

# RCP211 – Réseaux génératifs antagonistes

pix2pix – images HD – espaces latents

Nicolas Audebert `nicolas.audebert@lecnam.net`

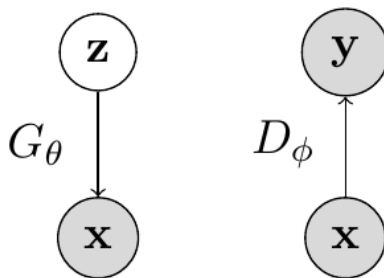
Conservatoire national des arts & métiers

24 novembre 2021

# Plan du cours

- 1 Rappels
- 2 Extensions des GAN conditionnels pour les images
- 3 GAN pour la synthèse d'images à haute résolution
- 4 Espace latent et contrôle des GAN

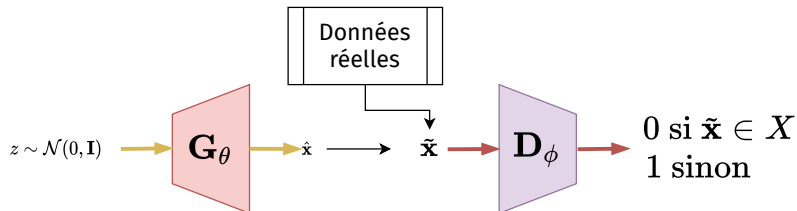
# Principe des GAN



Jeu minimax en opposition entre  $G$  et  $D$  :

- $\max_{\phi} \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}} \sim p(\hat{\mathbf{x}})} [\log(1 - \hat{\mathbf{x}})]$  : trouver les paramètres de  $D$  qui séparent les données réelles des données synthétiques.
- $\min_{\theta} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - D_{\phi}(G_{\theta}(\mathbf{z})))]$  : trouver les paramètres de  $G$  qui produisent des données qui trompent  $D$ .

# Schéma récapitulatif



# Plan du cours

- 1 Rappels
- 2 Extensions des GAN conditionnels pour les images
- 3 GAN pour la synthèse d'images à haute résolution
- 4 Espace latent et contrôle des GAN

# GAN conditionnel convolutif

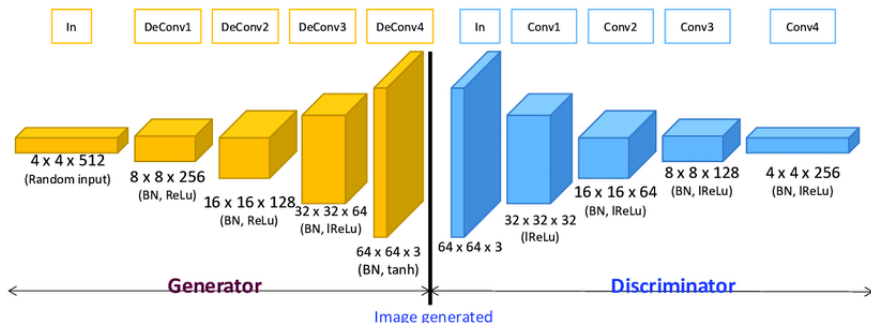


Schéma extrait de Suh et al., 2019

## Formule du GAN conditionnel

Jeu minimax à deux joueurs, conditionné sur une variable aléatoire  $y$  :

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

# GAN conditionnel convolutif

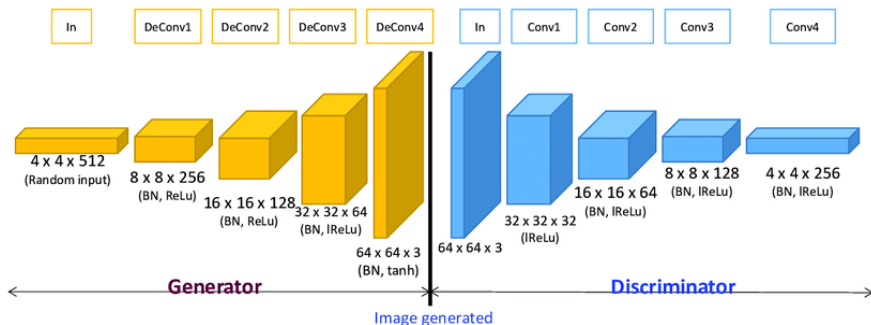


Schéma extrait de Suh et al., 2019

## Problème

Le discriminateur renvoie une unique réponse “réel” ou “faux” pour toute l'image.

