Rapport Projet 7

Développer une preuve de concept

Classification d'images de style cartoon avec un Vision Transformer (ViT)

Comparaison avec un réseau de neurones convolutionnels (CNN)

Marie-France LAROCHE-BARTHET

Mars 2022

Introduction

La classification d'images a toujours été un domaine de fort intérêt pour la recherche. L'émergence du Deep Learning a favorisé le développement de ce domaine et depuis 10 ans les avancées font des pas de géant. Par ailleurs, avec la généralisation des données au format numérique, de plus en plus d'applications sont développées. Celles-ci utilisent des images non pas dans le sens de photos d'objets réels, mais plutôt des images dessinées, voire de style cartoon dans certains cas. Nous nous sommes intéressés à la classification de telles images en comparant deux technologies.

Article de Chen *et al*.

Cet article publié en novembre 2021 fait un état des lieux des algorithmes de classification d'images basés sur les réseaux de neurones convolutifs (CNN). Ceux-ci sont progressivement devenus l'algorithme principal de classification d'images depuis 2012 avec AlexNet (Krizhevsky *et al*. **2012**). Cet article cite également les Transformers (Vaswani *et al.* 2017). Ces derniers ont été développés initialement pour le traitement automatique du langage naturel, appelés communément NLP (Natural Language Processing). Ils sont rapidement devenus le modèle de choix dans le domaine de NLP. Récemment, l'architecture des Transformers a été utilisée avec succès dans la classification d'images par Dosovitskiy *et al.* (*An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, octobre 2020). De tels transformers sont appelés Vision Transformers (ViT).

Travaux de Dosovitskiy *et al.*

La principale innovation décrite dans cet article est le découpage de l'image en petits patchs : ainsi, au lieu de considérer le pixel comme le coeur d'une image pour démêler les motifs, c'est le patch qui est considéré.

La figure ci-dessous montre le principe d'un modèle ViT :

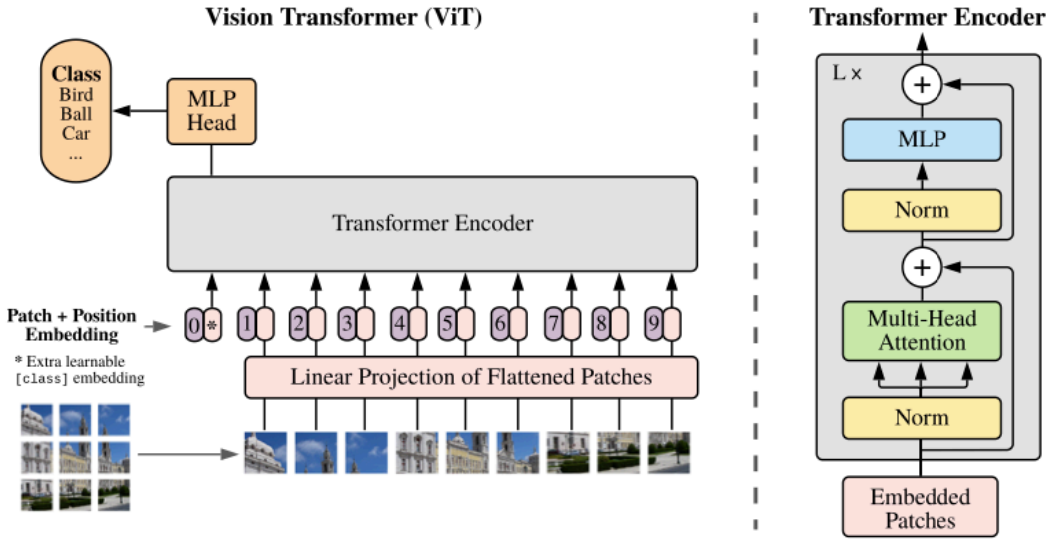


FIGURE 1 - DESCRIPTION D'UN MODÈLE VIT, DOSOVITSKIY *ET AL.*

L'image d'entrée est découpée en patchs qui sont ensuite aplatis, suivie d'une projection linéaire. La position de la projection est ensuite ajoutée à chaque patch, celle-ci est apprenable par le modèle. La séquence résultante de vecteurs est ensuite injectée dans le Transformer Encoder. Pour effectuer la classification, un « classification token » apprenable est ajouté à la

séquence de patchs. La sortie de l'encodeur est directement transmise à un réseau de neurones Feed Forward (MLP Multi Layers Perceptron) pour effectuer la classification.

Différents modèles de ViT sont décrits et testés (taille de patch 16 ou 32, nombre de couches de transformers 12, 24 ou 32). Les modèles sont entrainés sur ImageNet, ImageNet-21k et JFT-300M (ce dernier est un dataset privé de Google). Les auteurs ont montré que sur ce dernier dataset, les ViT les plus « gros » pouvaient surpasser les meilleurs CNN contemporains à cet article et avoir des coûts de pré-entrainement tout à fait corrects.

Objectif du projet

L'objectif de ce projet est donc de comparer une avancée du Deep Learning, à savoir les Vision Transformers, à un modèle de CNN sur la classification d'images de style cartoon. Dans ce contexte, nous utilisons des modèles pré-entrainés. Nous comparerons ainsi les résultats :

- avec un transfer learning appliqué à ces deux modèles
- puis en effectuant le fine tuning des deux modèles précédents.

Méthodologie

Outils utilisés

Le projet est rédigé sous forme d'un notebook exécuté sur *Google Colab*. Le langage de programmation choisi est *Python*. Pour la visualisation des images, nous utilisons entre autres la librairie *Pillow*. Les librairies usuelles de manipulation des données et de création de graphes sont également utilisées. Concernant les différents pré-traitements sur les données, le framework *TensorFlow* et son API *Keras* ont été utilisés.

Pour le modèle de CNN, nous choisissons MobileNetV2 (Sandler *et al.* 2018). Son architecture est disponible en Annexe 1. Nous avons utilisé ce CNN dans un précédent projet : il s'est avéré performant malgré sa « petite » taille. Ce modèle est adapté pour les solutions mobiles. Un modèle pré-entrainé sur ImageNet est disponible dans *Keras*.

