TP Intelligence artificielle : Dogs vs Cats

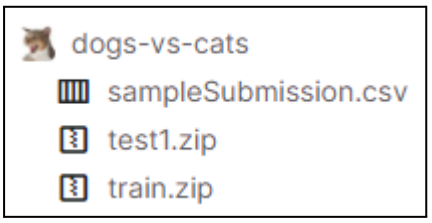
Louis MINGUET, M2 Informatique

GitHub address :
https://github.com/LouisMinguet/Cats-Dogs-Training

Dataset :
https://www.kaggle.com/code/benyaminghahremani/dog-vs-cat-classification-cnn

Instructions :
Using the dataset and the code examples on Kaggle :
Modify parameters such as: learning rate, regularization, optimizers, number of layers in CNN, activation function, epochs, dataset for training, validation and test.
Produce Cost and Accuracy graphs and the confusion matrix on the test data for the set of parameters that work best.
***Do not** use Transfer Learning (VGG etc) but use Conv2D functions.*

Le dataset utilisé pour ces entraînements est celui de "Dogs Vs Cats" :



Il y a **25000 images d'entraînements**, dont :
- 12500 images de chiens
- 12500 images de chats

Il y a **12500 images pour les tests**.

Les algorithmes d'optimisation utilisés sont Adam et RMSprop. Adams est un algorithme utilisé pour la formation de modèles d'apprentissage profond. Il permet de mettre à jour les poids de réseau itératifs en fonction des données d'apprentissage.
Ces avantages sont qu'il est simple à utiliser, efficace pour les calculs et il demande peu de mémoire.
RMSprop est un algorithme d'optimisation qui accélère la descente de gradient

Durant ces tests, je vais faire varier plusieurs paramètres, notamment l'**Epoch**, qui aura une influence sur les résultats mais surtout sur la durée des tests. Plus l'Epoch est grand, plus c'est long, mais les résultats seront plus précis avec un Epoch plus grand.

Learning rate : Ce paramètre de réglage détermine la taille du pas à chaque itération tout en se déplaçant vers un minimum d'une fonction de perte.

CNN : Convolutional Neural Networks : Réseau de neurones.

Résultats obtenus :

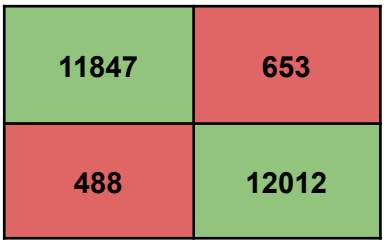
D'après les tests réalisés dans la partie suivante, on trouve que les paramètres suivant influent sur le comportement des tests :

- **Optimizer :** Adam et RMSprop :
- **Number of layers in CNN :** 31 et 15. Lorsqu'on met moins de layer, on remarque que de très bons résultats arrivent plus rapidement. On monte rapidement à des 0,7 / 0,8 d'accuracy, mais que ces résultats stagnent avec le temps, contrairement .
- **Epoch :** Plus il y a d'epoch, plus les résultats sont précis, ce qui est logique, car il y a plus d'entraînement, donc de meilleurs résultats. D'après les tests que j'ai pu effectuer avec 10-15-25 et 50 epoch, on voit que l'accuracy se rapproche de plus en plus vers 1, passant respectivement de de 0.8 à 0.9, puis de 0.9388 à 0.95 avec 50 epochs. Les tests étant vraiment très longs avec 50 epoch, j'ai réduit à 10 epoch pour les tests, mais je me doute que les résultats seront bien plus précis en utilisant 50 epoch.
- **Learning rate :** Lors des tests, j'ai pu apercevoir qu'augmenter cette valeur ne permettait pas d'obtenir de meilleurs résultats, au contraire, cela diminue le taux de réussite.

Pour obtenir de meilleurs résultats, les principaux éléments à régler sont le learning rate et les epoch. Mettre de meilleurs paramètres augmentent l'accuracy, mais **augmentent aussi considérablement le temps d'exécution**, pouvant passer de **25 minutes** à **plusieurs heures**.

