OPENCLASSROOMS

PROJET N°7

Développez une preuve de concept (option stage)

Guillaume Lemêle - *Machine Learning Engineer* -10/2022 – 06/2023

SOMMAIRE

DÉVELOPPEZ UNE PREUVE DE CONCEPT (OPTION STAGE)

Etat de l’art

ÉTAPE N° 1

CONCEPTS CLÉS

Descente de gradient

Learning rate

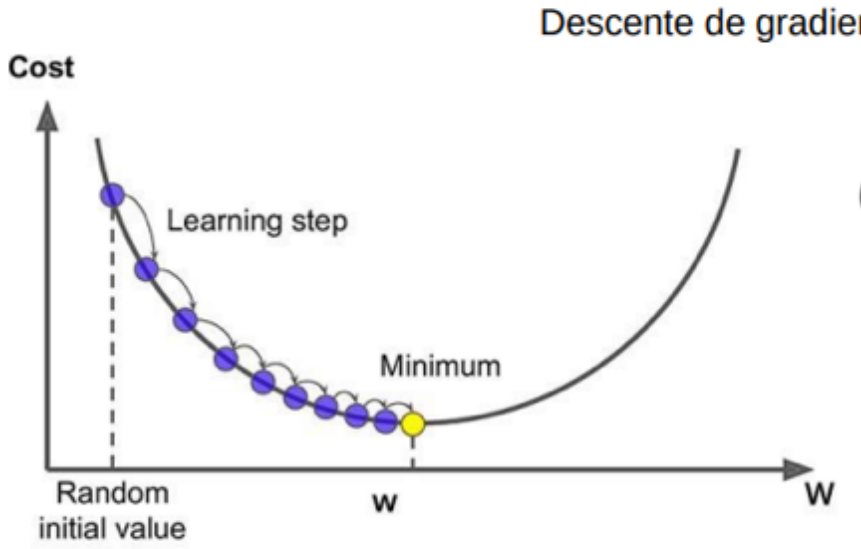
Descente de gradient stochastique

Learning Rate Scheduler

Momentum

Descente de gradien

Descente de gradient



$$\theta_i = \theta_{i-1} - \alpha * g$$

Learning rate

Gradient

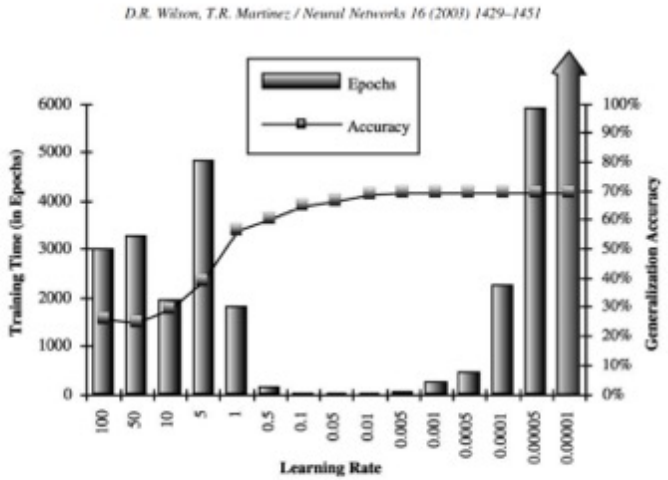
+ Facile à implémenter
- Lent
- Utilise beaucoup de mémoire

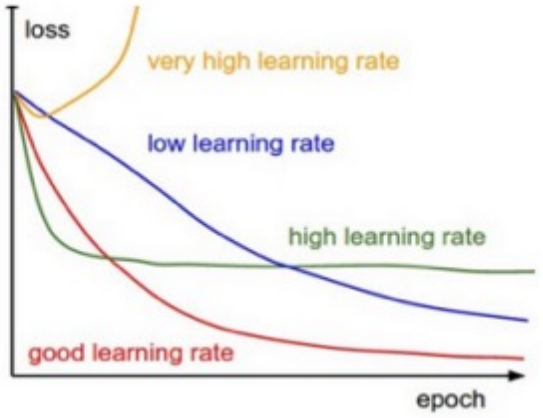
Calcule le gradient sur toutes les données avant la mise à jour

Learning rate

Learning Rate

Paramètre essentiel dans l'optimisation d'un modèle





Trop petit = Temps d'apprentissage trop long
Trop grand = Performances du modèle plus faibles

