Clustering Meets Implicit Generative Models

Francesco Locatello\textsuperscript{1,3}, Damien Vincent\textsuperscript{2}, Ilya Tolstikhin\textsuperscript{1}, Gunnar Rätsch\textsuperscript{3}, Sylvain Gelly\textsuperscript{2}, and Bernhard Schölkopf\textsuperscript{1}

\textsuperscript{1}Max-Planck Institute for Intelligent Systems
\textsuperscript{2}Google Brain
\textsuperscript{3}ETH Zurich

Abstract

Clustering is a cornerstone of unsupervised learning which can be thought as disentangling the multiple generative mechanisms underlying the data. In this paper we introduce an algorithmic framework to train mixtures of implicit generative models which we instantiate for variational autoencoders. Relying on an additional set of discriminators, we propose a competitive procedure in which the models only need to approximate the portion of the data distribution from which they can produce realistic samples. As a byproduct, each model is simpler to train, and a clustering interpretation arises naturally from the partitioning of the training points among the models. We empirically show that our approach splits the training distribution in a reasonable way and increases the quality of the generated samples.

1 Introduction

In recent years, (implicit) generative models have attracted significant attention in machine learning. Generative models are trained from unlabelled data and are capable of generating samples which resemble the ones from the training distribution. This task is considered of crucial importance for unsupervised learning. Two of the most prominent approaches are Generative Adversarial Networks (GANs) \cite{goodfellow2014generative} and Variational Autoencoders (VAEs) \cite{kingma2013auto}. Both approaches aim at minimizing the discrepancy between the true data distribution and the one learned by the model. The model distribution is typically parametrized with a neural network which transforms random vectors into samples in the space of the training data (e.g., images). Variational Autencoders maximize a log-likelihood and are able to perform efficient approximate inference on probabilistic models with continuous latent variables and an intractable posterior. Furthermore, they come with an encoder network which maps data points to the latent space. Unfortunately, VAEs are known to produce blurry samples when applied to natural images. GANs take a completely different approach, relying on adversarial training. This yielded impressive empirical results. On the other hand, adversarial training comes at a cost. GANs are harder to train and suffer from the mode collapse problem. Indeed, if the data distribution lies outside the class of functions that the generator network can learn, the network will ignore portions of the data and focus on the parts which can be approximated well given its limited capacity. A number of approaches have been presented to tackle this problem. The most relevant to our setting is the work of \cite{dimou2017generative} who proposed to train a sequence of multiple generators which are subsequently mixed. As long as each generator collapses on a different mode, and given a large enough number of generators, one can approximate the whole data distribution by combining them. Several works followed up, trying to avoid the sequential training of the generators. In particular, \cite{dimou2017generative} proposed to train multiple generators in parallel using adversarial training, and a classifier to help them specialize to different modes. In contrast to GANs, VAEs are not trained with a minimax game and always model the whole support of the data distribution. Arguably, both extremes are undesirable. While approximating the whole distribution is the aim of generative density estimation, it should not come at the cost that the approximation is too poor to be useful.

We aim at bridging this gap by developing a general approach to train multiple generative models in parallel which focus on different parts of the training distribution. We instantiate this framework for VAEs. As a consequence, each
VAE will be able to collapse on some modes while the mixture of generators (decoders) will still approximate the whole data distribution.

Mixture of generative models are well motivated in both the literature on clustering and on causality. We assume that the data was generated by independent mechanisms, i.e., that the generative process of the overall distribution is composed of separate modules that do not inform nor influence each other [25]. We aim at modeling each of these mechanisms with an implicit generative model. Consider the special case of a variable $X$ which is caused by (mixing) several independent sources $g_1, \ldots, g_K$ without parents in the causal graph. Then, generative models take the form

$$p(X, g_1, \ldots, g_K) = p(X|g_1, \ldots, g_K) \prod_{j=1}^K p(g_j).$$

The terms on the right hand side are referred to as causal conditionals, Markov kernels, or mechanisms. Note that only one of the mechanisms, $p(X|g_1, \ldots, g_K)$, implementing the mixing, is a conditional; the others reduce to unconditional distributions since the sources have no parents in the causal graph. The conditional can be written as a structural equation [24]

