are impossible to sample from, we derive a closed-form solution for Eq. (3). In particular for any fixed y and z we can decompose $D_{KL}(p(x|z, y)||q(x|z, y))$ as follows:

$$
\begin{aligned}
&D_{KL}(p(x|z, y)||q(x|z, y)) \\
=\ &\mathbb{E}_{p(x|z,y)}\left[\log \frac{p(x|z, y)}{q(x)} \frac{q(z, y)}{q(z, y|x)}\right] \\
=\ &\mathbb{E}_{p(x|z,y)}\left[\log \frac{p(x|z, y)}{q(x)} - \log q(y|z) - \log q(z|x) + \log q(z|y) + \log q(y)\right].
\end{aligned}
$$
$$\tag{4}$$

Adding $\log p(z|y) + \log p(y) - \log q(z|y) - \log q(y)$ to both sides of Eq. (4) and taking the expectation with respect to $p(z, y)$ the Eq. (3) is recovered. After adding these terms and taking the expectation we can collect the resulting right hand side of Eq. (4) as follows:

$$
\begin{aligned}
& \mathbb{E}_{p(z,y)}[D_{KL}(p(x|z,y)||q(x|z,y)) + D_{KL}(p(z|y)||q(z|y)) + D_{KL}(p(y)||q(y))] \\
= \ & \underbrace{\mathbb{E}_{p(y)p(z|y)}[D_{KL}(p(x|z,y)||q(x))]}_{\textbf{Loss I}} + \underbrace{\mathbb{E}_{p(y)p(z|y)p(x|z,y)}[-\log q(z|x) - \log q(y|z)]}_{\textbf{Loss II}} \\
+ \ & \underbrace{\mathbb{E}_{p(z|y)p(y)}[\log p(y) + \log p(z|y)]}_{\textbf{Loss III}},
\end{aligned} \tag{5}
$$

where **Loss I** seeks to match the true distribution $q(x)$, **Loss II** is related to the variational approximation of the latent variables and **Loss III** is associated with the distribution of the cluster parameters. The right hand term of Eq. (5) is a loss function, composed of three terms with distributions that we can sample from. In the next section we explain the strategy to optimize each of the terms of the proposed loss function.
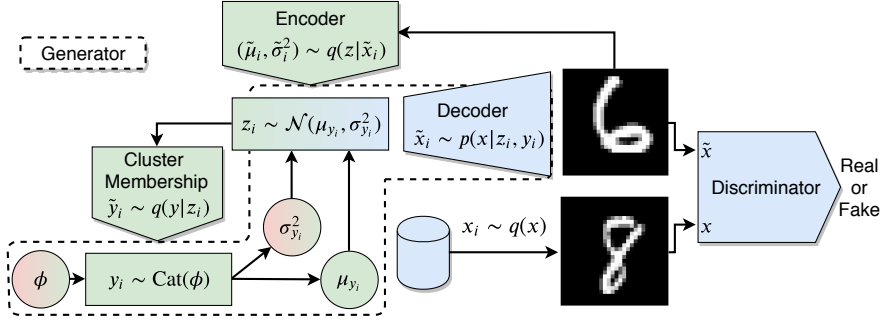
MPCC follows the idea that the data space $\mathcal{X}$ is compressed in the latent space $\mathcal{Z}$ and a separation in this space will likely partition the data in the most representatives clusters $p(z|y)$. The separability of these conditional distributions will be enforced by $q(y|z)$ which also backpropagates through the parameters of $p(z|y)$. The connection with the data space is through the decoder $p(x|z,y)$ for generation and the encoder $q(z|x)$ for inference.

3.3   Optimizing MPCC

In what follows we describe the assumptions made in the distributions of the graphical model and how to optimize Eq. 5. For simplicity we assume the conditional $p(z|y)$ to be a Gaussian distribution, but other distributions could be used with the only restriction being that their entropy should have a closed-form or at least a bound (second term in **Loss III**). In our experiments the latent variable $z|y \sim \mathcal{N}(\mu_y, \sigma_y^2)$ is sampled using the reparameterization trick [33], *i.e.* $z = \mu_y + \sigma_y \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$ and $\odot$ is the Hadamard product. The parameters μ_y, σ_y^2 are learnable and they are conditioned on y. Under Gaussian conditional distribution the latent space becomes a GMM, as we can observe mathematically $p(z) = \sum_y p(y)p(z|y) = \sum_y p(y)\mathcal{N}(\mu_y, \sigma_y^2)$.

The distribution $p(x|z,y)$ is modeled by a neural network and trained via adversarial learning, *i.e.* it does not require parametric assumptions. The inferential distribution $q(z|x)$ is also modeled by a neural network and its distribution is assumed Gaussian for simplicity. The categorical distribution $q(y|z)$ may also be modeled by a neural network but we propose a simpler approach based on the membership from the latent variable z to the Gaussian components. A diagram of the proposed model considering these assumptions is shown in Fig. 1. We now expand on this for each of the losses in Eq. 5.

Loss I: Instead of minimizing the Kullback-Leibler divergence shown in the first term on the right hand of Eq. (5) we choose to match the conditional decoder

Fig. 1. Diagram of the MPCC model. The blue colored elements are associated with **Loss I** (Eq. 6). The green colored elements are associated with **Loss II** (Equations 7 and 8). The red colored elements are associated with **Loss III** (Eq. 10). The dashed line corresponds to the generator (GMM plus decoder).

$p(x|z, y)$ with the empirical data distribution $q(x)$ using a generative adversarial approach. The GAN loss function can be formulated as [12]

$$\max_{D} \; \mathbb{E}_{x\sim q(x)}[f(D(x))] + \mathbb{E}_{\tilde{x}\sim p(x,z,y)}[g(D(\tilde{x}))],$$
$$\min_{G} \; \mathbb{E}_{\tilde{x}\sim p(x,z,y)}[h(D(\tilde{x}))], \tag{6}$$

where D and G are the discriminator and generator networks, respectively, and tilde is used to denote sampled variables. For all our experiments we use the hinge loss function [42], [67], *i.e.* $f = -\min(0, o - 1)$, $g = \min(0, -o - 1)$ and $h = -o$, being o the output of the discriminator. The parameters and distribution associated with **Loss I** are colored in blue in Fig.1.

Loss II: The first term of this loss is estimated through Monte Carlo sampling as

$$\mathbb{E}_{p(y)p(z|y)p(x|z,y)}[-\log q(z|x)]$$
$$= \mathbb{E}_{y_i\sim p(y),z_i\sim p(z|y=y_i),\tilde{x}_i\sim p(x|z=z_i,y=y_i)} \underbrace{\left[\sum_{j=1}^{J} \frac{1}{2}\log(2\pi\tilde{\sigma}_{ij}^2) + \frac{(z_{ij} - \tilde{\mu}_{ij})^2}{2\tilde{\sigma}_{ij}^2}\right]}_{L_q(\tilde{\mu}_i,\tilde{\sigma}_i^2,z_i)}, \tag{7}$$

where J is the dimensionality of the latent variable z. By minimizing Eq. (7) we are maximizing the log-likelihood of the encoder $q(z|x)$ with respect to the Gaussian prior $p(z|y)$. This reconstruction error is estimated by matching the samples $z_i \sim p(z|y = y_i)$ with the Gaussian distribution $(\tilde{\mu}_i, \tilde{\sigma}_i^2) \sim q(z|x = \tilde{x}_i)$, where $\tilde{x}_i$ is the decoded representation of z_i.

The second term of **Loss II** is equivalent to the cross-entropy between the sampled label $y_i \sim p(y)$ and the estimated cluster membership $\tilde{y}_i$

$$L_c(y_i, \tilde{y}_i) = -\sum_{k=1}^{K} y_{ik} \log \tilde{y}_{ik}, \tag{8}$$

where K is the number of clusters and

$$\tilde{y}_{im} = q(y = m | z = z_i) = \frac{\mathcal{N}(z_i | \mu_m, \sigma_m^2)}{\sum_{k=1}^{K} \mathcal{N}(z_i | \mu_k, \sigma_k^2)}, \tag{9}$$

is the membership of z_i to the m-th cluster. The parameters μ_m and σ_m^2 are learnable, and $m \in [1, \ldots, K]$ is the index corresponding to each cluster. The parameters and distribution associated with **Loss II** are colored in green in Fig. 1. In practice Eq. 9 is estimated using the log-sum-exp trick.

Loss III: This loss is associated with the regularization of the Gaussian mixture model parameters ϕ, μ and σ^2 and has a closed form

$$\mathbb{E}_{p(y)p(z|y)}[\log p(y) + \log p(z|y)]$$
$$= \underbrace{\sum_{k=1}^{K} \phi_k \left[\log \phi_k - \sum_{j=1}^{J} \left(\frac{1}{2} + \frac{1}{2} \log(2\pi\sigma_{kj}^2) \right) \right]}_{L_p(\phi, \sigma^2)}, \tag{10}$$

where the first term corresponds to the entropy maximization of the mixture weights, *i.e.* in general every Gaussian will not collapse to less than K modes of the data distribution which is a solution with lower entropy. In our experiments we fix $\phi_k = 1/K$, *i.e.* ϕ is not learnable. The second term is a regularization for the variance (entropy) of each Gaussian which avoids the collapse of $p(z|y)$. The parameters associated with **Loss III** are shown in red in Fig. 1.