→ Besoin de varier le *learning rate* au cours de l'entraînement

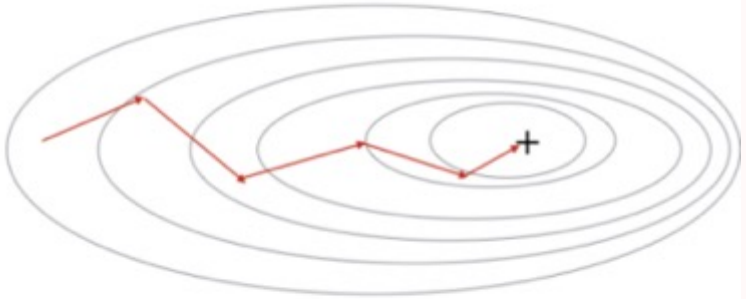
8

Descente de gradient stochastique

Descente de gradient stochastique

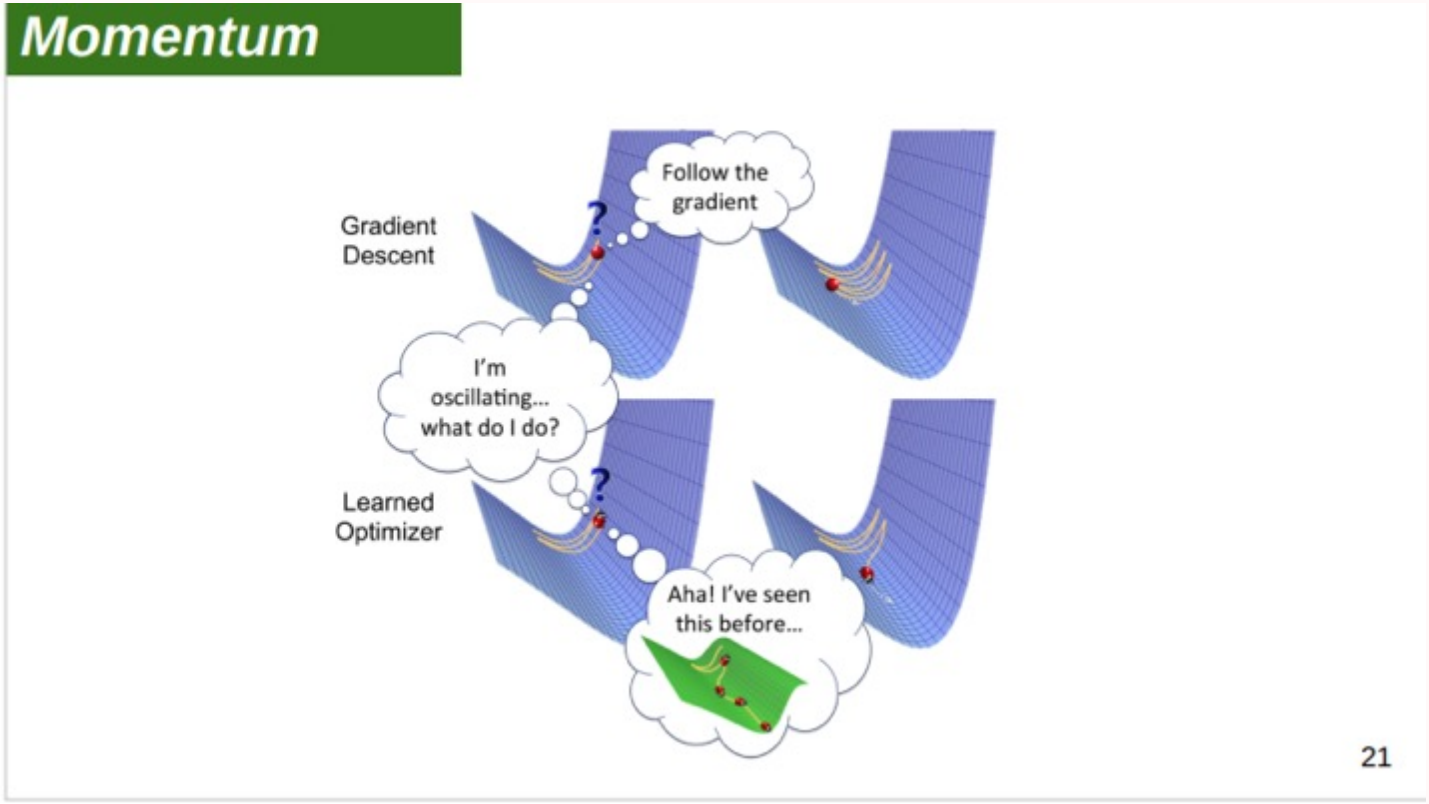
SGD = *Mini-batch Gradient Descent*

Similaire à la descente de gradient mais les poids sont mis à jour après chaque itération qui correspond à un petit groupe de données (mini batch)



+ Quantité raisonnable de mémoire nécessaire
+ Convergence rapide
+ Taille de *batch* adaptable selon les besoins
- Dépendant du *learning rate*

Momentum


Momentum
Gradient Descent
Follow the gradient
I'm oscillating... what do I do?
Learned Optimizer
Aha! I've seen this before...
21

ALGORITHMES D'OPTIMISATIONS

Adam

AdamW

Lion

Adam

Adam

Adam : Adaptative Moment estimation

$$m_i = \beta_1 * m_{i-1} + (1 - \beta_1) * g_i$$

Premier moment : moyenne glissante

$$v_i = \beta_2 * v_{i-1} + (1 - \beta_2) * g_i^2$$

Second moment : variance non centrée glissante

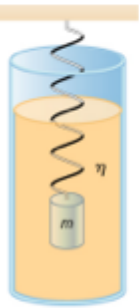
$$\hat{m}_i = \frac{m_i}{1 - \beta_1^i}$$

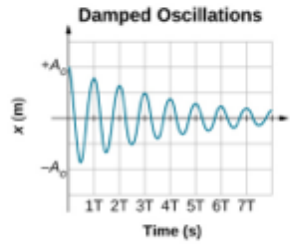
$$\hat{v}_i = \frac{v_i}{1 - \beta_2^i}$$

Correction des biais des premières itérations

$$\theta_i = \theta_{i-1} - \frac{\alpha}{\sqrt{\hat{v}_i + \epsilon}} * \hat{m}_i$$

Amortit les oscillations





Paramètres :

β_1 & β_2 = Taux de régression (β_1 = 0.9 & β_2 = 0.999)

ϵ — Très petite valeur pour éviter une division par zéro

But : Adapter l'importance des mise à jours de poids en fonction des précédents gradients et de la variabilité du gradient.

AdamW

Weight decay and decoupled weight decay

ADAM

For i = 1 to ...

$g_i = \nabla_\theta f_i(\theta_{i-1}) + \lambda\theta_{i-1}$

$m_i = \beta_1 * m_{i-1} + (1-\beta_1) * g_i$

$v_i = \beta_2 * v_{i-1} + (1-\beta_2) * g_i^2$

$\hat{m}_i = \frac{m_i}{1-\beta_1^i}$

$\hat{v}_i = \frac{v_i}{1-\beta_2^i}$

$\theta_i = \theta_{i-1} - \frac{\alpha}{\sqrt{\hat{v}_i + \epsilon}} * \hat{m}_i$

Return θ_i

Weight decay

ADAMW

For i = 1 to ...

$g_i = \nabla_\theta f_i(\theta_{i-1})$

$m_i = \beta_1 * m_{i-1} + (1-\beta_1) * g_i$

$v_i = \beta_2 * v_{i-1} + (1-\beta_2) * g_i^2$

$\hat{m}_i = \frac{m_i}{1-\beta_1^i}$

$\hat{v}_i = \frac{v_i}{1-\beta_2^i}$

$\theta_i = \theta_{i-1} - \frac{\alpha}{\sqrt{\hat{v}_i + \epsilon}} * \hat{m}_i - \alpha\lambda\theta_{i-1}$

Return θ_i

Decoupled weight decay

Évolution du *weight decay*: *Decoupled weight decay* (découplé du momentum !!)

- SGD et Adam avec le *weight decay*
- SGDW et AdamW avec le *decoupled weight decay*

SGD et SGDW sont à peu près équivalents en performance.

Cependant AdamW est notablement meilleur que Adam !!

Lion vs AdamW

A Pseudocode for AdamW and Lion

Algorithm 1 AdamW Optimizer

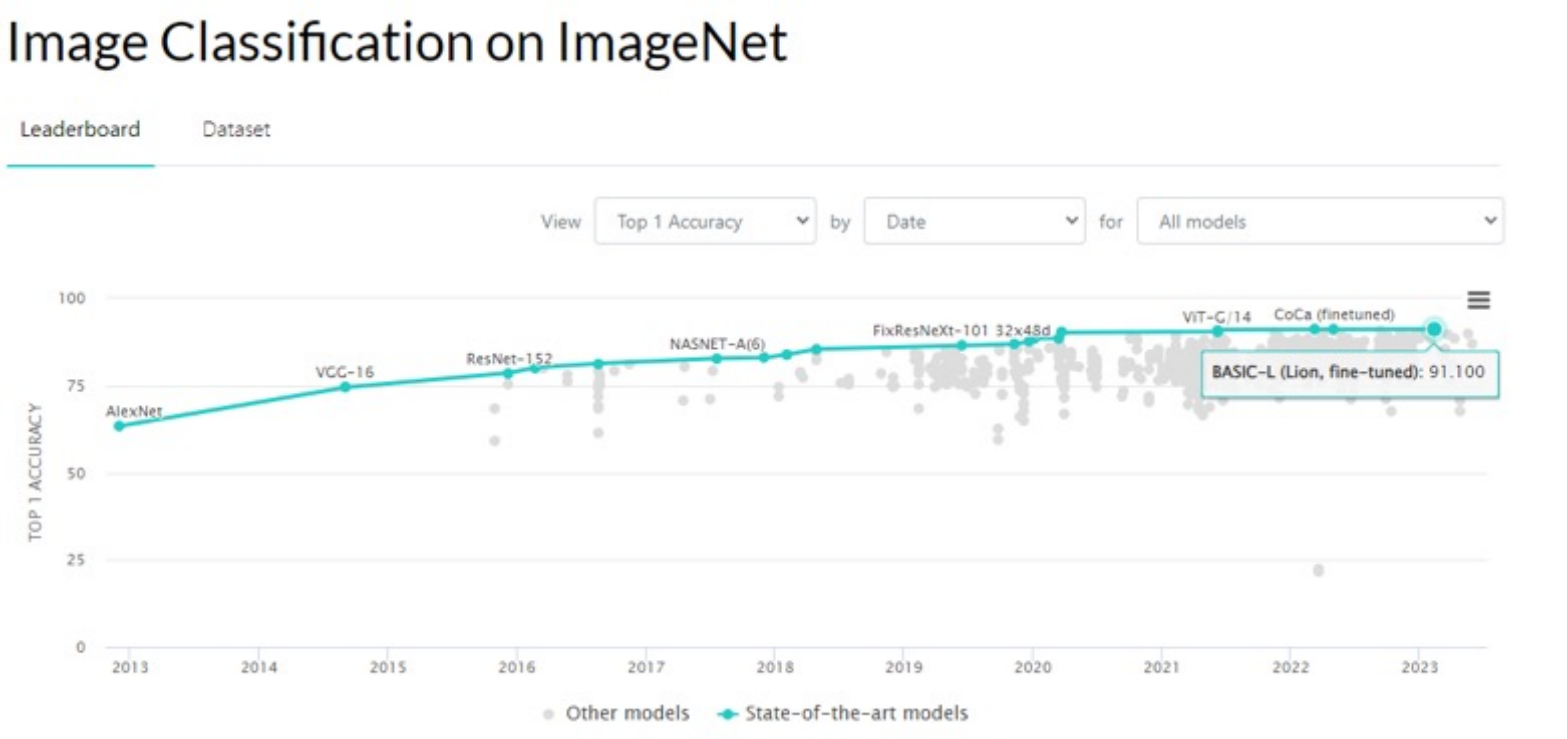
given $\beta_1, \beta_2, \epsilon, \lambda, \eta, f$
initialize $\theta_0, m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
  update EMA of g_t **and** g_t^2
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  bias correction
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
  update model parameters
  $\theta_t \leftarrow \theta_{t-1} - \eta_t (\hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + \lambda \theta_{t-1})$
end while
return θ_t

