Faculté des Sciences de Montpellier

Master 2 ICO

Classification Supervisé avec des Réseaux de Neurones

Etudiants: GUEYE Fagaye Sarr
CASIERRA Cristhian
FAYE Elie Wagane
EBERLIN Anthony

December 8, 2021

Table des matières

1 Introduction

Le but de ce projet est un classification d'images avec le jeu de données cifar-10. Pour ce projet nous avons utiliser les deux API proposés (Pytorch et Keras). Ce document présente notre démarche dans le choix des modéles et leurs évalutions.

2 Modèles

2.1 Keras

Dans cette partie, nous avons considéré d'utiliser deux modèles différents. Nous avons d'abord créer un modèle en ANN(réseaux de neurones artificiels) qu'on a évalué et tenté d'améliorer. Ensuite nous avons créer un modèle CNN (réseaux de neurones convolutifs), que nous avons aussi testé et tenté d'améliorer.

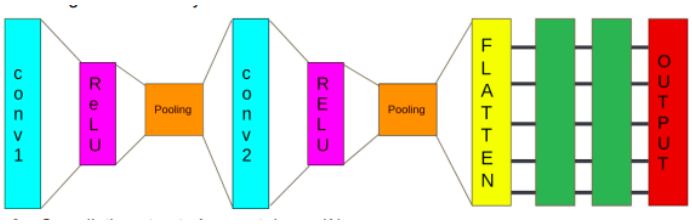
2.1.1 Description des modèles

2.1.1.1 Modéle 1 ANN

Ce modèle est composé de 4 couches, d'abord nous avons une couche de Flaten pour aplatir nos entrées, ensuite nous avons mis une couche dense composé de 3000 neurones avec une fonction d'activation Relu. Par dessus cette couche, nous avons rajouté une autre couche dense de 1000 neurones avec la même fonction d'activation et enfin nous avons rajouter une petite couche dense de 1000 neurones avec une fonction d'activation sigmoïd.

2.1.1.2 Modèle 2 CNN

Pour ce modèle nous avons utiliser la même architecture qui était proposé pour le projet. voir l'image ci dessous.



2.2 Pytorch

2.2.1 Réseau de neurones résiduels

Chaque architecture gagnante suivante utilise plus de couches dans un réseau neuronal profond pour réduire le taux d'erreur. Cela fonctionne pour moins de couches, mais lorsque nous augmentons le nombre de couches, il y a un problème courant dans deep learning associé à celui appelé gradient de disparition / explosion. Cela fait que le gradient devient 0 ou trop grand. Ainsi, lorsque nous

augmentons le nombre de couches, le taux d'erreur d'apprentissage et de test augmente également. Afin de résoudre le problème du gradient de disparition / explosion, cette architecture a introduit le concept appelé Réseau résiduel. Dans ce réseau, nous utilisons une technique appelée sauter les connexions . La connexion saute la formation de quelques couches et se connecte directement à la sortie.

2.2.2 Réseau de neurones convolutifs

Les réseaux de neurones à convolution profonde sont devenus les méthodes de pointe pour les tâches de classification d'images. Cependant, l'une de leurs plus grandes limites est qu'ils nécessitent beaucoup de données annotées (images dont la classe à prédire est connue). Par exemple, un réseau ayant pour unique tâche de reconnaître des chats, devra être entraîné avec des milliers de photos de chats avant qu'il ne puisse discerner cet animal d'une autre entité avec une bonne précision. Autrement dit, plus le jeu de données d'apprentissage est important, meilleure sera la précision de l'algorithme. On utilise l'approche deep learning dans lequel le modèle dispose de plusieurs instances (image) par classe dans son jeu de données d'apprentissage et doit apprendre à ré-identifier cette instance dans les données de test. L'image d'entrée est introduite dans une série de couches de convolution, qui génère une distribution de probabilités sur toutes les classes.

3 Entraînement et évaluation

3.1 Keras

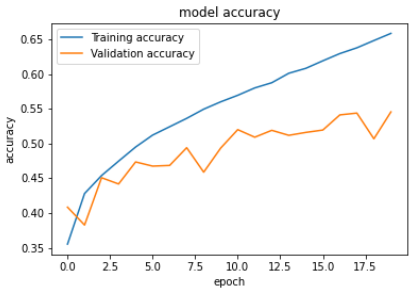
3.1.1 Modèle 1 Ann

Pour la configuration du modèle pour l'entraînement nous avons choisi en premier lieu un optimiseur stochastic gradient descent (SGD). Pour la fonction de perte nous avons choisi sparse_categorical_crossentropy qui est recommandé pour le jeu de données que nous avons. Et enfin on choisi comme moyen de mesure l'accuracy. Nous allons vous présenter les résultats obtenu pour ce modèle avec 20 époches (cf fig. 1).

