Generation of data on discontinuous manifolds via continuous stochastic non-invertible networks

Mariia Drozdova\textsuperscript{1,2}, Vitaliy Kinakh\textsuperscript{1}, Guillaume Quétant\textsuperscript{1,2}, Tobias Golling\textsuperscript{2} & Slava Voloshynovskiy\textsuperscript{1}
\textsuperscript{1}Department of Computer Science
\textsuperscript{2}Department of Particle Physics
University of Geneva
Switzerland
{mariia.drozdova,svolos}@unige.ch

Abstract

The generation of discontinuous distributions is a difficult task for most known frameworks such as generative autoencoders and generative adversarial networks. Generative non-invertible models are unable to accurately generate such distributions, require long training and often are subject to mode collapse. Variational autoencoders (VAEs), which are based on the idea of keeping the latent space to be Gaussian for the sake of a simple sampling, allow an accurate reconstruction, while they experience significant limitations at generation task. In this work, instead of trying to keep the latent space to be Gaussian, we use a pre-trained contrastive encoder to obtain a clustered latent space. Then, for each cluster, representing a unimodal submanifold, we train a dedicated low complexity network to generate this submanifold from the Gaussian distribution. The proposed framework is based on the information-theoretic formulation of mutual information maximization between the input data and latent space representation. We derive a link between the cost functions and the information-theoretic formulation. We apply our approach to synthetic 2D distributions to demonstrate both reconstruction and generation of discontinuous distributions using continuous stochastic networks.

1 Introduction

The generation of data with discontinuous distributions with non-invertible networks represents a great interest for many problems in high energy physics, astrophysics and chemistry all dealing with high dimensional data. The previous attempts to develop generative models for discontinuous distributions show limited performance of GANs \cite{1,2,3} and VAE models \cite{4}. Flow models \cite{5} can handle this problem to some extent but they face the complexity issues when the dimensionality of data increases. Hybrid models such as SurVAE \cite{6} try to solve this problem by a combination of non-invertible and invertible networks based on Flows.

In this paper, we present a new information-theoretic stochastic contrastive generative adversarial network SC-GAN. The SC-GAN is a hybrid system that is based on a deterministic encoder producing an interpretable latent space and a stochastic decoder representing a generator. The generator architecture is a set of fully connected layers implemented based on a stochastic EigenGAN network \cite{7} conditioned on a set of random noise vectors at each layer. The model is trained both in the reconstruction mode (with fixed noise vectors) and in the generative mode. The contrastive encoder is trained independently of the generator. The latent space of the encoder is then clustered using K-Means and approximated by many low-complexity mapping networks which try to shape Gaussians

\textsuperscript{*}M. Drozdova and S. Voloshynovskiy are corresponding authors.

the input into corresponding cluster distributions. Finally, the decoder is trained jointly for reconstruction and generation using the likelihood with the corresponding discriminators.

We provide an information-theoretical interpretation of the proposed model in Section 2. In Section 3 we perform the analysis of both auto-encoding mode and generation of toy 2D datasets: Eight Gaussians, Checkerboard, Two spirals, Abs, Sinewaved cube and Four circles.

2 Information-theoretic formulation

The proposed framework is schematically shown in Figure 1 and consists of three stages of training.

2.1 The training of the encoder (stage 1)

The encoder is trained to maximize the mutual information between the data $X$ and its latent space representation $E$:

$$\hat{\phi}_e = \arg\max_{\phi_e} I_{\phi_e}(X; E),$$  \hspace{1cm} (1)

where $I_{\phi_e}(X; E) = \mathbb{E}_{p(x, \epsilon)} \left[ \log \frac{q_{\phi_e}(\epsilon|x)}{q_{\phi_e}(\epsilon)} \right]$.

The encoder is trained independently from the decoder using contrastive losses (Figure 1a). The maximization can be considered in the scope of the InfoNCE framework [8] and technically implemented using for example SimCLR contrastive learning [9]. For our toy datasets we choose simple augmentations based on the addition of small noise to the input data.

2.2 The training of the latent space mapping networks (stage 2)

The mapping networks aim at generating the complex latent space $\epsilon$ from a simple distribution $p_\omega(\omega)$ (Fig. 1b). The latent space $\epsilon$ has a discontinuous clustered nature following the distribution $q_{\phi_e}(\epsilon)$. A simple continuous MLP mapper cannot generate complex $q_{\phi_e}(\epsilon)$ from a unimodal $p_\omega(\omega)$. For this reason, we consider splitting $q_{\phi_e}(\epsilon)$ on a set of unimodal sub-distributions in such a way that a simple MLP model can be used to generate each unimodal sub-distribution. We use a simple K-means [10] clustering to produce $K_\epsilon$ distinct subsets. Then for all $\epsilon \in C_i$, $i = 1, ..., K_\epsilon$, we trained adversarial auto-encoders (AAE) [11] with the latent space $p_\omega(\omega)$ following the Gaussian distribution. Alternatively, one can use Flows to construct a mapper from $p_\omega(\omega)$ to $q_{\phi_e}(\epsilon)$. Each AAE is defined by a pair of the encoder $q_{i\theta}(\omega|\epsilon)$ and the decoder $p_{i\theta}(\epsilon|\omega)$ for each cluster $i \in 1, ..., K_\epsilon$. The