Algorithm 2 Lion Optimizer (ours)

given $\beta_1, \beta_2, \lambda, \eta, f$
initialize $\theta_0, m_0 \leftarrow 0$
while θ_t not converged **do**
  $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
  update model parameters
  $c_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $\theta_t \leftarrow \theta_{t-1} - \eta_t (\text{sign}(c_t) + \lambda \theta_{t-1})$
  update EMA of g_t
  $m_t \leftarrow \beta_2 m_{t-1} + (1 - \beta_2) g_t$
end while
return θ_t

Lion

Date : Mars 2023 < 18 mois



Prototype à implémenter

ÉTAPE N° 2

Processus

1. Séparation du jeu de données.
- Création d'un ensemble d'entraînement (train set) et d'un ensemble de validation et de test (validation et test set de 10%).

2. Division stratifiée des ensembles de validation et de test.
- Garder la même proportion des données selon la race de chien.

3. Utilisation d'ImageDataGenerator avec Keras pour crée un dataset d'images prétraités (pas de traitement d'images sur notre test set).

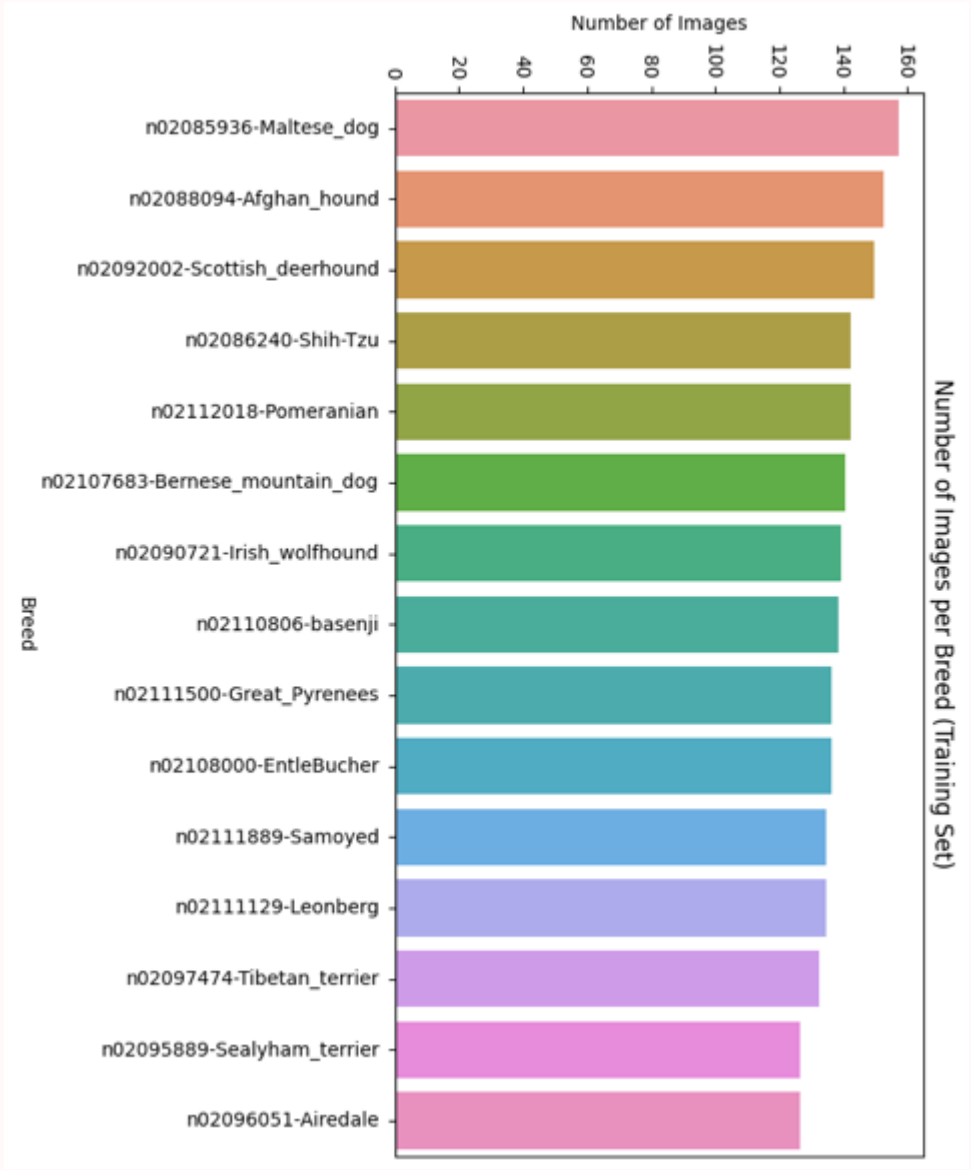
4. Hyperparameter tuning avec le modèle Xception.

