

Modèles génératifs

Réseaux génératifs antagonistes & Auto-encodeurs variationnels

Statistique en grande dimension et Apprentissage profond

Axel Carlier, **Juliette Chevallier**

2024 – 2025

INSA Toulouse, juliette.chevallier@insa-toulouse.fr

Plan du cours

Introduction et Motivation

Réseaux antagonistes génératifs (GAN)

Auto-encodeurs variationnels (VAE)

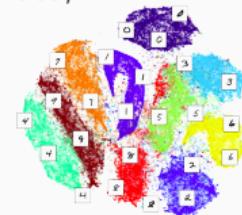
Auto-encodeurs antagonistes (AAE)

Apprentissage non-supervisé

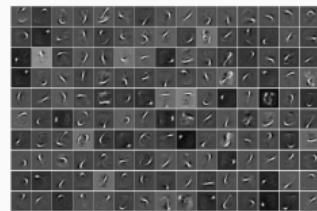
Dans le cadre de l'**apprentissage non supervisé**, on cherche à inférer de l'information à partir d'**observations uniquement** : $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$.

Des exemples :

1. Trouver des structures cachées dans des données,
2. Extraction de caractéristique,
3. Réduction de dimension,
4. Compression de données,
5. Détection de similitude dans des données,
6. Clustering (Partitionnement de données),
7. Éstimation de densité,
8. Génération de nouveaux exemples.



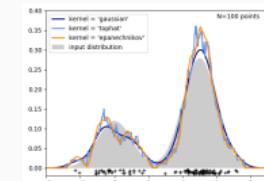
(3) Réduction de dimension



(1-2) Extraction de caractéristiques



(6) Clustering



(7) Éstimation de densité



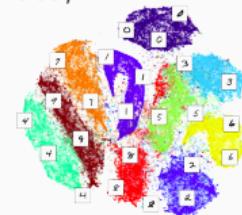
(8) Inpainting

Apprentissage non-supervisé

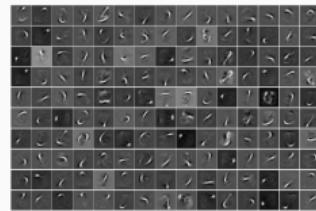
Dans le cadre de l'**apprentissage non supervisé**, on cherche à inférer de l'information à partir d'**observations uniquement** : $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$.

Des exemples :

1. Trouver des structures cachées dans des données,
2. Extraction de caractéristique,
3. Réduction de dimension,
4. Compression de données,
5. Détection de similitude dans des données,
6. Clustering (Partitionnement de données),
7. Éstimation de densité,
8. Génération de nouveaux exemples.



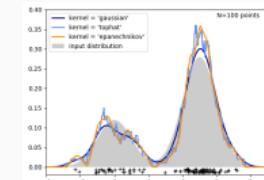
(3) Réduction de dimension



(1-2) Extraction de caractéristiques



(6) Clustering



(7) Éstimation de densité



(8) Inpainting

Modèles génératifs

Données d'entraînements



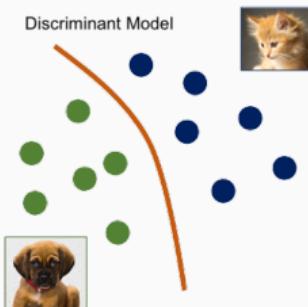
$$p_{\text{données}}(x)$$

Données générées

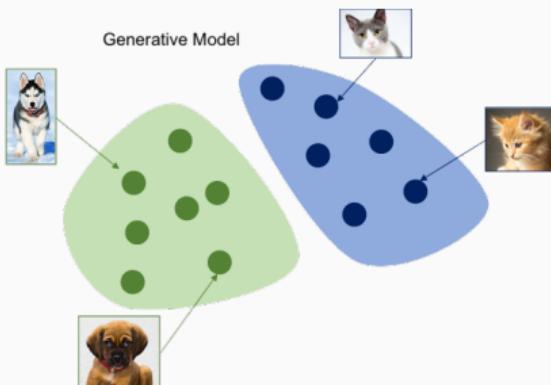


$$p_{\text{modèle}}(x)$$

Étant donné un ensemble de données dites d'entraînement (échantillons), nous voulons être capable de générer de *nouvelles* données suivant la *même distribution*.

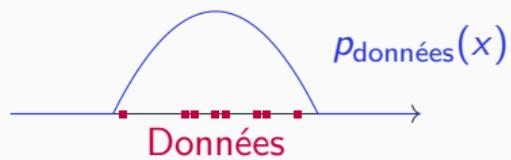


Generative Model



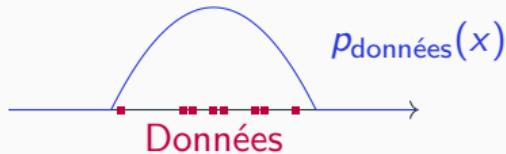
Apprentissage non-supervisé ?

Il s'agit donc bien d'un problème d'apprentissage non supervisé,

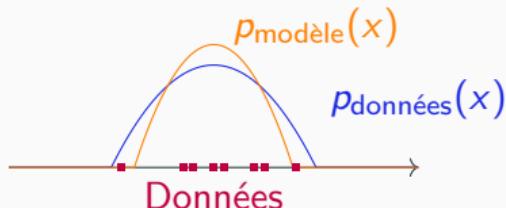


Apprentissage non-supervisé ? Estimation de densité !

Il s'agit donc bien d'un problème d'apprentissage non supervisé,
Plus spécifiquement d'**estimation de densité**.



Un bon modèle génératif doit *reproduire* au plus près la densité de probabilité des données.



Modèles génératifs - Applications

Les applications de ces modèles génératifs sont très nombreuses :

- Inpainting,
- Super-résolution,
- Colorisation,
- Traduction texte-image,
- Compression,
- Art
- Création de fake news,
- Projection de visage dans 50 ans,
- etc.



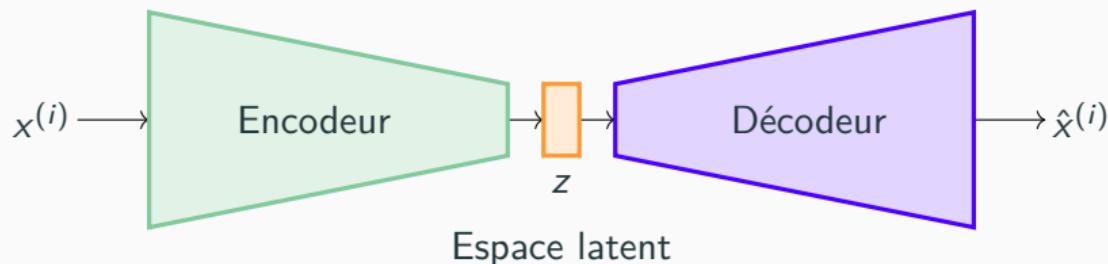
Tableau peint par une IA et vendu 432.500 \$



"The small bird has a red head with feathers that fade from red to gray from head to tail"



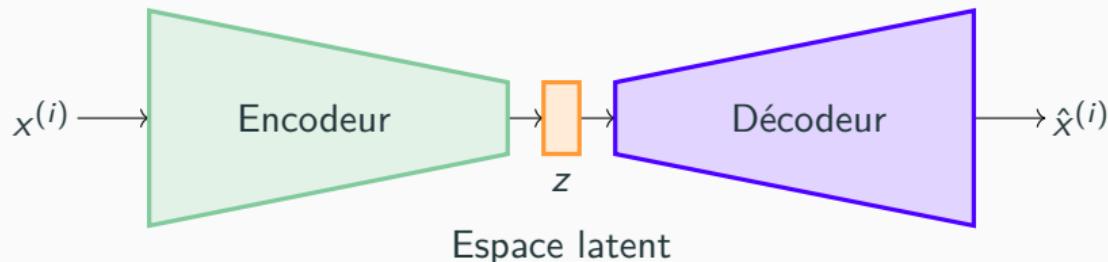
Rappels sur les auto-encodeurs



Les **auto-encodeurs** forment une classe d'algorithmes d'apprentissage non-supervisé, qui apprennent une représentation intéressante des données en minimisant une **erreur de reconstruction**.

On peut voir le décodeur comme un **modèle génératif**, qui apprend à *(re)créer une donnée à partir de sa représentation latente*.

Auto-encodeurs pour la génération de nouvelles images



Pour la génération d'image, l'*erreur de reconstruction quadratique* donne des résultats qui peuvent être décevants :



Plan du cours

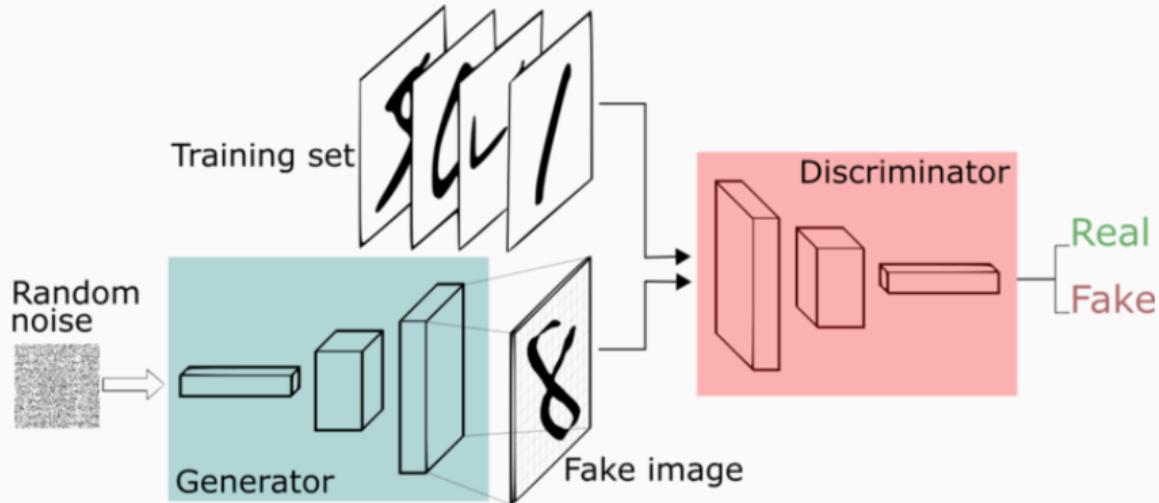
Introduction et Motivation

Réseaux antagonistes génératifs (GAN)

Auto-encodeurs variationnels (VAE)

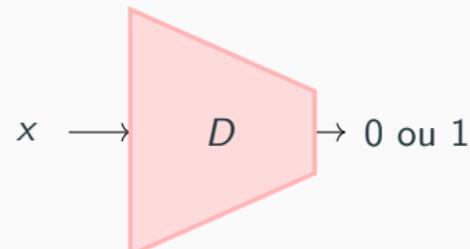
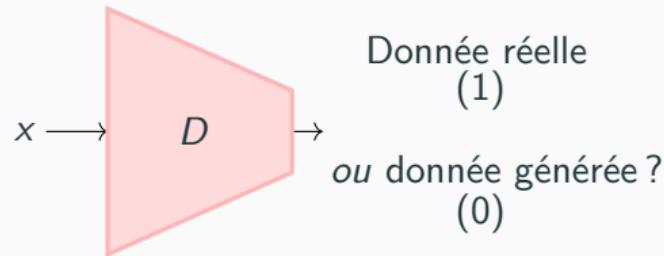
Auto-encodeurs antagonistes (AAE)

Réseaux antagonistes génératifs (GAN)



Réseaux **antagonistes** génératifs : Générateur et Discriminateur

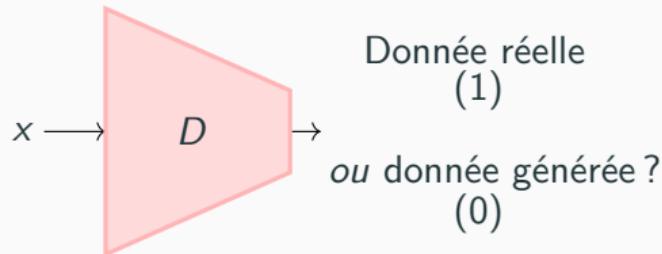
Discriminateur vs. Générateur



On définit un **classifieur binaire**, le **discriminateur** D , dont le but est de déterminer si :

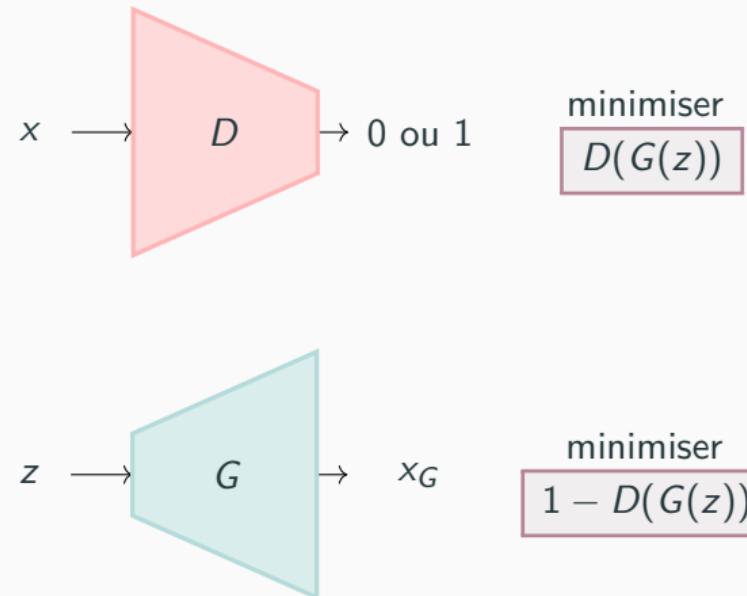
- la donnée x en entrée est une *donnée réelle*,
- ou si elle a été *générée* par un modèle.

