Rapport Intelligence Artificielle

VO Frédéric

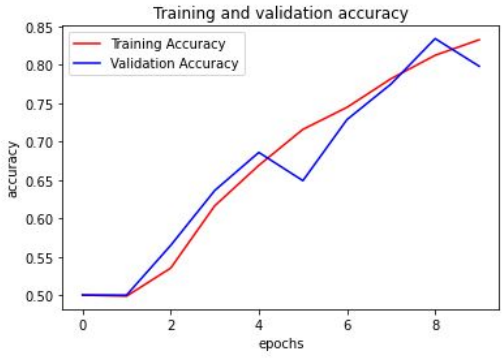
17 octobre 2022

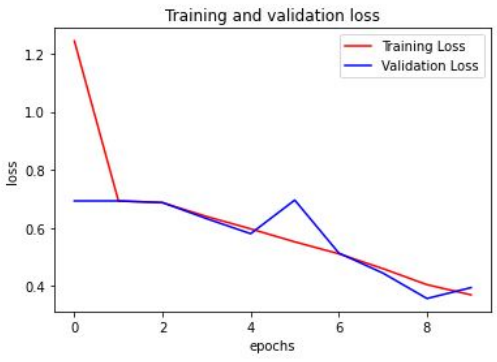
1 Modification des paramètres

Dans ce tp, nous allons prendre un code sur https://www.kaggle.com/ permettant de détecter les images de chiens et chats afin d'y changer les paramètres. Le code que j'ai choisi vient de https://www.kaggle.com/code/benyaminghahremani/dog-vs-cat-classification-cnn. Le but est de trouver les meilleurs paramètres afin d'avoir le meilleur pourcentage de validation.

1.1 Optimizer

Tout d'abord, j'ai décidé de changer le paramètre **optimizer**. L'optimizer est un algorithme qui va permettre de diminuer le loss et donc, d'avoir de meilleurs prédictions. Nous allons tester avec **Adam**, un optimizer très souvent utilisé.





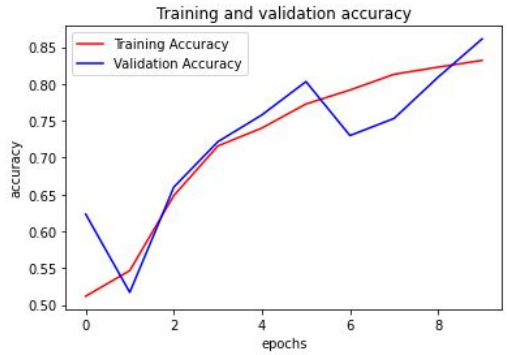
Les différents paramètres sont :

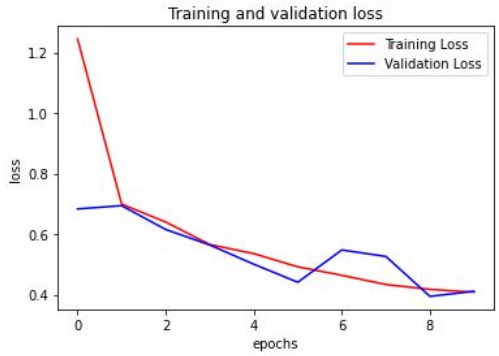
1. Epoch(s) : **10** ;
2. Optimizer : **Adam**
3. Learning rate : **0.0005**
4. Nombre de layers : **31**

La **validation et training accuracy** commence tout deux à **0.5**, correspondant à un pourcentage de **50%**. En effet, au départ, le model a seulement une chance sur deux de reconnaître un chien ou un chat puisque qu'il n'a subit aucun entraînement. Ensuite, nous pouvons voir une évolution allant jusqu'à **0.85** environ pour le training / validation accuracy.

Concernant le **loss**, on peut voir qu'il diminue aussi au fil des epochs. Comparé à l'accuracy, le loss n'est pas représenté par un pourcentage.

Nous allons comparer ces graphes avec d'autres qui ont utilisé un différent optimizer, **RMSprop**.