5. Entrainement et prédiction avec l'algorithme baseline Adam puis AdamW et Lion – Evaluation et analyse des résultats.

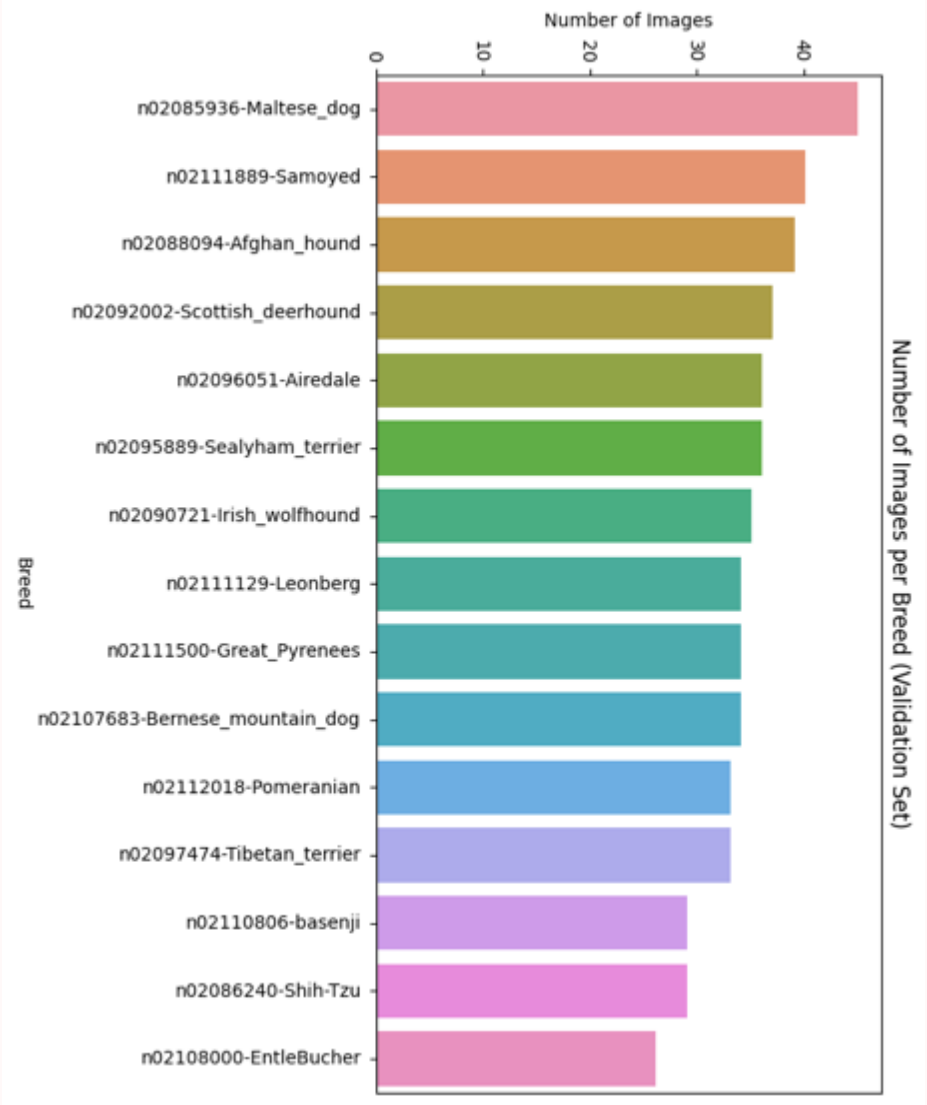
Modèle support : Xception

- Xception est un modèle de réseau neuronal convolutif profond introduit par François Chollet, le créateur de Keras. C’est une amélioration de l'architecture Inception, dont il tire son nom : "Extreme Inception" ou Xception.
- Transfert learning
- Features extraction

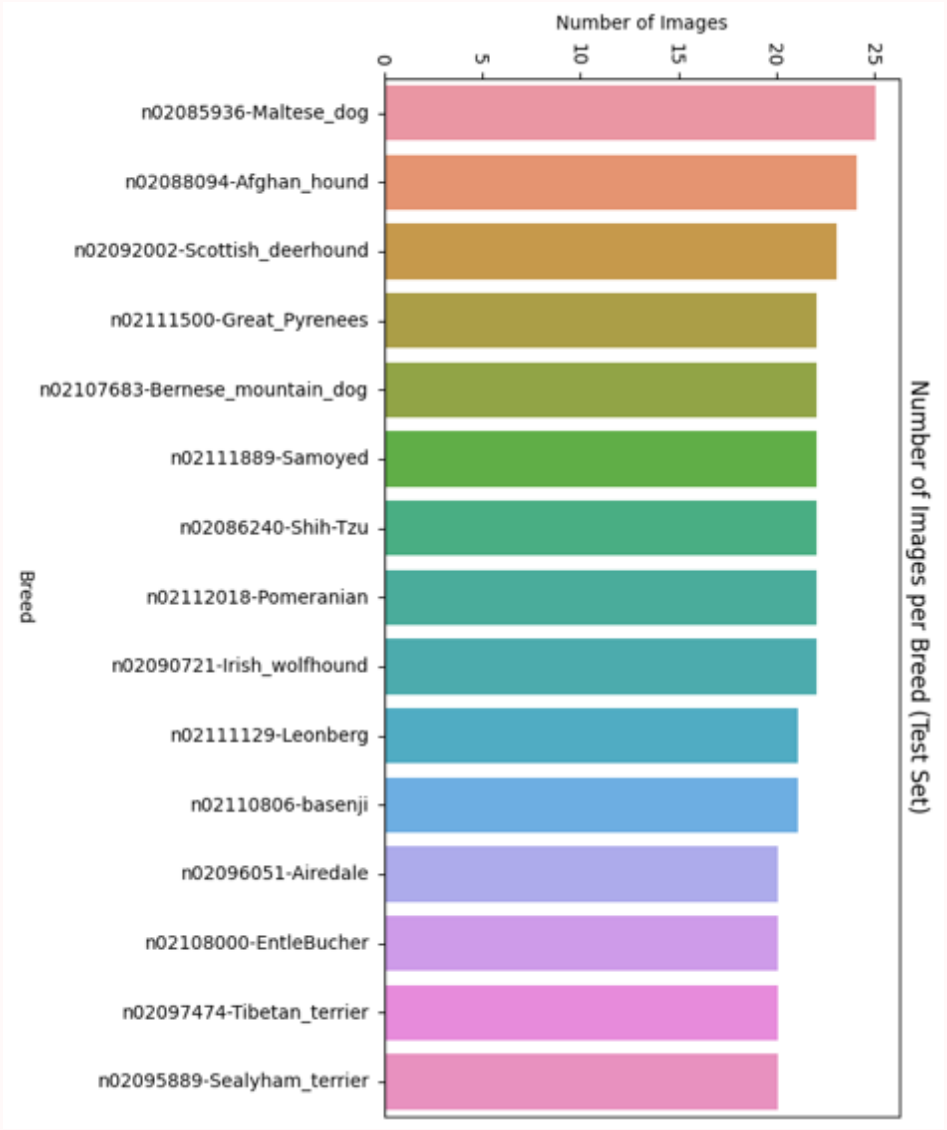
Dataset – train set



Dataset – validation set



Dataset – test set



Evaluation

1. **Accuracy**
2. **Precision**
3. **Recall**
4. **f1 score**

		Classe réelle	
		-	+
Classe prédite	-	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	+	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

- Présentation d'une matrice de classification
- Visualisation d'une matrice de confusion

$$Rappel = \frac{TP}{TP + FN}$$

$$Précision = \frac{TP}{TP + FP}$$

- F1 score : moyenne harmonique de la précision et du rappel

$$F - mesure = 2 \times \frac{Précision \times Rappel}{Précision + Rappel} = \frac{2TP}{2TP + FP + FN}$$