Discriminateur vs. Générateur : Des rôles antagonistes



On définit un **classifieur binaire**, le **discriminateur** D , dont le but est de déterminer si :

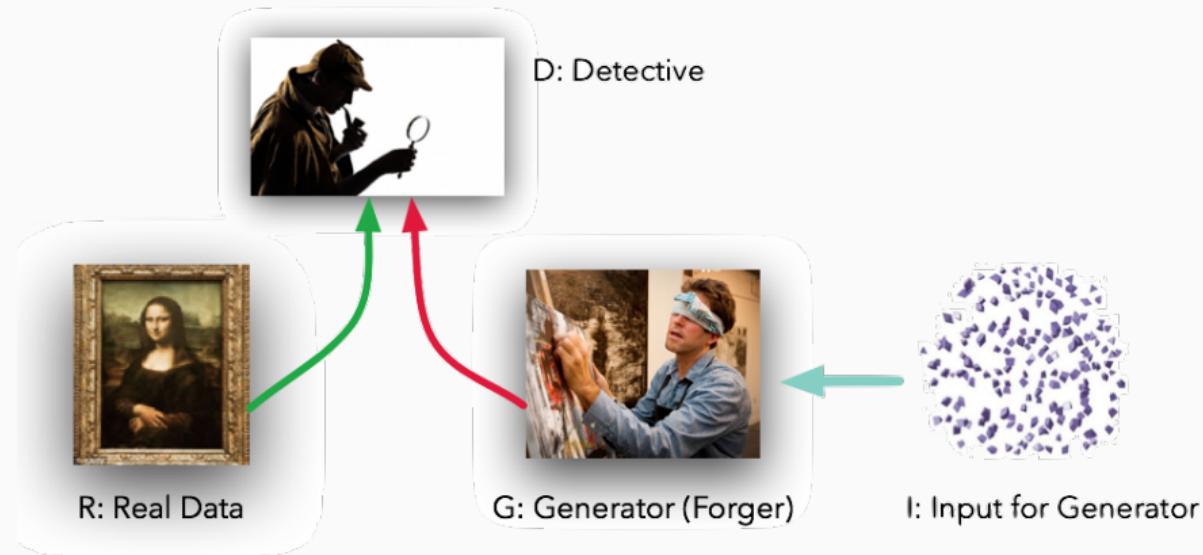
- la donnée x en entrée est une *donnée réelle*,
- ou si elle a été générée par un modèle.



Le **générateur** et le discriminateur ont des **objectifs antagonistes**.

Réseaux antagonistes génératifs : Une analogie

Une analogie : Le faussaire et le détective



Remarque : Pour visualiser l'entraînement des GANs : poloclub.github.io/ganlab

Réseaux antagonistes génératifs

- Quand le **générateur** s'améliore, il produit des **données de plus en plus réalistes** et le discriminateur a plus de difficultés à faire la différence entre données réelles et générées.
- A contrario, quand le **discriminateur** s'améliore, il **distingue mieux** les données réelles des données générées ; les données de synthèse du générateur ne ressemblent plus suffisamment aux données réelles.

Réseaux antagonistes génératifs

- Quand le **générateur** s'améliore, il produit des **données de plus en plus réalistes** et le discriminateur a plus de difficultés à faire la différence entre données réelles et générées.
- A contrario, quand le **discriminateur** s'améliore, il **distingue mieux** les données réelles des données générées ; les données de synthèse du générateur ne ressemblent plus suffisamment aux données réelles.

Cette situation est celle d'un **jeu à deux joueurs à somme nulle**, bien connue en **théorie des jeux**.

↝ Il existe une situation de **min-max**, c'est-à-dire une stratégie pour le générateur et le discriminateur où *chacun minimise le gain maximal de l'adversaire*.

Réseaux antagonistes génératifs : Fonctions de coût

- Fonction de coût du **discriminateur** (**entropie croisée**) :

$$J_D(\theta_G, \theta_D) = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} [\log D(x)] - \frac{1}{2} \mathbb{E}_z [\log (1 - D(G(z)))]$$

- Fonction de coût du **générateur** : $J_G(\theta_G, \theta_D) = -J_D(\theta_G, \theta_D)$

Réseaux antagonistes génératifs : Fonctions de coût

- Fonction de coût du **discriminateur** (**entropie croisée**) :

$$J_D(\theta_G, \theta_D) = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} [\log D(x)] - \frac{1}{2} \mathbb{E}_z [\log (1 - D(G(z)))]$$

- Fonction de coût du **générateur** : $J_G(\theta_G, \theta_D) = -J_D(\theta_G, \theta_D)$

On cherche donc les **paramètres optimaux** $\hat{\theta}$ d'un générateur qui vérifie l'équation du *min-max* suivante :

$$\operatorname{argmin}_{\theta_G} \left\{ \max_{\theta_D} \left(-J_D(\theta_G, \theta_D) \right) \right\}$$

Réseaux antagonistes génératifs : Paramètres optimaux

$$\operatorname{argmin}_{\theta_G} \left\{ \max_{\theta_D} \left(-J_D(\theta_G, \theta_D) \right) \right\}$$

La solution de ce problème est un **équilibre de Nash** $(\hat{\theta}_G, \hat{\theta}_D)$, où :

- $\hat{\theta}_G$ désigne les paramètres d'un générateur tel que $G(z) \sim p_{data}$,
- $\hat{\theta}_D$ désigne les paramètres d'un discriminateur tel que $D(x) = \frac{1}{2}$ pour tout x .

Il n'y a pas d'algorithme pratiquement utilisable qui permette de converger vers ce point d'équilibre. On utilise en fait une **descente de gradient simultanée**, qui n'a **aucune garantie théorique** de fonctionner !

Descente de gradient simultanée pour l'entraînement d'un GAN

pour $n = 1$ à N_{iter}

// Entraînement du discriminateur

Échantillonnage d'un mini-batch de bruit ($z^{(1)}, \dots, z^{(m)}$)

Échantillonnage d'un mini-batch de données ($x^{(1)}, \dots, x^{(m)}$)

Mise à jour du discriminateur pour maximiser la perte :

$$\frac{1}{m} \sum_{i=1}^m \log D(x^{(i)}) + \log (1 - D(G(z^{(i)})))$$

// Entraînement du générateur

Échantillonnage d'un mini-batch de bruit ($z^{(1)}, \dots, z^{(m)}$)

Mise à jour du générateur pour minimiser la perte :

$$\frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)})))$$

fin pour

Exemple de résultats

Sur l'exemple de la base de données *Labelled Faces in the Wild*, voici quelques exemples de résultats que l'on peut obtenir :



Step 400



Step 6000



Step 6400

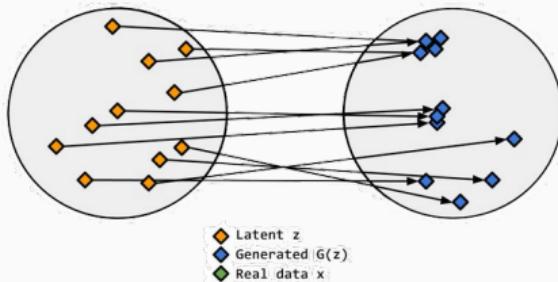


Step 10000

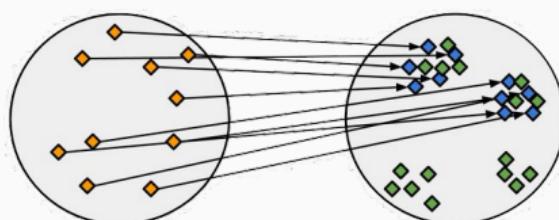
Entraînement en pratique

En pratique, les GANs sont **extrêmement compliqués** et **longs** à entraîner.

Mode Collapse: Many z to ~one x



Mode Dropping: Real Data not in $G(z)$



- Mode collapse and Mode Dropping can co-occur
- Complete training collapse is a separate phenomenon, for which extreme Mode Dropping and/or Mode Collapse are often symptoms

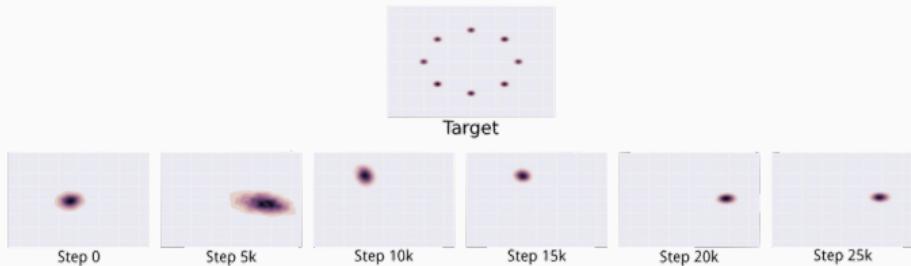


Mode Collapse on CelebA (Source: Geometric GAN)

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 9 | 0 | 6 | 9 | 1 | 1 | 1 | 9 | 4 | / |
| 7 | 9 | 1 | 1 | 1 | 1 | 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 2 | 1 | 1 | 1 | 1 | 7 | 7 | 1 | 1 | 2 | 9 | 1 | 6 | |
| / | 1 | 1 | 7 | 1 | 9 | 5 | 5 | 4 | 9 | 1 | 1 | 7 | 1 | | |
| 1 | 1 | 9 | 1 | 1 | 1 | 9 | 4 | 2 | 1 | 1 | 7 | 9 | 4 | 1 | |
| 6 | 1 | 1 | 1 | 5 | 1 | 1 | 1 | 1 | 9 | 9 | 9 | 2 | 1 | 1 | |
| 1 | 1 | 1 | 1 | 1 | 7 | 9 | 1 | 1 | 7 | 7 | 2 | 1 | 1 | | |
| 1 | 1 | 1 | 6 | 1 | 7 | 2 | 1 | 1 | 1 | 2 | 1 | 9 | 1 | 6 | 1 |
| 1 | 2 | 1 | 1 | 7 | 9 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 1 | 1 | 7 | 7 | 1 | 4 | 9 | 4 | 0 | 1 | 7 | 0 | 1 | 0 | 1 | |
| 1 | 7 | 1 | 0 | 0 | 1 | 1 | 1 | 9 | 7 | 1 | 1 | 1 | 7 | | |
| 1 | 1 | 9 | 6 | 1 | 7 | 3 | 1 | 1 | 1 | 7 | 5 | 9 | 6 | 4 | 1 |
| 1 | 9 | 7 | 1 | 4 | 1 | 7 | 1 | 1 | 1 | 1 | 7 | 1 | 1 | 7 | 1 |
| 1 | 0 | 1 | 1 | 1 | 7 | 4 | 7 | 1 | 6 | 0 | 1 | 1 | 1 | 9 | |
| 1 | 7 | 6 | 1 | 0 | 1 | 1 | 1 | 9 | 1 | 1 | 1 | 9 | 1 | | |
| 1 | 1 | 9 | 4 | 9 | 6 | 1 | 0 | 6 | 1 | 1 | 1 | 7 | 1 | | |

Mode Dropping on MNIST (Source:TripletGAN)

Mode collapse



Un des problèmes les plus courants lorsque l'on entraîne des GANs est celui du **mode collapse**.

Plutôt que de converger vers une distribution comprenant tous les modes de l'ensemble d'apprentissage, le générateur se contente de modéliser **un mode à la fois**, et **alterne entre les modes** au cours du temps quand le discriminateur apprend à rejeter le mode courant.



[Metz et al.] Unrolled generative adversarial networks.

Mode collapse : Pourquoi ?

On cherche à résoudre le problème :

$$\operatorname{argmin}_{\theta_G} \left\{ \max_{\theta_D} \left(-J_D(\theta_G, \theta_D) \right) \right\}.$$

Mode collapse : Pourquoi ?

On cherche à résoudre le problème :

$$\operatorname{argmin}_{\theta_G} \left\{ \max_{\theta_D} \left(-J_D(\theta_G, \theta_D) \right) \right\}.$$

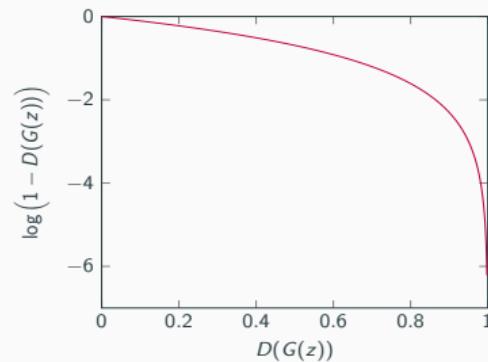
Mais, la descente de gradient simultanée peut tout aussi bien résoudre le problème

$$\max_{\theta_D} \left\{ \operatorname{argmin}_{\theta_G} \left(-J_D(\theta_G, \theta_D) \right) \right\}.$$

Cette dernière formulation conduit à un générateur qui cherche à *produire l'échantillon jugé le plus réaliste par le discriminateur*.

