Rapport de Veille technologique et POC pour l'identification améliorée de chiens

Projet 7

Auteur : J-F Voelker

Table des matières

1. Introduction

Lors du projet 6, j'ai été amené à travailler sur la classification d'images de chiens. Il s'agissait d'identifier automatique la race du chien présent sur l'image. La méthode de machine learning appliquée était un réseau de neurones convolutif dit CNN. Le meilleur résultat obtenu a été avec le modèle ResNet50, pré-entraîné sur ImageNet et avec un classifieur fité sur le Stanford Dogs dataset. L'accuracy obtenue était de 70%.

2. Veille technologique

Je souhaite appliquer une méthode qui me permettrait d'obtenir un meilleur score d'accuracy sur l'identification des chiens.

Sur le site « paperswithcode », le classement de classification sur le Sanford dogs dataset laisse apparaître que les meilleurs scores dernièrement obtenus, l'ont été avec des Transformers.

Rank	Model	Accuracy	Paper	Code	Result	Year
1	**TransFG**	92.3% (90.6%)	TransFG: A Transformer Architecture for Fine-grained Recognition			2021
2	**FFVT**	91.5%	Feature Fusion Vision Transformer for Fine-Grained Visual Categorization			2021
3	**API-Net**	90.3%	Learning Attentive Pairwise Interaction for Fine-Grained Classification			2020

Figure 1 - Classement des modèles sur Stanford Dogs

Il en est de même pour le dataset ImageNet-1k sur lequel le CNN avait été pré-entraîné :

Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	**PeCo** (ViT-H, 448)	88.3%			×	PeCo: Perceptual Codebook for BERT Pre-training of Vision Transformers			2021	Transformer, ImageNet-1k only
2	**MAE** (ViT-H, 448)	87.8%			×	Masked Autoencoders Are Scalable Vision Learners			2021	Transformer, ImageNet-1k only
3	**PeCo** (ViT-L, 224)	87.5%			×	PeCo: Perceptual Codebook for BERT Pre-training of Vision Transformers			2021	Transformer, ImageNet-1k only

Figure 2 - Classement des modèles sur ImageNet-1k

J'ai donc décidé d'évaluer la performance d'un ViT sur l'identification des races de chiens du Stanford Dogs dataset qui se compose de 20570 images de chiens réparties en 120 races.

Figure 3 - Exemples d'images du Stanford Dogs Dataset

3. Présentation d'un Visual Transformer

Les modèles de Transfomers sont d'abord apparus pour des tâches de Natural Language Processing (NLP) en 2017. Mais depuis fin 2020-début 2021, de nouveaux travaux ont démontré leur intérêt pour des tâches de Computer Vision. Cela a été notamment le cas pour la classification d'images.

Fonctionnement d'un transformer

Le fonctionnement d'un transformer repose uniquement sur le mécanisme de self-attention et se compose d'un encodeur selon le schéma suivant :

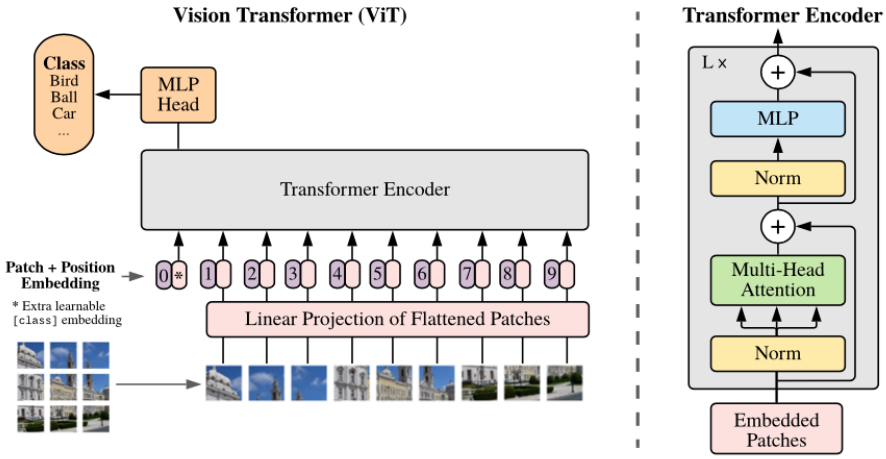


Figure 4 - Schéma de principe d'unVisual Transformer

Un ViT se compose des différentes étapes suivantes :

