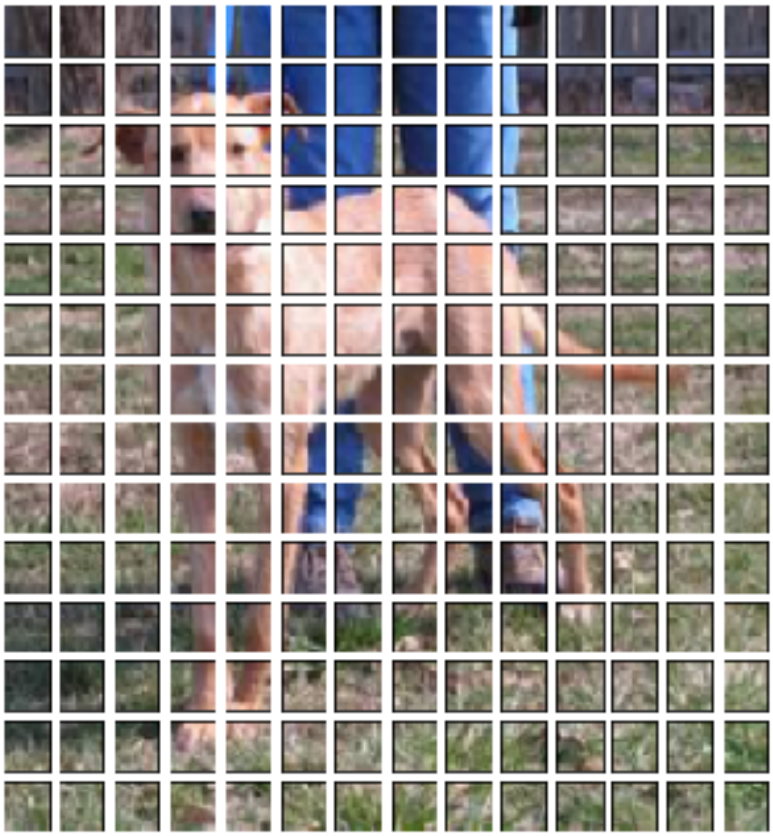
Classification d'image avec Vision Transformers

Rapport - Projet n°7 : « Développez une preuve de concept »

Luke Duthoit - Juillet-Août 2022



(Illustration trouvée sur le DIVA's BLOG de @DIVYAKAMAT)

Introduction

Ce rapport récapitule l'ensemble du travail accompli en deux semaines, dans le cadre du projet n°7 du parcours étudiant « Ingénieur *Machine Learning* » proposé par le site de formation en ligne OpenClassrooms.

Ayant eu des difficultés dans un 1er temps à trouver un article à la fois récent, intéressant, et pouvant être retranscrit sous forme de code relativement rapidement, j'ai mis ce projet en pause durant deux semaines et demi. Le temps écoulé lors de cette interruption a été sagement mis à profit pour, d'une part, réaliser/soutenir/valider le projet n°8, d'autre part, trouver « l'inspiration » pour le projet n°7. Le temps « réel » passé sur le travail présenté au cours de ce rapport est donc en réalité d'une semaine, ce qui est le délai de travail nécessaire estimé par OpenClassrooms, mais qui a pris une tournure d'autant plus urgente que cette semaine de travail intense a eu lieu lors de mes 11 derniers jours de financement de formation. En conséquence, j'ai dû faire des choix d'efficacité et de rapidité au détriment de l'exhaustivité des solutions envisagées.
Le travail accompli est récapitulé sous la forme de quatre livrables (dont le présent rapport), conformément au cahier des charges du projet. En particulier, le deuxième livrable, qui concerne le code, est constitué lui même de deux notebooks :

- **A_code_vit_01.ipynb**, qui concerne la nouvelle méthode mise en oeuvre,
- **A_code_baseline_02.ipynb**, qui concerne la *baseline*, ainsi que la comparaison des performances de la *baseline* avec celles de la nouvelle méthode.

