DCGAN
Les points à retenir de cette partie

Bouchra && Laura

M2MLDS

1-Comment fonctionne un DCGAN ?

1-1 Generator :

-Le générateur commence par un vecteur de bruit z . En utilisant une couche entièrement connectée, nous « reshape » le vecteur en une couche cachée en trois dimensions avec une petite base (largeur × hauteur) et une grande profondeur. En utilisant des convolutions transposées, l'entrée est progressivement remodelée de telle sorte que sa base croît tandis que sa profondeur diminue jusqu'à ce que nous atteignions la couche finale avec la forme de l'image que nous cherchons à synthétiser, 28 × 28 × 1.

-Après chaque couche de convolution transposée, nous appliquons la normalisation par bash et la fonction d'activation *Leaky ReLU* . À la couche finale, nous n'appliquons pas la normalisation par bash et, au lieu de ReLU, nous utilisons la fonction d'activation *tanh* .

Pour une résumé :

1. Prenez un vecteur de bruit aléatoire et remodelez-le en un tenseur 7 × 7 × 256 à travers une couche entièrement connectée.
2. Utilisez la convolution transposée, transformant le tenseur 7 × 7 × 256 en un tenseur 14 × 14 × 128.
3. Appliquez la normalisation par bash et la fonction d'activation *Leaky ReLU* .
4. Utilisez la convolution transposée, transformant le tenseur 14 × 14 × 128 en un tenseur 14 × 14 × 64. Notez que les dimensions de largeur et de hauteur restent inchangées; Pour ce faire, définissez le paramètre de foulée Conv2DTransposesur 1.
5. Appliquez la normalisation par lots et la fonction d'activation *Leaky ReLU* .
6. Utilisez la convolution transposée, transformant le tenseur 14 × 14 × 64 en taille d'image de sortie, 28 × 28 × 1.
7. Appliquez la fonction d'activation *tanh*
8.

1-2 Discriminateur

-*P*rend une image et génère un vecteur de prédiction : l'image d'entrée est réelle ou fausse.

L'entrée du Discriminateur est une image 28 × 28 × 1.En appliquant des convolutions, l'image est transformée de telle sorte que sa base (largeur × hauteur) devient progressivement plus petite et sa profondeur devient progressivement plus profonde. Sur toutes les couches convolutives, nous appliquons la fonction d'activation *Leaky ReLU* . La normalisation par bash est utilisée sur toutes

les couches convolutives sauf la première. Pour la sortie, nous utilisons une couche entièrement connectée et la fonction d'activation *sigmoïde* .

Pour une résumé des étapes d'implémentation de discriminator :

1. Utilisez une couche Convo pour transformer une image d'entrée 28 × 28 × 1 en un tenseur 14 × 14 × 32.
2. Appliquez la fonction d'activation *Leaky ReLU* .
3. Utilisez une couche Convo, transformant le tenseur 14 × 14 × 32 en un tenseur 7 × 7 × 64.
4. Appliquez la normalisation par bash et la fonction d'activation *Leaky ReLU* .
5. Utilisez une couche Convo, transformant le tenseur 7 × 7 × 64 en un tenseur 3 × 3 × 128.
6. Appliquez la normalisation par bash et la fonction d'activation *Leaky ReLU* .
7. « Flatten » le tenseur 3 × 3 × 128
8. Utilisez une couche entièrement connectée alimentant la fonction d'activation *sigmoïde* pour calculer la probabilité que l'image d'entrée

1-3.Construire et faire fonctionner le DCGAN

-Pourquoi former de dcgan : Le discriminateur classera les échantillons générés comme non réels (ou à faible probabilité d'être réels). Le processus de rétropropagation utilisé pour mettre à jour les poids du modèle verra cela comme une grande erreur et mettra à jour les poids du modèle (c'est-à-dire uniquement les poids dans le générateur) pour corriger cette erreur, ce qui rend le générateur plus efficace pour bien générer les échantillons.

-L'implémentation :

Prend comme arguments les modèles générateur et discriminateur déjà définis et crée le nouveau modèle.

1-4 Training

1-Nous obtenons un mini-batch des images MNIST comme exemples réels et générons un mini-bash d'images fausses à partir de vecteurs de bruit aléatoires *z* .Nous les utilisons ensuite pour former le réseau Discriminator tout en gardant les paramètres du générateur constants. Ensuite, nous générons un mini-batch de fausses images et les utilisons pour former le réseau Generator tout en gardant les paramètres du Discriminator fixes. Nous répétons cela pour chaque itération.

2- Nous utilisons des étiquettes : 1 pour les images réelles et 0 pour les fausses.

3-Le Discriminateur est formé pour attribuer la bonne étiquelle aux images. Le générateur est formé de telle sorte que le discriminateur attribue de *véritables* étiquettes aux faux exemples qu'il produit.

4- Nous redimensionnons les images réelles dans le jeu de données d'apprentissage de –1 à 1.

Le générateur utilise la fonction d'activation « *tanh* »au niveau de la couche de sortie, de sorte que les fausses images seront dans la plage (–1 , 1). Par conséquent, nous devons redimensionner toutes les entrées du discriminateur dans la même plage

5- On fixe un sample_interval pour sauvegarder les modèles : discriminator et generator pour pouvoir tester la performance de modèle sur les données testes.