Réseaux génératifs adversaires

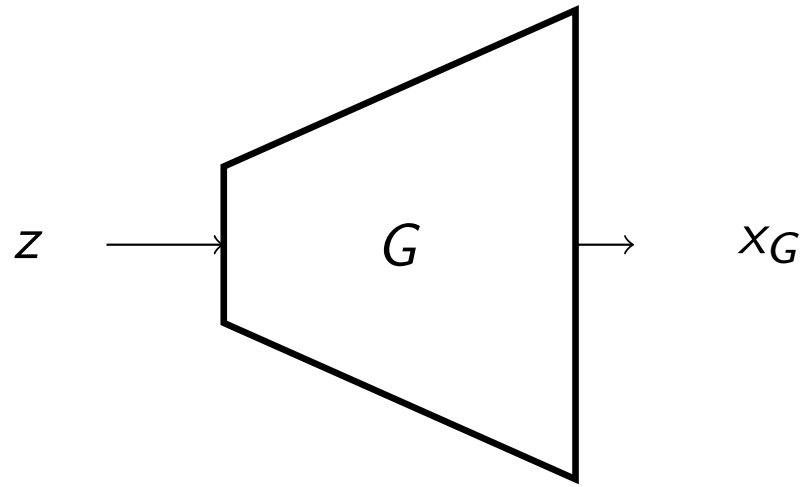
Apprentissage Profond

A. Carlier

2026

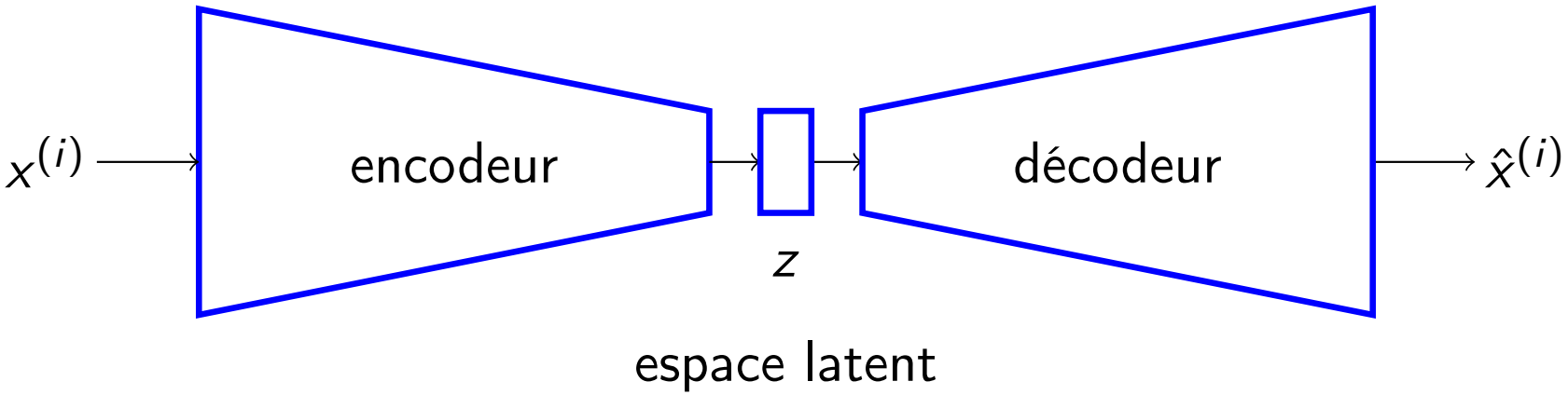
Générateur

Dans ce qui suit, nous appellerons **Générateur** un réseau de neurones dont la sortie est homogène aux données que l'on veut générer.



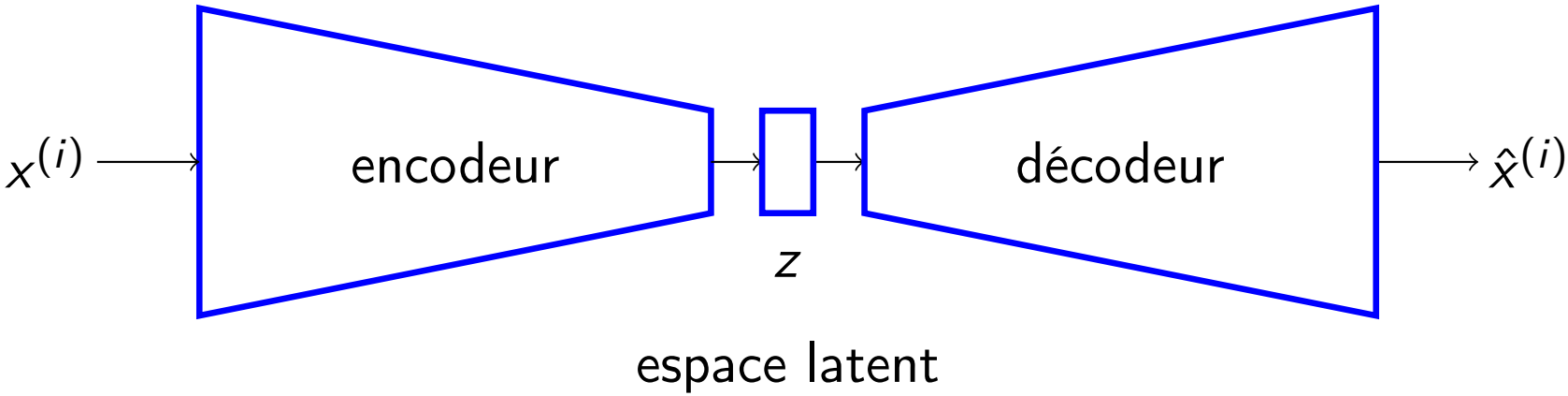
Le générateur a pour but d'apprendre à échantillonner la distribution $p_{\text{données}}$ à partir d'échantillons d'une distribution simple (typiquement, $z \sim \mathcal{N}(0, 1)$).

Auto-encodeurs

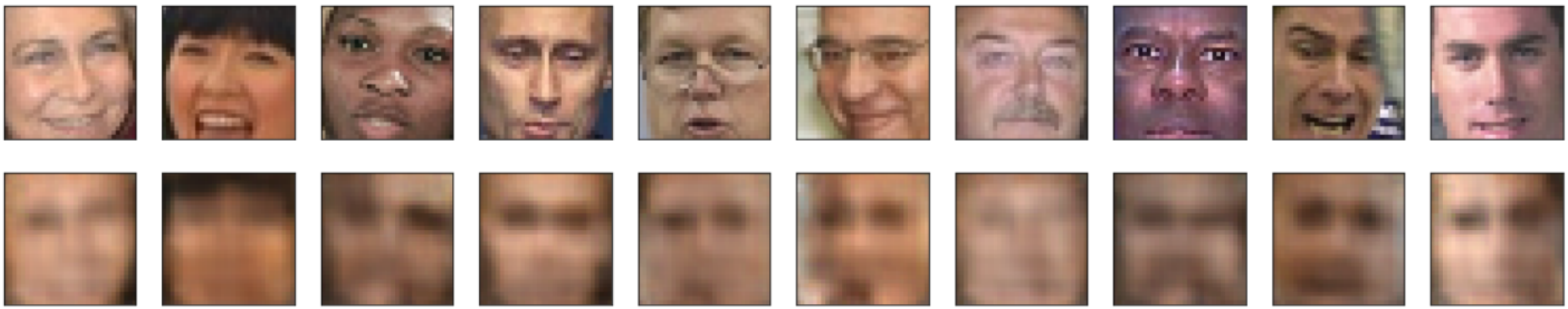


Les auto-encodeurs forment une classe d'algorithmes d'apprentissage non-supervisé, qui apprennent une représentation intéressante des données en minimisant une erreur de reconstruction.
On peut voir le décodeur comme un modèle génératif, qui apprend à (re)créer une donnée à partir de sa représentation latente.

Auto-encodeurs



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



Qualité d'exemples générés

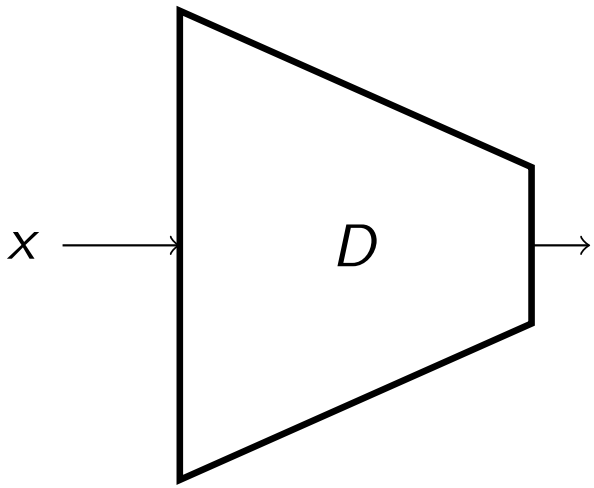
Une question est fondamentale dans le domaine des modèles génératifs : **comment évaluer la qualité d'un échantillon généré** ?