Analyse des résultats

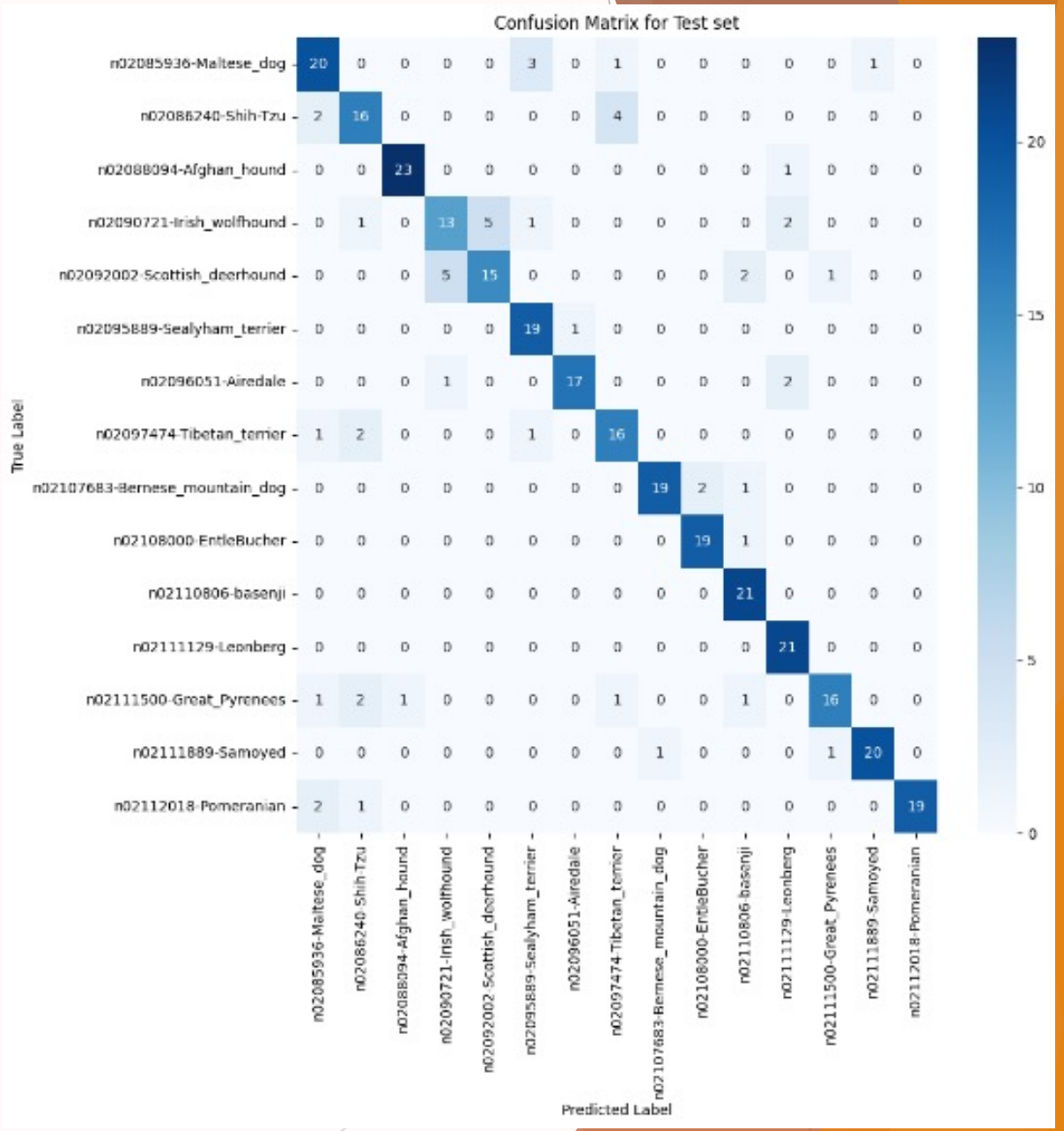
ÉTAPE N° 3

Baseline : Adam optimizer

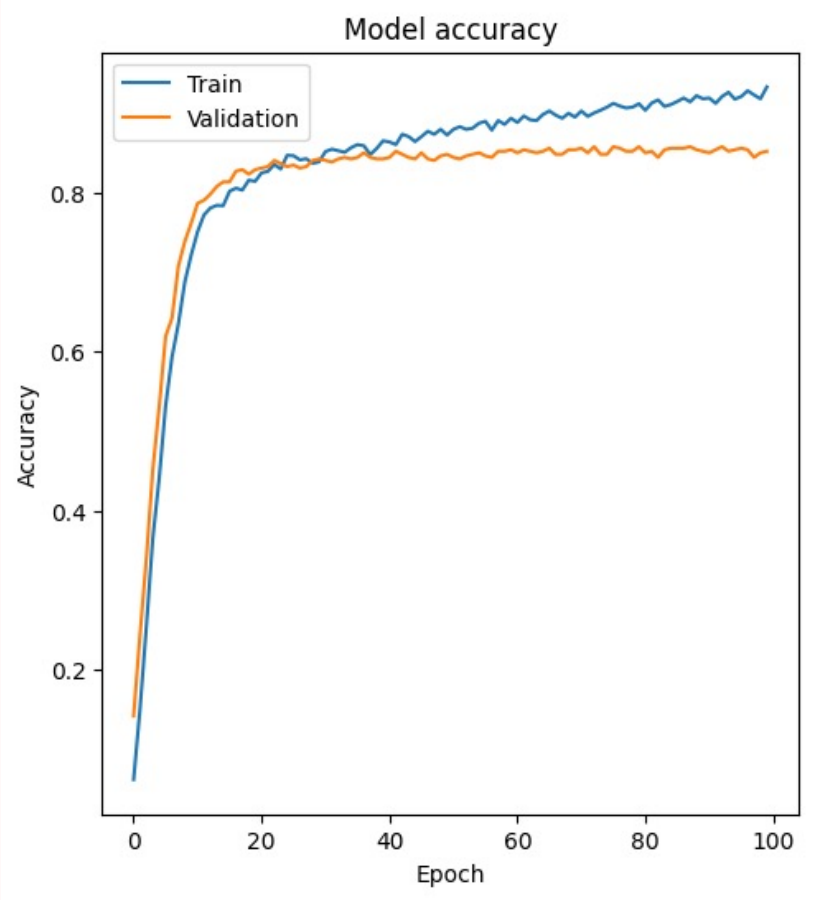
- **La performance du score d'accuracy globale est de 0.84.**
- **La précision parmi les 15 races de chiens oscille entre un minimum de 0.68 et un maximum de 1.**
- **Le rappel (recall) parmi les 15 races de chiens oscille entre un minimum de 0.65 et un maximum de 1.**
- **Le temps d'entraînement du modèle sur 100 épochs a été de 84 minutes.**
- **L'observation de l'évolution de la fonction de coût et du score d'exactitude montre un léger overfitting du jeu de données d'entraînement à partir de la 40ème époque.**

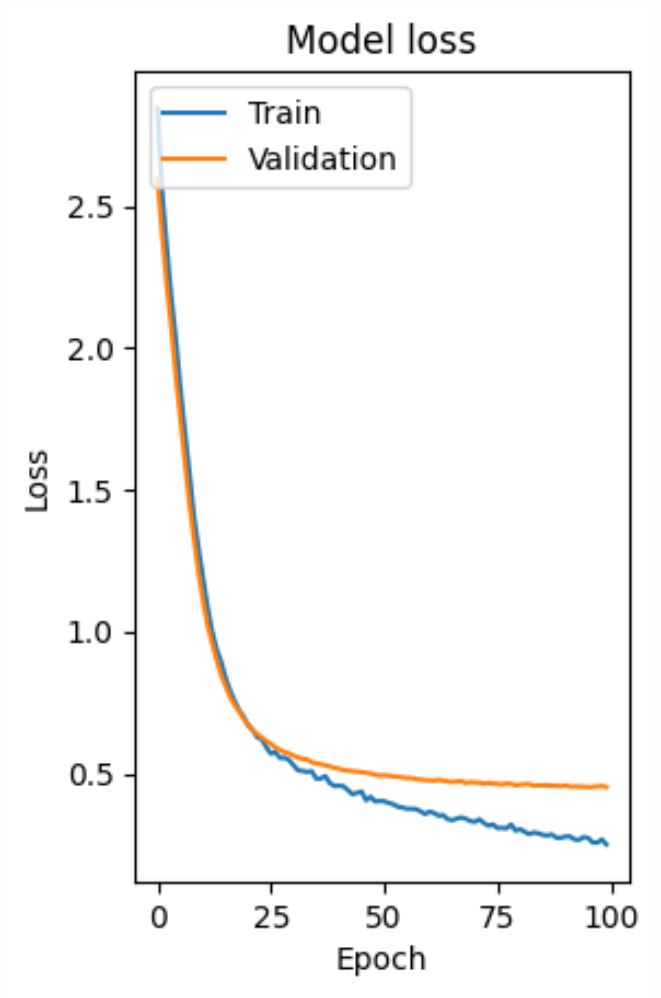
Baseline : Adam optimizer

	precision	recall	f1-score	support
n02085936-Maltese_dog	0.77	0.80	0.78	25
n02086240-Shih-Tzu	0.73	0.73	0.73	22
n02088094-Afghan_hound	0.96	0.96	0.96	24
n02090721-Irish_wolfhound	0.68	0.59	0.63	22
n02092002-Scottish_deerhound	0.75	0.65	0.70	23
n02095889-Sealyham_terrier	0.79	0.95	0.86	20
n02096051-Airedale	0.94	0.85	0.89	20
n02097474-Tibetan_terrier	0.73	0.80	0.76	20
n02107683-Bernese_mountain_dog	0.95	0.86	0.90	22
n02108000-EntleBucher	0.90	0.95	0.93	20
n02110806-basenji	0.81	1.00	0.89	21
n02111129-Leonberg	0.81	1.00	0.89	21
n02111500-Great_Pyrenees	0.89	0.73	0.80	22
n02111889-Samoyed	0.95	0.91	0.93	22
n02112018-Pomeranian	1.00	0.86	0.93	22
accuracy			0.84	326
macro avg	0.84	0.84	0.84	326
weighted avg	0.84	0.84	0.84	326