Pour le modèle de Vision Transformer, le *GitHub* officiel[1] des travaux de Dosovitskiy *et al.* est accessible. Nous avons préféré utiliser l'implémentation sous *Keras*, disponible également sous *GitHub*[2], afin de rester dans un environnement *Keras*/*TensorFlow*. Nous avons donc installé la librairie *vit-keras* et chargé le modèle pré-entrainé vit-b32 : c'est un modèle à 12 couches de transformers (vit-b), utilisant des patchs de taille 32. En effet, d'après l'article *How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers* (Steiner *et al.* 2021), les auteurs conseillent une taille de patch de 32. Le modèle vit-b est un modèle « léger » parmi les ViT. Ce modèle a été pré-entrainé sur ImageNet-21K et fine tuné sur ImageNet.

Pour les matrices de confusion et les métriques associées, nous utilisons la librairie *scikit-learn.*

Données

Création du dataset

Nous n'avons pas trouvé de dataset pertinent à disposition contenant des images de style cartoon. En 2018, Yanqing Zhou *et al.* ont constitué *Toonset* mais malheureusement il n'est pas

[1] https://github.com/google-research/vision_transformer

[2] https://github.com/faustomorales/vit-keras

disponible. Nous nous sommes inspirés de leurs travaux pour constituer le dataset.

En suivant deux tutoriels[3,4], nous avons récupéré différentes images sur Google Images. Nous avons cherché 10 catégories d'images :

- **cat** (chat)
- **dog** (chien)
- **panda** (panda)
- **plane** (avion)
- **car** (voiture)
- **penguin** (pingouin)
- **tiger** (tigre)
- **bird** (oiseau)
- **flower** fleur)
- **tree** (arbre)

Nous gardons dans la suite du présent document les dénominations en anglais pour ces classes. Les sites internet dont sont issues ces images sont indiqués dans l'Annexe 2.

Pour chaque classe, nous collectons 100 images de sorte à avoir 1000 images au total. Par souci de propriétés intellectuelles, ce dataset n'est pas publié.

Nettoyage des données

Malgré un format commun .jpg., toutes les images ne sont pas encodées en RGB (Red Green Blue). En effet, certaines images sont de type RGBA (Red Green Blue Alpha, cette dernière composante indiquant la transparence), issu d'un précédent format .png. Nous devons toutes les encoder en RGB car, lorsque les images seront passées dans le modèle, elles doivent être de ce type. En leur appliquant un « background » blanc, 211 images sont ainsi transformées.

Visualisation des images

Voici un exemple d'images pour chaque classe :

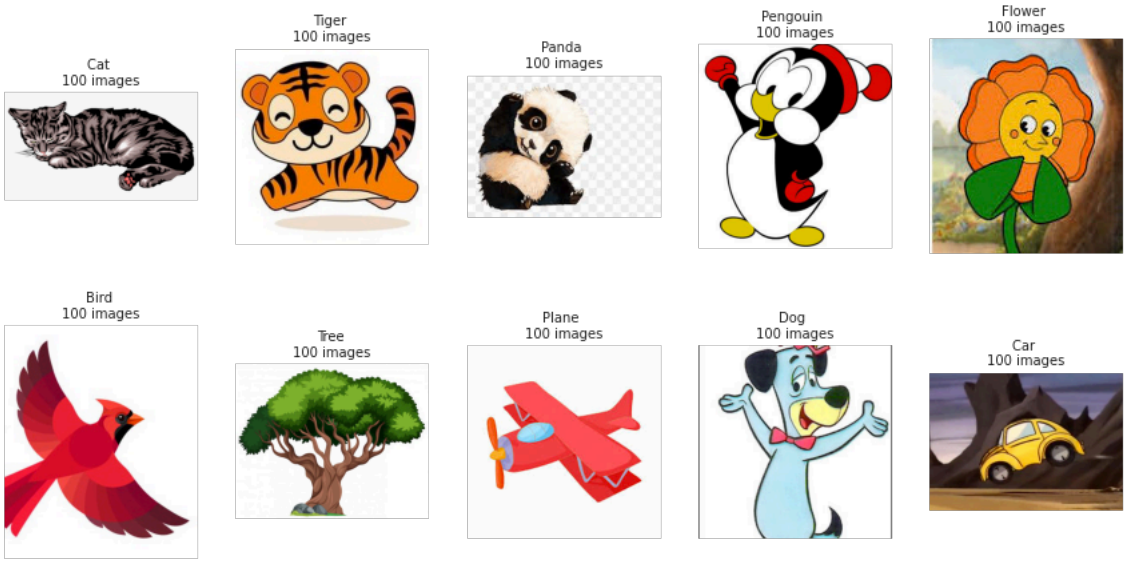


FIGURE 2 - EXEMPLE D'UNE IMAGE PAR CLASSE

Lors du traitement des images, il faudra les redimensionner car elles ne sont pas toutes de

[3] https://pyimagesearch.com/2017/12/04/how-to-create-a-deep-learning-dataset-using-google-images/

[4] https://medium.com/@bmrs2398/create-a-deep-learning-dataset-using-google-images-48654bd6ae96

la même taille. Par ailleurs, pour pouvoir être traitées par les modèles, la taille des images doit être de 224 x 224.

FIGURE 3 - EXEMPLE DE 4 IMAGES AU SEIN DES CLASSES BIRD ET PLANE

A l'aide d'un guide sur les Vision Transformers[5], nous pouvons visualiser une image de taille 224x224 découpée en patchs de taille 32 :

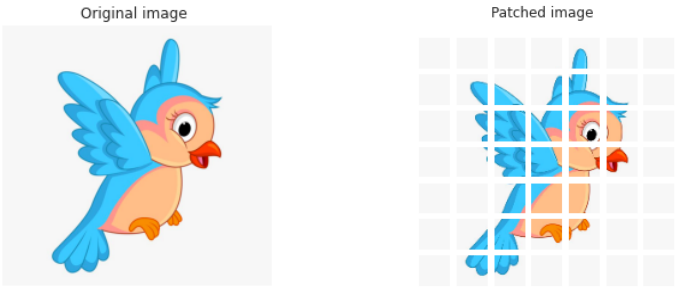


FIGURE 4 - EXEMPLE DE D'UNE IMAGE DÉCOUPÉE EN PATCHS DE TAILLE 32

Il y a 49 patchs sur cette image. Le ViT se servira de ces patchs pour son apprentissage.

