Le TD3a est un tutoriel proposé par MathWorks pour vous guider dans la création d'un réseau de neurones de classification à l'aide de Deep Learning Toolbox dans MATLAB.

Le tutoriel commence par présenter les étapes de base de la création d'un réseau de neurones, notamment la définition de l'architecture du réseau, la préparation des données d'entraînement et de test, la configuration des options d'entraînement et l'évaluation des performances du réseau.

Ensuite, vous apprendrez à définir l'architecture du réseau de neurones à l'aide de la fonction "neuralNetworkLayer" et à configurer les options d'entraînement à l'aide de la fonction "trainingOptions". Vous allez également découvrir comment préparer les données d'entraînement et de test à l'aide de la fonction "imds" (Image Datastore).

Après avoir configuré et entraîné le réseau de neurones, vous allez évaluer les performances du réseau en utilisant des mesures de précision et de rappel. Vous allez également visualiser les résultats de la prédiction en utilisant la matrice de confusion et les courbes de caractéristiques de fonctionnement du récepteur (ROC).

Enfin, le tutoriel se termine en présentant comment utiliser le réseau de neurones pour prédire de nouvelles images.

Dans le cadre de cet exercice, nous avons pu tester l’impact de plusieurs paramètres de l’architecture neuronale comme :

- La couche d'entrée : c'est la première couche du réseau de neurones, qui reçoit les données d'entrée. La taille de cette couche dépend de la taille de vos données d'entrée. Dans cet exercice, les images ont une taille de 28x28 pixels, donc la couche d'entrée est définie avec une taille de 28x28x1 (où 1 représente le nombre de canaux de couleur).
- Les couches cachées : ce sont les couches intermédiaires du réseau de neurones qui transforment les entrées en sorties souhaitées. Le nombre et la taille des couches cachées peuvent varier en fonction de la complexité de votre problème de classification. Dans cet exercice, nous avons défini une couche cachée dense de 100 neurones avec une fonction d'activation ReLU.
- La couche de sortie : c'est la dernière couche du réseau de neurones, qui produit les sorties finales. Le nombre de neurones de cette couche dépend du nombre de classes que vous essayez de classifier. Dans cet exercice, il y a 10 classes, donc la couche de sortie est définie avec une taille de 10, avec une fonction d'activation softmax.
- Les fonctions d'activation : chaque couche de neurones peut avoir une fonction d'activation différente qui est appliquée à la sortie de chaque neurone. Les fonctions d'activation aident à introduire des non-linéarités dans le réseau et à améliorer la capacité de généralisation du réseau. Dans cet exercice, nous avons utilisé la fonction d'activation ReLU pour la couche cachée et la fonction d'activation softmax pour la couche de sortie.
- La fonction de perte : c'est la fonction utilisée pour évaluer la différence entre la sortie du réseau de neurones et la vérité terrain. La fonction de perte est minimisée pendant l'entraînement pour améliorer la performance du réseau. Dans cet exercice, nous avons utilisé la fonction de perte de la croix-entropie catégorielle, qui est couramment utilisée pour la classification.
- Les optimiseurs : les optimiseurs sont utilisés pour minimiser la fonction de perte en ajustant les poids et les biais du réseau de neurones. Dans cet exercice, nous avons utilisé

l'optimiseur Adam, qui est une variante de l'optimiseur de descente de gradient stochastique (SGD) qui est plus adaptatif et plus rapide.

En résumé, la définition de l'architecture du réseau de neurones est une étape cruciale pour la création d'un réseau performant. Les choix que vous faites pour les paramètres d'architecture du réseau peuvent avoir un impact significatif sur la performance et la fonctionnalité du réseau. Il est donc important de comprendre les différentes options disponibles et de choisir celles qui conviennent le mieux à votre problème de classification spécifique.

Le TD3b propose d'entraîner un réseau de neurones à convolution (Convolutional Neural Network, CNN) pour la régression en utilisant MATLAB. L'objectif de la tâche de régression est de prédire une valeur continue plutôt qu'une classe, ce qui la différencie de l'exercice précédent qui était une tâche de classification. Pour cette tâche de régression, nous utiliserons un ensemble de données de prix de maisons à Boston pour entraîner le réseau de neurones.