Éviter le **mode collapse** est un **champ de recherche** très actif !

Évanescence des gradients

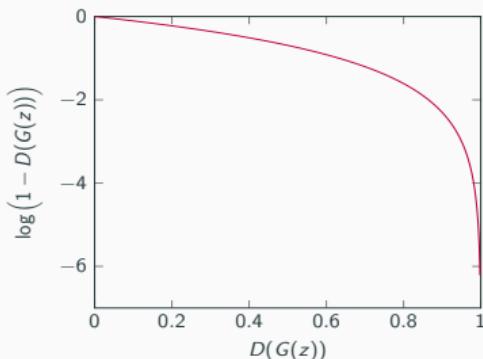


Si le générateur produit des échantillons trop irréalistes, le discriminateur peut rejeter ces échantillons avec une trop grande confiance.

$$\begin{aligned} G(z) \approx p_{data} &\implies D(G(z)) \approx 0 \\ &\implies \log(1 - D(G(z))) \approx 0 \end{aligned}$$

Les gradients sont trop *faibles* pour faire évoluer *efficacement* le générateur.

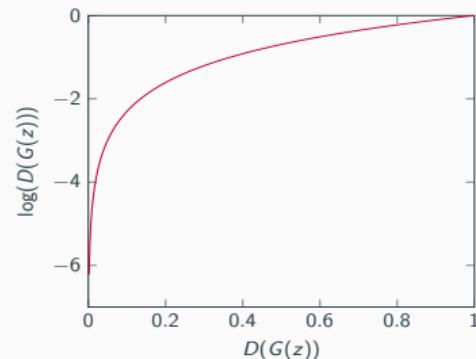
Évanescence des gradients



Si le générateur produit des échantillons trop irréalistes, le discriminateur peut rejeter ces échantillons avec une trop grande confiance.

$$\begin{aligned} G(z) \approx p_{data} &\implies D(G(z)) \approx 0 \\ &\implies \log(1 - D(G(z))) \approx 0 \end{aligned}$$

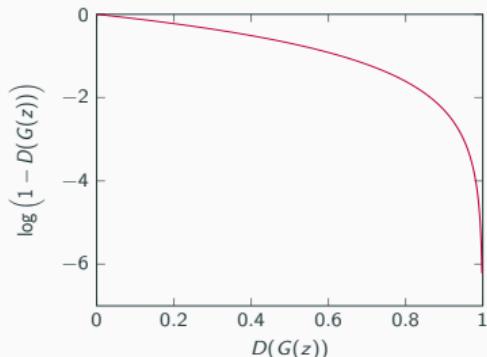
Les gradients sont trop faibles pour faire évoluer efficacement le générateur.



Possible de remplacer la minimisation de $\log(1 - D(G(z)))$ par la **maximisation de $\log(D(G(z)))$** , dont les propriétés des valeurs de gradients sont plus intéressantes.

Attention : Pas de justification théorique !
Mais fonctionne en pratique.

Instabilité du générateur

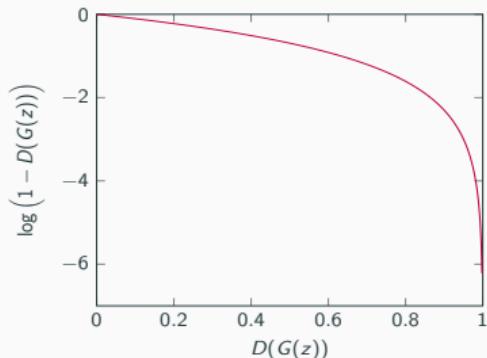


Les gradients sont très élevés et provoquent l'*instabilité* du générateur.

Si le générateur produit des échantillons trop réalistes, le discriminateur peut catégoriser ces échantillons comme réalistes avec une trop grande confiance.

$$\begin{aligned} G(z) \sim p_{data} &\implies D(G(z)) \approx 1 \\ &\implies \log(1 - D(G(z))) \rightarrow -\infty \end{aligned}$$

Instabilité du générateur



Si le générateur produit des échantillons trop réalistes, le discriminateur peut catégoriser ces échantillons comme réalistes avec une trop grande confiance.

$$\begin{aligned} G(z) \sim p_{data} &\implies D(G(z)) \approx 1 \\ &\implies \log(1 - D(G(z))) \rightarrow -\infty \end{aligned}$$

Les gradients sont très élevés et provoquent l'*instabilité* du générateur.

Pour limiter ce problème, on peut utiliser la technique du **Label Smoothing**.

Au lieu d'entraîner le discriminateur à prédire 1 pour les données réelles, on l'entraîne à prédire une **valeur aléatoire** typiquement entre 0.7 et 0.9.

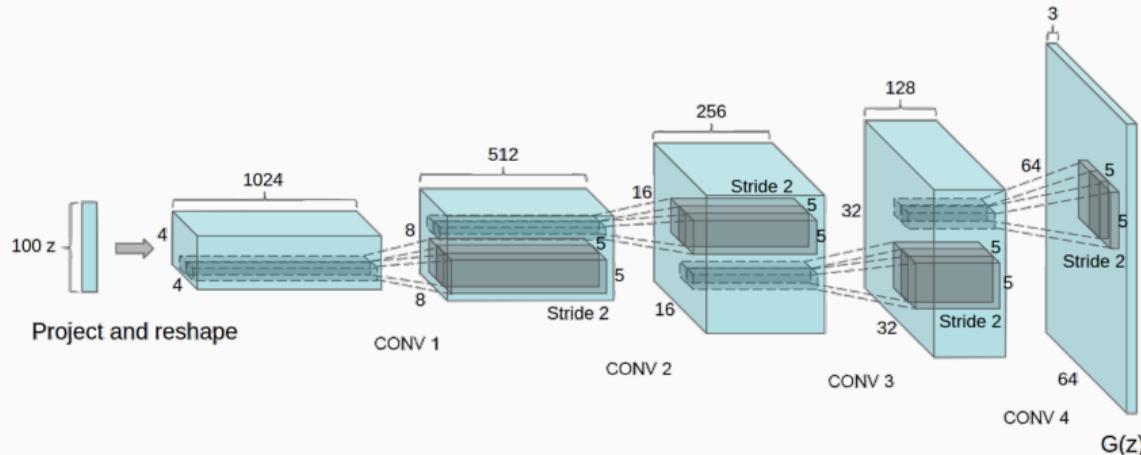
[Salimans et al.] Improved techniques for training gans.

DCGAN : Deep Convolutional Generative Adversarial Networks

L'article DCGAN est fondamental car il introduit une série de **recommandations** pour l'architecture du générateur et du discriminateur :

- Ne pas utiliser de couche de *pooling*, mais plutôt du **stride**,
- Introduire de la **batch normalization**,
- Utiliser des activations ReLU pour les couches cachées du **générateur**, et la fonction tanh en sortie,
- Utiliser des activations LeakyReLU pour les couches cachées du **discriminateur**.

DCGAN : Deep Convolutional Generative Adversarial Networks

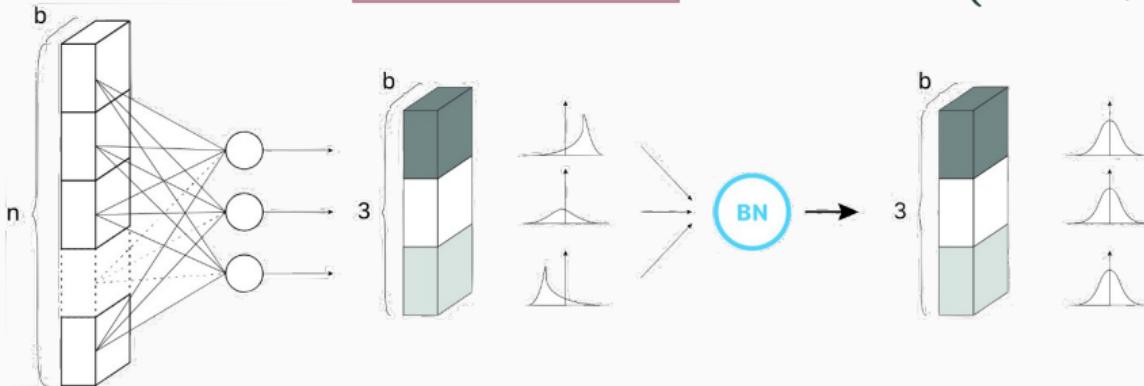


- Le **stride** permet au générateur d'apprendre les paramètres de mise à l'échelle plutôt que de la forcer avec de l'*upsampling*.
- L'activation de sortie tanh permet une **convergence** plus rapide du modèle et une meilleure gestion de la couleur, en raison de son caractère borné.

Batch Normalization

Pour chaque couche cachée,

$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 - \varepsilon}} \quad \text{avec } \varepsilon \in \mathbb{R}^+ \text{ et } \begin{cases} \mu = \frac{1}{n} \sum_{i=1}^n z^{(i)} \\ \sigma^2 = \frac{1}{n} \sum_{i=1}^n (z^{(i)} - \mu)^2 \end{cases}$$



- Les activations sont normalisées par batch pour avoir une moyenne nulle et une variance égale à 1.
- Le problème d'optimisation est **mieux conditionné**, ce qui stabilise l'entraînement, et est particulièrement utile pour les GANs.

Batch Normalization

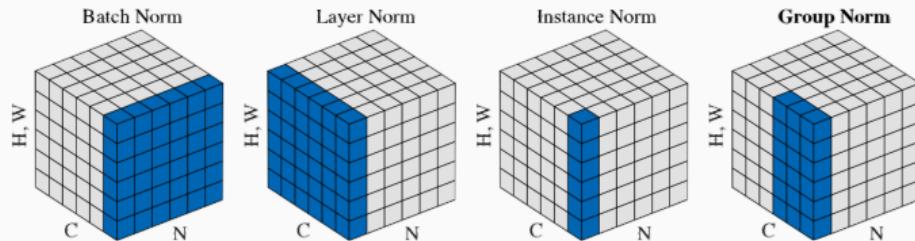
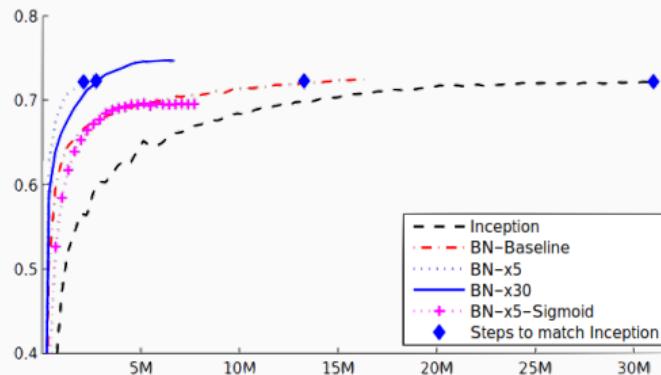
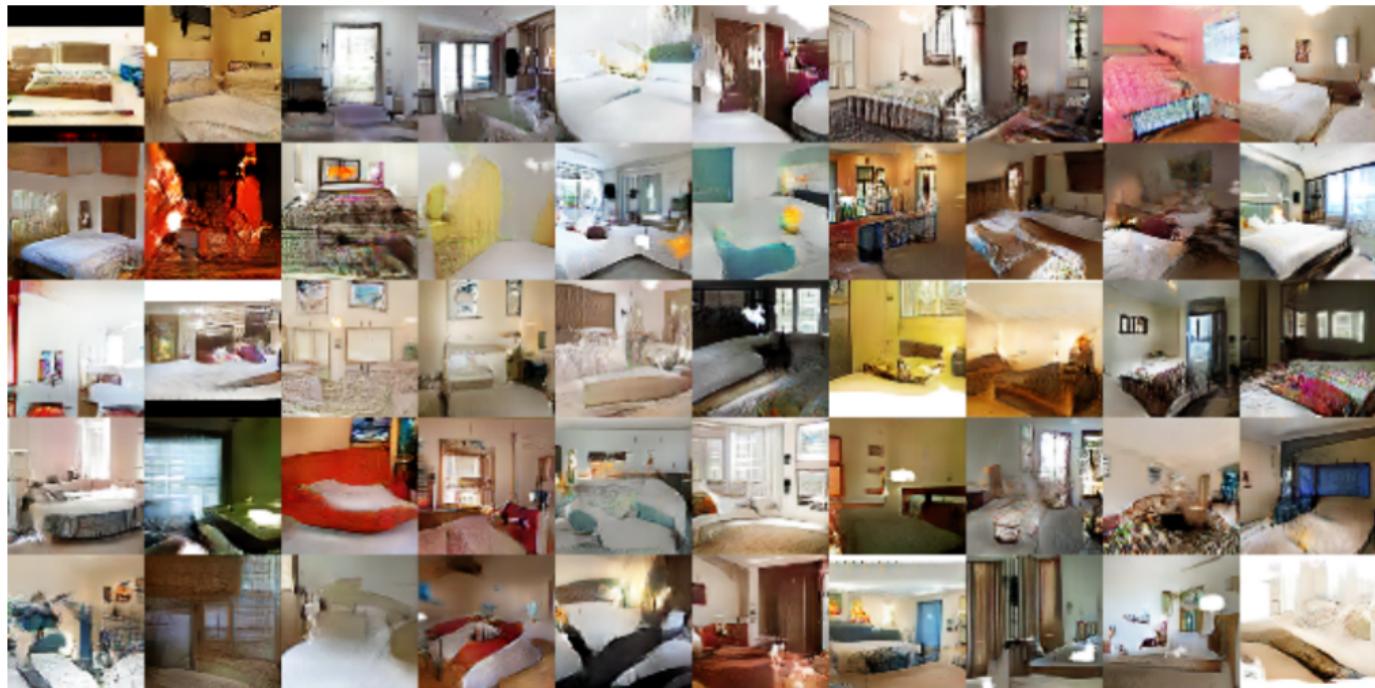


Figure 2. Normalization methods. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

DCGAN : Deep Convolutional Generative Adversarial Networks

Quelques résultats tirés de l'article :



DCGAN : Deep Convolutional Generative Adversarial Networks

Voici des résultats sur la base de données *Labelled Faces in the Wild* :

- Première ligne : Avec les recommandations de DCGAN,
- Deuxième ligne : Sans.