Split des données

Nous répartissons aléatoirement les données dans trois répertoires :
- Training Set : 70% des données
- Validation Set : 15% des données
- Test Set : 15% des données

Le Test Set sera utilisé lors de l'évaluation des modèles et pour la comparaison des résultats.

Métriques utilisées

Dans un premier temps, nous comparons les modèles en utilisant les métriques suivantes :
- l'accuracy : taux de bonnes prédictions
- le loss (ou fonction coût) : « distance » entre la classe prédite et la classe réelle.

Dans un second temps, lors de l'analyse des résultats sur le Test Set, nous comparons également :
- la precision : taux de prédictions correctes parmi les prédictions positives (capacité du modèle à ne pas faire d'erreur lors d'une prédiction positive)
- le recall : taux de positifs détectés par le modèle (capacité du modèle à détecter l'ensemble des positifs)
- le score de f1 : moyenne harmonique de la precision et du recall

[5] https://analyticsindiamag.com/hands-on-guide-to-using-vision-transformer-for-image-classification/

Entrainement des modèles par Transfer Learning

Preprocessing des images

Comme indiqué précédemment, les images sont redimensionnées en 224x224 avant d'être traitées par les modèles. Nous intégrons également les preprocessings inhérents à chaque modèle qui consistent en un rescaling des pixels entre -1 et 1.

Data augmentation

Steiner *et al.* conseille d'utiliser la data augmentation pour l'entrainement des ViT. Cette méthode permet de fournir artificiellement plus d'images aux modèles pour leur apprentissage.

Nous appliquons la data augmentation avec ImageDataGenerator. En effet, avec cette classe de l'API *Keras*, les images sont augmentées en direct (« online ») lors de l'entraînement du réseau. Elles ne sont donc jamais sauvegardées en mémoire, et le réseau ne rencontre jamais deux fois la même image (car ces augmentations sont aléatoires). Cette méthode permet de s'assurer que le réseau voit des images variées à chaque epoch (cycle d'entraînement complet).

Nous appliquons la combinaison des effets suivants sur une image et les visualisons :

- effet miroir (horizontal flip)
- rotation (0 à 45°)
- décalage (haut/bas et/ou droite/gauche, de 0 à 20%)
- changement de luminosité (de 0.5 à 1.5)
- zoom (0.75 à 1.25).

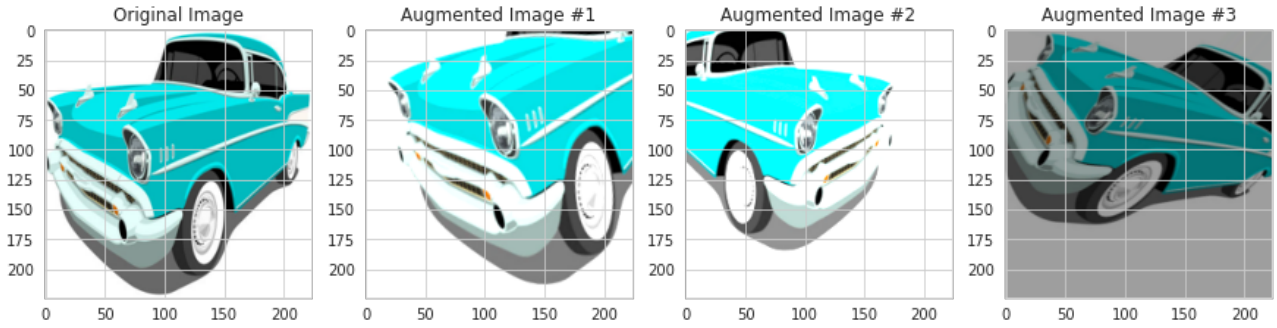


FIGURE 5 - EFFET DE LA DATA AUGMENTATION SUR UNE IMAGE. *Les 3 « augmented images » ont chacune reçu aléatoirement une combinaison d'effets modifiant l'image d'origine*

Constitution des modèles

Les modèles sont importés depuis les librairies adéquates. Dans un premier temps, nous « gelons » les couches des modèles et rajoutons une dernière couche Dense qui sert de classifieur. Pour le modèle MobileNetV2, nous ajoutons également une couche GlobalAverage2D en sortie des couches pré-entrainées. Ces modèles sont entrainés sur notre dataset.

Nous utilisons l'optimizer Adam avec un learning rate de 10^{-3} (valeur par défaut).

Nous entrainons sur 50 epochs, avec un early stopping arrêtant l'entrainement du modèle si la valeur de l'accuracy sur le jeu de validation n'a pas augmenté depuis 20 epochs.

Dans un second temps, nous effectuons un fine tuning des modèles : les couches « gelées » précédemment sont « dégelées » et nous ré-entrainons entièrement les modèles avec notre dataset.

Nous utilisons toujours l'optimizer Adam en baissant le learning rate à 10^{-5} pour éviter un overfitting trop rapide[6].

Nous entrainons jusqu'à 100 epochs (incluant les epochs de l'entrainement précédent), avec le même early stopping que précédemment.

[6] https://keras.io/guides/transfer_learning/

Nous obtenons ainsi quatre modèles à comparer :

- MobileNetV2
- Fine tuned MobileNetV2 : modèle précédent fine tuné
- ViT
- Fine tuned ViT : modèle précédent fine tuné.

Les modèles ont été entrainés en utilisant un environnement GPU disponible sous *Google Colab*.

Résultats

Comparaison des valeurs d'accuracy et de loss

Dans un premier temps, nous nous intéressons à ces deux métriques. Les courbes d'apprentissage des modèles sont disponibles dans l'Annexe 3.