Dans le but d'améliorer le modèle, nous avons changer l'optimiseur en Adam pour voir ce que ça pourrait faire. les résultats étaient moins bon. Pour ce modèle nous obtenons une accuracy 0.55, on ne s'est pas attardé à vouloir l'optimiser. Nous avons choisi le modèle cnn qui présente des résultats plus correctes.

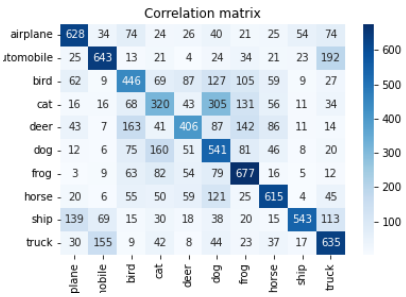
Nous constatons que la courbe d'accuracy et la courbe de validation accuracy vont bien dans le même sens. Néanmoins nous voyons qu'un écart important se creuse de plus en plus à partir de l'époches n°5 entre ces deux courbes. cela signifie que nous sommes dans une situation de sur apprentissage (overfitting) Nous pouvons faire un analyse analogue concernant le graphique de training loss. Le modèle ANN arrive à faire la distinction entre les 10 catégories d'images prises

depuis tensorflow. Les avions, voitures, bateaux, camions sont identifiés assez facilement par la matrice de corrélation. En revanche le modèle ANN a du mal à différencier les oiseaux,chats, crapauds cela est sûrement dû au faite qu'ils ont de nombreuses caractéristiques communes du faite qu'ils appartiennent tous au groupe des mammifères. Il faut trouver un modèle plus précis pour les distinguer.



(a) Training Accuracy

(b) Training Loss



(c) Matrice de correlation

Classfication Report

	precision	recall	f1-score	support
0	0.64	0.63	0.63	1000
1	0.67	0.64	0.66	1000
2	0.45	0.45	0.45	1000
3	0.38	0.32	0.35	1000
4	0.54	0.41	0.46	1000
5	0.38	0.54	0.45	1000
6	0.54	0.68	0.60	1000
7	0.63	0.61	0.62	1000
8	0.79	0.54	0.64	1000
9	0.54	0.64	0.59	1000
accuracy			0.55	10000
macro avg	0.56	0.55	0.55	10000
weighted avg	0.56	0.55	0.55	10000

(d) Classification report

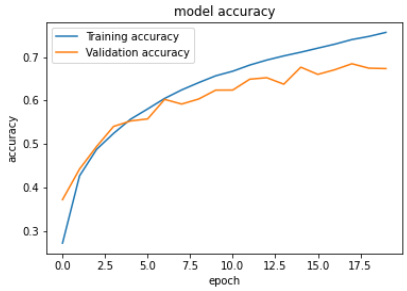
FIGURE 1 – Resultats modele 1 Ann

3.1.2 Modèle 2 Cnn

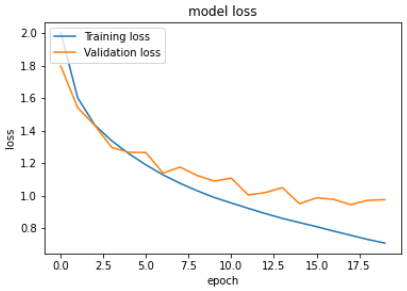
Pour la configuration du modèle pour l'entrainement nous avons choisi en premier lieu un optimseur Adam mais les résultats étaient meilleurs avec l'optimiseur SGD. Pour la fonction de perte nous avons choisi sparse categorical crossentropy qui est recommandé pour le jeu de données que nous avons. Et enfin on choisi comme moyen de mesure l'accuracy. Nous allons vous présenter les résultats obtenu pour ce modèle avec 20 epochs et avec l'optimiseur SGD. (cf fig. 2)

La courbe de training accuracy et de validation accuracy restent très peu espacés au fil de l'évolution de nombre d'epochs. De plus l'accuracy du modèle CNN au bout de 20 epochs est de 0.67. Nous pouvons en déduire que le modèle CNN est bien adapter pour effectuer des classifications pour les images.

