Formation DS-IML de marc.lefevre@noos.fr

Note technique PROJET-8

Petfinder.my
« pawpularity » contest

Avant-propos :

Ce document est la note technique de notre **projet 8** de formation, dont l'objet était une participation à une compétition **Kaggle**.

Dans un premier temps, nous nous sommes investis dans le concours **RSNA-MICCAI Brain Tumor Radiogenomic Classification**, avant, par manque de résultats probants, de devoir bifurquer sur une autre compétition. Comme il nous l'a été demandé, nous n'évoquerons globalement ici que cette seconde tentative, qui s'est avérée plus fructueuse en termes de résultat.

Sommaire :

1. Présentation du concours Petfinder.my

1.1. Kaggle

Pour commencer, quelques mots à propos du site internet **kaggle.com** qui héberge la compétition de machine learning dont il est question.

En premier lieu, c'est une mine de ressources pédagogiques autour de laquelle gravite une communauté d'utilisateurs du monde entier active et désireuse de partager ses expériences.

Mais c'est surtout par les concours qu'il héberge que le site a fait sa renommée.

Ces concours proviennent de diverses entités (sociétés, associations, organisations diverses…) et consistent chacun en une mission à accomplir sur un dataset donné avant une certaine deadline. Libre ensuite à chacun d'y participer quels que soient son niveau et ses ambitions.

En outre, Kaggle met à disposition de tout utilisateur un cloud et de la puissance de calcul pour travailler dans des conditions tout à fait respectables, le tout **gratuitement**.

Le site est tellement riche qu'il requiert une certaine pratique avant d'en bien saisir usages et possibilités, mais au-delà de cela, il constitue une adresse vraiment incontournable pour tout étudiant ou tout professionnel désireux de progresser en data science ou en machine learning.

1.2. Petfinder.my : mission

Le concours qui nous intéresse provient de **petfinder.my**, le site internet d'un refuge d'animaux malaisien, et s'intitule « Petfinder.my pawpularity contest ».

Comme tout refuge, Petfinder.my cherche à faire adopter le plus possible d'animaux. Et comme par ailleurs le meilleur moyen pour cela est qu'un maximum de ces animaux dispose d'une jolie photo, la mission proposée était de créer un modèle capable d'attribuer un « pawpularity score » à des photos.

Ceci permettant par exemple, pour un animal donné, d'automatiquement

mesurer la qualité de sa photo, et d'en soumettre de meilleures si la qualité n'était pas assez au rendez-vous.

1.3. Les données

Le dataset du concours contenait des données relatives à **9 912** individus.

Pour chacun d'eux nous disposions d'une photo de taille variable en mode RGB et de quelques métadonnées associées. Nos modélisations se feraient donc à partir de « données mixtes » (de types de données différentes…).

Subject Focus	Eyes	Face	Near	Action	Accessory	Group	Collage	Human	Occlusion	Info	Blur	Pawpularity
0	1	1	1	0	0	0	0	0	0	0	0	38

Ces métadonnées étaient constituées de 12 variables binaires liées à des caractéristiques éventuelles des images.

En plus de cela, elles contenaient dans la variable **Pawpularity** le score de chaque image qui ferait office de **label**, c'est à dire de valeur cible à prédire. Il s'agissait d'une note sur cent.

Enfin, le dataset ne contenait ni valeurs manquantes, ni valeurs erronées.

2. Approche générale

2.1. Une touche de feature engineering

L'observation des statistiques des images montrait, en plus d'une certaine variation de leurs tailles, la présence de petites photos.

	longueur	hauteur
count	9912.000000	9912.000000
mean	804.426251	904.284302
std	270.211921	156.905980
min	90.000000	113.000000
max	1280.000000	1280.000000

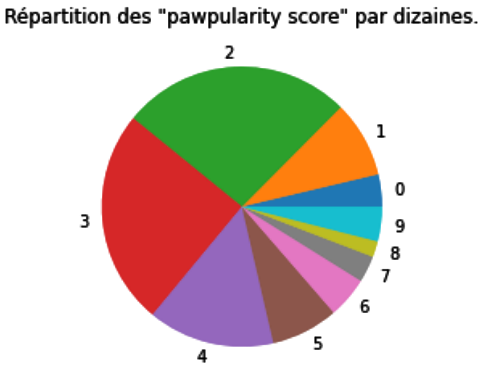
Comme une trop petite taille d'image constituait à notre sens un critère de (mauvaise…) qualité, nous avons décidé d'en tenir compte. De même avec leurs ratios hauteur / largeur (ou inversement).