Loss I scale differs from that of the terms associated with the latent variables. To balance all terms we multiply Eq. (7) by one over the dimensionality of x[5] and the second term of Eq. (10) by one over the dimensionality of the latent variables. During training **Loss III** is weighted by a constant factor λ_p. We explain how this constant is set in Section 5.3. The full procedure to train the MPCC model is summarized in Algorithm 1. Note that MPCC is scalable in the number of clusters since Eq. 7 is a Monte Carlo approximation in y and the cost of Eq. 10 is low since J is small in comparison to the data dimensionality.

4   Related methods

In Section 3 we showed that the latent space of MPCC it is reduced to a GMM under Gaussian conditional distribution. Because all the experiments are performed based on this assumption in this section we summarize the literature of generative and autoencoding models that consider GMMs.

---

[5] If x is an image then its dimensionality would be *channels* × *height* × *width*

Algorithm 1 MPCC algorithm

1: $K, J \leftarrow$ Set number of clusters and latent dimensionality
2: $\eta, \eta_p \leftarrow$ Set learning rates
3: $\theta_g, \theta_d, \theta_e \leftarrow$ Initialize network parameters
4: $\phi, \mu, \sigma^2 \leftarrow$ Initialize GMM parameters
5: $\theta_c \leftarrow [\phi, \mu, \sigma^2]$
6: **repeat**
7:    **for** D_{steps} **do**
8:       $x_1, \ldots, x_n \sim q(x)$ ▷ Draw n samples from empirical distribution
9:       $y_1, \ldots, y_n \sim p(y)$ ▷ Draw n samples from categorical prior
10:      $z_i \sim p(z|y = y_i), \quad i = 1, \ldots, n$ ▷ Draw n samples from Gaussian conditional prior
11:      $\tilde{x}_i \sim p(x|z = z_i, y = y_i), \quad i = 1, \ldots, n$ ▷ Generate samples using generator network
12:      $\theta_d \leftarrow \theta_d + \eta \nabla_{\theta_d} \left[\frac{1}{n} \sum_{j=1}^n f(D(x_j)) + \frac{1}{n} \sum_{i=1}^n g(D(\tilde{x}_i)) \right]$ ▷ Gradient update on
   discriminator network
13:    **end for**
14:    $y_1, \ldots, y_n \sim p(y)$ ▷ Draw n samples from categorical prior
15:    $z_i \sim p(z|y = y_i), \quad i = 1, \ldots, n$ ▷ Draw n samples from Gaussian conditional prior
16:    $\tilde{x}_i \sim p(x|z = z_i, y = y_i), \quad i = 1, \ldots, n$ ▷ Generate samples using generator network
17:    $(\theta_g, \theta_c) \leftarrow (\theta_g, \theta_c) - \eta \nabla_{(\theta_g, \theta_c)} \frac{1}{n} \sum_{i=1}^n h(D(\tilde{x}_i))$ ▷ Gradient update on generator network
18:    **for** E_{steps} **do**
19:       $y_1, \ldots, y_n \sim p(y)$ ▷ Draw n samples from categorical prior
20:       $z_i \sim p(z|y = y_i), \quad i = 1, \ldots, n$ ▷ Draw n samples from Gaussian conditional prior
21:       $\tilde{x}_i \sim p(x|z = z_i, y = y_i), \quad i = 1, \ldots, n$ ▷ Generate samples using generator network
22:       $(\tilde{\mu}_i, \tilde{\sigma}_i^2) \sim q(z|x = \tilde{x}_i), \quad i = 1, \ldots, n$ ▷ Encode $\tilde{x}$ to obtain mean and variance
23:       $\theta_e \leftarrow \theta_e - \eta \nabla_{\theta_e} \frac{1}{n} \sum_{i=1}^n L_q(\tilde{\mu}_i, \tilde{\sigma}_i^2, z_i)$ ▷ Gradient update on encoder network
24:       **if** first E_{step} **then**
25:          $\tilde{y}_i \sim q(y|z = z_i), \quad i = 1, \ldots, n$
26:          $\theta_c \leftarrow \theta_c - \eta_p \nabla_{\theta_c} \left[\frac{1}{n} \sum_{i=1}^n L_c(y_i, \tilde{y}_i) + \lambda_p \cdot L_p(\phi, \sigma^2)) \right]$ ▷ Gradient update on Prior
   parameters
27:       **end if**
28:    **end for**
29: **until** convergence

The combination of generative models and GMMs is not new. Several methods have applied GMM in autoencoding [68], [78] or GAN [20], [57] applications without clustering purposes. Other approaches have performed clustering but are not directly comparable since they use mixtures of various generators and discriminators [76] or fixed priors with ad-hoc set parameters [2].

Among the related works on generative models for clustering the closest approaches to our proposal are ClusterGAN [51] and Variational Deep Embedding [28]. ClusterGAN differs from our model in that it sets the dimensions of the latent space as either continuous or categorical while MPCC uses a continuous latent space which is conditioned on the categorical variable y. On the other hand, Variational Deep Embedding (VADE) differs greatly in the training procedure, despite its similar theoretical basis. VADE, as a variational autoencoder model, matches the joint distributions in the forward KL sense $D_{KL}(q(x, z, y)||p(x, z, y))$ by matching the posteriors and the marginals in data space as demonstrated in Appendix B. MPCC optimizes the reverse KL, *i.e.* matching the priors in latent space and conditionals in data space. Optimizing different KLs yield notably different decompositions and thus training procedures. For the forward KL [28] in addition to the challenges in scaling to larger dimension (Section 2) it is more difficult to generalize the latent space to any multi-modal distribution, we briefly discuss the reasons for this in Appendix B.

5   Experiments

5.1   Quantitative Comparison

Following [71], the performance of MPCC is measured using the clustering accuracy metric in which each cluster is assigned to the most frequent class in the cluster. Formally this is defined as

$$\text{ACC} = \max_{m \in \mathcal{M}} \frac{\sum_{i=1}^{N} \mathbb{1}\{y_i = m(c_i)\}}{N}, \tag{11}$$

where N is the total number of samples, y_i is the ground truth, $c_i = \arg\max_k q(y = k|z = z_i)$ is the predicted cluster and $\mathcal{M}$ is the space of all possible mappings between clusters and labels.

   To measure the quality of the samples generated by MPCC we use the inception score (IS) [60] and the Fréchet inception distance (FID) [23].

5.2   Datasets

In order to evaluate MPCC we performed clustering in five benchmark datasets: a handwritten digit dataset (MNIST, [38]), a handwritten character dataset (Omniglot, [37]), two color image dataset (CIFAR-10 and CIFAR-100 [36]) and a fashion products image dataset (Fashion-MNIST, [70]). For CIFAR-100 we consider the 20 superclasses. Omniglot was created using the procedure described in [24]. Because the task is fully unsupervised we concatenate the training and test sets as frequently done in the area [24], [4], [71]. All datasets have 10 classes except for Omniglot and CIFAR-20 with 100 and 20 respectively. All images were resized to 32×32 and reescaled to [-1,1] in order to use similar architectures. The CIFAR-10 experiments where IS and FID are reported (tables 1, 2 and 3) were trained using only the training set (50,000 examples) for a fair comparison with the literature. For all clustering experiments we use the same number of cluster as the datasets classes.

5.3   Empirical details

Our architecture is based on optimization techniques used in the BigGAN [3][6], we found that simpler architectures such as DCGAN [55] were not able to learn complex distributions like CIFAR-10 while optimizing the parameters of the prior. Architecture details are given in Appendix C. We consider parameter sharing between the encoder and discriminator, and we test the importance of this in Section 5.5. We set $D_{steps} = 4$ (see Algorithm 1), we found that using smaller values of D_{steps} causes mode collapse problems when training on CIFAR-10 (see Appendix D). A similar effect can be observed when choosing a low number of latent dimensions, therefore we set $J = 128$ in all CIFAR-10 experiments. We made small changes in the architecture and optimization parameters depending on the dataset (see Appendix C).

---

[6] https://github.com/ajbrock/BigGAN-PyTorch

We observed the same relation between batch size and (IS, FID) reported in [3]. However we found artifacts that hurt accuracy performance when using batch size larger than 50. For simplicity we used this value in all experiments.

We consider a weighting factor λ_p for **Loss III** (Eq. 10). We observed that if $\lambda_p = 1$, the standard deviation of the prior σ would increase monotonically, hindering training. On the other hand if λ_p is too small, σ decreases, collapsing at some point. We found empirically that a value of $\lambda_p = 0.01$ combined with a minimum threshold for σ of 0.5 allow the algorithm to converge to good solutions.