Pour conclure, on voit bien ici que **l'objectif est de trouver les paramètres les plus optimisés** afin que **l'exécution ne soit pas trop lente**, tout en obtenant des **résultats cohérents**.
Ici, j'ai vu qu'un des éléments principaux est **l'epoch**. Ce paramètre **change beaucoup les résultats** obtenus lors des tests, ce qui est cohérent. On pourrait donc se dire qu'il suffit de mettre une fois un très grand nombre d'epoch pour obtenir un résultat très proche de 1. Or, les différents tests ont pu montrer qu'une fois arrivé à un résultat convainquant (environ 0.93, 0.94), **les résultats commencent à stagner**, et il devient donc inutile de tester avec plus d'epoch, car on risque de rester sur des résultats similaires malgré la très longue durée d'exécution. C'est bon à partir de ces moments là que les autres paramètres, comme les **layers** ou le **learning** rate peuvent devenir des atouts intéressants.

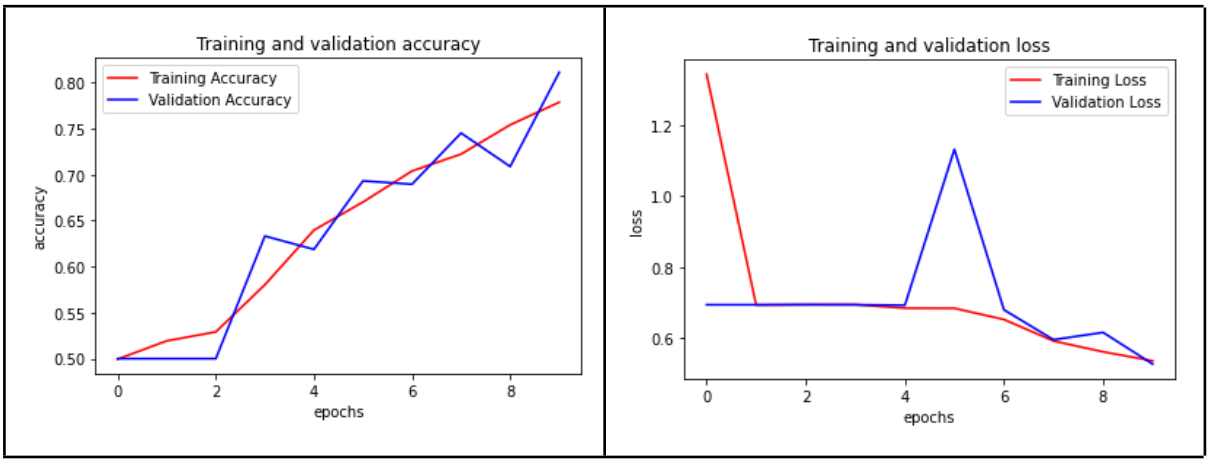
J'ai essayé dans le code d'afficher la matrice de confusion, mais je n'ai pas réussi à obtenir un résultat cohérent. Néanmoins, si je devais obtenir une matrice, je pense qu'elle pourrait ressembler à cela :



Tests réalisés :

10 Epoch

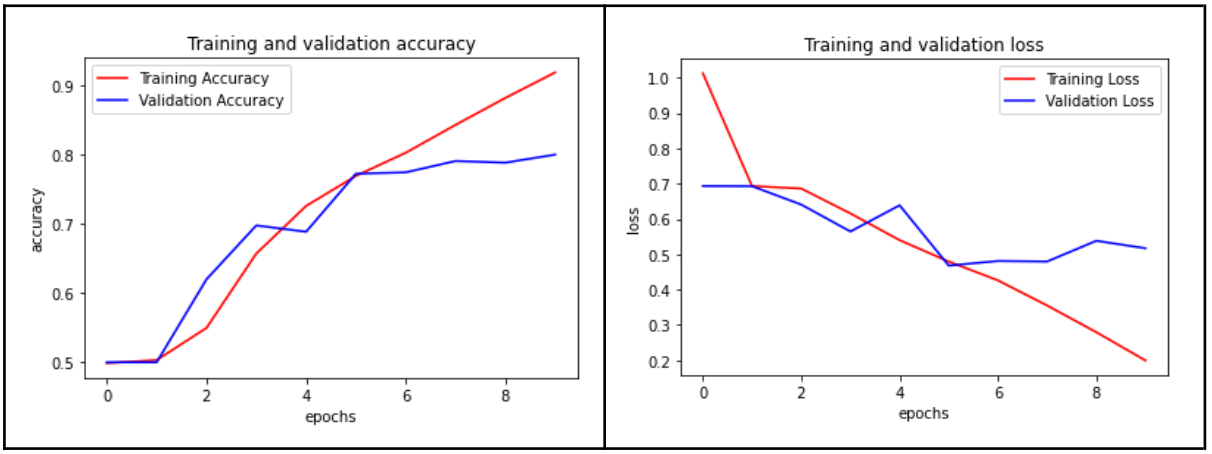
1) Test avec les paramètres par défaut :
   - ➢ Learning rate : **0.0005**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **10**




```
loss: 0.5749 - acc: 0.7760 - val_loss: 0.5608 - val_acc: 0.8136
```

