ASSIGNMENT 2 : LATENT SPACE REPRESENTATION

GANgsters

November 2024

Jacques SUN, Nour RIZK, Sakula HYS

INTRODUCTION

This project focuses on exploring the precision and recall trade-off of Generative Adversarial Networks (GANs). We will first start by reminding the standard objective of the Vanilla GAN as well as its limits. Second, we focus on exploring the latent space representation to guide image generation; specifically, clustering the latent space to achieve better results. In this project, we decide to favor the precision value.

Generative Adversarial Network

GANs are generative models based on a Generator-Discriminator architecture. While the generator G learns to create realistic data to fool the discriminator D, the latter learns to distinguish between real and generated data. Thus, the GAN training sets up a two-player game between G and D, which is defined by the minimax objective:

$$\mathcal{L}adv = \min \Theta_G \max_{\Theta_D}; \mathbb{E}x \sim P_x^r \left[q(D(x))\right] + \mathbb{E}z \sim P_z \left[q(1 - D(G(z)))\right] \quad (1)$$

where P_x^r is the distribution of real data samples, P_z is the prior noise distribution on the latent space, and $q(.)$ is the quality function. For the Vanilla GAN, $q(.)$ is the logarithm function.

We will specifically be working on the MNIST dataset. For relevant comparison, the architecture of both the generator and the discriminator is fixed. The goal is to generate visually realistic digits and optimize the Fréchet Inception Distance, precision (i.e., quality), and recall (i.e., diversity) of such a model, by acting on the latent space representation, for instance.

Limits of Vanilla GAN

While being a prominent generative model, the classic GAN exhibits some limitations, such as:

- Mode collapse: leading to generating low-diversity samples, resulting in low recall metrics.
- Instability due to the oscillation between the discriminator and the generator (e.g., vanishing gradients).

Furthermore, the image generation task raises the question of the quality-diversity trade-off (precision and recall).

FIRST APPROACHES

Wasserstein GAN

Our first idea, after running the vanilla GAN for 100 epochs, was to test the Wasserstein GAN [1]. This method is known to avoid mode collapse and thus would be more stable than the vanilla GAN. However, problems with convergence made us switch to the study of other methods.

Perceptual Loss

The standard loss function for the vanilla GAN is the binary cross-entropy loss. It is computed on the pixel space, i.e., it compares the difference in pixel values between the generated image and the real one. While this loss is relevant for the quality of the image, it struggles to capture the global structure and high-level features of the image.

To address this issue, we decided to add a perceptual loss to the standard GAN objective of the model. This loss compares the features, extracted by a frozen pretrained network such as VGG16 (instead of the pixel values), from the generated image and the real one. This way, the model is able to capture the global structure of the image.

The perceptual loss between a real image x and a generated image $\hat{x}$ is given by:

$$\mathcal{L}\text{perceptual}(x, \hat{x}) = \sum l = 1^{L} \lambda_l \left| \phi_l(x) - \phi_l(\hat{x}) \right|_2^2 \tag{2}$$

- $\phi_l(\cdot)$ is the feature map of layer l in the pretrained network ϕ.
- λ_l is the weight for the contribution of layer l, equal to 1 in our case.

In this case, the global GAN loss is given by the sum of the classic GAN loss of Eq. (1), and the weighted perceptual loss of Eq. (2). We chose a perceptual weight of 0.01. Using this technique partially improved our precision, at the expense of the FID value. Thus, we couldn't reach satisfaction.

This might be explained by the fact that the perceptual loss is generally used in the context of complex image datasets such as ImageNet. However, the MNIST images are relatively simple black-and-white digit pictures. Thus, the perceptual loss might be too strong in this case. To better understand the feature representation, we decided to focus on the latent space representation, which led us to explore the ClusterGAN method.