Baseline : Adam optimizer



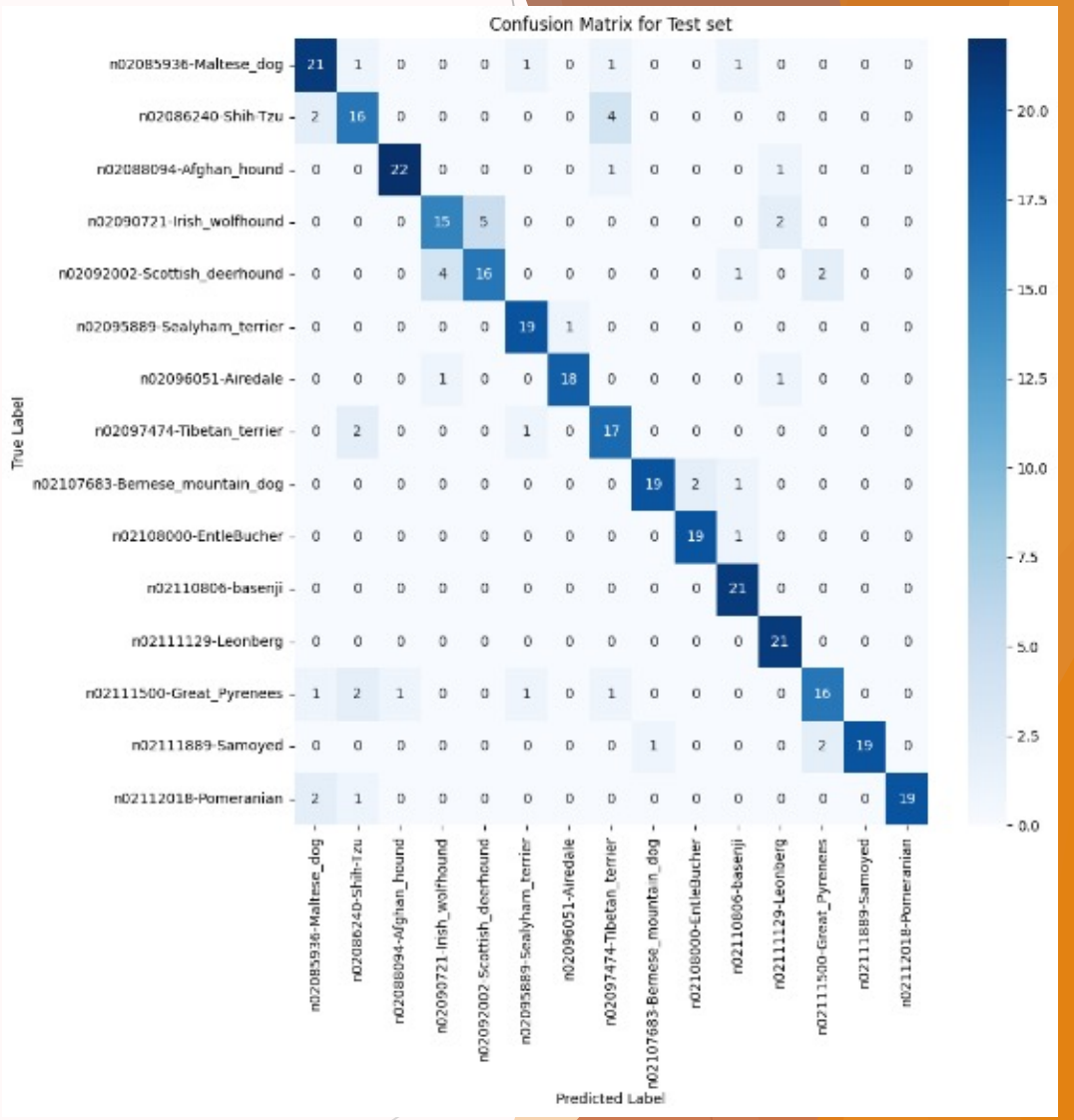


AdamW optimizer

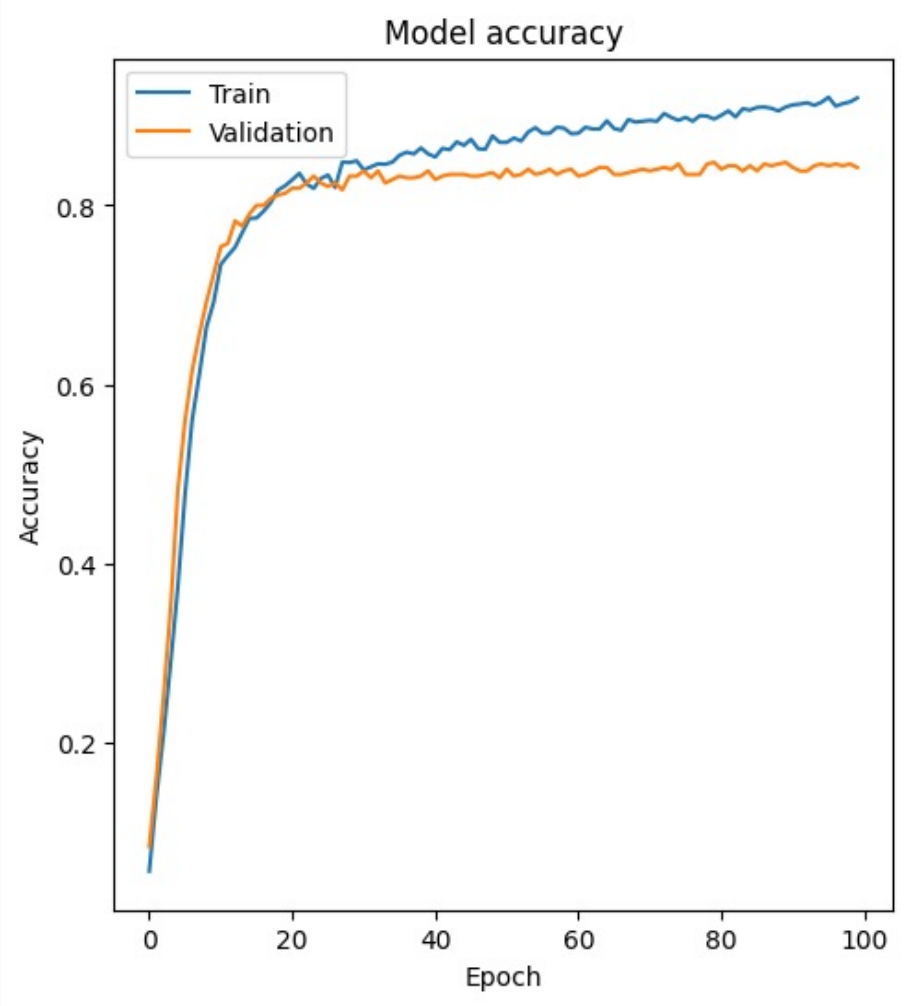
- **La performance du score d'accuracy globale est de 0.85**
- **La précision pour les 15 races de chiens oscille entre un minimum de 0.71 et un maximum de 1**
- **Le rappel (recall) pour les 15 races de chiens oscille entre un minimum de 0.68 et un maximum de 1**
- **Le temps d'entraînement du modèle sur 100 épochs a duré 88 minutes**
- **En ce qui concerne l'évolution de la fonction de coût et du score d'exactitude, nous constatons un léger overfitting du jeu de données d'entraînement à partir de la 30ème époque**