$$X := f(g_1, \ldots, g_K, c) \equiv g_c,$$

where $c$ is a discrete noise variable taking values in $\{1, \ldots, K\}$. The distribution of $c$ determines the mixing coefficients. Eq. (2) expresses the conditional as a mechanism represented by a noisy function. It has been argued that the true causal factorization is the simplest among all the possible factorizations of the random variables, in the sense that the sum of the complexities of the causal conditionals is minimized [11]. Suppose each training point was generated by one of the mechanisms $g_j$, but we get to observe only the mixture of all these realizations. Recovering the mechanisms is then a hard and ill-posed inverse problem, since there are many ways to represent the same mixture in terms of the different components. Solving this problem amounts to learning a particular kind of structural causal generative model, and it could form a building block of more complex causal models [27]. We make the simplifying assumption that the supports of the different generative mechanisms do not overlap, hence if we observe two identical realizations of $X$ we assume they were generated by the same mechanism. While a soft assignment is also possible, in this paper we focus on clustering with hard assignments which is known to converge faster [14]. From a practical perspective, this implies dividing the data distribution in different components with non overlapping support. Each of these components can be approximated by an independent generative model. This way, we allow each VAE to learn only a fraction of the data distribution. As a consequence, they are able to model each part of the distribution better than a single model trying to approximate the whole data distribution. While the optimal split of the data distribution is unknown, we rely on a competitive procedure based on a set of discriminators to separate the data into the different mechanisms. Each training point will be won by the model which generates the most similar samples according to the set of discriminators. These discriminators can only influence the distribution of the training signal of the VAEs and are not used to propagate explicit gradients. Intuitively, a model that approximates the true causal mechanism will be easier to learn, and hence will generate better samples.

In this paper we design a training procedure to split the data distribution into components which are approximated well by independent generative models. Our contributions can be summarized as:

- Inspired by clustering algorithms, we provide a clear algorithmic framework for training mixtures of generative models. We target the most abstract case of minimizing a general $f$-divergence using a mixture of generators which can be trained in parallel.

- We instantiate our framework to mixtures of VAEs. By doing so, we allow them to collapse on separate modes of the distribution. As a consequence, the VAEs no longer need to cover all modes of the possibly complex overall distribution, and can thus use their limited capacity to produce realistic samples within the support of their portion of the data distribution.

- We present a clear clustering interpretation of our approach in non-metric spaces and recover k-means as a special case.
We provide empirical evidence that shows that our training procedure splits the training distribution into distinct components, significantly increasing the log-likelihood estimate for synthetic data and improving the FID score on MNIST and celebA.

2 Generative Mixtures: Problem Setting

Let $D_X$ be a dataset composed of $N$ samples $x_i$ from $X$. Furthermore, let $P_X$ be an unknown data distribution defined over the data space $X$ with support $X$ to be approximated with an easy to sample distribution $P_{model}$. Given a probability distribution $P$, we denote $dP$ its density. The goal of implicit generative density estimation is to make the samples from both distributions $P_X$ and $P_{model}$ look alike. This is typically formulated as some optimization problem minimizing the disagreement between the two. To measure such disagreement it is common practice to use an $f$-divergence:

$$D_f(Q\|P) := \int_{x \in X} f\left( \frac{dQ}{dP}(x) \right) dP(x)$$

(3)

where $f(1) = 0$ and $f$ is convex. The goal is then to solve the following optimization problem:

$$\min_{P_{model}} D_f(P_{model}\|P_X)$$

Unfortunately, $P_X$ is unknown and only an empirical estimate of $D_f$ is available through the samples in the training set $D_X$. While this setting is at the heart of the adversarial training of GANs [7], VAEs [16] minimize a variational bound on $D_{KL}(P_X\|P_{model})$ which is one of the various divergences which can be written in the form of (3) with the appropriate choice of $f$ [22]. From the perspective of causal generative modeling, we can think of the generative process as the entanglement of simple and independent components [11]. Therefore, to better approximate the data distribution one can consider $P_{model}$ as a mixture of distributions:

$$P_{model} = \sum_{j=1}^{k} \alpha_j P_{g_j}$$

(4)

each of them specialized on one of the generating mechanisms. Training mixtures of experts with boosting algorithms like the one in [29] has favorable optimization properties. In particular, adding components to a mixture is a convex optimization problem. On the other hand, in the context of deep learning, sequential training comes at a great cost in time. The sequential nature of boosting-like algorithms requires that each model is fully trained before the subsequent ones begin training, and an already trained model is never changed afterwards. Subsequent models are trained to fit the parts of space that the previous mixture could not approximate well. As a consequence, there is no incentive for any of the models to focus on a mode. Each generator will try to cover the whole residual of the data distribution which is not yet approximated. In contrast to [29], rather than training the mixture of generative models sequentially, we train each of them at the same time. By doing so, we lose the convexity of the objective; however, if one is able to decouple the training procedure, each model can be trained in parallel. Furthermore, instead of relying on mode collapse to happen, we let the models compete for training points in order to force them to learn different parts of the data distribution.