Nous avons donc créé, puis ajouté aux métadonnées existantes, deux variables numériques normalisées (divisées par leur maximum) relatives aux tailles et ratios (allongement…) des photos.

2.2. Prise en compte des « inégalités »

Et nous ne sommes pas là dans un programme politique…

Nous nous référons en fait à la distribution des pawpularity score du dataset que nous avons observée après avoir classé ces scores par tranches de dix (grâce à une nouvelle variable catégorielle ad-hoc).



A cause du déséquilibre manifeste de cette distribution, nous avons décidé que nous ferions nos échantillonnages d'entraînement et de validation par tranche, afin que les modélisations se fassent ensuite sur des paquets de données de constitution assez proches de l'originale.

2.3. Apports d'un concours manqué…

Comme nous l'avons dit en préambule, avant de nous concentrer sur la compétition petfinder.my, nous avions tenté notre chance sur un autre concours.

C'est à cette occasion, qu'entre autres enseignements, nous avons appris à créer des **data generator** « customisés » grâce auxquels on peut modéliser « à la volée » à partir de virtuellement n'importe quel type de données.

```
class DataGenerator(Sequence):
    # Generates data
    def __init__(self,
                 list_IDs,
                 labels,
                 source,
                 (...)
```

Etant en présence de **mixed data**, nous avons dès le début mis en pratique ce nouveau savoir.

Comme nous fonctionnerions logiquement avec des modèles à input multiples (une entrée pour les images, une pour les métadonnées), nous avons utilisé des **couches de concaténation**, également découvertes lors de notre premier concours.

2.4. La régression par la classification

D'instinct, et cela constitue peut-être un manquement, nous nous sommes dirigés vers des options 100 % deep-learning. Du coup, ce projet allait être notre premier exercice de régression en deep learning.

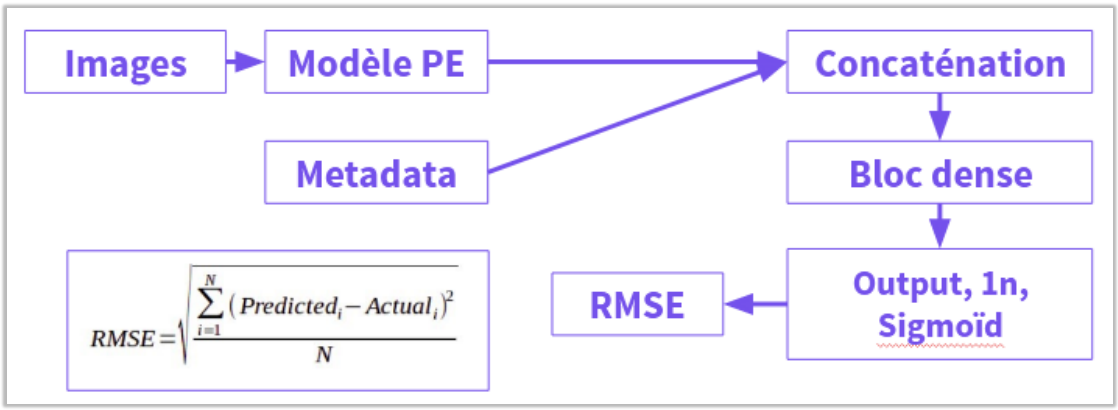
Il existe plusieurs façons de faire, et là encore, c'est peut-être un manque de curiosité, nous sommes restés collés à une méthode proche de ce qu'on connaissait, consistant à gérer notre régression en partie comme un problème de classification.

Le pawpularity score étant borné entre 0 et 100, en le normalisant nous nous retrouvions avec une régression entre 0 et 1 qui pouvait alors s'envisager comme une classification binaire dont on récupèrerait les prédictions exactes et non des arrondis.

Ainsi nous avons divisé nos valeurs cibles par 100. Nous avons utilisé **binary crossentropy** comme fonction de coût. Et notre neurone d'output avait une activation sigmoïde.

