

GENERATIVE ADVERSARIAL NETWORKS (GANs)

Jérémie Cabessa

Laboratoire DAVID, UVSQ

INTRODUCTION

- ▶ Les **generative adversarial networks (GANs)** sont des *modèles génératifs* capables de produire des data *réalistes*, i.e., similaires à celles d'un dataset réel [[Goodfellow et al., 2014](#)].
 - ▶ Les data (images) sont générées à partir d'un *bruit aléatoire*.
 - ▶ Applications:
 - Génération d'images de haute résolution (super-résolution) ou d'images photoréalistes.
 - Inpainting: reconstitution de parties d'images manquantes.
 - Transfert de style, coloration automatique, etc.

INTRODUCTION

- ▶ Les **generative adversarial networks (GANs)** sont des *modèles génératifs* capables de produire des data *réalistes*, i.e., similaires à celles d'un dataset réel [[Goodfellow et al., 2014](#)].
 - ▶ Les data (images) sont générées à partir d'un *bruit aléatoire*.
 - ▶ Applications:
 - Génération d'images de haute résolution (super-résolution) ou d'images photoréalistes.
 - Inpainting: reconstitution de parties d'images manquantes.
 - Transfert de style, coloration automatique, etc.

INTRODUCTION

- ▶ Les **generative adversarial networks (GANs)** sont des *modèles génératifs* capables de produire des data *réalistes*, i.e., similaires à celles d'un dataset réel [Goodfellow et al., 2014].
 - ▶ Les data (images) sont générées à partir d'un *bruit aléatoire*.
 - ▶ Applications:
 - Génération d'images de haute résolution (super-résolution) ou d'images photoréalistes.
 - Inpainting: reconstitution de parties d'images manquantes.
 - Transfert de style, coloration automatique, etc.

INTRODUCTION

- ▶ Les **generative adversarial networks (GANs)** sont des *modèles génératifs* capables de produire des data *réalistes*, i.e., similaires à celles d'un dataset réel [[Goodfellow et al., 2014](#)].
 - ▶ Les data (images) sont générées à partir d'un *bruit aléatoire*.
 - ▶ Applications:
 - Génération d'images de haute résolution (super-résolution) ou d'images photoréalistes.
 - Inpainting: reconstitution de parties d'images manquantes.
 - Transfert de style, coloration automatique, etc.

INTRODUCTION

- ▶ Les **generative adversarial networks (GANs)** sont des *modèles génératifs* capables de produire des data *réalistes*, i.e., similaires à celles d'un dataset réel [[Goodfellow et al., 2014](#)].
 - ▶ Les data (images) sont générées à partir d'un *bruit aléatoire*.
 - ▶ Applications:
 - Génération d'images de haute résolution (super-résolution) ou d'images photoréalistes.
 - Inpainting: reconstitution de parties d'images manquantes.
 - Transfert de style, coloration automatique, etc.

INTRODUCTION

- ▶ Les **generative adversarial networks (GANs)** sont des *modèles génératifs* capables de produire des data *réalistes*, i.e., similaires à celles d'un dataset réel [[Goodfellow et al., 2014](#)].
 - ▶ Les data (images) sont générées à partir d'un *bruit aléatoire*.
 - ▶ Applications:
 - Génération d'images de haute résolution (super-résolution) ou d'images photoréalistes.
 - Inpainting: reconstitution de parties d'images manquantes.
 - Transfert de style, coloration automatique, etc.

INTRODUCTION



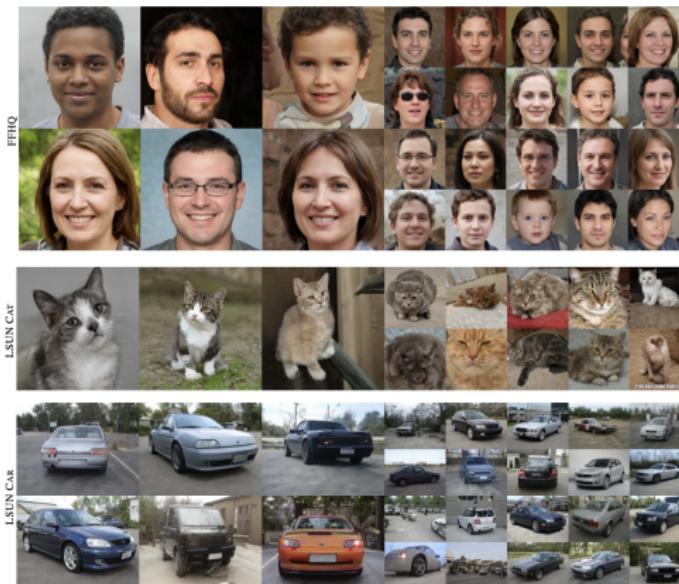
Figures taken from Li et al. (2017)

INTRODUCTION



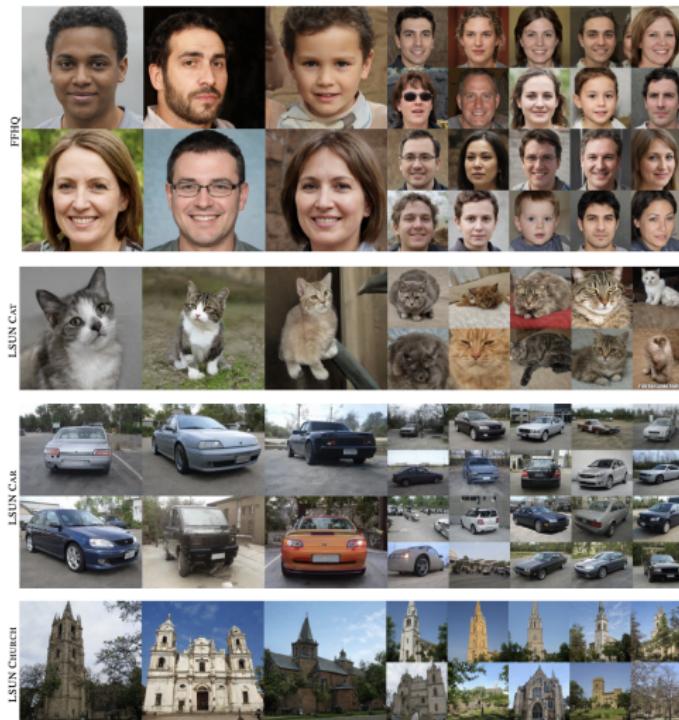
Figures taken from Li et al. (2017)

INTRODUCTION



Figures taken from Li et al. (2017)

INTRODUCTION



Figures taken from Li et al. (2017)

INTRODUCTION

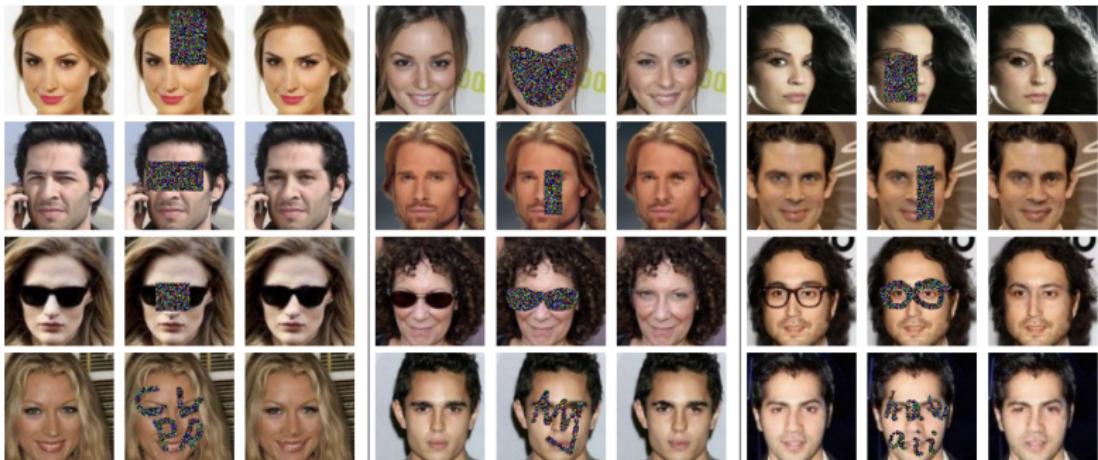
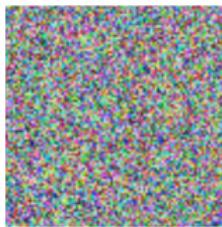