→ le GAN ne capture que la structure globale de l'image

# PatchGAN

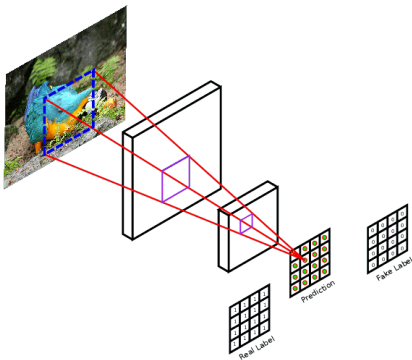


Schéma extrait de Demir et Unal, 2018

PatchGAN produit une **carte** de prédictions "réel/faux". Chaque pixel de la carte de sortie correspond à une région de l'image d'entrée du discriminateur.

## PatchGAN

PatchGAN est une architecture de discriminateur proposée par Isola et al. (2017) dans l'article pix2pix.

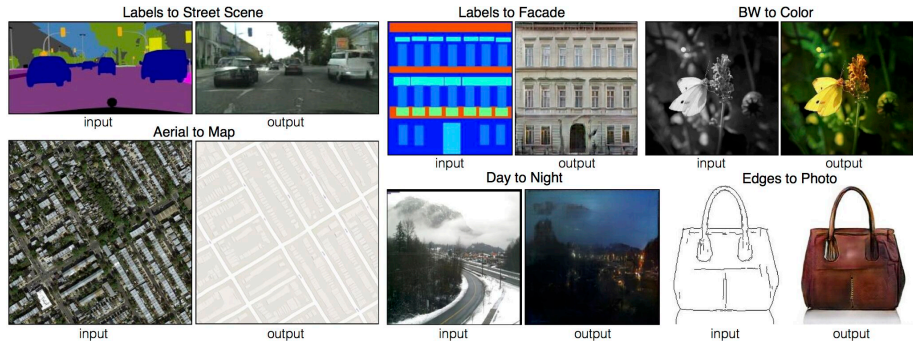
Principe : on ne conserve que la partie convolutive du discriminateur.

→ le discriminateur agrège des décisions locales

→  $D$  optimise la moyenne des entropies croisées sur toutes les prédictions  $(i, j)$



# pix2pix : principe



## pix2pix

Isola et al., Image-to-Image Translation with Conditional Adversarial Nets, 2017

**Principe** : convertir des images d'un domaine A vers un domaine B à partir de paires existantes.

**Architecture** : DCGAN conditionné à *une image* avec un discriminateur PatchGAN.

# pix2pix : architecture

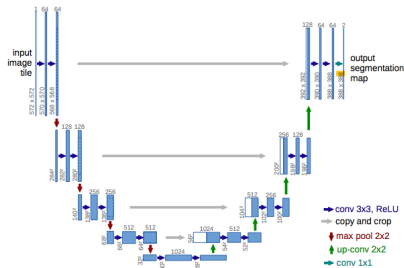
## Encodeur

Modèle entièrement convolutif :

U-Net

Entrée : image du domaine A

Sortie : image du domaine B

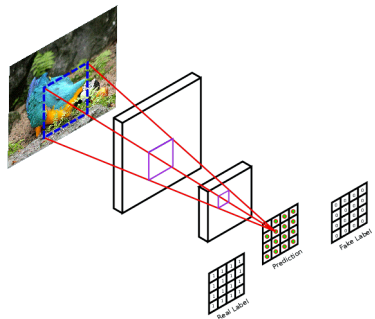


## Discriminateur

PatchGAN

Entrée : image (réelle ou fausse)

Sortie : une prédiction par région



## pix2pix : optimisation

### Optimisation

pix2pix est une méthode **supervisée** : il est nécessaire d'avoir des paires d'images. On optimise :

$$\min_G \max_D \mathcal{L}_{\text{GAN}} + \lambda \mathcal{L}_{\text{data}}(G)$$

avec une double fonction de coût :

- Attache aux données :  $\mathcal{L}_{\text{data}}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$
- GAN :  $\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z)))]$

avec  $x$  les images source,  $y$  les images cible et  $z$  un bruit.