Les deux notebooks peuvent être exécutés indépendamment l'un de l'autre. J'en ai écrits deux plutôt qu'un seul afin de pouvoir me concentrer sur l'implémentation des lignes relatives à l'une des deux méthode (la nouvelle ou l'ancienne), ainsi qu'à cause d'un manque de mémoire vive de mon ordinateur, qui avait du mal à exécuter un seul gros *notebook* nécessitant les ressources cumulées des deux *notebooks* individuels.
L'ensemble des lignes de code/figures/fichiers de données associées à ces descriptions sont accessibles sous forme de *notebook* Jupyter sur mon *drive* Google (*https://drive.google.com/drive/folders/1vIqD5s44s1_awo_P7WHgOt-b2XSl8Gj7*). Cela présente l'avantage, de pouvoir visualiser l'ensemble des étapes successives de rédaction des *notebooks*. Pour y avoir accès, n'hésitez pas à m'écrire à *luke.duthoit@gmail.com*.

I - Thématique

I.A - Présentation de nos ressources principales

J'ai basé mon travail sur la lecture de l'article scientifique *An Image Is Worth 16x16 Words : Transformers For Image Recognition At Scale*[1] (soumis à publication en octobre 2020 et publié

[1]A. DOSOVITSKIY *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. ArXiv: 2010.11929 (2020)

en tant que *conference paper* lors de l'édition de juin 2021 de l'*International Conference on Learning Representations*).
Cet article étant très technique pour un novice tel que moi, j'ai complété sa lecture (pour avoir une vision un peu plus globale et accessible des concepts qui y étaient présentés) en m'appuyant notamment sur des billets de blogs spécialisés : *How the Vision Transformer (ViT) works in 10 minutes:[…]*[2] et *Fine-Tune ViT for Image Classification with 🤗 Transformers*[3].
J'ai également repris le code développé lors du tutoriel *Image Classification with HuggingFace Transformers and Keras*[4], en l'appliquant sur un jeu de données différent de ceux présentés dans ce tutoriel ou dans les billets de blogs.
D'autres ressources secondaires, et consultées après le début de la rédaction du code, seront éventuellement mentionnées au cours de ce rapport.

I.B - Thématique effective

Cet article (et ces billets de blog) présentent **Vision Transformer** (noté *ViT* par la suite), un modèle servant à faire de la classification/reconnaissance d'images. Son originalité provient de son architecture, qui n'est pas inspirée de celle des réseaux de neurones convolutifs (en anglais *CNN*), qui sont pourtant très populaires en reconnaissance d'image. Au contraire, l'architecture de *ViT* s'inspire de celle de modèles appelés transformateurs (en anglais *Transformers*), qui sont des modèles très populaires en analyse de séquences de données textuelles. Ils ont la particularité de ne pas reposer sur des réseaux de neurones récurrents (en anglais *RNN*), mais de s'appuyer UNIQUEMENT[5] sur des mécanismes d'attention et d'auto-attention dîtes « multi-têtes » (en anglais *multi-head attention*).
C'est cette idée surprenante qu'on puisse prédire une classe d'image grâce à un algorithme qui appartienne à la même famille que celui de BERT qui m'a donné envie d'explorer ce sujet pour ce projet. En effet, j'ai eu énormément de difficultés sur les projets n°5 (« Catégorisez automatiquement des questions ») et n°6 (« Classez des images à l'aide d'algorithme de deep learning »), mais je les ai trouvés très intéressants. J'ai donc voulu en apprendre plus au sujet d'un nouveau genre de modèle qui si situerait à la frontière entre ces deux domaines.

I.C - ViT : méthode présentée dans l'article

[2] N. ADALOGLOU, AI Summer Blog (2021-01), *https://theaisummer.com/vision-transformer/*.

[3] N. RAW, HuggingFace Blog (2022-02), https://huggingface.co/blog/fine-tune-vit.

[4] P. SCHMID (2022-01), *https://www.philschmid.de/image-classification-huggingface-transformers-keras*.

[5] A. VASWANI et al. Attention is all you need. ArXiv:1706.03762 (2017), *https://arxiv.org/abs/1706.03762?amp=1*.