Nous obtenons les résultats suivants concernant l'accuracy sur les jeux de données :

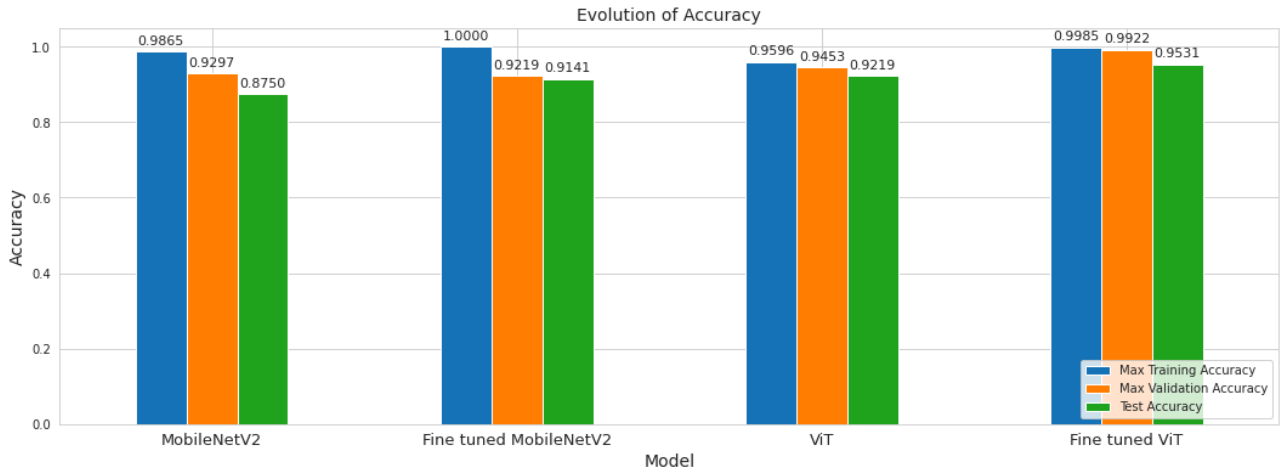


FIGURE 6 - EVOLUTION DES SCORES MAXIMUM D'ACCURACY SUR LES JEUX DE TRAIN ET VALIDATION, ET DE L'ACCURACY SUR LE JEU TEST SELON LE MODÈLE

Pour MobileNetV2, le fine tuning a amélioré la valeur maximale d'accuracy sur le jeu de train. Par contre nous observons une légère baisse sur le jeu de validation. Néanmoins, sur le jeu de test, l'accuracy passe de 0.8750 à 0.9141, soit une amélioration de presque 0.04 point.

Pour ViT, le fine tuning a amélioré les valeurs maximales d'accuracy sur les jeux de train et de validation. Sur le jeu de test, l'accuracy est également meilleure après le fine tuning : elle gagne plus de 0.03 point, passant de 0.9219 à 0.9531.

Dans les deux cas (modèles sans fine tuning et avec fine tuning), le ViT a de meilleures valeurs d'accuracy sur tous les jeux.

Nous obtenons les résultats suivants concernant le loss sur les jeux de données :

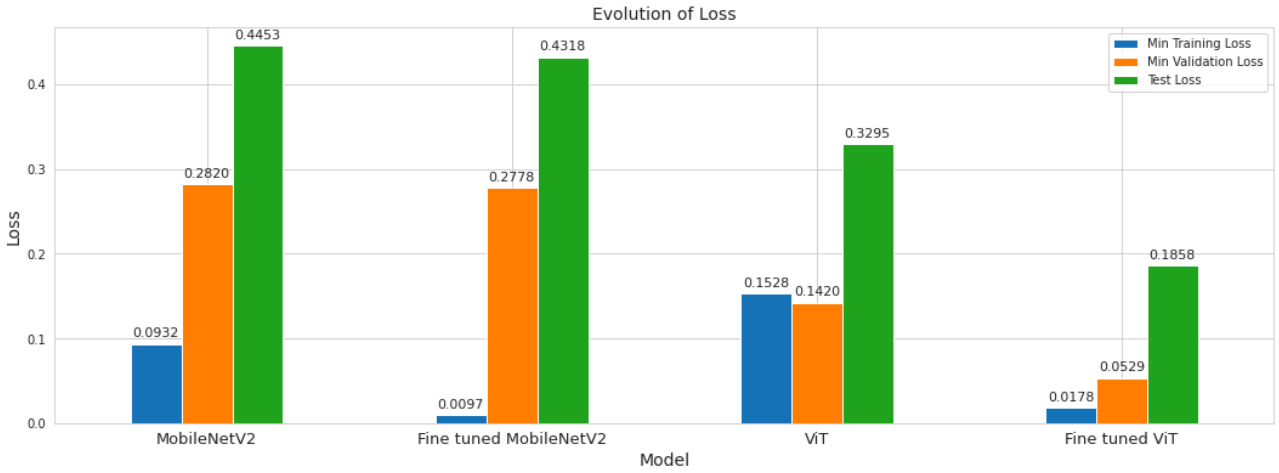


FIGURE 7 - EVOLUTION DES SCORES MINIMUM DU LOSS SUR LES JEUX DE TRAIN ET VALIDATION, ET DU LOSS SUR LE JEU TEST SELON LE MODÈLE

Pour MobileNetV2, le fine tuning a surtout amélioré la valeur minimale du loss sur le jeu de train. Par contre, sur les autres jeux nous observons une très légère amélioration. Sur le jeu de test, le loss passe de 0.4453 à 0.4318, soit une amélioration de seulement 3%.

Pour ViT, le fine tuning a amélioré les valeurs minimales du loss sur les jeux de train et de validation. Sur le jeu de test, le loss est également meilleur après le fine tuning avec une diminution de plus de 40%, passant de 0.3295 à 0.1858.

Dans les deux cas (modèles sans fine tuning et avec fine tuning), le ViT a de meilleures valeurs de loss sur les jeux, excepté sur le jeu de train.

Le tableau suivant indique les nombres d'epochs sur lesquels les modèles ont été entrainés, ainsi que le temps d'entraînement et la taille des modèles :

	MobileNetV2	**Fine tuned MobileNetV2**	**ViT**	**Fine tuned ViT**
Nombre d'epochs	50	51	33	36
Total epochs	101		69	
Temps	28 min	29 min	26 min	23 min
Total Temps	57 min		49 min	
Taille	9.1 Mo	26.4 Mo	333.9 Mo	1001.6 Mo

TABLEAU 1 - COMPARAISON DU NOMBRE D'EPOCHS, DU TEMPS D'ENTRAINEMENT ET DE LA TAILLE DES MODÈLES

Les modèles MobileNetV2 ont été entrainés sur plus d'epochs, d'où un temps d'entrainement plus grand. Néanmoins, rapporté au nombre d'epochs, ce temps est plus élevé pour les modèles ViT. Nous constatons que les modèles ViT sont « lourds » (presque 35 fois plus que les MobileNetV2), ce qui n'est pas optimal pour des applications mobiles.

