2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

ON THE EFFECTIVENESS OF TWO-STEP LEARNING FOR LATENT-VARIABLE MODELS

Cem Subakan[♭]*, Maxime Gasse*[♭]*, Laurent Charlin*[♭,♯]

[♭]Mila — Quebec Artificial Intelligence Institute, [♯]HEC Montréal

ABSTRACT

Latent-variable generative models offer a principled solution for modeling and sampling from complex probability distributions. Implementing a joint training objective with a complex prior, however, can be a tedious task, as one is typically required to derive and code a specific cost function for each new type of prior distribution. In this work, we propose a general framework for learning latent variable generative models in a two-step fashion. In the first step of the framework, we train an autoencoder, and in the second step we fit a prior model on the resulting latent distribution. This two-step approach offers a convenient alternative to joint training, as it allows for a straightforward combination of existing models without the hustle of deriving new cost functions, and the need for coding the joint training objectives. Through a set of experiments, we demonstrate that two-step learning results in performances similar to joint training, and in some cases even results in more accurate modeling.

1. INTRODUCTION

Latent-variable generative models typically consist of a stochastic prior model $q_p(h)$, i.e., a latent distribution, and a (stochastic) decoder model $q_d(x \mid h)$. While simple priors are commonly used in practice, such as a standard normal in the variational autoencoder (VAE) framework [1] or the generative adversarial network (GAN) framework [2], recent works suggest that learning (the parameters of) complex, multi-modal priors better captures the structure of the latent space and thereby yields more accurate generative models [3, 4]. For example, using a learned Gaussian mixture model (GMM) prior improves the performance of VAEs [5, 6].

Jointly training the decoder and prior parameters, however, can be a tedious task when complex prior distributions are used. In the VAE framework, one usually has to derive a closed-form expression of the evidence lower bound (ELBO) for each new type of prior modeling choice, which prevents the use of arbitrary distributions and requires lengthy derivations. For example, deriving a joint training objective for the GMM prior, including the description of their various implementation choices, takes [5] a considerable chunk of their paper. We provide details in Section 4.1.

We propose *two-step learning*, an alternative approach to joint training for latent-variable models. We show that it is unbiased, simple to implement, and performs well empirically.

The framework consists in decoupling the training into two steps: 1) fitting an autoencoder model to the data, 2) fitting a prior distribution on the resulting latent representation. This decoupling enables faster and more robust (less bug prone) exploration of modeling options since different priors and autoencoders can be plugged in and existing learning procedures reused. For example, in Section 4.2 we show how a VAE with a GMM prior can be turned into a generative model for time series, simply by changing the prior to a Hidden Markov Model (HMM).

We demonstrate across several tasks that two-step learning results in performances similar to joint training (when joint training is available), while being substantially easier to implement in each case. In addition, our experimental results suggest that two-step learning is more stable than joint training, especially in the presence of discrete latent variables. In particular, we showcase the versatility and performance of our method in different situations, and explore a broad range of models including VAEs with GMM prior, VAEs with HMM prior, and some models where it is not obvious how to carry out joint training, such as VAEs with GANs priors, GANs with GMM priors, and PCA autoencoders with GMM priors.

Our contributions are:

1. We define a two-step learning framework for learning latent variable-based generative models, and show it is consistent with the VAE framework.
2. We demonstrate on several use cases that the performance of two-step learning is comparable to that of joint training, when available, while being more robust.
3. We demonstrate the versatility of the approach, with a broad range of latent variable models mixing heterogeneous autoencoder and prior models.

