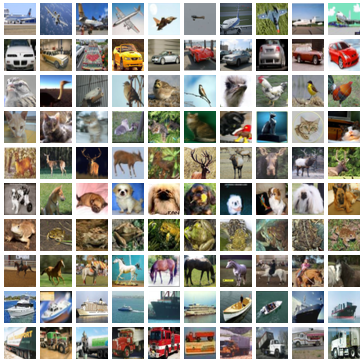
**Deep Learning**

**Projet CIFAR10**

Le CIFAR 10 :

C'est un Dataset contenant 60 000 images en 32 pixels par 32 en couleurs. Nous avons pour celui-ci 10 catégories d'image :

**Avions**

**Voitures**

**Oiseaux**

**Chats**

**Cerfs**

**Chiens**

**Grenouille**

**Chevaux**

**Bateaux**

**Camion**

Notre but est donc de lancer un algorithme d'apprentissage permettant de catégoriser ces images correctement grâce au Deep Learning.

Nous avons donc testé différents codes en python sous l'API Keras regroupant notamment les backends Tensorflow et Theano.

La première étape de notre code était d'importer le dataset cifar10 que nous avons fait à l'aide de la commande load\_data().

Une fois ceux-ci importés dans notre base d'apprentissage et de test, il nous fallut utiliser un reshape dans le but de transformer le dataset en un vecteur pour les entrées des deux bases.



Les options 32, 32, 3 correspondent à la longueur, la largeur de l'image ainsi que 3 tensors étant la couleur.

Ensuite nous transformons les vecteurs de la base d’apprentissage et de test pour qu’ils nous retournent une matrice de classe binaire.



La deuxième étape fut de préparer le modèle que nous allions utiliser. Nous avons pris le modèle Sequential qui est un groupe de couches linéaires.

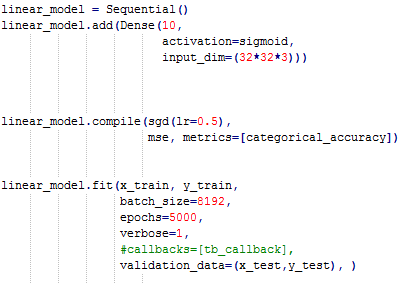
Dans ce modèle nous devons donc ajouter des couches de neurones puis les compilées à l’aide d’un optimizer pour enfin à l’aide de la commande fit() lancer l’apprentissage en précisant donc :

* **Le batchsize** = le nombre de test pouvant être réalisés en une fois durant une epoch (choisi en fonction de notre mémoire allouée ou bien de la puissance de notre carte graphique (pour tensorflow-gpu)).
* **Les epochs** = Le nombre d’itération d’apprentissage
* **Les callbacks** = type et emplacement des logs qui seront créés sur le run.
* **Validation\_data** = précise quelle est la base de test/ de validations

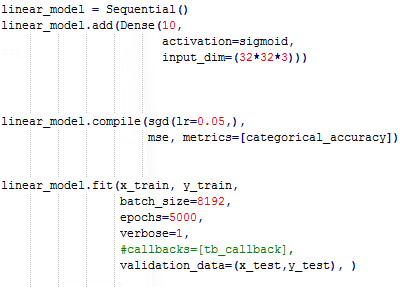
**Phase d’apprentissage**

**Perceptron Multi-couches (MLP)**

Pour commencer nos tests nous sommes partis comme pour le MNIST à tester l’apprentissage tout d’abord sur un perceptron multicouches pour avoir une référence d’apprentissage.

Nous avons donc commencé avec une seule couche

Ici nous avons donc une seule couche sigmoid de 10 neurones avec un learning rate de 0.5 sur 5k d’epochs. Ce run nous a donné %.

Nous avons alors modifiés le learning rate pour le mettre à 0.05 sur 5k d’epochs.

Ce run a donné un résultat évidemment inférieur ( %) puisque pour rendre le learning rate intéressant et plus important, il aurait fallu aussi augmenter significativement le nombre d’epochs.

Nous avons ensuite testés d’ajouter des couches à ce modèle en commençant par une couche de sigmoid de dense 32. Ce run a atteint %.

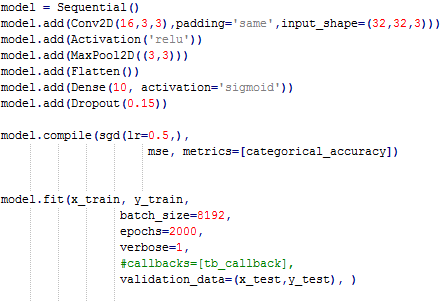
Le prochain run a été testé avec une tangeante hyperbolique à la place de la sigmoid toujours avec 32 de dense. Ce run a atteint % ce qui est mieux.

Nous avons alors comparés ces Runs grâce à Tensorboard :

**Réseau neuronal convolutif (CNN)**

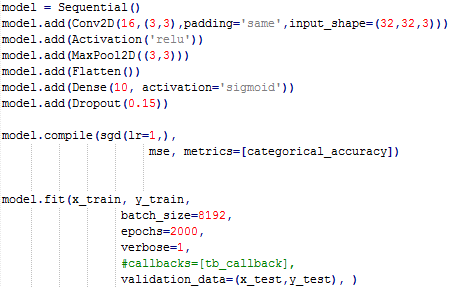
Comme nous avons pour travail de travailler sur des images, il est intéressant d’utiliser les réseaux de convolutions pour notre apprentissage.

Nous avons commencés par une seule couche de convolution de 16 filtres (nombre de neurones avec leurs poids associés) avec des matrices 3\*3 toujours de la même forme\*. Ensuite nous lui avons ajouté une couche d’activation ‘relu’ car ………. Ensuite nous avons ajouté une couche de maxpool2D qui consiste donc à diviser l’image en plusieurs images réduites (ici de 3\*3) en gardant les plus hautes valeurs de chaque matrice. (Particulièrement utilie dans la réduction du coût de calcul mais aussi dans la prévention contre l’overfitting).

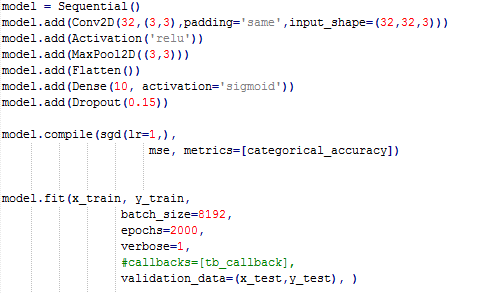
*\*A noté que l’input\_shape n’est précisée que sur la première couche car les couches suivantes utiliserons automatiquement la même que la première.*

Ce run a durée 2k d’epochs et a atteint % Ce qui est une nette amélioration grâce à la convolution.

Nous avons ensuite monté le learning rate à 1 pour avoir un réel résultat plus rapidement.



Ce Run nous a permis d’atteindre %

Ensuite nous avons tester le changement du nombre de filtre sur la couche de convolution et nous sommes donc monté à 32.

Le résultat fût de %