M2 Logiciel sûrs - IA

Cats & Dogs classification

Réalisé par :
Merzeg Ramzi
Khitous Rania

Lien git vers le code :https://github.com/ramzimerzeg/cats_and_dogs

On a utilisé le dataset disponible sur kaggle : https://www.kaggle.com/competitions/dogs-vs-cats/data

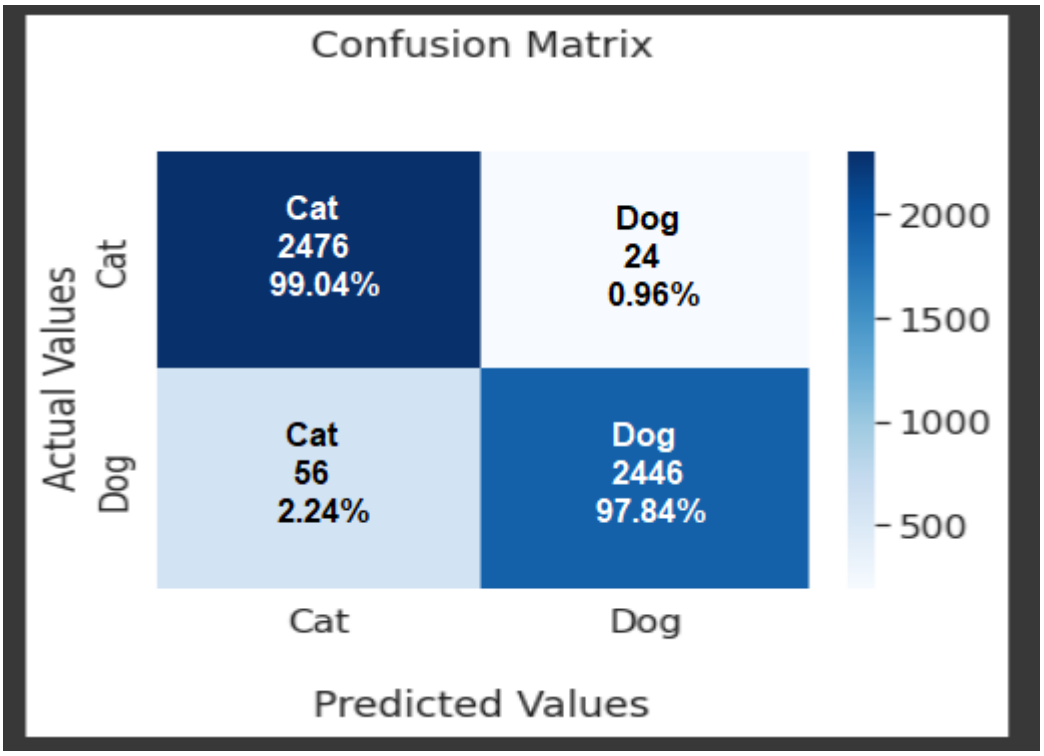
Meilleur essaie :

- Learning rate : varie entre 0.01 et 0.00001 automatiquement selon l'adaptation du modèle
- optimizers : **OPT**
- number of layers in CNN : conv2D(32 + 64 + 64 + 128 + 128) + Dense (64 + 32 + 2)
- Fonction de loss : **categorical_crossentropy**
- Batch size **: 32**
- activation function : **relu + sigmoid**
- epochs : 50
- validation split : 0.33