2. THE TWO-STEP PROCEDURE

Consider data $x \in \mathbb{R}^L$, and let $p(x)$ denote the unknown data distribution we want to estimate. A latent-variable generative model consists of a prior distribution $q_p(h)$ and a decoder $q_d(x \mid h)$, with latent codes $h \in \mathbb{R}^K$. Learning such a model amounts to finding a combination of q_p and q_d so that the marginal distribution,

$$q_{\text{model}}(x) := \sum_h q_p(h) q_d(x \mid h),$$

approximates the data distribution $p(x)$, ideally so that the Kullback-Leibler divergence $\mathrm{KL}(p(x) \parallel q_{\text{model}}(x))$ is close to zero. Since the direct optimization of that marginal distribution is in general intractable, the typical approach is to maximize the evidence lower bound (ELBO), or equivalently minimize

$$\begin{aligned} &\mathrm{KL}(p(x)q_e(h \mid x) \parallel q_e(h)q_d(x \mid h)) && \text{(1a)}\\ &\quad + \mathrm{KL}(q_e(h) \parallel q_p(h)) && \text{(1b)}\\ &\quad \geq \mathrm{KL}(p(x) \parallel q_{\text{model}}(x)), \end{aligned}$$

with $q_e(h \mid x)$ an encoder that maps observations x into latent variables h, and $q_e(h)$ the resulting implicit prior distribution over the latent variables, obtained after marginalizing the data distribution,

$$q_e(h) := \sum_x p(x) q_e(h \mid x).$$

978-1-7281-6662-9/20/$31.00 ©2020 IEEE

As introduced in Section 1, deriving the joint training loss with complex prior distributions can be cumbersome, as it requires analytically deriving the corresponding ELBO (or making an approximation for it) for each new type of prior modeling choice. We show a full example in Section 4.1.

The insight we exploit in this paper is that q_e and q_d together form a proper autoencoder (AE). In that view, the first term (1a) pushes the resulting AE to minimize the reconstruction error, while the second term (1b) pushes the implicit distribution of the encoder output and the prior model to match each other.

With this in mind, we propose to decouple the learning phase into two steps: 1) first learn an autoencoder model (q_e and q_d), and then 2) fit a prior model (q_p) on the resulting implicit prior. The resulting process, summarized in Algorithm 1, can be shown to recover the true data distribution under common assumptions, and is therefore unbiased. We showcase the difference in implementation of both approaches in Section 4.

Proposition 1 (Two-step correctness)**.** *Assuming statistical sufficiency (infinite data), model sufficiency (infinite capacity for q_e, q_d, q_p), and exact minimization in each step, the two-step algorithm recovers $q_{model}(x) = p(x)$.*

Proof. The proof is straightforward. Assuming unlimited data, infinite capacity, and exact minimization, both losses (1a) and (1b) reach zero. Then the sum of the terms reaches zero as well, which upper bounds $\mathrm{KL}(p(x)\|q_{\text{model}}(x))$. □

Algorithm 1 Two-step learning

Step 1: Learn the autoencoder model, $q_e(h \mid x)$ and $q_d(x \mid h)$, by minimizing (Eq. 1a).
Step 2: Learn the prior model $q_p(h)$ on the embeddings obtained from encoder, by minimizing (Eq. 1b).

Two-step training is an appealing alternative to joint training. As hinted earlier, jointly maximizing the ELBO with complex priors requires tedious derivations and implementation efforts, whereas two-step training simply amounts to wiring existing software routines for fitting the autoencoder and prior model. Another advantage that comes with using two-step learning is the flexibility of the learning pipeline, which enables straightforward modifications to the prior distribution, when the codes for learning the distributions are readily available. In the subsequent parts of the paper, we show several examples of using different distributions over the latent variables with readily available codes available in standard software packages.

Available AE options include popular models such as deterministic autoencoders, AliGAN/BiGAN [7, 8], adversarial autoencoders [9], adversarially regularized autoencoders [10], or the GLO model which does not explicitly learn an encoder [11]. As for the prior model, two-step learning can readily employ complex models such as GMMs [12], GANs [2] or VAEs for i.i.d. data, as well as HMMs [13], recurrent neural networks [14], or temporal convolutional networks such as WaveNet [15] for non-i.i.d. data.

3. RELATED WORK

Two-step learning is a relatively natural idea, and several related approaches can be found in the literature. A very recent and popular instantiation of two-step learning is the Vector-Quantized VAE [3]. The authors propose to first learn a VAE while performing vector quantization in the latent space, and in a second step to fit an autoregressive model on the resulting prior distribution, such as pixel-CNN [16], or wavenet [15]. In a recent version of this work, the approach has been shown to generate large images with great fidelity [17].