2.5. Dispositif de modélisation

Voici maintenant le dispositif général de modélisation que nous avons utilisé.



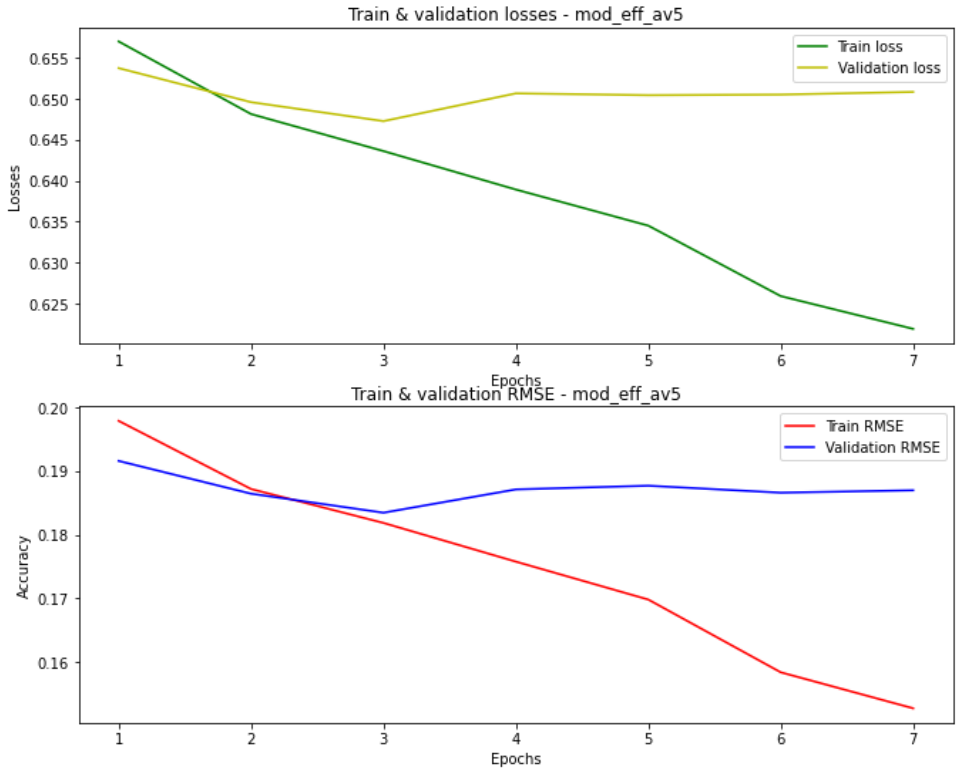
Les images allaient dans un modèle pré-entraîné pour extraction de features qui seraient par la suite concaténées aux métadonnées originales.

L'ensemble était envoyé ensuite dans un bloc MLP, en output duquel nous récupérions les prédictions.

Prédictions permettant le calcul de la **RMSE** (Root Mean Squared Error), interprétable comme la « marge d'erreur » du modèle qui était la métrique imposée du concours.

2.6. Modélisations avec modèles à convolution

Nous n'allons pas nous étendre sur cette phase de notre projet. Précisons seulement dans un premier temps que nous avons travaillé avec des modèles pré-entraînés du type de ceux que nous avions déjà utilisés lors du **projet 6** (Efficient Net, Inception ResNet, modèles VGG…), de classification d'images de canidés par race.



Comme le montre cette courbe indicative, nous manipulions alors des modèles globalement nerveux et prompts à l'**overfitting**, qui nous permettaient cependant d'atteindre des **RMSE** (honorables) de l'ordre de **18,5**.

C'est à ce stade que nous avons décidé d'utiliser la communauté Kaggle comme il se doit (…), en allant regarder ce qui se passait chez les voisins, particulièrement chez ceux qui semblaient obtenir mieux.

Et c'est ainsi que nous avons entendu parler des **SWIN Transformers**, de puissants modèles avec lesquels les meilleurs participants descendaient **en dessous** de **18**.

