Final Project – Deep Learning

SAN : Metrizability of GAN

Conducted at Université Paris-Dauphine PSL

by

Evelina Spac – Capucine Brisson

December 18, 2025

CONTENTS

1 Introduction

1.1 GAN Presentation

Generative Adversarial Networks learn a target probability distribution by training two neural networks in opposition : a generator that produces fake samples and a discriminator that tries to distinguish real samples from artificial ones. To achieve this, the generator and discriminator are optimized in a minimax way.

Let μ_0 denote the target data distribution on a space $\mathcal{X} \subset \mathcal{R}^D$, $\mu_\theta = g_{\theta\,\sharp}\,\sigma$ be the distribution induced by a generator $g_\theta : \mathcal{Z} \to \mathcal{X}$, a probability measure $\sigma \in \mathcal{P}(\mathcal{Z})$ and where f is the discriminator.

$$\max_{f\in\mathcal{F}} V(f;\mu_\theta), \qquad \min_{\theta} J(\theta; f), \tag{1}$$

1.2 GAN Evaluation

To evaluate the quality of a generative model, we rely on the notions of **precision** and **recall**. In the context of GANs, precision measures the quality of generated samples, while recall measures the diversity of the generator with respect to the real data distribution.

Let $P \in \mathcal{P}(\mathcal{X})$ denote the data distribution and $\hat{P} \in \mathcal{P}(\mathcal{X})$ the model distribution. The support-based precision α and recall β of P with respect to $\hat{P}$ are defined as

$$\alpha = \hat{P}(\mathrm{Supp}(P)), \qquad \beta = P(\mathrm{Supp}(\hat{P})), \tag{2}$$

where $\mathrm{Supp}(\cdot)$ denotes the support of a distribution.

1.3 GAN Limitations

Despite their empirical success, GANs suffer from practical limitations. In classical GANs, the discriminator is trained as a binary classifier, and the generator is updated using gradients derived from this classification objective. While this approach can work, it does not guarantee that minimizing this problem brings the learned distribution closer to the target one in a mathematical sense.

This raises a question: **How can GANs be trained so that they actually minimize a well-defined distance between the data distribution and the generator distribution ?**

Answering this question requires understanding under which conditions the discriminator can be interpreted as defining a valid notion of distance. Thus, we introduce the concept of metrizability, which plays a central role in SAN.

2 Metrizability of the Discriminator

Metrizability (DEF) : A discriminator f is said metrizable if minimizing the generator objective $J(\theta; f)$ is equivalent to minimizing a true distance $D(\mu_0, \mu_\theta)$ between probability measures.

In the SAN framework, metrizability is ensured when the discriminator satisfies three conditions : **direction optimality, separability, and injectivity.** These conditions are sufficient to guarantee that the discriminator implements a distance between distributions, so that the generator receives meaningful information through the gradient.

2.1 Sliced Optimal Transport

A classical way to compute a distance in GANs is to use the Wasserstein distance, but its limitation is that it is computationally intractable in high dimensions. However, in one dimension, it admits a closed-form expression.

$$W_p(\mu,\nu) = \left(\int_0^1 \left| F_\mu^{-1}(t) - F_\nu^{-1}(t) \right|^p \, dt \right)^{\frac{1}{p}}, \tag{3}$$

Sliced optimal transport reduces high-dimensional transport to a collection of one-dimensional problems by projecting data onto directions ω. A direct application of this idea leads to the Sliced Wasserstein GAN, in which the discriminator evaluates discrepancies between the real and generated distributions by computing Wasserstein distances along multiple projected directions.

However, this approach suffers from an important limitation : the projection directions ω are not optimized and may fail to capture the most discriminative discrepancies between the distributions. As a result, even though sliced transport can be implemented, it does not guarantee that the discriminator induces a meaningful distance.