Other examples of works that employ two-step training include [4], where the authors train a GAN in the latent space of a VAE in order to generate images and sounds with latent constraints. In the NLP domain, [18] fit a GAN on the latent space of a general-purpose sentence encoder, in order to generate natural-looking sentences. In [19], the authors fit a GMM in the latent space of an autoencoder, in order to model 3D point clouds.

There exist also others works that do not leverage modern deep learning, and yet showcase the statistical advantages of two-step learning. For example, decoupled HMM learning, first emission parameters and then transition parameters, is scalable and is robust to perturbations [20, 21].

While those works constitute specific instantiations of two-step learning tailored to their context, we frame two-step learning as a general framework. We establish formal connections with ELBO minimization (Sec. 2), and experimentally demonstrate that its performance is similar to that of joint training, while being significantly simpler in terms of software development (Sec. 4).

While joint training is standard for deep-learning models, the expectation-maximization (EM) algorithm is a classical example of an (iterative) two-step procedure [22] which is widely used for probabilistic inference. In EM, the latent variables and parameters are optimized in an alternating fashion. Whereas in our proposed two-step learning, it is the parameters that are split in two, and each subset is optimized until convergence. (For instance in step 1, we train the auto-encoder until convergence, and in step 2, the prior model is trained until convergence by using a fixed auto-encoder.)

4. TWO-STEP VS JOINT TRAINING

We now compare and contrast the two-step learning framework to joint training using: 1) VAEs with GMM prior, and 2) VAEs with HMM prior. The goal is to showcase the difference in coding requirements, and compare the accuracy of the learnt distributions.

4.1. VAE with GMM prior

In the vanilla VAE formulation [1] the latent codes are assumed to follow a standard Gaussian distribution. This type of prior is known to be too constraining, and the need for a multi-modal prior in VAEs has been argued for in several research papers [23, 5, 6].

One natural candidate for a multi-modal prior is the Gaussian Mixture Model. The generative process of a latent variable model with a GMM prior is:

$$c \sim \ \mathrm{Cat}(\pi),\ h \mid c \sim \mathcal{N}(\mu_c, \Sigma_c^2),\ x \mid h \sim \ q_d(x \mid h),$$

where $c \in \{1, \ldots, C\}$ is a discrete variable that chooses the Gaussian component to use for the latent variable $h \in \mathbb{R}^K$.

There have been multiple attempts at using a GMM prior over latent variables in a VAE framework [5, 6, 24], usually by maximizing the ELBO as a joint training objective. A successful implementation is given in [5], where the derivation of the joint objective takes a significant part of the paper. After a lengthy derivation, and conditioned on several architectural choices such as diagonal Gaussian components ($\Sigma_c^2 = \sigma_c^2 I$), a Gaussian encoder ($q_e(h \mid x) = \mathcal{N}(h; \mu(x), \sigma^2(x) I)$) and a factorized approximate posterior

$(q_e(h, c \mid x) = q_e(h \mid x)q_e(c \mid x))$, [5] arrive at the following expression:

$$\begin{aligned}\text{ELBO}_{\text{VAEGMM}}(x) &= \mathbb{E}_{q_e(h|x)}[\log q_d(x \mid h)] \\ &- \frac{1}{2}\sum_{c=1}^{C} q_e(c|x) \sum_j \left(\log \sigma^2_{c,j} + \frac{\sigma^2(x)_j}{\sigma^2_{c,j}} - \frac{(\mu(x)_j - \mu_{c,j}))^2}{\sigma^2_{c,j}} \right) \\ &+ \sum_{c=1}^{C} q_e(c|x) \log \frac{\pi_c}{q_e(c|x)} + \frac{1}{2}\sum_j (1 + \log \sigma^2(x)_j). \end{aligned} \tag{2}$$

Finally, the posterior over the cluster indicators is estimated using $q_e(c \mid x) = \frac{\pi_c p(h|c)}{\sum_{c^0} \pi_{c^0} p(h|c^0)}$, where $p(h \mid c) = \mathcal{N}(h; \mu_c, \sigma_c^2 I)$ is the c'th component of the GMM on the latent codes. Note that this way of implementing the posterior is a design choice of the authors.