**Interprétation** : la perte  $L_1$  capte les basses fréquences (= structure globale) tandis que PatchGAN intègre les hautes fréquences (= détails). Le GAN agit comme une **régularisation perceptuelle**.

# CycleGAN

Peut-on réaliser la “traduction” d’image sans paires pour la supervision ?

## CycleGAN

Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2018

Extension non supervisée de pix2pix

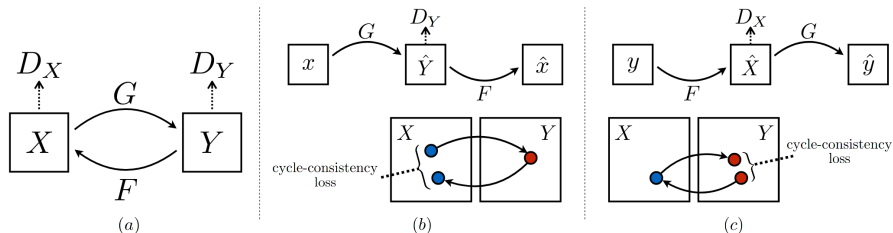
**Principe** : pour apprendre la transformation  $G : X \rightarrow Y$  sans paires d'exemples, on apprend aussi la transformation inverse  $F : Y \rightarrow X$ .

On souhaite avoir une idempotence du cycle  $F \circ G$  (et  $G \circ F$ ) :

- $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$  (cohérence *avant*),
- $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$  (cohérence *arrière*).

# CycleGAN : fonctionnement

On dispose d'images  $x \in X$  et  $y \in Y$ . Il n'y a *pas de correspondances* entre les images.



Le modèle apprend  $G : X \rightarrow Y$  et  $F : Y \rightarrow X$  par deux GAN :

$$L_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p(y)} [\log D_Y(y)] + \mathbb{E}_{x \sim p(x)} [\log(1 - D_Y(G(x)))]$$

et impose une contrainte de **cohérence cyclique** :

$$L_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p(x)} [\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p(y)} [\|G(F(y)) - y\|_1]$$

# CycleGAN : résultats

Monet ↔ Photos



Monet → photo

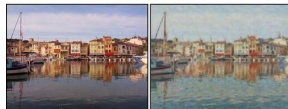
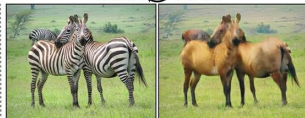


photo → Monet

Zebras ↔ Horses

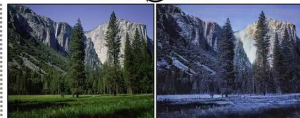


zebra → horse

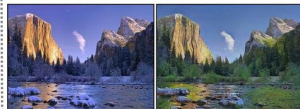


horse → zebra

Summer ↔ Winter



summer → winter



winter → summer



Photograph



Monet



Van Gogh



Cezanne

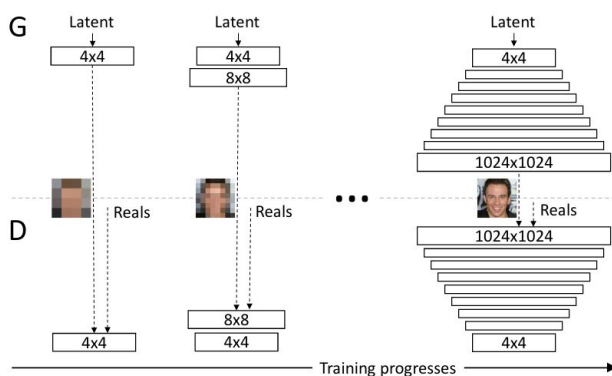


Ukiyo-e

# Plan du cours

- 1 Rappels
- 2 Extensions des GAN conditionnels pour les images
- 3 GAN pour la synthèse d'images à haute résolution
- 4 Espace latent et contrôle des GAN

