# Partie 1 : Reconnaissance d'Images avec les Réseaux de Neurones Convolutifs (CNN)

Quel a été le rôle des callbacks ModelCheckpoint et EarlyStopping ?

Les callbacks servent à exécuter des fonctions pendant le déroulement de l’entraînement. ModelCheckpoint sert à stocker le modèle sous le format *.keras* et EarlyStopping sert à stopper l’entraînement si la valeur passée en paramètre *monitor* n’évolue pas comme souhaité pendant *x* epochs (définies par le paramètre *patienc*e*)*.

Ont-ils été utiles pendant votre entraînement ?

ModelCheckpoint a bien créer le fichier souhaité, mais il n’a jamais été utilisé. EarlyStopping a, par contre, bien stoppé l’entraînement avant le nombre d’epochs définis initialement (Principalement avec l’entraînement du modèle par Transfer Learning).

Analysez les courbes d'apprentissage (perte et précision). Votre modèle présente-t-il des signes de sur-apprentissage ou de sous-apprentissage ?

Étant donné que

* La courbe d’accuracy (Donc basée sur le set de train) monte en permanence, alors que la courbe de val\_accuracy (Donc basée sur le set de validation, qui ne sert pas pour l’entraînement) stagne à partir d’un certain point.
* L’écart entre la courbe d’accuracy et la courbe de val\_accuracy augmente au fil des epochs.
* Et inversement pour les courbes loss et val\_loss

On peut en déduire qu’on est en présence d’un cas de début d’overfitting qui a été stoppé par le callback EarlyStopping (Vu que val\_loss stagne).

Comment la matrice de confusion vous aide t-elle à comprendre les erreurs du modèle ?

La matrice de confusion represente les valeurs réelles et les valeurs prédites par le modele. Chaque cellule indique le nombre d’exemples où la valeur prédite et la valeur réelle corresponde a la combinaison (Par exemple le nombre de valeurs ayant comme prédiction « Avion » et comme valeur réelle « Chat »). Plus les valeurs situées dans la diagonale (Donc les valeurs correctes cas la valeur prédite est égale a la valeur réelle) sont élevées, et les autres faibles, plus le taux d’erreur est faible.

Quels ont été les principaux défis rencontrés? Comment pourriez-vous améliorer les performances de ce premier modèle ?

Hormis les questions liées au hardware (Durée de l’entrainement et ressources nécessaires), les principaux défis viennent de la faible précision du modèle créé (~62% d’accuracy avec l’entraînement basique et ~78% avec le Transfer Learning), et de l’overfitting.

Pour régler le problème de l’overfitting, une solution potentielle serait l’ajout d’une couche Dropout dans le modèle. Pour la précision, plusieurs solutions sont possibles :

* Augmenter la complexité du modèle (Nombre de couches et / ou de neurones par couche)
* Tester différents optimizers et learning rate

Comparez les performances (précision, temps d'entraînement) du modèle simple et du modèle basé sur le transfert d'apprentissage.

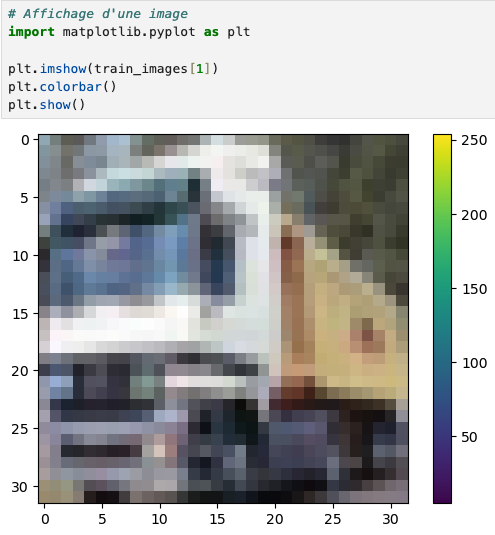
Le modèle basé sur le transfert d’apprentissage a été plus précis (~78% contre ~62% d’accuracy sur le set de test) tout en ayant un temps d’entraînement beaucoup plus long (22 secondes par epoch contre 8).