Overall, the ELBO function proposed by [5] requires a careful and potentially tedious implementation. An alternative way of going about learning a VAE with a GMM prior is to apply the two-step learning paradigm we propose in this paper. This is tantamount to simply re-using the software that is readily available, as we showcase in Figure 1. We start by fitting a regular VAE. We then sample latent codes by passing data samples through the encoder part, and finally learn a GMM on the resulting latent dataset. Note that in this particular implementation, we utilize the popular *scikit-learn* library [25] to learn the parameters of the GMM.

```
# train the VAE
model.VAE_trainer(train_loader, EP=EP)

# get the embeddings
all_hhats = get_embeddings(model,
    train_loader)

# fit the GMM
GMM = mix.GaussianMixture(n_components=K,
    covariance_type='diag')
GMM.fit(all_hhats.data.cpu().numpy())
```

Fig. 1. Two-step training of a VAE with a GMM prior.

As shown in Fig. 1, the 2-step learning paradigm results in easy-to-implement and easy-to-understand code. The advantage over joint optimization is not limited to convenience in software development. In latent-variable models that contain discrete variables such as GMMs, the optimization is generally tricky, as the relaxation to discrete variables likely causes the optimization procedures to get stuck in local optima [26]. This is frequent when the dataset exhibits a multi-modal structure, as we show in the next experiment.

4.1.1. Experimental Results

The purpose of this experiment is to showcase the effectiveness of 2-step learning in terms of optimization, on a multimodal dataset. To do so, we generate a simple two-dimensional dataset from a mixture of 16 isotropic Gaussians laid on a grid. We present in Figure 2 the generated samples (in observation space) obtained when using different VAE priors and learning procedures.

In the top-left panel are the samples obtained when using joint training with a standard Gaussian prior. We observe that the model fails to capture the data distribution correctly. In top-right are the samples obtained by 2-step learning with a GMM prior, that is, by keeping the previously learned auto-encoder parts of the model and only fitting the GMM prior. We observe that the new model very accurately captures the data distribution.

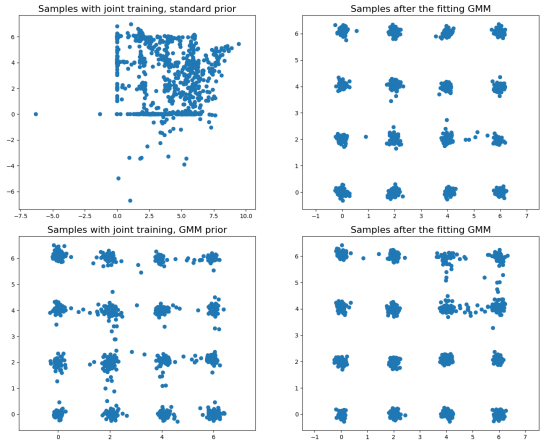

Fig. 2. Samples generated from different models, on a synthetic dataset with 16 Gaussian components. **(Top-left)** Joint training, standard-VAE. **(Top-right)** 2-step learning, fitting a GMM prior on a standard-VAE. **(Bottom-left)** Joint training, GMM-VAE. **(Bottom-right)** 2-step learning, fitting a new GMM prior on a GMM-VAE.

In the bottom left panel, we present the samples obtained after joint training with a GMM prior. This time the multimodal structure is better captured due to the more complex prior, however not perfectly. On the bottom right panel, we fit a new GMM on the latent space of the jointly learned VAE with GMM, and make the observation that the learnt distribution improves significantly. What this result underlines is that the inaccuracies resulting from joint training are due to the fact that optimizing for the discrete variables in the GMM with joint training is potentially difficult, especially when we initialize randomly. It is known that optimizing for discrete variables (such as GMM cluster indicators) is difficult to accomplish with gradient descent [26], and therefore it is advisable here to use two-step learning which enables us to utilize clever GMM initialization techniques such as kmeans++ [27].

4.1.2. Quantitative comparison with joint training

We now compare learning with two-step and joint training on the MNIST (handwritten digits) and CelebA (color human faces) datasets using the Fréchet inception distance (FID) [28], a widely accepted metric for measuring the quality of generated images.