# PGGAN

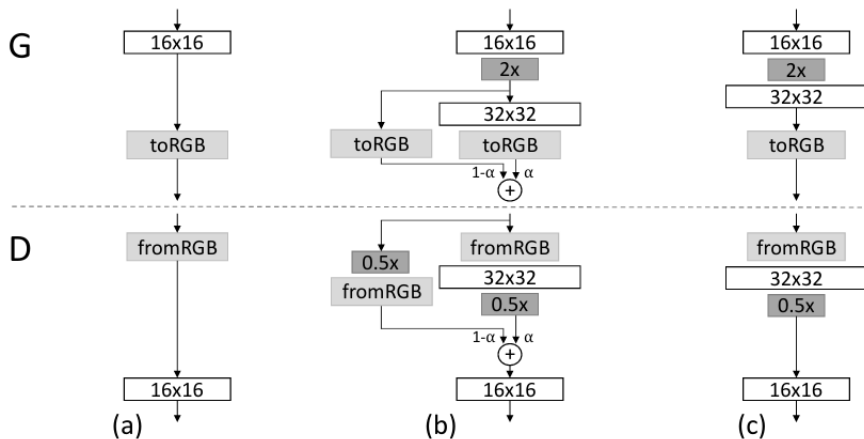


**Principe :** progressivement ajouter des couches au générateur et au discriminateur pour raffiner les images à basse résolution en images haute résolution.

Karras et al., Progressive Growing of GANs for Improved Quality, Stability and Variation, 2018



# PGGAN : ajout d'une couche



(a) Générateur à l'échelle  $16 \times 16$

(b) Transition :  $\alpha$  contrôle l'interpolation entre la nouvelle couche ( $32 \times 32$ ) et la couche précédente  $16 \times 16$  réinterpolée

(c) Générateur à l'échelle  $32 \times 32$

## PGGAN : PixelNorm

**Problème du jeu minimax** : les normes des paramètres du générateur et du discriminateur peuvent parfois croître sans limite.

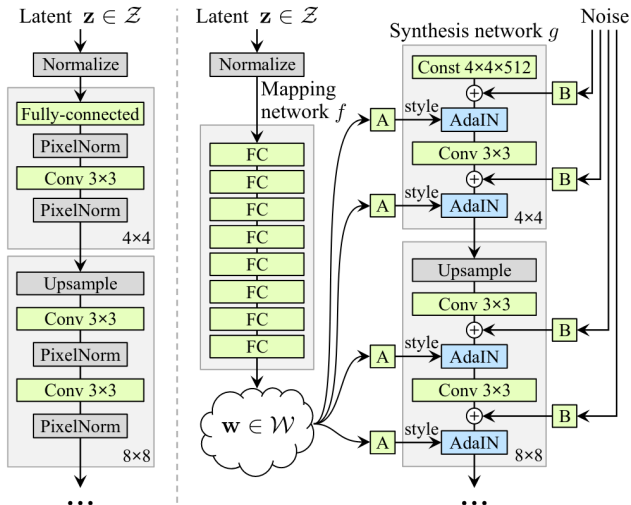
**PixelNorm** : *pixelwise feature vector normalization*

Les activations  $a_{x,y}$  après chaque couche convolutive du générateur sont normalisées selon une variante de la *Local Response Normalization* (Krizhevsky et al., 2012) :

$$b_{x,y} = \frac{a_{x,y}}{\sqrt{\frac{1}{N} \sum_{j=0}^{N-1} (a_{x,y}^j)^2}}$$

→ normalise les activations et permet de contrôler l'amplitude des paramètres

# StyleGAN



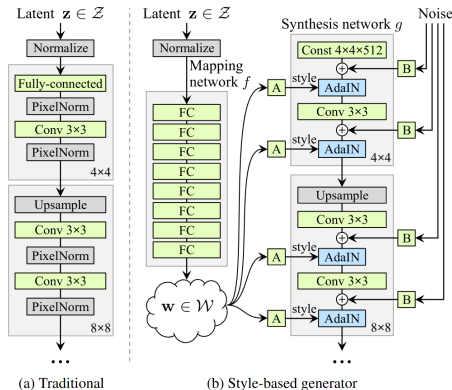
(a) Traditional

(b) Style-based generator

**Principe :** version “améliorée” de PGGAN, le code  $z \in \mathcal{Z}$  est transformé en style  $w \in \mathcal{W}$ . Ce style est injecté dans les différentes couches du générateur.