L'architecture de ViT est représentée dans la figure ci-dessous [Fig1], qui est issue telle quelle de l'article de A. DOSOVITSKIY *et al.* (2020)

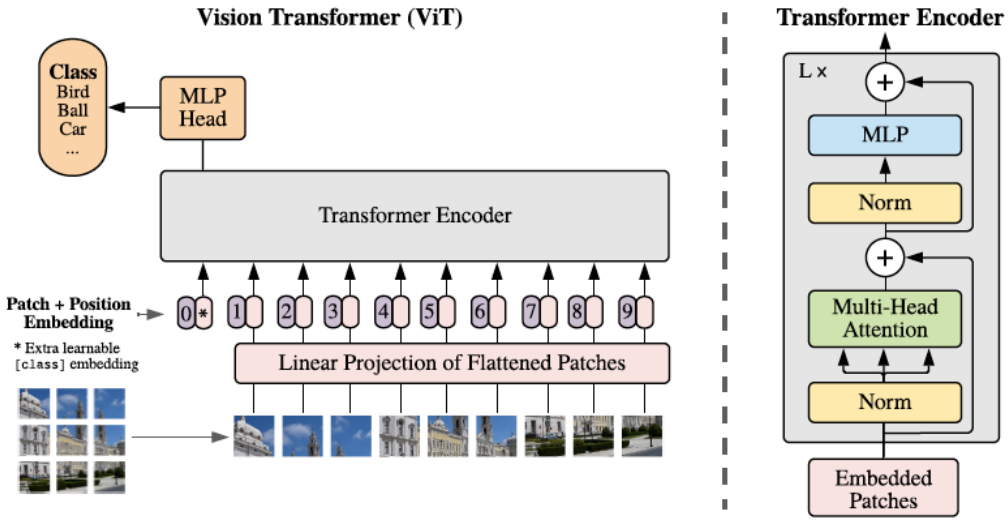


Fig1. Architecture de ViT, issue de l'article de A. Dosovitskiy et al. (2020)

Pour la résumer rapidement : chaque image mise en entrée est découpée en une série de sous-images (*patchs*), qui subissent une projection similaire à un plongement de mots (*word embedding*) au sein d'algorithme de traitement de données textuelles. On joint à ces vecteurs (*patchs embedding*) la représentation des position des patchs (*patchs embedding*) au sein de l'image initiale, afin de pouvoir reconstituer la séquence de sous-image. Les données ainsi formée sont envoyées en entrée d'un encodeur de transformateur (transformer encoder) où elles subissent des mécanismes d'attention multi-têtes sur plusieurs boucles. En sortie de l'encodeur, elles sont enfin envoyées en entrée d'une dernière couche dense qui s'occupe de prédire l'appartenance de l'image à l'une des classes.

I.D - Etat de l'art.

Les performances de *ViT*, après essais sur de nombreux *benchmarks*, sont très compétitives, et finissent par atteindre le même niveau que celle des *CNN*. Une telle annonce a déclenché une controverse pour savoir dans quelle mesure ce résultat était fiable, et quelles étaient les différences profondes de fonctionnement et d'analyse d'image entre ces deux modèles[6]. L'utilisation de ViT et des modèles inspirés des transformateurs est en train d'être abondamment testée dans des domaines telles que la classification d'image, la *visiolinguistic*[7], la détection d'objets, etc.

[6] M. RAGHU et al., Do Vision Transformers See Like Convolutional Neural Networks? arXiv:2108.08810 (2022)

[7] A. Shin et al., T. Perspectives and Prospects on Transformer Architecture for Cross-Modal Tasks with Language and Vision. Int J Comput Vis 130, 435–454 (2022)

II - Jeu de données

II.A - Nécessité d'un jeu de données originales

Comme je l'ai déjà écrit plus haut (section I.A), j'ai repris une partie importante d'un code développé lors d'un tutoriel en ligne par l'internaute @PhilSchmid. Ce code expliquait en particulier comment combiner les outils issues des librairies python *transformers* (pour réaliser l'extraction de *features* grâce à un modèle pré-entraîné), *tensorflow.keras* (pour faire l'apprentissage de la classification) et *datasets* (pour faire le lien entre les deux librairies précédentes). La tâche de classification était réalisée - sans *baseline* - sur la base de données ouverte « EuroSAT »[8], un ensemble d'images prises par le réseau de satellites d'observations « Sentinel-2 ».
Conformément au cahier des charges, j'ai donc pris une base de donnée alternative afin de faire une application différente de *ViT*. Ayant beaucoup apprécié le projet n°6 (« Classez des images à l'aide d'algorithme de deep learning »), il m'est apparu naturel de choisir de reprendre la base de données de ce projet, à savoir « Standford Dogs Dataset »[9] (elle même issue de la grande base de donnée ouverte « ImageNet »). C'est une base données constitué de plus de 20000 images, réparties entre 120 races de chiens différentes.