In Figure 3 (left panel), we compare the FID scores for the MNIST dataset. We use a feedforward autoencoder with VAE inference. We see that the FID scores increase significantly when the two-step learning is applied over the standard VAE model to learn a GMM prior. We also observe that the joint GMM training works slightly better than the two-step learning approach, but the improvement is marginal compared to the large improvement gained with the two-step approach. Note that the error bars represent the uncertainty over five batches of generated images with 1,000 images per batch.

In Figure 3 (right panel), we run the same study using CelebA. We use the DC-GAN architecture [29], and experiment with both diagonal and full-covariance GMM priors. We again observe that applying the two-step learning, with a diagonal GMM, significantly improves the VAE with a standard normal prior. We also see that joint training with a diagonal GMM is slightly better than 2-step.

When we use a full-covariance GMM, however, 2-step training matches the performance of joint training with diagonal GMM. Note that implementing two-step learning with a full covariance GMM comes at minimal extra effort in the case where we use a software package that supports both the diagonal- and full-covariance GMMs, we used the scikit-learn library. In contrast, implementing the ELBO for joint training with different types of covariances requires careful implementation. This property of the two-step learning framework is applicable to any distribution: if learning software is available, one can simply utilize it without any extra effort.

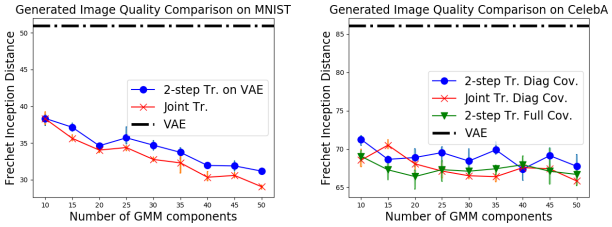

Fig. 3. FID scores on MNIST dataset (left), and CelebA dataset (right). The horizontal axis show the total number of Gaussian components.

4.2. VAE with HMM prior for time-series

In this section, we discuss learning an latent generative model in a situation where we must account for transitions between the data items, by incorporating a temporal structure into the prior distribution. One way to model such a temporal structure is with a hidden Markov model (HMM) [13], which results in the following generative process:

$$\begin{aligned} c_0 \sim \text{Cat}(\pi),\ c_t \mid c_{t\ 1} \sim\ \text{Cat}(A_{c_{t\ 1}}),\\ h_t \sim \mathcal{N}(\mu_{c_t}, \sigma^2_{c_t} I),\\ x_t \mid h_t \sim p(x \mid f_\theta(h_t)), \end{aligned}$$

where the cluster indicators c_t form a Markov chain, whose transition matrix is denoted by A. Note that an HMM can be seen in essence as a GMM whose cluster indicators follow a Markovian structure.

As articulated in Section 4.1, deriving the ELBO function for the GMM model, and carrying out the implementation is a significant undertaking which requires design decisions. The situation is even worse with a more involved prior distribution such as an HMM.

The main bottleneck in deriving an ELBO objective is computing its KL-term. The KL-term for an HMM is even more involved than a GMM, as the model does not produce i.i.d. observations, and therefore we would need to calculate the joint likelihood over all observations. The KL-term is:

$$\text{KL}(q_e(h_{1:T}) \| q_p(h_{1:T})) = \mathbb{E}_{q_e(h_{1:T})}\left[\log \frac{q_e(h_{1:T})}{q_p(h_{1:T})}\right].$$

One way to estimate it is with a Monte Carlo approximation,

$$\text{KL}(q_e(h_{1:T}) \| q_p(h_{1:T})) \propto \sum_n \log \frac{q_e(h^n_{1:T})}{q_p(h^n_{1:T})},$$

where n indexes samples obtained from the approximate posterior distribution $q_e(h_{1:T} \mid x_{1:T})$ (we refer to it as $q_e(h_{1:T})$ here for brevity). To compute this approximation, one needs to compute the model likelihood $q_p(h^n_{1:T})$. In HMMs, the model likelihood is given by the forward alpha-recursions [13]. While one can compute the gradients on the likelihood computed after the forward recursion, this implementation is again time-consuming. Moreover, we have experimentally observed that, when initialized randomly, the joint training is extremely unstable, and typically yields worse generations than that of the two-step learning.