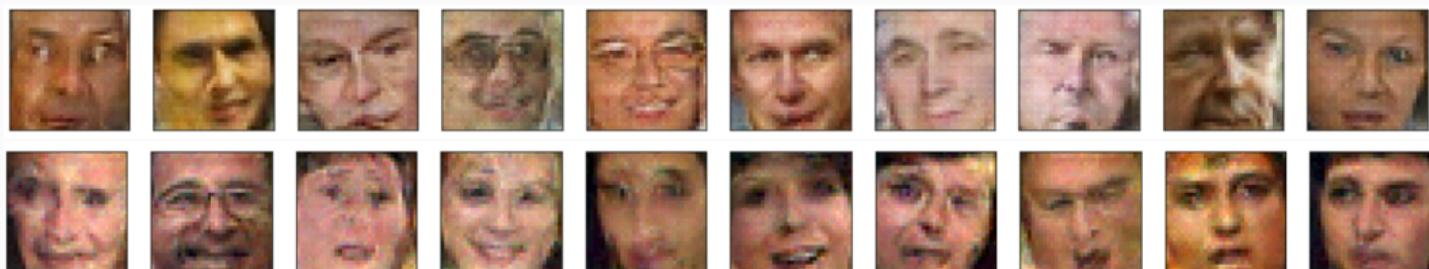


Ces résultats mettent en évidence une autre difficulté des GANs : *l'impossibilité de quantifier la qualité des résultats.*

DCGAN : Deep Convolutional Generative Adversarial Networks

Voici des résultats sur la base de données *Labelled Faces in the Wild* :

- Première ligne : Avec les recommandations de DCGAN,
- Deuxième ligne : Sans.

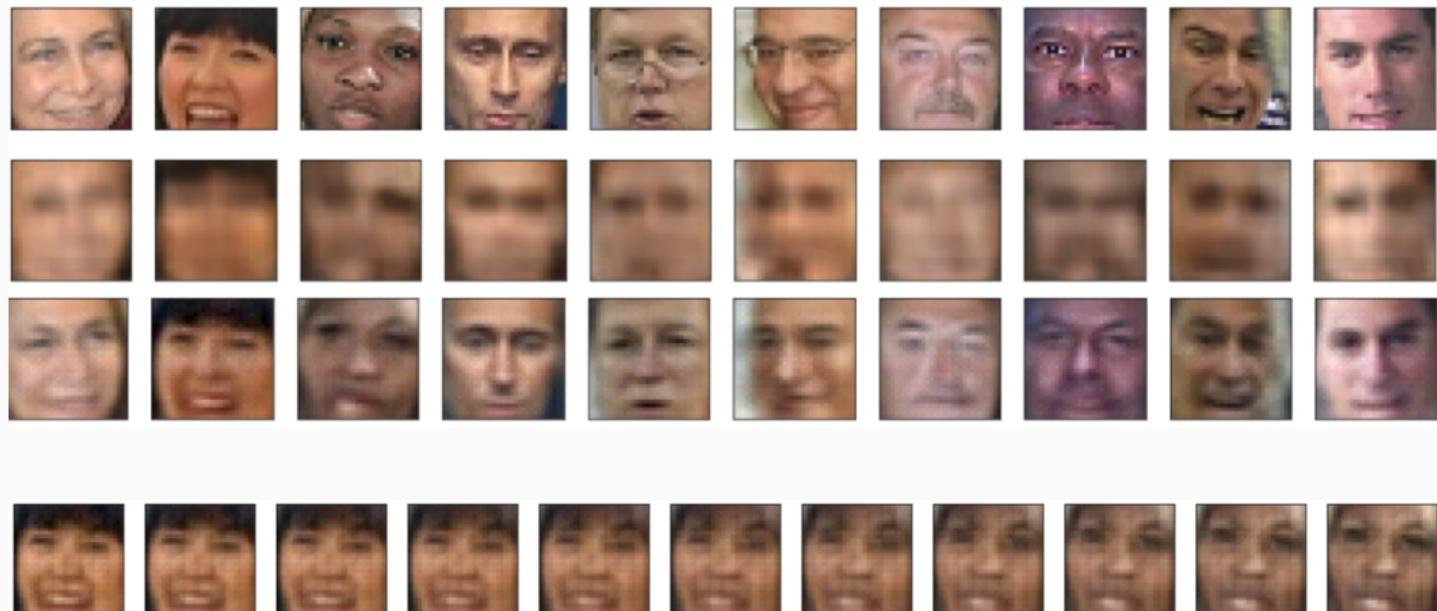


Ces résultats mettent en évidence une autre difficulté des GANs : *l'impossibilité de quantifier la qualité des résultats.*

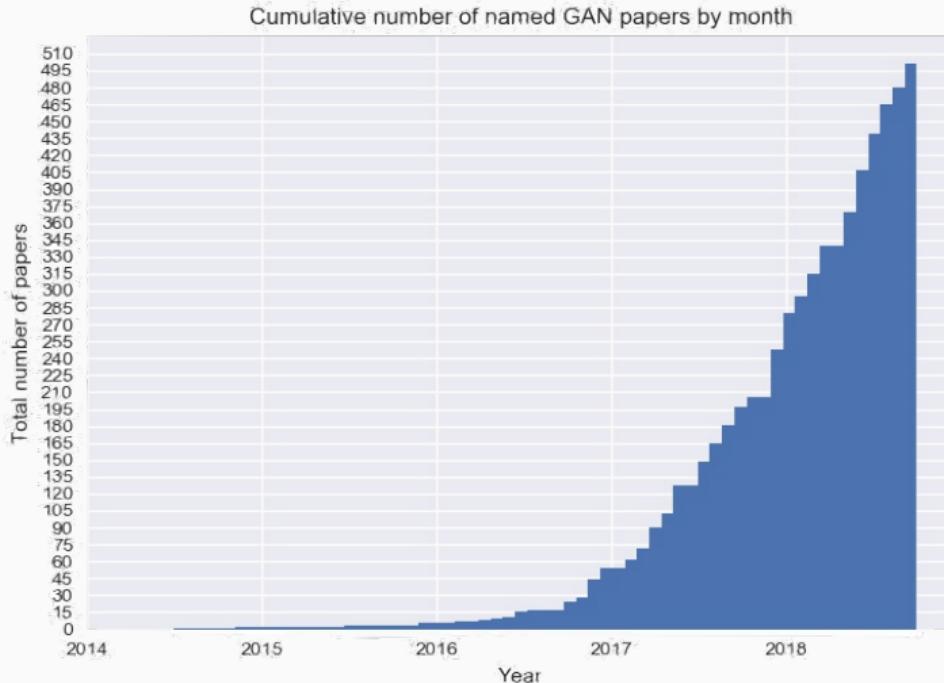
Une possibilité est d'évaluer l'entropie des prédictions d'un réseau entraîné sur ImageNet pour estimer la qualité de la représentation générée dans une image, ce qui est implémenté par l'**Inception score**.

DCGAN : Deep Convolutional Generative Adversarial Networks

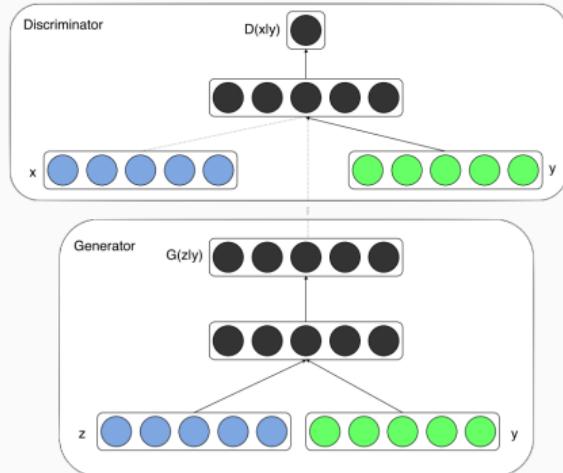
Les bonnes pratiques de DCGAN peuvent aussi être appliquées aux **auto-encodeurs**. Cela améliore grandement les résultats :



Les GANs : Tentative de zoologie



cGAN : Conditional Generative Adversarial Network



Étant données des données labellisées (x, y) :

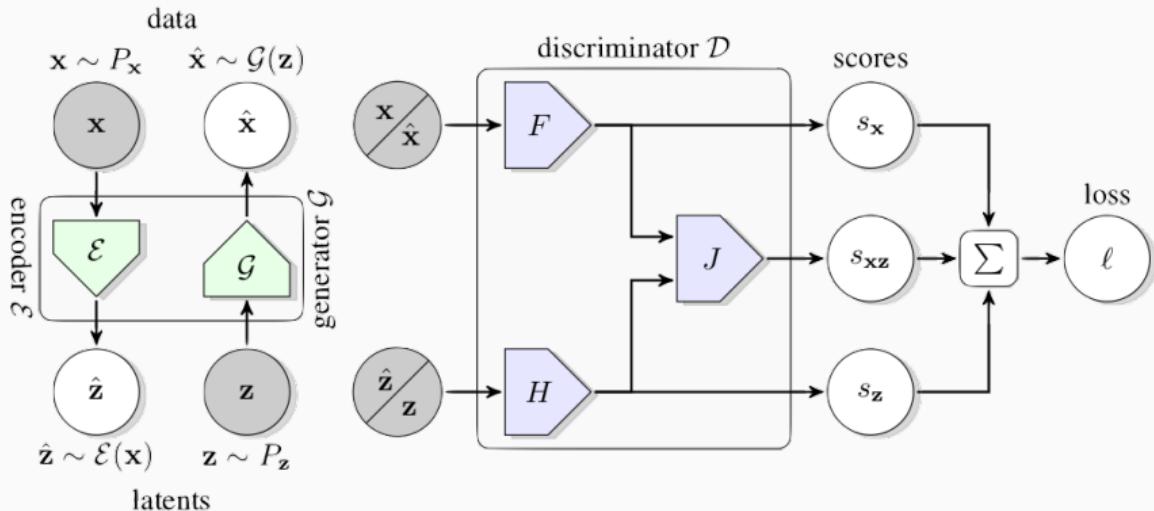
$$\min_G \max_D (D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z|y)))]$$

BigGAN

Un des défis principaux des GANs est maintenant de produire des images de qualité croissante, comme dans l'article (relativement) récent BigGAN.



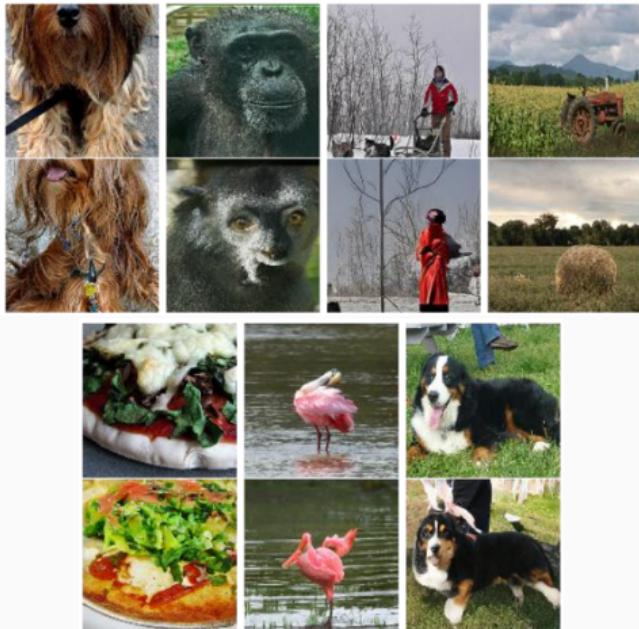
Extraction de caractéristiques : BigBiGAN



Un autoencodeur et trois discriminateurs !