3 Training Independent Generative Models

Borrowing ideas from the clustering literature, at each iteration, each generative model $g_j$ is trained on a different portion of the dataset. We assign each training point $x_i$ to a single model using a set of $K$ binary partition functions $c_j$ implementing the realization of the mixing distribution $c(x_i)$ (i.e. $c_j(x_i) = 1$ and $c_{[K] \setminus j}(x_i) = 0$ when $c(x_i) = j$). Intuitively, our training procedure is related to the k-means algorithm. In k-means, one first decouples the training data across the centroids and then updates the centroids based on the assignment. Our approach is outlined in Algorithm[1]
Algorithm 1 Mixture training

1: **init** $K$ generative models $g_j, c_j^{(0)}, j = 1, \ldots, K$
2: **for** $t = 0 \ldots T$
3: $\min_{P_{g_j}} D_f \left( P_{g_j} \| P_{X_j}^{(t)} \right)$ for every $g_j$
4: Update $c_j^{(t+1)}(x)$ for every $g_j$
5: **end for**

Formally, let us consider $K$ assignment functions $c_j(x) : \mathcal{X} \rightarrow \{0, 1 \}$. We assume that for any $x \in \mathcal{X}$ there is a unique $j$ such that $c_j(x) = 1$ and it is zero for all the others $c_{[K]\setminus j}$. Let us now use the $c_j$ to partition the support $\mathcal{X}$ of $dP_X$. First of all, let us define:

$$dP_{X_j}(x) := \begin{cases} \frac{dP_X(x)}{\int_x dP_X(x)c_j(x)}, & \text{if } c_j(x) = 1 \\ 0, & \text{otherwise.} \end{cases}$$

In Algorithm 1 when the assignment function is some fixed $c_j^{(t)}$, we write $dP_{X}^{(t)}$ to make the dependency on the particular assignment explicit. For a given $c_j$, we define the mixing proportions $\alpha_j$ of $P_{model}$ as the normalization constant of $P_{X_j}$. This can be empirically estimated by counting how many training points are assigned to the $j$-th generator. We now show how to decouple the training of the generators by minimizing an upper bound of the $f$-divergence.

**Lemma 1.** For a fixed partition function $c_j^{(t)}$, we minimize for all $j \in [K]$:

$$\min_{P_{g_j}} \sum_j \alpha_j D_f (P_{g_j} \| P_{X_j}^{(t)}),$$

which is an upper bound on the $f$-divergence for a mixture model. We defer the proof to Appendix A.

Since each term in the sum in Equation (5) is independent, each generative model can be trained independently to approximate $P_{X_j}^{(t)}$.

After training the generative models, we fix them and update the assignment of each training point. Intuitively, our goal is that the mixture of $P_{g_j}$ resembles as much as possible $P_X$. Therefore, at iteration $t$, we first compute the likelihood $P_{g_j}^{(t)}(x_i)$ of each training point $x_i$. Then, we update the partition function $c_j^{(t+1)}(x_i)$ assigning each training point to the maximum likelihood model.

We estimate the likelihood by training a discriminator to distinguish samples from $P_{g_j}$ and samples from $P_X$. Let $D_{g_j}(x)$ be the output of the $j$-th discriminator. Then:

$$D_{g_j}^{(t)}(x) \approx \frac{dP_X(x)}{dP_X(x) + dP_{g_j}^{(t)}(x)}.$$  \hspace{1cm} (6)

After training the classifier, we can rewrite Equation (6) as:

$$dP_{g_j}^{(t)}(x) \approx dP_X(x) \frac{1 - D_{g_j}^{(t)}(x)}{D_{g_j}^{(t)}(x)}$$

Then, we can approximate $P_{g_j}^{(t)}(x_i)$ as the empirical estimate over the training set $\mathcal{D}_X$:

$$P^{(t)}_{g_j}(x_i) \approx \frac{1}{Z_j} \frac{1 - D_{g_j}^{(t)}(x_i)}{D_{g_j}^{(t)}(x_i)}$$
Discriminator

We now instantiate the framework using VAEs and discuss the benefits of being combined as a discrete mixture. While when used as generative models and tend to produce samples outside the support of the data distribution. These bridges

VAE decoder

when the individual components are simpler to train. We leave an extension to GANs as future work. In addition, we focus on just a part of the data distribution. Instead, VAEs try to cover the whole data distribution as a consequence of

