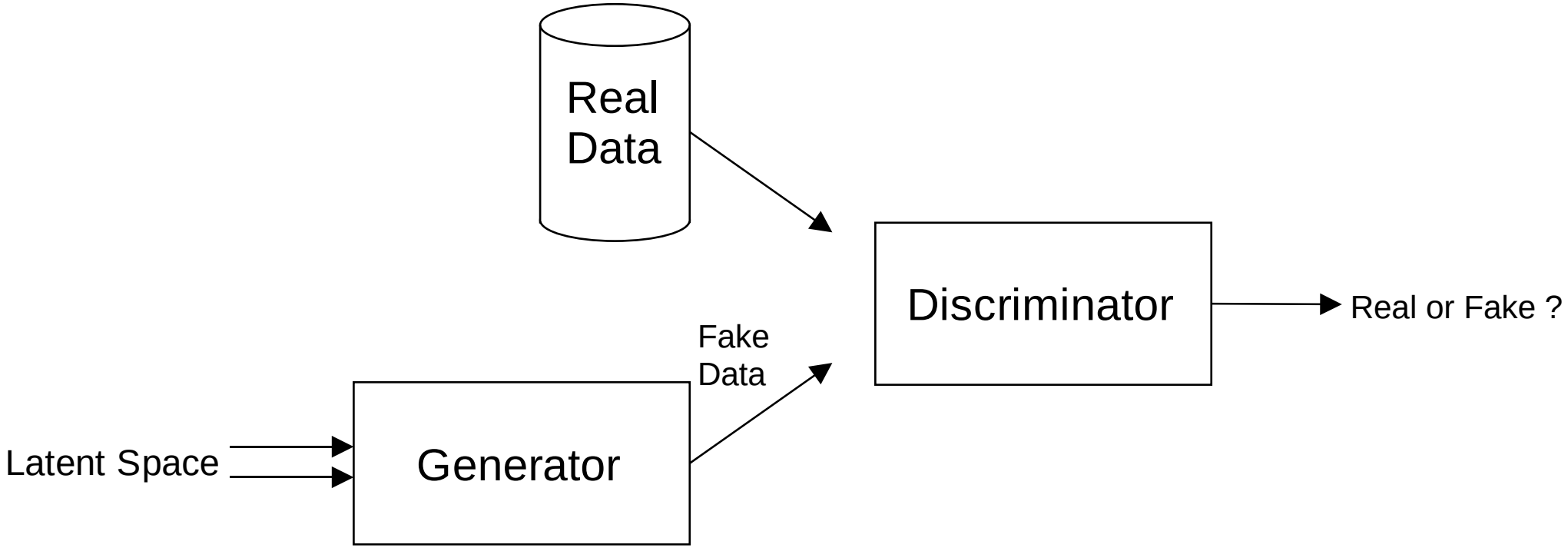
Generative Adversarial Networks

- 2 modèles aux objectifs opposés :



Generative Adversarial Networks

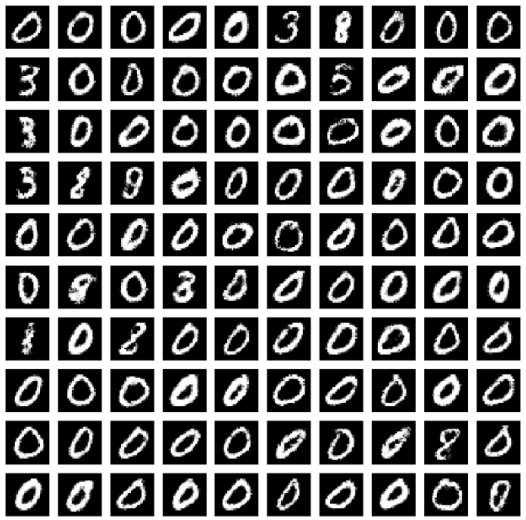
- Apprentissage :
  - Répéter :
    - Apprentissage du discriminateur :
      - Générer un demi batch de données à partir des données réelles => label target 1
      - Générer un demi batch de données à partir du générateur => label target 0
      - Mélanger ces deux demi batchs pour former un seul batch d'apprentissage
      - Effectuer n itérations d'apprentissage à l'aide de ce dernier ($1 \leq n \leq 5$)
    - Apprentissage du générateur :
      - Figer les variables du discriminateur
      - Générer un batch de données dans le latent space => label target 1
      - Effectuer 1 itération d'apprentissage à l'aide de ce dernier

Generative Adversarial Networks

- Faites varier :
  - Structure (Vanilla GAN, DeepConvolutionnal GAN, MSGGAN)
  - n (itérations du discriminateur)
  - Pas d'apprentissages (les deux)
  - Fonction de loss (MSE, KL, Wasserstein…)

Generative Adversarial Networks

- Exemple d'images obtenues avec un MSG GAN Dense avec un sampling aléatoire uniforme depuis le latent space :



- On remarque que les images ne sont plus floues !