[Donahue et al.] Large Scale Adversarial Representation Learning

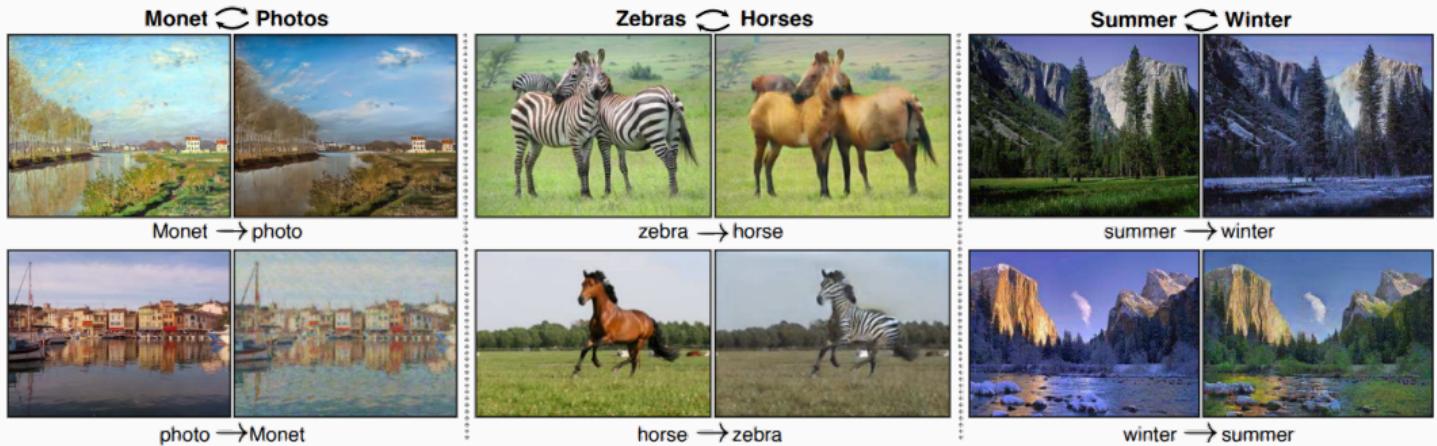
Extraction de caractéristiques : BigBiGAN



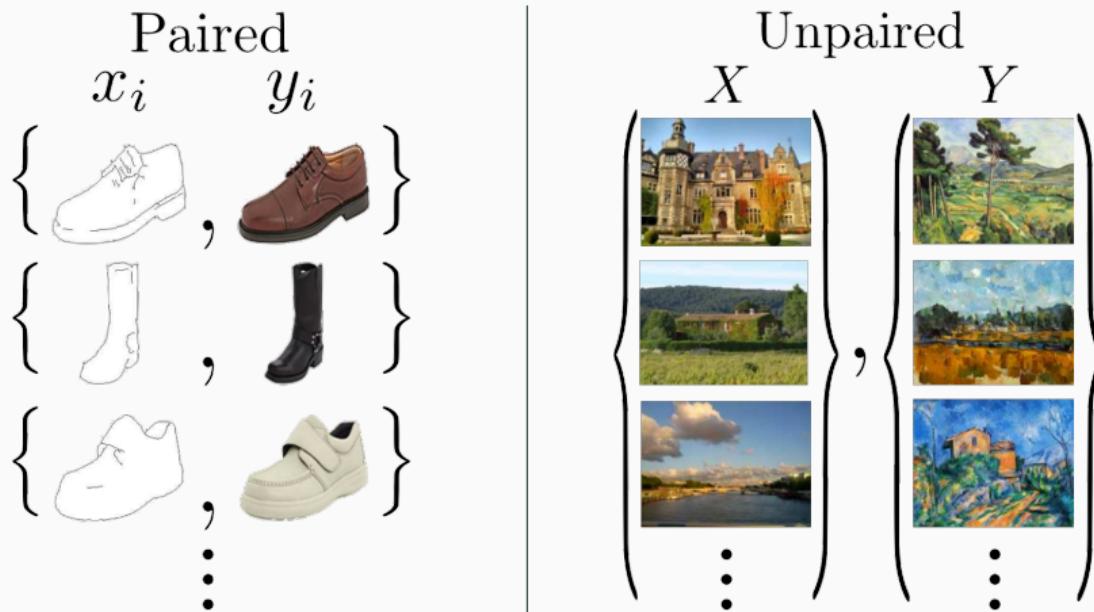
| Method | Arch. | Param. | Top1 |
|-------------------|-------|--------|-------------|
| Supervised | R50 | 24 | 76.5 |
| Colorization [65] | R50 | 24 | 39.6 |
| Jigsaw [46] | R50 | 24 | 45.7 |
| NPID [58] | R50 | 24 | 54.0 |
| BigBiGAN [15] | R50 | 24 | 56.6 |
| LA [68] | R50 | 24 | 58.8 |
| NPID++ [44] | R50 | 24 | 59.0 |
| MoCo [24] | R50 | 24 | 60.6 |
| SeLa [2] | R50 | 24 | 61.5 |
| PIRL [44] | R50 | 24 | 63.6 |
| CPC v2 [28] | R50 | 24 | 63.8 |
| PCL [37] | R50 | 24 | 65.9 |
| SimCLR [10] | R50 | 24 | 70.0 |
| MoCov2 [11] | R50 | 24 | 71.1 |
| SwAV | R50 | 24 | 75.3 |

[Donahue et al.] Large Scale Adversarial Representation Learning

Un exemple d'application : L'adaptation de domaine

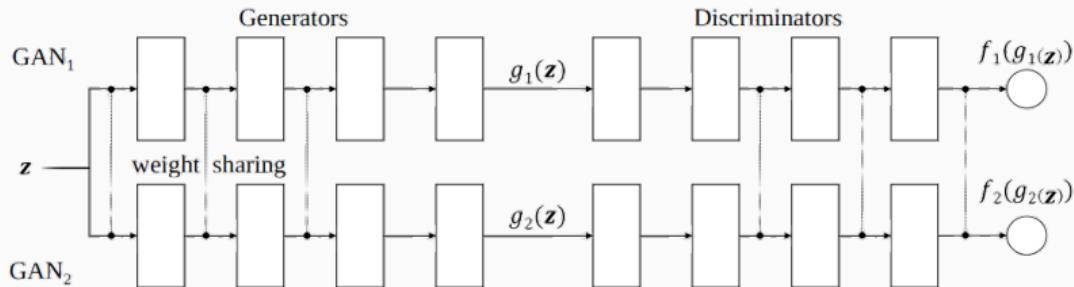


Adaptation de domaine supervisée ou non-supervisée



[Zhu et al.] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CoGAN : Coupled Generative Adversarial Network

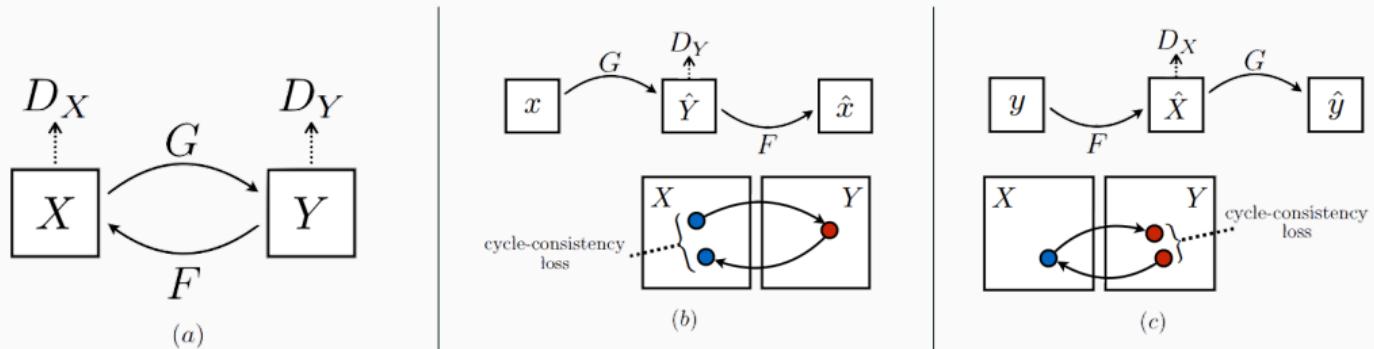


- *Adaptation de domaine non-supervisée*
- *Partage des poids* entre générateurs et discriminateurs de deux GANs différents.
- Entraînement simultané des 2 GANs avec des images des deux distributions.



Exemple de résultat : génération conjointe d'images RGB et de cartes de profondeur.

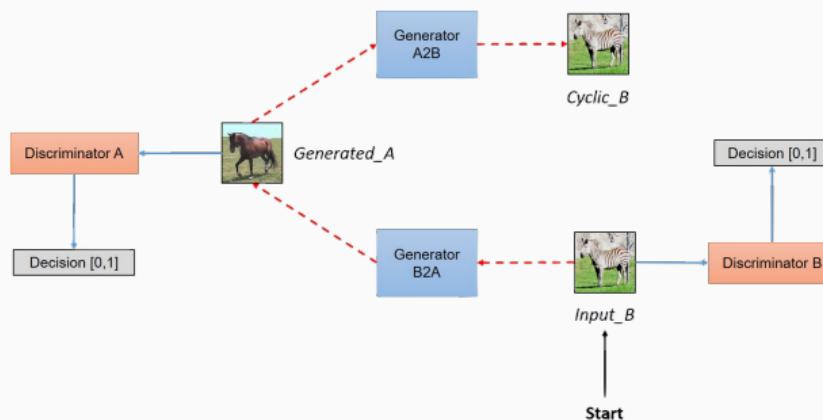
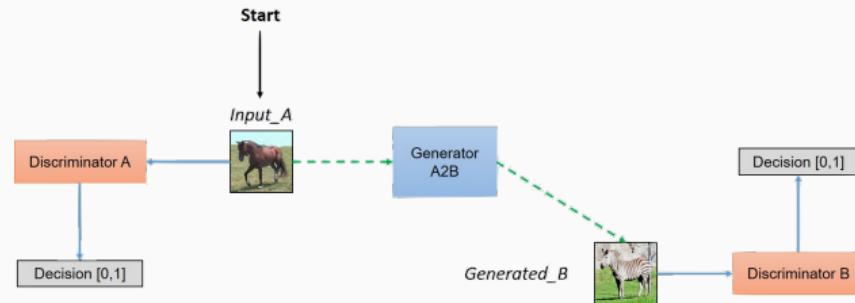
CycleGAN : Cycle-Consistent Adversarial Network



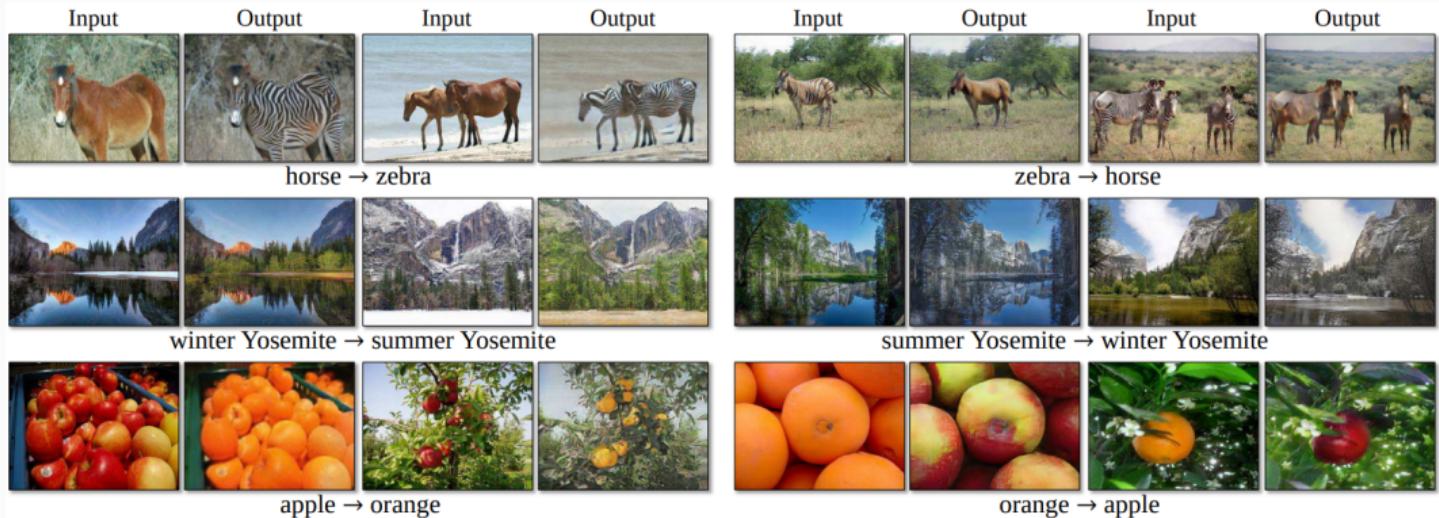
- Adaptation de domaine non-supervisée
- Entraînement simultané des 2 GANs avec des images des deux distributions.
- Ajout de deux fonctions de coût assurant la cohérence des cycles.