Dans cet exercice, nous allons construire un réseau de neurones à convolution avec deux couches de convolution, deux couches de sous-échantillonnage, et deux couches entièrement connectées pour résoudre cette tâche de régression. Nous allons également utiliser la fonction d'activation ReLU, la fonction de perte MSE, et l'optimiseur Adam.

Enfin, nous allons évaluer les performances de notre modèle en calculant l'erreur quadratique moyenne (Mean Squared Error, MSE) et la corrélation de Pearson entre les prédictions du modèle et les valeurs réelles de l'ensemble de données de test.

Voici les paramètres que nous avons rencontré dans l'exercice :

- Les couches de convolution : les couches de convolution sont utilisées pour extraire des fonctionnalités à partir des données d'entrée. Dans cet exercice, nous avons défini une couche de convolution 2D avec un noyau de 3x3 et un pas de 1, suivi d'une fonction d'activation ReLU. Cette couche est suivie d'une couche de sous-échantillonnage 2D qui réduit la taille de l'image en prenant le maximum sur des régions de 2x2.
- Les couches de mise en commun : les couches de mise en commun sont utilisées pour réduire la taille de la représentation des données en prenant la moyenne ou le maximum des régions adjacentes. Dans cet exercice, nous avons utilisé une couche de sous-échantillonnage 2D après chaque couche de convolution pour réduire la taille de l'image.
- Les couches entièrement connectées : les couches entièrement connectées sont utilisées pour transformer la sortie des couches précédentes en une sortie finale. Dans cet exercice, nous avons utilisé deux couches entièrement connectées avec 128 et 1 neurones respectivement, suivies de fonctions d'activation ReLU et linéaire. La fonction d'activation linéaire est utilisée pour produire une sortie continue pour la régression.
- La fonction de perte : contrairement à l'exercice précédent qui était une tâche de classification, cet exercice est une tâche de régression, ce qui signifie que nous cherchons à prédire une valeur continue plutôt qu'une classe. Par conséquent, nous avons utilisé une fonction de perte de la moyenne des erreurs quadratiques (Mean Squared Error, MSE) pour évaluer la différence entre la sortie du réseau de neurones et la vérité terrain.
- L'optimiseur : comme dans l'exercice précédent, nous avons utilisé l'optimiseur Adam pour minimiser la fonction de perte.

En résumé, les principales différences entre cet exercice et l'exercice précédent sont les couches de convolution et de sous-échantillonnage, qui sont utilisées pour extraire des fonctionnalités à partir des images d'entrée, ainsi que la fonction de perte, qui est la fonction MSE pour la régression plutôt que la fonction de perte de la croix-entropie catégorielle pour la classification. Ces différences reflètent les différences dans les tâches de classification et de régression, ainsi que les différentes approches nécessaires pour résoudre ces tâches avec un réseau de neurones.

Enfin, le TD3c traite de la conversion d'un réseau de neurones conçu pour la classification en un réseau de neurones pour la régression. Pour cela, l'exercice utilise un modèle de classification de fleurs, qui est un réseau de neurones pré-entraîné qui prend une image d'une fleur en entrée et prédit sa classe, par exemple, rose, tulipe, jonquille, etc.

Dans cet exercice, nous allons convertir ce modèle de classification en un modèle de régression pour prédire la longueur du pétale d'une fleur en fonction de l'image de la fleur. Nous allons d'abord remplacer la dernière couche de classification du modèle par une couche entièrement connectée qui aura une seule sortie. Cette nouvelle couche sera chargée de prédire la longueur du pétale. Ensuite, nous allons ré-entraîner le modèle en utilisant les mêmes images de fleurs mais avec des annotations de longueur de pétale au lieu de la classe de fleur.