# StyleGAN : astuces

## Adaptive Instance Normalization (AdaIN)



## AdaIN

$w$  est décomposé par une transformation affine en

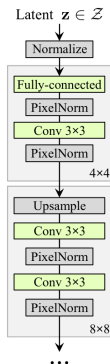
$$y = (y_s, y_b).$$

Le style permet ensuite de contrôler chaque carte de caractéristiques après une convolution dans l'AdaIN :

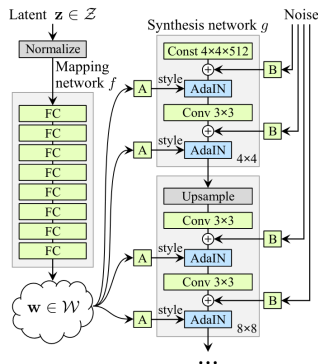
$$\text{AIN}(\mathbf{x}_i, \mathbf{y}) = y_{s,i} \frac{\mathbf{x}_i - \mu((x_i))}{\sigma(\mathbf{x}_i)} + y_{b,i}$$

# StyleGAN : astuces

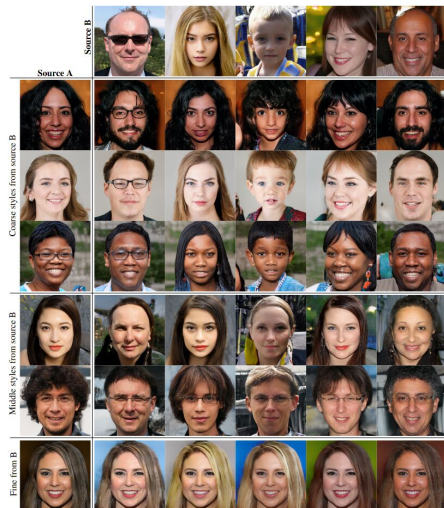
**Style mixing** : on remplace aléatoirement certains styles  $w$  par le style  $w'$  d'une autre image.



(a) Traditional



(b) Style-based generator



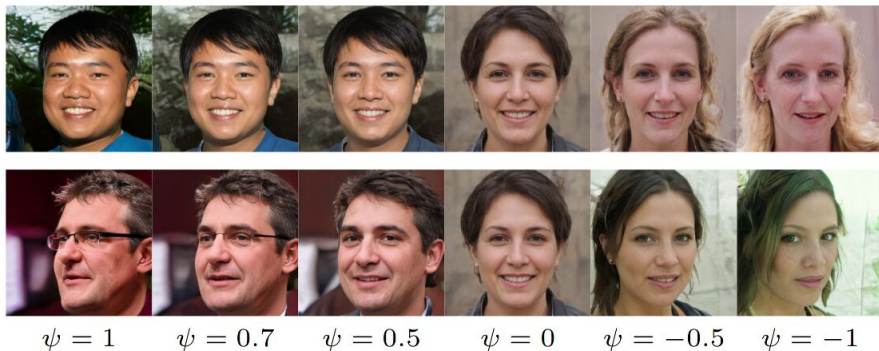
## StyleGAN : troncature

Pour simplifier l'échantillonnage, on considère une distribution tronquée :

$$w' = \bar{w} + \psi (w - \bar{w})$$

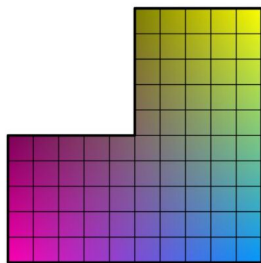
où  $\bar{w} = \mathbb{E}_{z \sim p(z)}[f(z)]$  est le centre de masse de  $\mathcal{W}$ .

$G(\bar{w})$  est l'image "moyenne" et s'en éloigner dans une direction ou son opposée "inverse" une image :

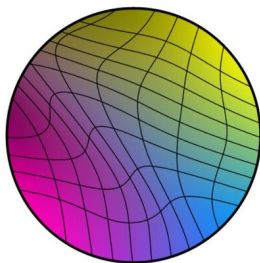


# StyleGAN : espace des styles

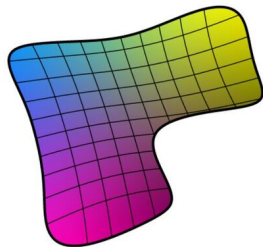
L'espace de style  $\mathcal{W}$  permet d'obtenir un espace latent structuré différemment de l'a priori gaussien de  $\mathcal{Z}$ . Le style correspond à une *embedding* que l'on espère plus interprétable.