- L'image est découpée en morceaux de taille variable selon le modèle de ViT (ex : 16x16 ou 32x32), ces morceaux sont appelés patches. Par exemple l'image ci-dessous est divisée en 16 patches.

Figure 5 - Image subdivisée en patches

- Ces patches sont aplanis (flatten) et subissent ensuite une transformation linéaire.

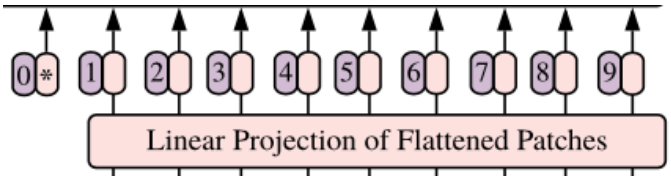
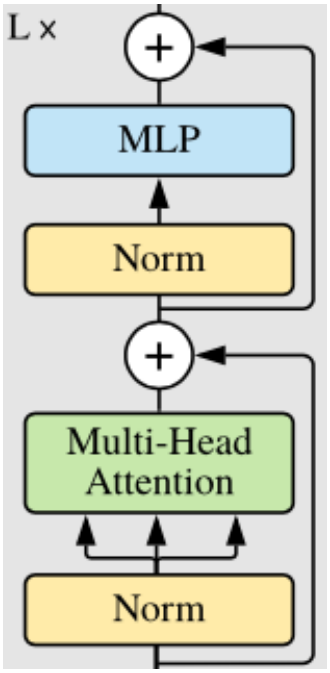


Figure 6 - Schéma des entrées

- A chacun de ces vecteurs est ajouté un vecteur qui contient l'encodage de la position du patche dans l'image de départ (voir figure 5). Ainsi le résultat tiendra compte de la position globale de chaque patche dans l'image initiale.

- Le vecteur 0 est créé pour transmettre la valeur y, c'est-à-dire la classe de l'image
- L'encodeur d'un transformer se compose d'un certain nombre de blocks « self-attention-feed-forward » (dans l'image ci-dessous, il y a L blocks)



- *Figure 7 - Boucle de rappel de la matrice d'entrée*

- A l'entrée un layer de normalisation (Norm) permet d'améliorer la généralisation du modèle et le temps d'entraînement.
- Les patches ainsi transformés forment une matrice E qui est ensuite multiplié par 3 matrices différentes apprises pendant la phase d'entraînement. Il en résulte les matrices :
    - Q pour Query,
    - K pour Key,
    - V pour value
- Ces 3 matrices passent dans un « Multi-Head Attention » et subissent les opérations suivantes :
    - Un produit scalaire est réalisé entre les vecteurs de Q et de K. Plus la valeur du produit scalaire est élevée plus un vecteur Key est pertinent pour un vecteur Query donné.
    - Réduction d'échelle en divisant par $\frac{1}{\sqrt{dimension\ des\ vecteur}}$
    - Application de la fonction d'activation softmax qui permet d'amplifier les valeurs élevées et d'écraser les valeurs déjà basses. La matrice qui résulte s'appelle la matrice d'attention A.
    - Finalement la matrice d'attention est multipliée avec la matrice V pour obtenir une matrice V filtrées sur les points d'attentions.