2) Paramètres utilisés :
   - ➢ Learning rate : **0.0005**
   - ➢ Number of layers in CNN : **17**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **10**

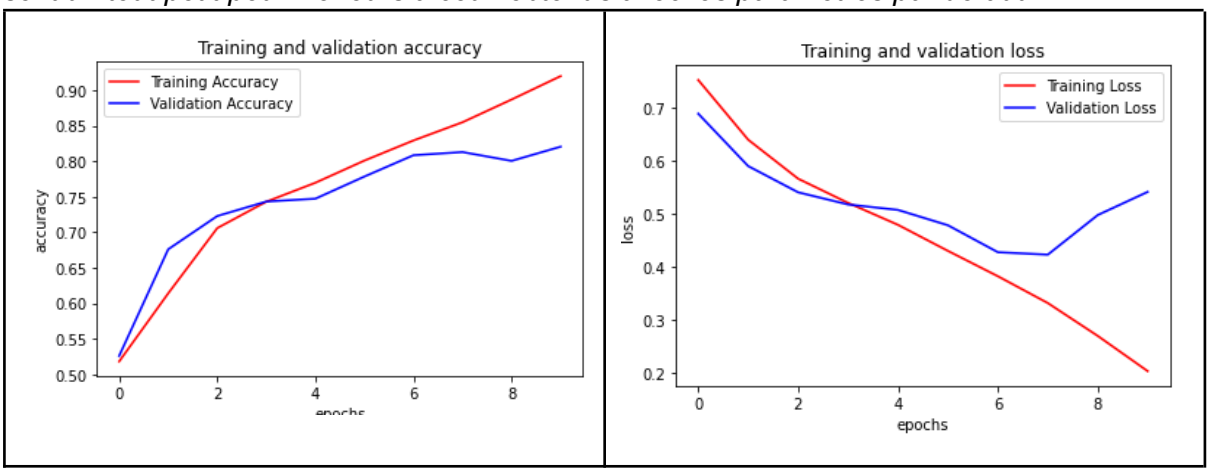
En réduisant les layers du CNN, on obtient de meilleurs résultats durant l'entraînement, mais de moins bons résultats lors de la validation, bien qu'ils soient plus stables.




```
loss: 0.1893 - acc: 0.9259 - val_loss: 0.5171 - val_acc: 0.8004
```

3) Paramètres utilisés :
   - ➢ Learning rate : **0.00005**
   - ➢ Number of layers in CNN : **32**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **10**