The parameter settings indicated above were fixed for all experiments and didn't show a big effect in accuracy performance. In section 5.4 we explore the parameters that most affect the training. We trained all experiments for 75,000 iterations, except for MNIST and Omniglot which iterate for 125,000. For unconditional and conditional training we kept the model of the last iteration.

5.4   Ablation study

We found that E_{steps}, the number of encoder updates per epoch, and η_p, the learning rate of the prior parameters, are the most relevant hyperparameters to obtain high accuracy and generation quality. Increasing E_{steps} improves the estimation of $q(z|x)$ since the prior and generator parameters are changing constantly. Rows 1-3 of Table 1 show that the reconstruction error (MSE) decreases with E_{steps}. Generation quality metrics (IS, FID) also improve with larger values of E_{steps} due to the shared parameters between encoder and discriminator.

At initialization the GMM components might not be separated. We observed that the clustering accuracy drops when the generators learns a good approximation of the real distribution before the clusters are separated. To avoid this we use a larger learning rate for the parameters of the GMM prior with respect to the parameters of the generator, encoder and discriminator. Rows 4-6 of Table 1 show that the clustering accuracy increases for larger values of η_p.

5.5   Comparison between GMM Prior and Normal Prior

Using the best configuration found in the ablation study, we performed a comparison with AIM [41], whose results are shown in Table 2. We can consider AIM as a particular case of MPCC where a standard Normal prior is used instead of the GMM prior. AIM does not perform clustering therefore we compare it with MPCC in terms of reconstruction and generation quality. We use the same architecture and parameter settings of MPCC for AIM and we denote this model as AIM-MPCC. To extend our analysis further, Table 2 includes the results of using parameter sharing between the encoder and the discriminator (Appendix C), an idea that was considered but not fully explored in [41].

Note that AIM-MPCC (NS) is considered a baseline because the prior is Gaussian and the encoder doesnt share parameters with the discriminator thus the existence of the encoder doesnt affect the generation quality. In Table 2 we can observe the relevance of parameter sharing, with this conguration ($E_{steps} = 4$) the baseline improves by 0.85 (IS) and 11.13 (FID) points. Adding the GMM

Table 1. E_{steps} correspond to the encoder updates and η_p to the learning rate of the prior parameters. The scale of MSE is in 10^{-3}. The statistics were obtained for at least three runs

E_{steps}	η_p	Acc %	IS	FID	MSE
1	2e-4	41.31	8.82	11.38	1.34
		±5.74	±0.07	±0.23	±0.96
2	2e-4	38.67	9.02	9.66	1.01
		±3.52	±0.05	±3.98	±1.11
4	2e-4	38.27	9.25	7.50	0.331
		±2.46	±0.09	±0.43	±0.09
4	4e-4	52.58	9.44	6.55	0.48
		±5.30	±0.06	±0.33	±0.22
4	6e-4	61.99	9.49	6.59	1.04
		±4.96	±0.15	±0.45	±1.03

Table 2. Comparison of MPCC and AIM-MPCC methods with sharing parameters (S) and without sharing (NS) on the CIFAR-10 dataset. The scale of MSE is in 10^{-3}. The statistics were obtained for five runs

Model	Acc %	IS	FID	MSE
AIM-MPCC (NS)	-	8.24	21.55	1.52
		±0.07	±1.47	±0.84
AIM-MPCC (S)	-	9.09	10.42	1.64
		±0.04	±0.36	±1.42
MPCC (S)	61.99	9.49	6.59	1.04
	±4.96	±0.15	±0.45	±1.03

in the prior improves an additional 0.4 (IS) and 3.93 (FID) points. In total when using the GMM Prior and parameter sharing with additional encoder updates we improve the baseline from 21.55 to 6.59 (69.4% improvement) in terms of FID score and 1.25 points (15.2% improvement) in terms of IS. It is important to notice that these techniques are general and easily applied to any GAN scheme.

5.6   Generation quality of MPCC

Using the configuration of row five from Table 1 we compare MPCC with nine state of the art methods, surpassing them in terms of IS and FID scores in both the unsupervised and supervised setting, as shown in Table 3. The unconditional generation is the most significant with an improvement of 46.9% (FID) over state-of-the-art (SOTA), AutoGAN [16]. Most notably its performance is better than the current best conditional method (BigGAN).

5.7   Clustering experiments

Table 4 shows the clustering results for the selected benchmarks. We observe that in all the available benchmarks MPCC outperform the related methods, VADE [28] and ClusterGAN [51]. In more complex datasets such as CIFAR10, MPCC notably surpass discriminative based models (*e.g.* [24], [27]) which are the most competitive methods in the current literature. For benchmarks with more classes the margin is even larger obtaining improvements over the SOTA of $\sim$ 42% and $\sim$ 9.7% points in Omniglot and CIFAR-20 respectively, demonstrating empirically the scalability of MPCC when using a high number of clusters.

It can be observed that for all datasets our proposed method either achieves or surpasses the SOTA in terms of clustering. Figures 2 and 3 show examples of generated and reconstructed images, respectively, using the MPCC model with the highest accuracy in the MNIST and CIFAR-10 datasets.

Table 3. Inception and FID scores for CIFAR-10, in unconditional and conditional training. Higher IS is better. Lower FID is better. [†]: Average of 10 runs. [‡]: Best of many runs. [††]: Average of 5 runs. Results without symbols are not specified

Model	IS	FID	Model	IS	FID
DCGAN [55]	6.64 ± 0.14	–	WGAN-GP [19]	8.42 ± 0.10	–
SN-GAN[†] [50]	8.22 ± 0.05	21.7 ± 2.1	SN-GAN [†] [50]	8.60 ± 0.08	17.5
AutoGAN [16]	8.55 ± 0.10	12.42	Splitting GAN [‡] [18]	8.87 ± 0.09	–
PG-GAN [‡] [30]	8.80 ± 0.05	–	CA-GAN [†] [52]	9.17 ± 0.13	–
NCSN [63]	8.91	25.32	BigGAN [3]	9.22	–
MPCC[††]	9.49 ± 0.15	6.59 ± 0.45	**MPCC**[††]	9.55 ± 0.08	5.69 ± 0.17

(a) Unconditional (unsupervised) generation   (b) Conditional (supervised) generation

6   Discussion

Our results show that MPCC achieves a superior performance with respect to the SOTA on both clustering and generation quality. We note that the current SOTA on unsupervised and semisupervised learning relies on consistency training [72] and/or data augmentation [27], *i.e.* techniques that are complementary to MPCC and could be used to further improve our results.

To the best of our knowledge MPCC is the first deep generative clustering model capable of dealing with more complex distributions such as CIFAR-10/20 and the first to report clustering accuracy on these datasets. Additionally we empirically prove the scalability of MPCC showing significant improvements in datasets with a larger number of classes, 20 in case of CIFAR-20 and 100 in case of Omniglot, such scalability has not been proven for the current literature on generative models [28], [51], [64], [75].

Our experiments show that MPCC's key innovations: GMM prior, loss function and optimization scheme (*e.g.* extra encoder updates with parameter sharing) are not only relevant to achieve a good clustering accuracy but also allows us to obtain unprecedented results in terms of generation quality (Table 3). Which translates in improvements of 69.4% over the baseline (Table 2) and 46.9% over the SOTA (Table 3) in terms of FID score. We think that the exceptional generation capabilities of MPCC are related to the support that each cluster covers of the real domain. Since each cluster learns a subset of the real distribution the interpolation between two points within a cluster is smoother compared to the case where no latent separation exists. The latter is explained by the learnable shared features which exploit the similarities existing in a cluster and are not present in a fixed global prior (*i.e.* ALI, AIM).