2.2 Direction optimality

To address this issue, we introduce the notion of direction optimality. Instead of averaging over arbitrary projections, we seek the optimal direction $\omega^\star$ defined as follows :

$$\omega^\star = \arg\max_{\omega \in S^{D-1}} d_{\langle \omega, h\rangle}(\mu_0, \mu_\theta) \tag{4}$$

The condition of direction optimality ensures that the discriminator projects the data onto the one-dimensional direction ω that maximizes the difference between the real and generated distributions, providing the generator gradients aligned with the true discrepancy between distributions.

2.3 Separability

The separability condition implies that, once the data are projected onto the optimal one-dimensional direction ω, the cumulative distribution functions do not cross.

$$F_{\mu_0}^{h,\omega^\star}(\xi) \;\leq\; F_{\mu_\theta}^{h,\omega^\star}(\xi), \qquad \forall\, \xi \in R \tag{5}$$

This condition ensures that the discriminator provides a coherent and unambiguous notion of distance between distributions, so that the resulting gradients consistently guide the generator toward the target distribution.

2.4 Injectivity

Finally, the injectivity of the discriminator ensures that no information is lost when mapping original samples $x \sim \mu_\theta$ and $x \sim \mu_0$ into the feature space. Without injectivity, different samples may collapse to the same representation, making it impossible to distinguish distributions even if direction optimality and separability hold.

3 Slicing Adversarial Network (SAN)

While some GAN variants satisfy one or two of these conditions, no classical GAN satisfies all three simultaneously. In particular, although Wasserstein GANs are theoretically well-founded, separability is often violated in practice. Similarly, we have shown that SWGANs do not satisfy the direction optimality condition.

This limitation motivates the Slicing Adversarial Network (SAN), which introduces a simple but powerful modification to standard GANs by defining a metrizable optimization problem. SAN ensures that the training process optimizes both the feature representation and a meaningful distance in a separable manner. It enforces direction optimality while preserving separability and injectivity.

3.1 Discriminator Formulation

In SAN, the discriminator is no longer formulated as a binary classifier, but rather as a directional projection operator that measures a signed distance between distributions. Given an input sample $x \in \mathcal{X}$, the discriminator is defined as

$$f(x) = \langle \omega, h(x) \rangle, \tag{6}$$

where $h : \mathcal{X} \to R^D$ denotes a feature mapping and $\omega \in S^{D-1}$ is a unit projection direction.

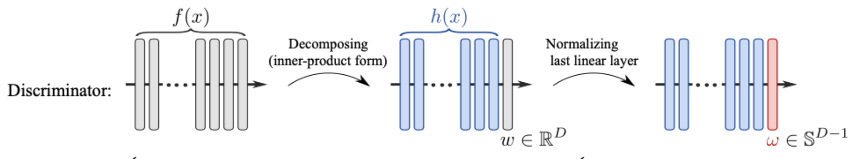
Discriminator:
$f(x)$
Decomposing (inner-product form)
$h(x)$
$w \in \mathbb{R}^D$
Normalizing last linear layer
$\omega \in \mathbb{S}^{D-1}$

Figure 1: Converting GAN to SAN

3.2 Decoupled Optimization of Features and Direction

A key innovation of SAN lies in its decoupled optimization strategy, which separately optimizes the feature representation and the projection direction. The discriminator optimization problem is split into two distinct components in order to achieve direction optimality of ω while preserving separability of h.

During training, the discriminator output is decomposed into two components : a functional output used to optimize the feature representation and a directional output used to optimize the projection direction.

$$\begin{aligned} f_{\text{fun}}(x) &= \langle \hat{\omega}^{-}, h(x) \rangle, \\ f_{\text{dir}}(x) &= \langle \hat{\omega}, h(x)^{-} \rangle. \end{aligned} \tag{7}$$

where $(\cdot)^{-}$ denotes the stop-gradient operator.