(a) (b) (c)

L'échantillon b) ressemble plus à l'échantillon a) (MSE de 0.0387) que l'échantillon c) (MSE de 0.2693), au sens de l'erreur quadratique moyenne.

Discriminateur

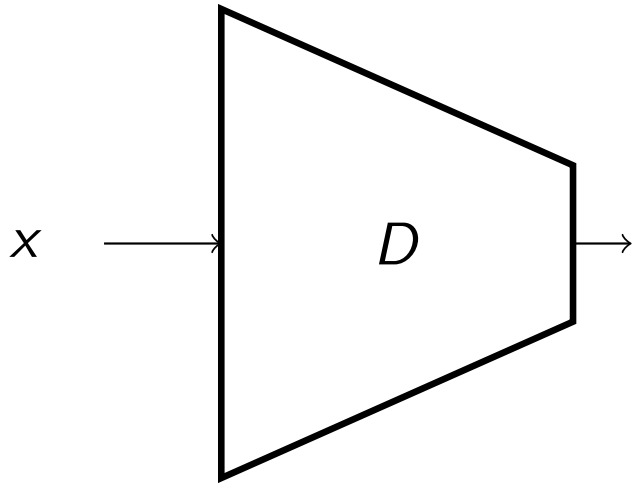


Donnée réelle ou donnée générée ?

(1) (0)

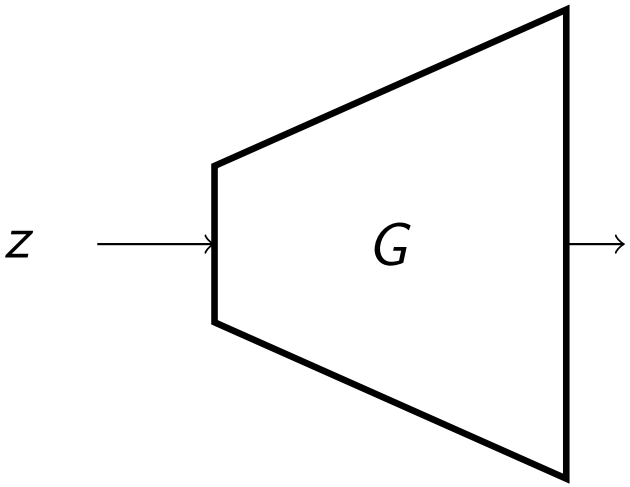
On définit un classifieur binaire, que l'on appellera dans la suite **discriminateur** D, dont le but est de déterminer si la donnée x en entrée est une donnée réelle, ou si elle a été générée par un modèle.

Réseaux génératifs adversaires



Objectif

minimiser $D(G(z))$

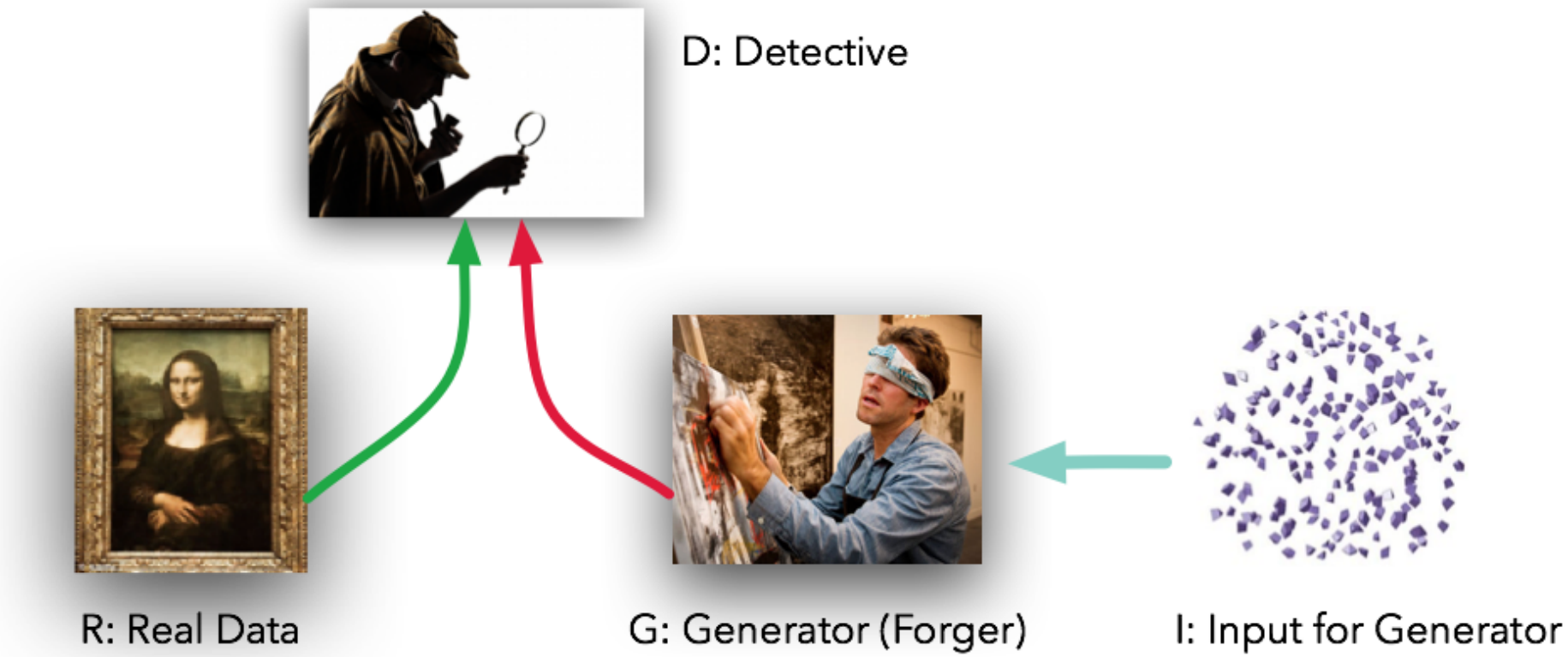


minimiser $1 - D(G(z))$

Le générateur et le discriminateur ont des objectifs antagonistes.

Réseaux génératifs adversaires

Une analogie :



Réseaux génératifs adversaires

- 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. On sait dès lors qu'il existe une situation de *minimax*, c'est-à-dire une stratégie pour le générateur et le discriminateur où chacun minimise le gain maximal de l'adversaire.

[Goodfellow et al.] Generative Adversarial Nets.

Réseaux génératifs adversaires : modèle

Fonction de coût du discriminateur (entropie croisée) :

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z)))$$

Fonction de coût du générateur :

$$J^{(G)} = -J^{(D)}$$

On cherche les paramètres $\theta^{(G)*}$ optimaux d'un générateur qui vérifie l'équation du minimax suivante :

$$\theta^{(G)*} = \arg\min_{\theta^{(G)}} \max_{\theta^{(D)}} -J^{(D)}$$

[Goodfellow et al.] Generative Adversarial Nets.

Réseaux génératifs adversaires : modèle

$$\theta^{(G)*} = \arg\min_{\theta^{(G)}} \max_{\theta^{(D)}} -J^{(D)}$$

La solution de ce problème est un équilibre de Nash $(\theta^{(D)}, \theta^{(G)})$, où :

- $\theta^{(G)}$ désigne les paramètres d'un générateur tel que

$$G(z) \sim p_{\text{données}}$$

- $\theta^{(D)}$ 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 !

[Goodfellow et al.] Generative Adversarial Nets.