Comparaison des prédictions

Nous récupérons pour chaque modèle les prédictions de chaque image du jeu de test. Nous construisons les classification reports des quatre modèles. Les voici :

FIGURE 8 - CLASSIFICATION REPORT DES 4 MODÈLES

Nous constatons que les modèles, surtout MobileNetV2, ont plus de mal à catégoriser les classes « Dog » et « Cat ». D'après les matrices de confusion des différents modèles (disponibles dans l'Annexe 4), nous voyons que ces deux classes sont souvent confondues entre elles, ainsi que la classe « Tiger ». Pour celle-ci, le fine tuning n'améliore pas le recall mais la precision et le score de f1.

Le modèle MobileNetV2 a amélioré le score de precision pour la classe « Dog » au détriment du recall, alors que pour la classe « Cat » c'est l'inverse. Par ailleurs, la classe « Plane » avait un score parfait avant fine tuning de MobileNetV2 et le perd après. Après fine tuning, MobileNetV2 a deux classes avec des scores parfaits, les classes « Car » et « Panda », soit 20% des classes.

Pour le modèle ViT, nous constatons une amélioration systématique des scores, excepté pour les classes « Bird » et « Pengouin » dont les scores ne varient pas. Peut être est-ce du au fait que les pingouins sont en fait des oiseaux…

Dès le début, ViT a trois classes avec des scores parfaits : « Car » , « Flower » et « Tree ». Après fine tuning, les scores parfaits sont conservés pour ces classes, et d'autres classes acquièrent des scores parfaits : « Panda », « Plane ». Nous avons donc des scores parfaits sur 50% des classes.

D'après les matrices de confusion, le ViT s'améliore avec le fine tuning sans entrainer de « changements » dans les mauvaises prédictions, contrairement au MobileNetV2. En effet, pour ce dernier après fine tuning, une image est prédite « Plane » au lieu de « Dog » alors qu'avant fine tuning il n'y avait pas d'erreur pour la classe « Plane ». De même pour la classe « Bird », aucune image n'était prédite « Cat » avant fine tuning et après, deux prédictions « Cat » apparaissent (dont une peut être issue d'un « Tiger » corrigé, puisque deux images « Bird » sont prédites « Tiger » avant fine tuning, puis une seule est prédit « Tiger »).

En Annexe 5, se trouve un exemple de prédictions des modèles pour une image prise au hasard parmi chaque classe dans le jeu de test.

Conclusion

Nous avons ainsi comparé un modèle de CNN et un modèle de ViT (initialement pré-entrainés) sur la classification d'images de style cartoon.

Après fine tuning des modèles, il s'avère que le modèle ViT a montré de meilleures performances en terme de classification. Les métriques sont supérieures à 95%. Néanmoins, le modèle ViT a été pré-entrainé sur un dataset de plus grande taille que MobileNetV2, ce qui pourrait expliquer, entre autre, la différence de performance.

Par ailleurs, le modèle ViT est presque 35 fois plus lourd que MobileNetV2, ce qui peut être un frein pour son déploiement en production.

Concernant les points d'amélioration de ce projet, un dataset de plus grande taille peut être à envisager pour l'entrainement des modèles.

Nous avons utilisé comme CNN MobileNetV2, modèle « léger », rapide et performant. Néanmoins ce n'est pas le CNN le plus performant actuellement. Un réseau tel que ResNet ou EfficientNet pourrait être utilisé. Ces réseaux sont par contre plus lourd, mais de ce point de vue ils se rapprochent des ViT.

Des réseaux hybrides CNN-ViT ont émergé. Récemment, Mehta *et al.* (octobre 2021) ont développé MobileViT, un modèle qui combine des blocs de type MobileNetV2 avec des ViT. Ce nouveau modèle est de poids comparable à MobileNetV2 et offre de meilleures performances. Néanmoins, les auteurs indiquent que ce modèle est plus lent que MobileNetV2 et qu'une utilisation sur appareils mobiles n'est pas optimale.

Comme pour les CNN, l'évolution des modèles ViT est rapide. L'avenir nous dira s'ils vont devenir à leur tour les modèles de référence pour la classification d'images.

Références

Chen,L.;Li,S.;Bai,Q.; Yang, J.; Jiang, S.; Miao, Y. Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sens*. **2021**, *13*, 4712.
https://doi.org/10.3390/rs13224712

Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Inf. Process*. *Syst.* **2012**, *25*, 1097–1105
https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Vaswani,A.;Shazeer,N.;Parmar,N.;Uszkoreit,J.;Jones,L.;Gomez,A.N.;Kaiser,L.;Polosukhin,I. Attention Is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, **December 2017**; NIPS'17. Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6000–6010.
https://arxiv.org/pdf/1706.03762.pdf

Dosovitskiy,A.;Beyer,L.;Kolesnikov,A.;Weissenborn,D.;Zhai,X.;Unterthiner,T.;Dehghani,M.;Minderer, M.;Heigold,G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **October 2020**, arXiv:2010.11929.
https://arxiv.org/pdf/2010.11929.pdf

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 18–23 **June 2018**; pp. 4510–4520
https://arxiv.org/pdf/1801.04381.pdf

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, Lucas Beyer. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. *arXiv* **2021**, arXiv:2106.10270
https://arxiv.org/pdf/2106.10270.pdf

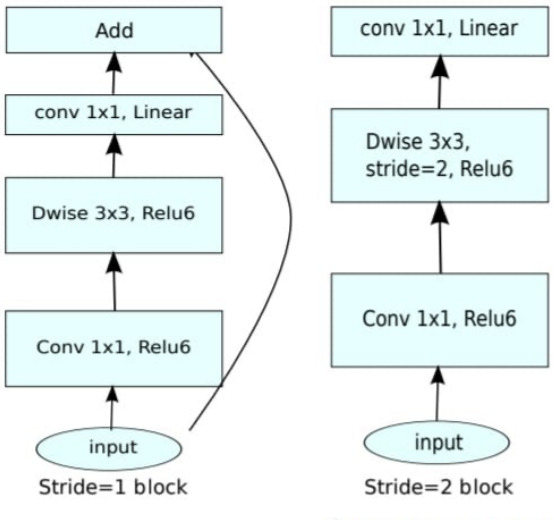
Yanqing Zhou, Yongxu Jin, Anqi Luo, Szeyu Chan, Xiangyun Xiao, and Xubo Yang. **2018**.
ToonNet: A cartoon image dataset and a DNN-based semantic classification system. In *International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI '18), December 2–3, 2018, Hachioji, Japan*. ACM, New York, NY, USA, 8 pages.
https://web.stanford.edu/~yxjin/pdf/toonnet.pdf