Pourquoi observe t-on une telle différence ?

Ces différences sont principalement liées au fait que MobileNetV2 est un modèle pré-entraîné, beaucoup plus complexe et entraîné sur un très grand dataset. L’entraînement effectué dans l’exercice ne concerne que la dernière couche de neurones, les autres couches étant gelées.

Captures d’ecran

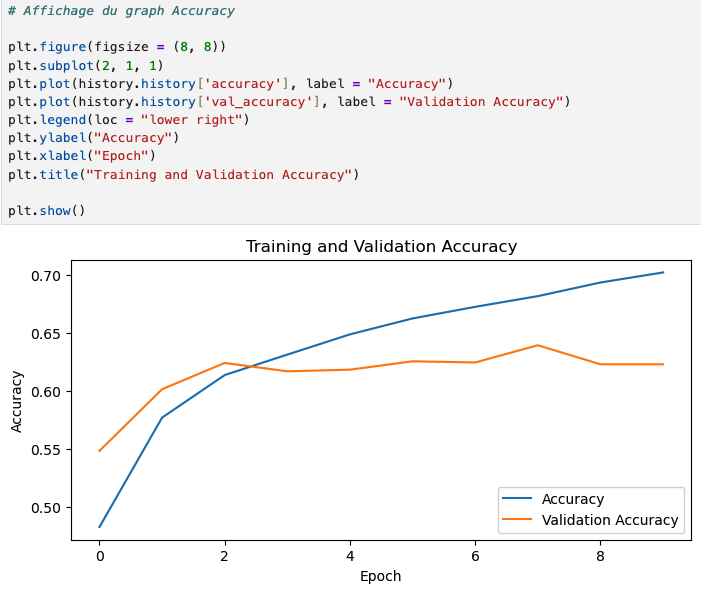
**Image du dataset :**



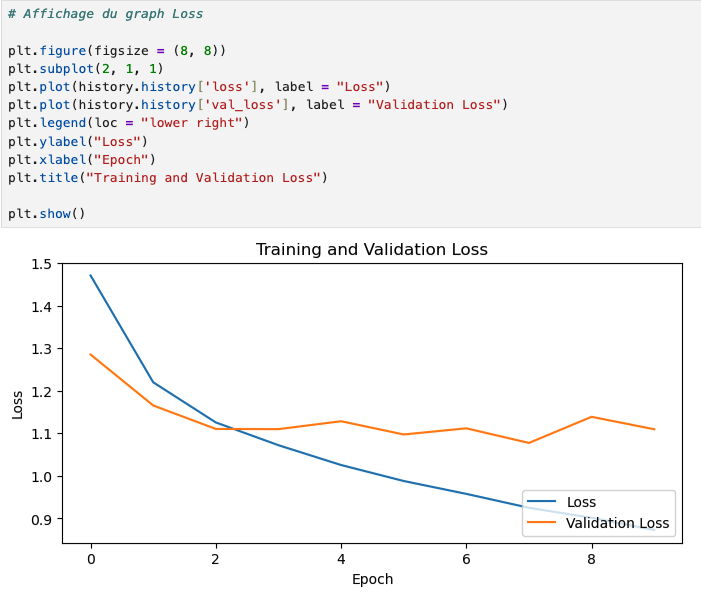
**Image de l’entrainement :**



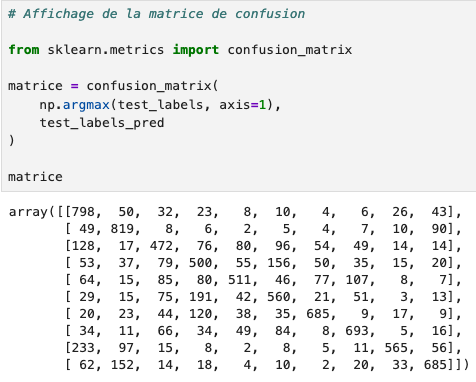
**Graph Accuracy**:



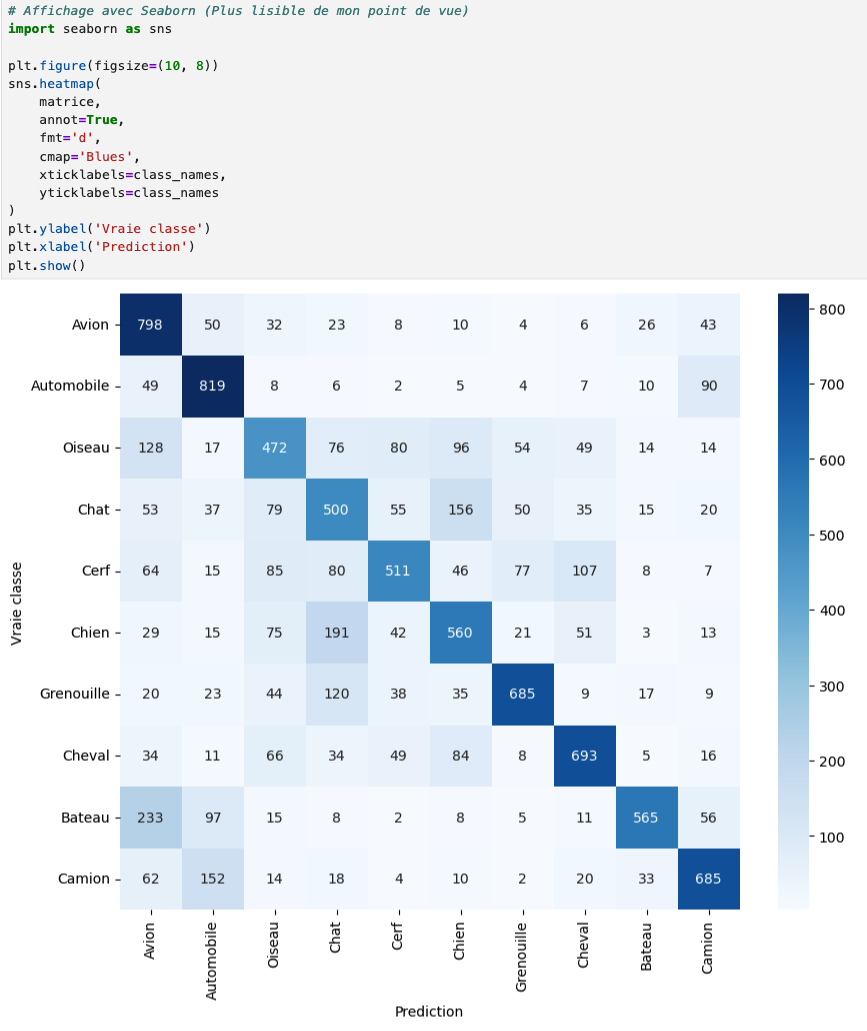
**Graph Loss** :