Therefore, while each model will diligently try to cover the whole support of the assigned portion of the data distribution,

Discriminators can only implicitly influence the generators by acting on their training distribution. To sample from a

\[ x \sim P_{g_i}(x) \]

\[ x_i \sim P_X \]

\[ P_{g_i}(x) \equiv \frac{1 - D_{g_j}(x_i)}{D_{g_j}(x_i)} \]

\[ Q_j(z|x) \]

\[ D_{g_j}(x_i) \]

\[ Q_2(z|x) \]

\[ x \sim P_{g_2}(x) \]

\[ x \sim P_{g_1}(x) \]

\[ \mathcal{L} = \mathbb{E}_{z \sim P_Z} \left[ \mathbb{E}_{x \sim Q} \log P_g(x|z) - D_{KL}(Q(z|x) || P(z)) \right] . \]

The aim is to maximize this objective w.r.t. the parameters of the encoders and the generator (decoder). If

\[ P_g(x|z) = 0 \text{ for a point in the data distribution, the VAE incurs an infinite loss. Therefore, VAEs are overly “inclusive” }\]

when used as generative models and tend to produce samples outside the support of the data distribution. These bridges

4 kVAEs

We now instantiate the framework using VAEs and discuss the benefits of being combined as a discrete mixture. While using GANs is also possible, we wanted to first understand the properties and the stability of our training procedure when the individual components are simpler to train. We leave an extension to GANs as future work. In addition, we find that the behavior of VAEs is particularly interesting in our setting. First of all, VAEs do not exhibit mode-collapse. Therefore, while each model will diligently try to cover the whole support of the assigned portion of the data distribution, the different models will actively compete with each other trying to generate better and better samples. Note that while this approach is somehow inspired by GAN training, we do not directly receive gradients from the discriminators. The discriminators can only implicitly influence the generators by acting on their training distribution. To sample from a VAE decoder \( g_j \) as an implicit generative model we mean that we sample from

\[ \mathcal{L} = \mathbb{E}_{z \sim P_Z} \left[ \mathbb{E}_{x \sim Q} \log P_g(x|z) - D_{KL}(Q(z|x) || P(z)) \right] . \]

The aim is to maximize this objective w.r.t. the parameters of the encoders and the generator (decoder). If

\[ P_g(x|z) = 0 \text{ for a point in the data distribution, the VAE incurs an infinite loss. Therefore, VAEs are overly “inclusive” }\]

when used as generative models and tend to produce samples outside the support of the data distribution. These bridges
between different modes can be seen in Figure 2. While we do not claim that the only reason for blurriness are samples out of the support of the data distribution, we argue that it is unlikely that these samples are realistic. Furthermore, if one wants to generate approximate samples from the data distribution, there should be an incentive for the support of the model to be as close as possible to $X$. In particular, if $X$ is disjoint, $P_{\text{model}}$ should also have disjoint support.

5 Clustering Interpretation

The general approach we introduced is closely related to k-means clustering. In this section we revisit classical clustering notions in view of our framework. We show that we generalize k-means in non-metric spaces, and we recover it when the space is euclidean.

In the generative interpretation of clustering, one assumes that the data was generated from each centroid $\mu$ with an additive Gaussian noise vector, i.e., $x = \mu + \epsilon$. This formulation naturally yields an euclidean cost for the cluster assignment when decoupling the data between the different centroids. Unfortunately, the euclidean distance is known not to be a good metric for images or many other entities of interest. Our goal in the present paper is to find a clustering of the data across the generating mechanism in a setting in which the metric of the space is not known. We now show that k-means clustering is a special case of our framework. Assume that the data is generated by a mixture of Gaussians.

We can lower bound the log-likelihood of the data using a variational bound:

$$\log(P(D_X)) \geq \sum_{i=1}^{N} \sum_{j=1}^{K} q_i(j) \log \left( \frac{P(x_i,j)}{q_i(j)} \right)$$

where $q$ is the variational approximation of the posterior and $j = 1, \ldots, K$ is the index of the components. One can then simply rewrite $P(x_i,j) = P(x_i|j)p(j)$. Then, for a Gaussian mixture model one parametrizes $P(x_i|j)$ with a Gaussian distribution.