The high generation quality can be appreciated in Fig. 2 (more samples in Appendix E), where many clusters sample consistently different classes. However we can still see some classes mixed in some clusters, for example in columns 7-8 with cats and dogs. MPCC also presents a competitive performance in terms of

Table 4. Clustering accuracies for several methods and datasets. All the results of CIFAR-20 dataset were extracted from [27], the results of IMSAT and DEC from [24], the results of InfoGAN and ClusterGAN from [51] and the remaining from their respective papers. [†]: average of 5 or more runs. [‡]: best of 5 runs. [§]: best of 10 or more runs. [‖]: best of 3 runs. Results without symbols are not specified

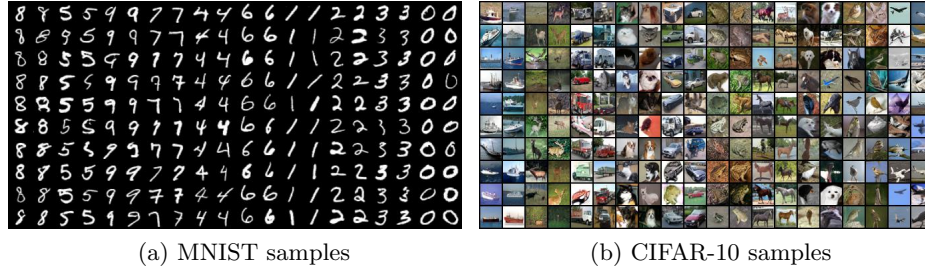| Methods	Datasets				
	MNIST	Onmiglot	FMNIST	CIFAR-10	CIFAR-20
DEC [71]	$84.3^{\S}$	$5.3 \pm 0.3^{\dagger}$	–	$46.9 \pm 0.9^{\dagger}$	18.5
VADE [28]	$94.46^{\S}$	–	–	–	-
InfoGAN [5]	$89.0^{\ddagger}$	–	$61.0^{\ddagger}$	–	-
ClusterGAN [51]	$95.0^{\ddagger}$	–	$63.0^{\ddagger}$	–	-
DAC [4]	$97.75^{\|}$	–	–	$52.18^{\|}$	23.8
IMSAT (VAT) [24]	$98.4 \pm 0.4^{\dagger}$	$24.0 \pm 0.9^{\dagger}$	–	$45.6 \pm 0.8^{\dagger}$	-
ADC [21]	$98.7 \pm 0.6^{\dagger}$	-	-	$29.3 \pm 1.5^{\dagger}$	16.0
SCAE [35]	$98.5 \pm 0.10^{\dagger}$	-	-	$33.48 \pm 0.3^{\dagger}$	-
IIC [27]	$98.4 \pm 0.65^{\dagger}$	-	-	$57.6 \pm 0.3^{\dagger}$	$25.5 \pm 0.46^{\dagger}$
MPCC (Five runs)	98.48 ± 0.52	65.87 ± 1.46	62.56 ± 4.16	64.25 ± 5.31	35.21 ± 1.69
MPCC (Best three runs)	98.76 ± 0.03	66.95 ± 0.62	64.99 ± 2.22	67.73 ± 2.50	36.51 ± 0.71

conditional distribution matching (Fig. 3). The errors observed in reconstruction are semantic and similar to those observed in [11].

MPCC opens the possibility of future research in many relevant topics which are out of the scope of this paper. Based on our experiments the most important extensions are: 1) Experiment with other conditional distributions $p(z|y)$, *e.g.* other exponential-family distributions or other flexible distributions by bounding their entropy (Section 3). This can be suitable for more expressive priors as it's shown in recent work [69]. 2) Experiment with imbalanced distribution of classes by changing ϕ accordingly, we consider this to be a relevant problem in the unsupervised setting which only a few works have addressed [66]. 3) Experiment with higher resolution datasets such as ImageNet [8] or CelebA [43]. Current works on clustering have not focus their attention to higher-resolution due to its complexity, MPCC is a promising approach to tackle this task from a semantic perspective [11].
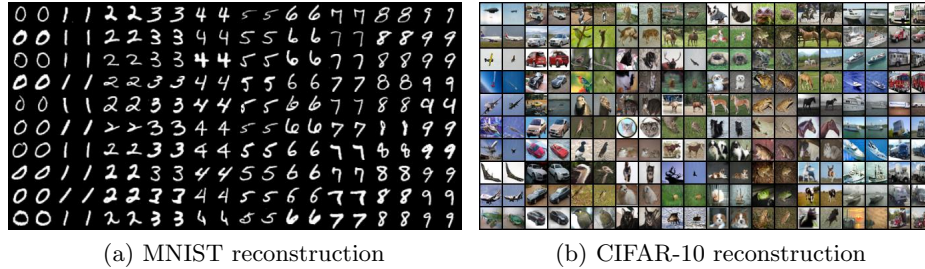
7   Conclusions

We developed a new clustering algorithm called MPCC, derived from a joint distribution matching perspective with a latent space modeled by a mixture distribution. As a deep generative model this algorithm is suitable for interpretable representations, having both an inference and a generative network. The inference network allows us to infer the cluster membership and latent variables from data, while the generator performs sampling conditioned on the cluster category.

An important contribution of this work lays in the solid mathematical and optimization framework on which MPCC is based. This framework is general

(a) MNIST samples

(b) CIFAR-10 samples

Fig. 2. Generated images for a) MNIST and b) CIFAR-10 datasets, respectively. Every two columns we set a different value for the categorical latent variable y. *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$.

(a) MNIST reconstruction

(b) CIFAR-10 reconstruction

Fig. 3. Reconstructions for a) MNIST, and b) CIFAR-10 datasets, respectively. Odd columns represent real data and even columns correspond to their reconstructions.

and we recognize several opportunities to further enhance our model. The results obtained with MPCC improve over the state of the art in clustering methods. Most notably, our model is able to generate samples with an unprecedented high quality, surpassing the state of the art performances in both conditional and unconditional training in the CIFAR-10 dataset.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 214–223. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017), `http://proceedings.mlr.press/v70/arjovsky17a.html`
2. Ben-Yosef, M., Weinshall, D.: Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images. CoRR **abs/1808.10356** (2018), `http://arxiv.org/abs/1808.10356`
3. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=B1xsqj09Fm`
4. Chang, J., Wang, L., Meng, G., Xiang, S., Pan, C.: Deep adaptive image clustering. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 5880–5888 (Oct 2017). https://doi.org/10.1109/ICCV.2017.626
5. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems. pp. 2172–2180 (2016)
6. Chyzak, F., Nielsen, F.: A closed-form formula for the Kullback-Leibler divergence between Cauchy distributions (Dec 2019), `https://hal.inria.fr/hal-02420591`, 8 pages
7. Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.R.: Good semi-supervised learning that requires a bad gan. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 6510–6520. Curran Associates, Inc. (2017), `http://papers.nips.cc/paper/7229-good-semi-supervised-learning-that-requires-a-bad-gan.pdf`
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
9. Dilokthanakul, N., Mediano, P.A., Garnelo, M., Lee, M.C., Salimbeni, H., Arulkumaran, K., Shanahan, M.: Deep unsupervised clustering with gaussian mixture variational autoencoders. arXiv preprint arXiv:1611.02648 (2016)
10. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017), `https://openreview.net/forum?id=BJtNZAFgg`
11. Donahue, J., Simonyan, K.: Large scale adversarial representation learning. ArXiv **abs/1907.02544** (2019)
12. Dong, H.W., Yang, Y.H.: Towards a deeper understanding of adversarial losses. ArXiv **abs/1901.08753** (2019)
13. Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., Courville, A.C.: Adversarially learned inference. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017), `https://openreview.net/forum?id=B1ElR4cgg`
14. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. ICLR (2017), `https://arxiv.org/abs/1610.07629`
15. Figurnov, M., Mohamed, S., Mnih, A.: Implicit reparameterization gradients. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N.,

Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 441–452. Curran Associates, Inc. (2018), `http://papers.nips.cc/paper/7326-implicit-reparameterization-gradients.pdf`

16. Gong, X., Chang, S., Jiang, Y., Wang, Z.: Autogan: Neural architecture search for generative adversarial networks. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2019)

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

18. Grinblat, G.L., Uzal, L.C., Granitto, P.M.: Class-splitting generative adversarial networks (2017)

19. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 5769–5779. NIPS'17, Curran Associates Inc., USA (2017), `http://dl.acm.org/citation.cfm?id=3295222.3295327`

20. Gurumurthy, S., Kiran Sarvadevabhatla, R., Venkatesh Babu, R.: Deligan : Generative adversarial networks for diverse and limited data. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)

21. Häusser, P., Plapp, J., Golkov, V., Aljalbout, E., Cremers, D.: Associative deep clustering: Training a classification network with no labels. In: GCPR (2017)

22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (June 2016). https://doi.org/10.1109/CVPR.2016.90

23. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 6626–6637. Curran Associates, Inc. (2017)

24. Hu, W., Miyato, T., Tokui, S., Matsumoto, E., Sugiyama, M.: Learning discrete representations via information maximizing self-augmented training. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. pp. 1558–1567. ICML'17, JMLR.org (2017), `http://dl.acm.org/citation.cfm?id=3305381.3305542`

25. Jankowiak, M., Obermeyer, F.: Pathwise derivatives beyond the reparameterization trick. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 2235–2244. PMLR, Stockholmsmssan, Stockholm Sweden (10–15 Jul 2018), `http://proceedings.mlr.press/v80/jankowiak18a.html`

26. Jesson, A., Low-Kam, C., Nair, T., Soudan, F., Chandelier, F., Chapados, N.: Adversarially learned mixture model. CoRR **abs/1807.05344** (2018), `http://arxiv.org/abs/1807.05344`

27. Ji, X., Henriques, J.F., Vedaldi, A.: Invariant information clustering for unsupervised image classification and segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9865–9874 (2019)

