Openclassrooms

Rapport

Projet 7 - Développez une preuve de concept (option stage)

Guillaume lemêle

11/06/2023

Table des matières

1. Introduction

Dans le dynamisme de la data science et du machine learning, la capacité d'un ingénieur en apprentissage automatisé à se tenir à jour et à maîtriser rapidement de nouvelles compétences est essentielle.

Par exemple ; l'aptitude à explorer, comprendre et mettre en œuvre de façon autonome les récents algorithmes d'optimisation est particulièrement pertinente ; souvent démontrée par des preuves de concept (POC).

Les POC, bien que généralement succinctes et partielles, sont des démonstrations pratiques fondamentales de nouveaux concepts, et servent de tremplin vers la création de prototypes entièrement fonctionnels.

Elles sont applicables dans divers domaines, notamment celui du machine learning.

La conception d'une POC requiert une exploration approfondie du domaine cible, comprenant la création d'un plan de travail qui spécifie ici l'algorithme d'optimisation proposé, le dataset pour évaluation, des arguments soutenant ces choix, et des références clés pour appuyer l'approche.

En conclusion, la POC doit détailler le sujet retenu, le dataset d'évaluation, la méthode mise en œuvre, ainsi que des analyses comparatives et discussions pertinentes.

2. Thématique

Les réseaux de neurones convolutifs (CNN) ont démontré une compétence remarquable pour extraire des caractéristiques complexes des images, ce qui facilite grandement la classification d'images.

Cependant, les modèles CNN génériques peuvent ne pas être idéaux pour toutes les tâches, d'où l'importance du fine-tuning, qui ajuste les poids d'un modèle pré-entraîné pour des tâches spécifiques. Cette approche peut améliorer significativement les performances de classification.

Dans le cadre du fine-tuning, les algorithmes d'optimisation jouent un rôle clé. Adam, en particulier, a été largement reconnu pour sa capacité à accélérer la convergence du gradient dans les réseaux de neurones profonds. Cet algorithme, introduit par Diederik P. Kingma et Jimmy Ba dans leur article de 2014 intitulé "Adam : Une Méthode pour l'Optimisation Stochastique.", ajuste le taux d'apprentissage pour chaque poids dans le réseau de manière adaptative, en se basant sur les estimations du premier et du second moment des gradients.

Récemment, des avancées importantes ont été réalisées dans le développement et l'optimisation des algorithmes d'optimisation.

Nous allons donc nous concentrer sur un des derniers algorithme d'optimisation développé dans l'état de l'art ; en date de mars 2023 : Lion, signifiant Evo**L**ved S**i**gn M**o**me**n**tum.

3. Etat de l'art

La descente du gradient est une méthode d'optimisation couramment utilisée pour minimiser les erreurs des algorithmes d'apprentissage automatique. Elle est basée sur l'idée de progresser itérativement vers le minimum global de la fonction de coût en se déplaçant dans la direction de la pente la plus abrupte à chaque pas.

Le taux d'apprentissage, ou "learning rate", est un paramètre clé dans la plupart des algorithmes d'optimisation de l'apprentissage automatique. Il détermine la taille des pas que fait l'algorithme pour atteindre le minimum global d'une fonction coût. Un taux d'apprentissage élevé peut permettre à l'algorithme de converger rapidement, mais il risque également de passer au-delà du minimum et de causer une instabilité. À l'inverse, un taux d'apprentissage faible peut conduire à une convergence plus stable, mais l'algorithme pourrait prendre beaucoup de temps pour atteindre ce minimum, voire rester coincé dans un minimum local. Par conséquent, le choix du taux d'apprentissage est souvent un compromis entre rapidité de convergence et stabilité de l'apprentissage.

De nombreuses méthodes ont été développées pour adapter dynamiquement le taux d'apprentissage pendant l'entraînement, comme l'échéancier du taux d'apprentissage ou "Learning Rate Scheduler", qui ajuste le taux d'apprentissage en fonction de la progression de l'entraînement. Par ailleurs, des algorithmes tels qu'Adam et AdamW se distinguent par leur capacité d'adaptation.

