EPI ECOLE PLURISDISCIPLINAIRE INTERNATIONALE AU : 2022-2023	TP4	IA2 Marwa Jabberi Slama Yousra Haj Hassen

Objectifs :

Evaluation et test d'un modèle MLP.

Dans cet exemple, nous définissons un modèle MLP avec deux couches cachées **de 64 neurones chacune**, et une couche **de sortie de 10 neurones** pour la classification des chiffres de 0 à 9.

Le modèle est compilé avec **l'optimiseur Adam** et la fonction de perte de **catégorisation croisée**.

Nous utilisons les données **MNIST** qui sont chargées et prétraitées avant d'être utilisées pour entraîner le modèle pendant **10 époques**. Enfin, le modèle est évalué sur **les données de test** et la précision est affichée.

Epochs (les époques) : correspondent au nombre de fois que l'ensemble des données d'entraînement est passé à travers le réseau de neurones. En d'autres termes, une époque signifie que chaque exemple d'entraînement a été vu une fois par le réseau de neurones. L'entraînement est généralement effectué sur plusieurs époques afin que le réseau de neurones puisse ajuster progressivement ses paramètres pour mieux s'adapter aux données d'entraînement.

Batch size : correspond au nombre d'exemples d'entraînement utilisés dans chaque étape de mise à jour des poids du réseau de neurones. Les données d'entraînement sont généralement divisées en lots (ou mini-lots) de taille fixe pour être traitées par le réseau de neurones. Le choix d'une taille de lot appropriée dépend de divers facteurs, tels que la taille de l'ensemble de données, la complexité du modèle et la quantité de mémoire disponible.

1. **Importations :**

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

2. **Définir le modèle MLP :**

```
# Définir le modèle MLP
model = keras.Sequential(
[
            layers.Dense(64, activation="relu", input_shape=(784,)),
            layers.Dense(64, activation="relu"),
            layers.Dense(10, activation="softmax"),
        ]
    )
```

3. **Compilation du modèle :**

```
    # Compiler le modèle
    model.compile(
        optimizer=keras.optimizers.Adam(lr=0.001),
        loss="categorical_crossentropy",
        metrics=["accuracy"],
    )
```

4. **Chargement et prétraitement des données :**

```
    # Charger les données MNIST
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
# Prétraiter les données
x_train = x_train.reshape(60000, 784).astype("float32") / 255
x_test = x_test.reshape(10000, 784).astype("float32") / 255
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)
```

5. **Lancement de l'apprentissage:**

```
# Entraîner le modèle
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=
0.2)
```

6. **Evaluation du modèle :**

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print("Accuracy:", test_acc)
```

7. **Avec le même exemple précèdent, essayez de varier les hyperparamètres, métriques d'évaluations, … que remarquez-vous ?**