TD1 : Identification des chiffres – MNIST

1. Prétraitement des données

- Télécharger *fashion mnist* de la bibliothèque *Keras.Datasets*.
- Comment sont codées les données, de quel type sont-elles. Quelle est forme des données taille des données d'entrée et des *targets* pour le *train* et *test*.
- Comment se répartissent les *targets* au sein des deux jeux de données. Cette répartition est-elle cohérente pour les modèles d'apprentissage.
- Affichez quelques images en utilisant la fonction imshow de pyplot.
- Quels traitements doivent être appliqués à ces données pour un traitement par un MLP. Expliquez pourquoi faire ces prétraitements.
- Qu'aurait-il fallu faire si la répartition était déséquilibrée.
- Transformer vous données afin qu'elles puissent être traitées correctement par un réseau de neurones.

Info : vous utiliserez une couche input(shape=...) comme première couche du modèle correspondant à la taille des images, ainsi qu'une seconde couche permettant d'aplatir les images : couche flattent sans paramètre.

2. Création d'un premier modèle

- De quel type doit être la couche de sortie (nombre de neurones, fonction d'activation). Expliquez vos choix.
- On souhaite créer un premier model avec simplement deux couches cachées identiques de même taille (256 neurones) et avec des fonctions d'activation de type *relu*.
- Combien y-a-t-il de poids (paramètres) à caler. Détaillez les nombres de poids obtenus par couches et par neurones.
- Quel est selon vous les meilleurs paramètres pour la fonction *compile*. Donnez une petite explication pour les trois paramètres (fonction de perte, fonction d'apprentissage, mesure de la qualité du réseau).

- Si l'on souhaite mettre à jour nos poids moins de 200 fois en ayant un pourcentage de validation de 20%, quelle est la taille des *batchs*.
- Entrainez le modèle sur 20 *epochs*, avec les paramètres précédents (*batchs* et pourcentage de validation).
- Affichez maintenant le taux d'erreur sur le test et la validation. Que constatez-vous, pensez-vous que pour notre réseau il soit judicieux de faire plus de 20 *epochs*.

3. Traitement des résultats

- Affichez les courbes correspondant aux erreurs et à l'accuracy sur les parties train et validation. Commentez les résultats obtenus.
- Effectuer une prédiction sur les données de test. Évaluer les performances du modèle. Ces performances sont-elles cohérentes par rapport aux résultats obtenus sur la partie validation au cours du *fit*.
- Transformer les résultats de la prédiction, en valeurs comprises entre 0 et 9, correspondant au numéro. Enregistrez les prédictions et les valeurs des sorties test en *dataframe*, pour faciliter les traitements ultérieurs.
- Affichez pour chacun des objets de la base de données le nombre d'erreurs commis. Quel est le nombre le plus mal identifié.
- Affichez quelques images mal classées que vous choisirez parmi celles qui ont été le plus mal classées.
- Faites de même en choisissant maintenant des classements qui peuvent correspondre à a des valeurs très éloignées en termes de forme, comme par exemple un 1 classé comme 8.
- Isolation forest de scklearn est une technique de détection d'anomalies (outliers). Identifiez grâce à cette technique les anomalies pour les images correspondants à des 1 (par exemple). Vous paramétrez la contamination afin d'identifier moins de 10 anomalies.

```
Exemple d'utilisation
from sklearn.ensemble import IsolationForest
#Model susceptible d'identifier 1% des images les plus anormales
model = IsolationForest(contamination=0.01)
model.fit(X)
#predict retourne un vecteur avec -1 sur les index des anomalies
model.predict(X)
```

- Affichez les résultats obtenus. Vérifiez si ces anomalies correspondent aux erreurs commises par le réseau de neurones.
- Affichez les taux d'erreurs sur les parties test et validation. Que constatez-vous par rapport au modèle à deux couches cachées. Est-il judicieux d'ajouter cette nouvelle couche.

Info : si vous récupérez les résultats de la fonction fit, vous pourrez accéder aux colonnes correspondantes aux erreurs et aux accuracy sur le train et la validation dans l'attributs history.

4. Modification du modèle

- Ajoutez une troisième couche cachée de même taille que les précédentes. Le modèle est-il meilleur.
- Reprenez le modèle à deux couches cachées et divisez le nombre de neurones par deux. Les résultats sont-ils dégradés.
- Un couche dropout(val) placée à après une couche Dense désactive aléatoirement une fraction (égale à val) des neurones. A chaque bachs des neurones différents sont désactivés. Ajouter à la sortie des deux couches cachées des couches de dropout avec un taux de désactivation de 40%.
- Effectuer l'apprentissage et affichez à nouveau les courbes des taux d'erreurs. Que peut-on en conclure.
- Augmentez le nombre de mise à jour des poids, en divisant par deux la taille de batchs. Quel impact cela a-t-il sur les taux d'erreurs, et sur les temps de calcul.
- Si maintenant on utilise des fonctions d'activation de type sigmoïde ou relu sur les couches cachées, cela a-t-il un impact réel pour ce jeu de données.
- Effectuer maintenant en conservant les couche dropout() des tests avec [16, 32, 64, 128] neurones par couche.
- Compte tenu de tous ces tests quel bilan peut-on faire, quel serait selon vous le meilleur modèle, justifiez votre réponse.