En effet, ils ajustent de manière automatique le taux d'apprentissage pour chaque poids du réseau, s'appuyant pour cela sur l'historique des gradients. Bien que les algorithmes comme Adam et AdamW ajustent le taux d'apprentissage pour chaque poids du réseau de manière adaptative, ils peuvent encore bénéficier d'un Learning Rate Scheduler. En somme, l'utilisation d'un Learning Rate Scheduler avec Adam ou AdamW peut combiner les avantages de l'ajustement adaptatif du taux d'apprentissage pour chaque poids et de l'ajustement global du taux d'apprentissage au cours du temps. Cette combinaison peut offrir une plus grande flexibilité et potentiellement améliorer les résultats d'apprentissage du modèle.

Nous devons aussi parler du momentum qui a pour but d'optimiser la descente du gradient et d'éviter les minimums locaux. En termes d'optimisation, le momentum aide à surmonter les petits obstacles et à éviter les minima locaux en accumulant la direction précédente du gradient pour déterminer la direction suivante à prendre, pour réaliser des mises à jour plus fluides et rapides des poids du modèle.

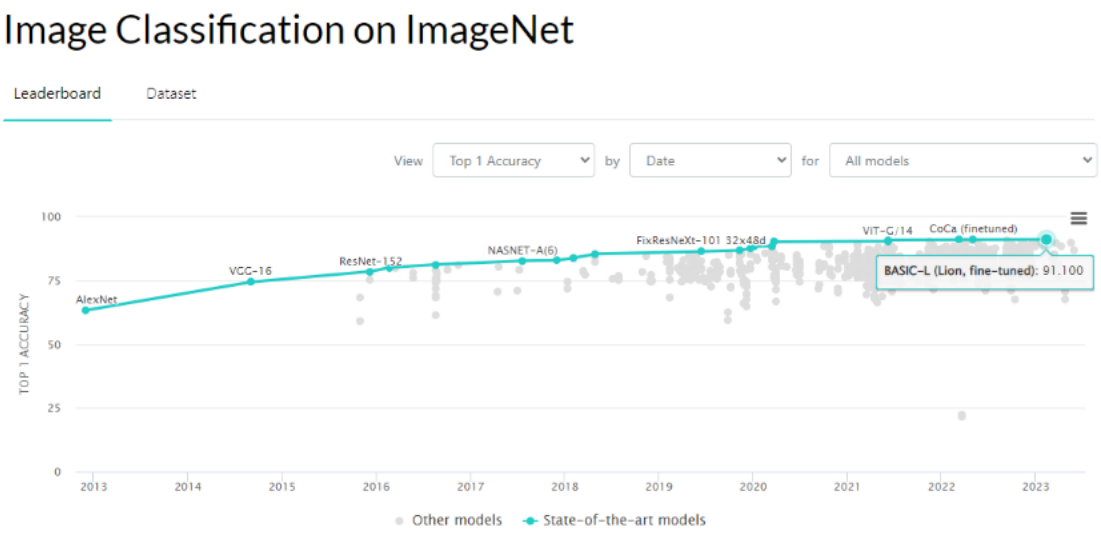
Adam (Adaptive Moment Estimation) est un algorithme d'optimisation qui a pris de l'ampleur au fils des ans. Il combine les avantages du momentum et de l'adaptation du taux d'apprentissage (learning

rate). Adam utilise deux coefficients de momentum, B1 et B2, permettant à Adam de s'adapter dynamiquement au cours de l'entraînement. Adam intègre le weight decay comme stratégie de régularisation afin de prévenir le surajustement du modèle. Le weight decay fonctionne en ajoutant une pénalité à la fonction de coût, favorisant la stabilité et la généralisation de l'apprentissage. Cependant, il est important de noter que l'interaction d'Adam avec le weight decay n'est pas toujours optimale, ce qui a conduit à l'introduction d'AdamW et du concept de "Decoupled Weight Decay".

AdamW introduit le "Decoupled Weight Decay" en séparant le weight decay de la mise à jour des poids, ce qui le rend compatible avec l'adaptation du taux d'apprentissage. En conséquence, AdamW tend à surperformer Adam dans diverses tâches d'apprentissage.