4.2.1. *Experimental Results*

To demonstrate the quality of the two-step learning on a VAE framework with an HMM prior, for modeling audio data in the waveform domain. We work on audio data with 8kHz sampling rate. We dissect the audio into 100ms long chunks, where consecutive chunks overlap by 50ms, and each window is multiplied by a Hann window. In the first step, we learn a VAE with 80 dimensional latent representations for each chunk which is 800 samples long. We used a three-layer convolutional network both in the encoder and decoder, where we use filters of length 200 samples.

We then fit an HMM on the extracted latent representations, to learn the transitions between 800 sample long chunks. We use 100 HMM states, where each state has a diagonal covariance Gaussian emission model. The random samples are obtained by sampling from the fitted HMMs, and passing the sampled latent representation through the decoder. To reconstruct the generated chunks as an audio waveform, we follow the overlap-add procedure [30]. We overlap each generated chunk by 50 percent and add.

We train the model on a synthetic dataset that involves monophonic clarinet notes played within a two octave range. The dataset is comprised of 40 sequences which are all of 6 seconds long. In Figure 4, we show the spectrograms of the real data, and the generated data. We observe that the two-step learning approach results in much better generated data, compared to the joint training with random initialization. With two-step training, we can clearly see the harmonic structure of the played notes. We also see that joint training followed by initialization with two-step learning does not conspicuously improve the learning over two-step learning. We believe that this makes sense since the standard training method for HMMs is the Baum-Welch algorithm [13, 12], and backpropagation through the forward pass is not amenable to obtain a clear gradient signal. The two-step learning framework enables the incorporation of learning algorithms such as the Baum-Welch algorithm which is more stable than plain gradient descent, and therefore in case of prior distributions which involve discrete variables such as HMMs, two-step learning is highly preferable.

5. TWO-STEP LEARNING WHEN JOINT TRAINING IS NOT STRAIGHTFORWARD

As argued in Sections 4.1 and 4.2, two-step learning provides a convenient and robust way of learning latent generative models. Moreover, decoupling the learning of the prior distribution from the learning of the autoencoder enables other learning options than gradient descent via back-propagation. As previously showcased, using an expectation-maximization based learning procedure for fitting a GMM prior can result in a more accurate model of multi-modal data.

In some model combinations however, applying joint training is not straightforward, since the model does not define a joint distribution and hence the standard technique of deriving an ELBO is unavailable. We explore three such cases: 1) VAEs with GAN prior,

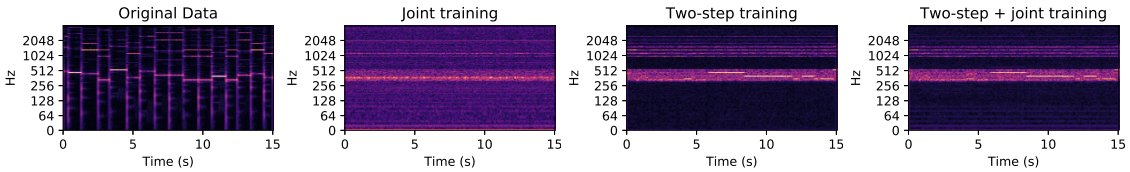

Fig. 4. Comparing 2-step training of the VAE-HMM model with joint training. **(left)** Spectrogram of a 15 seconds long excerpt from real data. **(middle-left)** Generated data with VAE-HMM trained with joint training. **(middle-right)** Generated data with VAE-HMM trained with two-step training. **(right)** Generated data with VAE-HMM trained with joint training initialized with two-step training.

2) GANs with GMM prior, and 3) generative models where the autoencoder is a principal component analysis (PCA) model.