28. Jiang, Z., Zheng, Y., Tan, H., Tang, B., Zhou, H.: 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)

29. Johnson, M.J., Duvenaud, D.K., Wiltschko, A., Adams, R.P., Datta, S.R.: Composing graphical models with neural networks for structured representations and

fast inference. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2946–2954. Curran Associates, Inc. (2016)

30. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=Hk99zCeAb`

31. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. CoRR **abs/1812.04948** (2018), `http://arxiv.org/abs/1812.04948`

32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014), `http://arxiv.org/abs/1412.6980`, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015

33. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014), `http://arxiv.org/abs/1312.6114`

34. Kingma, D.P., Mohamed, S., Jimenez Rezende, D., Welling, M.: Semi-supervised learning with deep generative models. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 3581–3589. Curran Associates, Inc. (2014), `http://papers.nips.cc/paper/5352-semi-supervised-learning-with-deep-generative-models.pdf`

35. Kosiorek, A., Sabour, S., Teh, Y.W., Hinton, G.E.: Stacked capsule autoencoders. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 15512–15522. Curran Associates, Inc. (2019), `http://papers.nips.cc/paper/9684-stacked-capsule-autoencoders.pdf`

36. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)

37. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science **350**(6266), 1332–1338 (2015)

38. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

39. LI, C., Xu, T., Zhu, J., Zhang, B.: Triple generative adversarial nets. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 4088–4098. Curran Associates, Inc. (2017), `http://papers.nips.cc/paper/6997-triple-generative-adversarial-nets.pdf`

40. Li, C., Liu, H., Chen, C., Pu, Y., Chen, L., Henao, R., Carin, L.: Alice: Towards understanding adversarial learning for joint distribution matching. In: Advances in Neural Information Processing Systems. pp. 5495–5503 (2017)

41. Li, H., Wang, Y., Chen, C., Gao, J.: AIM: Adversarial inference by matching priors and conditionals (2019), `https://openreview.net/forum?id=rJx_b3RqY7`

42. Lim, J.H., Ye, J.C.: Geometric gan (2017)

43. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV) (December 2015)

44. Lucas, J., Tucker, G., Grosse, R.B., Norouzi, M.: Understanding posterior collapse in generative latent variable models. In: DGS@ICLR (2019)

45. Maale, L., Fraccaro, M., Livin, V., Winther, O.: Biva: A very deep hierarchy of latent variables for generative modeling (2019)
46. Maale, L., Snderby, C.K., Snderby, S.K., Winther, O.: Auxiliary deep generative models. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1445–1453. PMLR, New York, New York, USA (20–22 Jun 2016), `http://proceedings.mlr.press/v48/maaloe16.html`
47. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I.: Adversarial autoencoders. In: International Conference on Learning Representations (2016), `http://arxiv.org/abs/1511.05644`
48. Mao, X., Li, Q., Xie, H., Lau, R., Wang, Z., Smolley, S.: Least squares generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2813–2821 (10 2017). https://doi.org/10.1109/ICCV.2017.304
49. Mescheder, L., Nowozin, S., Geiger, A.: Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 2391–2400. PMLR, International Convention Centre, Sydney, Australia (06–11 Aug 2017)
50. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=B1QRgziT-`
51. Mukherjee, S., Asnani, H., Lin, E., Kannan, S.: Clustergan: Latent space clustering in generative adversarial networks. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. pp. 4610–4617. AAAI Press (2019). https://doi.org/10.1609/aaai.v33i01.33014610, `https://doi.org/10.1609/aaai.v33i01.33014610`
52. Ni, Y., Song, D., Zhang, X., Wu, H., Liao, L.: Cagan: Consistent adversarial training enhanced gans. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 2588–2594. International Joint Conferences on Artificial Intelligence Organization (7 2018). https://doi.org/10.24963/ijcai.2018/359, `https://doi.org/10.24963/ijcai.2018/359`
53. Nielsen, F., Nock, R.: Entropies and cross-entropies of exponential families. In: 2010 IEEE International Conference on Image Processing. pp. 3621–3624 (Sep 2010). https://doi.org/10.1109/ICIP.2010.5652054
54. Nowozin, S., Cseke, B., Tomioka, R.: f-gan: Training generative neural samplers using variational divergence minimization. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 271–279. Curran Associates, Inc. (2016)
55. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), `http://arxiv.org/abs/1511.06434`
56. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26

June 2014. pp. 1278–1286 (2014), `http://jmlr.org/proceedings/papers/v32/rezende14.html`

57. Richardson, E., Weiss, Y.: On gans and gmms. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31, pp. 5847–5858. Curran Associates, Inc. (2018), `http://papers.nips.cc/paper/7826-on-gans-and-gmms.pdf`

58. Rosca, M., Lakshminarayanan, B., Mohamed, S.: Distribution matching in variational inference. CoRR **abs/1802.06847** (2018)

59. Ruiz, F.R., Titsias RC AUEB, M., Blei, D.: The generalized reparameterization gradient. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 460–468. Curran Associates, Inc. (2016), `http://papers.nips.cc/paper/6328-the-generalized-reparameterization-gradient.pdf`

60. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved techniques for training gans. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2234–2242. Curran Associates, Inc. (2016), `http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf`

61. Saxe, A.M., Mcclelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In: In International Conference on Learning Representations (2014)

62. Soch, J., Allefeld, C.: Kullback-leibler divergence for the normal-gamma distribution (2016)

63. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution (2019)

64. Springenberg, J.T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), `http://arxiv.org/abs/1511.06390`

65. Srivastava, A., Valkov, L., Russell, C., Gutmann, M.U., Sutton, C.: Veegan: Reducing mode collapse in gans using implicit variational learning. In: Advances in Neural Information Processing Systems. pp. 3308–3318 (2017)

66. Tao, Y., Takagi, K., Nakata, K.: Rdec: Integrating regularization into deep embedded clustering for imbalanced datasets. In: Zhu, J., Takeuchi, I. (eds.) Proceedings of The 10th Asian Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 95, pp. 49–64. PMLR (14–16 Nov 2018), `http://proceedings.mlr.press/v95/tao18a.html`

67. Tran, D., Ranganath, R., Blei, D.M.: Hierarchical implicit models and likelihood-free variational inference (2017)

68. Wang, L., Schwing, A., Lazebnik, S.: Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 5756–5766. Curran Associates, Inc. (2017)

69. Wu, Y., Donahue, J., Balduzzi, D., Simonyan, K., Lillicrap, T.P.: LOGAN: latent optimisation for generative adversarial networks. CoRR **abs/1912.00953** (2019), `http://arxiv.org/abs/1912.00953`

70. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)

71. Xie, J., Girshick, R.B., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. pp. 478–487 (2016), `http://jmlr.org/proceedings/papers/v48/xieb16.html`
72. Xie, Q., Dai, Z., Hovy, E., Luong, M.T., Le, Q.V.: Unsupervised data augmentation for consistency training. arXiv preprint arXiv:1904.12848 (2019)
73. Yang, B., Fu, X., Sidiropoulos, N.D., Hong, M.: 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)
74. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5147–5156 (2016)
75. Yang, L., Cheung, N.M., Li, J., Fang, J.: Deep clustering by gaussian mixture variational autoencoders with graph embedding. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
76. Yu, Y., Zhou, W.J.: Mixture of gans for clustering. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 3047–3053. International Joint Conferences on Artificial Intelligence Organization (7 2018). https://doi.org/10.24963/ijcai.2018/423, `https://doi.org/10.24963/ijcai.2018/423`
77. Zhao, S., Song, J., Ermon, S.: The information autoencoding family: A lagrangian perspective on latent variable generative models. arXiv preprint arXiv:1806.06514 (2018)
78. Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., Chen, H.: Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=BJJLHbb0-`

A   Matching marginals and conditionals is equivalent to matching joints

In the following sections we demonstrate that for the KullbackLeibler divergence matching the marginals and conditionals is equivalent to matching the joint distributions. In Section A.1 we demonstrate this equivalence for the two variable case using the forward KL [33], this can be trivially demonstrated for the reverse KL [41] interchangeably replacing the variables z and x, and the models p and q. In Section A.2 we demonstrate the equivalence for the three variable case using the reverse KL (MPCC). The demonstration for the three-variable forward KL is equivalent requiring only to replace the variables y and x, and the models p and q.