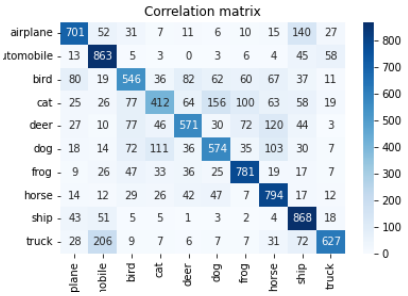
La corrélation entre les images représentants des mammifères est beaucoup plus faible dans le modèle CNN que dans le modèle ANN vu précèdemment.



(a) Training Accuracy



(b) Training Loss



(c) Matrice de corrélation

Classfication Report

	precision	recall	f1-score	support
0	0.73	0.70	0.72	1000
1	0.67	0.86	0.76	1000
2	0.61	0.55	0.58	1000
3	0.60	0.41	0.49	1000
4	0.67	0.57	0.62	1000
5	0.63	0.57	0.60	1000
6	0.72	0.78	0.75	1000
7	0.65	0.79	0.72	1000
8	0.65	0.87	0.75	1000
9	0.79	0.63	0.70	1000
accuracy			0.67	10000
macro avg	0.67	0.67	0.67	10000
weighted avg	0.67	0.67	0.67	10000

(d) Classification report

FIGURE 2 – Résultats modèle 1 Cnn

Au vu de la précision assez limité des modèles ANN et CNN sous tensorflow nous avons décidé d'effectuer des modèles sous pytorch pour obtenir des modèles plus performants.

3.2 Pytorch

Pour la classification on a créé une classe de base avec des fonctions et deux classes un pour le modèle utilisant le réseau de neurone convolutif et l autre pour le réseau résiduel

3.2.1 Modèle à Neurones convolutives

Le processus consiste a linéariser après avoir passer sur plusieurs couches convolutions. Chaque couche reçoit en entrée des données et les renvoie transformées. Pour cela, elle calcule une combinaison linéaire puis applique éventuellement une fonction non-linéaire, appelée fonction d'activation. Les coefficients de la

combinaison linéaire définissent les paramètres (ou poids) de la couche.Cet empilement de couches définit la sortie finale du réseau. La dernière couche calcule les probabilités finales en utilisant pour fonction d'activation la fonction logistique (classification binaire) ou la fonction softmax (classification multi-classes). Une fonction de perte (loss function) est associée à la couche finale pour calculer l'erreur de classification.

3.2.2 modèle à Neurones Residuels

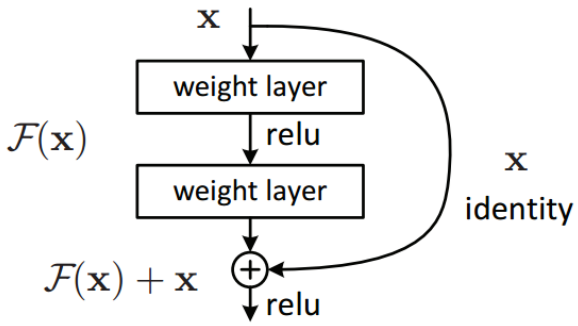
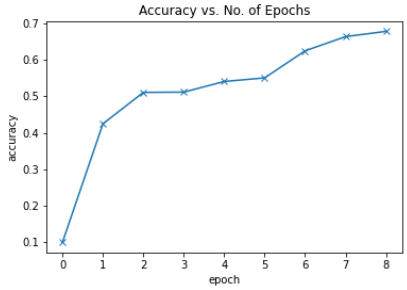
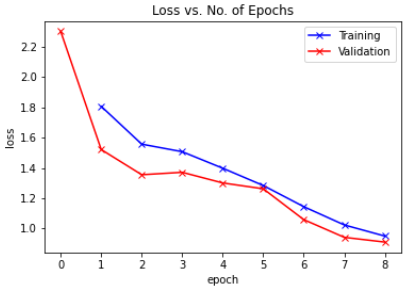


FIGURE 3 – Bloc Résiduel

Dans le Réseaux de Neurones Résiduels, le processus se concentre dans les blocs, chaque bloc composé d'une couche convolution suivi, d'une couche de normalisation qui approfondissent les relations, et ajoutent, dans les cas des couches résiduels, les relations résiduels de la couche précédente, permettant de réduire la perte de manière conséquente entre les donnés d'entraînements et les données de validation. Cette structure nous a permis d'obtenir les meilleurs résultats inter-changeant sur les paramètres de taille de batch, optimiseur, taux d'apprentissage et de quantité d'époches (qui sont changés dynamiquement grâce à une fonction de programme)(cf fig. 5). La différence entre paramètres n'étant pas significative par rapport au comparaison avec le structure utilisé dans le modèle précèdent. une validation finale avec une augmentation de données en entraînant et les meilleurs paramètres sélectionnés nous a donnée une précision à hauteur de 90% (cf fig. 6)