Figure taken from Li et al. (2017)

INTRODUCTION

Noise $\sim \mathbf{N}(0,1)$



Generative Model

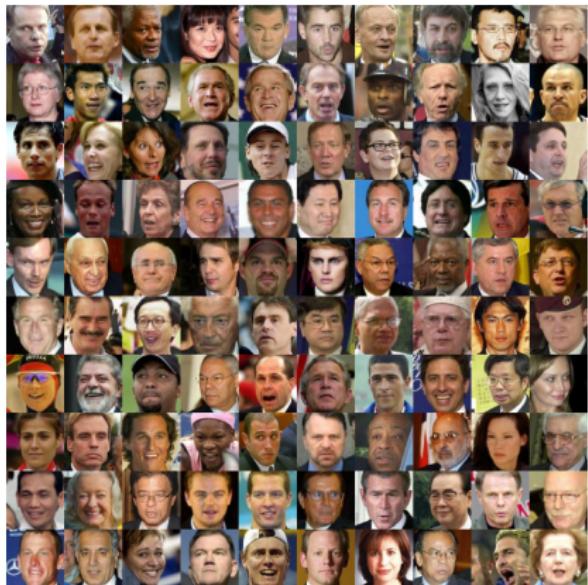


Figure taken from the torch.ch blog

ARCHITECTURE

- ▶ Les GANs utilisent une *architecture duale*: **un générateur (G)** et **un discriminateur (D)**.
- ▶ Le générateur (G) et le discriminateur (D) sont deux réseaux de neurones, e.g. convolutifs (CNNs) (dans le cas d'images).
- ▶ **Le générateur (G)** génère des images à partir d'un dataset.
But: générer des images ressemblantes à celles du dataset.
- ▶ **Le discriminateur (D)** reçoit des images du dataset (vraies) et des images du générateur (fausses).
But: distinguer les vraies images (dataset) des fausses images (générées).

ARCHITECTURE

- ▶ Les GANs utilisent une *architecture duale*: un **générateur (G)** et un **discriminateur (D)**.
- ▶ Le générateur (G) et le discriminateur (D) sont deux réseaux de neurones, e.g. convolutifs (CNNs) (dans le cas d'images).
- ▶ Le générateur (G) génère des images à partir d'un dataset.
But: générer des images ressemblantes à celles du dataset.
- ▶ Le discriminateur (D) reçoit des images du dataset (vraies) et des images du générateur (fausses).
But: distinguer les vraies images (dataset) des fausses images (générées).

ARCHITECTURE

- ▶ Les GANs utilisent une *architecture duale*: un **générateur (G)** et un **discriminateur (D)**.
- ▶ Le générateur (G) et le discriminateur (D) sont deux réseaux de neurones, e.g. convolutifs (CNNs) (dans le cas d'images).
- ▶ **Le générateur (G)** génère des images à partir d'un dataset.

But: générer des images ressemblantes à celles du dataset.

- ▶ **Le discriminateur (D)** reçoit des images du dataset (vraies) et des images du générateur (fausses).

But: distinguer les vraies images (dataset) des fausses images (générées).

ARCHITECTURE

- ▶ Les GANs utilisent une *architecture duale*: un **générateur (G)** et un **discriminateur (D)**.
- ▶ Le générateur (G) et le discriminateur (D) sont deux réseaux de neurones, e.g. convolutifs (CNNs) (dans le cas d'images).
- ▶ **Le générateur (G)** génère des images à partir d'un dataset.

But: générer des images ressemblantes à celles du dataset.

- ▶ **Le discriminateur (D)** reçoit des images du dataset (vraies) et des images du générateur (fausses).

But: distinguer les vraies images (dataset) des fausses images (générées).

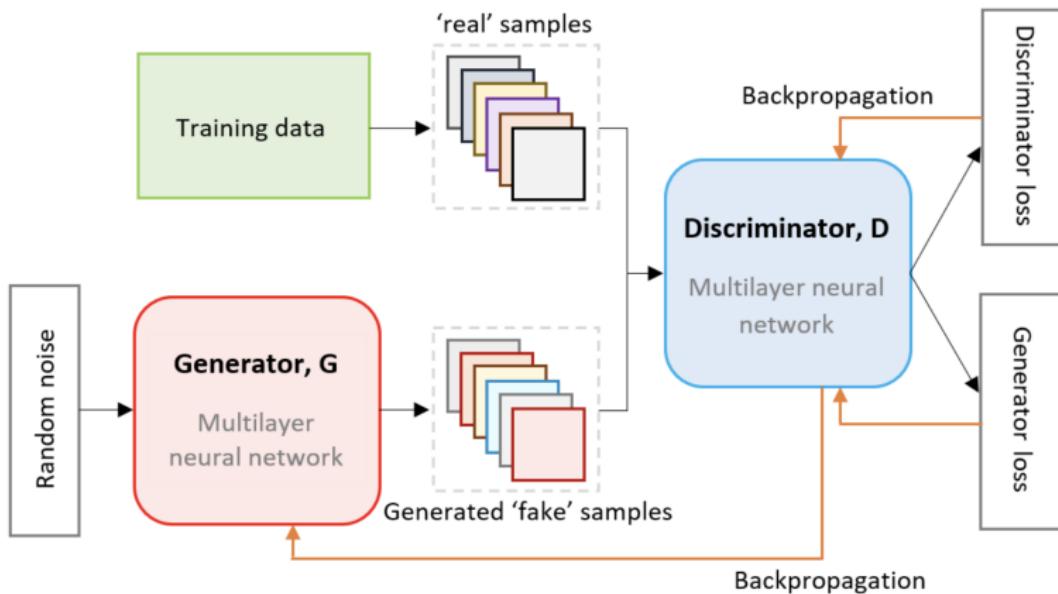
ARCHITECTURE

- ▶ Les GANs utilisent une *architecture duale*: un **générateur (G)** et un **discriminateur (D)**.
- ▶ Le générateur (G) et le discriminateur (D) sont deux réseaux de neurones, e.g. convolutifs (CNNs) (dans le cas d'images).
- ▶ **Le générateur (G)** génère des images à partir d'un dataset.
But: générer des images ressemblantes à celles du dataset.
- ▶ **Le discriminateur (D)** reçoit des images du dataset (vraies) et des images du générateur (fausses).
But: distinguer les vraies images (dataset) des fausses images (générées).

ARCHITECTURE

- ▶ Les GANs utilisent une *architecture duale*: un **générateur (G)** et un **discriminateur (D)**.
- ▶ Le générateur (G) et le discriminateur (D) sont deux réseaux de neurones, e.g. convolutifs (CNNs) (dans le cas d'images).
- ▶ **Le générateur (G)** génère des images à partir d'un dataset.
But: générer des images ressemblantes à celles du dataset.
- ▶ **Le discriminateur (D)** reçoit des images du dataset (vraies) et des images du générateur (fausses).
But: distinguer les vraies images (dataset) des fausses images (générées).

ARCHITECTURE



Figures taken from Little et al. (2021)

ARCHITECTURE

- ▶ Le principe fondamental repose sur une *compétition constructive* ou un *jeux à somme nul* entre le générateur (G) et le discriminateur (D).
 - ▶ G essaie de “tromper” D en générant des images de plus en plus réaliste. Simultanément, D essaie de “contrer” G, en améliorant dans sa capacité à discriminer le vrai du faux. Ceci pousse G à s'améliorer d'autant plus dans sa capacité à générer des images réaliste. Etc.
- ⇒ *L'entraînement concurrent de G et D pousse à produire des data de plus en plus réalistes.*