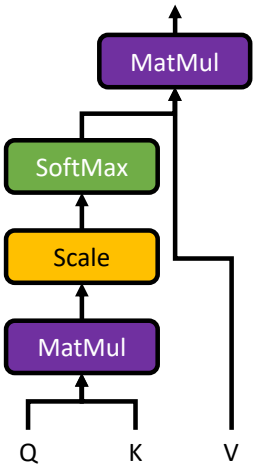


Figure 8 - Séquences d'un sub-layer self-attention

- Plusieurs jeux de matrices Q, K et V sont créés et permettent d'identifier simultanément plusieurs zones d'attention dans l'image, c'est pourquoi on parle de « Multi-Head Attention » layer.
- A cette matrice est rajoutée la matrice résiduelle pour éviter le problème de disparition de gradient (gradient vanishing problem) liée à la non-linéarité de certaines fonctions appliquées.
- Dans le feed-forward layer (MLP), ces matrices sont concaténées et subissent une transformation linéaire pour revenir aux dimensions de la matrice E et ainsi pouvoir être traitée par le block « self-attention/feed-forward » suivant.
- Pour notre problème de classification, seule la sortie correspondant au vecteur 0 est conservée et cette sortie passe dans classifieur (MLP) entraîné avec notre Stanford Dogs Dataset.

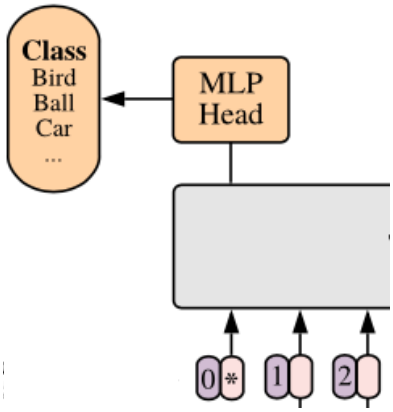


Figure 9 - Sortie vers le classifieur

4. Entrainements et résultats

J'ai récupéré les modèles suivants pré-entraînés sur les datasets ImageNet-21K et ImageNet2012 via la librairie « vit-keras ». J'ai entraîné le classifieur de chacun de ces modèles avec et sans data-augmentation sur le Stanford Dogs dataset.

Tableau 1 - Modèles récupérés avec pré-entraînement

Modèle	Attention Layers	Heads	Patch Size
ViT-B16	12	12	16x16
ViT-B32	12	12	32x32
ViT-L16	24	16	16x16
ViT-L32	24	16	32x32

4.1. Configuration ViT-B16 avec Data-Augmentation

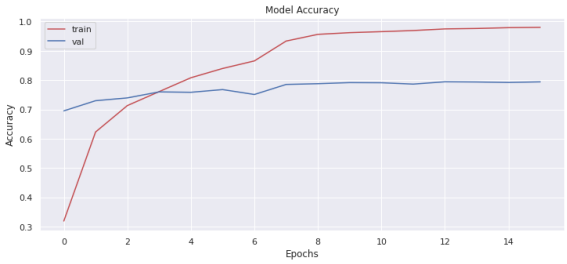


Figure 10 - Evolution Accuracy ViT-B16 avec Data-Aug

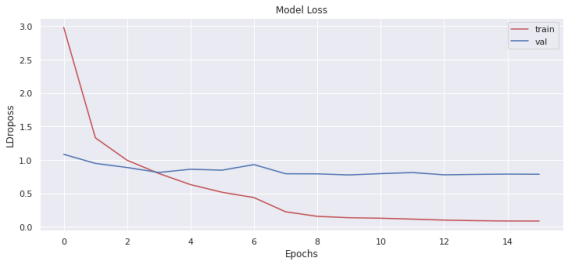


Figure 11 - Evolution Loss ViT-B16 avec Data-Aug

Test Accuracy : 0.826
Test Loss : 0.620

Observations :
Je constate que le modèle converge rapidement avec une bonne capacité de généralisation.