If the Gaussian is isotropic with vanishing covariance, the variational approximation of the posterior $q_i(j)$ degenerates to a hard assignment. Instead of approximating the generative model with a Gaussian distribution, we parametrize $P(x_i|j)$ with an implicit generative model from which it is easy to sample, i.e. the decoder of a VAE marginalized over the prior. Note that VAEs are trained to maximize the log-likelihood as in EM. Assume we have a Gaussian encoder which maps all the input to a single point (degenerate Gaussian with $\sigma = 0$ and constant mean $\mu$ independent from the input $x$). Now, say we have the identity as decoder. Then, training the autoencoder amounts to minimizing:

$$\min_{\mu_j} \mathbb{E}_{x \sim P_X} \left[ -\log P_{g_j}(x|\mu_j) \right] = \min_{\mu_j} \mathbb{E}_{x \sim P_X} \left[ \frac{1}{2} \|x - \mu_j\|^2 \right].$$

Then, using EM, we compute the update for the (degenerate) variational distribution:

$$q_i(j = 1) = \lim_{\sigma \rightarrow 0} \frac{\alpha e^{-\|x - \mu_j\|^2/2\sigma}}{\sum_j \alpha_j e^{-\|x - \mu_j\|^2/2\sigma}}$$

And recalling that $\log P_{g_j} = -\|x_i - \mu_j\|^2/2$ we notice that the degenerate posterior is obtained by maximizing the likelihood. In our approach, we estimate $P_{g_j}$ using a discriminator to account for the fact that we might not have a clear notion of distance. In an euclidean space, one can simply use a nearest neighbor classifier between the output of the VAEs (i.e., the centroids) and the training points. Note that this procedure is exactly k-means.

6 Related Work

The main influences for our work are the literature on clustering, mixture of experts, causality, and implicit generative models.

Clustering is a cornerstone of unsupervised learning and the literature is vast, see [1] for a recent overview. [18] focused on the inference in structural equation models where the target variable is hidden and accessible only through indicator variables. In order to make the problem tractable they look for indicators that only depend on the latent
variable and are independent from any other variable in the causal graph transforming the structure of the learning problem to a clustering problem. Clustering for causal discovery was already explored in [21]. In the opposite direction, [26] looked at clustering through the lens of causal inference and proposed a semi-distance based on algorithmic complexity estimates.

Our framework is related to competition of experts, which can be used to invert independent causal mechanisms, see [23]. Mixtures of experts with a gating network trained with EM are a classical idea which was introduced by [13]. A more recent example applied to lifelong learning is the one in [2]. [17] proposed to use a GAN to learn a generative model with true observational and interventional distributions for a given causal graph while [19] proposed to couple several GANs to learn a joint distribution of images from different domains.

Allowing for discrete mixtures in implicit generative model is a new trend in the community which have been explored in the setting of both VAEs [5, 12] and GANs [8]. Note that these authors consider mixtures in the latent space rather than mixtures of independent generative models. A clustering in the latent space of VAEs was recently proposed by [30]. The work of [29] first introduced the idea of training multiple implicit generative models, in order to address the mode collapse problem of GANs. [6] proposed to train multiple generators in parallel with a single discriminator. [10] proposed to adversarially train a system with multiple generators and a classifier which encourages the models to split using adversarial learning. In contrast to our approach, they use the classifier entropy computed on synthetic examples as a regularizer for the objective function. This requires a delicate tuning parameter which controls the ability of the networks to split. In our setting the splitting is implicit in the training procedure. Furthermore, as the classifier is trained only on synthetic data, it is not a reliable assignment function for clustering, as real data points could be in a part of the space that the generators do not cover. Training mixtures of GANs with EM was introduced by [4]. They target soft assignments using a kernel to measure the similarity between the true data points and the generated ones. In contrast to our work, they do not learn the mixing components (they are assumed uniform), and they use a kernel in the pixel space rather than a discriminator. [3] presents a technique to achieve pure equilibrium with GANs using multiple generators and discriminators which are selected through a multi-way selector implemented with a neural network. Their approach is specific to GANs and they use a single set of discriminators which provides explicit training signal for the generators. This is very different from both the boosting and the clustering rationale as there is not a notion of assignment for the training points. Therefore, they can not address the separation of the generative mechanisms in terms of training samples and can not obtain the corresponding clustering notion.

7 Experiments

In this section we evaluate the proposed framework. We test if the model can successfully learn a set of generative mechanisms for the data. Furthermore, we prove that our split of the training set indeed helps the generative models producing better samples and is indeed a scalable approach to effectively increase the capacity of the model. We want to remark that to test whether our method could be used as a tool to improve the performances of generic VAEs, we used standard architectures with standard hyperparameters in all the experiments. All models are initialized by training for a few epochs on the whole training set. Details of the architectures and hyperparameters along with additional visual comparisons can be found in the appendix. We will release the code after publication.