3.3 Training Objective

SAN combines a hinge-based adversarial loss to enforce separability, with a directional loss that promotes direction optimality. The novel maximization problem is defined as follows :

$$\max_{\omega \in S^{d-1},\, h \in \mathcal{F}(\mathcal{X})} \mathcal{V}^{\text{SAN}}(\omega, h; \mu_\theta) \;:=\; \mathcal{V}\big(\langle \omega^{-}, h \rangle; \mu_\theta\big) \;+\; \lambda\, d_{\langle \omega, h^{-} \rangle}(\mu_0, \mu_\theta)\,, \tag{8}$$

where $(\cdot)^{-}$ denotes the stop-gradient operator.

The Hinge Loss is defined as follows :

$$\mathcal{V}_{\text{hinge}}(\langle \omega, h \rangle; \mu_\theta) \;:=\; E_{x\sim\mu_0}[\min(0, -1 + \langle \omega, h(x) \rangle)] + E_{x\sim\mu_\theta}[\min(0, -1 - \langle \omega, h(x) \rangle)]\,. \tag{9}$$

Due to the margin constraint imposed by the hinge loss, samples satisfying $f(x) \geq 1$ for real data and $f(x) \leq -1$ for generated data do not contribute to the optimization. As a consequence, the discriminator objective is effectively optimized over truncated distributions rather than the original ones.

And the directional loss :

$$\sup_{\|\omega\|=1} \left(E_{x\sim\mu_0}[\langle \omega, h(x) \rangle] - E_{x\sim\mu_\theta}[\langle \omega, h(x) \rangle] \right). \tag{10}$$

This combination of components ensures that the discriminator simultaneously learns a structured feature space and a meaningful distance between distributions.

On the generator side, training is also performed using a hinge-based adversarial loss. However, the generator benefits from a discriminator that induces a stable and metrizable distance between distributions, resulting in improved convergence and a significant reduction of mode collapse.

4 Results

We first trained a standard GAN, which achieved a recall of 0.23 and a precision of 0.52. After switching to the SAN framework, we observed a significant improvement in both metrics, with recall increasing to 0.59 and precision to 0.76.

As we can see in Figure 2, SW-GAN produces noisy and poorly structured samples because the projection directions are not optimized and remain essentially random, leading to weak and inconsistent training signals. In contrast, in Figure 3-4, SAN explicitly optimizes the slicing direction, resulting in more informative gradients and significantly improved sample quality.

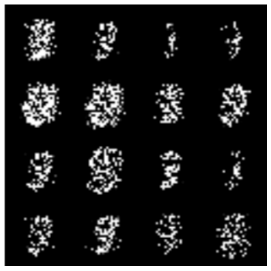
Figure 2: SW-GAN with 100 epochs and 64 batch size

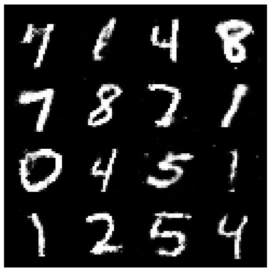
Figure 3: SAN with 100 epochs and 128 batch size

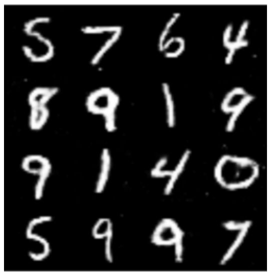
Figure 4: SAN with 300 epochs and 128 batch size

The CDFs of the real and generated (fake) distributions are plotted at different training epochs. Their maintained separability while converging illustrates that SAN identifies a right projection direction, enabling the generator to move closer to the target distribution as training progresses.

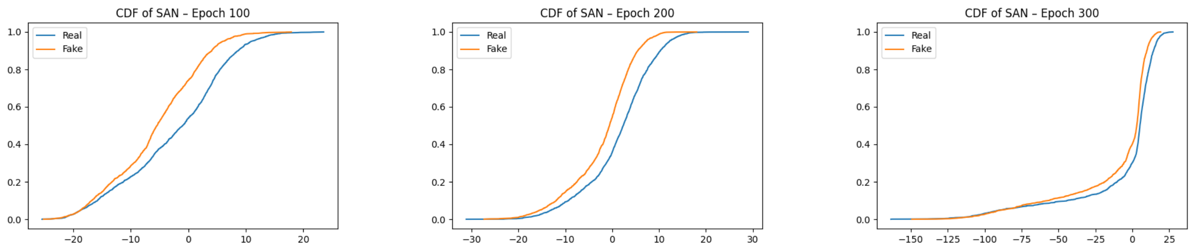

Figure 5: Evolution of CDF during training. From left to right: SAN (300 epochs, batch size 128)

5 Future Work and Perspectives

Several directions could be explored to further extend and improve the SAN framework.