Réseaux génératifs adversaires : entraînement

pour $n = 1$ **à** N_{iter}

// Entraînement du discriminateur

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

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

Mise à jour du discriminateur pour minimiser la perte :

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

// Entraînement du générateur

Mise à jour du générateur pour maximiser la perte :

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

fin pour

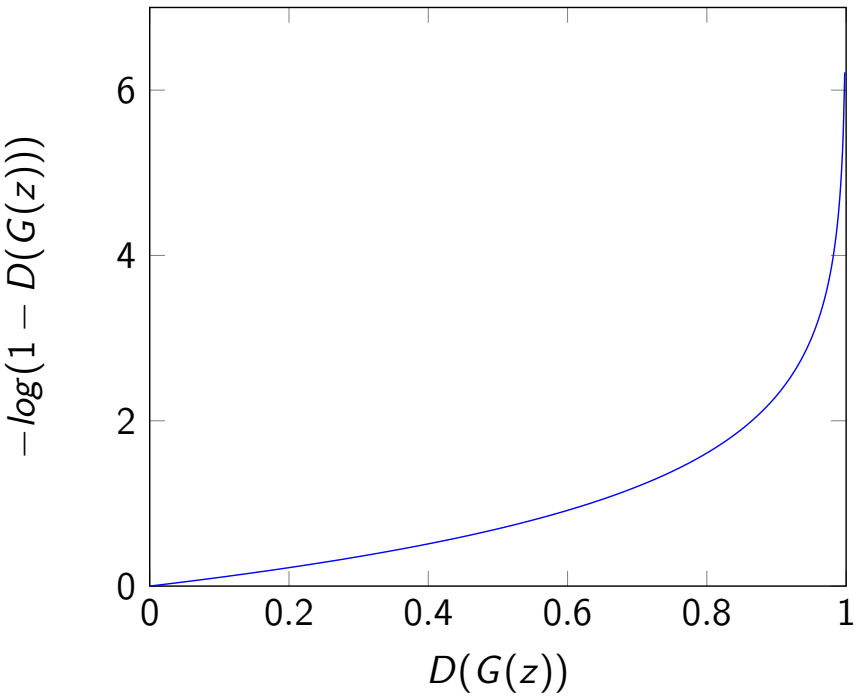
Figure – Descente de gradient simultanée pour l'entraînement d'un GAN.

Difficulté principale

En pratique, les réseaux génératifs adversaires sont **extrêmement** compliqués à entraîner.



L'évanescence des gradients

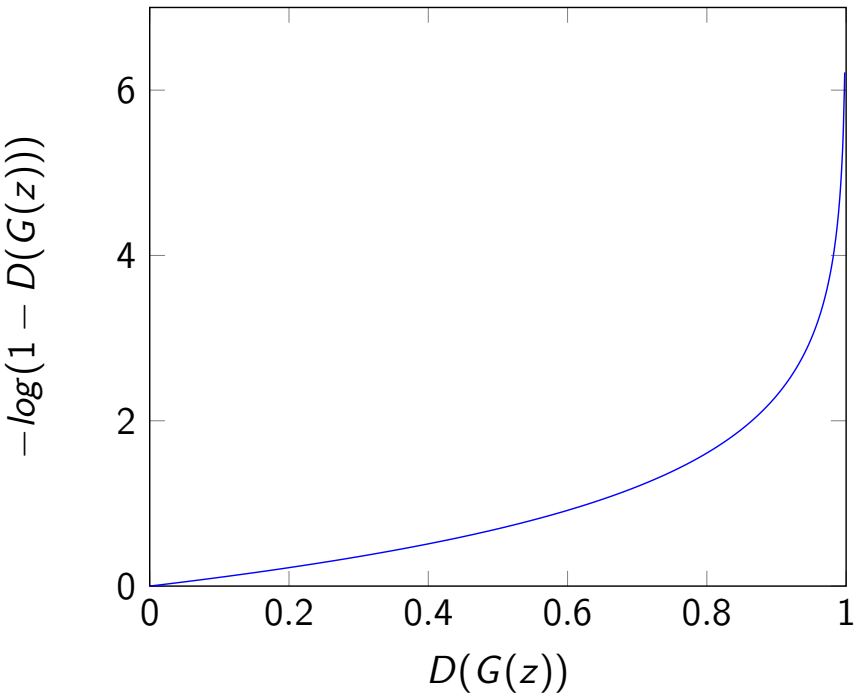


Si le générateur produit des échantillons trop irréalistes, le discriminateur peut rejeter ces échantillons avec une trop grande confiance.

$$G(z) \nsim p_{data} \implies D(G(z)) \approx 0 \implies \nabla(-log(1 - D(G(z)))) \approx 0$$

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

L'instabilité du générateur

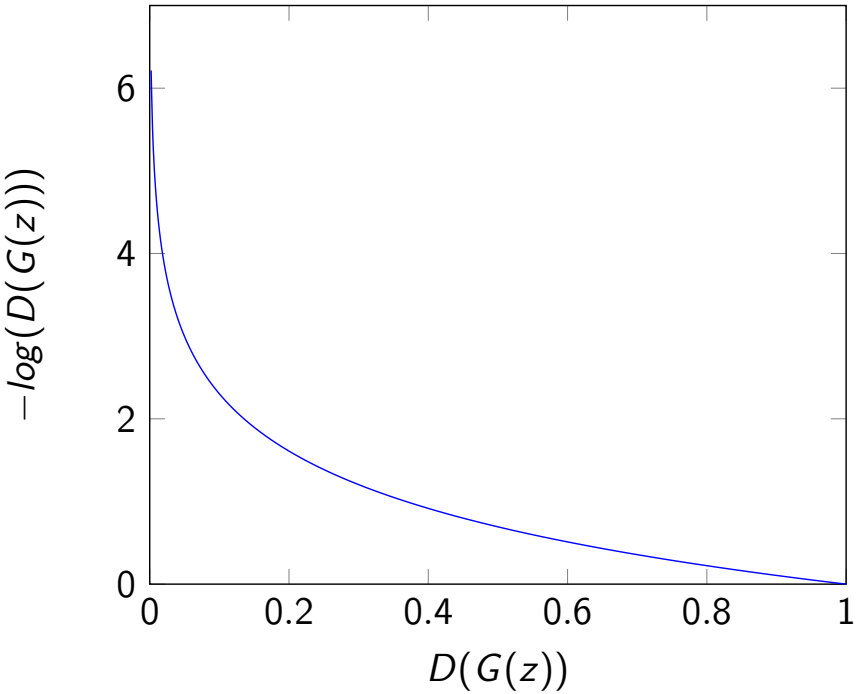


Si le générateur produit des échantillons trop réalistes, le discriminateur peut catégoriser ces échantillons comme réalistes avec une trop grande confiance.

$$G(z) \sim p_{data} \implies D(G(z)) \approx 1 \implies \nabla(-log(1 - D(G(z)))) \to -\infty$$

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

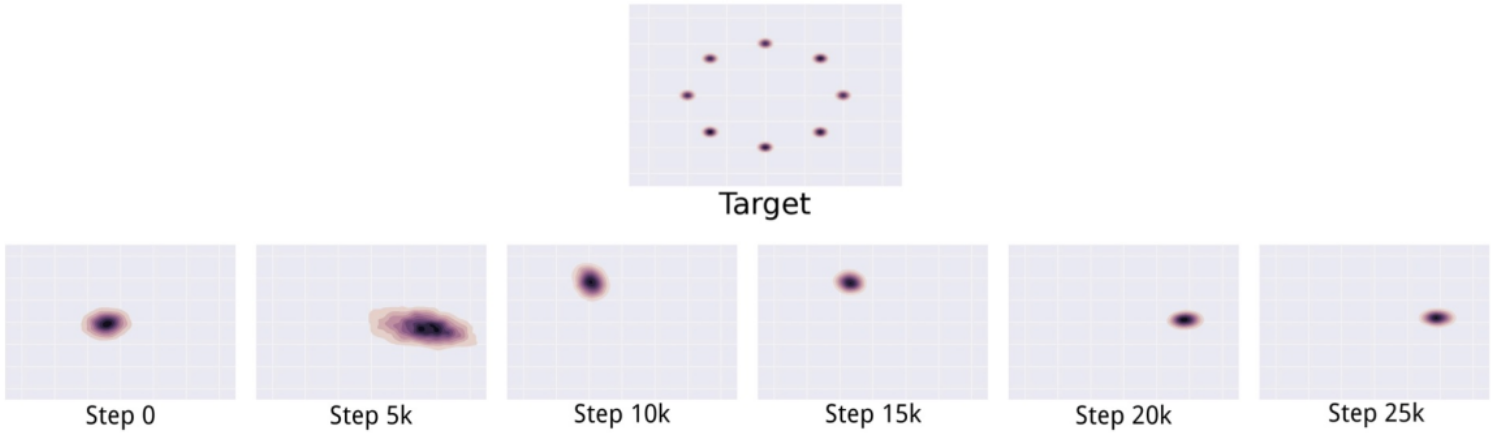
Problèmes d'entraînement



Il est possible de remplacer la maximisation de $-log(1 - D(G(z)))$ par une minimisation de $-log(D(G(z)))$, qui permet de faire évoluer plus rapidement le générateur au début de l'apprentissage et diminue l'instabilité.

[Goodfellow et al.] Generative Adversarial Nets.