Depuis, de nombreux algorithmes ont été inspirés par Adam et AdamW, tel que Lion, acronyme de Evo**L**ved s**I**gn m**O**me**N**tum, qui est un algorithme d'optimisation qui se distingue de nombreux autres par son suivi exclusif du momentum. Cela entraîne une réduction de la surcharge de mémoire. Par rapport à AdamW, Lion est plus simple et présente donc moins d'hyperparamètres, car il n'exige pas le facteur ε ni ceux liés à la factorisation. Lion n'enregistre que le momentum, ce qui réduit son empreinte mémoire par rapport à des optimiseurs adaptatifs populaires comme AdamW. Cela est bénéfique lors de l'entraînement de grands modèles ou de l'utilisation de batch de grande taille.

Toutefois, l'évaluation de Lion reste limitée aux tâches choisies. Sur des tâches de vision, les différences entre Lion, AdamW et le SGD avec momentum sont minimes sur les ResNets, probablement car les ConvNets sont plus faciles à optimiser comparativement aux Transformers. Une autre limitation potentielle concerne la taille du batch. Lion pourrait ne pas être meilleur qu'AdamW si la taille du lot est petite (<64). En ce qui concerne la classification des images basé sur le dataset ImageNet, nous pouvons voir sur ce benchmark les performances de notre optimisateur sur le score top-1. Ce benchmark est disponible sur le site Paper with code.



4. Prototype à implémenter (Dataset, Baseline, méthode mise en œuvre)

Nous désirons repartir de notre modèle pré-entraîné Xception du projet 6, qui se servait de l'algorithme d'optimisation Adam avec Tensorflow et Keras, et qui performe à 0.77 sur son score d'accuracy global. 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.

Ce POC utilisera donc Xception en tant que modèle support. Notre algorithme d'optimisation baseline est celui que nous avions utilisé sur ce modèle lors du projet 6 ; soit Adam, que nous comparerons avec les algorithmes d'optimisations AdamW et Lion (Evo**L**ved S**i**gn M**o**me**n**tum).

Lion peut permettre à notre modèle Xception de converger plus rapidement et d'obtenir des résultats plus stables. De plus, Lion présente un avantage en termes d'empreinte mémoire : en ne sauvegardant que le momentum, Lion consomme moins de mémoire que Adam ou AdamW.

Nous utiliserons le même type de prétraitement sur notre dataset crée à partir du dataset fournit par le projet 6 : « Classez des images à l'aide d'algorithmes de Deep Learning », soit le « Stanford Dogs Dataset ». Nous utiliserons un dataset d'images prétraités avec un générateur Keras et qui sera fourni avec les livrables.

Nous disposons des ressources suivantes en ce qui concerne les temps d'entraînement : 32go de ram et un Intel Core i5 10400F 6x 4.3GHz (pas d'utilisation de GPU).

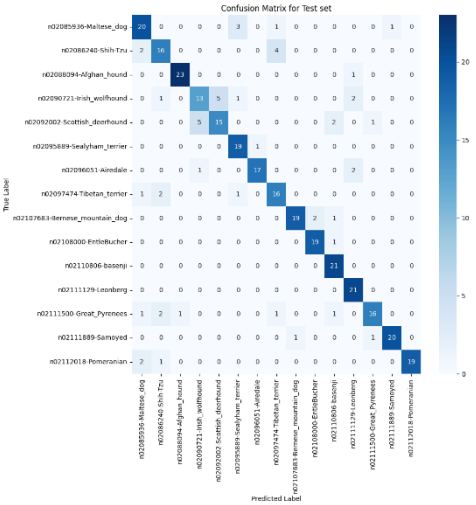
En fin de compte, l'objectif du POC est de démontrer les avantages potentiels des nouveaux algorithmes d'optimisation sur l'efficacité de la classification d'images. Nous comparons différents optimisateurs grâce au score d'accuracy, precision, recall et F1. Nous ferons aussi une visualisation graphique grâce à une matrice de confusion.