Sachin Mehta, Mohammad Rastegari. MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer. *arXiv* **October 2021**, arXiv:2110.02178v2
https://arxiv.org/pdf/2110.02178.pdf

ANNEXES

Annexe 1 : Architecture MobileNetV2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	



ARCHITECTURE MOBILENETV2, SANDLER *ET AL.*

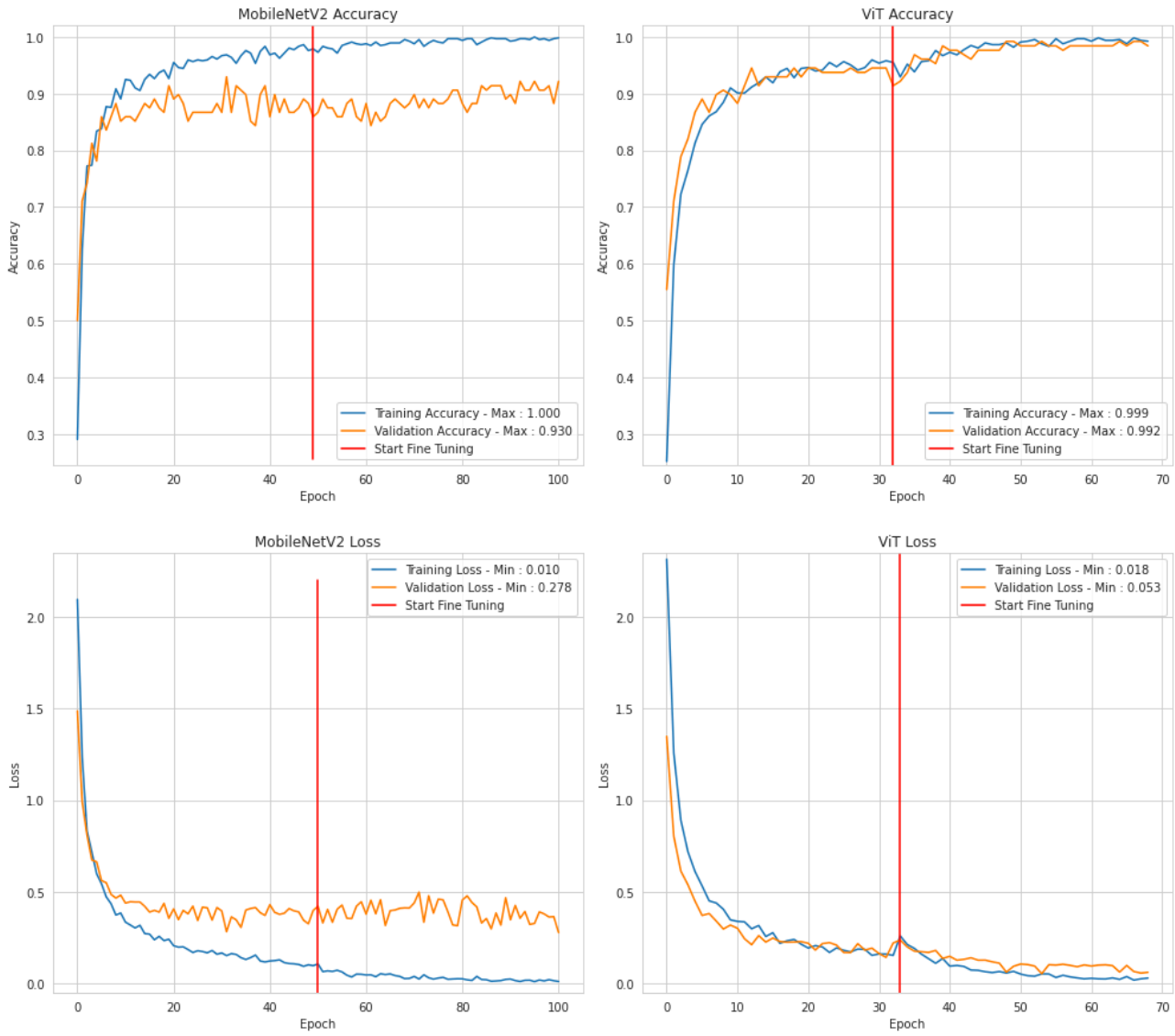
La spécificité de MobileNetV2 est la présence de « bottlenecks » et de blocs résiduels inversés : 1x1 (expansion) → 3x3 → 1×1 (compression). Cette dernière couche 1x1 est connectée avec une activation linéaire (au lieu d'une activation ReLU) pour éviter la perte d'informations

Annexe 2 : Liste des sites internet dont sont issues les images récupérées via Google Image