4.3. Configuration ViT-B16 sans Data-Augmentation

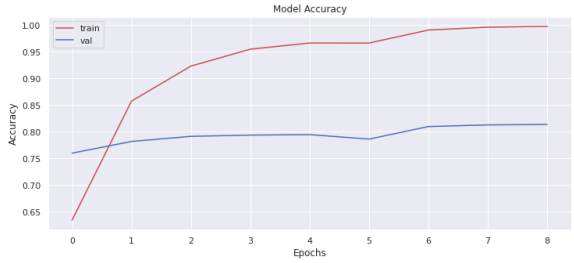


Figure 12 - Evolution Accuracy ViT-B16 sans Data-Aug

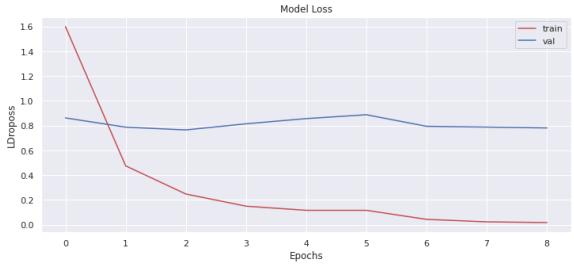


Figure 13 - Evolution Loss ViT-B16 sans Data-Aug

Test Accuracy : 0.838
Test Loss : 0.606

Observations :
Je constate que sans data-augmentation le modèle converge plus rapidement qu'avec. La performance est même légèrement supérieure.

4.4. Configuration ViT-B32 avec Data-Augmentation

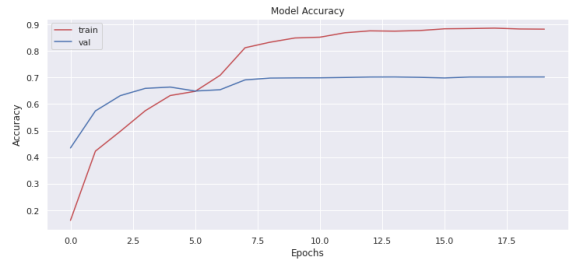


Figure 14 - Evolution Accuracy ViT-B32 avec Data-Aug

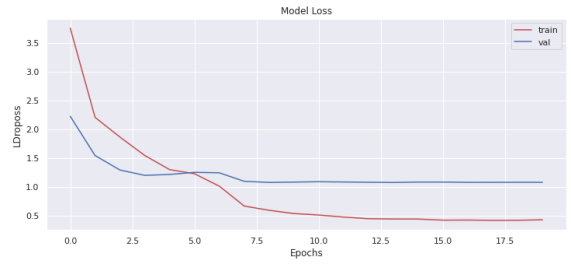


Figure 15 - Evolution Loss ViT-B32 avec Data-Aug

Test Accuracy : 0.725
Test Loss : 0.965

Observations :
Lorsque la dimension du patche augmente la performance finale décroît de l'ordre de 10%. La performance sur les training et validation dataset décroit aussi de 10%.

4.5. Configuration ViT-B32 sans Data-Augmentation

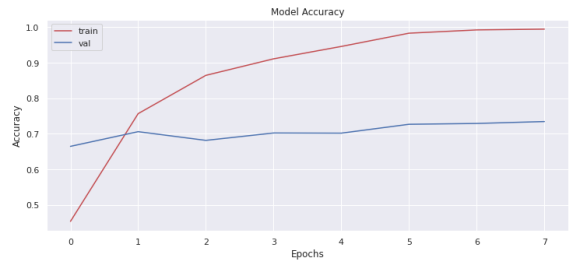


Figure 16 - Evolution Accuracy ViT-B32 sans Data-Aug

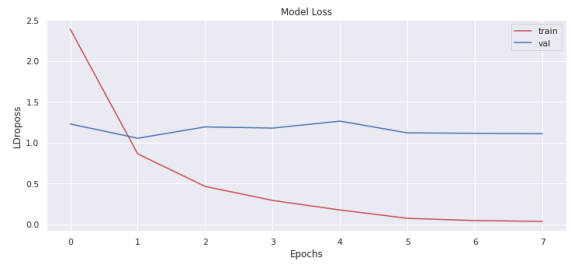


Figure 17 - Evolution Loss ViT-B32 sans Data-Aug

Test Accuracy : 0.756

Test Loss : 0.907

Observations :
J'observe qu'avec le patching de 32x32, que la performance s'améliore sans data-augmentation.