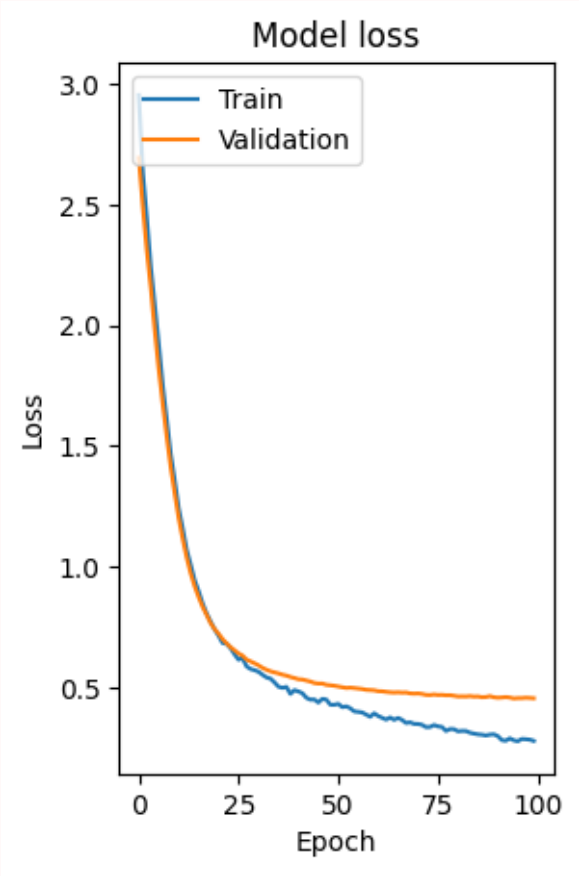
AdamW optimizer

	precision	recall	f1-score	support
n02085936-Maltese_dog	0.81	0.84	0.82	25
n02086240-Shih-Tzu	0.73	0.73	0.73	22
n02088094-Afghan_hound	0.96	0.92	0.94	24
n02090721-Irish_wolfhound	0.75	0.68	0.71	22
n02092002-Scottish_deerhound	0.76	0.70	0.73	23
n02095889-Sealyham_terrier	0.86	0.95	0.90	20
n02096051-Airedale	0.95	0.90	0.92	20
n02097474-Tibetan_terrier	0.71	0.85	0.77	20
n02107683-Bernese_mountain_dog	0.95	0.86	0.90	22
n02108000-EntleBucher	0.90	0.95	0.93	20
n02110806-basenji	0.84	1.00	0.91	21
n02111129-Leonberg	0.84	1.00	0.91	21
n02111500-Great_Pyrenees	0.80	0.73	0.76	22
n02111889-Samoyed	1.00	0.86	0.93	22
n02112018-Pomeranian	1.00	0.86	0.93	22
accuracy			0.85	326
macro avg	0.86	0.86	0.85	326
weighted avg	0.86	0.85	0.85	326