https://www.freevector.com
http://clipart-library.com
https://img.favpng.com/
https://cdn.pixabay.com/
https://www.kindpng.com
https://www.pngitem.com
https://files.123freevectors.com
https://i.pinimg.com
https://media.istockphoto.com/
https://png.pngtree.com/
https://img.freepik.com
https://www.pngfind.com
https://pngset.com
https://365psd.com/
https://www.nicepng.com/
https://www.pikpng.com/
https://www.designersnexus.com
https://static.vecteezy.com/
https://cdn2.vectorstock.com/
https://www.pinclipart.com
https://thumbs.dreamstime.com
https://i2.wp.com/
https://c8.alamy.com
https://vectorportal.com
https://www.freepatternsarea.com
https://www.ameede.com
https://www.vhv.rs
https://png.vector.me
https://images.all-free-download.com
http://www.vectors1.com
https://www.seekpng.com/
https://images.vexels.com
https://freedesignfile.com
https://previews.123rf.com
https://toppng.com
https://www.welovesolo.com
https://cdn.tutsplus.com
https://images.unsplash.com
https://freesvg.org
https://freepikpsd.com/
https://cdn.w600.comps.canstockphoto.com/
https://icon2.cleanpng.com/
https://st4.depositphotos.com/
https://illustoon.com/
https://www.how-to-draw-funny-cartoons.com/
https://d2gg9evh47fn9z.cloudfront.net/
https://camo.envatousercontent.com/
https://d31xsmoz1lk3y3.cloudfront.net/
https://media.gettyimages.com/
https://www.gannett-cdn.com/
https://wallpaperaccess.com/
https://t3.ftcdn.net/
https://sexadodeaves.com/
https://s3.envato.com/
https://decadescdn.decades.com/
https://en.pimg.jp/
https://
graphicriver.img.customer.envatousercontent.com
https://cdn4.vectorstock.com/
https://www.jamiesale-cartoonist.com/
https://lookaside.fbsbx.com/
https://www.rd.com/
https://static.boredpanda.com
https://artprojectsforkids.org/
http://www.clker.com/
https://glasbergen.b-cdn.net
https://lowres.cartooncollections.com/
https://images.saymedia-content.com
https://w7.pngwing.com/
https://assets.findcatnames.com/
https://s3.amazonaws.com/
https://www.thehappycatsite.com/
https://i.ytimg.com/
https://upload.wikimedia.org/
https://www.thepurringtonpost.com/
https://assets.dragoart.com/
https://www.iizcat.com/
https://mymodernmet.com/
https://media.newyorker.com/
https://videohive.img.customer.envatousercontent.com
https://www.drawingforall.net/
https://www.chrismadden.co.uk/
https://www.wikihow.com/
https://s36537.pcdn.co/
https://us.123rf.com/
https://cdn1.vectorstock.com
https://cdn5.vectorstock.com
https://petguidereviews.com
https://www.urbandognyc.com
https://thehappypuppysite.com
https://cdn.pastemagazine.com/
https://www.publicdomainpictures.net
https://bunnyears.com/
https://besttoppers.com/
http://4.bp.blogspot.com/
https://images-wixmp-
ed30a86b8c4ca887773594c2.wixmp.com/
https://assets.whichcar.com.au
https://mpng.subpng.com
https://drawinghowtos.com
https://cliparting.com/
https://cutewallpaper.org/
https://easydrawingguides.com/
http://drawdoo.com/
https://s1.cdn.autoevolution.com/
https://media.gq-magazine.co.uk/
https://resources.stuff.co.nz/
https://img.lovepik.com/
https://www.cuteeasydrawings.com/
https://osoq.com/

http://www.onallcylinders.com/
https://hips.hearstapps.com/
https://library.kissclipart.com/
http://www.aei.org
http://travelinlibrarian.info/
http://cdn.shopify.com/
https://atlas-content-cdn.pixelsquid.com/
https://friendlystock.com/
https://img.joomcdn.net/
https://images.fineartamerica.com/
https://t4.ftcdn.net/
https://thenewswheel.com/
https://buzzdrives.com/
https://www.crushpixel.com
http://wheelsca.s3.amazonaws.com/
https://cdn.quotesgram.com/
https://p4.wallpaperbetter.com/
https://www.chevronwithtechron.com/
https://e7.pngegg.com
http://www.kartoons.co.uk/
https://www.saturdayeveningpost.com/
https://images-na.ssl-images-amazon.com
https://imgix.bustle.com/
https://www.pngkey.com/
https://grid.gograph.com/
https://w0.peakpx.com/
https://www.cartoonsmag.com/
https://coloringbay.com
https://i.graphicmama.com/
https://spaces-cdn.clipsafari.com
https://rlv.zcache.com
https://www.teahub.io
https://img.myloview.com
https://imgix.ranker.com
https://cdna.artstation.com
http://www.supercoloring.com
https://images.assetsdelivery.com
https://www.drawingtutorials101.com
https://ichef.bbci.co.uk
https://media.baamboozle.com
https://ae01.alicdn.com
https://www.askideas.com
https://i.kym-cdn.com
https://s2.dmcdn.net
https://esquilo.io
https://www.businessballs.com
https://bestanimations.com