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

La descente de gradient simultanée est un algorithme qui ne résoud pas nécessairement le problème

$$\theta^{(G)*} = \arg\min_{\theta^{(G)}}(\max_{\theta^{(D)}} -J^{(D)})$$

mais qui peut tout aussi bien résoudre le problème

$$\theta^{(G)*} = \max_{\theta^{(D)}}(\arg\min_{\theta^{(G)}} -J^{(D)})$$

Ce dernier problème conduit à un générateur qui cherche à produire l'échantillon jugé le plus réaliste par le discriminateur.

Pour éviter le *mode collapse*, il faut souvent effectuer une recherche exhaustive des hyperparamètres, ou utiliser d'autres types de loss (e.g. Wasserstein)

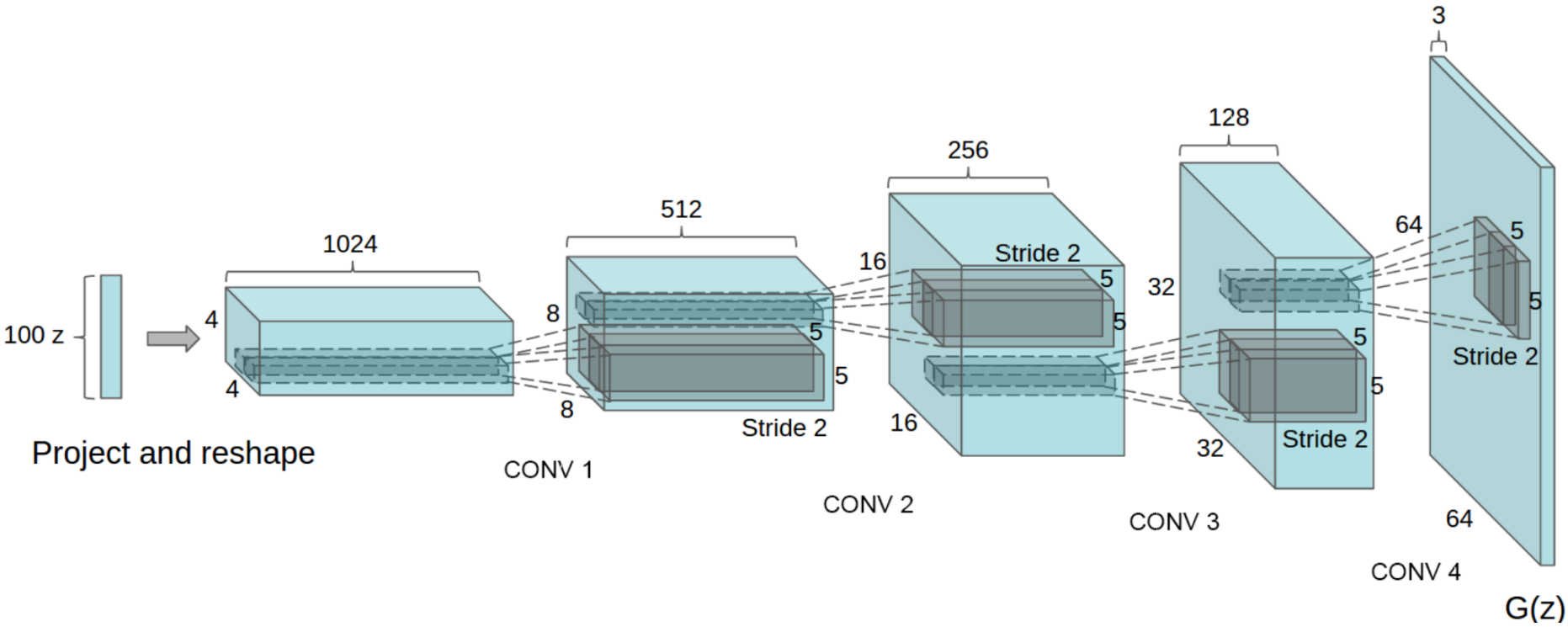
DCGAN

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

[Radford et al.] Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

DCGAN



- 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.

[Radford et al.] Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

DCGAN

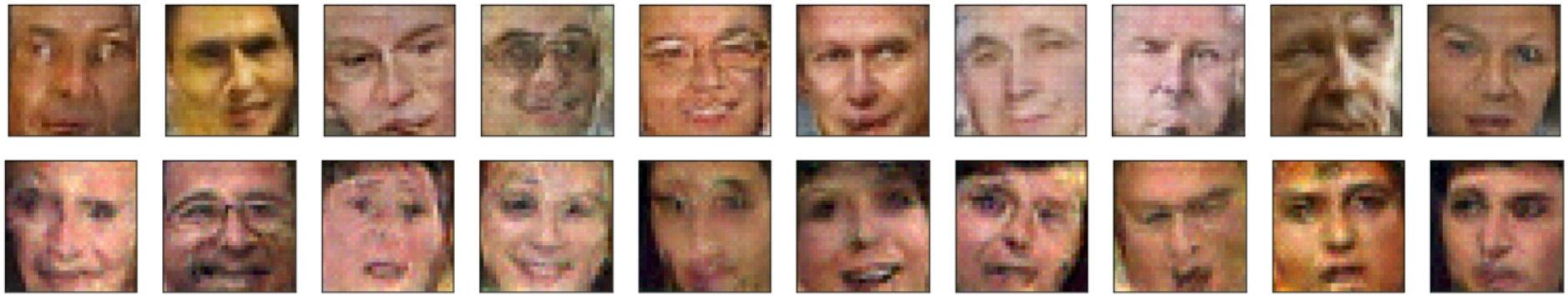
Quelques résultats tirés de l'article :



[Radford et al.] Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

DCGAN

Voici des résultats sur la base de données *Labelled Faces in the Wild* avec (première ligne) et sans (deuxième ligne) les recommandations de DCGAN :

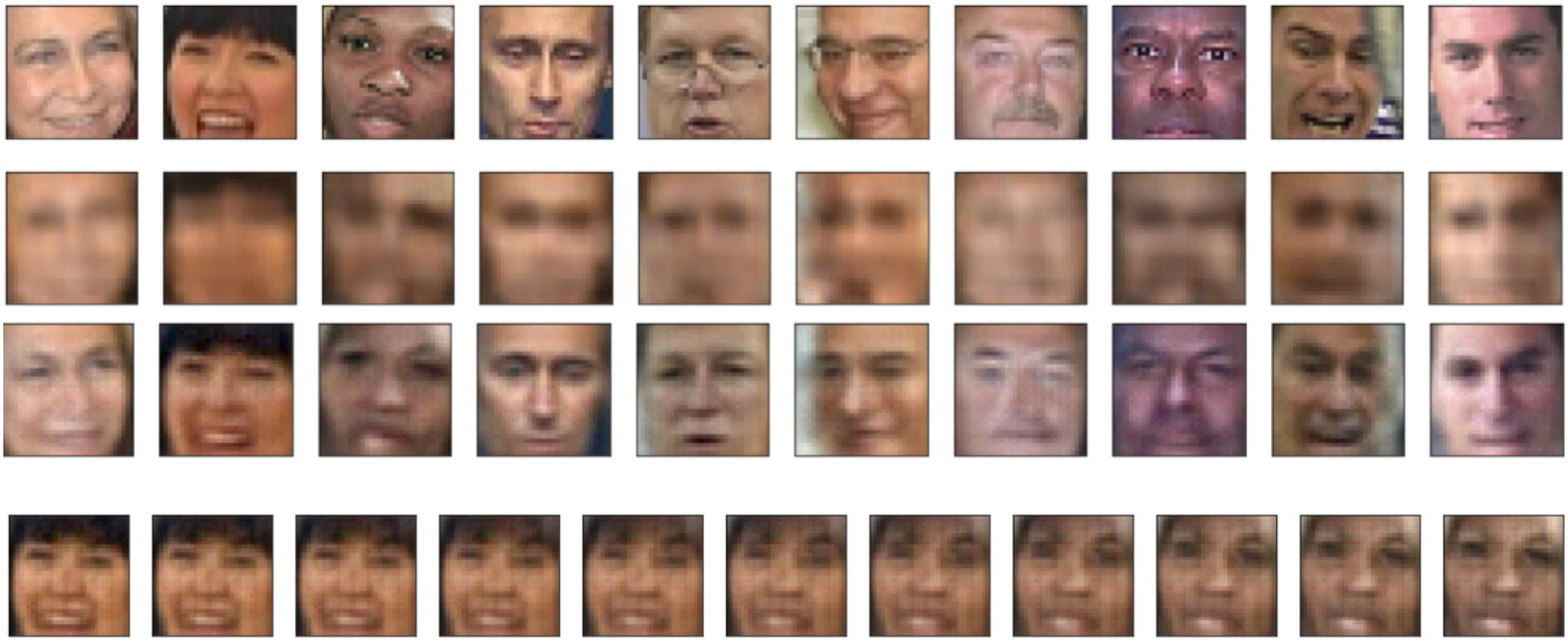


Ces résultats mettent en évidence une autre difficulté des GANs : l'impossibilité de quantifier la qualité des résultats.
Une possibilité est d'évaluer l'entropie des prédictions d'un réseau entraîné sur ImageNet pour estimer la qualité de la représentation générée dans une image, ce qui est implémenté par l'`Inception` score (amélioré avec la *FID*).