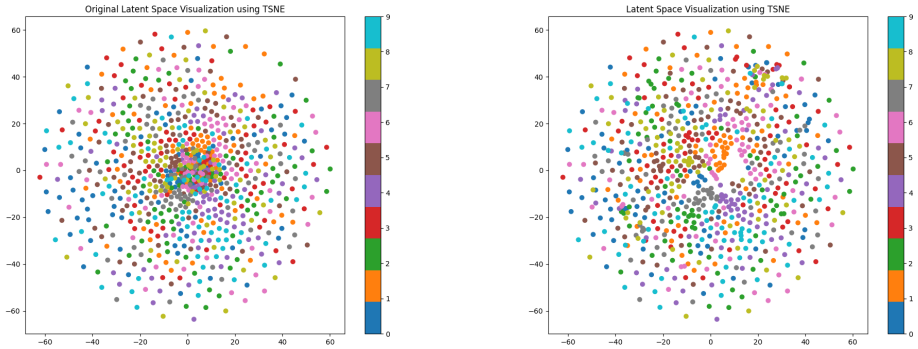
LATENT SPACE CLUSTERING

Latent Space Exploration

To visualize trends within our latent space, we start by sampling data points from it and using a classifier to determine the initial labels associated with the sampled noise. Next, we iteratively adjust these noise vectors through gradient ascent, optimizing them to maximize the probability of the desired label.

In practice, we implement the t-SNE [2] function for visualizing the latent space embeddings, with colors corresponding to their digit classes. Even though t-SNE is a nonlinear projection of high-dimensional latent space into a 2-dimensional space, it provides great intuition about the structure of the latent space.

Before applying any major changes, we visualize the original Fig. (1a) and trained Fig. (1b) standard GAN latent distributions. Neither distribution demonstrates a clear pattern or cluster structure with respect to class labels.



(a) Original latent vectors (b) Latent vectors after gradient ascent

Figure 1: Performing gradient ascent

ClusterGAN

Our strategy is to restructure our latent space via clustering to discover latent noise samples maximizing the generation of each digit class. Consequently, we constrain the GAN to generate images of better quality (i.e., increased precision).

• Method

ClusterGAN [3] focuses on using a modified GAN architecture to achieve unsupervised clustering by using a discrete-continuous sampling technique for the latent space and integrating an encoder network. In fact, ClusterGAN samples latent vectors using a mixture of continuous Gaussian noise z_n and discrete one-hot encoded vectors z_c. This combination allows the generator to learn distinct modes that correspond to different clusters in the data space, which in turn prevents mode collapse. Then, the method incorporates an encoder network that maps generated data back into the latent space. This inversion mapping ensures accurate reconstruction while guiding our sampling strategy by exploring the latent space.
By training the Generator, Discriminator, and Encoder networks together, we aim to achieve a latent space representation that has well-separated clusters. Consequently, by guiding the noise sampling in the latent space, we hope to obtain images with higher quality and greater diversity (since we enforce sampling from each cluster).

• Implementation

We jointly train the generator and the encoder on the $\mathcal{L}_{GE}$ loss given by:

$$\mathcal{L}_{GE} = \mathcal{L}_{adv} + \beta_n \cdot \mathcal{L}_{zn} + \beta_c \cdot \mathcal{L}_{zc} \tag{3}$$

where $\mathcal{L}_{zn}$ is the MSE loss for the continuous part of the latent vector, and $\mathcal{L}_{zc}$ is the Cross-Entropy loss for the discrete part of the latent vector:

$$\mathcal{L}_{zn} = \|z_n - E(G(z))_n\|_2^2 \qquad \mathcal{L}_{zc} = -\sum_{i=1}^{10} z_{c,i} \log(E(G(z))_{c,i}) \tag{4}$$

and β_n and β_c are the weighting factors for the continuous and discrete latent losses, respectively. As for the discriminator, it is trained on the $\mathcal{L}_{adv}$ loss given in Eq.(1). The architecture of the clusterGAN (as detailed in the original paper) is shown in Fig.2.

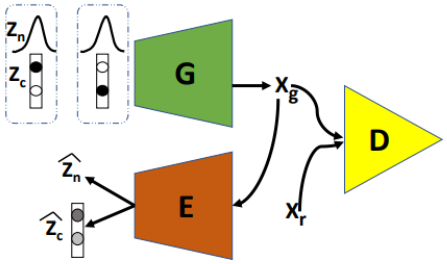


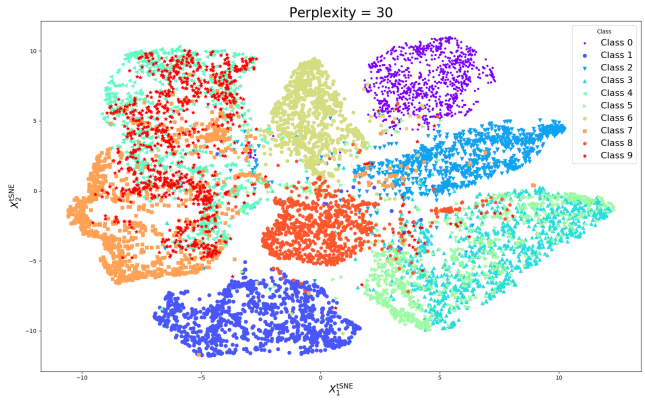
Figure 2: ClusterGAN architecture, source[3]

