1. Mise en forme des données

- Télécharger *fashion mnist* de la bibliothèque *Keras.Datasets*.
- On souhaite utiliser un réseau de convolution pour traiter les données. Expliquer comment doit être constitué un réseau de convolution pour le traitement de ces données.
- On prévoit d'utilise la fonction de perte *sparse_categorical_crossentropy*. Comment doivent être modifié les données d'entrée et de sortie. Expliquer pourquoi réaliser (ou de pas réaliser) ces transformations.
- Créer la liste des *targets*, correspondant aux données suivantes :

  T-shirts, Pantalons, Pulls, Robes, Manteaux, Sandales, Chemises, Baskets, Sacs, Bottines.

- Afficher les neuf premières entrées avec le nom de leur *target*.
- Transformer vous données afin qu'elles puissent être traitées correctement par un réseau de convolution.

2. Création d'un premier modèle

- De quel type doit être la couche de sortie (nombre de neurones, fonction d'activation).
- Créez un model avec deux couches cachées pour la partie convolution avec 16 noyaux de taille (5x5) et 32 de taille (3x3), et deux couches pour la partie de classification de taille 256 et 128 neurones. Les couches de *pooling* doivent diviser par deux la taille des images. Vous utiliserez un layer (input) et des données d'entrée non régularisées.
- Combien y-a-t-il de poids (paramètres) à caler. Détaillez les nombres de poids obtenus par couches et par neurones.
- Analysez les résultats obtenus.
- Ajoutez des couches de dropout après chaque couche de *pooling* et chaque couche *dense*. Que constatez-vous.
- Affichez 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*.

- Si l'on ajoute une couche dense de 128 neurones le modèle est-il amélioré. Que peut-on en déduire. En se basant sur les connaissances acquises sur les images pensez-vous que le modèle peut être amélioré.

3. Traitement des données cifar

- Télécharger *cifar10* 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*.
- Créer la liste des *targets*, correspondant aux données suivantes :

  Avion, Voiture, Oiseau, Chat, Biche, Chien, Grenouille, Cheval, Bateau, Camion.

- Afficher les vingt-cinq premières entrées avec le nom de leur *target*.
- Utiliser un premier modèle de convolution identique au premier utilisé pour le traitement des données *fashion*.
- Combien y-a-t-il de poids (paramètres) à caler. Détaillez les nombres de poids obtenus par couches et par neurones.
- Effectuez un apprentissage avec les mêmes paramètres que pour le modèle précédent, et afficher les résultats obtenus sur les données de test. Que constatez-vous.
- Normalisez maintenant les données d'entrée (en les divisant par 255), analysez les résultats obtenus. Le modèle doit-il être amélioré.
- Ajouter deux couches de dropout avec un taux de désactivation de 50% dans la partie de classification.
- Les performances sont-elles cohérentes par rapport aux résultats de la partie validation. Analysez et expliquez les résultats obtenus.
- Afficher via la fonction *heatmap* la matrice de confusion entre les valeurs de test et de prédiction. Quelle est la classe la plus mal évaluée.

4. Amélioration du modèle

- Compte tenu de la mauvaise qualité des images, on va essayer d'augmenter la recherche de *features* en ajoutant des couches de

convolution. Que se passerait-il si l'on ajoutait deux nouvelles couches de convolution avec des couches de *pooling*.

- On se propose d'ajouter deux couches de convolution sans *pooling* avant les deux couches de convolution précédentes. Les noyaux de toutes les couches seront de taille 3.
- Combien y-a-t-il de poids (paramètres) à caler. Détaillez les nombres de poids obtenus par couches et par neurones.
- Afficher via la fonction *heatmap* la matrice de confusion entre les valeurs de test et de prédiction.
- On souhaite ne pas perdre d'informations sur les bords. Utilisez alors une valeur de *padding* permettant de conserver tous les pixels.
- Comment évolue l'*accurancy* entre les valeurs d'apprentissage et de validation. Ajouter alors des *dropout(0.25)* après les couches de *pooling*. L'apprentissage est-il amélioré sur la partie train.
- Quelles sont les trois catégories d'erreurs les plus fréquentes. Affichez 25 des erreurs les plus fréquentes.

5. Etude de quelques modifications

- Devant la qualité plus que mauvaise des images, on cherche à savoir si en les analysant plus profondément le modèle peut être amélioré. Ajouté deux couches de convolution identiques aux précédentes avec 64 convolutions chacune, d'un *pooling* et d'un *dropout(0,25)*.
- Le modèle étant plus profond, on souhaite éviter le *Vanishing gradient* dans le modèle pour cela on ajoute des couches de normalisation, au total 5. Trois après 2 couches de *convolution* et 2 après les couches *Dense*.
- Comment se comporte le modèle, évaluer les résultats obtenus sur la partie de test. Quels sont maintenant les deux classes les plus mal prédites.
- Comment évolue la courbe de validation. Pensez-vous qu'il soit nécessaire d'augmenter le nombre d'*epoch*.
- Effectuer le même test avec 20 *epoch* que deviennent les résultats.
- Afin d'avoir un petit gradient lorsque les sorties des couches d'apprentissage sont négatives on peut utiliser des fonctions

d'activation de type *LeakyRelu* avec une valeur de 0.3 pour la *negative_slop*. Réaliser un apprentissage avec 40 *epoch*, de *batch* de 128.

- Evaluer les résultats obtenus sur le train, le test et la validation. Quel modèle semble être le plus efficace si on prend en compte tous les paramètres, les résultats sur la partie test, le sur-apprentissage, le temps de calcul ….