4.6. Configuration ViT-L16 avec Data-Augmentation

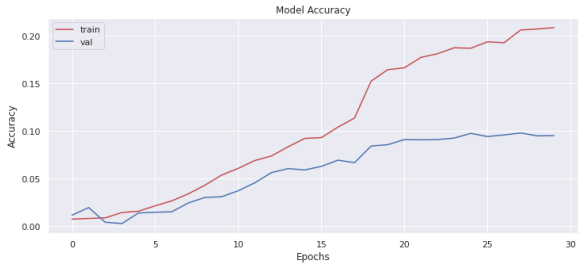


Figure 18 - Evolution Accuracy ViT-L16 avec Data-Aug

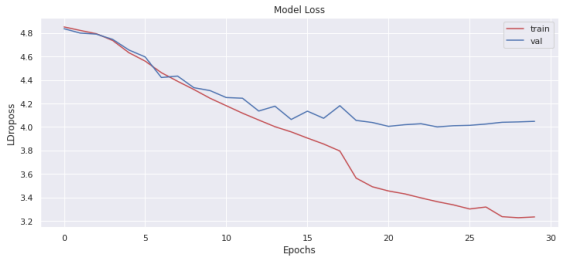


Figure 19 - Evolution Loss ViT-L16 avec Data-Aug

Test Accuracy : 0.153
Test Loss : 3.821

Observations :
Ce modèle donne les plus mauvais résultats de tous les scénarios testés. Sa performance presque 5 fois inférieure au modèle sans data-augmentation.

4.7. Configuration ViT-L16 sans Data-Augmentation

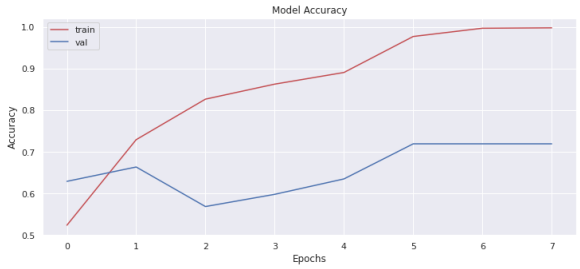


Figure 20 - Evolution Accuracy ViT-L16 sans Data-Aug

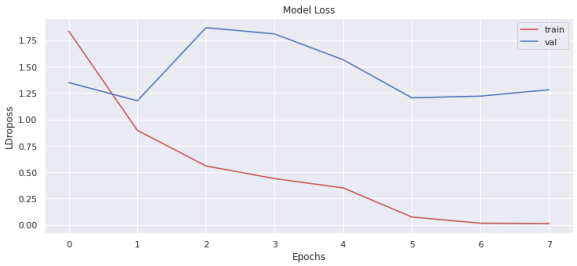


Figure 21 - Evolution Loss ViT-L16 sans Data-Aug

Test Accuracy : 0.745
Test Loss : 1.088

Observations :
J'observe des performances similaires au modèle ViT-B32 pour une durée de calcul par epoch cependant plus longue.