Annexe 3 : Courbes d'apprentissage des modèles

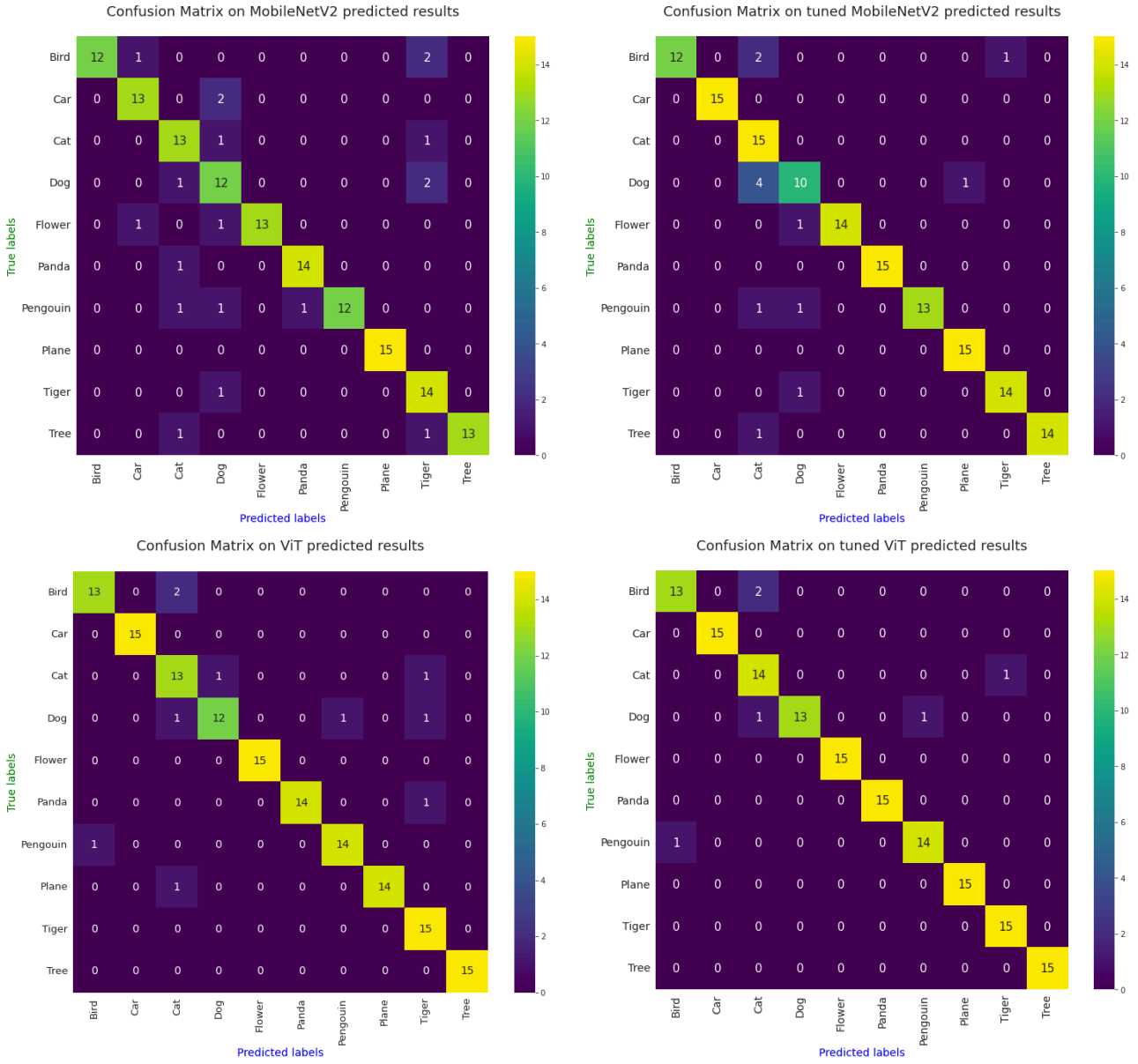


EVOLUTION DE L'ACCURACY ET DU LOSS AU COURS DE L'APPRENTISSAGE DES MODÈLES MOBILENETV2 ET VIT, *la ligne verticale rouge indique le début du fine tuning*

Le début du fine tuning se caractérise par une petite détérioration de l'accuracy et du loss, notamment sur le jeu de validation.

MobileNetV2 a tendance à overfitter sur le jeu de train, alors que pour le ViT les scores sur le jeu de validation « suivent » ceux sur le jeu de train, avec néanmoins un écart plus important qui apparait pour le loss.

Annexe 4 : Matrices de confusion des modèles sur le jeu de test

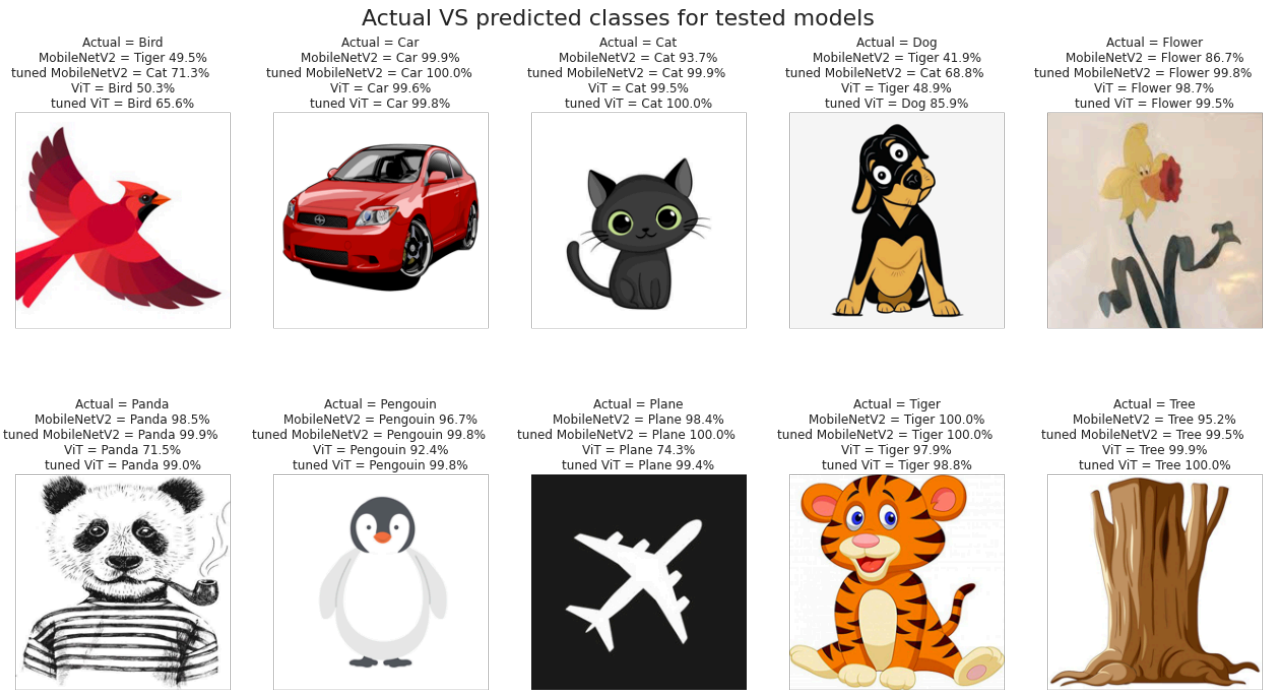


MATRICES DE CONFUSION DES MODÈLES MOBILENETV2 ET VIT AVANT ET APRÈS FINE TUNING

La diagonale indique la concordance entre la classe réelle de l'image et la classe prédite par le modèle. Plus les valeurs sur la diagonale sont élevées, meilleur est le modèle.

Le modèle ViT après fine tuning présente les meilleurs résultats.

Annexe 5 : Exemple de prédictions des modèles pour une image prise au hasard parmi chaque classe dans le jeu de test



COMPARAISON ENTRE LA CLASSE RÉELLE D'UNE IMAGE ET LES CLASSES PRÉDITES PAR LES MODÈLES, *les pourcentages indiquent la probabilité d'appartenir à la classe prédite*

Globalement les prédictions sont correctes. Pour les classes « Bird » et « Dog », MobileNetV2 ne prédit pas la bonne classe, même après fine tuning.

Quand les prédictions sont correctes, le fine tuning améliore les scores de probabilité. Certaines images ont un score « perfect » de prédiction de 100%.