[Salimans et al.] Improved Techniques for Training GANs

DCGAN

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



Les GANs :

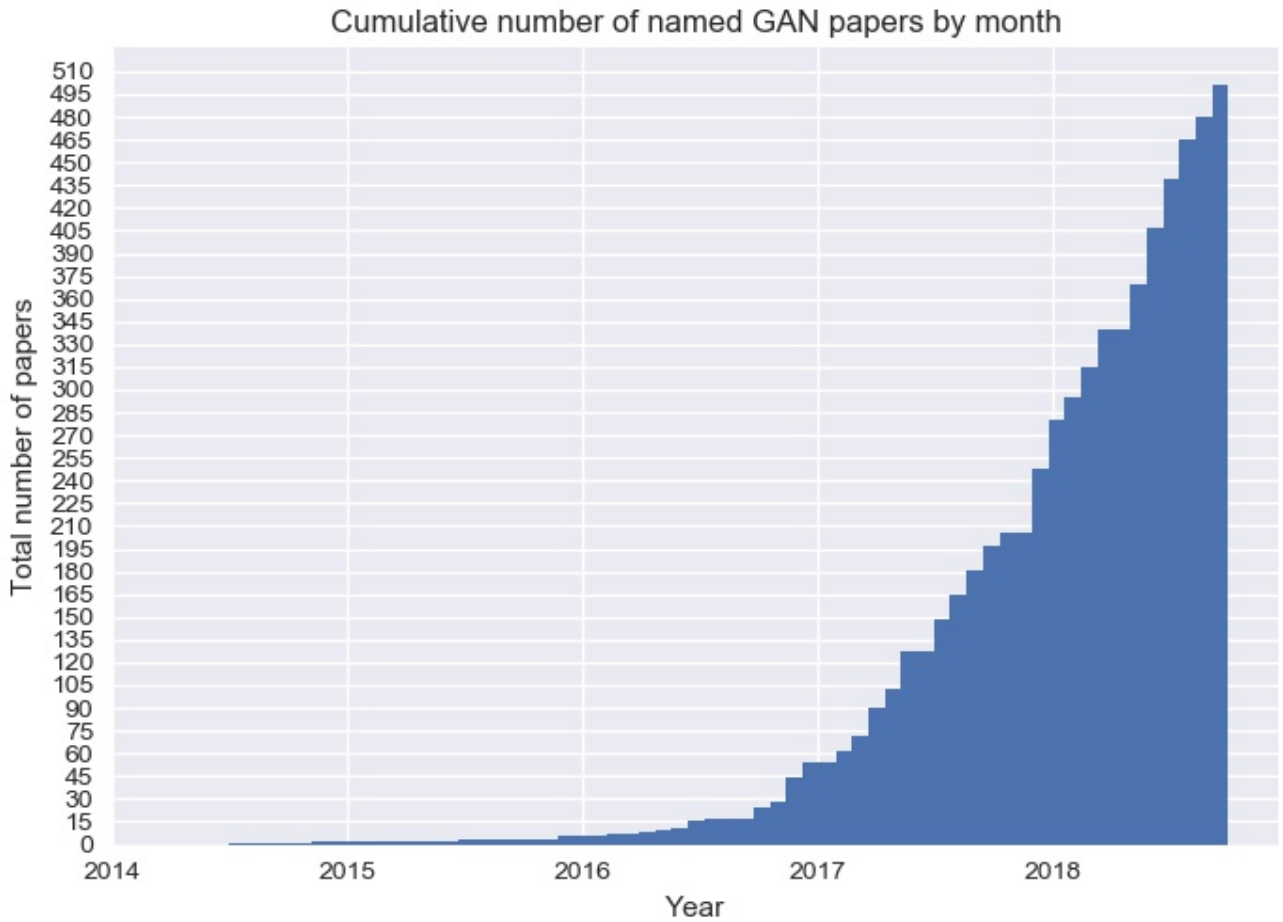
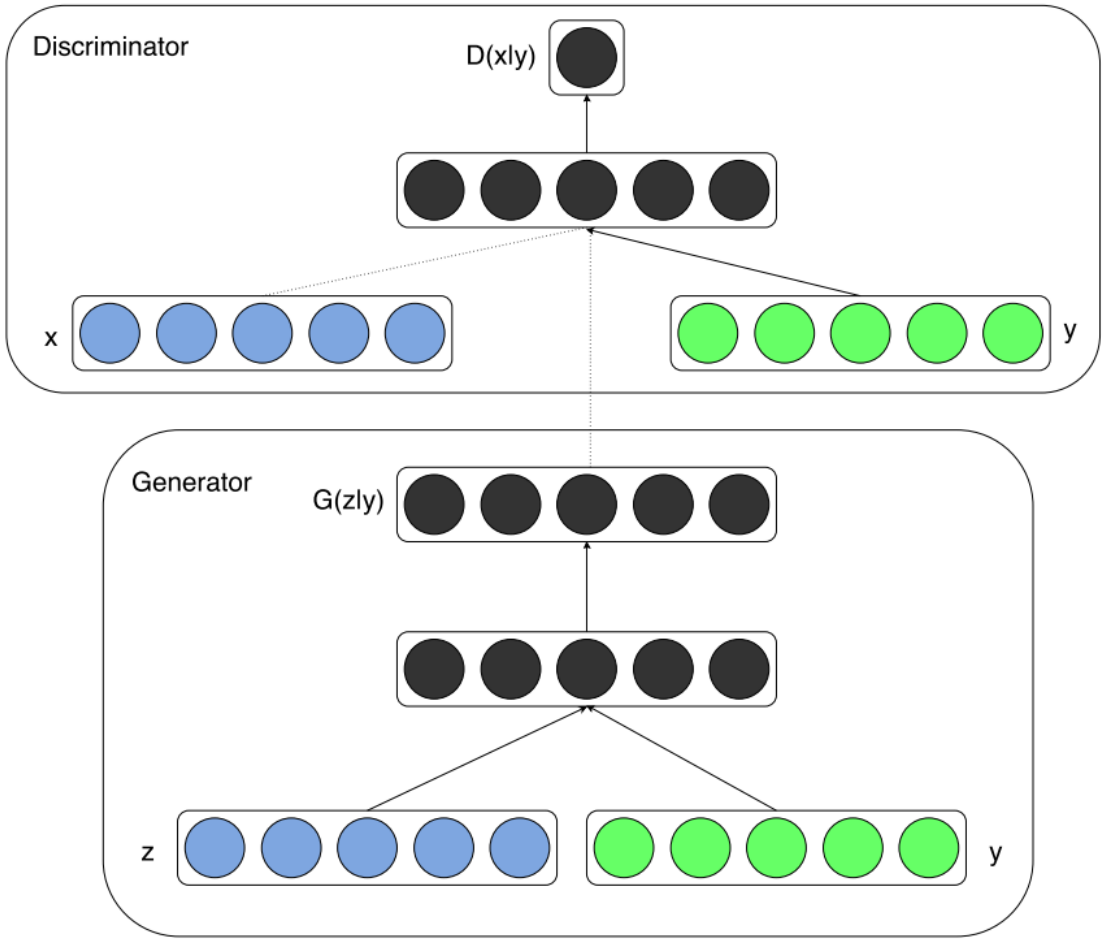