Variational Auto Encoders

- On repart de l'auto encoder

- Idée :
  - L'encoder ne projette plus directement dans l'espace latent, mais produit les paramètres de plusieurs distributions de probabilités.
  - On choisit en général des Gaussiennes (cf. théorème central limite).
  - Pour générer, le générateur va échantillonner ('sampling') ces distributions de probabilités pour produire des valeurs dans l'espace latent et ensuite fonctionner comme précédemment.

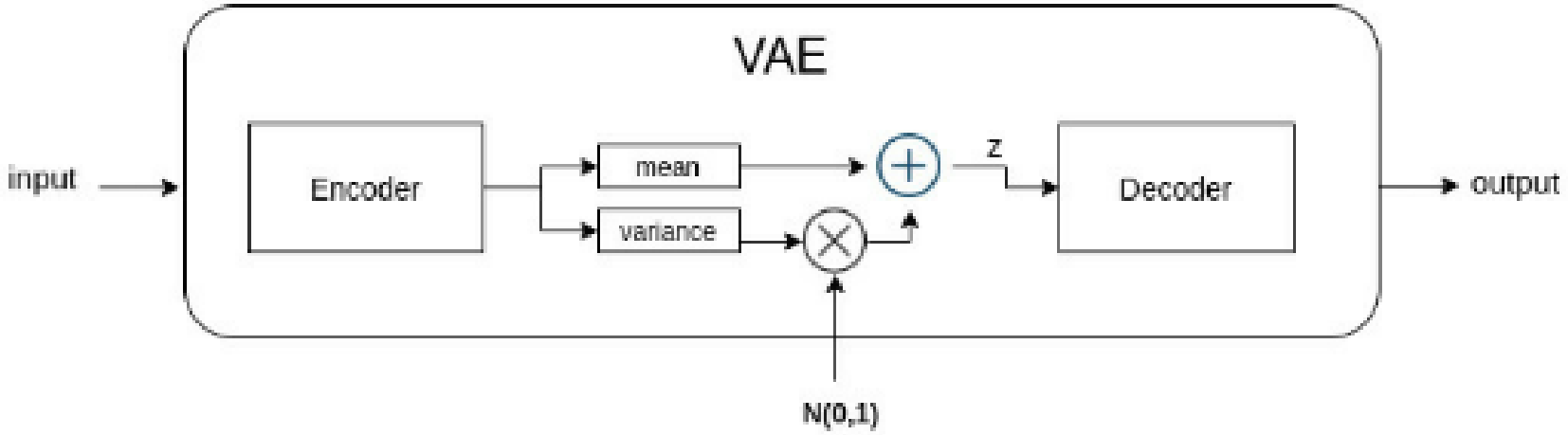
Variational Auto Encoders

- Problème:
  - On ne peut pas ‘dériver’ un échantillonnage (pour la rétropropagation) !

- Solution
  - Le ‘Reparameterization Trick’ :
    - Au lieu d’échantillonner la distribution d’espérance μ et d’écart type σ, on va échantillonner la distribution Normale standard : $\aleph(0,1)$ puis multiplier le résultat par σ et ajouter μ.

Variational Auto Encoders

- Solution
  - Le ‘Reparameterization Trick’ :
    - Au lieu d’échantillonner la distribution d’espérance μ et d’écart type σ, on va échantillonner la distribution Normale standard : $\aleph(0,1)$ puis multiplier le résultat par σ et ajouter μ.



