1 Introduction a keras

Keras est un API populaire et haut niveau de machine learning qui permet de faire entre autre de la classification d'images et de textes mais aussi de la regression (estimation de données).

On s'interesse à la segmentation et à la classification d'images. À chaque image l'intelligence artificielle va apprendre à classifier les differentes regions de l'image en les associants à des etiquettes. Dans le cas d'une image thoraxique d'un chien, une etiquette peut être le coeur, les os, les vertebres, etc...

Cette API permet de construire des reseaux de neurones convolutifs à partir de :

- couches convolutives qui scannent l'image à partir de filtre en realisation une convolution
- couches de pooling qui sous-echantillonent l'image selon deux operations :
  - Chaque opération de pooling sélectionne la valeur maximale de la surface
  - Chaque opération de pooling sélectionne la valeur moyenne de la surface
- couches denses (connectés)

Elle manipule des couches de neurones complexes qui realise different traitement sur un jeu de données en entrée d'un modele.

On instance un modele avec la classe *keras.Sequential*, elle definit l'ensemble des couches de neurones du modele. Le module *keras.layers* definit une couche de neurones, ce module contient tous les types de couches possible pour construire un reseaux de neurones.

Un exemple d'instanciation :
$model = tf.keras.Sequential([$
tf.keras.layers.Flatten(input$_shape = (28, 28)$),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(10)])

La classe Sequential génère trois couches de neurones avec les modules de layers.

Avant que le modèle ne soit prêt pour l'entraînement, il a besoin de quelques paramètres supplémentaires:

- Fonction de perte - Cela mesure la précision du modèle pendant l'entraînement. Vous voulez minimiser cette fonction pour avoir un modele fiable et eviter le sur-apprentissage
- Optimiseur: c'est ainsi que le modèle est mis à jour en fonction des données qu'il voit et de sa fonction de perte

- Métriques - Utilisées pour surveiller les étapes de formation et de test. L'exemple suivant utilise la précision , la fraction des images correctement classées

Ceux-ci sont ajoutés lors de l'étape de compilation du modèle:
$model.compile(optimizer =, loss =, metrics =)$
L'apprentisage du modèle de réseau neuronal nécessite les étapes suivantes:

- Envoyez les données d'entraînement au modèle
- Le modèle apprend à associer des images d'entrainement et des étiquettes
- Vous demandez au modèle de faire des prédictions sur un jeu de test

Et enfin on commence l'entrainement du modele avec la methode *fit*:
$model.fit(train_images, train_labels)$