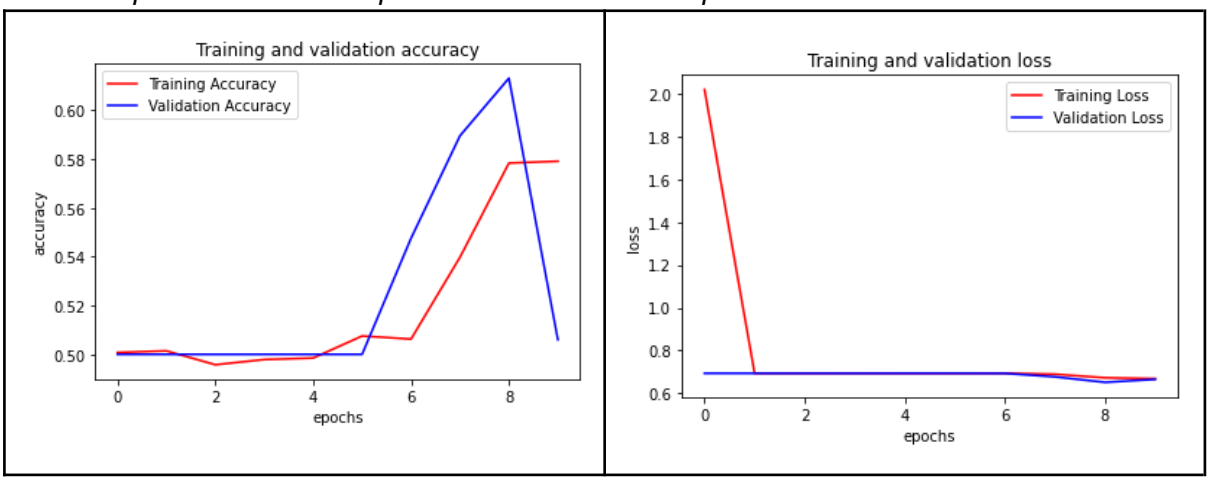
En réduisant le learning rate et les layers du CNN, on obtient de très bons résultats durant l'entraînement. Ceux obtenus durant la validation restent corrects, mais un peu inférieurs. Ils sont un tout petit peu inférieurs à ceux obtenus avec les paramètres par défaut.




```
loss: 0.2095 - acc: 0.9283 - val_loss: 0.5628 - val_acc: 0.9140
```

4) Paramètres utilisés :
   - ➢ Learning rate : **0.001**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **10**

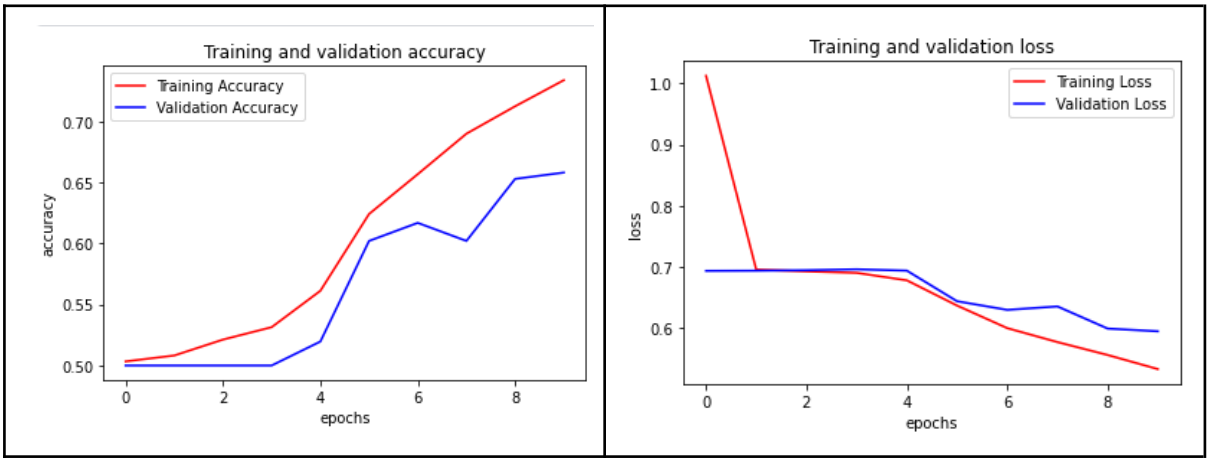
En augmentant le learning rate, on voit les résultats chuter drastiquement. Ce n'est donc sûrement pas dans ce sens qu'il faut faire évoluer les paramètres.




```
loss: 0.7021 - acc: 0.5802 - val_loss: 0.7034 - val_acc: 0.5152
```

5) Paramètres utilisés :
   - ➢ Learning rate : **0.0001**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **10**

