TD réseau de neurone 2 : faire du transfert learning avec un model pre-entrainé : partie training du NN

**But :**

Utiliser pytorch + resnet pre-entrainé faire un transfert learning et avoir un détecteur de chat.

Si l’image est un chat => prédiction du NN : classe = CAT

Si l’image n’est pas un un chat => prédiction du NN : classe = NO CAT

**fichier du TD** : **TP\_pytorch\_xxxx.py**

**Le repo et les images :**

**data/test\_images** on trouve 2 images => elles sont utilisées pour le TP\_1 pour voir comment fonctionne le resnet pré-entrainé

**data/eval\_inference** : on trouve 2 repo (cat/no\_cat) => ce repo est utilisé pour générer le dataset qui sera utilisé pour tester l’inference une fois le training terminé. Les images ici ne doivent pas être présentes dans le dataset de training

**data/tmp** => des images de réserves, non présente ailleurs pour tester l’impact des data sur le resultat du training

**images** => on trouve 2 repo (cat/no\_cat) => ce repo est utilisé pour générer le dataset utilisé lors du training. Il sera divisé en 2 parties train\_dataset, eval\_dataset

pour faire un training il nous faut :

* un dataset
* un model
* une boucle sur les epochs :
  + model.train (en utilisant le dataset de training)
  + model.eval (en utilisant le dataset de validation)

les trucs a tester pour voir ce qui se passe :

* changer le learning rate
* changer le weight\_decay
* changer le nombre d’epoch
* changer les images qui sont dans le dataset de training (prendre dans data/tmp pour faire des variations
* changer l’optimizer => prendre un optimizer pytorch différent d’Adam
* changer la loss function => prendre une loss function differente de CrossEntropy
* ajouter un scheduler

pour ceux qui veulent voir ce qui se passer : ajouter tensorboard