6. Réseaux Génératifs Adverses (GAN)

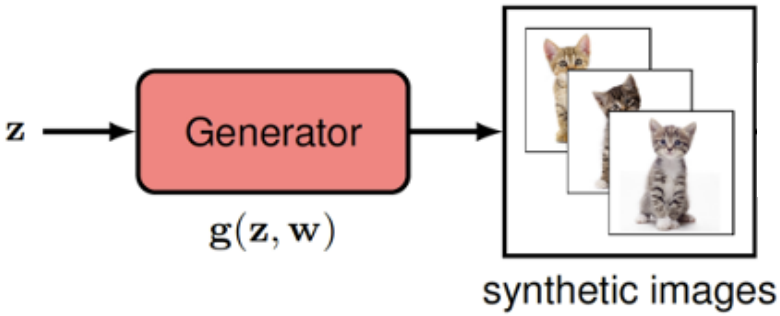
6.1 Position du problème

Apprendre la distribution des exemples $p(x|w)$ pour pouvoir en générer de nouveaux : de faux exemples

x un exemple

w les paramètres du modèle générateur

Le générateur n'est pas déterministe, et dépend de variables aléatoires cachées z, de sorte que les exemples générés ont un caractère aléatoire, ils sont tous différents et comportent un caractère imprévisible.



(d'après Bishop et al., 2024)

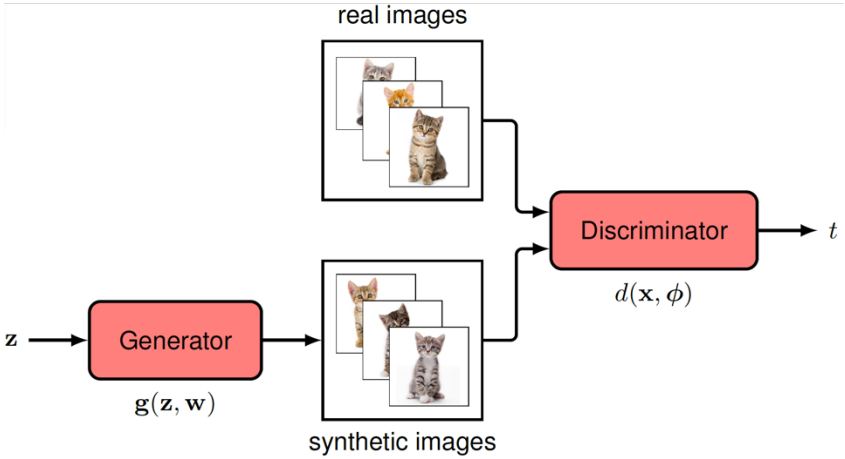
Souvent on souhaite générer des exemples d'une certaine classe c. Dans ce cas on doit apprendre les distributions conditionnellement à chaque classe, soit $p(x|w,c)$

Etant donné un ensemble d'exemples d'apprentissage $\{x_n\}$, comment déterminer les meilleurs paramètres w? Il existe une solution pour les mélanges Gaussiens (GMM) qui consiste à maximiser la vraisemblance $\sum_n \log\big(p(x_n|w)\big)$. Mais pour des modèles plus complexes comme des réseaux profonds on ne connaît pas de solution.

6. Réseaux Génératifs Adverses (GAN)

6.2 Entrainement à l'aide de réseaux adverses

la solution proposée par Ian Goodfellow en 2014 consiste à introduire un second réseau qui est utilisé uniquement en phase d'apprentissage.
Ce second réseau (paramétré par ϕ) est entrainé à distinguer (discriminer) les vrais exemples des faux, fournis par le générateur.



(d'après Bishop et al., 2024)

Les deux réseaux, générateur et discriminateur, sont entrainés l'un contre l'autre, d'où la dénomination de réseaux adverses générateurs (Generative Adversarial Networks ou GAN).

