Generative Adversarial Networks (GANs)

=> classe d'algorithmes d'apprentissage non-supervisé

=> Ils utilisent deux réseaux neuronaux s'opposant l'un à l'autre (donc "adversaires") pour générer de nouvelles instances synthétiques de données qui peuvent passer pour des données réelles.

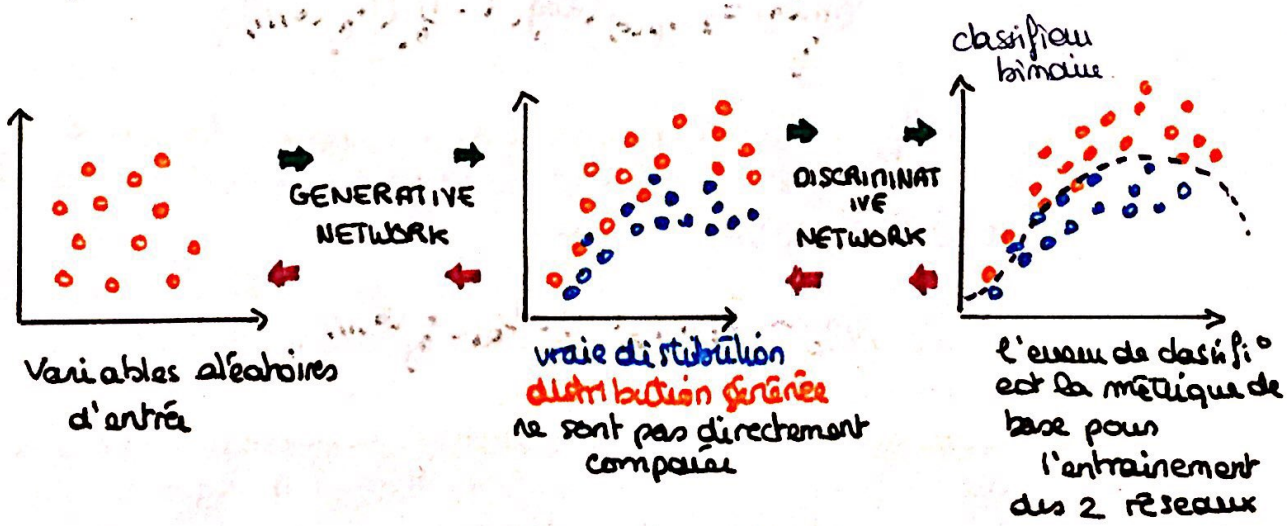
Utilisation :

- génération d'images
- génération de vidéos.
- génération de voix.

Différence entre VAE et GAN = avec les VAE, on avait une modélisa° explicite de la densité, avec les GAN, on a une version implicite (on ne pourra pas la calculer ou l'approximer).

=> on veut générer des échantillons de puis une distribution complexe, on a des données d'entrées en très gde dimension (trop grand pour utiliser les méthodes classiques).

=> Pour cela, on va générer un bruit de la distribution (qui on sait générer), on donne cela à un RN qui nous donne une sortie.



=> le réseau doit être capable de dire de quel jeu les données proviennent.

GENERATEUR = réseau de neurones qui modélise une fonction de transformation. Il prend en entrée une va et retourne une va qui suit la distribution ciblée.

DISCRIMINATEUR = un autre réseau de neurones qui modélise une fonction discriminante. Il prend en entrée un point et retourne en sortie la probabilité que ce point soit un vrai point.

=> on travaille dans des espaces paramétrés donc les points optimaux peuvent être considérés comme des arrondis.

* Fonction objectif.

The minmax game.

$$\min_{\theta_g} \max_{\theta_d} \left[E_{x \sim p_{data}}(\log(D_{\theta_d}(x))) + E_{z \sim p(z)}(\log(1 - D_{\theta_d}(G_{\theta_g}(z)))) \right]$$

$D_{\theta_d}(x)$: sortie du discriminateur pour données x réelles

$D_{\theta_d}(G_{\theta_g}(z))$: sortie du discriminateur pour les fausses données générées $G(z)$

- le but du générateur est de tromper le discriminateur, donc le réseau générateur est entraîné pour maximiser l'erreur de classification finales (entre les données vraies et les données générées).
  - => le générateur veut minimiser la fonction objectif tel que $D(G(z))$ est proche de 1.

$$\min_{\theta_g} E_{z \sim p(z)}(\log(1 - D_{\theta_d}(G_{\theta_g}(z))))$$

en pratique on a un problème de convergence. (le gdt ne va jamais essayer d'améliorer les img qui ne sont pas réelles) on modifie la fonction objectif :

$$\max_{\theta_g} E_{z \sim p(z)}(\log(D_{\theta_d}(G_{\theta_g}(z))))$$

- le but du discriminateur est de détecter les fausses données générées de sorte que le réseau de neurones discriminant soit formé pour minimiser l'erreur de classification finale.

$$\max_{\theta_d} \left[E_{x \sim p_{data}}(\log D_{\theta_d}(x)) + E_{z \sim p(z)}(\log(1 - D_{\theta_d}(G_{\theta_g}(z)))) \right]$$

$\log D_{\theta_d}(x)$: maximiser quand on lui donne une vraie image.

$\log(1 - D_{\theta_d}(G_{\theta_g}(z)))$: maximiser quand on lui donne une fausse img

* Optimisation

Etape 1 : Mise à jour des paramètres du discriminateur en gardant ceux du générateur fixe.

- Pour k steps :
  - Echantillonner en mini batch du bruit $z_1, \dots z_m$ et des données réelles $x_1, \dots x_m$
  - Mise à jour des paramètres de D par SGA sur

$$\frac{1}{m} \sum_m \left[\log(D(x_m)) + \log(1 - D(G(z_m))) \right]$$

Etape 2 : Mise à jour du générateur

- Echantillonner des bruits en mini-batch $z_1, \dots z_m$
- Mise à jour des paramètres de G par SGA sur

$$\frac{1}{m} \sum_m \log(D(G(z_m)))$$

=> on veut aligner les distributions des vraies et fausses données de seule moment où on voit les données réelles c'est quand on met à jour le discriminateur.