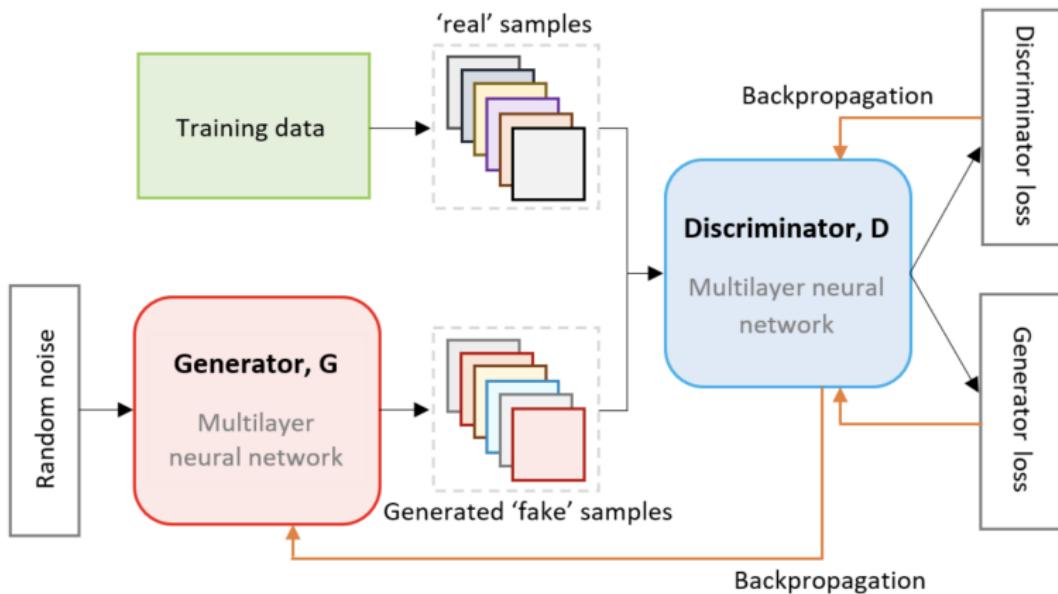
ARCHITECTURE

- ▶ Le principe fondamental repose sur une *compétition constructive* ou un *jeux à somme nul* entre le générateur (G) et le discriminateur (D).
 - ▶ G essaie de “tromper” D en générant des images de plus en plus réaliste. Simultanément, D essaie de “contrer” G, en améliorant dans sa capacité à discriminer le vrai du faux. Ceci pousse G à s'améliorer d'autant plus dans sa capacité à générer des images réaliste. Etc.
- ⇒ *L'entraînement concurrent de G et D pousse à produire des data de plus en plus réalistes.*

ARCHITECTURE

- ▶ Le principe fondamental repose sur une *compétition constructive* ou un *jeux à somme nul* entre le générateur (G) et le discriminateur (D).
- ▶ G essaie de “tromper” D en générant des images de plus en plus réaliste. Simultanément, D essaie de “contrer” G, en améliorant dans sa capacité à discriminer le vrai du faux. Ceci pousse G à s'améliorer d'autant plus dans sa capacité à générer des images réaliste. Etc.
- ⇒ *L'entraînement concurrent de G et D pousse à produire des data de plus en plus réalistes.*

ARCHITECTURE



Figures taken from Little et al. (2021)

GENERATIVE ADVERSARIAL NETWORK (GAN)

- ▶ Soit $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^d : i = 1, \dots, N\}$ un dataset.
- ▶ Le générateur est un deep neural network $G(\cdot; \Phi)$.

G prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une data $G(\mathbf{z}) \in \mathbb{R}^d$

$$G : \mathbf{z} \in \mathbb{R}^l \mapsto G(\mathbf{z}) \in \mathbb{R}^d$$

- ▶ Le discriminateur est un deep neural network $D(\cdot; \Theta)$.
- ▶ D prend une data $\mathbf{x} \in \mathbb{R}^d$ et génère une probabilité $D(\mathbf{x})$ que cette data \mathbf{x} provienne du dataset \mathcal{S}

$$D : \mathbf{x} \in \mathbb{R}^d \mapsto D(\mathbf{x}) = \Pr(y = \text{real} \mid \mathbf{x}) \in [0, 1]$$

$D \circ G$ prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une probabilité $D(G(\mathbf{z}))$ que la data générée $\mathbf{x}' = G(\mathbf{z})$ provienne de \mathcal{S}

$$D \circ G : \mathbf{z} \in \mathbb{R}^l \mapsto D(G(\mathbf{z})) = \Pr(y = \text{real} \mid G(\mathbf{z})) \in [0, 1]$$

GENERATIVE ADVERSARIAL NETWORK (GAN)

- ▶ Soit $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^d : i = 1, \dots, N\}$ un dataset.
- ▶ Le **générateur** est un deep neural network $G(\cdot; \Phi)$.

G prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une data $G(\mathbf{z}) \in \mathbb{R}^d$

$$G : \mathbf{z} \in \mathbb{R}^l \mapsto G(\mathbf{z}) \in \mathbb{R}^d$$

- ▶ Le **discriminateur** est un deep neural network $D(\cdot; \Theta)$.

D prend une data $\mathbf{x} \in \mathbb{R}^d$ et génère une probabilité $D(\mathbf{x})$ que cette data \mathbf{x} provienne du dataset \mathcal{S}

$$D : \mathbf{x} \in \mathbb{R}^d \mapsto D(\mathbf{x}) = \Pr(y = \text{real} \mid \mathbf{x}) \in [0, 1]$$

$D \circ G$ prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une probabilité $D(G(\mathbf{z}))$ que la data générée $\mathbf{x}' = G(\mathbf{z})$ provienne de \mathcal{S}

$$D \circ G : \mathbf{z} \in \mathbb{R}^l \mapsto D(G(\mathbf{z})) = \Pr(y = \text{real} \mid G(\mathbf{z})) \in [0, 1]$$

GENERATIVE ADVERSARIAL NETWORK (GAN)

- ▶ Soit $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^d : i = 1, \dots, N\}$ un dataset.
- ▶ Le **générateur** est un deep neural network $G(\cdot; \Phi)$.

G prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une data $G(\mathbf{z}) \in \mathbb{R}^d$

$$G : \mathbf{z} \in \mathbb{R}^l \longmapsto G(\mathbf{z}) \in \mathbb{R}^d$$

- ▶ Le **discriminateur** est un deep neural network $D(\cdot; \Theta)$.
- D prend une data $\mathbf{x} \in \mathbb{R}^d$ et génère une probabilité $D(\mathbf{x})$ que cette data \mathbf{x} provienne du dataset \mathcal{S}

$$D : \mathbf{x} \in \mathbb{R}^d \longmapsto D(\mathbf{x}) = \Pr(y = \text{real} \mid \mathbf{x}) \in [0, 1]$$

$D \circ G$ prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une probabilité $D(G(\mathbf{z}))$ que la data générée $\mathbf{x}' = G(\mathbf{z})$ provienne de \mathcal{S}

$$D \circ G : \mathbf{z} \in \mathbb{R}^l \longmapsto D(G(\mathbf{z})) = \Pr(y = \text{real} \mid G(\mathbf{z})) \in [0, 1]$$

GENERATIVE ADVERSARIAL NETWORK (GAN)

- ▶ Soit $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^d : i = 1, \dots, N\}$ un dataset.
- ▶ Le **générateur** est un deep neural network $G(\cdot; \Phi)$.

G prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une data $G(\mathbf{z}) \in \mathbb{R}^d$

$$G : \mathbf{z} \in \mathbb{R}^l \longmapsto G(\mathbf{z}) \in \mathbb{R}^d$$

- ▶ Le **discriminateur** est un deep neural network $D(\cdot; \Theta)$.

D prend une data $\mathbf{x} \in \mathbb{R}^d$ et génère une probabilité $D(\mathbf{x})$ que cette data \mathbf{x} provienne du dataset \mathcal{S}

$$D : \mathbf{x} \in \mathbb{R}^d \longmapsto D(\mathbf{x}) = \Pr(y = \text{real} \mid \mathbf{x}) \in [0, 1]$$

$D \circ G$ prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une probabilité $D(G(\mathbf{z}))$ que la data générée $\mathbf{x}' = G(\mathbf{z})$ provienne de \mathcal{S}

$$D \circ G : \mathbf{z} \in \mathbb{R}^l \longmapsto D(G(\mathbf{z})) = \Pr(y = \text{real} \mid G(\mathbf{z})) \in [0, 1]$$

GENERATIVE ADVERSARIAL NETWORK (GAN)

- ▶ Soit $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^d : i = 1, \dots, N\}$ un dataset.
- ▶ Le **générateur** est un deep neural network $G(\cdot; \Phi)$.

G prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une data $G(\mathbf{z}) \in \mathbb{R}^d$

$$G : \mathbf{z} \in \mathbb{R}^l \longmapsto G(\mathbf{z}) \in \mathbb{R}^d$$

- ▶ Le **discriminateur** est un deep neural network $D(\cdot; \Theta)$.
- D prend une data $\mathbf{x} \in \mathbb{R}^d$ et génère une probabilité $D(\mathbf{x})$ que cette data \mathbf{x} provienne du dataset \mathcal{S}

$$D : \mathbf{x} \in \mathbb{R}^d \longmapsto D(\mathbf{x}) = \Pr(y = \text{real} \mid \mathbf{x}) \in [0, 1]$$

$D \circ G$ prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une probabilité $D(G(\mathbf{z}))$ que la data générée $\mathbf{x}' = G(\mathbf{z})$ provienne de \mathcal{S}

$$D \circ G : \mathbf{z} \in \mathbb{R}^l \longmapsto D(G(\mathbf{z})) = \Pr(y = \text{real} \mid G(\mathbf{z})) \in [0, 1]$$

GENERATIVE ADVERSARIAL NETWORK (GAN)

- ▶ Soit $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^d : i = 1, \dots, N\}$ un dataset.
- ▶ Le **générateur** est un deep neural network $G(\cdot; \Phi)$.

G prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une data $G(\mathbf{z}) \in \mathbb{R}^d$

$$G : \mathbf{z} \in \mathbb{R}^l \longmapsto G(\mathbf{z}) \in \mathbb{R}^d$$

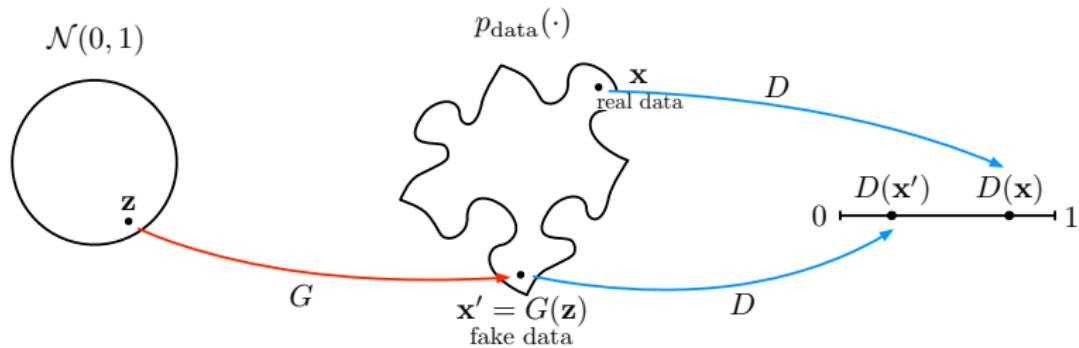
- ▶ Le **discriminateur** est un deep neural network $D(\cdot; \Theta)$.
- D prend une data $\mathbf{x} \in \mathbb{R}^d$ et génère une probabilité $D(\mathbf{x})$ que cette data \mathbf{x} provienne du dataset \mathcal{S}

$$D : \mathbf{x} \in \mathbb{R}^d \longmapsto D(\mathbf{x}) = \Pr(y = \text{real} \mid \mathbf{x}) \in [0, 1]$$

$D \circ G$ prend un point latent $\mathbf{z} \in \mathbb{R}^l$ et génère une probabilité $D(G(\mathbf{z}))$ que la data générée $\mathbf{x}' = G(\mathbf{z})$ provienne de \mathcal{S}

$$D \circ G : \mathbf{z} \in \mathbb{R}^l \longmapsto D(G(\mathbf{z})) = \Pr(y = \text{real} \mid G(\mathbf{z})) \in [0, 1]$$

GENERATIVE ADVERSARIAL NETWORK (GAN)



DISTRIBUTIONS

- ▶ $p_{\text{data}}(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution empirique des data.
 - ▶ $p_z(\mathbf{z}) \in \Delta(\mathbb{R}^l)$: distribution des variables latentes.
En pratique, on choisit souvent $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.
 - ▶ $p_g(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution induite (image directe de p_z) par le générateur $G : \mathbb{R}^l \rightarrow \mathbb{R}^d$, définie par (cas G bij. et diff.)

DISTRIBUTIONS

- ▶ $p_{\text{data}}(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution empirique des data.
 - ▶ $p_z(\mathbf{z}) \in \Delta(\mathbb{R}^l)$: distribution des variables latentes.
En pratique, on choisit souvent $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.
 - ▶ $p_g(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution induite (image directe de p_z) par le générateur $G : \mathbb{R}^l \rightarrow \mathbb{R}^d$, définie par (cas G bij. et diff.)

DISTRIBUTIONS

- ▶ $p_{\text{data}}(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution empirique des data.
 - ▶ $p_z(\mathbf{z}) \in \Delta(\mathbb{R}^l)$: distribution des variables latentes.
En pratique, on choisit souvent $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.
 - ▶ $p_g(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution induite (image directe de p_z) par le générateur $G : \mathbb{R}^l \rightarrow \mathbb{R}^d$, définie par (cas G bij. et diff.)

$$p_g(\mathbf{x}) := p_z(G^{-1}(\mathbf{x}))$$

DISTRIBUTIONS

- ▶ $p_{\text{data}}(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution empirique des data.
 - ▶ $p_z(\mathbf{z}) \in \Delta(\mathbb{R}^l)$: distribution des variables latentes.
En pratique, on choisit souvent $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.
 - ▶ $p_g(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution induite (image directe de p_z) par le générateur $G : \mathbb{R}^l \rightarrow \mathbb{R}^d$, définie par (cas G bij. et diff.)

$$p_g(\mathbf{x}) := p_z(G^{-1}(\mathbf{x}))$$

- ▶ Le GAN est performant dans la situation d'équilibre où:
 - 1) le générateur induit une distribution des data p_g proche de p_{data}
 - 2) le discriminateur devient aussi mauvais que le hasard

DISTRIBUTIONS

- ▶ $p_{\text{data}}(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution empirique des data.
- ▶ $p_z(\mathbf{z}) \in \Delta(\mathbb{R}^l)$: distribution des variables latentes.
En pratique, on choisit souvent $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.
- ▶ $p_g(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution induite (image directe de p_z) par le générateur $G : \mathbb{R}^l \rightarrow \mathbb{R}^d$, définie par (cas G bij. et diff.)

$$p_g(\mathbf{x}) := p_z(G^{-1}(\mathbf{x}))$$

- ▶ Le GAN est performant dans la situation d'équilibre où:
 - (1) le générateur induit une distribution des data p_g proche de p_{data}
 - (2) le discriminateur devient aussi mauvais que le hasard

$$p_g(\mathbf{x}) \simeq p_{\text{data}}(\mathbf{x}) \quad \text{et} \quad D(\mathbf{x}) \simeq \frac{1}{2}, \quad \forall \mathbf{x} \in \mathbb{R}^d.$$

DISTRIBUTIONS

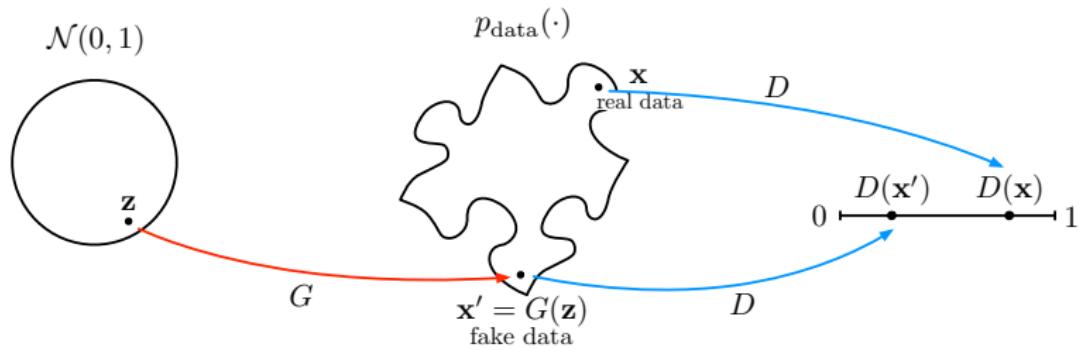
- ▶ $p_{\text{data}}(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution empirique des data.
 - ▶ $p_z(\mathbf{z}) \in \Delta(\mathbb{R}^l)$: distribution des variables latentes.
En pratique, on choisit souvent $p_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.
 - ▶ $p_g(\mathbf{x}) \in \Delta(\mathbb{R}^d)$: distribution induite (image directe de p_z) par le générateur $G : \mathbb{R}^l \rightarrow \mathbb{R}^d$, définie par (cas G bij. et diff.)