[Zhu et al.] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN : Cycle-Consistent Adversarial Network

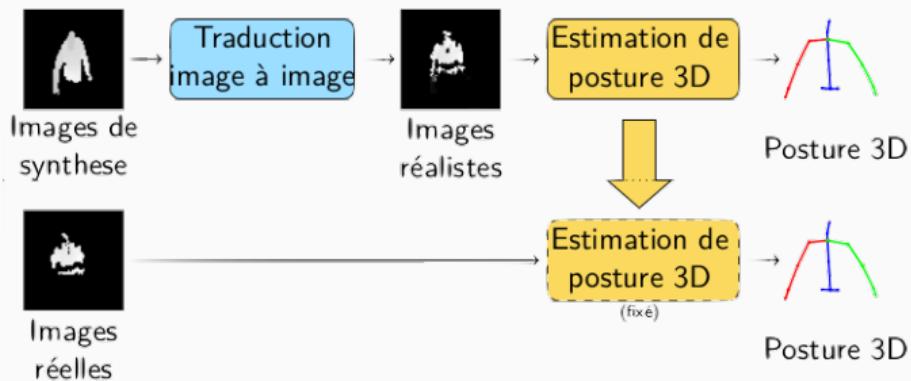
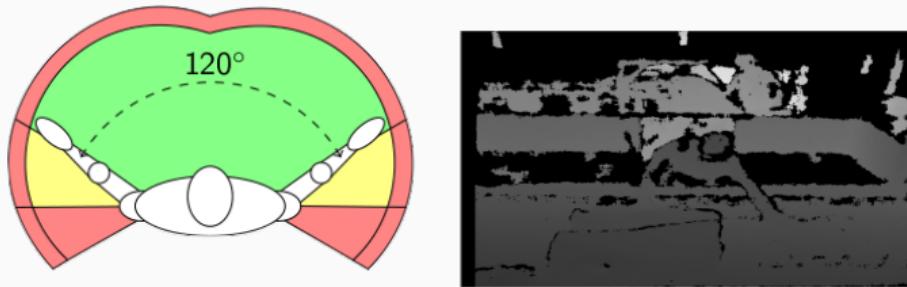


CycleGAN : Cycle-Consistent Adversarial Network



Exemple de résultats du CycleGAN.

Un exemple d'application industrielle des GANs



[Blanc-Beyne] Estimation de posture 3D à partir de données imprécises et incomplètes.

Plan du cours

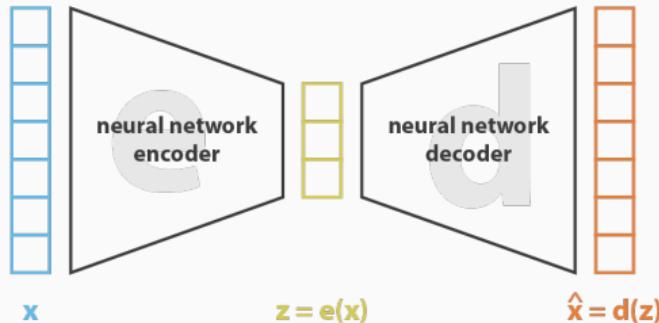
Introduction et Motivation

Réseaux antagonistes génératifs (GAN)

Auto-encodeurs variationnels (VAE)

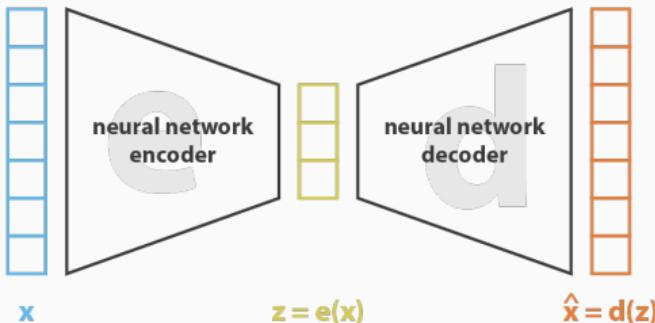
Auto-encodeurs antagonistes (AAE)

Retour sur les auto-encodeurs



$$\text{loss} = \| \mathbf{x} - \hat{\mathbf{x}} \|^2 = \| \mathbf{x} - \mathbf{d}(\mathbf{z}) \|^2 = \| \mathbf{x} - \mathbf{d}(e(\mathbf{x})) \|^2$$

Retour sur les auto-encodeurs



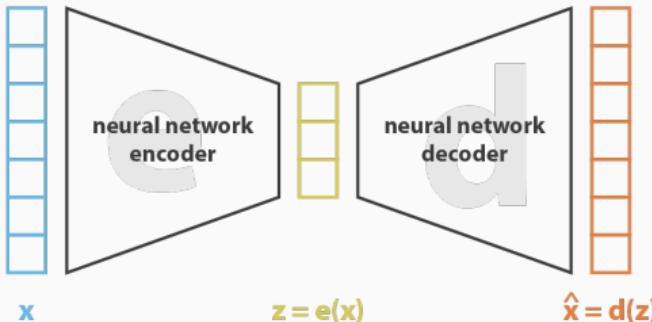
$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

Cela revient à supposer que :

- Nos *données* x sont issues de *variables latentes* z .

Exemple : Photos de voitures représentent des véhicules d'une certaine marque, d'un certain modèle, d'une certaine couleur, prises sous un certain angle, etc.

Retour sur les auto-encodeurs



$$\text{loss} = \|x - \hat{x}\|^2 = \|x - d(z)\|^2 = \|x - d(e(x))\|^2$$

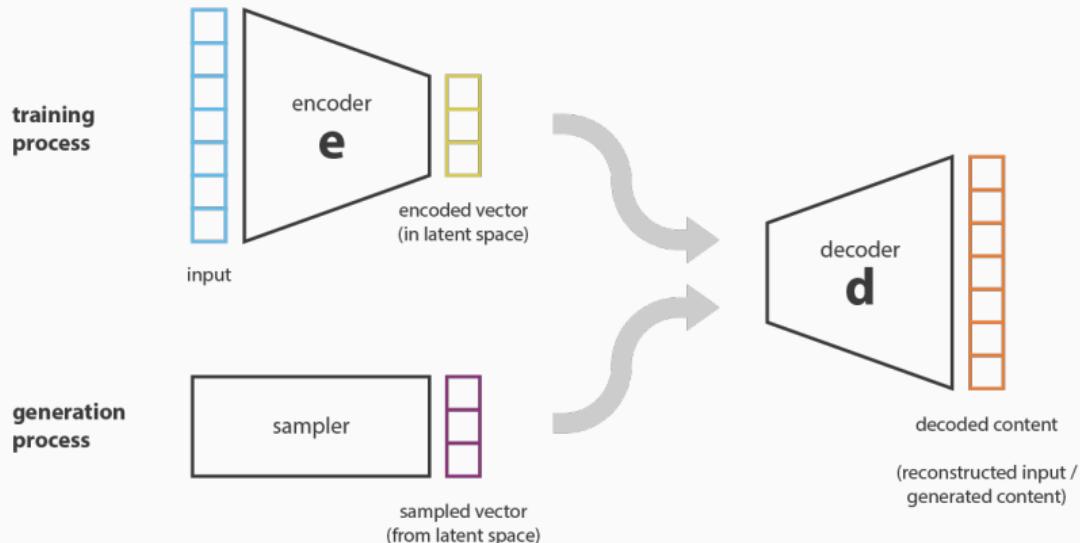
Cela revient à supposer que :

- Nos *données* x sont issues de *variables latentes* z .

Exemple : Photos de voitures représentent des véhicules d'une certaine marque, d'un certain modèle, d'une certaine couleur, prises sous un certain angle, etc.

- Il existe une fonction qui *génère* les données x à partir de valeurs des variables latentes z et on modélise cette fonction par un réseau de neurones.

Usage des auto-encodeurs pour générer de nouvelles données



- On peut utiliser les auto-encodeurs pour générer de nouvelles données en décodant des points qui sont **échantillonnés** de manière aléatoire dans l'**espace latent**.
- La qualité et la pertinence des données générées dépendent de la **régularité** de l'espace latent, et est donc plutôt mauvaise !

Régularité de l'espace latent des auto-encodeurs

Pour améliorer la génération de nouvelles données, il faut donc améliorer la **régularité de l'espace latent**.



Régularité de l'espace latent des auto-encodeurs

Pour améliorer la génération de nouvelles données, il faut donc améliorer la **régularité de l'espace latent**.

Remarque : Manque de structure parmi les données encodées dans l'espace latent normal.



- *Rien* dans l'entraînement d'un auto-encodeur oblige à une telle organisation : Auto-encodeur uniquement entraîné à coder et décoder avec le **moins de pertes possible**, peu importe comment l'espace latent est organisé ;
- Ainsi, sans contrainte sur l'architecture, le réseau profite de toutes les possibilités d'**overfitting** possible pour accomplir au mieux sa tâche.

Régularité de l'espace latent des auto-encodeurs

Pour améliorer la génération de nouvelles données, il faut donc améliorer la **régularité de l'espace latent**.

Remarque : Manque de structure parmi les données encodées dans l'espace latent normal.



- *Rien* dans l'entraînement d'un auto-encodeur oblige à une telle organisation : Auto-encodeur uniquement entraîné à coder et décoder avec le **moins de pertes possible**, peu importe comment l'espace latent est organisé ;
- Ainsi, sans contrainte sur l'architecture, le réseau profite de toutes les possibilités d'**overfitting** possible pour accomplir au mieux sa tâche.

~~> Régularisation explicite !

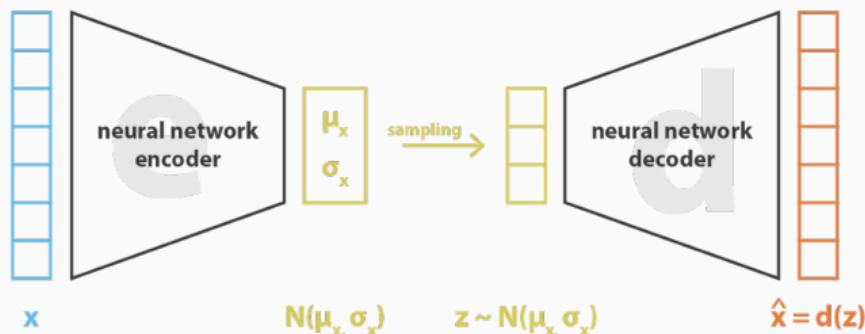
Auto-encodeur variationnel \equiv *Auto-codeur dont l'apprentissage est régularisé éviter l'overfitting et s'assurer que l'espace latent a de bonnes propriétés permettant le processus génératif.* 45

Régularisation explicite de l'espace latent

- Au lieu d'encoder une entrée comme un point unique, nous l'encodons comme une **distribution sur l'espace latent** ;
- En pratique, **loi normale** : Encodeur entraîné à renvoyer la **moyenne** μ_x et la **matrice de covariance** Σ_x décrivant ces gaussiennes ;

Régularisation explicite de l'espace latent

- Au lieu d'encoder une entrée comme un point unique, nous l'encodons comme une **distribution sur l'espace latent** ;
- En pratique, **loi normale** : Encodeur entraîné à renvoyer la **moyenne** μ_x et la **matrice de covariance** Σ_x décrivant ces gaussiennes ;
- Entrainement :
 1. Entrée codée comme une distribution (gausienne) sur l'espace latent,
 2. Un point de l'espace latent est échantillonné,
 3. Le point échantillonné est décodé et l'erreur de reconstruction calculée,
 4. L'erreur de reconstruction est rétro-propagée à travers le réseau.



Régularisation explicite de l'espace latent

Remarque : Contrainte de régularité attendues :

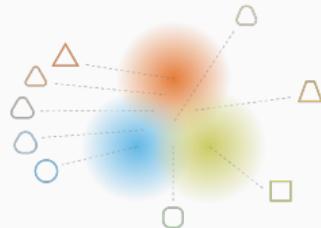
- *Continuité* : Deux points proches dans l'espace latent ne doivent pas donner deux résultats complètement différents une fois décodés.
- *Complétude* : Un point échantillonné dans l'espace latent doit donner un contenu "significatif" une fois décodé.



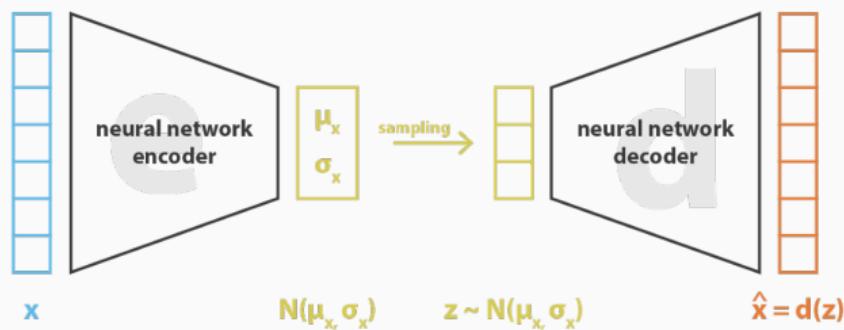
Régularisation explicite de l'espace latent

Remarque : Contrainte de régularité attendue :

- *Continuité* : Deux points proches dans l'espace latent ne doivent pas donner deux résultats complètement différents une fois décodés.
- *Complétude* : Un point échantillonné dans l'espace latent doit donner un contenu "significatif" une fois décodé.