- An interesting extension would be to investigate **conditional SAN** models by incorporating class into both the discriminator and the generator.
- While our experiments focus on a simple linear GAN architecture, applying SAN to more expressive GAN architectures such as **BigGAN** or **StyleGAN** could lead to improved performance.

6 SAN implementation

6.1 SAN Discriminator Implementation - Code

Listing 1: SAN discriminator implementation

```python
class SAN_Discriminator(nn.Module):
    def __init__(self, d_input, dim=1024, num_class=1):
        super(SAN_Discriminator, self).__init__()
        self.fc1 = nn.Linear(d_input, dim)
        self.fc2 = nn.Linear(self.fc1.out_features, self.fc1.out_features // 2)
        self.fc3 = nn.Linear(self.fc2.out_features, self.fc2.out_features // 2)
        self.fc_w = nn.Parameter(torch.randn(1, dim // 4))

    def forward(self, x, class_ids=None, flg_train=True):
        x = F.leaky_relu(self.fc1(x), 0.2)
        x = F.leaky_relu(self.fc2(x), 0.2)
        x = F.leaky_relu(self.fc3(x), 0.2)

        h_feature = x
        h_feature = torch.flatten(h_feature, start_dim=1)

        omega = self.fc_w # Direction parameter
        direction = F.normalize(omega, dim=1)
        norme = torch.norm(omega, dim=1).unsqueeze(1)

        h_feature = h_feature * norme # Feature projection

        # Functional / directional separation
        if flg_train:
            out_fun = (h_feature * direction.detach()).sum(dim=1)
            out_dir = (h_feature.detach() * direction).sum(dim=1)
            out = dict(fun=out_fun, dir=out_dir)
        else:
            out = (h_feature * direction).sum(dim=1)
        return out
```

6.2 SAN Discriminator Training - Pseudocode

Algorithm 1 SAN Discriminator Training

```
1: Input: $x_{\text{reel}}, G, D, optimizer$
2:     Initialize gradients
3:     $D_{\text{reel}} \leftarrow D(x_{\text{real}})$
4:     Compute the SAN Loss on real images
5:     Sample noise $z \sim \mathcal{N}(0, I)$
6:     Generate fake samples using the generator
7:     $D_{\text{fake}} \leftarrow D(x_{\text{fake}})$
8:     Compute the SAN Loss on fake images
9:     $L_D \leftarrow L_{\text{real}} + L_{\text{fake}}$
10:    Backpropagate the combined loss through the discriminator
11:    Update the discriminator parameters using the optimizer
12:    return $L_D$
```

6.3 SAN Loss Function - Pseudocode

Algorithm 2 SAN Loss Function

```
1: Input: discriminator outputs $f$ (function), $d$ (direction), label is_real
2: if is_real then
3:     $L_f \leftarrow \max(0, 1 - f)$
4:     $L_d \leftarrow -d$
5: else
6:     $L_f \leftarrow \max(0, 1 + f)$
7:     $L_d \leftarrow d$
8: end if
9: return $L_f + L_d$
```

References

[1] Yuhta Takida, Masaaki Imaizumi, Takashi Shibuya, Chieh-Hsin Lai, Toshimitsu Uesaka, Naoki Murata, and Yuki Mitsufuji. SAN: Inducing metrizability of GAN with discriminative normalized linear layer. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] Ishan Deshpande, Ziyu Zhang, and Alexander Schwing. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. Mnist handwritten digit database. `http://yann.lecun.com/exdb/mnist/`, 2010. Accessed: 2010.