Variational Auto Encoders

- Solution
  - Le ‘Reparameterization Trick’ :
    - Au lieu d’échantillonner la distribution d’espérance μ et d’écart type σ, on va échantillonner la distribution Normale standard : $\aleph(0,1)$ puis multiplier le résultat par σ et ajouter μ.
    - $z = \mu + \sigma \times \aleph(0,1)$
  - En pratique pour des raisons de stabilité numérique, l’encoder prédit le logarithme de la variance $\log(\sigma^2)$ plutôt que la variance ou l’écart type ainsi :
    - $z = \mu + e^{0.5 \times \log(\sigma^2)} \times \aleph(0,1)$

Variational Auto Encoders

- Ajout de régularisation :
  - On va ajouter une pénalité (loss) aux paramètres des gaussiennes s'éloignant trop de la distribution normales standard (le retour de la 'KL Divergence'!) :

$$D_{KL}(N(\mu,\sigma)\,||N(0,1)) = -0.5 \sum_{i=1}^{zdim} \log(\sigma_i^2) - \sigma_i^2 - \mu_i^2 + 1$$

Variational Auto Encoders

- Ajout de régularisation :
  - On va ajouter une pénalité (loss) aux paramètres des gaussiennes s'éloignant trop de la distribution normales standard (le retour de la 'KL Divergence'!) :

$$D_{KL}(N(\mu,\sigma)\,||N(0,1)) = -0.5 \sum_{i=1}^{zdim} \log(\sigma_i^2) - \sigma_i^2 - \mu_i^2 + 1$$

  - Ainsi le loss total peut s'exprimer :

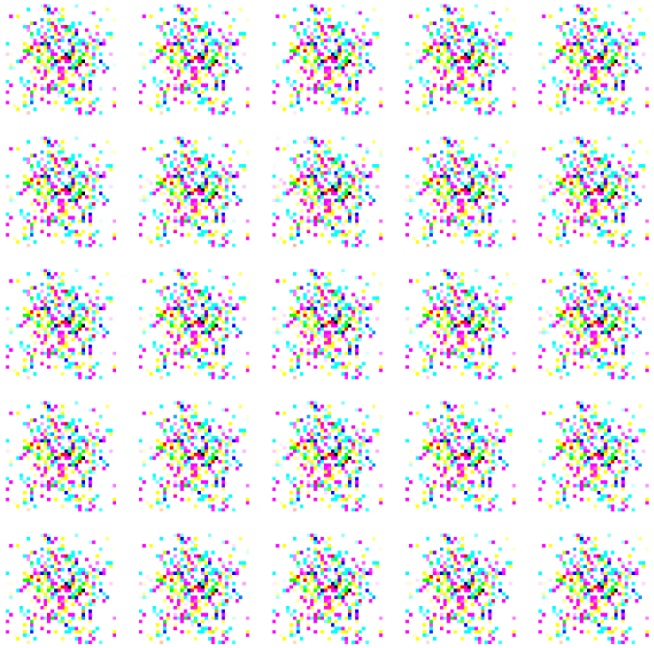
$$loss = mse(x, x') + coef_{KL} \times D_{KL}$$

Variational Auto Encoders

- N’hésitez pas non plus à utiliser des Blocs de Convolutions dans le l’encoder, et des couches de convolution et d’Upsampling dans le decoder.

Entraînement des GANs

- Entraîner des GANs est difficile (stabilité)



WGAN (Wasserstein GAN)

- Nouvelle fonction de loss !
  - « Earth mover's distance » ou « Wasserstein distance »

```
def wasserstein_loss(self, y_true, y_pred):
    w_loss = -tf.reduce_mean(y_true*y_pred)
    return w_loss
```

  - Avec les changements suivants :
    - vraie image : 1
    - Fausse image : -1
    - Activation du neurone de la dernière couche du discriminateur : linear
    - Contraindre tous les poids des couches du discriminateur entre -0.01 et 0.01

WGAN-GP (Gradient Penalty)

- Nouvelle fonction de loss !
  - « Earth mover's distance » ou « Wasserstein distance »

```
def wasserstein_loss(self, y_true, y_pred):
    w_loss = -tf.reduce_mean(y_true*y_pred)
    return w_loss
```

  - Avec les changements suivants :
    - vraie image : 1
    - Fausse image : -1
    - Activation du neurone de la dernière couche du critique : linear
    - ~~Contraindre tous les poids des couches du discriminateur entre -0.01 et 0.01~~

WGAN-GP (Gradient Penalty)

- ~~Contraindre tous les poids des couches du discriminateur entre -0.01 et 0.01~~
- https://arxiv.org/pdf/1704.00028.pdf