(a) Distribution of features in training set



(b) Mapping from  $\mathcal{Z}$  to features



(c) Mapping from  $\mathcal{W}$  to features

# Plan du cours

- 1 Rappels
- 2 Extensions des GAN conditionnels pour les images
- 3 GAN pour la synthèse d'images à haute résolution
- 4 Espace latent et contrôle des GAN



# Espace latent

L'espace latent des modèles génératifs permet de parcourir la distribution des données synthétiques.

Mais les directions de l'espace latent n'ont en général pas de signification particulière. Les attributs sémantiques d'une image sont mélangés : on parle d'*entanglement*.

Comment explorer cet espace latent pour :

- comprendre la structure des données ?
- interpréter sémantiquement les variations statistiques latentes ?
- modifier une donnée réelle de façon plausible ?

# Manipulation d'image

Pour une image générée  $G(z) = \hat{x}$ , comment déplacer  $z$  de sorte à produire un effet donné sur  $\hat{x}$  (e.g.pose, expression, couleur, etc.) ?

## Problématique

Comment trouver des **directions sémantiques** dans l'espace latent ?

**Idée naïve** : arithmétique des vecteurs dans l'espace latent

« homme avec lunettes » - « homme sans lunettes » + « femme sans lunettes » = « femme avec lunettes »

# InterFaceGAN

**Prérequis** : un classifieur  $C$  capable d'identifier un attribut binaire  $y \in \{0, 1\}$ .

**Objectif** : trouver une direction  $d \in \mathcal{Z}$  qui permet de contrôler la présence de l'attribut  $y$  dans les images générées  $G(z)$ .

Shen et al., Interpreting the latent space of GANs for semantic face editing, 2020

## Algorithme

- 1 Échantillonner  $N$  codes latents
- 2 Générer les  $N$  images  $\hat{x}_i$  correspondantes
- 3 Obtenir les pseudo-étiquettes  $\hat{y}_i = C(G(z_i))$  grâce au classifieur
- 4 Conserver les  $n$  codes latents  $z_i^+$  (resp.  $z_i^-$ ) qui ont produit les images qui obtiennent les scores les plus élevés (resp. les plus faibles)
- 5 Trouver la direction  $d$  comme la normale à l'hyperplan séparateur d'une SVM linéaire qui séparent  $\mathcal{Z}^+$  et  $\mathcal{Z}^-$

# GANSpace

Härkönen et al., Discovering Interpretable GAN Controls, 2020

## Principe

GANSpace exploite la structure de  $\mathcal{W}$  et applique une ACP dans l'espace latent de StyleGAN.

La découverte des directions est non-supervisée mais leur sémantique est interprétée *a posteriori*.



# Édition d'image

## Modifier une image existante

Pour modifier une image réelle, il est nécessaire de connaître son équivalent  $z$  dans l'espace latent : c'est le problème de l'**inversion d'image**.

## Comparaison GAN/VAE

Un auto-encodeur (variationnel) possède un encodeur qui réalise une projection  $\mathcal{X} \rightarrow \mathcal{Z}$ . Ce n'est pas le cas des GAN.

## Recherche par descente de gradient

Une solution coûteuse pour retrouver  $z^*$  qui génère une image cible  $x$  consiste à minimiser :

$$\min_{z \in \mathcal{Z}} \|G(z) - x\|_p$$

par descente de gradient.

# Apprentissage d'un encodeur pour l'inversion

Une solution moins coûteuse que la descente de gradient consiste à apprendre un réseau de neurones  $P$  paramétré par  $\theta$  qui réalise la projection  $\mathcal{X} \rightarrow \mathcal{Z}$ .

## Algorithme

- 1 Échantillonner  $n$  codes  $z_i$  et les images  $G(z_i) = x_i$  correspondantes
- 2 Apprendre  $P$  par descente de gradient :  
$$\theta_P^* = \arg \min_{\theta} \sum_{i=1}^n \mathcal{L}(G(P(z_i; \theta)), z_i)$$
- 3 Pour une image  $x'$ , le code associé est  $z' = P(x')$ .

Une approche rapide et performante consiste à utiliser  $z = P(x)$  comme initialisation pour une recherche par descente de gradient. (Zhu et al., 2016)