Néanmoins, il ne faut pas oublier la performance d'un optimiseur peut dépendre de nombreux facteurs, y compris la taille et la qualité de l'ensemble de données, et d'autres aspects de la configuration de l'entraînement.

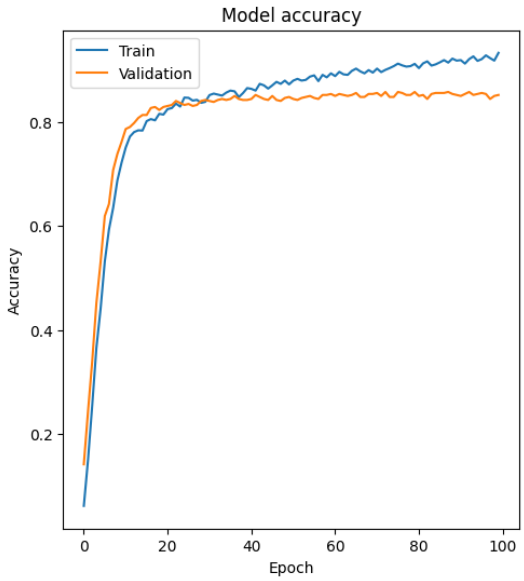
Par conséquent, bien que Lion ait le potentiel d'être plus efficace qu'Adam ou AdamW, cela devrait être vérifié expérimentalement dans le cadre d'une étude de performance plus globale que notre simple dataset avec un modèle donné.

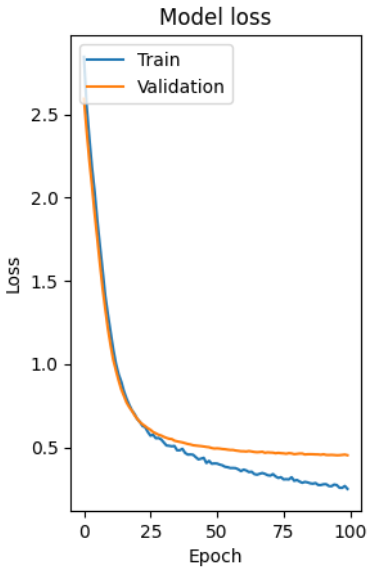
5. Analyse des résultats

A) Algorithme Baseline : Adam



	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





Nous observons une performance du score d'exactitude globale équivalent à : 0.84.

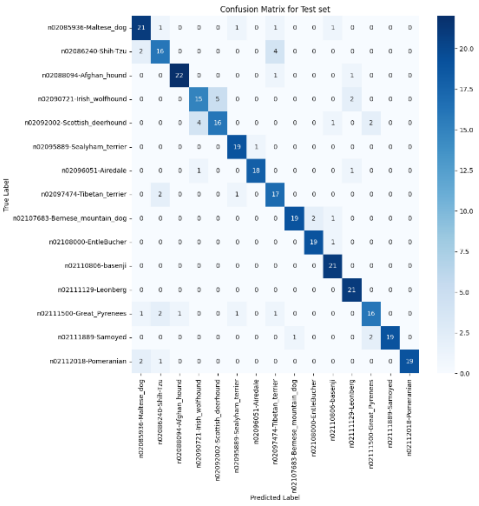
Nous observons une performance du score de précision au sein des 15 races de chiens oscillants entre un minimum de 0.68 et un maximum de 1.

Nous observons une performance du score de recall au sein des 15 races de chiens oscillants entre un minimum de 0.65 et un maximum de 1.