7.1 Synthetic Data

| Modes | kVAEs | bag | VAE-150 |
|-------|-------|-----|---------|
| 3     | -4.59 | -6.49 | -5.42 |
| 5     | -2.74 | -7.7 | -5.71 |
| 9     | -2.51 | -7.05 | -6.83 |

We generate synthetic data in two dimensions by first sampling 64,000 points from a mixture of Gaussian distributions and then we skew the second dimension \( x_2 \) with the non-linear transformation

\[
 x_2 = x_2 + 0.04 \cdot x_1^2 - 100 \cdot 0.04
\]

We use a small and standard architecture for the VAE: a neural network with two hidden layers with 50 units each as both the decoder and the encoder. The discriminator has a similar architecture. We keep the size of the dataset and the architecture fixed and progressively increase the difficulty of the task by increasing the number of modes. In the first two experiments, each model perfectly covers a single mode.
Figure 3: Synthetic data experiment with different number of modes. Our models split the data distribution and only learn simple components.

Figure 4: MNIST: samples (above) separated per model (mixture component) and real digits clustered after just 10 iterations of Algorithm 1 (below).

Figure 5: celebA: generated samples separated per component after 26 iterations of Algorithm 1.

seven when a mode is significantly different than the others. With nine modes, some models are still trying to approximate multiple modes, yielding samples outside the support of the data distribution at the end of the training procedure. On the other hand, a single VAE completely fails in this task (see Appendix B). Finally, we compute the log-likelihood of the true data under the generated distribution using a Kernel Density Estimation with Gaussian kernel. We compare against a larger VAE with 150 (VAE-150) units per layer (instead of 50), trained uniformly over the training set, and a bag of VAEs with 50 units trained on a random subsample of the training set containing \( N/K \) training points. We note that the random splitting of the training set did not help the VAE to specialize and actually made the log-likelihood worse after training for 100 epochs. Overall, our approach gives the best log-likelihood in this experiment as depicted in Table 1.

7.2 MNIST and CelebA

Table 2: FID score on MNIST.

| kVAEs | bag | VAE-8 | VAE-64 |
|-------|-----|-------|-------|
| 9.99  | 15.33 | 17.96 | 9.44  |

For the experiment on MNIST, we do not know the number of modes. There is no reason to believe the optimal number of modes should be the number of digits. We arbitrarily use 15 models to capture the different strokes of the digits, following the insights from [29]. We again use a small and simple architecture; the encoders and the decoders have 4 and 3 convolutional layers with 8-16-32-64 and 32-16-8 \( 4 \times 4 \) filters respectively. In Figure 4 we show the clustering of MNIST on which each VAE is trained and the corresponding generated digits.
Note that the different VAEs did specialize on distinct parts of the data distribution: similar digits tend to be grouped together (for example 3 and 8, 6 and 4, but also 5, 9 and 3 can be very similar depending on the style), as well as similar styles (tilted, thin, large, round, bold and combinations thereof). To evaluate our generated samples we used the FID score \[ 9 \]. We remark that our FID score is competitive with the one obtained with a large VAE (VAE-64, 64-128-256-512 and 256-128-64 filters), as well as the one that can be obtained with GANs (slightly less than 10 was reported in Figure 5 of \[ 20 \]). It also is significantly better than both bagging and a single model with the same capacity.

Finally, we test our algorithm on celebA using 5 components. We use the same architecture we used for MNIST but with 64-128-256-512 filters for the encoder, 256-128-64 for the decoder and 128 dimensional latent space. We note that the generated samples are visually clustered in Figure 5. We note that the background and the hair color plays a significant role in the clustering (see for example cluster 3 and 5). Again, our procedure improves the FID score as depicted in Table 3 which are competitive with the ones reported in Figure 5 of \[ 20 \] and better than both bagging (same architecture as ours) and a single VAE with 96-192-384-768 and 384-192-96 filters for encoder and decoder respectively with latent space with 64 dimensions trained for 60 epochs.