Image de `https://github.com/hindupuravinash/the-gan-zoo/`

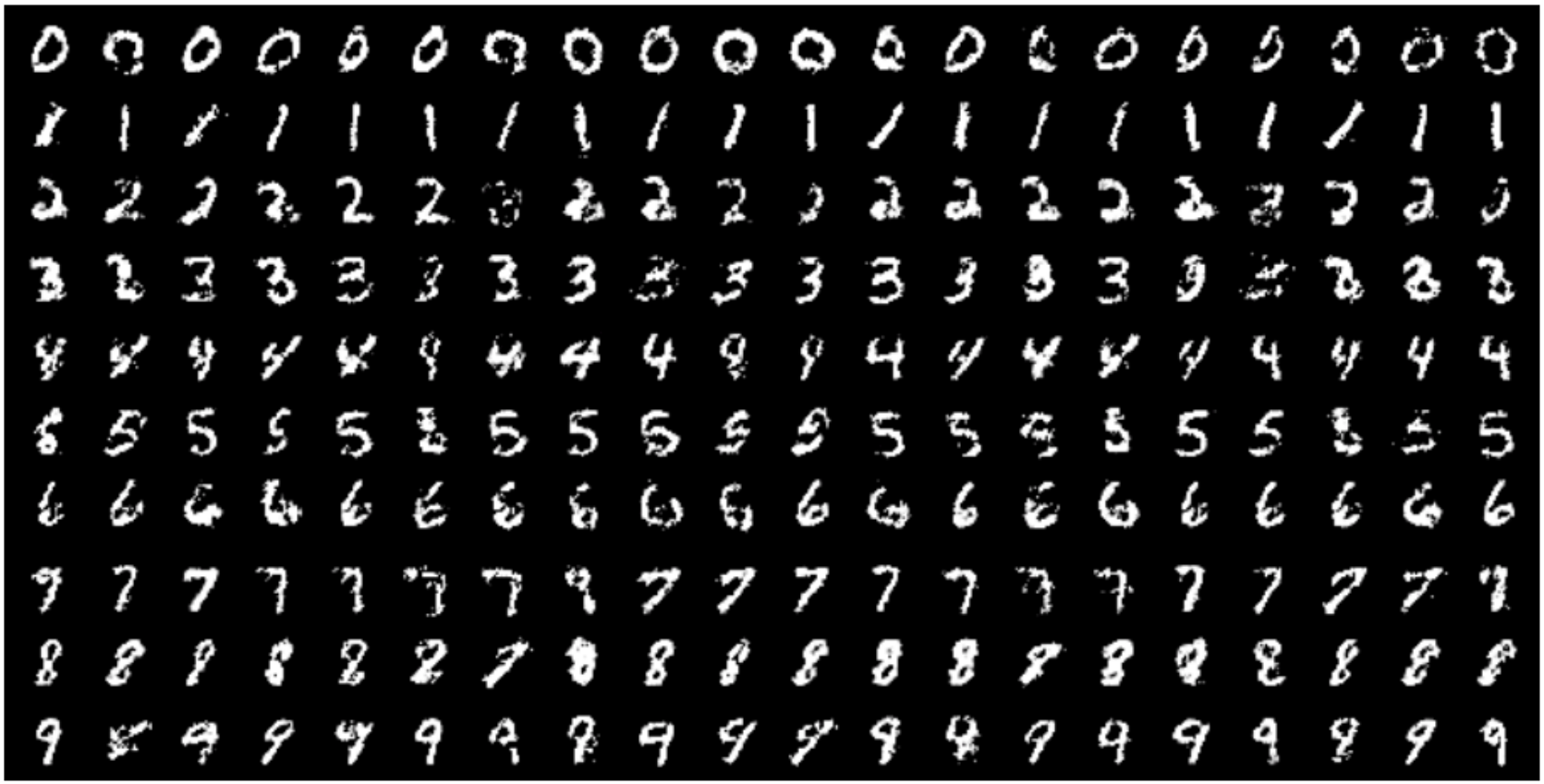
Conditional GAN



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

[Mirza et al.] Conditional Generative Adversarial Networks.

Conditional GAN



[Mirza et al.] Conditional Generative Adversarial Networks.

BigGAN

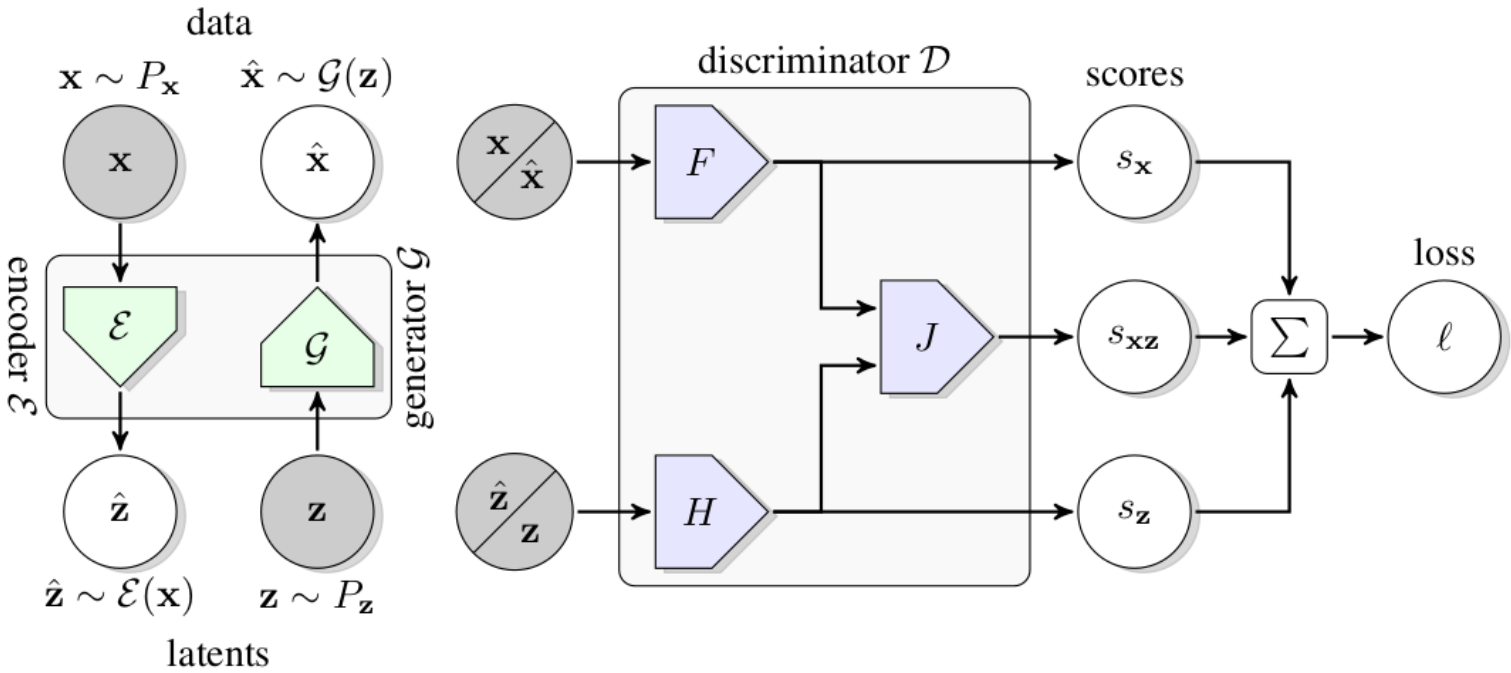
Produire des images de haute résolution, comme dans l'article BigGAN, nécessite des ajustements importants du processus d'apprentissage : augmentation de la *batch size*, doublement du nombre de filtres de convolution etc.



`https://thispersondoesnotexist.com/`

[Brock et al.] Large Scale GAN Training for High Fidelity Natural Image Synthesis.

Extraction de caractéristiques : BigBiGAN



Un autoencodeur et trois discriminateurs !

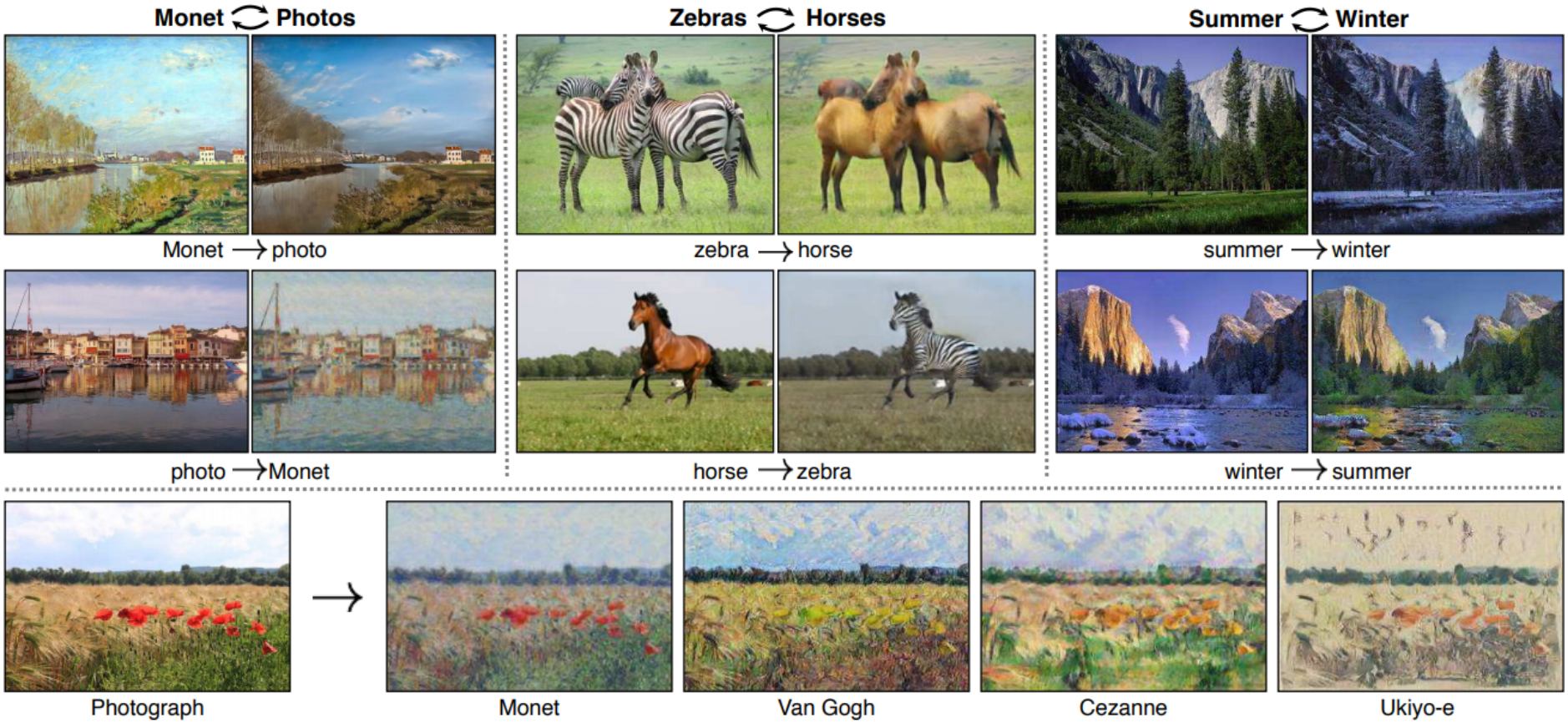
[Donahue et al.] Large Scale Adversarial Representation Learning