4.9. Configuration ViT-L32 avec Data-Augmentation

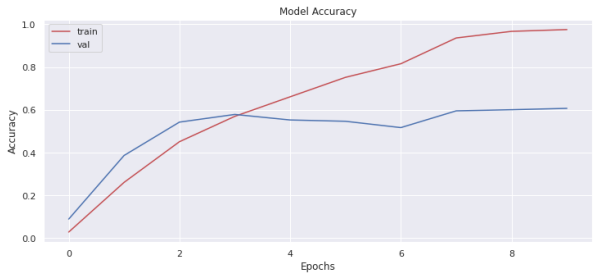


Figure 22 - Evolution Accuracy ViT-L32 avec Data-Aug

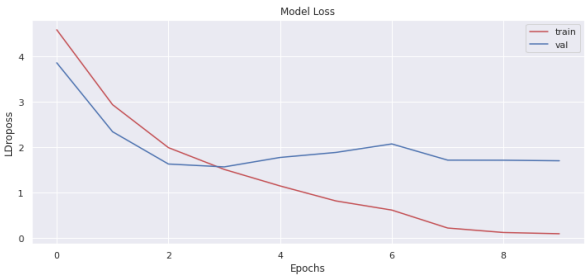


Figure 23 - Evolution Loss ViT-L32 avec Data-Aug

Test Accuracy : 0.642
Test Loss : 1.516

Observations :
J'observe un important écart entre les performances du training dataset et celles du validation dataset. Ces dernières demeurent cependant proches de celles du test dataset.

4.10. Configuration ViT-L32 sans Data-Augmentation

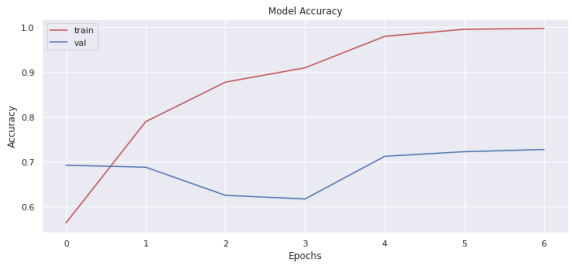


Figure 24 - Evolution Accuracy ViT-L32 sans Data-Aug

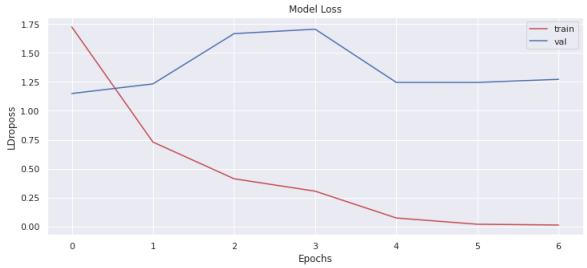


Figure 25 - Evolution Loss ViT-L32 sans Data-Aug

Test Accuracy : 0.729
Test Loss : 1.225

Observations :
L'absence de data-augmentation améliore les performances.

5. Conclusions

Des différents scénarios testés, je constate que :

- Le modèle ViT-L plus complexe réalise des performances moins bonnes que le modèle ViT-B.
- Les configurations basées sur le modèle ViT-B sont systématiquement plus performantes que la baseline CNN-ResNet
- Un ViT avec un patching à 16x16 est plus performant mais est plus long à entraîner qu'un modèle avec un patching 32x32
- Les modèles de ViT testés ont de meilleures performances sans Data-Augmentation
- A l'exception des modèles ViT-L avec data-augmentation, tous les ViT sont plus performant que le CNN-ResNet.

- A l'exception du modèle ViT-L16, tous les ViT s'entraînement plus rapidement que le CNN-ResNet.