Le discriminateur apprend à discriminer les deux classes FRAI /FAU
Le générateur apprend à rapprocher ses exemples des vrais et à tromper le discriminateur

6. Réseaux Génératifs Adverses (GAN)

6.2 Entrainement à l'aide de réseaux adverses

Formulation du problème

on défnit la variable binaire $t = 1$ donnée réelle, $t = 0$ donnée synthétique

$N = N_{real} + N_{synth}$ le nombre d'exemples d'apprentissage

en général $N_{real} = N_{synth}$

Le discriminateur estime $P(t = 1) = d(x, \phi)$

pour cela il est entrainé avec une loss cross-entropy

$$E(w, \phi) = \frac{1}{N} \sum_{n=1}^{N} t_n log d(x_n, \phi) + (1 - t_n) log\big(1 - d(x_n, \phi)\big)$$

$$= \frac{1}{N_{real}} \sum_{n \in real} log d(x_n, \phi) + \frac{1}{N_{synth}} \sum_{n \in synth} log\big(1 - d(g(z, w), \phi)\big)$$

Le réseau peut donc être entrainé par descente de gradient

Mais le discriminateur cherche à minimiser l'erreur tandis que le générateur cherche à maximiser l'erreur du discriminateur sur les exemples qu'il génère.

La mise à jour des poids suit donc les deux équations suivantes

$$\Delta\phi = -\lambda \nabla_\phi E_n(w, \phi)$$

$$\Delta w = \lambda \nabla_w E_n(w, \phi)$$

$\nabla_\phi E_n(w, \phi)$ gradient de l'erreur par rapport aux paramètres du discriminateur

$\nabla_w E_n(w, \phi)$gradient de l'erreur par rapport aux paramètres du générateur

6. Réseaux Génératifs Adverses (GAN)

6.2 Entrainement à l'aide de réseaux adverses

Mise en œuvre de l'apprentissage adverse

En pratique on alterne des batch d'exemples synthétiques et des batch d'exemples réels

```
for epoch in range(max_epoch):
    for k in range(K_STEPS):
        generate a batch of m fake examples
        select a batch of m true examples
        train discriminator only on a batch of size 2m

    generate m fake examples
    train generator only
```

6. Réseaux Génératifs Adverses (GAN)

6.2 Entrainement à l'aide de réseaux adverses

Mise en œuvre de l'apprentissage

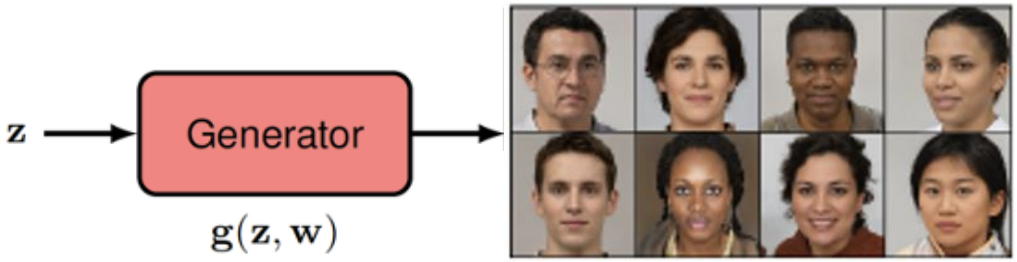
Si l'apprentissage converge correctement alors le discriminateur devient incapable de discriminer les vrais exemples des faux et il produit une sortie toujours identique

$$P(t_n = 1) = P(t_n = 0) = d(x_n, \phi) = 0.5$$

A la convergence on espère atteindre un équilibre de Nash où chaque joueur prédit parfaitement le jeu de l'autre.

En théorie des jeux l'équilibre de Nash est, d'un point de vue mathématique, un point fixe d'un processus où les joueurs, pris successivement, maximisent leur gain après avoir « observé » celui de leur prédécesseur — le « premier », quel qu'il soit, faisant d'abord son choix au hasard, puis quand son tour vient à nouveau, le fait après avoir observé celui du dernier de la « chaîne ». Il y a point fixe, équilibre, quand le « premier » qui choisit dans le processus, quel qu'il soit, puis tous ceux qui le suivent, ne modifie pas son choix au vu du choix fait par son « prédécesseur ».

A la converger

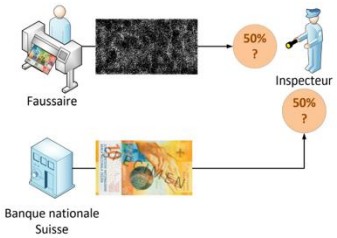


6. Réseaux Génératifs Adverses (GAN)

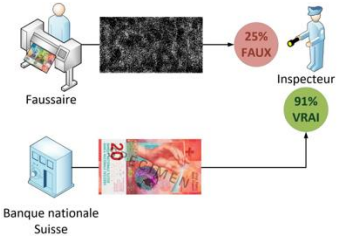
6.2 Entrainement à l'aide de réseaux adverses

exemple illustratif : *la fabrique de faux billets*

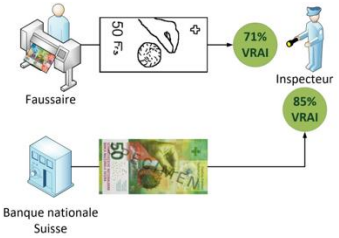
round 1 : le discriminateur discrimine aléatoirement