Les différents paramètres sont :

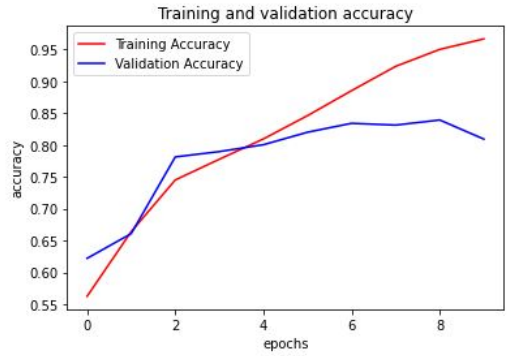
1. Epoch(s) : **10**
2. Optimizer : **RMSprop**
3. Learning rate : **0.0005**
4. Nombre de layers : **31**

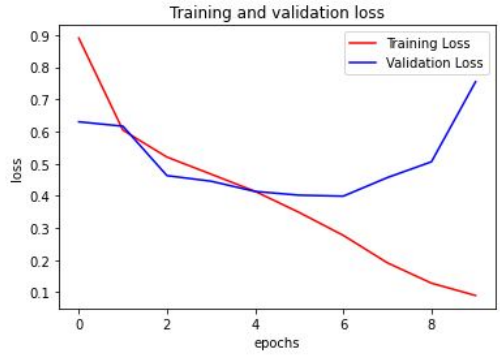
La **validation et trading accuracy** au bout de **10** epochs correspondent à peu près à celles du model utilisant **Adam**. Cependant, nous pouvons remarquer qu'au bout de **5 epochs**, le graphe avec RMSprop atteint **0.8** contre **0.65** pour Adam. Le training et validation accuracy avec RMSprop augmente vite au départ mais varie après.

Le graphe du **loss** avec **RMSprop** a que très peu de différence avec celle de **Adam**.

1.2 Layers

Par rapport au code de base, j'ai décidé de réduire le nombre de layers pour voir la différence avec le nombre de base. Au départ, on avait **31** layers, et en enlevant **14** layers, on obtient ceci :



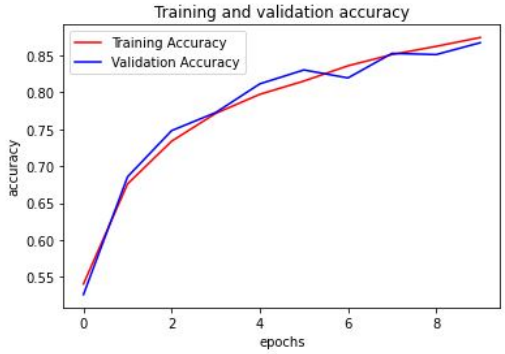


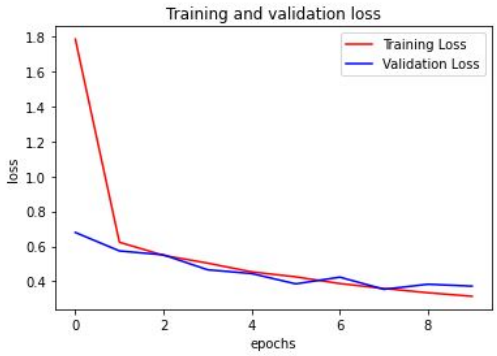
Les différents paramètres sont :

1. Epoch(s) : **10**
2. Optimizer : **Adam**
3. Learning rate : **0.0005**
4. Nombre de layers : **17**

Nous remarquons ici que la training accuracy augmente au long des epochs, alors que la validation accuracy augmente rapidement à **0.80** avant de stagner. A la fin, la courbe a une tendance régressive.

Pareillement avec le graphe de loss, le training loss baisse tandis que la validation loss stagne et augmente vers la fin. Réduire le nombre de layers rend les résultats moins précis.





Les différents paramètres sont :

1. Epoch(s) : **10**
2. Optimizer : **RMSprop**
3. Learning rate : **0.0005**
4. Nombre de layers : **17**

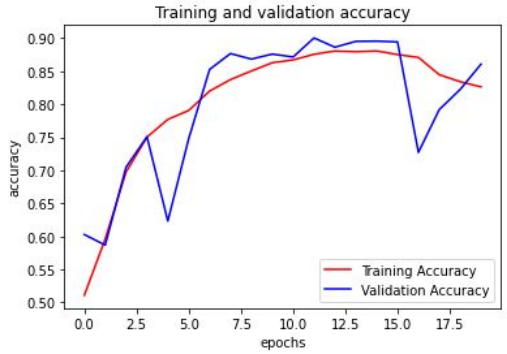
La validation et training accuracy augmentent de manière très similaire.

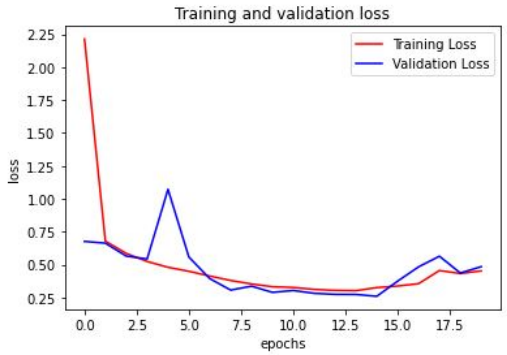
La validation et training loss baissent également de manière similaire

RMSprop a tendance a être plus précis et a avoir moins de loss que Adam quand il s'agit d'avoir très peu de layers dans le model.