II.B - Réduction du jeu de données

Pour des raisons qui m'ont échappé jusqu'ici, le code ne peut fonctionner qu'avec un nombre d'image par catégories rigoureusement identique d'une catégorie à une autre. N'ayant pas eu le temps d'adapter le code pour empêcher cela, j'ai donc dû me résoudre à réduire la base de donnée à un nombre restreints de races de chiens contenant toutes le même nombre d'image. Deux bases de données ont été ainsi créées (et mise à l'essai) :

- **/150_dogs_per_race/**, qui contient les 13 races ayant chacune 150 images de chiens (c'est le nombre le plus fréquent d'images par catégories) ;
- **/150_dogs_per_race+/**, qui contient les 13 races précédentes ainsi que 15 races supplémentaires dont le nombre d'images par race a été ramené à 150.

On sélectionne la base avec laquelle on veut travailler en attribuant son nom à la variable **dossier_b2d_chiens** (première cellule, section 0.1, de chaque *notebook*).
Ces deux bases de données sont accessibles dans le dossier /P7/Notebook/ de mon drive. Les figures (/P7/Figures/) et les fichiers de résultats (/P7/Cache_fichiers/) produits par les calculs sur chacune d'entre elles portent leurs noms en guise de préfixe.

[8] EuroSAT, sur PapersWithCode, https://paperswithcode.com/dataset/eurosat

[9] http://vision.stanford.edu/aditya86/ImageNetDogs/

III - Méthode implémentée et *baseline*

III.A - Notre baseline

Ayant repris la base de donnée du projet n°6, il m'est paru naturel d'utiliser en guise de *baseline* le meilleur modèle crée à l'occasion de ce projet. En l'occurence, il s'agissait de faire du *transfert learning* à partir de l'architecture du *CNN* MobileNetV2[10]. La *baseline* utilise l'architecture de MobileNetV2 sans faire de *fine tuning*, c'est à dire :

- en conservant les poids appris lors du pré-entraînement sur ImageNet,
- en supprimant le bloc original de couches denses,
- en faisant la classification grâce à une couche dense inédite en sortie du bloc convolutif, avec autant de sorties qu'il y a de classes à prédire.

Seuls les poids de connexion entre cette couche dense et le bloc convolutif sont à apprendre, ce qui rend la phase d'apprentissage assez rapide (même lorsqu'on le faisait sur 120 races). Par ailleurs, les images subissent au préalable un pré-traitement spécifique, qui les passe en résolution 224x224 pixels^2, et décale leur spectre d'intensité de pixels de l'intervalle [0,255] à l'intervalle [-1,1] en centrant sa moyenne sur 0.
Cette *baseline* est codée dans le *notebook* **A_code_baseline_02.ipynb** (*cf* le livrable n°2 **Duthoit_Luke_2_code_072022**), où elle est ré-entraînée et re-évaluée sur les bases de données à 13 ou 28 races.

III.B - Nouvelle méthode implémentée

Notre nouvelle méthode, au contraire de la baseline, fonctionne en faisant du fine tuning, c'est à dire en ré-apprenant tous les poids de connexion d'une architecture issue du modèle *ViT* nommée « google/vit-base-patch16-224-in21k »[11]. Cette architecture découpe les images (remise à la résolution 224x224 pixels^2) en *patchs* de 16x16 pixels^2, et a été pré-entraînée sur la base de données « ImageNet-21k » (qui contient environ 10 fois plus d'images et 20 fois plus de catégories qu'ImageNet). Nous n'avons pas rajouté de couches/blocs de couches supplémentaires à cet architecture, son entraînement étant déjà bien plus long que celui de la *baseline*, et limitant donc dans le temps l'exhaustivité de notre travail de recherche (et pour cause, il y avait environ 80 fois plus de poids de connexions à apprendre que pour la *baseline*). Cette nouvelle méthode est codée dans le *notebook* **A_code_vit_01.ipynb** du livrable n°2.