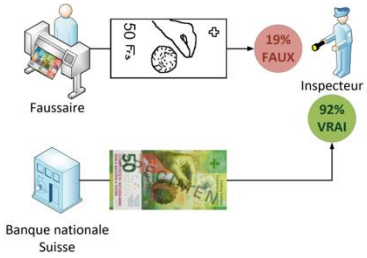
round 2 : le discriminateur détecte beaucoup de faux



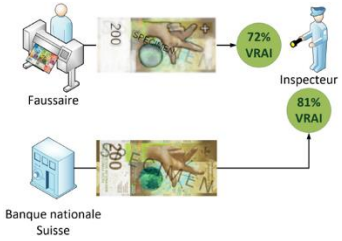
round 3 : le générateur progresse, et trompe le discr.



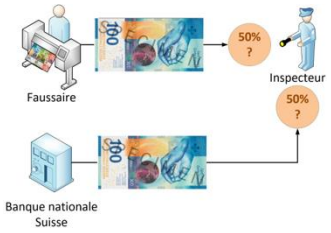
round 4 : le discriminateur progresse et diminue encore ses erreurs



round 5 : le générateur progresse à nouveau



round ??? : générateur et discriminateur ont progressé et convergé



(d'après Frédéric CHARBONNIER, 2021)

6. Réseaux Génératifs Adverses (GAN)

6.3 Problèmes rencontrés

L'apprentissage adverse est difficile à contrôler car la loss ne tend pas vers 0.

L'effondrement de mode : mode collaps
Ce phénomène survient lorsque le générateur se spécialise trop sur certains types d'exemples, mais ne génère pas tous les types d'exemples.

Exemple sur MNIST: au bout d'un certain temps, le générateur peut se spécialiser à générer des chiffres de la classe '3' parfaitement, mais le discriminateur ne s'en aperçoit pas parce que :

- les exemples 3 générés sont parfaits, donc le discriminateur n'apprend plus
- le discriminateur ne sait pas que la distribution des exemples générés ne couvre pas tout l'espace de représentation

Une solution:
- introduire un conditional GAN
- mais le phénomène peut tout de même survenir au sein d'une même classe, dans le cas où le générateur se spécialise sur une sous classe

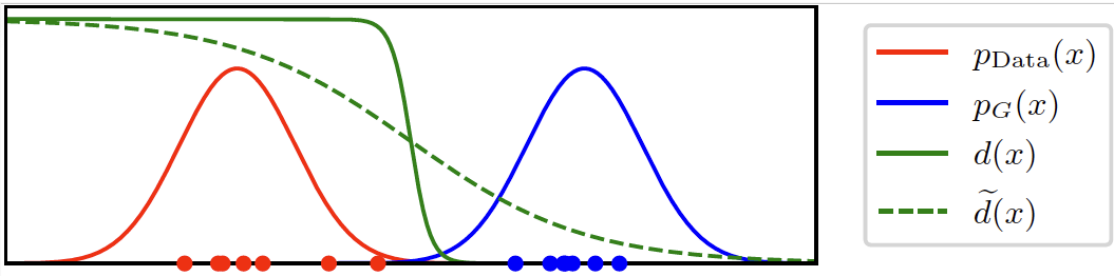
6. Réseaux Génératifs Adverses (GAN)

6.3 Problèmes rencontrés

Convergence lente
Exemple d'une situation où les distributions des vraies et des fausses données sont initialement très différentes.

Alors le discriminateur $d(x)$ est très performant dans les premières itérations, et le gradient de l'erreur est très faible, dans les deux zones. L'apprentissage est lent.