AdamW optimizer



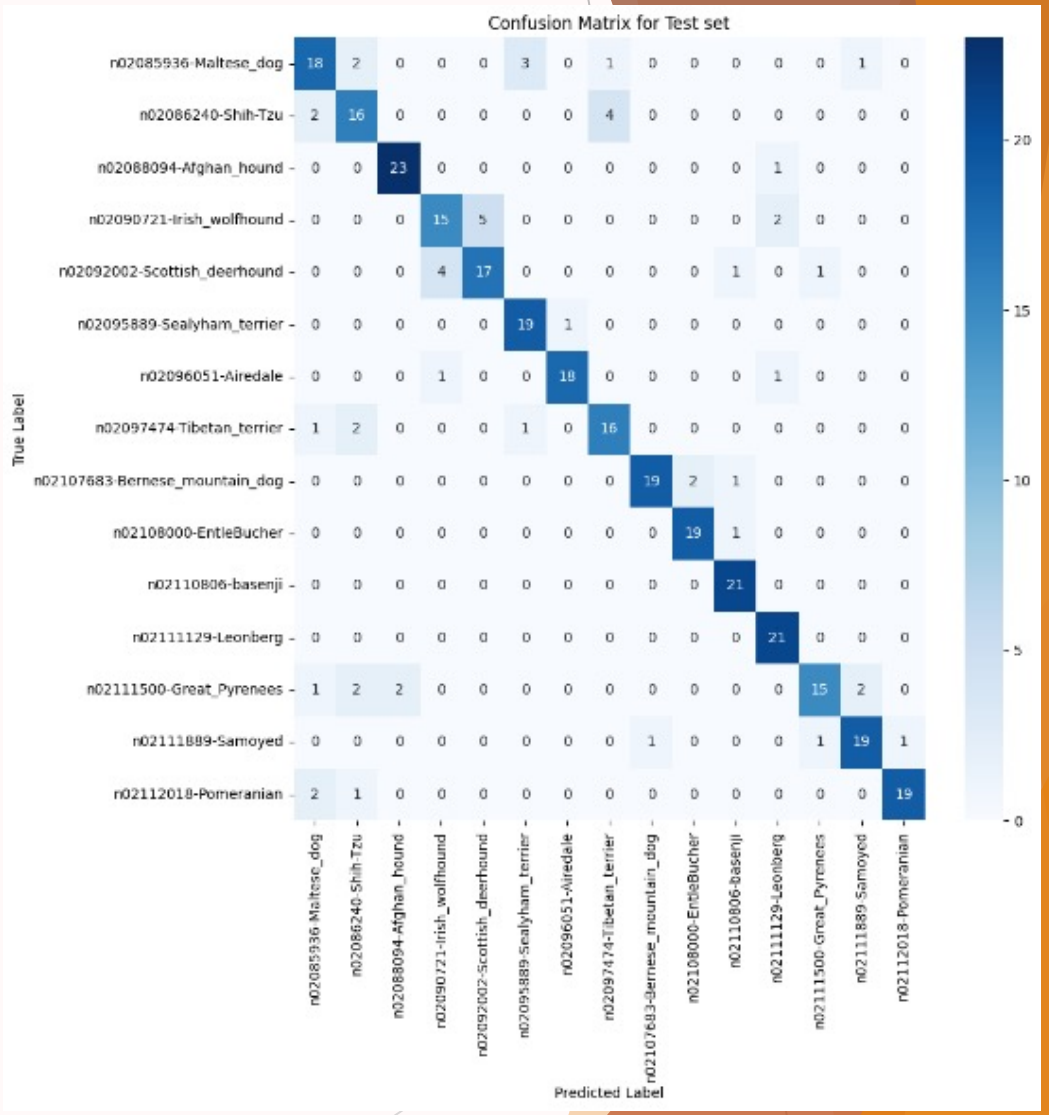


Lion optimizer

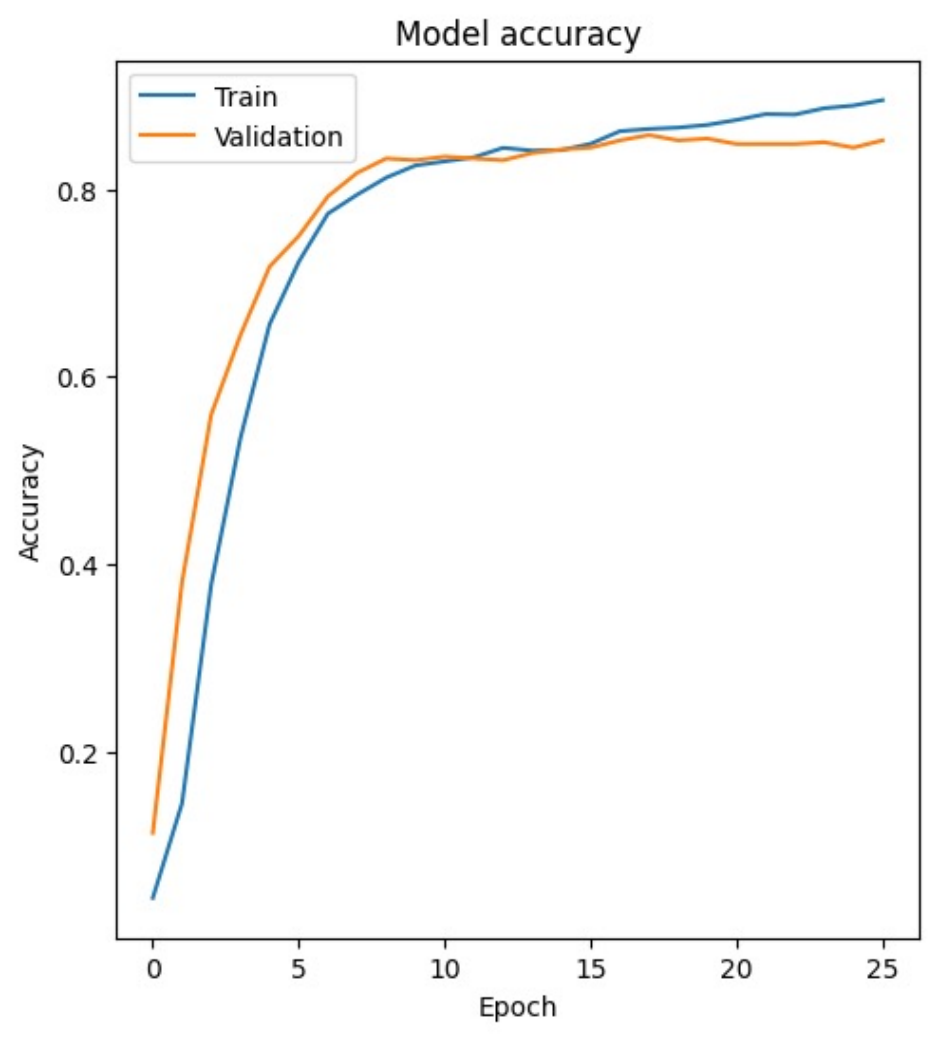
- **La performance du score d’accuracy globale est de 0.84**
- **La précision pour les 15 races de chiens oscille entre un minimum de 0.70 et un maximum de 1**
- **Le rappel (recall) pour les 15 races de chiens oscille entre un minimum de 0.68 et un maximum de 1**
- **Le temps d'entraînement du modèle sur 26 épochs a duré 22 minutes**
- **En ce qui concerne l'évolution de la fonction de coût et du score d'exactitude, nous constatons un léger overfitting du jeu de données d'entraînement à partir de la 15ème époque.**

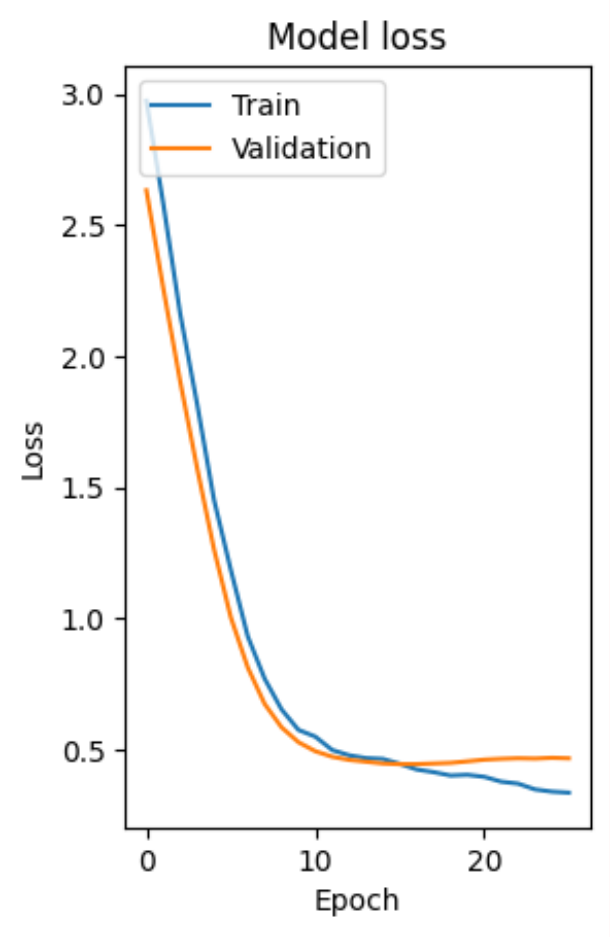
Lion optimizer

	precision	recall	f1-score	support
n02085936-Maltese_dog	0.75	0.72	0.73	25
n02086240-Shih-Tzu	0.70	0.73	0.71	22
n02088094-Afghan_hound	0.92	0.96	0.94	24
n02090721-Irish_wolfhound	0.75	0.68	0.71	22
n02092002-Scottish_deerhound	0.77	0.74	0.76	23
n02095889-Sealyham_terrier	0.83	0.95	0.88	20
n02096051-Airedale	0.95	0.90	0.92	20
n02097474-Tibetan_terrier	0.76	0.80	0.78	20
n02107683-Bernese_mountain_dog	0.95	0.86	0.90	22
n02108000-EntleBucher	0.90	0.95	0.93	20
n02110806-basenji	0.88	1.00	0.93	21
n02111129-Leonberg	0.84	1.00	0.91	21
n02111500-Great_Pyrenees	0.88	0.68	0.77	22
n02111889-Samoyed	0.86	0.86	0.86	22
n02112018-Pomeranian	0.95	0.86	0.90	22
accuracy			0.84	326
macro avg	0.85	0.85	0.84	326
weighted avg	0.84	0.84	0.84	326



Lion optimizer





CONCLUSION

Ces résultats démontrent l'efficacité présumée de l'utilisation de l'algorithme d'optimisation Lion, ayant dans le cadre de ce POC pour modèle support Xception, en ce qui concerne la classification des races de chiens.

- Les algorithmes d'optimisation AdamW et Lion affichent de légères meilleures performances, avec une légère avance pour AdamW.
- Cependant, cette différence n'est pas significative.
- Une différence majeure est observée en termes de temps d'entraînement : l'algorithme Lion converge rapidement vers le minimum global de la fonction de coût.
- Le système d'arrêt anticipé (early stopping) s'active après une patience de 10 époques, en se basant sur le score de validation.

ALLER PLUS LOIN

- Point n° 1

L'efficacité de Lion peut varier en fonction de la tâche spécifique, du modèle pré-entraîné choisi, de la taille et de la qualité de l'ensemble du jeu de données, de leurs prétraitements, et autres.

- Point n° 2

L'exploration de différentes valeurs pour Lion, ainsi que l'utilisation d'autres combinaisons d'architectures et de modèles différents, voir d'autres d'optimiseurs en termes de comparatif peut s'avérer utile.

DISCUSSION

MERCI POUR VOTRE ÉCOUTE !

Guillaume Lemêle

Parcours : Machine Learning Engineer

10/2022 – 06/2023