[10] M. SANDLER *et al*. MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR, pp. 4510-4520 (2018), https://arxiv.org/abs/1801.04381

[11] Documentation sur HuggingFace : https://huggingface.co/google/vit-base-patch16-224-in21k

IV - Performances comparées

IV.A - Choix stratégiques

On divise les images en *train/validation/test sets* rigoureusement identiques entre ViT et la *baseline*.(en s'assurant qu'il y ait des exemplaires de chaque race au sein de chaque jeu). De même, à chaque fois, les images d'un même jeux sont regroupées en batch de 32 images. La phase entraînement/validation se fait en 5 *epochs*. On a choisit un nombre aussi « faible » car on savait que, pour notre *baseline* en tout cas, on aurait un bon score de précision en peu d'itérations sur si peu de races (notre *baseline* ayant conduit à près de 80% de précision sur le jeu de test à 120 races au bout de 15 itérations lors du projet n°6). La fonction coût (*loss*) est la fonction d'entropie croisée adaptée à de la classification à plus de deux classes. Nous avons fait le choix de tester deux métriques :

- le score de précision,
- le score de « top-3-précision », qui permet de compenser une mauvaise prédiction si l'étiquette réelle de l'image fait malgré tout partie des 3 catégories les plus probable à l'issue de la prédiction.

Nous avons également évalué les performances temporelles des deux modèles, en terme de temps pour calculer les différentes métriques lors :

- de l'évaluation sur le jeu de test,
- de la phase d'entraînement/validation, mais rapporté au nombre de poids de connexion à apprendre.

En effet, cette précaution a permis de mieux prendre en compte cette différence, qui fait qu'en nombre secondes écoulées, la *baseline* est beaucoup plus rapide que la nouvelle méthode (car elle a beaucoup moins à apprendre).

IV.B - Comparaison des performances

Dans les images ci-dessous, on présente les performances en terme de :

- d'évolution de la fonction coût (en haut à gauche) ;
- d'évolution du score de précision (en haut à droite) ;
- d'évolution du score de « top-3 précision » (en bas à gauche) ;
- du temps **absolu** d'évaluation en fonction de la durée **relative** de la phase d'entraînement/validation (en bas à droite).

La légende :

- en trait plein : sur jeu d'entraînement ;
- en trait pointillés : sur jeu de validation ;
- la croix finale (dernière *epoch*) : sur jeu de test.

Le code couleur :

- en bleu : *baseline* ;
- en rouge : ViT.

On représente enfin à gauche, les performances sur la base de données à 13 races [Fig2] (**150_dogs_per_race**), et à droite, sur 28 races [Fig3] (**150_dogs_per_race+**).

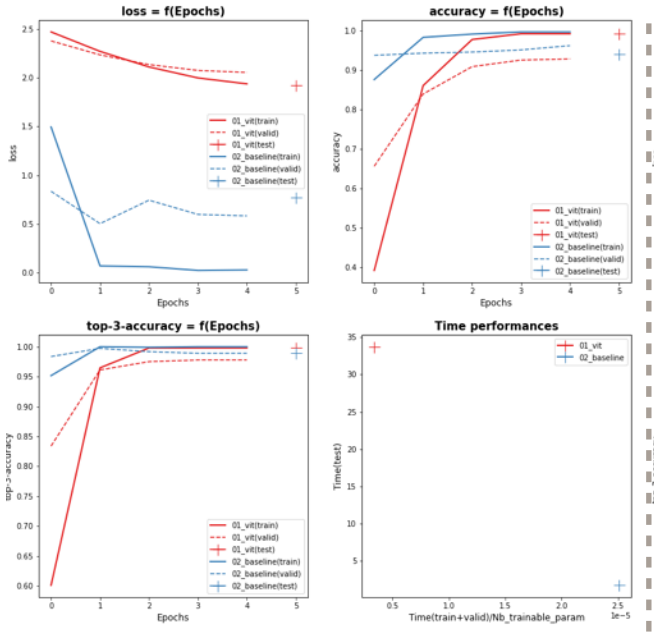


Fig2. Performances comparées sur la base de données à 13 races.