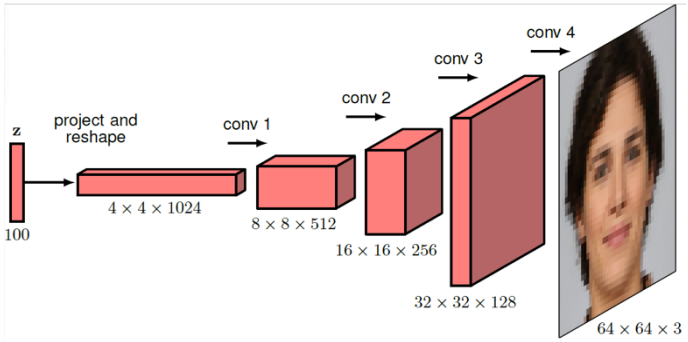
Une solution: utiliser un discriminateur moins performant (lissé) $\tilde{d}(x)$



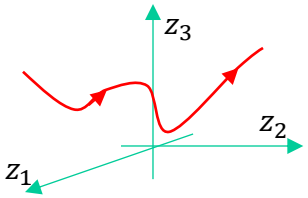
6. Réseaux Génératifs Adverses (GAN)

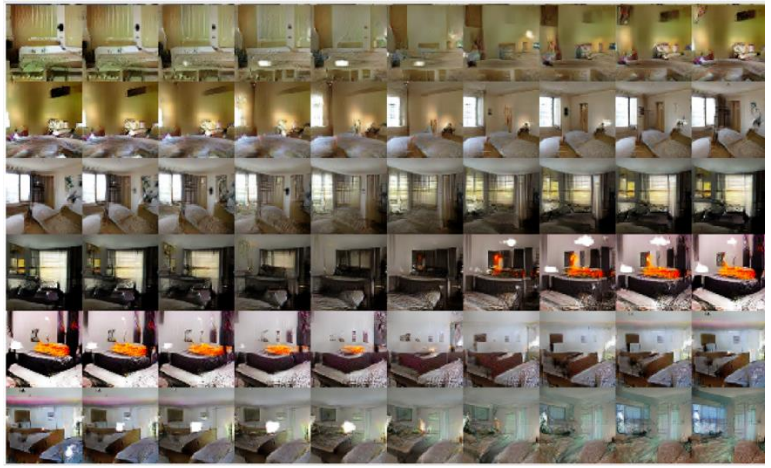
6.4 Exemples *(d'après Bishop et al., 2024)*

Architecture profonde convolutive + transpose convolution pour la génération d'images de visages



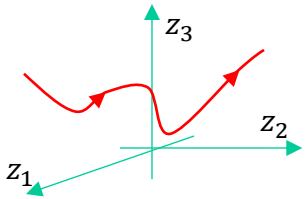
Cheminement dans l'espace latent d'un GAN entrainé sur un dataset de chambres





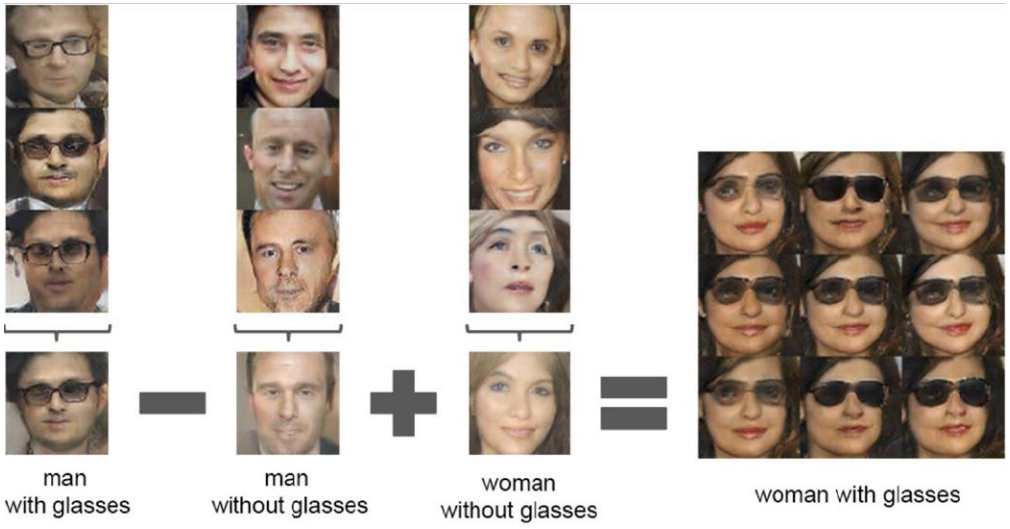
6. Réseaux Génératifs Adverses (GAN)