Finally, in regards to the image generation task, and as to preserve the original GAN architecture, we sample our input latent noise from the joint discrete and continuous distribution but feed a single vector (of dimension $\dim(z_n) + \dim(z_c)$) to the generator. We remind that the z_c are one-hot encoded vectors of dimension 10, ie. the number of classes.

• Results

The clusterGAN method is expected to perform better than the Vanilla GAN because clustering the latent space helps capture meaningful structures of the data. This should encourage the GAN to better understand the characteristics of each class of digits, which in returns should generate images of better quality. The use of clusters also helps generating samples from different categories, as well as choosing specific categories to generate from.

Once we train our three neural networks models, we visualize the latent space via a t-SNE projection. As can be seen in Fig.(3a), inter-class separability increases and we observe clear clusters corresponding to each digits. Additionally, we can see in Fig.(3b) that some classes have a good image quality, whereas the model still struggles with classes with similar features such as 4 & 9, and 5 & 3. Overall, the implementation of the clusterGAN based approach yields a lower FID value, and a slightly better precision value, compared to the previous implementations.



(a) ClusterGAN latent vectors

(b) Generated imaged based on ClusterGAN

Figure 3: Cluster GAN visual performance

Improvement

• Improving Recall (IR)

Surprisingly, ClusterGAN manages to maintain the continuity of the latent space by fixing the continuous latent vector z_n and interpolating between two discrete labels, $z_{c,1}$ and $z_{c,2}$. This observation led us to the idea of improving recall by adjusting the label values. Instead of setting the label to exactly 1, we sampled the labels from a normal distribution centered at $\mu = 1.1$ with a standard deviation of $\sigma = 0.15$.

• Gaussian Mixture

In an attempt to improve the quality and diversity of the generated images, we modified the ClusterGAN strategy by sampling latent vectors from a Gaussian Mixture Model (GMM) instead of sampling from pure noise and selecting the categorical label afterward. The intention was to directly capture the multi-modal nature of the data in the latent space by using a mixture of Gaussian distributions to represent each class, rather than relying on the discrete one-hot encoding for class identification. This approach was expected to better reflect the underlying structure of the data, allowing for more refined control over the generation process and potentially improving the separation between clusters. However, the results did not meet expectations; the GMM-based sampling led to less sharp and diverse images.

METRIC EVALUATION

In conclusion, by using a discrete one-hot encoded vector as part of the latent space, ClusterGAN enforces a clear separation between different classes or modes in the data. This helps the generator focus on producing high-quality samples for each specific mode, improving precision. The discrete component may introduce a limitation in exploring the full data distribution because the one-hot encoding inherently limits the generator to a fixed number of modes (defined by the one-hot vector dimension). As can be seen in the Tables, this reduced

Metrics	FID	Precision	Recall
ClusterGAN (d=20)	**9.60**	**0.77**	**0.32**
ClusterGAN (d=100)	9.66	0.76	0.31
ClusterGAN (d=200)	12.61	0.65	0.41

Table 1: Metrics over the ClusterGAN model with different latent space dimensions

Metrics	FID	Precision	Recall
VanillaGAN	15.13	0.62	0.47
ClusterGAN (GMM)	14.0	0.8636	0.14
ClusterGAN	9.60	0.77	0.32
ClusterGAN with IR	10.06	0.76	0.33

Table 2: Metrics over the different models with d=20

recall. However, the continuous Gaussian component of the latent space helps mitigate this by allowing for some variability within each mode, improving coverage of the data distribution.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[2] T. Tony Cai and Rong Ma. Theoretical foundations of t-sne for visualizing high-dimensional clustered data, 2022.

[3] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan : Latent space clustering in generative adversarial networks, 2019.