Figure 1: The proposed framework: Stage 1 - training of the encoder, Stage 2- training of the mapping network, Stage 3 - training of the decoder for the simultaneous reconstruction and generation.
training of the AAE is based on the optimization problem:

$$(\hat{\phi}_i^{\omega}, \hat{\theta}_i^\varepsilon) = \arg\max_{\phi_i^{\omega}, \theta_i^\varepsilon} I_{\phi_i^{\omega}, \theta_i^\varepsilon}^i (E; W) + \lambda_{\varepsilon} I_{\phi_i^{\omega}, \theta_i^\varepsilon}^i (W; E),$$

where

$$I_{\phi_i^{\omega}}^i (E; W) = \mathbb{E}_{q_{\phi_i^{\varepsilon}}(\varepsilon)} \left[ \mathbb{E}_{q_{\phi_i^{\omega}}(\omega|\varepsilon)} \left[ \log \frac{q_{\phi_i^{\omega}}(\omega|\varepsilon)}{q_{\phi_i^{\omega}}(\omega)} \cdot \frac{p_\omega(\omega)}{p_\omega(\omega)} \right] \right] = \mathbb{E}_{q_{\phi_i^{\varepsilon}}(\varepsilon)} \left[ \mathbb{D}_{KL}(q_{\phi_i^{\omega}}(\omega|\varepsilon)||p_\omega(\omega)) - \mathbb{D}_{KL}(q_{\phi_i^{\omega}}(\omega)||p_\omega(\omega)) \right],$$

and

$$I_{\phi_i^{\omega}, \theta_i^\varepsilon}^i (W; E) = \mathbb{E}_{q_{\phi_i^{\varepsilon}}(\varepsilon)} \left[ \mathbb{E}_{q_{\phi_i^{\omega}}(\omega|\varepsilon)} \left[ \log \frac{p_{\theta_i^{\varepsilon}}(\varepsilon|\omega)}{q_{\phi_i^{\varepsilon}}(\varepsilon)} \right] \right] = H_{\phi_i^{\varepsilon}} (E) - H_{\phi_i^{\omega}, \theta_i^\varepsilon} (E|W),$$

where $H_{\phi_i^{\varepsilon}} (E) = -\mathbb{E}_{q_{\phi_i^{\varepsilon}}(\varepsilon)} \left[ \log q_{\phi_i^{\varepsilon}} (\varepsilon) \right], H_{\phi_i^{\omega}, \theta_i^\varepsilon} (E|W) = -\mathbb{E}_{q_{\phi_i^{\varepsilon}}(\varepsilon)} \left[ \mathbb{E}_{q_{\phi_i^{\omega}}(\omega|\varepsilon)} \left[ \log p_{\theta_i^{\varepsilon}} (\varepsilon|\omega) \right] \right]$ and $\lambda_{\varepsilon}$ is a constant for the importance of each term. As $\mathbb{E}_{q_{\phi_i^{\varepsilon}}(\varepsilon)} \left[ \mathbb{D}_{KL}(q_{\phi_i^{\omega}}(\omega|\varepsilon)||p_\omega(\omega)) \right] \geq 0$ and $H_{\phi_i^{\varepsilon}} (E)$ does not depend on the parameters of the networks, the training objective can be rewritten as:

$$(\hat{\phi}_i^{\omega}, \hat{\theta}_i^\varepsilon) = \arg\min_{\phi_i^{\omega}, \theta_i^\varepsilon} \mathbb{D}_{KL}(q_{\phi_i^{\omega}}(\omega)||p_\omega(\omega)) + \lambda_{\varepsilon} H_{\phi_i^{\omega}, \theta_i^\varepsilon} (E|W),$$

as the original AAE loss [11].
2.3 The training of the decoder (stage 3)

The decoder training is performed for both reconstruction and generation modes based on the optimization problem:

\[
\hat{\theta}_x = \text{argmax}_{\theta_x} \ I_{\theta_x; \theta_h}(E; X) + \lambda_x I_{\theta_x; \theta_h}(E; X),
\]

where the reconstruction mode corresponds to the term:

\[
I_{\theta_x; \theta_h}(E; X) = \mathbb{E}_{p_{\theta}(x)} \left[ \mathbb{E}_{q_{\theta^*}(e|x)} \left[ \log \frac{p_{\theta}(x|e)}{p_x(x)} \times \hat{p}_{\theta}(x) \right] \right] \\
= -H_{\theta_x; \theta_h}(X|E) - \mathbb{D}_{KL}(p_X(x)||\hat{p}_{\theta}(x)) + H(p_X(x); \hat{p}_{\theta}(x))
\]