6.4 Exemples *(d'après Bishop et al., 2024)*
Interprétation de l'espace latent d'un GAN



Opération arithmétique sur

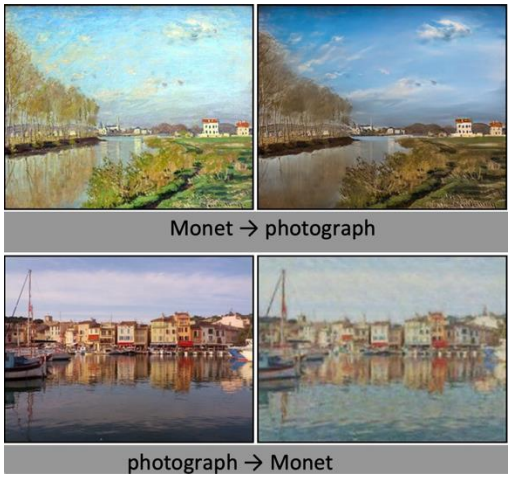
a) moyenne de la représentation latente z



b) sur les pixels

6. Réseaux Génératifs Adverses (GAN)

Cycle GAN : Transfert de style

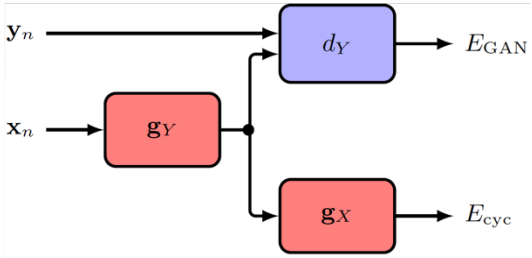


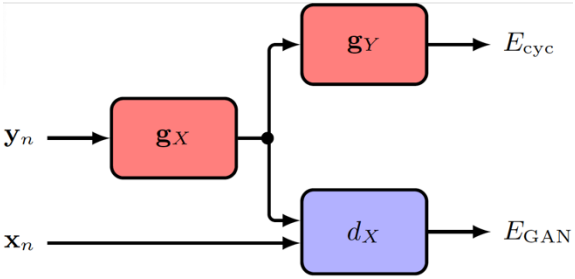
Apprendre les 2 transferts de style

style Photographe → style peinture de Monet

style peinture de Monet → style Photographe

On met en œuvre 2 générateurs g_x et g_y un pour chaque style
et 2 discriminateurs d_x et d_y



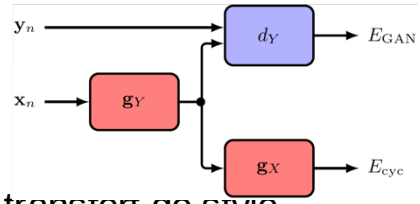


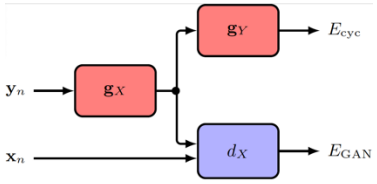
Si l'on entrainait chaque architecture avec une Loss de type GAN (E_{gan}), on parviendrait à générer des fausses peintures de Monet et des fausses photographies, mais on ne parviendrait par à réaliser le transfert de style souhaité.

(d'après Bishop et al., 2024)

6. Réseaux Génératifs Adverses (GAN)

Cycle GAN : Transfert de style





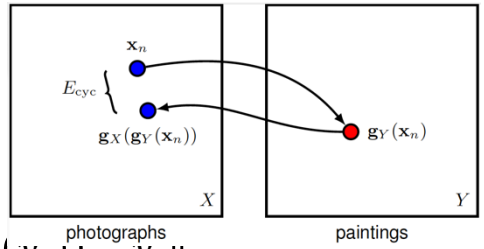
On introduit la los cycle GAN E_{cycl} qui va forcer le transfert de style

La transformation de la transformation doit ramener sur l'exemple de départ

$$g_x(g_y(x_n)) \approx x_n$$

et réciproquement

$$g_v(g_x(y_n)) \approx y_n$$



La loss permettant d'imposer le transfert de style est donc

$$E_{cycl}(w_x, w_y) = \frac{1}{N_x}\sum_{x_n \in X} \left\| g_x\left(g_y(x_n)\right) - x_n \right\| + \frac{1}{N_y}\sum_{x_n \in Y} \left\| g_y\left(g_x(y_n)\right) - y_n \right\|$$

la loss totale somme les deux loss GAN et la loss cycle GAN

$$E_{GAN}(w_x, \phi_x) + E_{GAN}(w_y, \phi_y) + \eta\, E_{cycl}(w_x, w_y)$$

(d'après Bishop et al., 2024)