A.1   Matching marginals and conditionals, two variable case

In Section 2 of the paper we show that maximizing the VAE objective (ELBO) can be interpreted as matching the conditional distributions in observed space and the marginal distributions in latent space. Let $p(x, z)$ and $q(x, z)$ be the decoder (*generation*) and encoder (*inference*) joint distributions, respectively, where x represents the observed variable (data) and z represents the continuous latent variable. The dependencies in these distributions are expressed as $p(x, z) = p(z)p(x|z)$ and $q(x, z) = q(z|x)q(x)$. In what follows we prove, for the two variable case, that matching conditionals and marginals is equivalent to matching the joint distributions in terms of the Kullback-Leibler divergence. Starting from the KL divergence between the joint distributions we can show that

$$
\begin{aligned}
&D_{\mathrm{KL}}\left(q(x, z)||p(x, z)\right) \\
&= \int_x \int_z q(x, z) \log \frac{q(x, z)}{p(x, z)} \, dz \, dx \\
&= \int_x q(x) \int_z q(z|x) \log \frac{q(z|x)}{p(z|x)} \, dz \, dx + \int_x q(x) \log \frac{q(x)}{p(x)} \, dx \\
&= \mathbb{E}_{x \sim q(x)}\left[D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right)\right] + D_{\mathrm{KL}}\left(q(x)||p(x)\right),
\end{aligned} \tag{12}
$$

i.e. the forward KL divergence between the joints is indeed equivalent to the forward KL between the marginals in data space plus the expected value under the data distribution of the forward KL between the conditionals in latent space.

A.2   Matching priors and conditionals for three variables *a.k.a* MPCC case

In Section 3 of the paper we define the MPCC model starting from a joint distribution matching perspective. Let $p(x, z, y)$ and $q(x, z, y)$ be the decoder (*generation*) and encoder (*inference*) joint distributions, respectively, where x represents the observed variable (data), z represents the continuous latent variable and y represents the cluster membership. The dependencies in these distributions are

expressed as $p(x, z, y) = p(y)p(z|y)p(x|z, y)$ and $q(x, z, y) = q(y|z)q(z|x)q(x)$. The only assumption is that $q(y|x, z) = q(y|z)$. In what follows we prove, for the three variable case, that matching conditionals and marginals is equivalent to matching the joint distributions in terms of the Kullback-Leibler divergence.

Starting from the (reverse) KL divergence between the joint distributions we show that

$$
\begin{aligned}
& D_{\mathrm{KL}}\left(p(x, z, y)||q(x, z, y)\right) \\
= {} & \int_x \int_y \int_z p(x, z, y) \log \frac{p(x, z, y)}{q(x, z, y)}\, dx\, dy\, dz \\
= {} & \int_y \int_z p(z, y) \int_x p(x|z, y) \log \frac{p(x|z, y)}{q(x|z, y)}\, dx\, dy\, dz \\
& + \int_y p(y) \int_z p(z|y) \log \frac{p(z|y)}{q(z|y)}\, dz\, dy + \int_y p(y) \log \frac{p(y)}{q(y)}\, dy \\
= {} & \mathbb{E}_{z, y \sim p(z, y)}\left[D_{\mathrm{KL}}\left(p(x|z, y)||q(x|z, y)\right)\right] \\
& + \mathbb{E}_{y \sim p(y)}\left[D_{\mathrm{KL}}\left(p(z|y)||q(z|y)\right)\right] + D_{\mathrm{KL}}\left(p(y)||q(y)\right),
\end{aligned}
\tag{13}
$$

i.e. the KL between the joints is the sum of the KL divergences for $x|(z, y)$, $z|y$ and y, respectively. The KL divergence is non-negative so if we match the priors and conditionals then the joints have to match too.

B   Variational Deep Embedding (VaDE)

B.1   Variational Deep Embedding matches conditionals and marginals in data space

Here we show that Variational Deep Embedding (VaDE) [28] is in fact matching the joint distributions of the encoder and decoder by matching posteriors and marginal in the space of the observed variable x (data). We start by expanding the divergence between the joint distribution of the encoder and decoder as:

$$
\begin{aligned}
& D_{\mathrm{KL}}\left(q(x, z, y)||p(x, z, y)\right) \\
= {} & \int_x \int_y \int_z q(x, z, y) \log \frac{q(x, z, y)}{p(x, z, y)}\, dx\, dy\, dz \\
= {} & \mathbb{E}_{z, x \sim q(z, x)}\left[D_{\mathrm{KL}}\left(q(y|z, x)||p(y|z, x)\right)\right] \\
& + \mathbb{E}_{x \sim q(x)}\left[D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right)\right] + D_{\mathrm{KL}}\left(q(x)||p(x)\right).
\end{aligned}
\tag{14}
$$

The first divergence in the right hand side of Eq. (14) is

$$
\begin{aligned}
D_{\mathrm{KL}}\left(q(y|z, x)||p(y|z, x)\right) &= \int q(y|x) \log \frac{q(y|x)p(z)}{p(z|y)p(y)}\, dy \\
&= \mathbb{E}_{y \sim q(y|x)}\left[\log \frac{q(y|x)}{p(z|y)p(y)}\right] + \log p(z),
\end{aligned}
\tag{15}
$$

where we used the replacements $q(y|z, x) = q(y|x)$ and $p(y|z, x) = \frac{p(x|z)p(z|y)p(y)}{p(x|z)p(z)}$, which come from the graphical model assumptions considered in [28].

The second divergence in the right hand side of Eq. (14) is

$$D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right) = \int q(z|x) \log \frac{q(z|x)p(x)}{p(x|z)p(z)}\, dz$$

$$= \mathbb{E}_{z \sim q(z|x)}\left[\log \frac{q(z|x)}{p(x|z)} - \log p(z)\right] + \log p(x), \qquad (16)$$

and the third divergence in the right hand side of Eq. (14) is

$$D_{\mathrm{KL}}\left(q(x)||p(x)\right) = \mathbb{E}_{x \sim q(x)}\left[\log q(x) - \log p(x)\right]. \qquad (17)$$

If we add the expectation over $q(z, x) = q(z|x)q(x)$ of Eq. (15) with the expectation over $q(x)$ of Eq. (16) and Eq. (17) we obtain:

$$\mathbb{E}_{z,x \sim q(z,x)}\left[D_{\mathrm{KL}}\left(q(y|z,x)||p(y|z,x)\right)\right] + \mathbb{E}_{x \sim q(x)}\left[D_{\mathrm{KL}}\left(q(z|x)||p(z|x)\right)\right]$$
$$+\ D_{\mathrm{KL}}\left(q(x)||p(x)\right)$$
$$=\ \mathbb{E}_{q(x)}[\log q(x) - \mathbb{E}_{z,y \sim q(z,y|x)}[\log p(x|z) - \log q(z|x) - \log q(y|x)$$
$$+\ \log p(z|y) + \log p(y)]]$$
$$=\ \mathbb{E}_{q(x)}\left[\log q(x) - \mathcal{L}_{\mathrm{VaDE}}(x)\right],$$

where $\mathcal{L}_{\mathrm{VaDE}}(x)$ corresponds to Eq. (9) in [28]. This means that by maximizing VaDE's loss function one is matching the conditionals and marginals between encoder and decoder in data space. Note that the entropy of the data distribution $\mathbb{E}_{q(x)}\left[\log q(x)\right]$ is constant during optimization.

B.2   Why extending VaDE to any multi-modal distribution is harder than MPCC?

In MPCC the latent space can be naturally extended to any mixture of distributions, the only requirement being that the entropy of each distribution component $p(z|y)$ should have a closed-form or at least a bound. In general any model decomposed by the reverse KL enjoy this property.

Forward KL decompositions, such as the case of VAE and VaDE, need a closed-form solution for the divergence between the posterior and the prior. In VaDE this term corresponds to

$$\mathbb{E}_{q(x)}\mathbb{E}_{q(z,y|x)}[\log q(z|x) - \log p(z|y)] = \mathbb{E}_{q(x)}\mathbb{E}_{q(y|x)}[D_{KL}(q(z|x)||p(z|y))], \quad (18)$$

which has a closed-form since $q(z|x)$ and $p(z|y)$ are Gaussians. Other distributions can be used however they need to be from the exponential family and to have the same distribution [53], although some exceptions exists [62], [6]. In addition to the exponential family requirement, a reparameterization trick is

needed for the posterior distribution further limiting the distributions that can be used and requiring other forms of reparameterization [15], [59], [25].

Alternatively, adversarial training can be used to match the marginal posterior with more flexible priors. However it has been observed [58] that this kind of optimization [47], [49] underestimates its Kullback-Leibler divergence and also worsen the likelihood of the decoder likely affecting its clustering capabilities.

C   Details on neural network architectures

In MPCC we use the BigGAN model techniques [3] as a base for all our experiments. This architecture employs ResNet [22] and Spectral Normalization [50]. The residual block components of the generator and discriminator/encoder are shown in Fig. 4 (a) and (b), respectively. All the 3×3 Conv use a padding equal to one while 1×1 Conv have no padding. The upsampling operation of the generator is done using bilinear interpolation. A general scheme of the generator, discriminator and encoder architecture is shown in Fig. 5. The first residual block of the discriminator/encoder inverts the order of the 1×1 Conv and the average pooling and omits the first ReLU activation. Residual blocks with an asterisk correspond to the ones that do not perform average pooling and as a consequence they do not use 1×1 Conv.