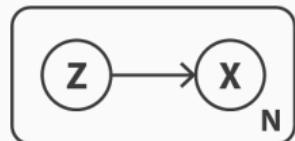
Espace latent de lois gaussiennes pas suffisant pour assurer continuité et complétude
~~> **Régularisation** en imposant que $\mathcal{N}(\mu_x, \sigma_x)$ soit proche de $\mathcal{N}(0, 1)$.



$$\text{loss} = \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

Cadre probabiliste

Modèle hiérarchique : Nos observations x dépendent d'une variable latente z , i.e. d'une variable qu'on ne peut observer.



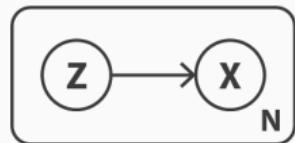
- Distribution *a priori* sur l'**espace latent** : $p(z)$;
- **Encodeur** \equiv Loi conditionnelle de z sachant x : $p(z|x)$;
- **Décodeur** \equiv Loi conditionnelle de x sachant z : $p(x|z)$

Theorem (Loi de Bayes)

$$\begin{cases} p(z|x) = \frac{p(x|z) p(z)}{p(x)} \\ p(x) = \int_{\mathcal{Z}} p(x|z) p(z) dz \end{cases}$$

Cadre probabiliste et hypothèse de modélisation

Modèle hiérarchique : Nos observations x dépendent d'une variable latente z , i.e. d'une variable qu'on ne peut observer.



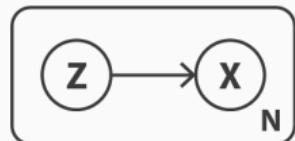
- Distribution *a priori* sur l'**espace latent** : $p(z) \equiv \mathcal{N}(0, 1)$;
- **Encodeur** \equiv Loi conditionnelle de z sachant x : $p(z|x)$;
- **Décodeur** \equiv Loi conditionnelle de x sachant z : $p(x|z) \equiv \mathcal{N}(f(z), \sigma^2)$ $f \in F, \sigma \in \mathbb{R}^+$

Theorem (Loi de Bayes)

$$\begin{cases} p(z|x) = \frac{p(x|z) p(z)}{p(x)} \\ p(x) = \int_{\mathcal{Z}} p(x|z) p(z) dz \end{cases}$$

Cadre probabiliste et hypothèse de modélisation

Modèle hiérarchique : Nos observations x dépendent d'une variable latente z , i.e. d'une variable qu'on ne peut observer.



- Distribution *a priori* sur l'**espace latent** : $p(z) \equiv \mathcal{N}(0, 1)$;
- **Encodeur** \equiv Loi conditionnelle de z sachant x : $p(z|x)$;
- **Décodeur** \equiv Loi conditionnelle de x sachant z : $p(x|z) \equiv \mathcal{N}(f(z), \sigma^2)$ $f \in F, \sigma \in \mathbb{R}^+$

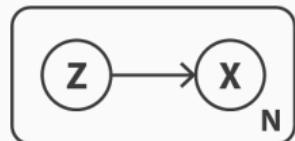
Theorem (Loi de Bayes)

$$\begin{cases} p(z|x) = \frac{p(x|z) p(z)}{p(x)} \\ p(x) = \int_{\mathcal{Z}} p(x|z) p(z) dz \end{cases}$$

- En *théorie*, on peut calculer $p(z|x)$ grâce à la loi de Bayes,
- En *pratique*, calcul non raisonnable du fait de l'intégrale au dénominateur

Cadre probabiliste et hypothèse de modélisation

Modèle hiérarchique : Nos observations x dépendent d'une variable latente z , i.e. d'une variable qu'on ne peut observer.



- Distribution *a priori* sur l'**espace latent** : $p(z) \equiv \mathcal{N}(0, 1)$;
- **Encodeur** \equiv Loi conditionnelle de z sachant x : $p(z|x)$;
- **Décodeur** \equiv Loi conditionnelle de x sachant z : $p(x|z) \equiv \mathcal{N}(f(z), \sigma^2)$ $f \in F, \sigma \in \mathbb{R}^+$

Theorem (Loi de Bayes)

$$\begin{cases} p(z|x) = \frac{p(x|z) p(z)}{p(x)} \\ p(x) = \int_{\mathcal{Z}} p(x|z) p(z) dz \end{cases}$$

- En *théorie*, on peut calculer $p(z|x)$ grâce à la loi de Bayes,
- En *pratique*, calcul non raisonnable du fait de l'intégrale au dénominateur

↝ **Approximation par inférence variationnelle.**

$$p(z|x) \longleftrightarrow q_x(z) \equiv \mathcal{N}(g(x), h(x)) \quad g \in G, h \in H.$$

Formulation de l'inférence variationnelle

Inférence variationnelle :

- Technique générale en statistiques pour **approcher des distributions complexes**
- **Idée** : Se donner une famille de **distributions paramétriques** (ici des lois gaussiennes) et chercher la meilleure approximation au sein de cette famille.
La plupart du temps, on choisit comme critère la **divergence de Kullback-Leibler**, optimisée par descente de gradient.

Formulation de l'inférence variationnelle

Inférence variationnelle :

- Technique générale en statistiques pour **approcher des distributions complexes**
- **Idée** : Se donner une famille de **distributions paramétriques** (ici des lois gaussiennes) et chercher la meilleure approximation au sein de cette famille.
La plupart du temps, on choisit comme critère la **divergence de Kullback-Leibler**, optimisée par descente de gradient.

Critère de régularisation :

$$(g^*, h^*) \in \operatorname{argmin}_{(g,h) \in G \times H} KL\left(q_x(z) \mid p(z|x)\right)$$

Formulation de l'inférence variationnelle

Inférence variationnelle :

- Technique générale en statistiques pour **approcher des distributions complexes**
- **Idée** : Se donner une famille de **distribtions paramétriques** (ici des lois gaussiennes) et chercher la meilleure approximation au sein de cette famille.
La plupart du temps, on choisit comme critère la **divergence de Kullback-Leibler**, optimisée par descente de gradient.

Critère de régularisation :

$$(g^*, h^*) \in \operatorname{argmin}_{(g,h) \in G \times H} KL\left(q_x(z) \mid p(z|x)\right)$$

Ou encore, en utilisant la loi de Bayes,

$$(g^*, h^*) \in \operatorname{argmax}_{(g,h) \in G \times H} \mathbb{E}_{Z \sim q_x} [\log p(x|Z)] - KL\left(q_x(z) \mid p(z)\right),$$

~~ **Compromis** entre maximiser la **vraisemblance** des observations
et le fait de rester **proche** de la distribution a priori.

Formulation de l'inférence variationnelle

Choix du décodeur optimal, i.e. comment déterminer f .

- **But** : Avoir le meilleur schéma encodeur-décodeur possible
 - ↝ Nous voulons choisir la fonction f qui maximise la log-vraisemblance de $x|z$ lorsque $z \sim q_x(z)$ défini Slide 48 : $q_x(z) \equiv \mathcal{N}(g(x), h(x))$

Autrement dit :

$$f^* \in \underset{f \in F}{\operatorname{argmax}} \mathbb{E}_{Z \sim q} [\log p(x|Z)] = \underset{f \in F}{\operatorname{argmax}} \mathbb{E}_{Z \sim q_x} \left[-\frac{\|x - f(Z)\|^2}{2\sigma} \right].$$

Formulation de l'inférence variationnelle

Choix du décodeur optimal, i.e. comment déterminer f .

- **But** : Avoir le meilleur schéma encodeur-décodeur possible
 - ↝ Nous voulons choisir la fonction f qui maximise la log-vraisemblance de $x|z$ lorsque $z \sim q_x(z)$ défini Slide 48 : $q_x(z) \equiv \mathcal{N}(g(x), h(x))$

Autrement dit :

$$f^* \in \underset{f \in F}{\operatorname{argmax}} \mathbb{E}_{Z \sim q} \left[\log p(x|Z) \right] = \underset{f \in F}{\operatorname{argmax}} \mathbb{E}_{Z \sim q_x} \left[-\frac{\|x - f(Z)\|^2}{2\sigma} \right].$$

- **Fonction objectif** :

$$(f^*, g^*, h^*) \in \underset{(f,g,h) \in F \times G \times H}{\operatorname{argmax}} \mathbb{E}_{Z \sim q_x} \left[-\frac{\|x - f(Z)\|^2}{2\sigma} \right] - KL(q_x(z) \mid p(z))$$

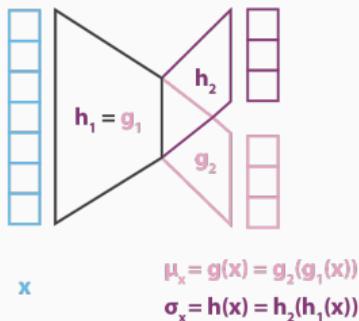
Formulation de l'inférence variationnelle



■ Fonction objectif :

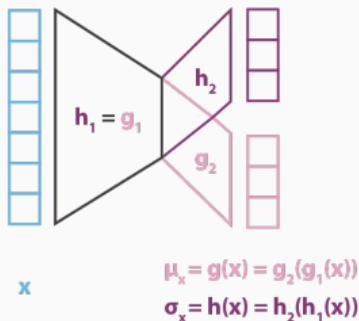
$$(f^*, g^*, h^*) \in \underset{(f,g,h) \in F \times G \times H}{\operatorname{argmax}} \underbrace{\mathbb{E}_{Z \sim q_x} \left[-\frac{\|x - f(Z)\|^2}{2\sigma} \right]}_{\text{Erreur de reconstruction}} - \underbrace{KL(q_x(z) \mid p(z))}_{\text{Critère de régularisation}}$$

Ré-intégrer les réseaux neuronaux dans le modèle : L'encodeur



- En fait d'ensemble de fonctions, on exprime f , g et h sous forme de **réseaux neuronaux**.
Ainsi, F , G et H correspondent aux familles de fonctions définies par les architectures de réseaux et l'optimisation se fait sur les **paramètres** de ces réseaux.

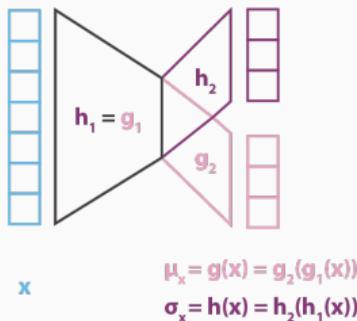
Ré-intégrer les réseaux neuronaux dans le modèle : L'encodeur



- En fait d'ensemble de fonctions, on exprime f , g et h sous forme de **réseaux neuronaux**.
Ainsi, F , G et H correspondent aux familles de fonctions définies par les architectures de réseaux et l'optimisation se fait sur les **paramètres** de ces réseaux.
- En *pratique*, g et h ne sont pas définis par deux réseaux totalement indépendants mais **partagent** une partie de leur **architecture** et de leurs **poids** :

$$g(x) = g_2 \circ g_1(x) \quad h(x) = h_2 \circ h_1(x) \quad g_1(x) = h_1(x)$$

Ré-intégrer les réseaux neuronaux dans le modèle : L'encodeur



- En fait d'ensemble de fonctions, on exprime f , g et h sous forme de **réseaux neuronaux**.
Ainsi, F , G et H correspondent aux familles de fonctions définies par les architectures de réseaux et l'optimisation se fait sur les **paramètres** de ces réseaux.

- En *pratique*, g et h ne sont pas définis par deux réseaux totalement indépendants mais **partagent** une partie de leur **architecture** et de leurs **poids** :

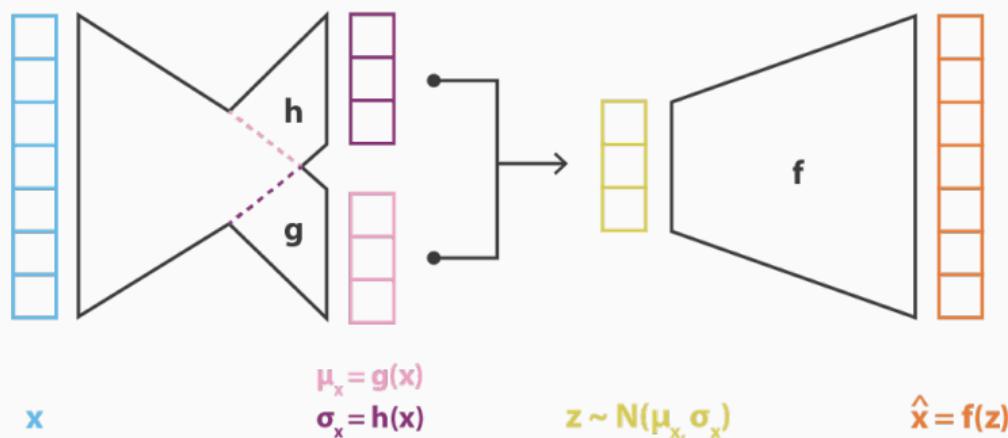
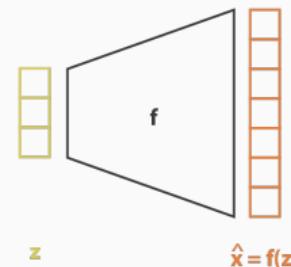
$$g(x) = g_2 \circ g_1(x) \quad h(x) = h_2 \circ h_1(x) \quad g_1(x) = h_1(x)$$

- De plus, afin de simplifier le calcul et de réduire le nombre de paramètres, nous supposons que $h(x) = \sigma_x^2 Id$
 - Équivalent à supposer l'**indépendance des variables**,
 - En particulier, $h(x) = \sigma_x$ de la même taille que $g(x) = \mu_x$,
 - Attention !* Comme nous réduisons la famille de distributions considérées pour l'inférence variationnelle, l'approximation peut être **moins précise**.