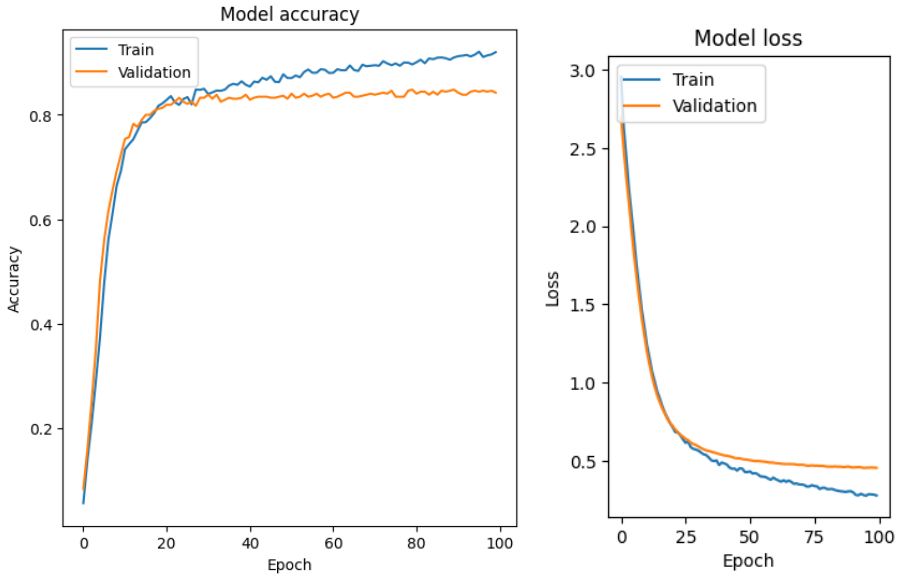
Le temps d'entraînement du modèle sur 100 époch a pris 84 minutes.

En ce qui concerne l'évolution de la fonction de coût et du score d'exactitude, nous observons un entraînement qui conduit à un léger l'overfitting du train set à partir du 40ème époque.

B) Algorithme 1 : AdamW



	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



Nous observons une performance du score d'exactitude globale équivalent à : 0.85.

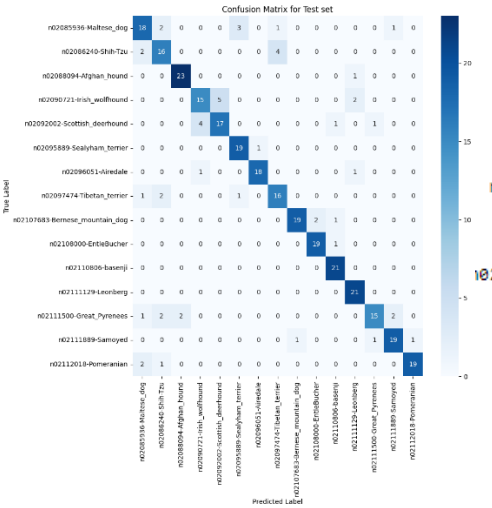
Nous observons une performance du score de précision au sein des 15 races de chiens oscillants entre un minimum de 0.71 et un maximum de 1.

Nous observons une performance du score de recall au sein des 15 races de chiens oscillants entre un minimum de 0.68 et un maximum de 1.

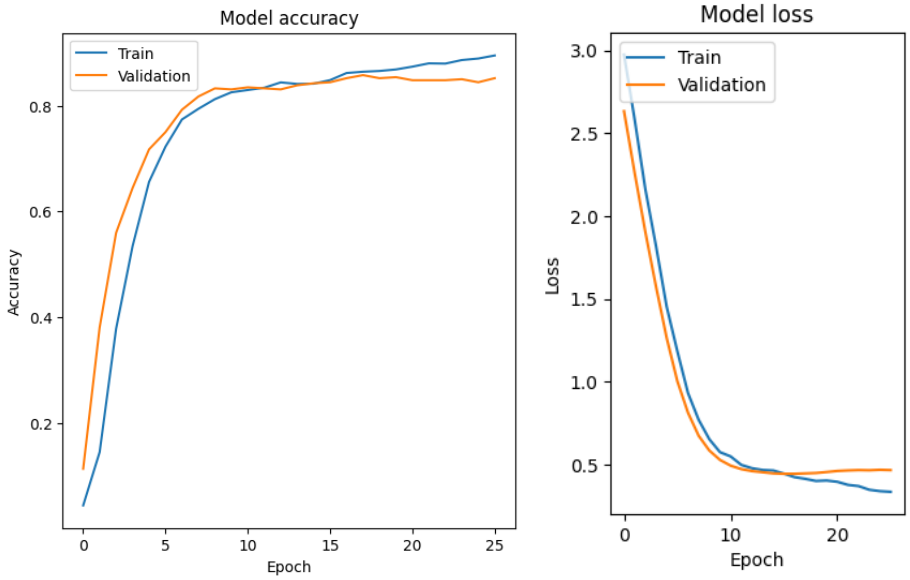
Le temps d'entraînement du modèle sur 100 époch a pris 88 minutes.