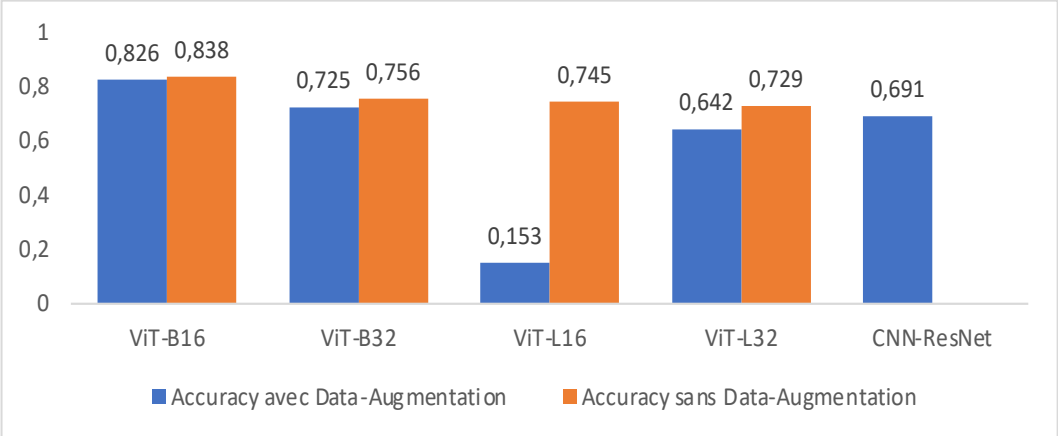


Figure 26 - Accuraccy des différentes configurations

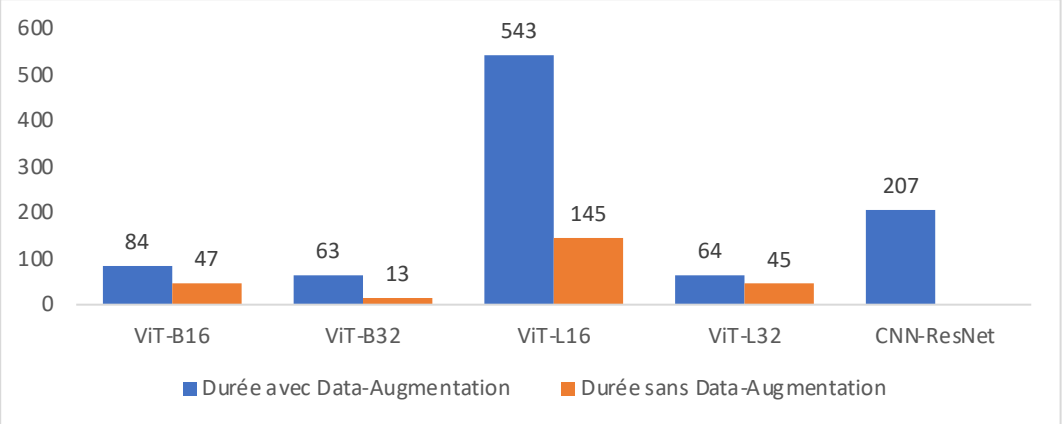


Figure 27 - Temps d'entraînement des différentes configurations

J'obtiens avec le ViT-B16 sans Data-Augmentation un score d'accuracy amélioré de 14,7% par rapport à la baseline constituée par le modèle CNN-ResNet avec Data-Augmentation. Je recommande donc de remplacer le modèle actuellement déployé en production par un modèle ViT-B16. Ceci permettra de diviser quasiment par 2 le nombre d'images mal identifiées.

6. Annexes

Tableau 2 - Tableau récapitulatif des résultats

Modèle	Data-Aug	Test Loss	Test Accuracy	Durée Epoch (s)	Nombre Epochs	Durée (min)
ViT-B16	OUI	0.620	0.826	314	16	84
ViT-B16	NON	0.606	0.838	314	9	47
ViT-B32	OUI	0.965	0.725	189	20	63
ViT-B32	NON	0.907	0.756	96	8	13
ViT-L16	OUI	3.822	0.153	1085	30	543
ViT-L16	NON	1.088	0.745	1085	8	145
ViT-L32	OUI	1.516	0.642	385	10	64
ViT-L32	NON	1.225	0.729	382	7	45
CNN-ResNet	OUI	1.233	0.691	170	73	207

Références bibliographiques :

1-Attention Is All You Need, https://arxiv.org/abs/1706.03762

2-An Image Is Worth 16x16 Words Transformers for Image Recognition at Scale, https://arxiv.org/abs/2010.11929

3-How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers, https://arxiv.org/abs/2106.10270

4-Visual Guide to Transformer Neural Networks:
https://www.youtube.com/watch?v=dichIcUZfOw&t=0s
https://www.youtube.com/watch?v=mMa2PmYJlCo&t=0s
https://www.youtube.com/watch?v=gJ9kaJsE78k&t=0s

5-The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

6-Getting meaning from text: self-attention step-by-step video, https://peltarion.com/blog/data-science/self-attention-video