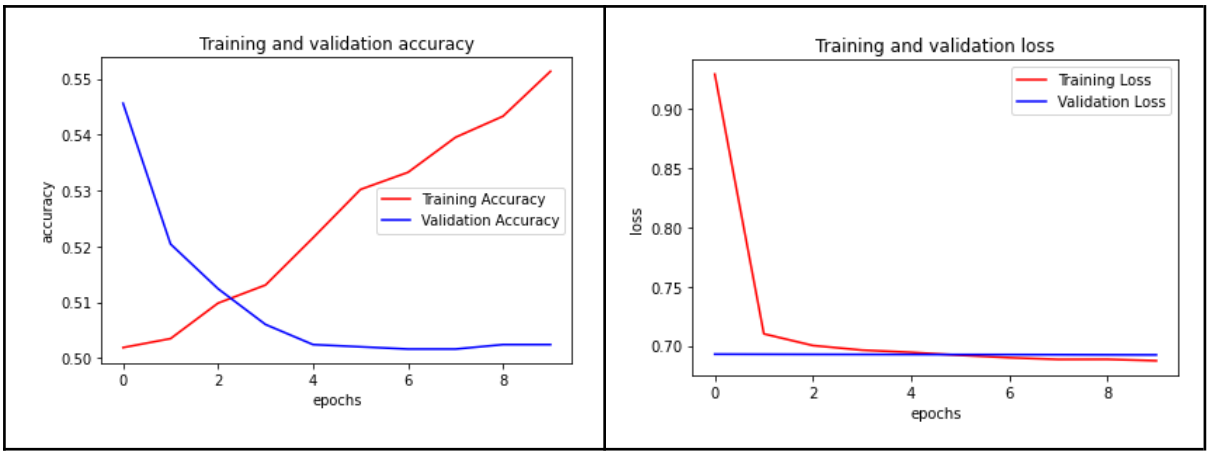
Lors de ces tests, j'ai réduit le learning rate dans un but d'améliorer les résultats, mais ceux-là n'ont pas étés très convaincants.




```
loss: 0.5230 - acc: 0.7542 - val_loss: 0.6014 - val_acc: 0.6591
```

6) Paramètres utilisés :
   - ➢ Learning rate : **0.0005**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **SGD**
   - ➢ epoch : **10**

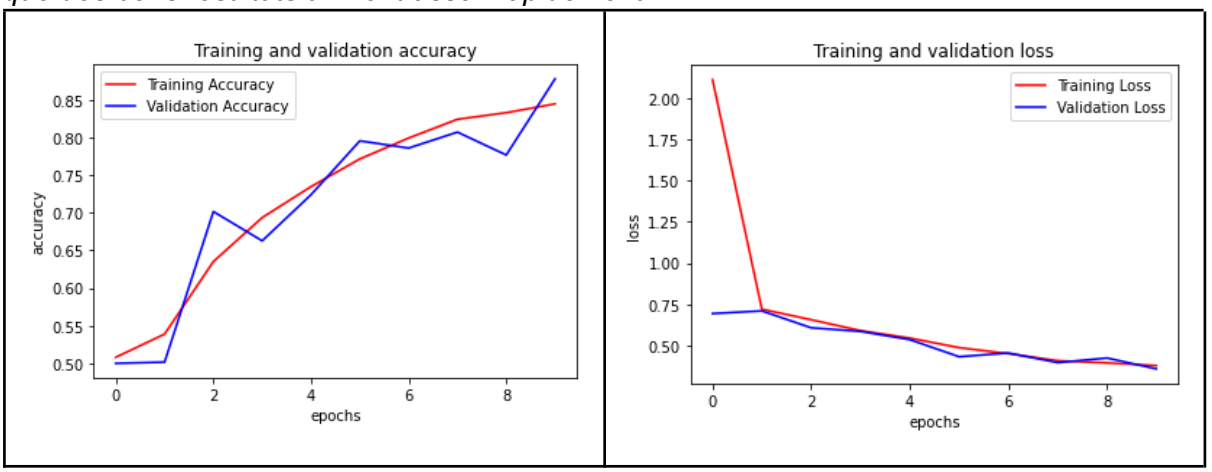
J'ai testé l'optimizer SQG, il m'a donné des résultats moyens lors des tests, et très mauvais lors de la validation.




```
loss: 0.6959 - acc: 0.5517 - val_loss: 0.6974 - val_acc: 0.5015
```