$$p_g(\mathbf{x}) := p_z(G^{-1}(\mathbf{x}))$$

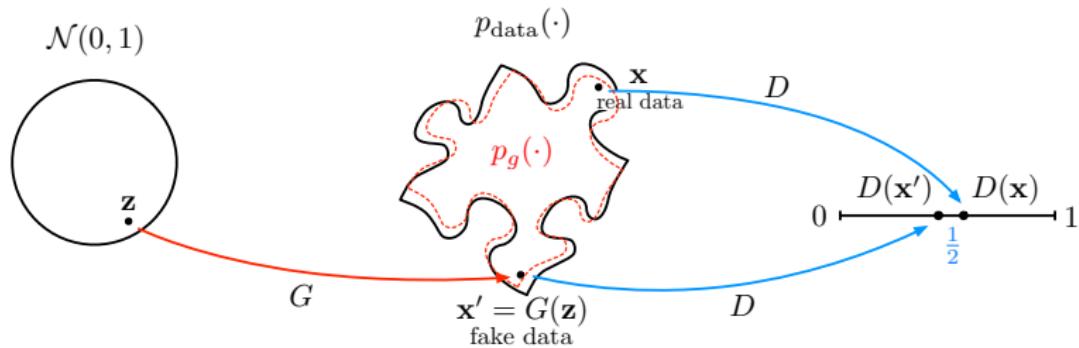
- Le GAN est performant dans la situation d'équilibre où:
 - 1) le générateur induit une distribution des data p_g proche de p_{data}
 - 2) le discriminateur devient aussi mauvais que le hasard

$$p_g(\mathbf{x}) \simeq p_{\text{data}}(\mathbf{x}) \quad \text{et} \quad D(\mathbf{x}) \simeq \frac{1}{2}, \quad \forall \mathbf{x} \in \mathbb{R}^d.$$

GENERATIVE ADVERSARIAL NETWORK (GAN)



GENERATIVE ADVERSARIAL NETWORK (GAN)



FONCTION DE LOSS

- ▶ Le discriminateur $D : \mathbb{R}^d \rightarrow [0, 1]$ est un *classifieur binaire*.
- ▶ D produit une prédiction $\hat{y} = D(\mathbf{x}) = \Pr(y = 1 \mid \mathbf{x}) \in [0, 1]$.
- ▶ On veut que D prédise correctement les fake ($y = 0$) et real ($y = 1$) data, i.e.:

$$\begin{aligned} \mathbf{x} \text{ real} &\Leftrightarrow \mathbf{x} \sim p_{\text{data}} &\Leftrightarrow y = 1 &\Rightarrow \hat{y} = D(\mathbf{x}) \simeq 1 \\ \mathbf{x} \text{ fake} &\Leftrightarrow \mathbf{x} \sim p_g &\Leftrightarrow y = 0 &\Rightarrow \hat{y} = D(\mathbf{x}) \simeq 0 \end{aligned}$$

FONCTION DE LOSS

- ▶ Le discriminateur $D : \mathbb{R}^d \rightarrow [0, 1]$ est un *classifieur binaire*.
- ▶ D produit une prédiction $\hat{y} = D(\mathbf{x}) = \Pr(y = 1 \mid \mathbf{x}) \in [0, 1]$.
- ▶ On veut que D prédise correctement les fake ($y = 0$) et real ($y = 1$) data, i.e.:

$$\begin{aligned} \mathbf{x} \text{ real} &\Leftrightarrow \mathbf{x} \sim p_{\text{data}} &\Leftrightarrow y = 1 &\Rightarrow \hat{y} = D(\mathbf{x}) \simeq 1 \\ \mathbf{x} \text{ fake} &\Leftrightarrow \mathbf{x} \sim p_g &\Leftrightarrow y = 0 &\Rightarrow \hat{y} = D(\mathbf{x}) \simeq 0 \end{aligned}$$

FONCTION DE LOSS

- ▶ Le discriminateur $D : \mathbb{R}^d \rightarrow [0, 1]$ est un *classifieur binaire*.
- ▶ D produit une prédiction $\hat{y} = D(\mathbf{x}) = \Pr(y = 1 \mid \mathbf{x}) \in [0, 1]$.
- ▶ On veut que D prédise correctement les fake ($y = 0$) et real ($y = 1$) data, i.e.:

$$\begin{array}{llllll} \mathbf{x} \text{ real} & \Leftrightarrow & \mathbf{x} \sim p_{\text{data}} & \Leftrightarrow & y = 1 & \Rightarrow \hat{y} = D(\mathbf{x}) \simeq 1 \\ \mathbf{x} \text{ fake} & \Leftrightarrow & \mathbf{x} \sim p_g & \Leftrightarrow & y = 0 & \Rightarrow \hat{y} = D(\mathbf{x}) \simeq 0 \end{array}$$

FONCTION DE LOSS

- ▶ Pour cela, on maximise la “vraisemblance populationnelle” suivante (abstraction du cas discret, pas rigoureusement définie):

$$\begin{aligned}\mathcal{L}(D) &:= \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=1\}} \Pr(y=1 \mid \mathbf{x})^{p_{\text{data}}(\mathbf{x})} \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=0\}} \Pr(y=0 \mid \mathbf{x})^{p_g(\mathbf{x})} \\ &:= \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=1\}} D(\mathbf{x})^{p_{\text{data}}(\mathbf{x})} \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=0\}} (1 - D(\mathbf{x}))^{p_g(\mathbf{x})}\end{aligned}$$

- ▶ Plus les $\hat{y} = D(\mathbf{x})$ sont proches des y correspondants, plus la vraisemblance $\mathcal{L}(D)$ est élevée.

FONCTION DE LOSS

- ▶ Pour cela, on maximise la “vraisemblance populationnelle” suivante (abstraction du cas discret, pas rigoureusement définie):

$$\begin{aligned}\mathcal{L}(D) &:= \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=1\}} \Pr(y=1 \mid \mathbf{x})^{p_{\text{data}}(\mathbf{x})} \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=0\}} \Pr(y=0 \mid \mathbf{x})^{p_g(\mathbf{x})} \\ &:= \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=1\}} D(\mathbf{x})^{p_{\text{data}}(\mathbf{x})} \prod_{\{\mathbf{x} \in \mathbb{R}^d : y=0\}} (1 - D(\mathbf{x}))^{p_g(\mathbf{x})}\end{aligned}$$

- ▶ Plus les $\hat{y} = D(\mathbf{x})$ sont proches des y correspondants, plus la vraisemblance $\mathcal{L}(D)$ est élevée.

FONCTION DE LOSS

- ▶ La formulation rigoureuse, dans le cas continu, s'obtient par passage au log:

$$\log \mathcal{L}(D) = \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log (1 - D(\mathbf{x})) d\mathbf{x}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log (1 - D(\mathbf{x}))]$$

$$\mathbf{x} \sim p_g \text{ssi} \quad \begin{matrix} \exists \mathbf{z} \sim p_z \\ \text{s.t. } \mathbf{x} \sim p_{\mathbf{z}} \end{matrix} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

On peut alors écrire la fonction de perte comme :

$\mathcal{L}(D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$

FONCTION DE LOSS

- ▶ La formulation rigoureuse, dans le cas continu, s'obtient par passage au log:

$$\log \mathcal{L}(D) = \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log (1 - D(\mathbf{x})) d\mathbf{x}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log (1 - D(\mathbf{x}))]$$

$$\mathbf{x} \sim p_g \text{ssi } \begin{matrix} \exists \mathbf{z} \sim p_z \\ \text{s.t. } \mathbf{x} \sim p_z \end{matrix} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

La fonction de loss est alors la somme des deux termes ci-dessus. Celle-ci est aussi connue sous le nom de **cross-entropy loss**.