### Table 3: FID score on celebA.

| kVAEs | bag    | VAE-96 |
|-------|--------|--------|
| 64.55 | 71.90  | 67.06  |

### 8 Conclusions

In this paper, we introduced a clustering procedure using implicit generative models, which encourages them to generate more realistic samples. Our approach is inspired by the fact that the underlying causal generative model is simpler when decomposed into separate modules. Therefore, we train networks competing with each other in the pursuit of generating more realistic samples. We enforce the competition between the models by relying on a set of discriminators that judge the quality of the samples produced by each model. We demonstrated how to decouple the loss of the combined model into parts that can be trained independently and showed that our approach is a generalization of the classical k-means clustering. We empirically validated that the model can successfully recover the true generative mechanisms. Even when the recovery is not perfect, the competitive procedure allows to generate samples which are closer to the support of the data distribution. The approach we presented is extremely modular and there are several possible extensions. First, given enough computational resources, one can dramatically increase the number of generative models. Furthermore, it can be extended by either a better latent manifold structure, for example using Wasserstein autoencoders \[ 28 \], or replacing the VAEs with GANs. Finally, adaptively selecting the number of components does not have a trivial solution. Exploiting the vast clustering literature could shed some light on how to perform model selection on the fly, but we leave this problem as future work.
References

[1] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications*. CRC press, 2013.

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. *CoRR, abs/1611.06194*, 2, 2016.

[3] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573*, 2017.

[4] Ershad Banijamali, Ali Ghodsi, and Pascal Popuart. Generative mixture of networks. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 3753–3760. IEEE, 2017.

[5] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

[6] Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. *arXiv preprint arXiv:1704.02906*, 2017.

[7] 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*, pages 2672–2680, 2014.

[8] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, and V Babu Radhakrishnan. Deligan: Generative adversarial networks for diverse and limited data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2017.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6629–6640, 2017.

[10] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. Multi-generator generative adversarial nets. *arXiv preprint arXiv:1708.02556*, 2017.

[11] Dominik Janzing and Bernhard Scholkopf. Causal inference using the algorithmic markov condition. *IEEE Transactions on Information Theory, 56*(10):5168–5194, 2010.

[12] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[13] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation, 6*(2):181–214, 1994.

[14] Michael Kearns, Yishay Mansour, and Andrew Y Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Learning in graphical models*, pages 495–520. Springer, 1998.

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

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] Murat Kocaoglu, Christopher Snyder, Alexandros G Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. *arXiv preprint arXiv:1709.02023*, 2017.

[18] Erich Kummerfeld and Joseph Ramsey. Causal clustering for 1-factor measurement models. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1655–1664. ACM, 2016.
[19] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In Advances in neural information processing systems, pages 469–477, 2016.

[20] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. arXiv preprint arXiv:1711.10337, 2017.

[21] Andrzej Matuszewski. Double clustering: A data mining methodology for discovery of causality. In Intelligent Information Systems 2002, pages 227–236. Springer, 2002.

[22] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In Advances in Neural Information Processing Systems, pages 271–279, 2016.

[23] Giambattista Parascandolo, Mateo Rojas-Carulla, Niki Kilbertus, and Bernhard Schölkopf. Learning independent causal mechanisms. arXiv preprint arXiv:1712.00961, 2017.

[24] Judea Pearl. Causality. Cambridge University Press, 2000.

[25] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Elements of Causal Inference - Foundations and Learning Algorithms. Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, MA, USA, 2017.

[26] Marion Revolle, François Cayre, and Nicolas Le Bihan. Clustering and causality inference using algorithmic complexity. In Signal Processing Conference (EUSIPCO), 2017 25th European, pages 843–847. IEEE, 2017.

[27] Bernhard Schölkopf, Dominik Janzing, and David Lopez-Paz. Causal and statistical learning. In Oberwolfach Reports, volume 13, pages 1896–1899, 2016.

[28] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. arXiv preprint arXiv: 1711.01558, 2017.

[29] Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. AdaGAN - Boosting Generative Models. NIPS, 2017.

[30] Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In Advances in Neural Information Processing Systems, pages 6309–6318, 2017.
A Proof of Lemma 1

Trivially, all of the below hold from the definition of $c_j$:

- $\int_{X_j} dP_{X_j}(x) = 1$ where $X_j$ is the support of $dP_{X_j}$
- $\int_{X} dP_{X_j}(x) = 1$ as $dP_{X_j}(x)$ is zero outside its support
- $\cup_{j \in [K]} X_j = X$ and $X_j \cap X^l = \emptyset \forall l \neq j$

By definition of the model we write the $f$-divergence as:

$$D_f(P_{model} \parallel P_X) = D_f(\sum_{j=1}^{k} \alpha_j P_{g_j} \parallel P_X)$$

Now, we have that $\alpha_j = \int_{X} dP_{X_j}(x)c_j^{(t)}(x)$. Since $X_j \cap X^k = \emptyset$ for $j \neq k$, we can write:

$$D_f(\sum_{j=1}^{k} \alpha_j P_{g_j} \parallel P_X) = D_f(\sum_{j=1}^{k} \alpha_j P_{g_j} \parallel \sum_{j=1}^{k} \alpha_j P_{X_j}^{(t)})$$

Joint convexity of $D_f$ concludes the proof [22].