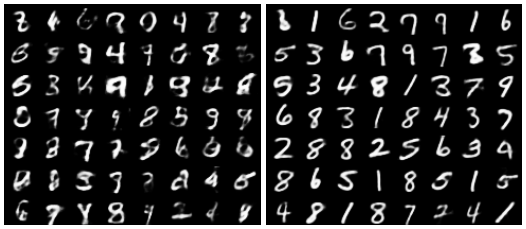

Fig. 5. MNIST digits generated from a standard VAE **(left)** versus a VAE with GAN prior learned via 2-step **(right)**.

5.1. VAEs with GAN priors

In Section 4.1 we used a GMM prior in order to model complex, multi-modal distributions in the latent space. Another option, as studied in [4], is to have a prior that is trained with a GAN objective. It is not clear how to directly write down an ELBO function to train a VAE with a GAN-trained prior. In the original paper [4], the authors effectively employ a 2-step learning protocol, where they first train a VAE, and then train a GAN as a prior distribution. In Figure 5, we show the results obtained with a GAN prior. On the left panel, we show the generated MNIST digits obtained from a standard VAE. On the right panel, we see that the generated MNIST digits improve significantly when we fit a GAN prior to learn the latent distribution.

5.2. GANs with GMM prior

GANs are known to generate sharp looking textures in images [29, 31, 32]. They are also known to generate out-of-distribution images. GANs typically use fixed priors such as standard Gaussian [2], and one way to improve GANs is to learn a GMM prior to account for the multi-modality in the latent space.

We propose to do this by training an encoder, that is previously trained with a GAN objective. Namely, we train a decoder $f_\theta(.)$ with a GAN objective. And then we train an encoder $g_\phi(.)$ to minimize the reconstruction errors,

$$\min_{\phi} \|x - f_\theta(g_\phi(x))\| \tag{3}$$

where the decoder parameters θ are not updated. We then fit a GMM on the latent codes $\widehat{h}$ obtained via passing the data items through the encoder, such that $\widehat{h} := g_\phi(x)$. We follow this procedure for different number of GMM components on the MNIST dataset. In Figure 6, we observe an improvement in terms of FID scores over the WGAN-GP [32] baseline (shown with the horizontal curve above), when we apply the 2-step learning procedure described above, for various number of GMM components. The error bars represent the uncertainty over 5 sample sets of 1,000 samples each.

5.3. PCA autoencoder with GMM prior

Another interesting case where applying 2-step learning is not straightforward is one in which we want to use an algorithm other than gradient descent for training the autoencoder. An example is combining a linear autoencoder learnt with PCA, with a multimodal prior. The resulting generative process is as follows:

$$h \sim \text{GMM}(.), \quad x \mid h \sim \mathcal{N}(Uh, I),$$

where U is a low rank (tall) matrix. It is possible to write an Expectation-Maximization algorithm (EM) [22] for this model, however the EM algorithm is known to get stuck in local optima [12]. The advantage of learning this model via two step is that, the PCA algorithm gives the globally optimal solution for the autoencoder, which results in sharp reconstructions. In a case where original data items are well aligned, we observed that this model can generate sharp and detailed images.

Our dataset consists of brain images of mice (an example is provided in Figure 7 left). The images are highly aligned, and therefore suitable for PCA. Note that the images are of size 300×400, but despite this large size, this model is able to generate images with very high fidelity (Figure 7 right), compared to the model which uses a convolutional autoencoder. The convolutional autoencoder model typically loses the fine grained details (Figure 7 middle), whereas the PCA model is able to preserve details.

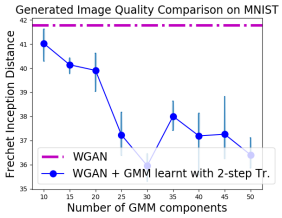

Fig. 6. Improving GANs with 2-step training: FID score (y-axis, lower the better) vs number of GMM components (x-axis).

6. CONCLUSIONS AND FUTURE WORK

In this paper, we formally introduce a two-step approach for learning latent generative models which effectively combines the learning of an autoencoder with the learning of a prior distribution of the latent codes. We demonstrate its effectiveness in terms of learning quality, required implementation efforts, and most importantly flexibility. We hope that practitioners will be convinced by both the correctness and the efficiency of two-step learning, and will be willing to explore new combinations of models which until now had