7) Paramètres utilisés :
   - ➢ Learning rate : **0.0005**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **RMSprop**
   - ➢ epoch : **10**

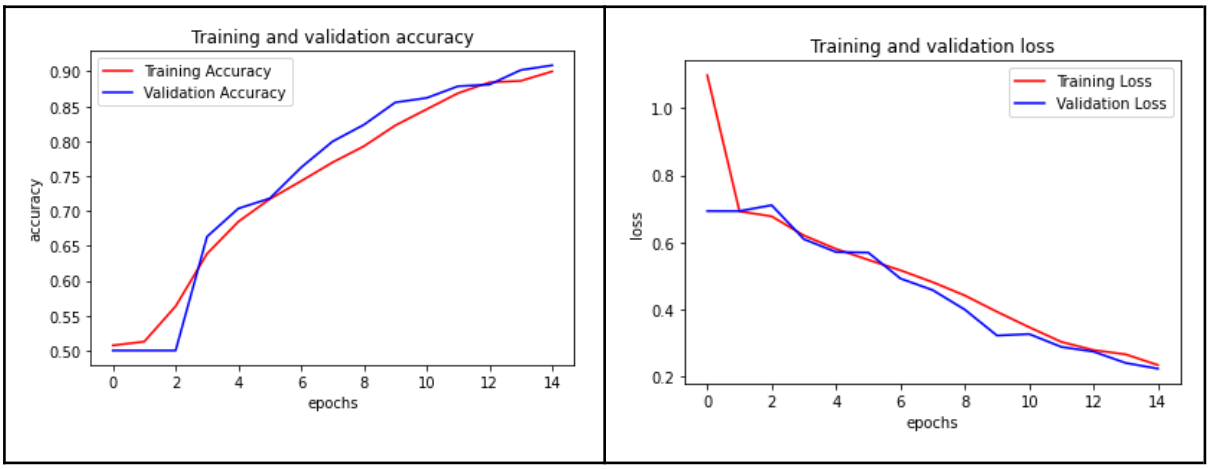
J'ai aussi testé l'optimizer RMSprop, il m'a donné de très bons résultats lors des tests et de la validation. Ils sont presque égaux à ceux obtenus avec Adam. Avec cet optimizer, on voit que des bons résultats arrivent assez rapidement.




```
loss: 0.4582 - acc: 0.8758 - val_loss: 0.4571 - val_acc: 0.8460
```

15 Epoch

1) Paramètres utilisés :
   - ➢ Learning rate : **0.0005**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **15**
   - ➢ csv généré : 15epoch_normal.csv

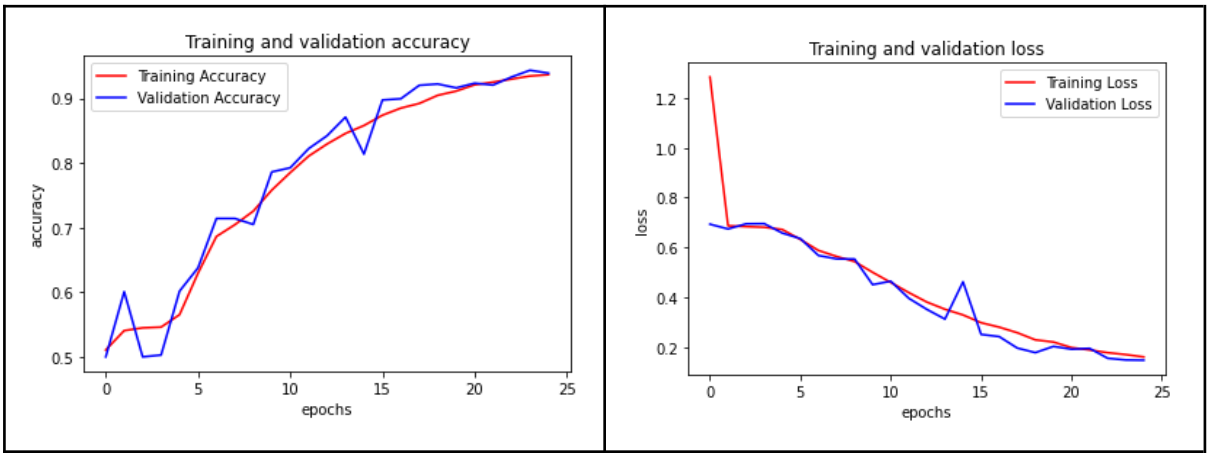



```
loss: 0.2134 - acc: 0.9003 - val_loss: 0.2159 - val_acc: 0.8982
```