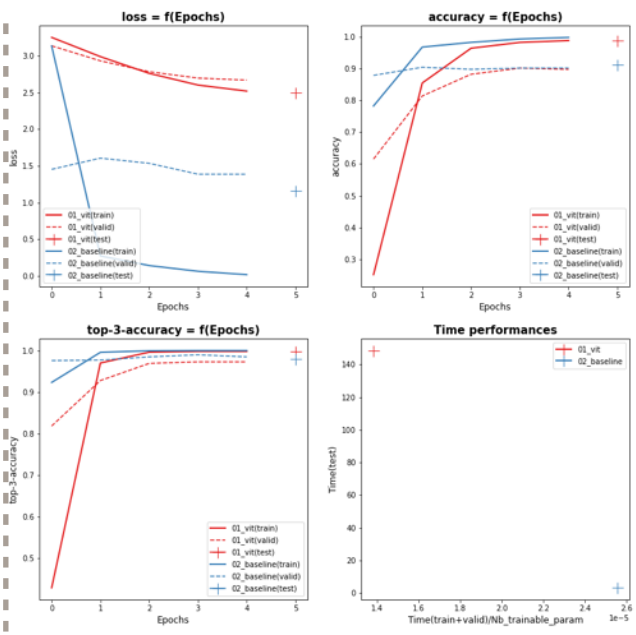


Fig3. Performances comparées sur la base de données à 28 races.

On y voit tout d'abord que les performances (hormis en durées temporelles bien sûr) sont sensiblement les mêmes sur les deux bases de données, ce qui est plutôt bon signe dans la mesure où on n'a pas doublé le nombre d'*epochs* lorsqu'on est passé de 13 à 28 races.
Pour les deux méthodes, on note un sur-apprentissage léger mais persistant entre jeu d'entraînement et de validation, ce qui relativise un peu les excellents scores sur les jeux de test.
La méthode basée sur l'architecture ViT conduit toujours à une minoration de la fonction coût moins importante que pour la *baseline*, mais ses performances en terme de métriques finissent :

- par être les mêmes sur le jeu d'entraînement ;
- convergent vers les mêmes valeurs sur le jeu de validation ;
- sont supérieures à celles de la *baseline* lors de l'évaluation sur le jeu de test, en particulier lorsqu'il s'agit de la précision simple !

IV.C - Introspection des capacités de prédictions de la nouvelle méthode

IV.C.1) Matrice de confusion

On trace ci-dessous [Fig4] l'équivalent (sur 28 races) de la matrice de confusion de la méthode issue de ViT, les lignes représentant les races possibles [les étiquettes réelles des images], et les colonnes représentant la probabilité de prédiction de chaque race en moyenne sur l'ensemble des images correspondant à une même étiquette réelle. Cette figure ne montre pas qu'initialement, les valeurs n'étaient pas toutes comprises entre 0 et 1 (certaines sont légèrement négatives, d'autres supérieures à 1). Cela vient du fait que, pour la couche dense finale implémentée dans le module python

TFViTForImageClassification (section I.C.2.a du *notebook* **A_code_vit_01.ipynb**), la fonction d'activation par défaut est la rampe linéaire [*ie* la fonction identité], et non la fonction softmax. Nous avons donc refait ce calcul afin de pouvoir directement interpréter ces coefficients comme des probabilités.