B Synthetic Data: Additional Results

We use a small and standard architecture for the VAE: a neural network with two hidden layers with 50 units each as both the decoder and encoder. The discriminator has a similar architecture. We use a 5 dimensional latent space and assume a Gaussian encoder. At each iteration, we train each VAE for 10 epochs on a split of the dataset (VAEs are pretrained uniformly on the dataset), and the classifier is trained for 2 epochs for the first two experiments. We use Adam [15] with step size 0.005, $\beta = 0.5$ batch size 32. The first two experiments contains 3 and 5 separate modes respectively and are depicted in Figure 6 and 7. In the latter, we have one mode which is more complex than the others. In both cases, each model perfectly covers a single mode. This experiment illustrates that VAEs can learn how to generate samples in the support of the data, provided that are only asked to capture a sufficiently simple distribution.

In the next experiment, we consider 9 modes, each containing significantly fewer points than the previous ones. We first train each VAE for 1000 epochs because this task is significantly harder. We note that even after long training, the models perform poorly, as the data is too complex for such simple models. Then, we run our algorithm on the pre-trained models and each model is re-trained for 10 epochs on the split given by the classifiers. Remarkably, after only 10 iterations of Algorithm 1 the generators split to cover only limited parts of the data distribution, as one can see in Figure 8.
C MNIST: Additional Results

We again use a small and simple architecture. The encoders and the decoders have 4 and 3 convolutional layers with 8-16-32-64 and 64-32-16 $4 \times 4$ filters respectively. We use batch normalization with $\epsilon = 10^{-5}$ and decay 0.9. Each VAE has latent space dimension 8, and we fix the learning rate of Adam for all networks to 0.005. The discriminator has 3 convolutional layers and a linear layer with number of filters 64-128-256. As opposed to the synthetic data example, we do not reinitialize the classifier at each iteration of the meta algorithm. The reason is that we found the classifier output to be too sensitive to the initialization if it is not trained sufficiently long. On the other hand, training a full discriminator in every iteration was too expensive, and if trained too much, it would learn to distinguish fake examples by just looking at specific bluriness patterns. In the synthetic experiments, the data produced by each VAE was indistinguishable from the real data if the support was correct, so training a classifier from scratch was feasible and gave best results.

In Figure 10 we show the number of training samples assigned to each model divided by digit for the kVAE algorithm with 15 VAEs. In Figure 9 we show samples from our model trained for 10 iterations of Algorithm 1. In Figure 11 we show samples from the bagging mixture of 15 VAEs trained for 100 epochs. We use the same architecture as the one used for the kVAEs Algorithm. In Figure 12 and 13 we show samples from a single small and large VAE. We notice that our model produces a large variety of different styles which are visually clustered.
Figure 9: Samples from the mixtures of kVAE with 15 components trained for 10 iterations of Algorithm 1. We notice a large variety of different strokes and styles.

Figure 10: Clustering of MNIST using the discriminators.

Figure 11: Samples from the bagging of 15 VAEs with the same architecture as ours trained on random splits of the training data.
Figure 12: Samples from a single VAE with the same architecture as ours

Figure 13: Samples from a single VAE with 64-128-256-512 and 256-128-64 filters per layer
D CelebA: Additional Results

The encoders and the decoders have 4 and 3 convolutional layers with 64-128-256-512 and 256-128-64 $5 \times 5$ filters respectively. We use batch normalization with $\epsilon = 10^{-5}$ and decay 0.9. Each VAE has latent space dimension 128, and we fix the learning rate of Adam to 0.0002. The discriminator has 3 layers with 128 $3 \times 3$ filters and is trained with SGD with stepsize $10^{-4}$. We perform an assignment after every 10 epochs of training. In Figure 14 we show the samples from 5 models trained with bagging using the same architecture for each VAE and in Figure 15 the samples from our model. In figure 16 we show samples from a VAE with 96-192-384-768 and 384-192-96 filters for encoder and decoder respectively with latent space with 64 dimensions trained for 60 epochs. We notice that our model produces more visually appealing and diverse samples than both bagging and a single larger model.
Figure 15: samples from 5 VAEs trained with our competitive procedure

Figure 16: samples from a single larger VAE