SCIENCES SORBONNE UNIVERSITÉ

BIoMétrie : Generative Adversarial Networks

L'objectif de ce TP est d'implémenter une architecture neuronale complexe, comme les réseaux génératifs antagonistes, pour apprendre à générer des images. Le TP est aussi l'occasion d'appréhender en TF l'apprentissage de réseaux complexes avec des fonctions de pertes qui n'affectent qu'une partie des poids du réseau seulement. Nous utiliserons la base de chiffres `MNIST` accessibles depuis le module dataset de Keras.

Instanciation de G et D

Nous allons commencer par instantier le générateur G et le discriminateur D utilisé dans notre GAN avec Keras.

1. Créer un générateur G avec une architecture CNN. On rappelle que la dimension de l'entrée de G est un code latent aléatoire dont on fixera la dimension à 32, et la dimension de la sortie correspond à la tailles des images (pour MNIST, 28x28). On prendra plusieurs couches CNN avec les spécifications suivantes :
   — Une première couche `Dense` qui transforme l'entrée du générateur dans une carte d'activation de dimension (14x14x128)
   — Des couches `Conv2DTranspose` avec 128 ou 256 filtres de taille (4x4) ou (5x5). On remarquera que la couche `Conv2DTranspose` permet de gérer directement le `UpSampling2D` nécessaire pour arriver à une taille d'image désirée, avec le `stride` de la convolution transposée.
   — Des fonctions d'activation `LeakyReLU`
   — Des couches de `BatchNormalization`
   — On veillera à chaque fois à que les dimensions soient préservées lors des convolutions successives
2. Créer un discriminateur D avec une architecture CNN. On rappelle que la dimension de l'entrée de D est la tailles des images (réelles ou générées) et que la sortie est un scalaire qui est utilisé pour prendre la décision "réel" ou "faux" On prendra plusieurs couches avec les spécifications suivantes :
   — Une succession de couches `Conv2D` et de `LeakyReLU`
   — Une dernière couche dense avec une activation `Sigmoid` pour faire la prédiction de la décision du discriminateur entre 0 et 1.

Une fois les modules G et D instanciés, vérifier l'architecture et le nombre de paramètres entrainables avec la méthode `summary`

Création du GAN

Maintenant que nous avons instancié les deux modules élémentaires G et D, nous allons pouvoir créer le réseau GAN utilisé pour l'apprentissage antagoniste.

3. Pour la partie du discriminateur D, simplement compiler le module D avec la loss `binary_crossentropy` et l'optimiseur `Adam`. Utiliser la fonction summary pour visualiser l'architecture de votre réseau D.
4. Pour la partie du générateur G, l'apprentissage consiste à optimiser les poids du générateur G pour un état donné du discriminateur D.

— Faire une recopie du discriminateur D pour créer le discriminateur `discriminateur_gan`
— Déclarer les poids de ce discriminateur comme non-entrainables, en fixant la valeur de son attribut `trainable` à `False`
— Instancier le réseau GAN, avec comme entrée la taille du code latent aléatoire et comme sortie la composition de `discriminator_gan(generator(gan_input))`
— Définir la loss `binary_crossentropy` et l'optimiseur `Adam`

Apprentissage du GAN

Nous allons maintenant procédé à l'apprentissage alterné des poids du générateur G et du discriminateur. Pour ce faire, nous allons créer notre propre boucle pour traiter les batches. Nous reproduirons pour chaque batches les étapes suivantes :

5. Réaliser un tirage aléatoire du code latent, en utilisant un tirage aléatoire dans une distribution normale. Pour ce faire on utilisera la fonction `random` de `numpy`.
6. Réaliser la prédiction des images générées à partir de G et du code latent
7. Construire un mélange d'images réelles tirées aléatoirement et d'images générées avec les labels correspondant
8. Entrainer le discriminateur D à partir de ce mélange. On utilisera la méthode `train_on_batch`
9. Réaliser un nouveau tirage aléatoire du code latent, et créer les labels correspondants à "réel"
10. Entrainer le réseau antagoniste à partir de ces données. On utilisera la méthode `train_on_batch`

Nous ferons un affichage des losses du discriminateur D et réseau antagoniste GAN toutes les 10 époques, et un tracé de 10 images produites par le générateur.

Les plus avancés pourront essayer de construire un GAN pour la génération de visages à partir de la base de visages `celeb_a` incluse dans le même module.

GAN conditionné (BONUS)

Le réseau GAN défini précédemment ne permet pas de définir les chiffres que nous souhaitons générer. Pour pallier à cette lacune, nous allons implémenter un GAN conditionné par l'attribut du chiffre. Pour ce faire, modifier l'architecture précédemment définie pour reproduire l'architecture du CGAN telle que vue en cours. Faire un tracé des images produites en fonction de la valeur du conditionnement utilisé.