3. Utilisation de SWIN Transformers

3.1. Qu'est-ce qu'un SWIN Transformer ?

Lors du **Projet 7**, nous avions fait connaissance avec **BERT** et les **transformers** en NLP. Nous n'allons pas reprendre nos explications à leurs sujets mais juste en deux mots, il s'agit de puissants et complexes modèles, qui, grâce au système de l'attention, parviennent à représenter les mots d'un texte en fonction

de tout leur contexte, c'est-à-dire en fonction de tous les mots qui les entourent.

Ce système offre les représentations du langage les plus riches à ce jour.

Ensuite sont apparus les ViT, les « vision transformers », adaptations des transformers originaux de NLP dans le domaine de la computer vision.

Ceux-ci travaillent sur des images découpées en patchs (des morceaux généralement de dimension 16 x 16 pixels) qui font office de tokens.

Les SWIN transformers sont des versions « light » de ViT, apparus pour des raisons de scalabilité en raison de la puissance de calcul phénoménale requise par ces derniers dans certains cas.

« SWIN » signifie « shifted window » et renvoie au fonctionnement de ces modèles, basé sur une application restreinte du système d'attention et sur le balayage des images à plusieurs niveaux qu'ils opèrent sur les images (principe qui n'est pas sans rappeler la convolution…).

3.2. Recherche SWIN sous Tensorflow désespérément…

En voulant utiliser des SWIN transformers nous avons fait face à un problème, déjà rencontré lors du concours précédent avec les modèles EfficientNet 3D, à savoir que ceux-ci n'étaient disponibles que pour Pytorch, bibliothèque que nous ne pratiquons pas encore.

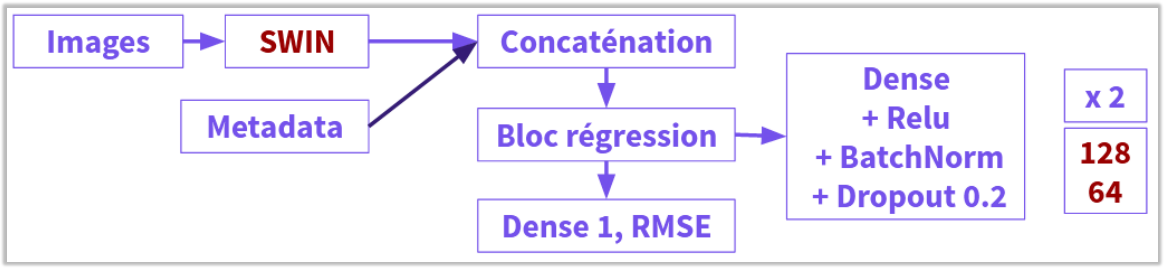
Sauf que cette fois, nous avons pu mettre la main sur des adaptations non officielles compatibles Tensorflow, notamment certaines qui étaient déjà hébergées dans des espaces « database » de Kaggle.

Malgré quelques craintes quant à leur stabilité, nous sommes parvenus à les insérer dans notre pipeline sans trop d'anicroches. Un point notable cependant, nous n'avons pas réussi à les faire fonctionner avec des TPU.

Cela a son importance car les transformers sont beaucoup plus rapides avec des TPU, et sans eux, nos modélisations duraient plus longtemps qu'elles ne l'auraient dû.

3.3. Dispositif final avec SWIN transformer

L'utilisation de ces modèles pré-entraînés ne changeait rien de notre approche initiale puisqu'en fin de compte, ils ne faisaient que remplacer, pour l'extraction de features depuis les images, les modèles à convolution pré-entraînés que nous utilisions juste avant.