Fig. 5 (a) shows the generator used for the CIFAR10/20 datasets. Fig 5 (b) shows the unconditional architecture of the discriminator. In the case of the conditional discriminator a term $\text{Embed}(y) \cdot h$ is added where h is the output of the global sum pooling (see Table 6). We can write the architectures for all datasets in a general way as in Fig. 5 or more specific as in Tables 5, 6 and 7, where C, J and D change between datasets.

As we observed in the paper (Table 3) we found an improvement in terms of sampling quality and reconstruction error when parameters between the discriminator and the encoder are shared. We experimented on the number of residual blocks shared and found that the best performance was obtained when sharing the first three residual blocks.

We use the Adam optimizer [32] with its default parameters $\beta_1 = 0$ and $\beta_2 = 0.999$. We use exponential moving average (EMA) with a decay rate of 0.9999 for the Generator for both sampling and reconstruction task. EMA is applied after the 1000th iteration. All generator, discriminator and encoder parameters use Spectral Normalization and are initialized with $\mathcal{N}(0, 0.02I)$, while an orthogonal initialization is used for prior parameters [61]. With the exception of the prior parameters we use a learning rate of $2e - 4$ for all networks and experiments. For evaluation we use standing statistics [3] *i.e.* in evaluation mode we run many times (in our case 16) the forward propagation of the generator model $\tilde{x} \sim p(x, z, y)$ storing the means and variances aggregated across all forward passes.

We use three techniques depending on the dataset to deliver the latent information z and y into the decoder distribution $p(x|z, y)$. The first two correspond to a hierarchical latent space architecture [3] which concatenate $\text{Embed}(y)$ with
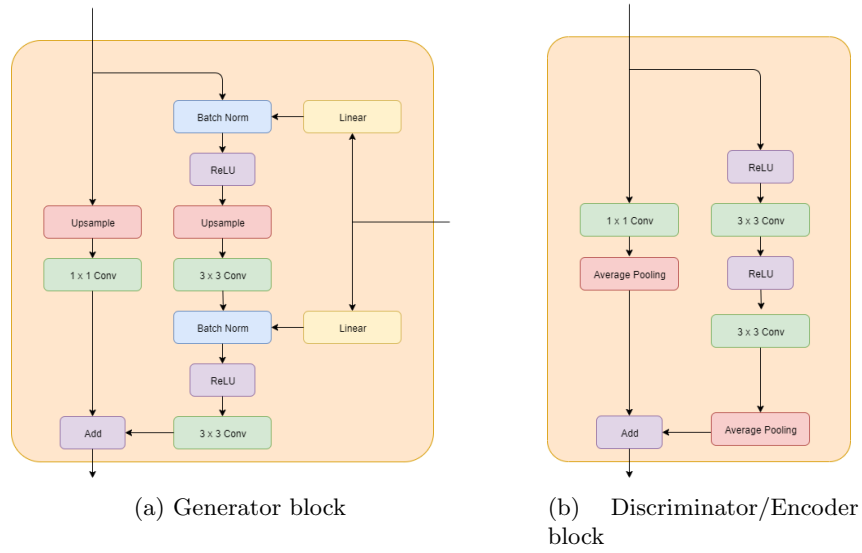
a subset of z, and then a linear transformation to estimate the statistics of the batch norm layers is applied (see Fig. 4). The first method is the one observed in Fig. 5 which splits the latent variable z into equal chunks, delivering each one to a different part of the network. In this case we have four chunks (1 entry + 3 residual blocks). The second method is similar to first, the only difference being that all z is shared and no split operation is done. The schematic of this generator is equivalent to Fig. 5 (a) except that the purple box performs a copy instead of split operation. The third method passes all the latent z as usual [55] and uses conditional batch normalization [14]. This method learns embeddings conditioned on y which are different for each layer, *i.e.* the linear transformation in the yellow boxes of Fig. 4 correspond to an embedding, and the shared embedding should be ignored.

- For CIFAR10 and CIFAR20 we use the first method since this is the default architecture used in BigGAN. For simplicity we kept this configuration for all ablation and clustering experiments with these datasets. We found that mode collapse problems would appear if the third method is used in these datasets. The configuration of the parameters for these datasets is $C = 96$, $D = 3$ (RGB), $J = 128$ and $\eta_p = 6 \cdot 10^{-4}$.

- For datasets with simpler distributions such as MNIST and Omniglot, the third method is more stable and yields the best results. We found that if we use hierarchical latent space architectures poor results were obtained. In particular we observed that the chunks in the first method are decorrelated, which is particularly bad for simpler datasets such as MNIST and Omniglot because the network gains lot of capacity ignoring the embedding y and learning the full real distribution in all the clusters. The configuration of the parameters for these datasets is $D = 1$ (grayscale), $J = 24$ and $\eta_p = 1.6 \cdot 10^{-3}$. For MNIST $C = 12$ and for Omniglot $C = 16$.

- For FMNIST we observed that a poor performance was obtained with both the first and third method. The best results for this dataset were obtained using the second method. The configuration of the parameters for this dataset is $C = 24$, $D = 1$ (gray), $J = 16$ and $\eta_p = 1.6 \cdot 10^{-3}$.
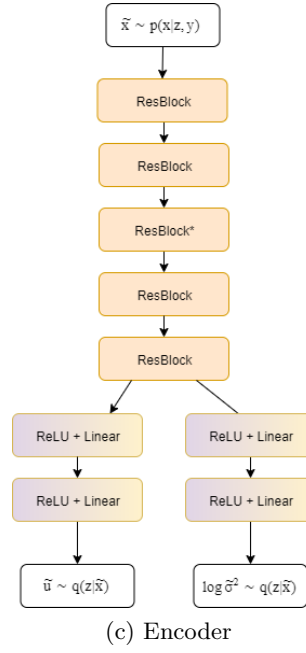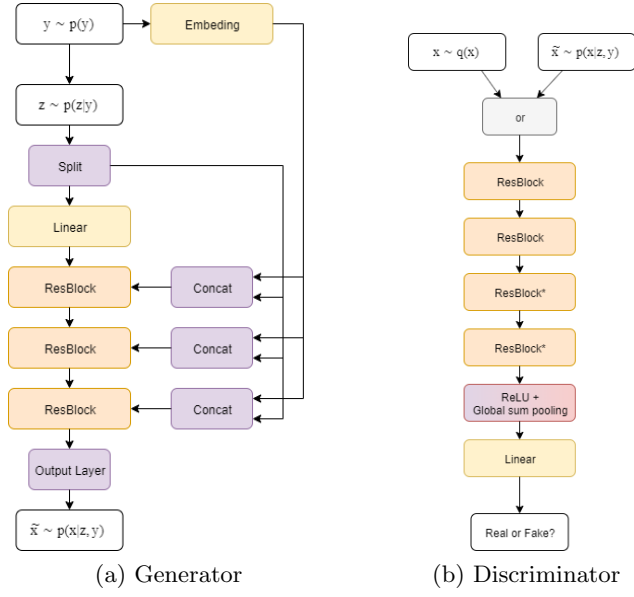
We develop a Pytorch implementation for MPCC based on the implementation of BigGAN[7]. The IS and FID scores are calculated using the official implementations[8]. We run each model in a GeForce RTX 2080 Ti, the amount of time that MPCC iterates depends on the dataset but it is within the range of 12-24 hours.

---

[7] https://github.com/ajbrock/BigGAN-PyTorch
[8] https://github.com/bioinf-jku/TTUR

(a) Generator block

(b)   Discriminator/Encoder block

Fig. 4. Residual blocks used for MPCC generator, discriminator and encoder networks.

(a) Generator

(b) Discriminator

(c) Encoder

Fig. 5. Architectures of MPCC generator, discriminator and encoder networks, respectively

$y_i \in \{0, \ldots, K-1\} \sim Cat(\phi)$
$z_i \in \mathbb{R}^J \sim \mathcal{N}(\mu_{y_i}, \sigma^2_{y_i})$
Share Embed$(y) \in \mathbb{R}^J$
Linear$(J) \to 4 \times 4 \times 4C$
Resblock up $4C \to 4C$
Resblock up $4C \to 4C$
Resblock up $4C \to 4C$
Output Layer: BN, ReLU, 3×3 Conv $C \to 3$ Tanh

Table 5. Generator

$x \in \mathbb{R}^{32 \times 32 \times D}$
Resblock down $4C \to 4C$
Resblock down $4C \to 4C$
Resblock $4C \to 4C$
Resblock $4C \to 4C$
ReLU, Global sum pooling
(linear $\to 1$) if conditional $:+$ Embed$(y)\cdot h$

Table 6. Discriminator

$x \in \mathbb{R}^{32 \times 32 \times D}$
Resblock down $4C \to 4C$
Resblock down $4C \to 4C$
Resblock $4C \to 4C$
Resblock down $4C \to 4C$
Resblock down $4C \to 4C$
Flatten
$\times 2 :$ Linear $(32//2^4 \times 4C) \to (32//2^4 \times 4C)//2$
$\times 2 :$ Linear $((32//2^4 \times 4C)//2) \to J$

Table 7. Encoder

D   Optimization problems

We observed two types of errors which restrict the architecture and the optimization techniques. Theses difficulties are particularly relevant for the CIFAR10 and CIFAR20 datasets which present the more complex distributions. We used the default parameters of the CIFAR10 architecture unless otherwise stated.