FONCTION DE LOSS

- La formulation rigoureuse, dans le cas continu, s'obtient par passage au log:

$$\log \mathcal{L}(D) = \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log (1 - D(\mathbf{x})) d\mathbf{x}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log (1 - D(\mathbf{x}))]$$

$$\mathbf{x} \sim p_g \text{ssi} \quad \exists \mathbf{z} \sim p_z \quad \text{s.t.} \quad \mathbf{z} \sim p_z = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

- Ceci correspond à la fonction de loss $V(G, D)$ que G et D cherchent à *minimiser* et *maximiser*, respectivement:

$$G, D = \arg \min_{G} \max_{D} V(D, G)$$

$$= \arg \min_{G} \max_{D} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

FONCTION DE LOSS

- ▶ La formulation rigoureuse, dans le cas continu, s'obtient par passage au log:

$$\log \mathcal{L}(D) = \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log (1 - D(\mathbf{x})) d\mathbf{x}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log (1 - D(\mathbf{x}))]$$

$$\mathbf{x} \sim p_g \text{ ssi } \begin{array}{l} \exists \mathbf{z} \sim p_z \\ \text{s.t. } \mathbf{x} \sim p_z \end{array} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

- ▶ Ceci correspond à la fonction de loss $V(G, D)$ que G et D cherchent à minimiser et maximiser, respectivement:

$$G, D = \arg \min_{G, D} V(G, D)$$

$$= \arg \min_{G, D} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

FONCTION DE LOSS

- ▶ La formulation rigoureuse, dans le cas continu, s'obtient par passage au log:

$$\log \mathcal{L}(D) = \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log (1 - D(\mathbf{x})) d\mathbf{x}$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log (1 - D(\mathbf{x}))]$$

$$\mathbf{x} \sim p_g \text{ ssi } \begin{array}{l} \exists \mathbf{z} \sim p_z \\ \text{s.t. } \mathbf{z} \sim p_z \end{array} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

- ▶ Ceci correspond à la **fonction de loss** $V(G, D)$ que G et D cherchent à *minimiser* et *maximiser*, respectivement:

$$\hat{G}, \hat{D} = \arg \min_G \max_D V(D, G)$$

$$= \arg \min_G \max_D \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))]$$

FONCTION DE LOSS

- ▶ On a donc la **fonction de loss** $V(G, D)$:

$$V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))] \quad (1)$$

- ▶ Pour un batch de data $\{\mathbf{x}_i\}_{i=1}^m$ et $\{\mathbf{z}_j\}_{j=1}^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, on a l'*estimateur empirique* suivant:

$$\hat{V}(D, G) = \frac{1}{m} \sum_{i=1}^m \log D(\mathbf{x}_i) + \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j))) \quad (2)$$

FONCTION DE LOSS

- ▶ On a donc la **fonction de loss** $V(G, D)$:

$$V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))] \quad (1)$$

- ▶ Pour un batch de data $\{\mathbf{x}_i\}_{i=1}^m$ et $\{\mathbf{z}_j\}_{j=1}^m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, on a l'*estimateur empirique* suivant:

$$\hat{V}(D, G) = \frac{1}{m} \sum_{i=1}^m \log D(\mathbf{x}_i) + \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j))) \quad (2)$$

FONCTION DE LOSS

- ▶ En pratique, le **discriminateur D** maximise la loss (2), et donc il minimise:

$$L_D = -\frac{1}{m} \sum_{i=1}^m \log D(\mathbf{x}_i) - \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j)))$$

- ▶ Le **Générateur G** minimise sa partie de la loss (2):

$$L_G = \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j)))$$

En général, on utilise la forme non-saturante plus stable:

$$L_G^{\text{NS}} = -\frac{1}{m} \sum_{j=1}^m \log D(G(\mathbf{z}_j))$$

FONCTION DE LOSS

- ▶ En pratique, le **discriminateur D** maximise la loss (2), et donc il minimise:

$$L_D = -\frac{1}{m} \sum_{i=1}^m \log D(\mathbf{x}_i) - \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j)))$$

- ▶ Le **Générateur G** minimise sa partie de la loss (2):

$$L_G = \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j)))$$

En général, on utilise la forme non-saturante plus stable:

$$L_G^{\text{NS}} = -\frac{1}{m} \sum_{j=1}^m \log D(G(\mathbf{z}_j))$$

FONCTION DE LOSS

- ▶ En pratique, le **discriminateur D** maximise la loss (2), et donc il minimise:

$$L_D = -\frac{1}{m} \sum_{i=1}^m \log D(\mathbf{x}_i) - \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j)))$$

- ▶ Le **Générateur G** minimise sa partie de la loss (2):

$$L_G = \frac{1}{m} \sum_{j=1}^m \log (1 - D(G(\mathbf{z}_j)))$$

En général, on utilise la forme non-saturante plus stable:

$$L_G^{\text{NS}} = -\frac{1}{m} \sum_{j=1}^m \log D(G(\mathbf{z}_j))$$

DIVERGENCES ENTRE DISTRIBUTIONS

- ▶ **Kullback-Leibler divergence (KL)** : pour deux distributions p et q sur \mathcal{X} :

$$D_{\text{KL}}(p \parallel q) := \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx$$

- ▶ Non symétrique : $D_{\text{KL}}(p \parallel q) \neq D_{\text{KL}}(q \parallel p)$
- ▶ **Jensen-Shannon divergence (JS)** : basée sur KL

$$D_{\text{JS}}(p \parallel q) := \frac{1}{2} D_{\text{KL}}\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2} D_{\text{KL}}\left(q \parallel \frac{p+q}{2}\right)$$

- ▶ Symétrique : $D_{\text{JS}}(p \parallel q) = D_{\text{JS}}(q \parallel p) \in [0, \log 2]$

DIVERGENCES ENTRE DISTRIBUTIONS

- ▶ **Kullback-Leibler divergence (KL)** : pour deux distributions p et q sur \mathcal{X} :

$$D_{\text{KL}}(p \parallel q) := \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx$$

- ▶ Non symétrique : $D_{\text{KL}}(p \parallel q) \neq D_{\text{KL}}(q \parallel p)$
- ▶ **Jensen-Shannon divergence (JS)** : basée sur KL

$$D_{\text{JS}}(p \parallel q) := \frac{1}{2}D_{\text{KL}}\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2}D_{\text{KL}}\left(q \parallel \frac{p+q}{2}\right)$$

- ▶ Symétrique : $D_{\text{JS}}(p \parallel q) = D_{\text{JS}}(q \parallel p) \in [0, \log 2]$

DIVERGENCES ENTRE DISTRIBUTIONS

- ▶ **Kullback-Leibler divergence (KL)** : pour deux distributions p et q sur \mathcal{X} :

$$D_{\text{KL}}(p \parallel q) := \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx$$

- ▶ Non symétrique : $D_{\text{KL}}(p \parallel q) \neq D_{\text{KL}}(q \parallel p)$
- ▶ **Jensen-Shannon divergence (JS)** : basée sur KL

$$D_{\text{JS}}(p \parallel q) := \frac{1}{2} D_{\text{KL}}\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2} D_{\text{KL}}\left(q \parallel \frac{p+q}{2}\right)$$

- ▶ Symétrique : $D_{\text{JS}}(p \parallel q) = D_{\text{JS}}(q \parallel p) \in [0, \log 2]$

DIVERGENCES ENTRE DISTRIBUTIONS

- ▶ **Kullback-Leibler divergence (KL)** : pour deux distributions p et q sur \mathcal{X} :

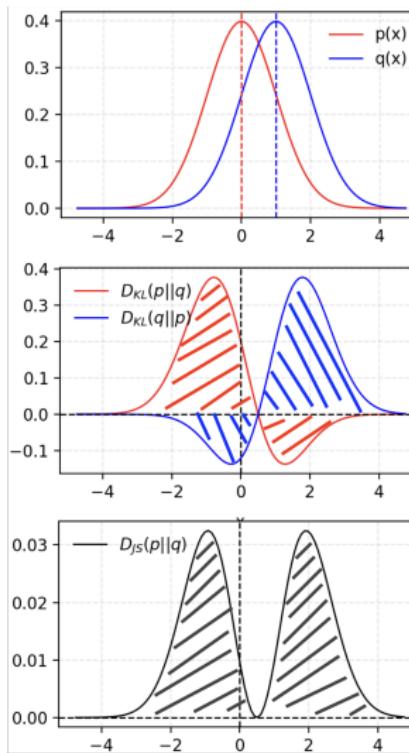
$$D_{\text{KL}}(p \parallel q) := \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx$$

- ▶ Non symétrique : $D_{\text{KL}}(p \parallel q) \neq D_{\text{KL}}(q \parallel p)$
- ▶ **Jensen-Shannon divergence (JS)** : basée sur KL

$$D_{\text{JS}}(p \parallel q) := \frac{1}{2} D_{\text{KL}}\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2} D_{\text{KL}}\left(q \parallel \frac{p+q}{2}\right)$$

- ▶ Symétrique : $D_{\text{JS}}(p \parallel q) = D_{\text{JS}}(q \parallel p) \in [0, \log 2]$