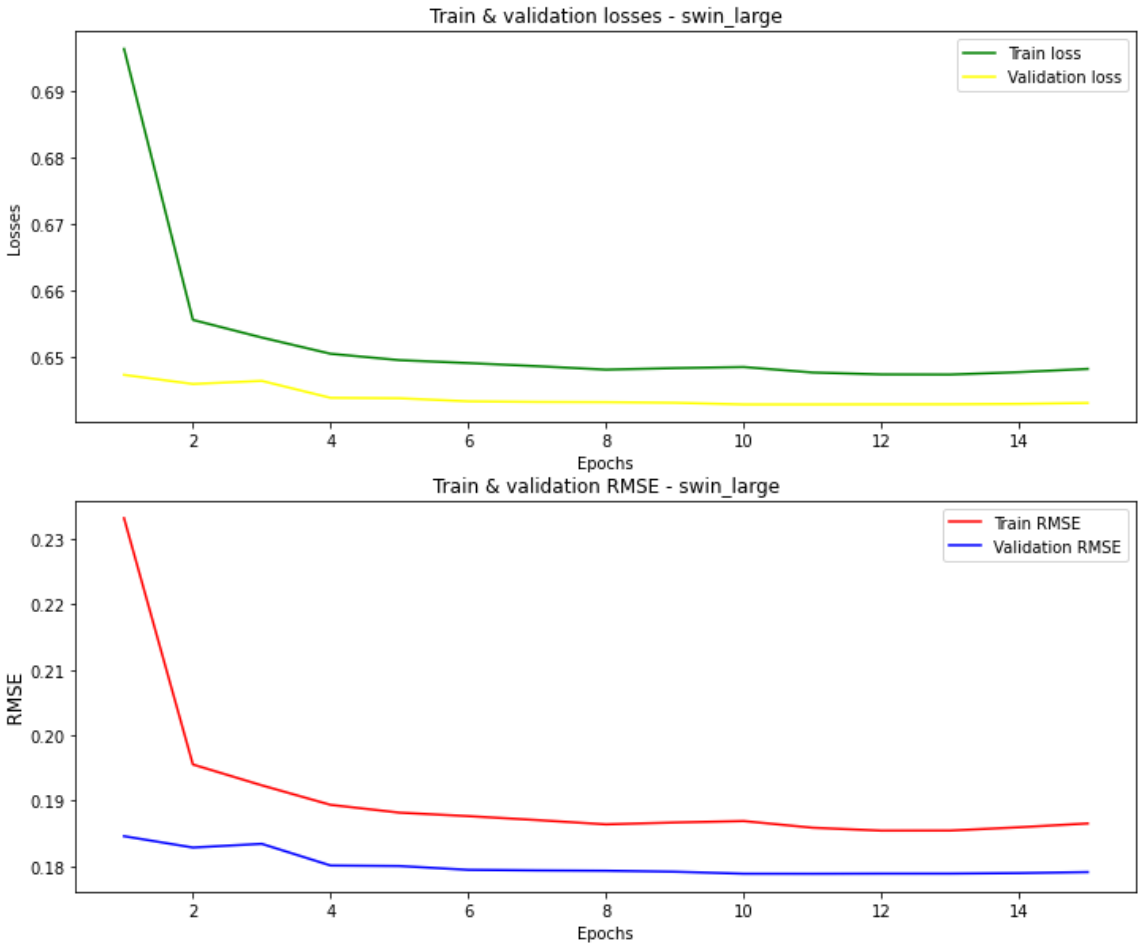


Notre pipeline restait le même. Notre travail à ce stade a principalement consisté à choisir, parmi les modèles pré-entraînés disponibles, celui qui nous semblait le meilleur, puis à composer un bloc dense semblant amener aux meilleurs résultats.

Notre choix se portera sur le modèle **SWIN Transformer Large**, pré-entraîné sur **ImageNet 22K** et fonctionnant au format **224x224**. C'était le second plus gros modèle de la bibliothèque avec **197** millions de paramètres, une taille dépassant **700** méga-octets, et un output de dimension **1536**.

Comme indiqué sur la figure ci-dessus, le bloc dense vers lequel nous avons convergé grâce à nos essais était composé de deux couches, avec activation **relu**, **batch normalisation**, et **0.2** de **dropout**, de respectivement **128** et **64** neurones.

Les courbes de modélisation obtenues étaient toutes de ce type.



Le but du jeu, par nos choix décrits plus haut, fut d'abord de descendre autant

que possible le « plancher » que le modèle atteignait assez vite.

Puis une fois ce plancher atteint, d'essayer de « gratter » des fractions de performance sur la longueur, en utilisant le callback **ReduceLROnPlateau** de manière assez agressive (division par un facteur 4 du learning rate à chaque rebond).

Le tout couplé avec un « early stopping » de « patience » 3 ou 4 sur des modélisations initialement assez longues de 25 epochs.