The first problem is associated with the batch size. We found that we can't optimize MPCC with a big batch size while using a large learning rate of the prior parameter η_p. Note that the latter is necessary to obtain good accuracy performance as it was shown in the paper (Table 2 in the paper). The batch size is relevant to increase the IS and FID scores [3]. Artifacts or saturation

problems would appear when doing a small modification in the optimization. The examples shown in Fig. 6 (a) use a batch size slightly larger than the one used in the paper (50). We observe that using a slightly larger batch size (64) with a prior learning rate of $\eta_p = 8 \cdot 10^4$ the results change drastically and the generated images show notable saturation.

Mode collapse is an important topic in GANs research and is the second problem that we observed in MPCC. Usually it is associated with the limitations in generation quality caused by the model, which memorize only a small part of the real distribution affecting the performance of the GAN. In MPCC the mode collapse problem can make an entire cluster collapse. Setting $D_{step} = 4$ solves this problem partially for a large amount of models and is sufficient to obtain good performance. In Fig. 6 (b) we show samples from a model trained with $bs = 64$ and $\eta_p = 2 \cdot 10^{-4}$ where we can see how a mode collapse problem looks in MPCC. We observed that when using a large prior learning rate $\eta_p = 6 \cdot 10^{-4}$ this problem would regularly appear after 150,000 iterations. This doesn't occur for the best configuration of MPCC (the one reported in the paper) and setting $\eta_p = 2 \cdot 10^{-4}$ even after a large number of iterations, however we would like to increase η_p further as we observed that it correlates with better clustering accuracy (Table 2 in the paper).
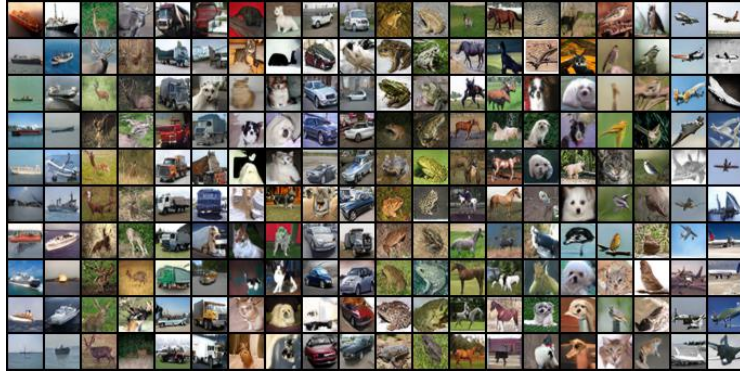

(a) Saturation problems   (b) Mode collapse problems

Fig. 6. Generated images with bad optimization setting at iteration 50000. Sub-figure (a) shows images associated with saturation problems and (b) with mode collapse problems. Each row represents a different cluster.

E   Additional qualitative results

In this section we provide additional reconstructions and samples for the CIFAR-10 dataset in Figures 7 and 8, and for the MNIST dataset in 9 and 10. To give more insight about MPCC's capacity we also include samples for datasets with a high number of classes, CIFAR-20 and Omniglot in Figures 11 and 12 respectively.

Fig. 7. Generated images for the CIFAR-10 dataset. Every two columns we set a different value for the categorical latent variable y. *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$.
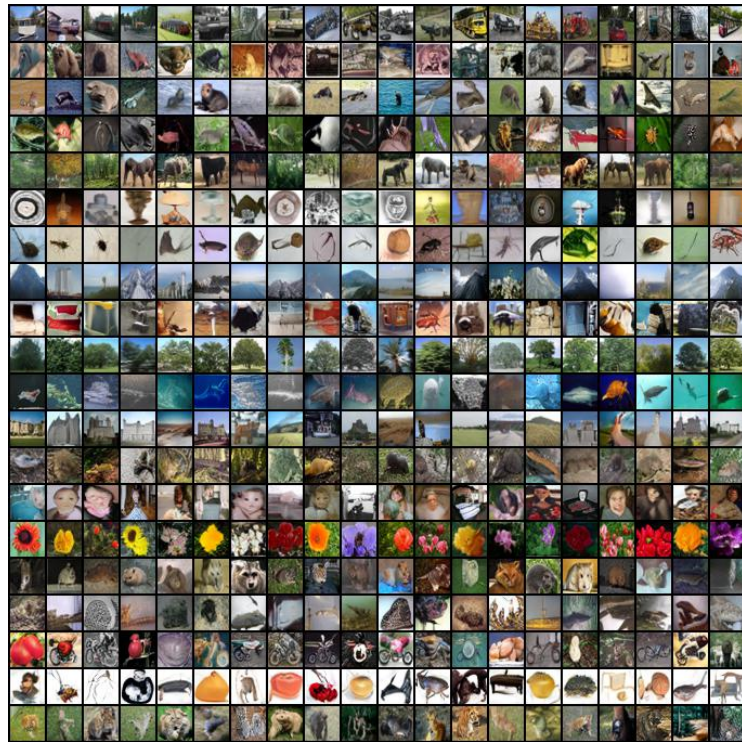

Fig. 8. Reconstructions for rhe CIFAR-10 dataset. Odd columns represent real data and even columns correspond to their reconstructions. The real label is used to sort the column pairs.

Fig. 9. Generated images for the MNIST dataset. Every two columns we set a different value for the categorical latent variable y. *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$.
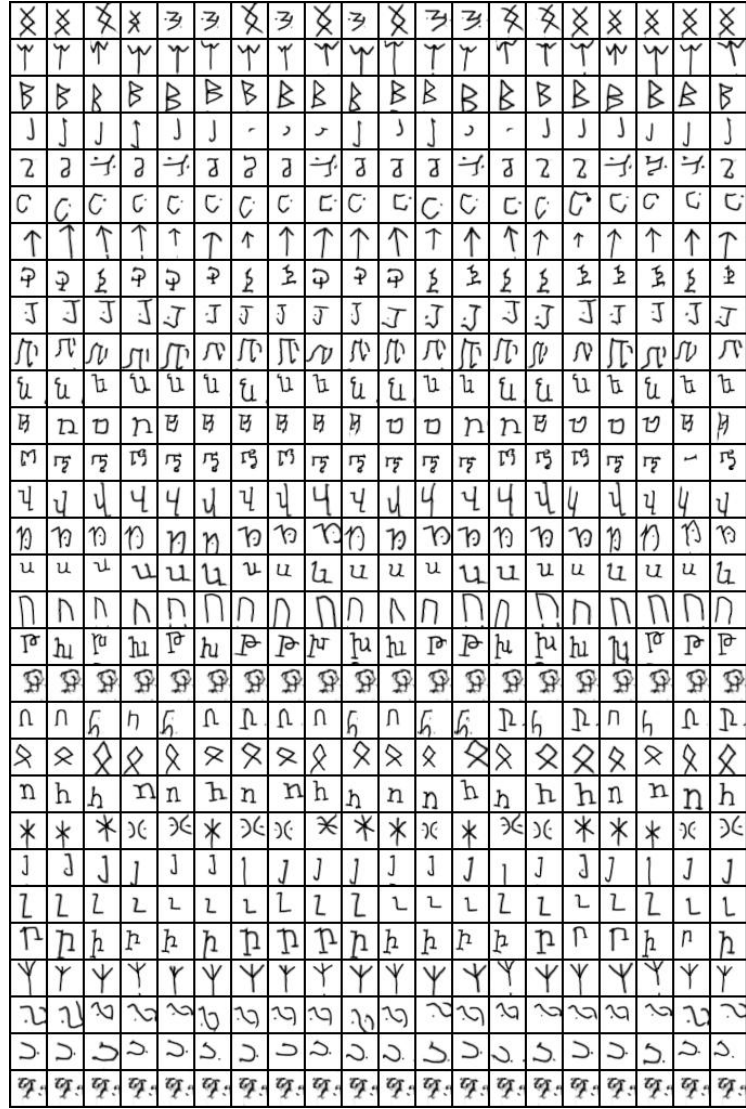

Fig. 10. Reconstructions for the MNIST dataset. Odd columns represent real data and even columns correspond to their reconstructions. The real label is used to sort the column pairs.

Fig. 11. Generated images for CIFAR-20 dataset. In every row we set a different value for the categorical latent variable y, *i.e.* the samples shown correspond to a different conditional latent space $z \sim p(z|y)$.

Fig. 12. Generated images for Omniglot dataset. In every row we set a different value for the categorical latent variable y, $i.e.$ the samples shown correspond to a different conditional latent space $z \sim p(z|y)$. 30 cluster were randomly chosen.