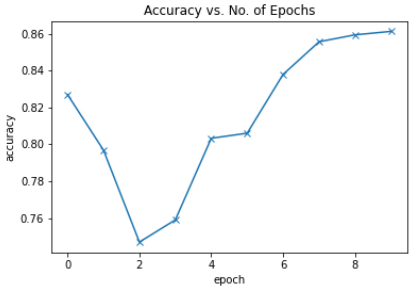


(a) CNN Accuracy à batch$_size$128

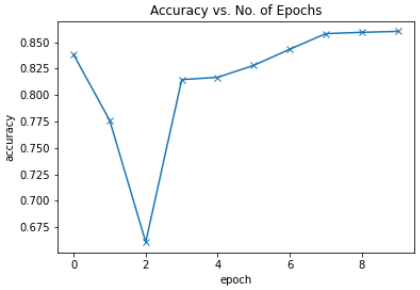


(b) CNN Accuracy à batch$_size$128

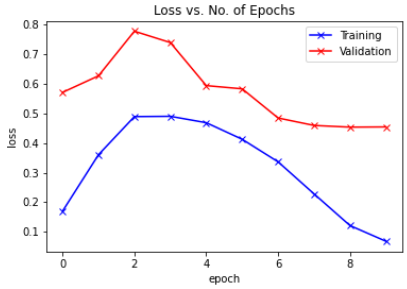
FIGURE 4 – Entraînement Modèle CNN



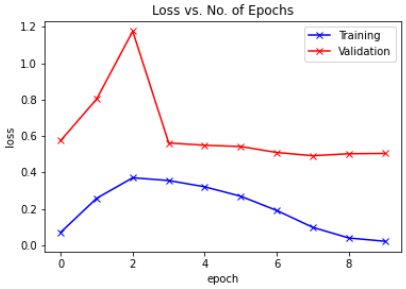
(a) ResNet Accuracy à batch$_size$64



(b) ResNet Accuracy à batch$_size$128



(c) ResNet Loss à batch$_size$64



(d) ResNet Loss à batch$_size$64

FIGURE 5 – Entraînement Modèle ResNet

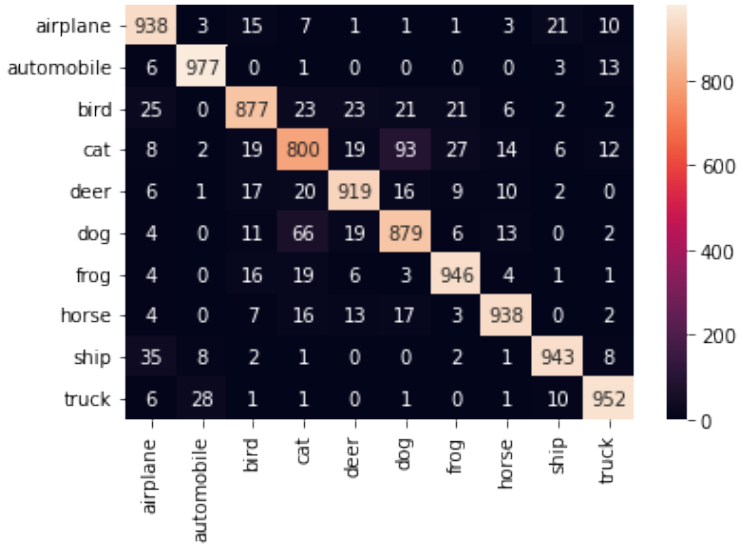


FIGURE 6 – Matrice de confusion

4 Transfert Learning

4.1 Fixed Feature Extrator

Pour raffiner les résultats de notre modèle, avec l'enregistrement, on a essaye de appliquer un méthode de transfert learning sur notre modèle précédemment entraîné, pour cela nous avons suivi une procédure pour freiner l'apprentissage des couches en conservant les poids sur ces couches intactes, et en donnant au modèle un couche finale avec Linear avec des poids aléatoires. cependant au moment d'appliquer cette procédure les résultats n'étaient pas ceux attendus, la précision de notre modèle est descendu de manière drastique, ne pouvant pas réussir à raffiner notre modèle.