Ré-intégrer les réseaux neuronaux dans le modèle : Le décodeur

Cette fois, la variance est fixée, égale à σ^2 .

L'architecture globale est alors obtenue en concaténant les parties encodeur et décodeur.

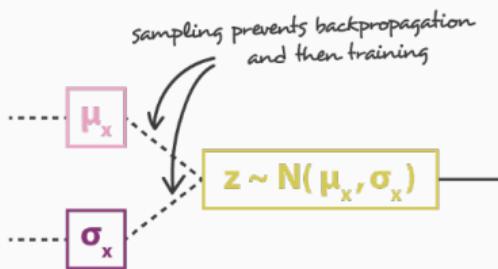


$$\text{loss} = C \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = C \|x - f(z)\|^2 + \text{KL}[N(g(x), h(x)), N(0, I)]$$

Entrainement \longleftrightarrow Astuce de la reparamétrisation

— no problem for backpropagation

----- backpropagation is not possible due to sampling



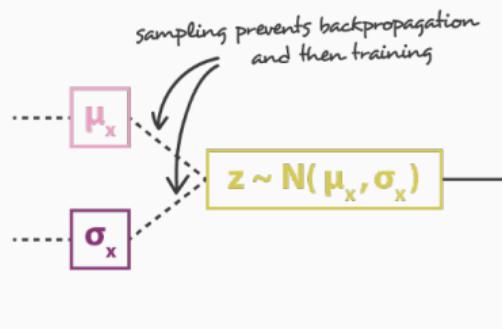
sampling without reparametrisation trick

Attention ! Le processus d'échantillonage peut bloquer la rétro-propagation du gradient !

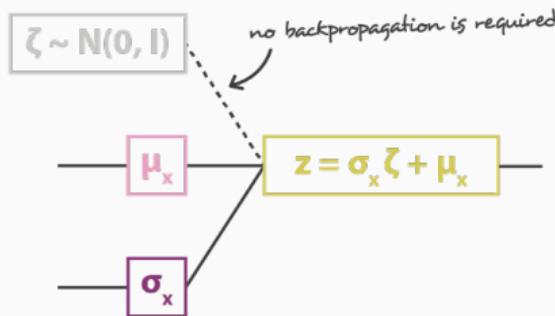
Entrainement \longleftrightarrow Astuce de la reparamétrisation

— no problem for backpropagation

----- backpropagation is not possible due to sampling



sampling without reparametrisation trick



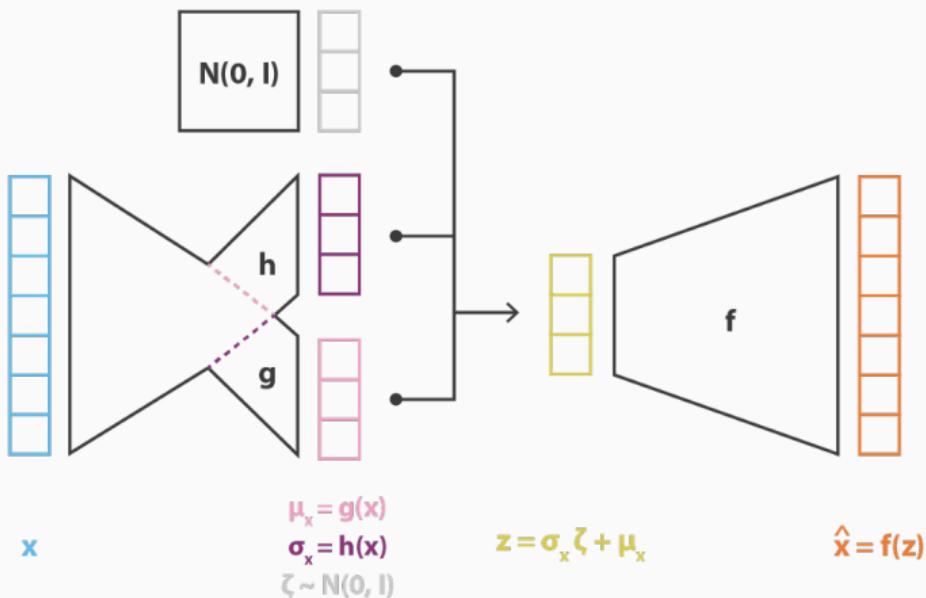
sampling with reparametrisation trick

Attention ! Le processus d'échantillonage peut bloquer la rétro-propagation du gradient !

~~> **Astuce de reparamétrisation** : Plutôt que de simuler $z \sim \mathcal{N}(\mu_x, \sigma_x^2)$, on fait :

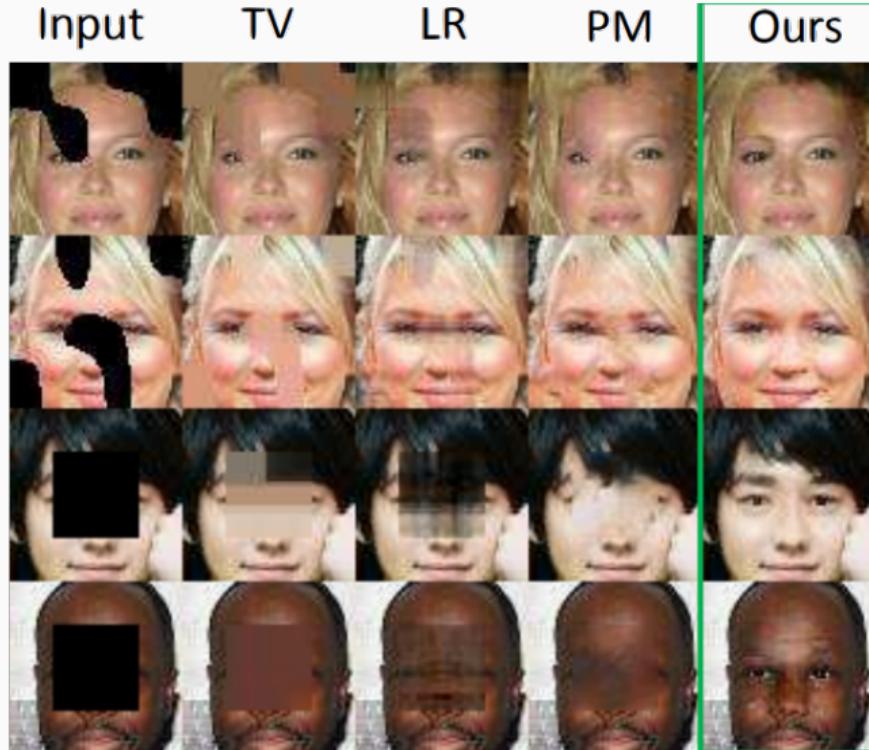
$$z = \mu_x + \sigma_x \cdot \zeta \quad \text{où} \quad \zeta \sim \mathcal{N}(0, 1).$$

Auto-encodeurs variationnels

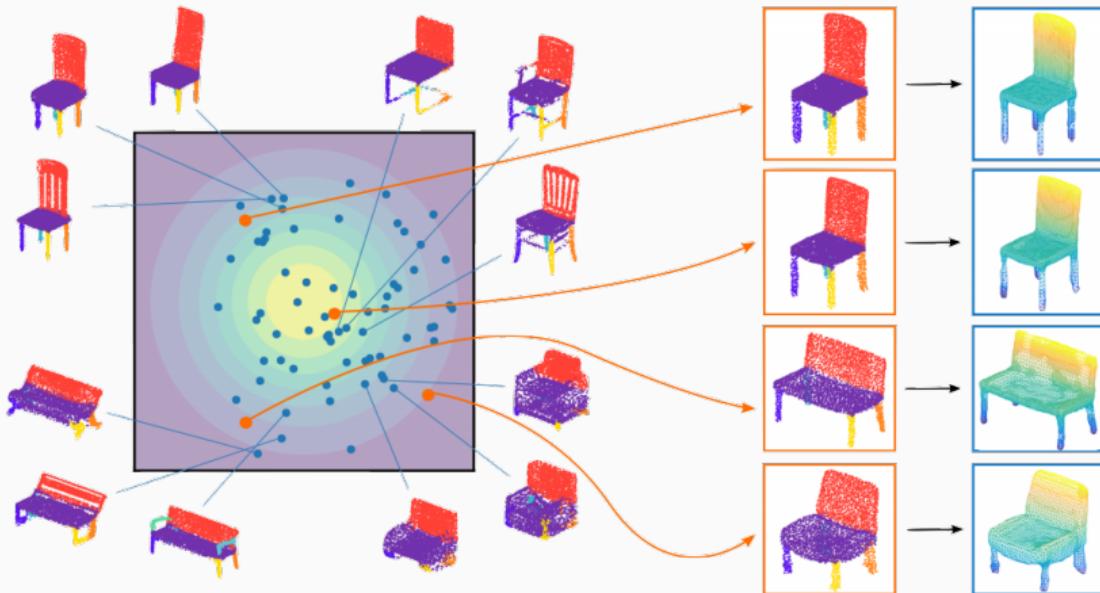


$$\text{loss} = C \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = C \|x - f(z)\|^2 + \text{KL}[N(g(x), h(x)), N(0, I)]$$

Auto-encodeurs variationnels – Applications



Auto-encodeurs variationnels – Applications



[Nash et al.] The shape variational autoencoder : A deep generative model of part-segmented 3D objects

Auto-encodeurs variationnels – Applications



Auto-encodeurs variationnels – Applications

VQ-VAE samples (left) and BigGAN deep samples (right) trained on ImageNet.



Plan du cours

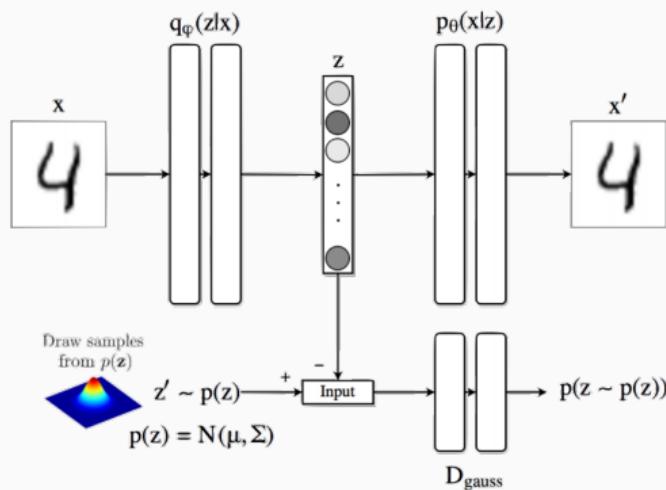
Introduction et Motivation

Réseaux antagonistes génératifs (GAN)

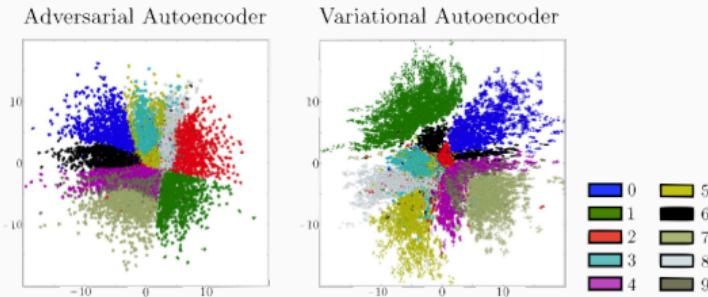
Auto-encodeurs variationnels (VAE)

Auto-encodeurs antagonistes (AAE)

Auto-encodeurs antagonistes (AAE)



[Makhzani et al.] Adversarial Autoencoders, ICLR 2016.



Manifold of Adversarial Autoencoder

0000005333335558882
00000053333355588222
0000005533333335882222
0000000533333338822222
6000000053333335882222
666000033333555882222
66666665555558882222
66666665555558882222
66666665555558882222
66666444445558822222
444499444444499911112
449999999999999911111
499999999999999911111
99777777999994411111
777777777777777711111

Vers de nouveaux auto-encodeurs variationnels

- Complexifier le modèle statistique sur l'espace latent (modèle de mélange par exemple),
- Utiliser des processus gaussien à la place des “simples” gausiennes pour gérer les séries temporelles,
- *Etc.*