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}.$$

WGAN-GP (Gradient Penalty)

- ~~Contraindre tous les poids des couches du discriminateur entre -0.01 et 0.01~~

- https://arxiv.org/pdf/1704.00028.pdf

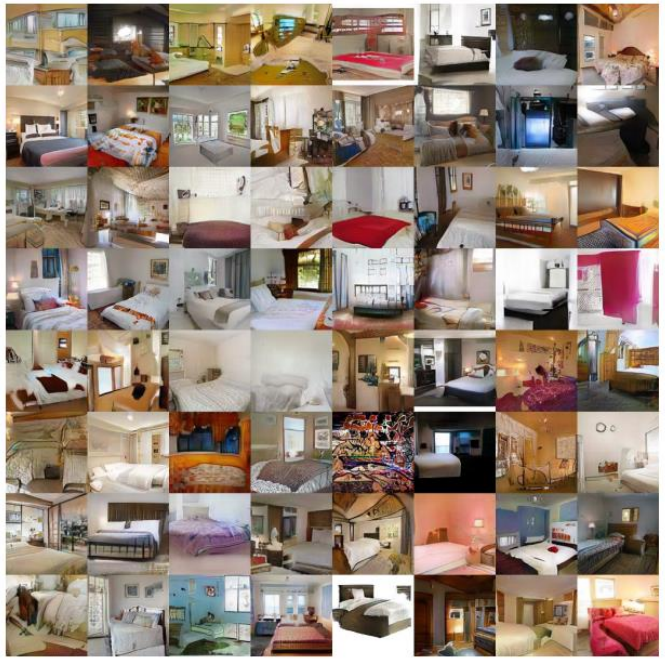
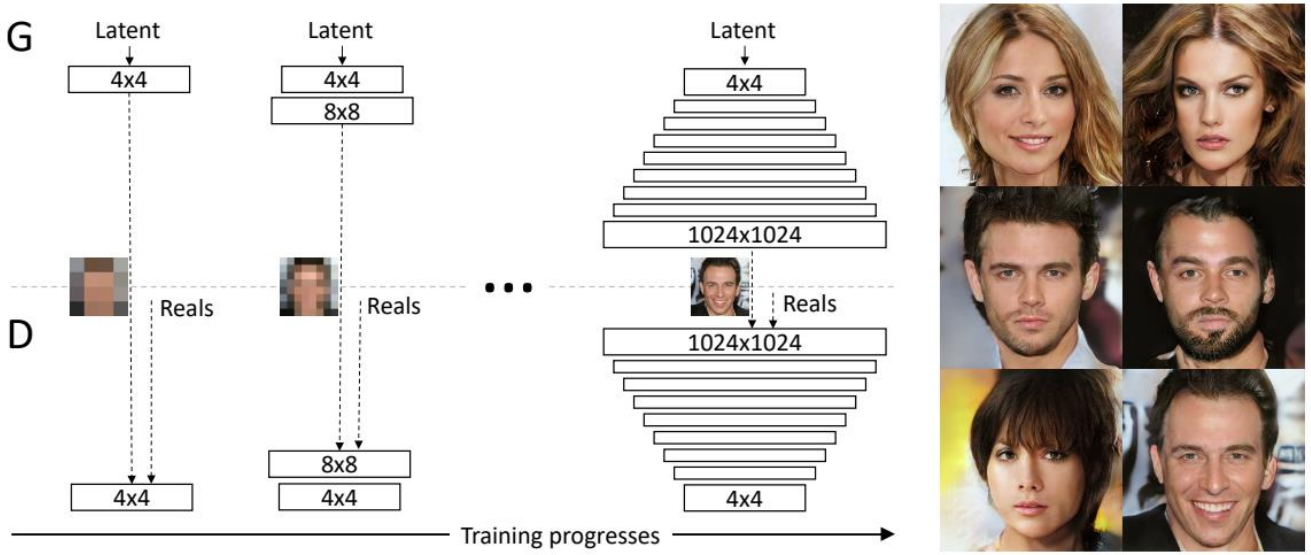


Figure 4: Samples of 128×128 LSUN bedrooms. We believe these samples are at least comparable to the best published results so far.