Extraction de caractéristiques : BigBiGAN



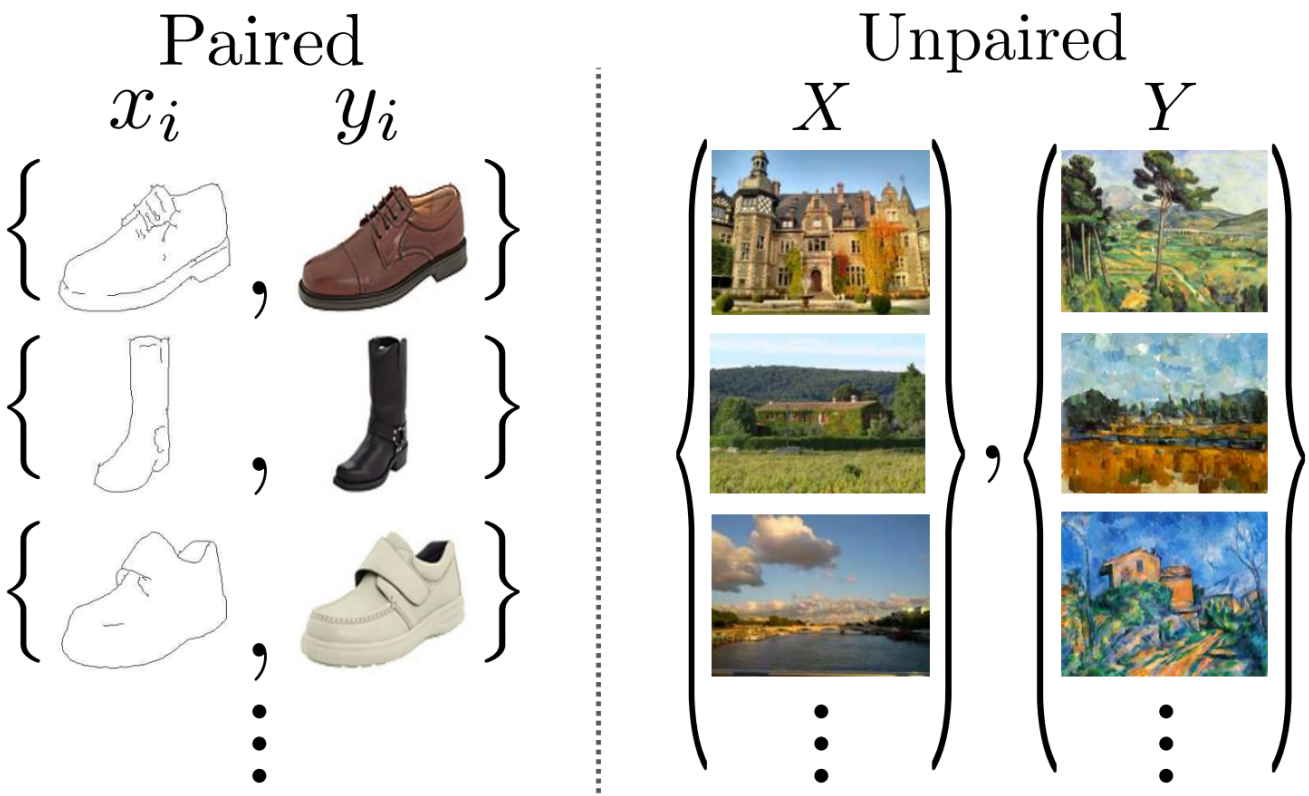
Method	Arch.	Param.	Top1
Supervised	R50	24	76.5
Colorization [65]	R50	24	39.6
Jigsaw [46]	R50	24	45.7
NPID [58]	R50	24	54.0
BigBiGAN [15]	R50	24	56.6
LA [68]	R50	24	58.8
NPID++ [44]	R50	24	59.0
MoCo [24]	R50	24	60.6
SeLa [2]	R50	24	61.5
PIRL [44]	R50	24	63.6
CPC v2 [28]	R50	24	63.8
PCL [37]	R50	24	65.9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

Un exemple d'application : l'adaptation de domaine



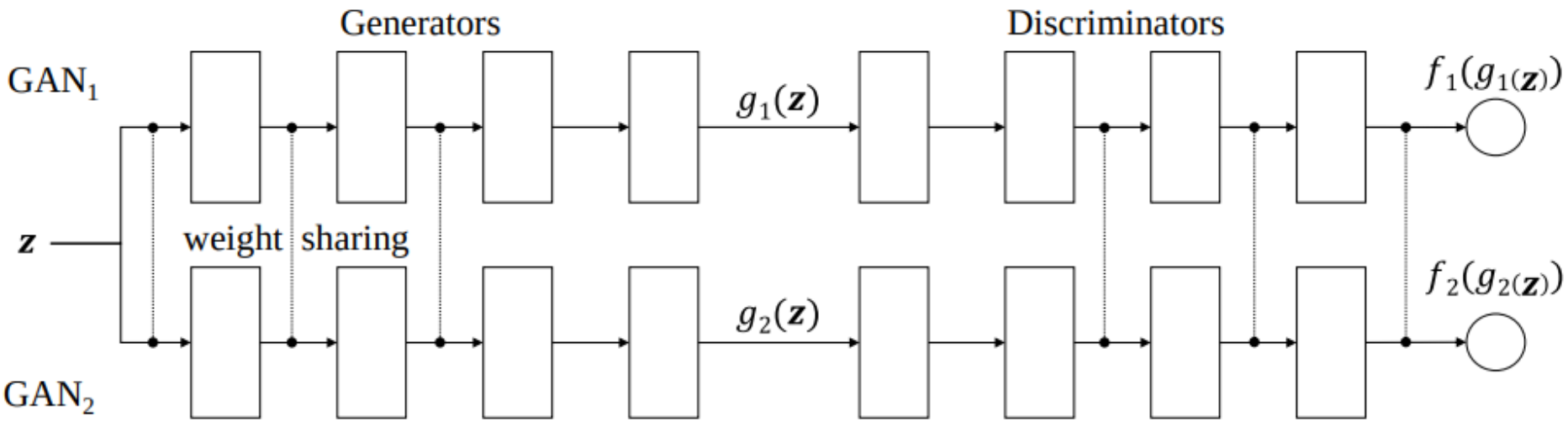
[Zhu et al.] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Adaptation de domaine supervisée ou non-supervisée



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

CoGAN



- 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.