En ce qui concerne l'évolution de la fonction de coût et du score d'exactitude, nous observons un entraînement qui conduit à un léger l'overfitting du train set à partir du 30ème époque.

C) Algorithme 2 : Lion



	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



Nous observons une performance du score d'exactitude globale équivalent à : 0.84.

Nous observons une performance du score de précision au sein des 15 races de chiens oscillants entre un minimum de 0.70 et un maximum de 1.

Nous observons une performance du score de recall au sein des 15 races de chiens oscillants entre un minimum de 0.68 et un maximum de 1.

Le temps d'entraînement du modèle sur 26 époch a pris 22 minutes.

En ce qui concerne l'évolution de la fonction de coût et du score d'exactitude, nous observons un entraînement qui conduit à un léger l'overfitting du train set à partir du 15ème époque.

6. Conclusion

Dans le cadre de ce projet, j'ai testé différents algorithmes d'optimisation, Adam pour l'algorithme baseline, puis AdamW et Lion pour la classification des races de chiens. En comparant les résultats obtenus, des différences significatives ont été observées au niveau du temps d'entraînement.

Avec l'algorithme d'optimisation Adam, l'accuracy globale obtenue est de 0,84 et la précision varie entre 0,68 et 1, tandis que le rappel varie entre 0,65 et 1. Le modèle a été entraîné sur 100 époques et a durée 84 minutes.

Avec l'algorithme d'optimisation AdamW, l'accuracy globale obtenue est de 0,85 et la précision varie entre 0,71 et 1, tandis que le rappel varie entre 0,68 et 1. Le modèle a été entraîné sur 100 époques et a durée 88 minutes.

Avec l'algorithme d'optimisation Lion, l'accuracy globale obtenue est de 0,84 et la précision varie entre 0,70 et 1, tandis que le rappel varie entre 0,68 et 1. Le modèle a été entraîné sur 25 époques et a durée 22 minutes.

Globalement, les deux algorithmes d'optimisations AdamW et Lion sont légèrement meilleurs en termes de performance, avec AdamW en tête, même si cela ne reste tout de même pas significatif.

Néanmoins, une différence éclatante ce fait lors du temps d'entraînement avec l'algorithme Lion qui arrive très rapidement à converger vers le minimum global de la fonction de coût ; le early stopping s'activant après une patience de 10 époques (suivant le score de validation).

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.

Cependant, il est important de nuancer en notant que l'efficacité de cette combinaison 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 prétraitement, et autres.

En conclusion, ces résultats ouvrent des pistes intéressantes pour des recherches futures, notamment par l'exploration de différentes valeurs pour les hyperparamètres de Lion, ainsi que l'utilisation d'autres combinaisons d'architectures et de modèles différents, voir d'autres d'optimiseurs en terme de comparatif, pour ces mêmes ou d'autres tâches de classification d'images, ou autre.

7. Bibliographie et webographie

Arxiz. Symbolic Discovery of Optimization Algorithms. Retrieved May 31, 2023, from https://arxiv.org/pdf/2302.06675v4.pdf

Hasty.ai. Lion. Retrieved May 31, 2023, from

https://hasty.ai/docs/mp-wiki/solvers-optimizers/lion

Papers with Code. Image Classification on ImageNet. Retrieved May 31, 2023, from https://paperswithcode.com/sota/image-classification-on-imagenet

CNRS – IDRIS. Deep Learning Optimisé - Jean Zay. Retrieved June 6, 2023, from http://www.idris.fr/media/formations/dlo-jz/dlojz4-optimiseurs.pdf

Apprendre_le_ML_en_une_semaine Retrieved June 6, 2023, from https://machinelearnia.com/apprendre-le-machine-learning-en-une-semaine/