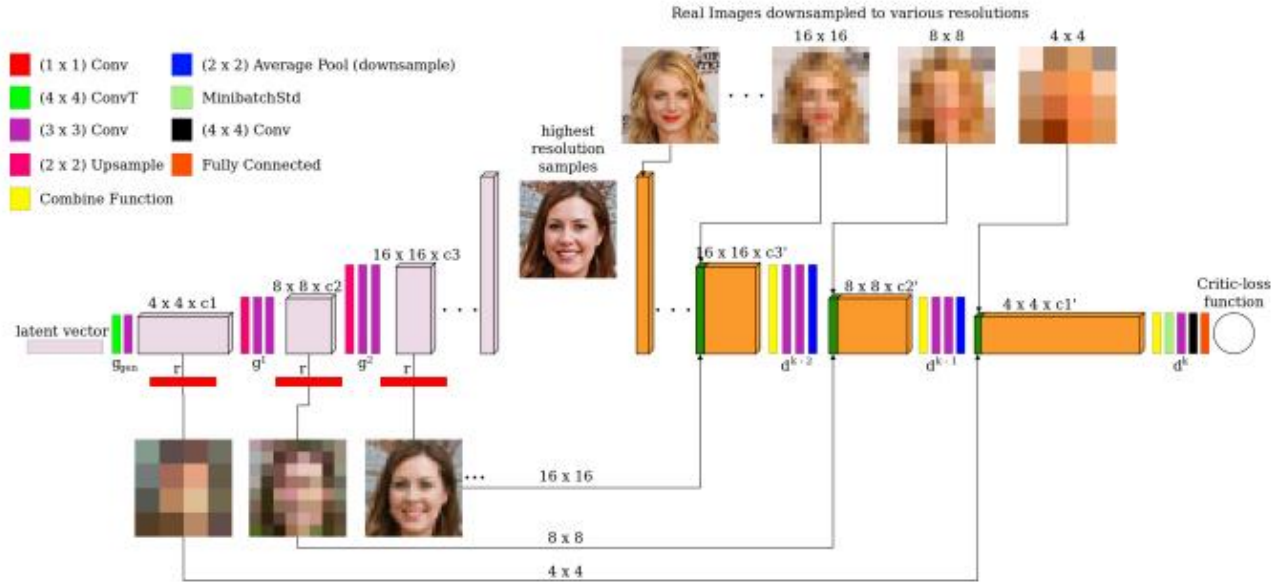
Progressive Growing of GANs

- Entraîner un GAN sur de basses résolutions puis rajouter des couches progressivement :
  - https://research.nvidia.com/sites/default/files/pubs/2017-10_Progressive-Growing-of/karras2018iclr-paper.pdf



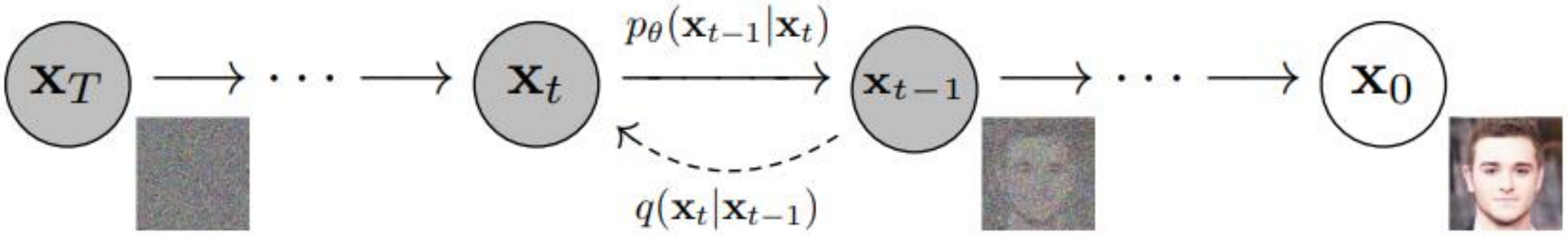
MSGGAN (Multi Scale Gradient GAN)

- Décomposer l'espace des images en plusieurs résolutions pour stabiliser l'apprentissage :
  - https://arxiv.org/abs/1903.06048



Diffusion Models

- DDPM et ses dérivés
  - https://arxiv.org/abs/2006.11239
    https://arxiv.org/abs/2102.09672
    https://arxiv.org/abs/2105.05233



Blog de Lilian Weng
(pour se tenir à jour à moindre cout)

- https://lilianweng.github.io/posts/2021-07-11-diffusion-models/