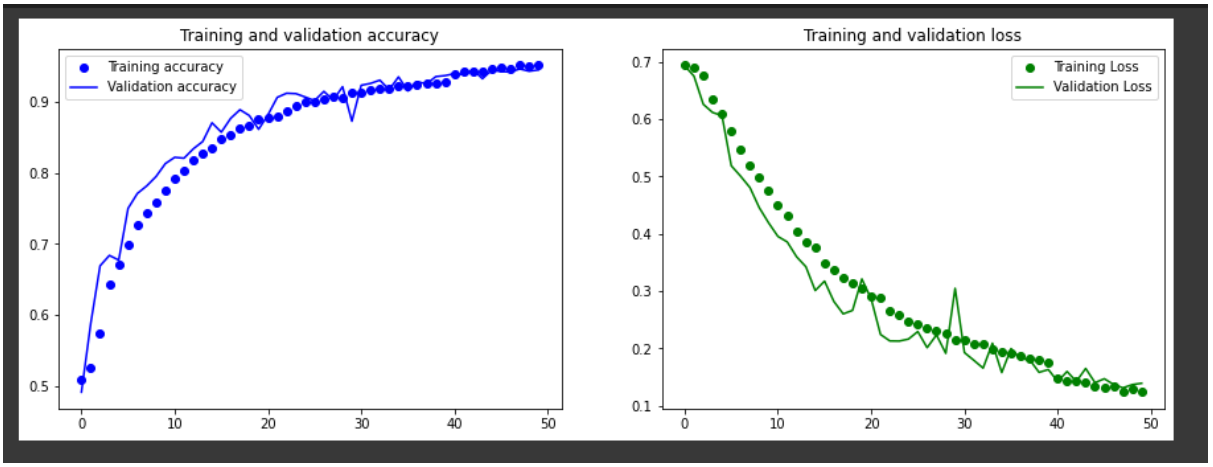
Résultat :

acc : 95%
loss : 12%
validation accuracy : 94 %
validation loss : 12%
avec un learning rate qui commence à 0.01 et qui descend jusqu'à 0.0025

Matrice de Confusion :



Graphe de training et validation d'accuracy et de loss :



Evaluation du modèle avec les donnée de test :

Essaie 1 :

- Learning rate : 0.01
- optimizers : **OPT**
- number of layers in CNN : conv2D(32 + 64 + 128) + Dense (32 + 1)
- Fonction de loss : **binary_crossentropy**
- Batch size **: 10**
- activation function : **tanh + softmax**
- epochs : 20
- validation split : 0.33

Résultat :

acc : 69%
loss : 54%
validation accuracy : 59 %
validation loss : 60%

Essaie 2 :

- Learning rate : 0.001
- optimizers : **Adam**
- number of layers in CNN : conv2D(32 + 64 + 128) + Dense (32 + 1)
- Fonction de loss : **binary_crossentropy**
- Batch size **: 32**
- activation function : **tanh + sigmoid**
- epochs : 50
- validation split : 0.25

Résultat :