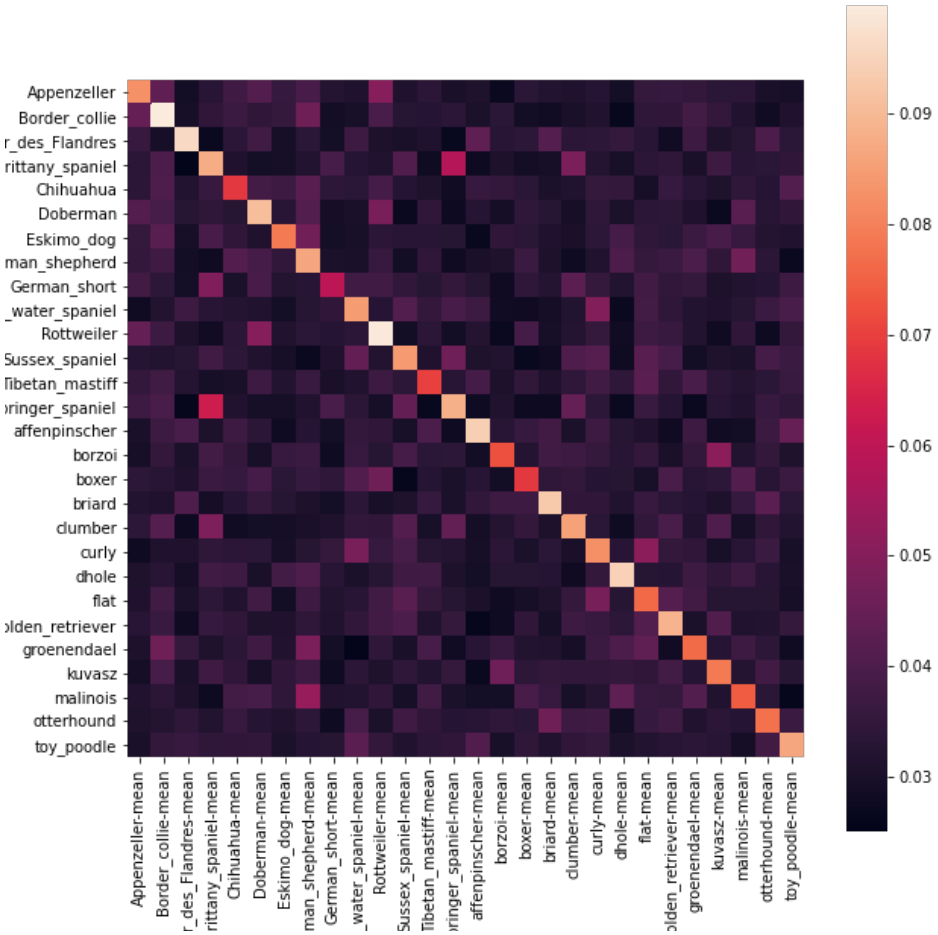


Fig4. Matrice de confusion des prédictions faites par le modèle issu de ViT sur la base de données à 28 races.

On constate que, pour chaque ligne, les coefficients diagonaux sont systématiquement supérieurs aux coefficients hors diagonales. Ce qui est logique en vue des très bons scores sur jeu de test : ce modèle prédit la bonne étiquette de façon très précise, en moyenne sur 28 races.
Néanmoins, on voit que toutes les étiquettes ne sont pas aussi bien prédites les unes que les autres, et que les coefficients hors diagonales sont loin d'être très inférieurs à ceux sur la diagonale. Il nous faut donc chercher à comprendre pourquoi.

IV.C.2) Analyse des races les moins bien prédites

On affiche donc dans la figure ci-dessous [Fig5] les des 5 races les moins bien prédites (1ère ligne, 1 race par colonne) accompagnées dans les lignes suivantes des 2 races avec lesquelles elles ont chacune le plus souvent confondues (les probabilités moyennes sont

affichées dans les titres de chaque image, chaque image étant choisie au hasard pour illustrer les races).

Fig5. Races les moins bien prédites, et races avec lesquelles elles sont confondues, par le modèle issu de ViT sur la base de données à 28 races.

On constate que les races confondues sont souvent proches morphologiquement parlant, ce qui expliquerait certaines confusions. Certaines confusion pourraient d'ailleurs être faites par des êtres humains qui s'y connaîtrait mal ou peu en classification canine, ce qui tend à relativiser les erreurs faites par cette nouvelle méthode.

IV.D - Conclusions

On a donc montré que l'architecture issue de ViT pouvait tout aussi bien prédire des races de chiens que la baseline issue de MobileNetV2 après entraînement sur autant d'itérations. Sa capacité de généralisation sur un nombre réduit de races est sensiblement égales, alors qu'il lui a été appliqué une stratégie de *fine tuning*, bien plus exigeante en ressources de calculs et en nombre de poids de connexions à ré-apprendre. C'est une expérience concluante sur 13 et 28 races, il faudrait à l'avenir chercher à l'étendre à l'ensemble des 120 races possibles, car les faibles différences de probabilité de prédictions entre les bonnes étiquettes et les mauvaise pourraient révéler là une faiblesse de ce système..