JENSEN-SHANNON DIVERGENCE



OPTIMUM DU DISCRIMINATEUR

- ▶ Pour un générateur G fixé, on peut chercher le discriminateur D^* qui *maximise* la fonction de loss (1) :

$$\begin{aligned} V(D, G) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log(1 - D(\mathbf{x}))] \\ &= \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \\ &= \int_{\mathbb{R}^d} \left(p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) \right) d\mathbf{x} \end{aligned}$$

- ▶ Pour une valeur \mathbf{x} fixée, on cherche donc à maximiser :

$$f(D(\mathbf{x})) := p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x}))$$

- ▶ On dérive par rapport à $D(\mathbf{x})$ et on annule la dérivée :

$$f'(D(\mathbf{x})) = \frac{p_{\text{data}}(\mathbf{x})}{D(\mathbf{x})} - \frac{p_g(\mathbf{x})}{1 - D(\mathbf{x})} = 0 \quad \Rightarrow \quad D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$$

OPTIMUM DU DISCRIMINATEUR

- ▶ Pour un générateur G fixé, on peut chercher le discriminateur D^* qui *maximise* la fonction de loss (1) :

$$\begin{aligned} V(D, G) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log(1 - D(\mathbf{x}))] \\ &= \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \\ &= \int_{\mathbb{R}^d} \left(p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) \right) d\mathbf{x} \end{aligned}$$

- ▶ Pour une valeur \mathbf{x} fixée, on cherche donc à maximiser :

$$f(D(\mathbf{x})) := p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x}))$$

- ▶ On dérive par rapport à $D(\mathbf{x})$ et on annule la dérivée :

$$f'(D(\mathbf{x})) = \frac{p_{\text{data}}(\mathbf{x})}{D(\mathbf{x})} - \frac{p_g(\mathbf{x})}{1 - D(\mathbf{x})} = 0 \quad \Rightarrow \quad D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$$

OPTIMUM DU DISCRIMINATEUR

- ▶ Pour un générateur G fixé, on peut chercher le discriminateur D^* qui *maximise* la fonction de loss (1) :

$$\begin{aligned} V(D, G) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log(1 - D(\mathbf{x}))] \\ &= \int_{\mathbb{R}^d} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}^d} p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \\ &= \int_{\mathbb{R}^d} \left(p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) \right) d\mathbf{x} \end{aligned}$$

- ▶ Pour une valeur \mathbf{x} fixée, on cherche donc à maximiser :

$$f(D(\mathbf{x})) := p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x}))$$

- ▶ On dérive par rapport à $D(\mathbf{x})$ et on annule la dérivée :

$$f'(D(\mathbf{x})) = \frac{p_{\text{data}}(\mathbf{x})}{D(\mathbf{x})} - \frac{p_g(\mathbf{x})}{1 - D(\mathbf{x})} = 0 \quad \Rightarrow \quad D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$$

SUBSTITUTION DE D^* DANS $V(D, G)$

- En remplaçant $D^*(\mathbf{x})$ dans $V(D, G)$, on obtient :

$$V(D^*, G) = \int_{\mathbb{R}^d} \left[p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} + p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] d\mathbf{x}$$

- En posant $M = \frac{1}{2}(p_{\text{data}} + p_g)$, la moyenne des distributions, on peut réécrire (après développement) :

$$V(D^*, G) = -2 \log 2 + 2 D_{\text{JS}}(p_{\text{data}} \| p_g)$$

SUBSTITUTION DE D^* DANS $V(D, G)$

- En remplaçant $D^*(\mathbf{x})$ dans $V(D, G)$, on obtient :

$$V(D^*, G) = \int_{\mathbb{R}^d} \left[p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} + p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \right] d\mathbf{x}$$

- En posant $M = \frac{1}{2}(p_{\text{data}} + p_g)$, la moyenne des distributions, on peut réécrire (après développement) :

$$V(D^*, G) = -2 \log 2 + 2 D_{\text{JS}}(p_{\text{data}} \| p_g)$$

INTERPRÉTATION DE LA FONCTION DE LOSS

- ▶ Ainsi, entraîner un GAN revient à minimiser la divergence de Jensen–Shannon entre p_{data} et p_g .
- ▶ À l'équilibre, on a $p_g = p_{\text{data}}$ (pour minimiser la JS div.) et $D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} = \frac{1}{2}$ (fct cste, pour maximiser la loss).
- ▶ Ceci qui correspond aux conditions (C1) et (C2) mentionnées précédemment.
- ▶ Ceci est un *équilibre de Nash* du jeu min–max : ni D ni G ne peuvent améliorer leur performance sans détériorer celle de l'autre.

INTERPRÉTATION DE LA FONCTION DE LOSS

- ▶ Ainsi, entraîner un GAN revient à minimiser la divergence de Jensen–Shannon entre p_{data} et p_g .
- ▶ À l'équilibre, on a $p_g = p_{\text{data}}$ (pour minimiser la JS div.) et $D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} = \frac{1}{2}$ (fct cste, pour maximiser la loss).
- ▶ Ceci qui correspond aux conditions (C1) et (C2) mentionnées précédemment.
- ▶ Ceci est un *équilibre de Nash* du jeu min–max : ni D ni G ne peuvent améliorer leur performance sans détériorer celle de l'autre.

INTERPRÉTATION DE LA FONCTION DE LOSS

- ▶ Ainsi, entraîner un GAN revient à minimiser la divergence de Jensen–Shannon entre p_{data} et p_g .
- ▶ À l'équilibre, on a $p_g = p_{\text{data}}$ (pour minimiser la JS div.) et $D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} = \frac{1}{2}$ (fct cste, pour maximiser la loss).
- ▶ Ceci qui correspond aux conditions (C1) et (C2) mentionnées précédemment.
- ▶ Ceci est un *équilibre de Nash* du jeu min–max : ni D ni G ne peuvent améliorer leur performance sans détériorer celle de l'autre.

INTERPRÉTATION DE LA FONCTION DE LOSS

- ▶ Ainsi, entraîner un GAN revient à minimiser la divergence de Jensen–Shannon entre p_{data} et p_g .
- ▶ À l'équilibre, on a $p_g = p_{\text{data}}$ (pour minimiser la JS div.) et $D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} = \frac{1}{2}$ (fct cste, pour maximiser la loss).
- ▶ Ceci qui correspond aux conditions (C1) et (C2) mentionnées précédemment.
- ▶ Ceci est un *équilibre de Nash* du jeu min–max : ni D ni G ne peuvent améliorer leur performance sans détériorer celle de l'autre.

ENTRAÎNEMENT ALTERNÉ

- ▶ L'entraînement d'un GAN se fait par alternance :

- ▷ Mise à jour de D (quelques itérations) :

$$\theta_D \leftarrow \theta_D + \eta \nabla_{\theta_D} V(D, G).$$

- ▷ Puis mise à jour de G :

$$\theta_G \leftarrow \theta_G - \eta \nabla_{\theta_G} V(D, G).$$

- ▶ En pratique, il faut maintenir un *équilibre* entre les deux joueurs : si l'un progresse trop vite, l'autre n'apprend plus.

ENTRAÎNEMENT ALTERNÉ

- ▶ L'entraînement d'un GAN se fait par alternance :
 - ▷ Mise à jour de D (quelques itérations) :

$$\theta_D \leftarrow \theta_D + \eta \nabla_{\theta_D} V(D, G).$$

- ▷ Puis mise à jour de G :

$$\theta_G \leftarrow \theta_G - \eta \nabla_{\theta_G} V(D, G).$$

- ▶ En pratique, il faut maintenir un *équilibre* entre les deux joueurs : si l'un progresse trop vite, l'autre n'apprend plus.

ENTRAÎNEMENT ALTERNÉ

- ▶ L'entraînement d'un GAN se fait par alternance :
 - ▷ Mise à jour de D (quelques itérations) :

$$\theta_D \leftarrow \theta_D + \eta \nabla_{\theta_D} V(D, G).$$

- ▷ Puis mise à jour de G :

$$\theta_G \leftarrow \theta_G - \eta \nabla_{\theta_G} V(D, G).$$

- ▶ En pratique, il faut maintenir un *équilibre* entre les deux joueurs : si l'un progresse trop vite, l'autre n'apprend plus.