Enfin, nous allons tester les performances du modèle de régression en utilisant des images de fleurs de test et en comparant les prédictions du modèle avec les annotations de longueur de pétale réelles. Ce processus de conversion de modèle de classification en modèle de régression peut être utile si vous avez déjà entraîné un modèle de classification sur des données similaires, mais que vous voulez utiliser le même modèle pour effectuer une tâche de régression sur ces données.

Les paramètres d'architecture testés dans l'exercice sont

- Remplacement de la dernière couche : Pour convertir le modèle de classification en un modèle de régression, la dernière couche de classification du réseau de neurones est remplacée par une couche entièrement connectée avec une seule sortie. Cette nouvelle couche sera chargée de prédire la longueur du pétale de la fleur en fonction de l'image en entrée. Cette modification est nécessaire car la tâche a changé de la classification à la régression, et la couche de sortie doit donc être modifiée en conséquence.
- Fonction d'activation de la dernière couche : Comme la tâche est maintenant une régression, il est important de choisir une fonction d'activation appropriée pour la couche de sortie. Dans cet exercice, une fonction d'activation linéaire est utilisée, car elle permet de prédire des valeurs numériques continues plutôt que des classes discrètes.
- Ré-entraînement du modèle : Une fois que la dernière couche a été remplacée, le modèle est ré-entraîné sur les annotations de longueur de pétale. Le processus de ré-entraînement permet d'adapter le modèle à la nouvelle tâche de régression et de modifier les poids du modèle pour prédire la longueur du pétale plutôt que la classe de la fleur.

Les différences entre ce dernier exercice et les deux premiers exercices sont les suivantes :

1. La tâche à accomplir : L'exercice "Create Simple Deep Learning Classification Network" et l'exercice "Train a Convolutional Neural Network for Regression" traitent de tâches différentes, tandis que l'exercice "Convert Classification Network into Regression Network" se concentre sur la conversion d'un modèle de classification en un modèle de régression.
2. Architecture du réseau de neurones : L'exercice "Create Simple Deep Learning Classification Network" utilise un réseau de neurones simple avec une couche cachée, tandis que l'exercice "Train a Convolutional Neural Network for Regression" utilise un réseau de neurones convolutif pour la tâche de régression. Dans l'exercice "Convert Classification Network into Regression Network", le modèle de classification existant est utilisé comme base pour le modèle de régression, et seule la dernière couche est modifiée.
3. Fonction d'activation de la dernière couche : Dans l'exercice "Convert Classification Network into Regression Network", une fonction d'activation linéaire est utilisée pour la dernière couche, car elle permet de prédire des valeurs numériques continues plutôt que des classes discrètes, contrairement à l'exercice "Create Simple Deep Learning Classification Network" où une fonction d'activation softmax est utilisée pour la dernière couche.

Pour résumer le TD3, ses trois exercices couvrent différents aspects de l'apprentissage profond, tels que la classification et la régression, en utilisant différents types de réseaux de neurones. Voici un résumé de ce que nous avons appris grâce à ces exercices :

Le premier exercice nous a permis de comprendre les concepts fondamentaux de l'apprentissage profond en construisant un réseau de neurones simple pour la classification d'images. Nous avons également appris à évaluer les performances d'un modèle en utilisant la matrice de confusion.

Le deuxième nous a permis de comprendre le fonctionnement des réseaux de neurones convolutifs et leur utilisation dans la tâche de régression. Nous avons également appris à évaluer les performances d'un modèle de régression en utilisant différentes mesures d'erreur, telles que la MSE et la MAE.

Et le troisième nous a permis de comprendre comment convertir un modèle de classification en un modèle de régression en modifiant la dernière couche du réseau de neurones. Nous avons également appris à utiliser une fonction d'activation linéaire pour prédire des valeurs continues, plutôt qu'une fonction d'activation softmax pour prédire des classes discrètes.

En somme, ces exercices nous ont permis de comprendre les concepts fondamentaux de l'apprentissage profond et nous ont fourni des connaissances pratiques pour construire et évaluer des modèles de réseaux de neurones pour des tâches de classification et de régression.