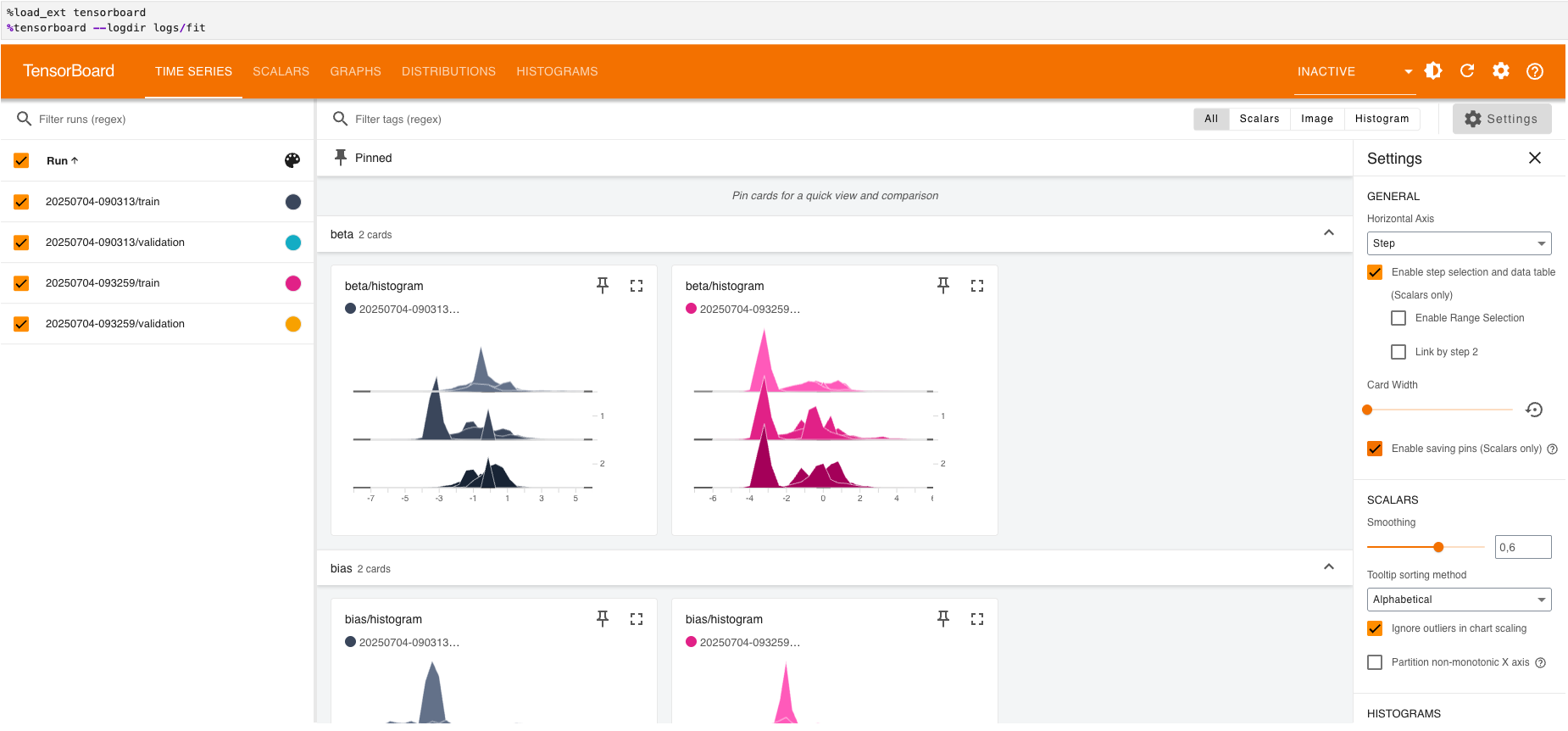
**Matrice de Confusion (Console) :**



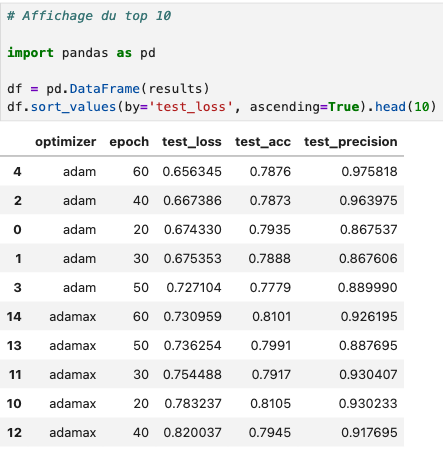
**Matrice de Confusion (Visuel) :**



**Tensorboard :**



**Tableau de Comparaison :**



**Comparaison des Optimizers :**

