# Tensorflow - DL

Ali.zainoul.az@gmail.com
Ali ZAINOUL

9 janvier 2024

# Objectif de l'exemple

L'objectif de cet exemple est de créer, entraîner et évaluer un modèle de réseau neuronal dense en utilisant TensorFlow. Le modèle est conçu pour effectuer une classification multi-classes, comme la reconnaissance de chiffres écrits à la main.

## Architecture du Modèle

Le modèle est une séquence de trois couches : une couche dense, une couche de dropout, et une autre couche dense pour les prédictions.

## Utilité

Ce code illustre la création d'un modèle simple pour des tâches de classification avec TensorFlow, en mettant en avant les principaux paramètres et détails architecturaux.

# Source Code

```
import tensorflow as tf

# Creation du modele
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu',
        input_shape=(784,)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Affichage de l'architecture
print(model.summary())
```

# Explication des Paramètres

## Couche Dense (Première Couche)

- 128 neurones : Chaque neurone reçoit des entrées de toutes les 784 caractéristiques et contribue au processus d'apprentissage.
- `activation='relu'` : Fonction d'activation ReLU pour introduire une non-linéarité.

## Couche Dropout (Deuxième Couche)

- 0.2 : Taux de dropout. 20% des neurones sont désactivés de manière aléatoire pendant l'entraînement pour prévenir le surajustement.

## Couche Dense (Troisième Couche - Couche de Sortie)

- 10 neurones : Pour une classification multi-classes.
- `activation='softmax'` : Fonction d'activation softmax pour la classification multi-classes.

# Conclusion

En résumé, cet exemple illustre la création d'un modèle de réseau neuronal dense avec TensorFlow, en mettant en évidence les paramètres clés et l'architecture. Ce modèle peut être utilisé comme point de départ pour des tâches de classification simples.

# Introduction

1. **Chargement des données :** L'ensemble de données MNIST, composé d'images en niveaux de gris de 28x28 pixels de chiffres écrits à la main (0 à 9), est chargé en utilisant l'ensemble de données intégré de TensorFlow.

# Introduction (suite)

2. **Prétraitement des données :** Les valeurs des pixels des images sont normalisées dans la plage [0, 1] pour faciliter l'entraînement du modèle. Les étiquettes de classe sont encodées en one-hot à l'aide de la fonction utilitaire `to_categorical`.

# Architecture du Modèle

3. **Architecture du modèle :** Un modèle séquentiel est défini avec trois couches :
   - Couche `Flatten` : Aplatit l'image en un vecteur.
   - Couche dense (cachée) : 128 neurones avec une activation ReLU.
   - Couche dense (de sortie) : 10 neurones avec une activation softmax pour la classification multi-classe.

# Compilation du Modèle

4. **Compilation du modèle :** Le modèle est compilé avec l'optimiseur Adam, la perte catégorielle crossentropy et la précision comme métrique d'évaluation.

# Entraînement du Modèle

5. **Entraînement du modèle :** Le modèle est entraîné sur les données d'entraînement pendant 5 époques avec une division de validation de 10% pour surveiller la progression de l'entraînement.

# Évaluation du Modèle

6. **Évaluation du modèle :** Le modèle entraîné est évalué sur les données de test, et la précision du test est imprimée.

# Sauvegarde du Modèle

7. **Sauvegarde du modèle :** Le modèle entraîné est sauvegardé au format SavedModel pour une utilisation future.

# Chargement et Prédiction du Modèle

8. **Chargement et prédiction du modèle :** Le modèle sauvegardé est chargé, et des prédictions sont effectuées sur un petit sous-ensemble de données de test.