acc : 70%
loss : 40%
validation accuracy : 62 %
validation loss : 51%

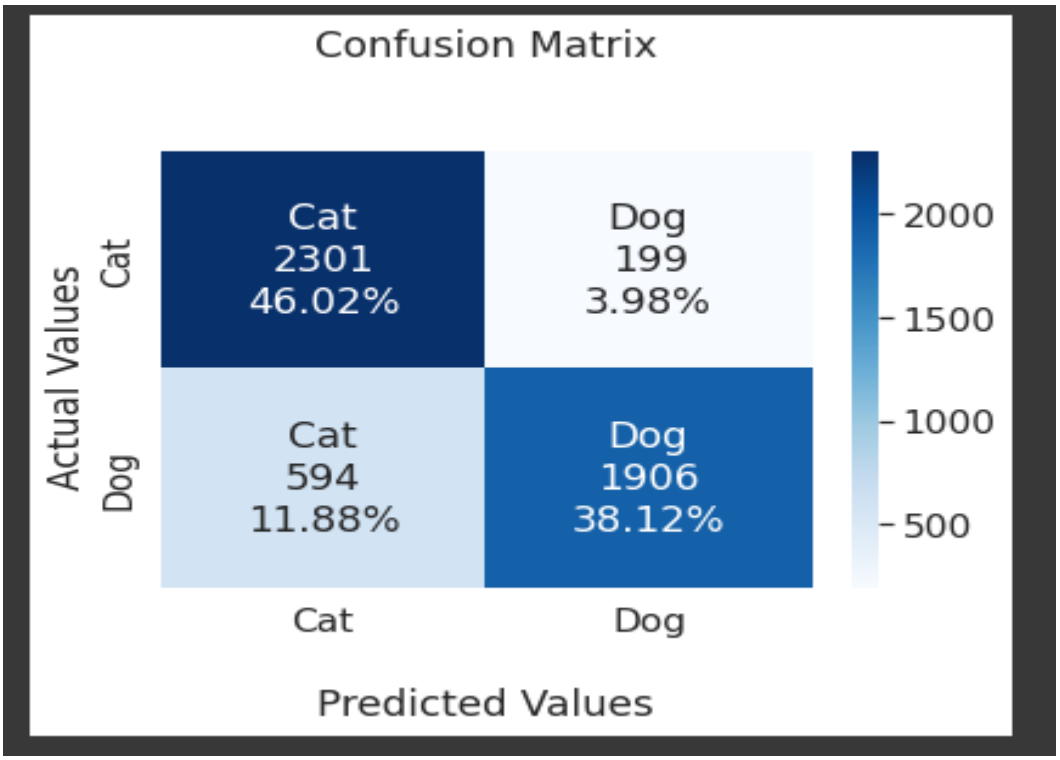
Essaie 3 :

- Learning rate : 0.001
- optimizers : **Adam**
- number of layers in CNN : conv2D(256 + 256 + 128 + 128 + 64) + Dense (64 + 1)
- Fonction de loss : **binary_crossentropy**
- Batch size **: 32**
- activation function : **relu + sigmoid**
- epochs : 7
- validation split : 0.1

Résultat :

acc : 88%
loss : 26%
validation accuracy : 85 %
validation loss : 35%

Matrice de confusion :



Essaie 4 :

- Learning rate : 0.01
- optimizers : **Adam**
- number of layers in CNN : conv2D(256 + 256 + 128 + 128 + 64) + Dense (64 + 1)
- Fonction de loss : **binary_crossentropy**
- Batch size **: 22**
- activation function : **relu + sigmoid**
- epochs : 30
- validation split : 0.25

Résultat :

acc : 85%
loss : 28%
validation accuracy : 82 %
validation loss : 38%

Essaie 5 :

- Learning rate : 0.001
- optimizers : **Adam**
- number of layers in CNN : conv2D(400 + 300+ 200+ 100+ 50) + Dense (10+ 1)
- Fonction de loss : **binary_crossentropy**
- Batch size **: 100**
- activation function : **relu + sigmoid**
- epochs : 8
- validation split : 0.1

Résultat :

acc : 86%
loss : 34%
validation accuracy : 81 %
validation loss : 51 %

Essaie 6 :

- Learning rate : 0.01
- optimizers : **adam**
- number of layers in CNN : conv2D(32 + 64 + 64 + 128 + 128) + Dense (64 + 32 + 2)
- Fonction de loss : **binary_crossentropy**
- Batch size **: 22**
- activation function : **relu + sigmoid**
- epochs : 30
- validation split : 0.25

Résultat :

acc : 92%
loss : 16%
validation accuracy : 90 %
validation loss : 17%
avec un learning rate qui commence fixé à 0.01

Essaie 8 :

- Learning rate : 0.001
- optimizers : **adam**
- number of layers in CNN : conv2D(32 + 64 + 64 + 128 + 128) + Dense (64 + 32 + 2)
- Fonction de loss : **binary_crossentropy**
- Batch size **: 40**
- activation function : **relu + sigmoid**
- epochs : 40
- validation split : 0.2

Résultat :

acc : 93%
loss : 20%
validation accuracy : 90 %
validation loss : 24%
avec un learning rate qui commence fixé à 0.001