1.3 Layers et Optimizer

Cette fois-çi, j'ai décidé d'augmenter un peu le nombre de layers et de changer de nombre d'epochs. Je garde RMSprop pour ce test.





Les différents paramètres sont :

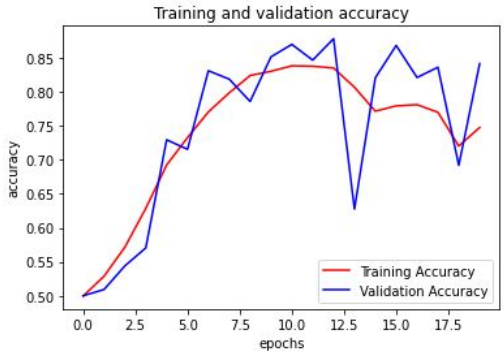
1. Epoch(s) : **20**
2. Optimizer : **RMSprop**
3. Learning rate : **0.0005**
4. Nombre de layers : **24**

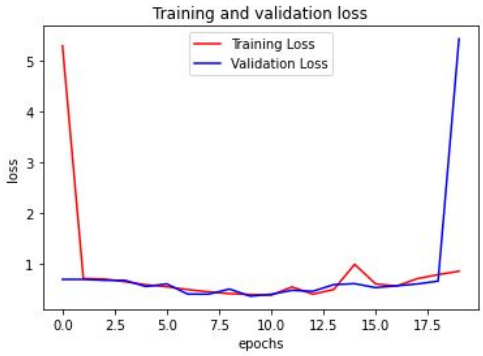
La validation accuracy fluctue énormément comparé à la training accuracy qui augmentent puis a une tendance régressive vers l'epoch **15**.

La validation loss fluctue aussi et le training loss diminue avant d'avoir une tendance progressive à la fin.

1.4 Epochs

Enfin, je décide de mettre le nombre d'epochs à **20** et de laisser les autres paramètres de base. Je vais tester avec **RMSprop** et **Adam**.



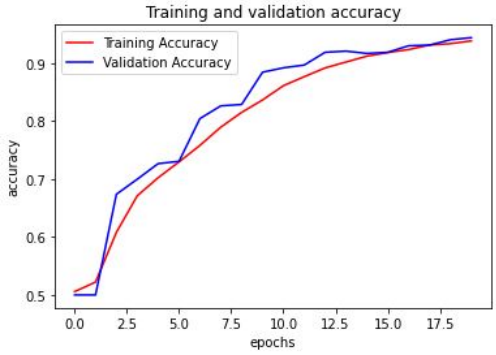


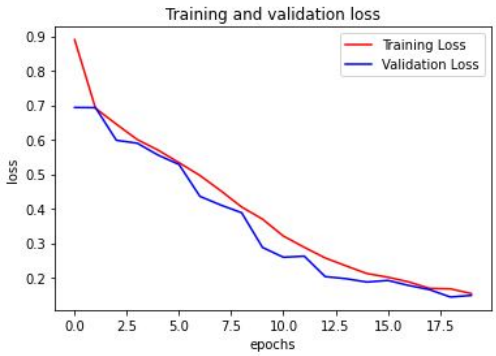
Les différents paramètres sont :

1. Epoch(s) : **20**
2. Optimizer : **RMSprop**
3. Learning rate : **0.0005**
4. Nombre de layers : **31**

On voit très bien que les courbes de training et validation accuracy fluctuent énormément (provenant peut-être d'overfitting).

La validation et training loss semblent stagner jusqu'à la fin où la validation loss augmentent énormément.





Les différents paramètres sont :

1. Epoch(s) : **20**
2. Optimizer : **Adam**
3. Learning rate : **0.0005**
4. Nombre de layers : **31**

Cette fois-çi, nous remarquons que les courbes de training et validation accuracy augmentent jusqu'à **0.94** et stagnent dans les alentours. Le courbe de validation accuracy reste au dessus de celle de training accuracy tout au long des epochs.

De même pour le loss, la courbe de validation loss reste en dessous de celle de training loss. Les deux courbes baissent au long des **20** epochs.

Pour conclure ce TP, on voit donc que l'optimizers **Adam**, ainsi qu'un nombre d'epochs d'environ **15**, permettent d'obtenir un accuracy dans les alentours de **94%** et des loss inférieurs à **0.1**.