[Liu et al.] Coupled Generative Adversarial Networks

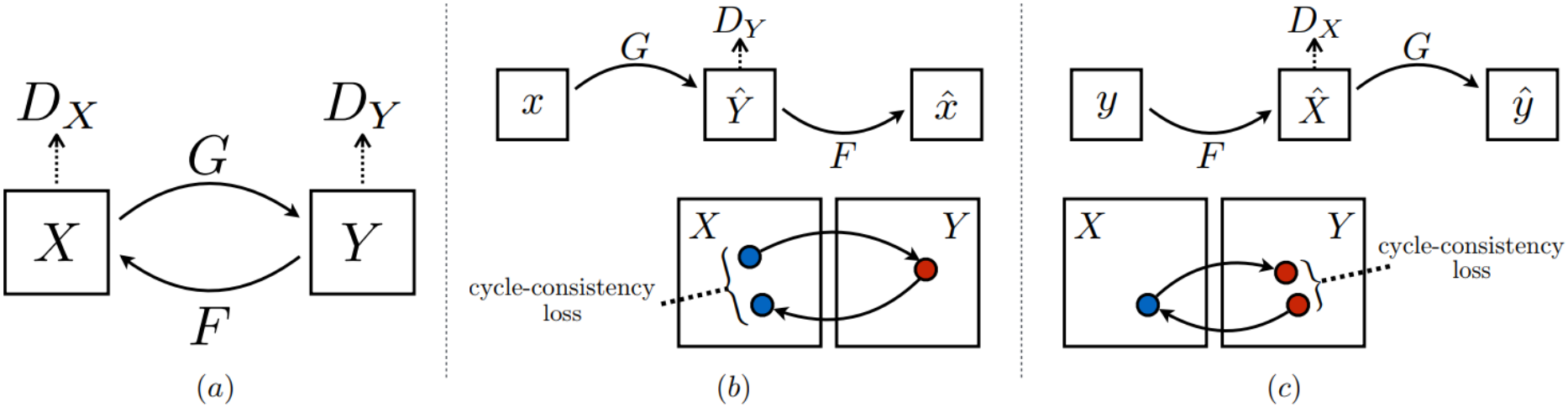
CoGAN



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

[Liu et al.] Coupled Generative Adversarial Networks

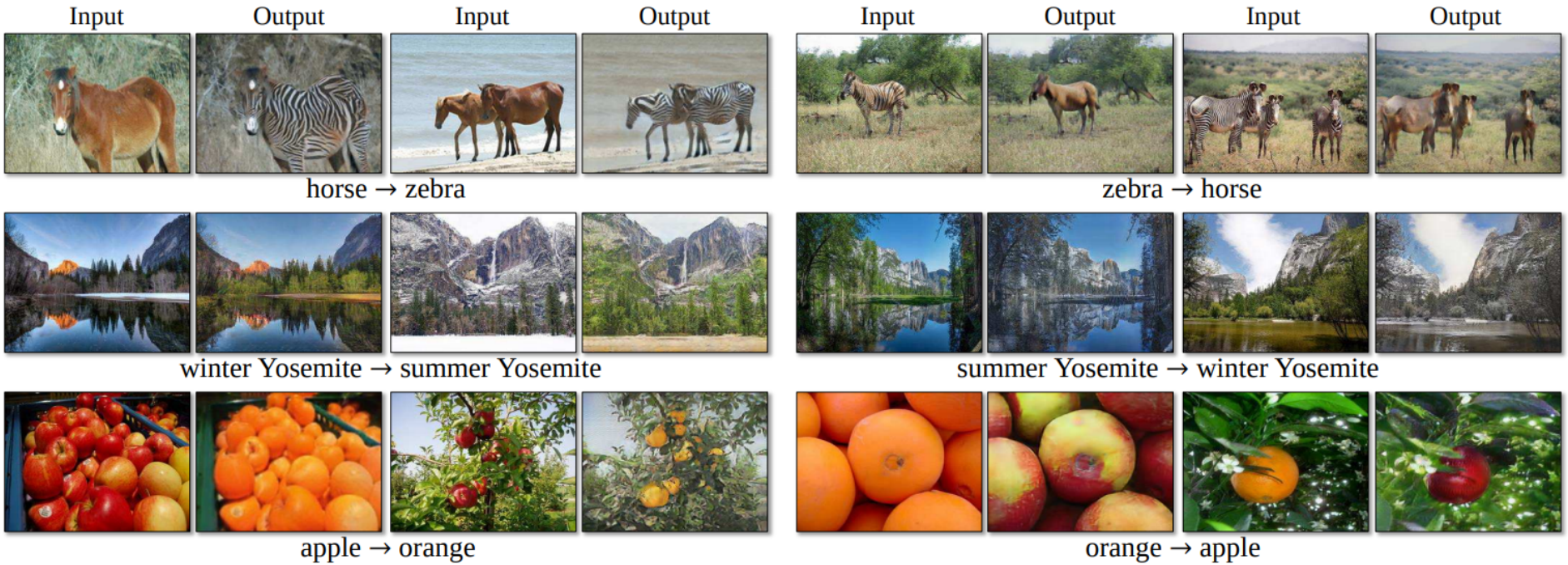
CycleGAN



- 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

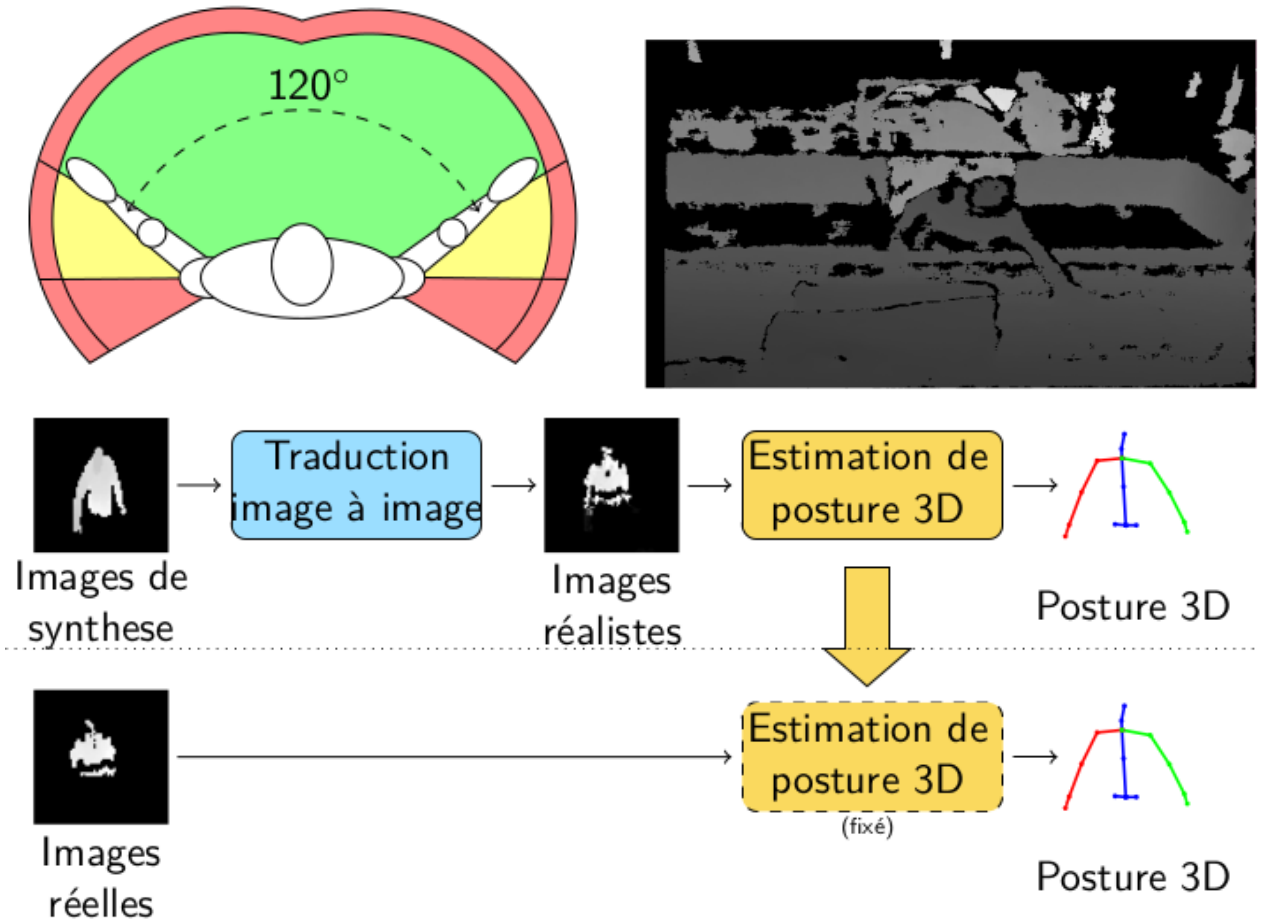
CycleGAN



Exemple de résultats du CycleGAN.

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

Un exemple d'application industrielle des GAN



[Blanc–Beyne] Estimation de posture 3D à partir de données imprécises et incomplètes : application à l'analyse d'activité d'opérateurs humains dans un centre de tri