Ce modèle, en local, aboutissait à des **RMSE** de **17,9** (on rappelle le facteur 100 entre les résultats de modélisation et le format dans lequel ils sont attendus pour le concours…) et moins.

```
Epoch 16/20
991/991 [==============================] - 453s 457ms/step
val_root_mean_squared_error: 0.1768
```

Nous sommes même descendus à **17,68**, ce qui était meilleur que le meilleur score visible sur le « public leaderboard » du concours du moment.

Dès lors, satisfaits de notre modèle, nous sommes passés à l'étape de soumission.

4. Commentaires et conclusion

4.1. Un modèle qui généralise mal

Malheureusement, les scores obtenus après soumission ont été globalement décevants, le meilleur d'entre eux étant **18,38** (ce qui nous place dans le ventre mou du classement…).

Même s'il s'agissait là d'estimations provisoires, cela suffisait à mettre en évidence que notre modèle avait un réel problème de généralisation (les écarts sont moindres chez la concurrence…).

Les causes en étaient probablement multiples mais nous en feront ressortir une dont nous redoutions à l'avance l'influence : nos choix, en créant notre modèle basé sur un SWIN transformer, n'avaient pas été basés sur des résultats assez consistants.

La cause en est notre utilisation forcée de GPU au lieu de TPU (normalement beaucoup plus rapides avec des transformers…).

Chaque modélisation pouvant s'étendre sur plus de deux heures, avec en plus des quotas d'utilisation contraignants, nous avons basé nos choix sur un ou deux résultats maximum, au lieu de plus, comme une certaine variance de performance qu'on a observée par la suite l'aurait exigé.

Autrement dit, si on voulait jouer la « gagne » plus que simplement participer à notre compétition Kaggle, il faudrait probablement reprendre de manière plus minutieuse la construction de notre modèle.

4.2. Commentaire sur le concours lui-même

En l'état, il semble que les meilleurs modèles produiront une RMSE entre **17** et **18**. Par ailleurs, une prédiction naïve de la valeur moyenne des « pawpularity score » des données disponibles donnait une RMSE proche de **20,6**.

Nous pouvons en tirer deux observations. D'abord qu'aucun modèle, même aussi élaboré qu'un SWIN transformer, n'a beaucoup appris des données. Puis, qu'avec une marge d'erreur de 17 points sur la prédiction d'un score sur 100, ces modèles n'ont probablement pas réussi à remplir leur mission. Du moins aussi bien qu'attendu.

Cela nous renvoie à une intuition qui nous avait effleurés au moment de nous pencher sur le concours.

Nous sortions de notre échec sur celui de la RSNA-MICCAI, dont nous apprendrions peu de temps après qu'il était probablement impossible, et nous nous demandions alors si mesurer une « note d'adorabilité » d'animaux sur des photos n'était pas, encore, un projet un peu ésotérique.

Tout simplement parce que ça revenait en (grande ?) partie à demander à des modèles de mesurer à quel point un chat ou un chien était « mignon », ce qui à notre sens pouvait sortir de leur champ de compétence.

Ou tout du moins, cela nous paraissait autrement plus complexe à déterminer pour des modèles, que de se baser sur des éléments graphiques objectifs pour classer des chiens par race (notre expérience précédente en computer vision…).

Heureusement, ce concours nous a permis de produire des modèles capables d'apprentissage et de résultats, ce qui était notre principal objectif. Mais à côté de cela, les résultats observés montrent que notre intuition était peut-être fondée.

5. Ressources

5.1. Kaggle

- Petfinder.my Pawpularity Contest, accueil du concours objet de la présente note.
- RSNA-MICCAI Brain Tumor Radiogenomic Classification, accueil du premier concours auquel nous avons participé.
- Baseline TF Swin Transformer Model, présentation de notre modèle sur le forum de discussion du concours.
- Swin Transformer (Tensorflow), adaptation Tensorflow du module « Swin Transformer » que nous avons utilisée, hébergée par **Hyeong Chan Kim**. Merci à lui !

5.2. SWIN Transformers

- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, (mars 2021) par Ze Liu, Yutong Lin, et al.
- SWIN Transformers animés et expliqués, pour mieux comprendre la publication officielle ci-dessus, une vidéo de la chaîne **AI Coffee Break with Letitia** (qui ne sait pas écrire son prénom… mais on lui pardonnera).