with

\[
H_{\theta_x; \theta_h}(X|E) = \mathbb{E}_{p_{\theta}(x)} \left[ \mathbb{E}_{q_{\theta^*}(e|x)} \left[ \log p_{\theta}(x|e) \right] \right], \quad \mathbb{D}_{KL}(p_X(x)||\hat{p}_{\theta}(x)) = \mathbb{E}_{p_{\theta}(x)} \left[ \log \frac{p_{\theta}(x)}{\hat{p}_{\theta}(x)} \right]
\]

and the generation mode corresponds to the term:

\[
I_{\theta_x; \theta_h}(E; X) = \mathbb{E}_{p_{\theta}(x)} \left[ \mathbb{E}_{\theta_h(x|e)} \left[ \mathbb{E}_{p_{\theta}(x|e)} \left[ \log \frac{p_{\theta}(x|e)}{p_x(x)} \times \hat{p}_{\theta}(x) \right] \right] \right] \\
= \mathbb{E}_{p_{\theta}(x)} \left[ \mathbb{D}_{KL}(p_{\theta}(x|E = e)||\hat{p}_{\theta}(x)) \right] - \mathbb{D}_{KL}(p_X(x)||\hat{p}_{\theta}(x)),
\]

with \( p_{\theta}(e) = \mathbb{E}_{p_{\theta}(e)} \left[ \mathbb{D}_{KL}(p_{\theta}(x|E = e)||\hat{p}_{\theta}(x)) \right] + \lambda_x \mathbb{D}_{KL}(p_X(x)||\hat{p}_{\theta}(x)) \geq 0 \), the above optimization problem can be reduced to:

\[
\hat{\theta}_x = \text{argmin}_{\theta_x} H_{\theta_x; \theta_h}(X|E) + \mathbb{D}_{KL}(p_X(x)||\hat{p}_{\theta}(x)) + \lambda_x \mathbb{D}_{KL}(p_X(x)||\hat{p}_{\theta}(x)).
\]

3 Experiments and Conclusions

<table>
<thead>
<tr>
<th>Model</th>
<th>single Gaussian pdf</th>
<th>noisy cluster centers</th>
<th>mapping network</th>
<th>real data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eight Gaussians</td>
<td>2.339</td>
<td>0.055</td>
<td>0.065</td>
<td>0.016</td>
</tr>
<tr>
<td>Checkerboard</td>
<td>0.337</td>
<td>0.034</td>
<td>0.015</td>
<td>0.009</td>
</tr>
<tr>
<td>Two Spirals</td>
<td>1.771</td>
<td>0.058</td>
<td>0.062</td>
<td>0.011</td>
</tr>
<tr>
<td>Abs</td>
<td>0.133</td>
<td>0.021</td>
<td>0.029</td>
<td>0.019</td>
</tr>
<tr>
<td>Sinewaved cube</td>
<td>0.062</td>
<td>0.023</td>
<td>0.024</td>
<td>0.021</td>
</tr>
<tr>
<td>Four circles</td>
<td>0.031</td>
<td>0.034</td>
<td>0.044</td>
<td>0.035</td>
</tr>
</tbody>
</table>

Table 1: The reconstruction results for the different latent space models: latent vector sampled from a single Gaussian pdf, from cluster centers with Gaussian noise, from the mapping network output and from random training data latents.

We perform generation experiments on 2D datasets using different ways for latent space modeling. In the first setting \( \varepsilon \) is sampled from the Gaussian probability density function. Then we cluster the \( \varepsilon \)-space using K-Means. In the second setting we place the Gaussian in the cluster centers and use this as input. Finally, we train an individual network for each cluster to shape the Gaussian closer to the real shape of the cluster (stage 2 of the training). For generation we can also use \( \varepsilon \) from the subset used to train the encoder. This case is an extreme case when the number of clusters is equal to the size of the dataset. We show the results of the generation in Fig. 2.

Covering the latent space with the clusters which are approximated by simple fully connected layers leads to state-of-the-art results (the fourth column in Fig. 2) for the generative models which are not based on INNs.

We show the Mean Square Error in Table 1 to demonstrate reconstruction error. We fix the noise vectors and we take the latent vector directly from the output of the encoder from Stage 1. The
training stage of generation is different: we take the latent vector from a single Gaussian pdf, from cluster centers with the Gaussian noise, from the mapping network and from real data. We notice that despite poor generation, network can still perform good results in reconstruction with simplest latent space modeling (one Gaussian) which is a sign of overfitting. It is also interesting to note that the better generation is, the worse are the results of reconstruction and vice versa. Modeling the latent space as discontinuous allow us to marry the mode of reconstruction and generation. In the limit case with number of clusters equal to the number of points in the train set we get the best results.

Acknowledgments and Disclosure of Funding

This research was partially funded by the SNF Sinergia project (CRSII5-193716): Robust Deep Density Models for High-Energy Particle Physics and Solar Flare Analysis (RODEM). The authors are thankful to Johnny Raine and Sebastian Pina-Otey for their feedback on the paper and discussion.

References