ENTRAÎNEMENT ALTERNÉ

- ▶ L'entraînement d'un GAN se fait par alternance :
 - ▷ Mise à jour de D (quelques itérations) :

$$\theta_D \leftarrow \theta_D + \eta \nabla_{\theta_D} V(D, G).$$

- ▷ Puis mise à jour de G :

$$\theta_G \leftarrow \theta_G - \eta \nabla_{\theta_G} V(D, G).$$

- ▶ En pratique, il faut maintenir un *équilibre* entre les deux joueurs : si l'un progresse trop vite, l'autre n'apprend plus.

PROBLÈMES D'ENTRAÎNEMENT DES GANs

- ▶ Formation lente et instable.
- ▶ Difficile d'atteindre un équilibre de Nash.
- ▶ Supports des distributions p_{data} et p_g de faible dimension \Rightarrow gradients très faibles (vanishing gradients).
- ▶ Mode collapse : génère toujours les mêmes exemples !
- ▶ Absence de métrique d'évaluation fiable.

PROBLÈMES D'ENTRAÎNEMENT DES GANs

- ▶ Formation lente et instable.
- ▶ Difficile d'atteindre un équilibre de Nash.
- ▶ Supports des distributions p_{data} et p_g de faible dimension \Rightarrow gradients très faibles (vanishing gradients).
- ▶ Mode collapse : génère toujours les mêmes exemples !
- ▶ Absence de métrique d'évaluation fiable.

PROBLÈMES D'ENTRAÎNEMENT DES GANs

- ▶ Formation lente et instable.
- ▶ Difficile d'atteindre un équilibre de Nash.
- ▶ Supports des distributions p_{data} et p_g de faible dimension \Rightarrow gradients très faibles (vanishing gradients).
- ▶ Mode collapse : génère toujours les mêmes exemples !
- ▶ Absence de métrique d'évaluation fiable.

PROBLÈMES D'ENTRAÎNEMENT DES GANs

- ▶ Formation lente et instable.
- ▶ Difficile d'atteindre un équilibre de Nash.
- ▶ Supports des distributions p_{data} et p_g de faible dimension \Rightarrow gradients très faibles (vanishing gradients).
- ▶ Mode collapse : génère toujours les mêmes exemples !
- ▶ Absence de métrique d'évaluation fiable.

PROBLÈMES D'ENTRAÎNEMENT DES GANs

- ▶ Formation lente et instable.
- ▶ Difficile d'atteindre un équilibre de Nash.
- ▶ Supports des distributions p_{data} et p_g de faible dimension \Rightarrow gradients très faibles (vanishing gradients).
- ▶ Mode collapse : génère toujours les mêmes exemples !
- ▶ Absence de métrique d'évaluation fiable.

SOLUTIONS POUR STABILISER LES GANs

- ▶ Feature Matching : aligner les statistiques des data réelles et générées via une loss du type

$$\left\| \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} f(G(\mathbf{z})) \right\|_2^2$$

- ▶ Minibatch Discrimination : une métrique de clusterisation inter-batch est ajoutée en input du modèle...
- ▶ Historical Averaging et Label Smoothing : ajouter un terme de régularisation dans la loss et adoucir les labels (0 ou 1) pour stabiliser l'apprentissage.
- ▶ Virtual Batch Normalization : normalisation des data par rapport à un batch de référence fixe.
- ▶ Ajouter du bruit et utiliser la *distance de Wasserstein (WGAN)* pour éviter les gradients nuls.

SOLUTIONS POUR STABILISER LES GANs

- ▶ Feature Matching : aligner les statistiques des data réelles et générées via une loss du type

$$\left\| \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} f(G(\mathbf{z})) \right\|_2^2$$

- ▶ Minibatch Discrimination : une métrique de clusterisation inter-batch est ajoutée en input du modèle...
- ▶ Historical Averaging et Label Smoothing : ajouter un terme de régularisation dans la loss et adoucir les labels (0 ou 1) pour stabiliser l'apprentissage.
- ▶ Virtual Batch Normalization : normalisation des data par rapport à un batch de référence fixe.
- ▶ Ajouter du bruit et utiliser la *distance de Wasserstein (WGAN)* pour éviter les gradients nuls.

SOLUTIONS POUR STABILISER LES GANs

- ▶ Feature Matching : aligner les statistiques des data réelles et générées via une loss du type

$$\left\| \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} f(G(\mathbf{z})) \right\|_2^2$$

- ▶ Minibatch Discrimination : une métrique de clusterisation inter-batch est ajoutée en input du modèle...
- ▶ Historical Averaging et Label Smoothing : ajouter un terme de régularisation dans la loss et adoucir les labels (0 ou 1) pour stabiliser l'apprentissage.
- ▶ Virtual Batch Normalization : normalisation des data par rapport à un batch de référence fixe.
- ▶ Ajouter du bruit et utiliser la *distance de Wasserstein (WGAN)* pour éviter les gradients nuls.

SOLUTIONS POUR STABILISER LES GANs

- ▶ Feature Matching : aligner les statistiques des data réelles et générées via une loss du type

$$\left\| \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} f(G(\mathbf{z})) \right\|_2^2$$

- ▶ Minibatch Discrimination : une métrique de clusterisation inter-batch est ajoutée en input du modèle...
- ▶ Historical Averaging et Label Smoothing : ajouter un terme de régularisation dans la loss et adoucir les labels (0 ou 1) pour stabiliser l'apprentissage.
- ▶ Virtual Batch Normalization : normalisation des data par rapport à un batch de référence fixe.
- ▶ Ajouter du bruit et utiliser la *distance de Wasserstein (WGAN)* pour éviter les gradients nuls.

SOLUTIONS POUR STABILISER LES GANs

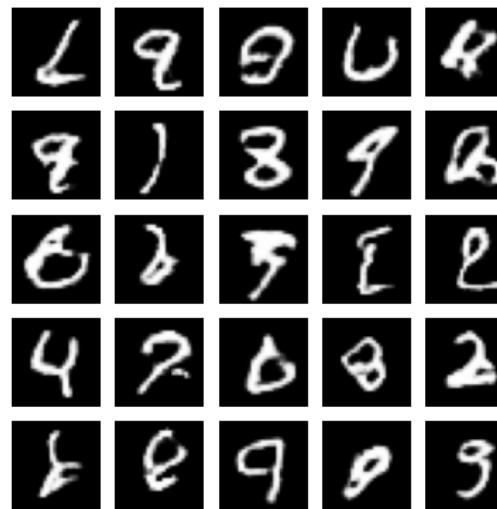
- ▶ Feature Matching : aligner les statistiques des data réelles et générées via une loss du type

$$\left\| \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} f(G(\mathbf{z})) \right\|_2^2$$

- ▶ Minibatch Discrimination : une métrique de clusterisation inter-batch est ajoutée en input du modèle...
- ▶ Historical Averaging et Label Smoothing : ajouter un terme de régularisation dans la loss et adoucir les labels (0 ou 1) pour stabiliser l'apprentissage.
- ▶ Virtual Batch Normalization : normalisation des data par rapport à un batch de référence fixe.
- ▶ Ajouter du bruit et utiliser la *distance de Wasserstein (WGAN)* pour éviter les gradients nuls.

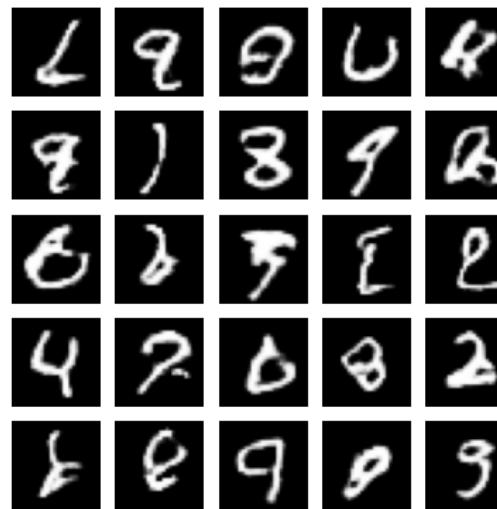
EXEMPLE

- ▶ Présenter le notebook...
- ▶ Implémentation d'un GAN sur le dataset MNIST et génération de données : bof... :(



EXEMPLE

- ▶ Présenter le notebook...
- ▶ Implémentation d'un GAN sur le dataset MNIST et génération de données : bof... :(



BIBLIOGRAPHIE



Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014).

Generative adversarial nets.

In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.



Weng, L. (2017).

From gan to wgan.

lilianweng.github.io.



Weng, L. (2019).

From gan to wgan.