25 Epoch

1) Paramètres utilisés :
   - ➢ Learning rate : **0.0005**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **25**

Lorsqu'on effectue les tests avec 25 epoch, on se rend rapidement compte que les résultats obtenus sont beaucoup plus proches de 1 qu'avec 10 et 15 epoch, ce qui parait normal. J'ai donc réalisé un test avec les paramètres les plus adaptés et j'ai trouvé de très bons résultats (0.94 de validation accuracy pour 0.2 de pertes).

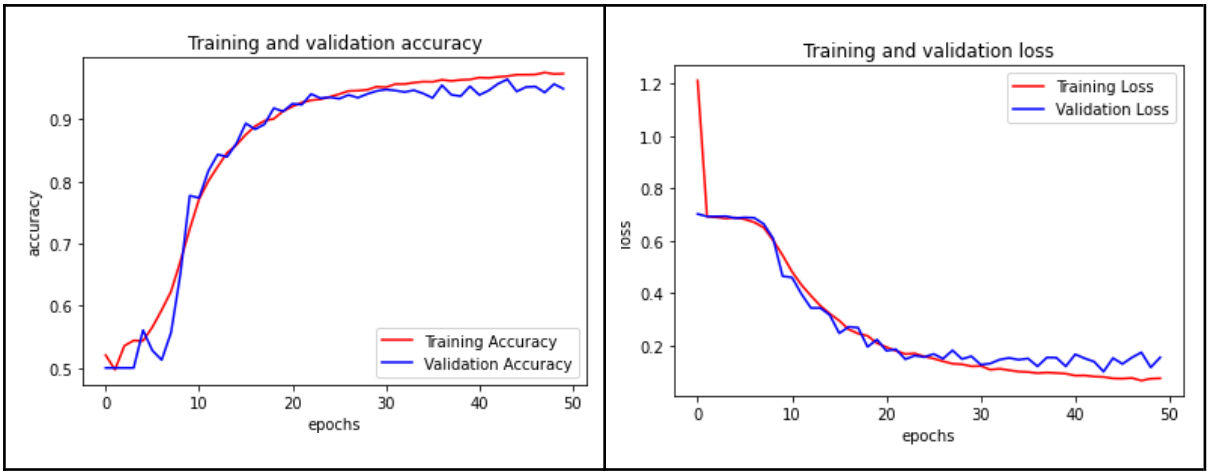



```
loss: 0.1902 - acc: 0.09427 - val_loss: 0.1958 - val_acc: 0.09429
```

50 Epoch

1) Paramètres utilisés :
   - ➢ Learning rate : **0.0005**
   - ➢ Number of layers in CNN : **31**
   - ➢ Optimizer : **Adam**
   - ➢ epoch : **50**

J'ai pu observer qu'avec 25 epoch on obtient de bien meilleurs résultats qu'avec 10 ou 15. Je me suis donc demandé si les résultats ne faisaient que croître en augmentant encore les epoch. En passant à 50 epoch, on se rend compte que plus le nombre d'epoch est grand, plus le résultat va stagner. Il ne suffit pas de mettre 2000 epoch pour obtenir un résultat encore plus proche de 1. Avec 25 epoch, nous obtenions un validation accuracy d'environ 0.94, et avec 50 epoch, nous avons obtenu globalement le même résultat : 0.9492.
(Le graphique et les résultats sont en dessous)




```
loss: 0.0734 - acc: 0.9744 - val_loss: 0.1553 - val_acc: 0.9492
```