

ARN - PW5

Auteurs: Jonathan Friedli, Valentin Kaelin

Date: 17.06.2022

Tables des matières

[Introduction](#)
[Le problème](#)
[Préparation des données](#)
[Création du modèle](#)
 [Hyper paramètres et architecture](#)
 [Transfer learning](#)
[Résultats](#)
 [a\) Matrice de confusion](#)
 [b\) F-score obtenus pour chaque classe](#)
 [c\) Résultats sur le test set](#)
 [d\) Analyse du grad-cam](#)
 [e\) Exemples d'images mal classées](#)
 [f\) Comment améliorer notre dataset ?](#)
 [g\) Quelles classes sont confuses ?](#)
[Conclusions](#)

Introduction

Dans ce travail pratique, vous devons effectuer toutes les étapes nécessaires à la création d'une application de classification d'objets :

- 1) Création d'un ensemble de données
- 2) Exploration des données
- 3) Augmentation des données
- 4) Prétraitement
- 5) Sélection de modèles
- 6) Évaluation des performances.

Durant ce laboratoire, notre application devra apprendre à reconnaître différents types de balle de différents sports (Football, Basketball, Tennis, Ping Pong, SpikeBall et Volleyball). Pour ce faire, nous allons prendre en photo les différents types de balles/ballons. Le but étant d'avoir entre 15 et 20 images par type de balle. Nous allons essayer de ne pas utiliser d'images supplémentaires d'internet afin d'avoir une expérience plus exotique lors de ce laboratoire comparée à nos travaux précédents. Nous allons ensuite utiliser des techniques du transfert learning afin de créer notre modèle. Nous expliquerons par la suite (au point 3) pourquoi et comment utiliser le transfer learning.

Dans les dernières parties de ce laboratoire, nous allons transférer le modèle sur notre téléphone pour ensuite pouvoir tester ce dernier via la caméra du téléphone.

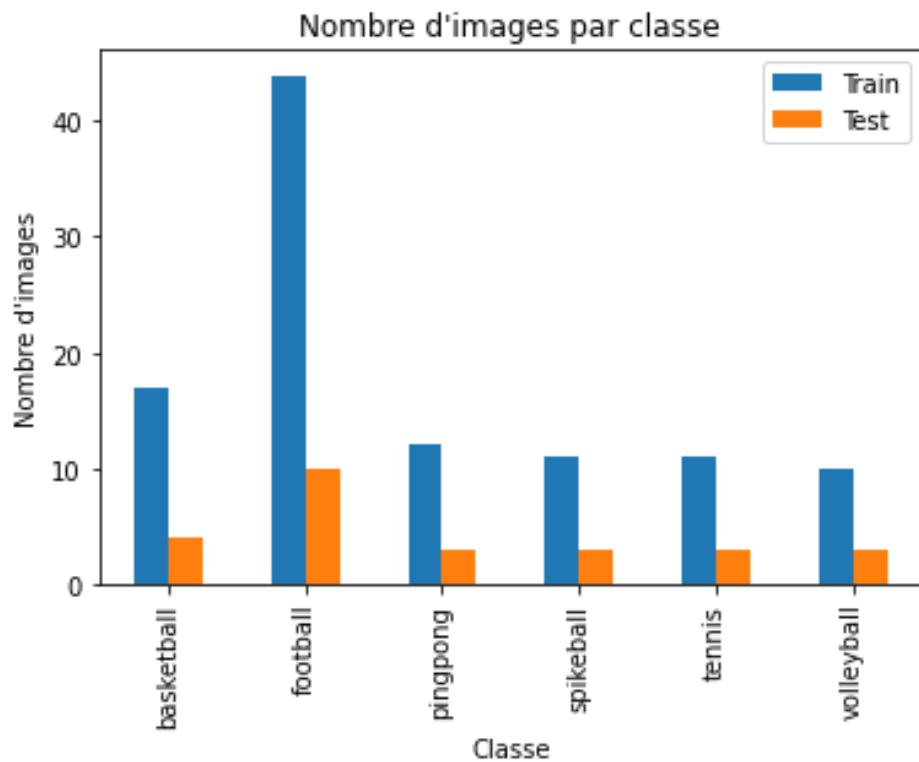
Le problème

Nous utilisons les 6 classes suivantes qui correspondent au sport de la balle en question:

1. basketball
2. football
3. pingpong
4. spikeball
5. tennis
6. volleyball

Nous avons pris une centaine de photos de nos différents ballons de sport, toutes les photos sont dans le même format (.jpg) afin d'uniformiser le plus possible le dataset.

Nous avons essayé de le rendre le plus équilibré possible, en faisant environ 8 photos par modèle de balle pour chaque sport. Chaque photo a été prise à environ 1m de distance et sur fond neutre (herbe ou goudron). Comme les balles de certains sports peuvent beaucoup varier (ex: le football), le dataset n'est pas le plus équilibré qui existe:



Nous avons préféré ajouter plus de balles de football différentes afin d'espérer que le modèle puisse mieux s'adapter et pas se restreindre à 2-3 ballons. Vu le nombre peu élevé de données par classe, il ne s'agit clairement pas de Big Data mais plutôt de Small Data.



La difficulté principale que nous observons sont l'énorme diversité des ballons de football, comme nous pouvons le voir sur les images ci-dessus. Contrairement aux autres sports, leur design varie beaucoup.



Une autre difficulté est la ressemblance entre les balles de tennis et de Spikeball. Elles sont en effet de même couleur et quasiment de même taille. Il n'y a que le petit design en plus sur les balles de Spikeball qui permet de les différencier.

Préparation des données

Lorsque nous avons pris nos photos, nous avons essayé d'avoir toujours un background assez semblable (de l'herbe ou du goudron). Nous avons également pris les balles en photo à environ 1m afin que le modèle puisse comparer les différentes tailles. Nous avons supprimé les photos où le sujet était un peu flou ou celles où le background était trop bruité.

Nous avons augmenté le dataset de train en utilisant de nombreux filtres:

```
image_augmentations = Sequential([
    RandomFlip(seed=None),
    RandomContrast(factor=[0.3, 0.7], seed=None),
    RandomZoom(height_factor=(0.0, -0.3), width_factor=(0.0, -0.3), fill_mode="reflect", interpolation="bilinear", seed=None),
    RandomRotation(0.4),
])
```

Nous avons donc appliqué des rotations, zooms, modifications de contraste et potentiels flip à nos images. Ces différentes modifications ont été bénéfiques à notre dataset car nos images étaient avant ces modifications assez semblables, comme nous avions pris plusieurs photos de chaque balle. Pour finir, nous avons redimensionner les images aux dimensions (224pixels x 224pixels) afin qu'elles aient toutes la même taille avant d'entraîner le modèle.

Pour finir, nous avons split nos images en deux datasets: 80% pour le train et 20% pour le test. La séparation n'a pas été faite aléatoirement mais plutôt à la main. Nous avons essayé de faire en sorte que le test set, bien que plus petit, contienne des images assez variées, et pas uniquement des images de la même balle par exemple. Nous avons donc gardé 20% des images de chaque type de balles pour le test.

Création du modèle

Hyper paramètres et architecture

Nombre d'epochs final: 5

Nombre de dense layers: 1 layer de 250 neurones

Optimizer: RMSprop, qui correspond à une descente de gradient avec un momentum classique. Pour être tout à fait honnête, nous sommes partis sur cet optimizer car il s'agissait de celui utilisé dans la démonstration de la première partie.

Learning rate: Nous utilisons le learning rate par défaut de l'optimizer RMSprop qui est de 0.001

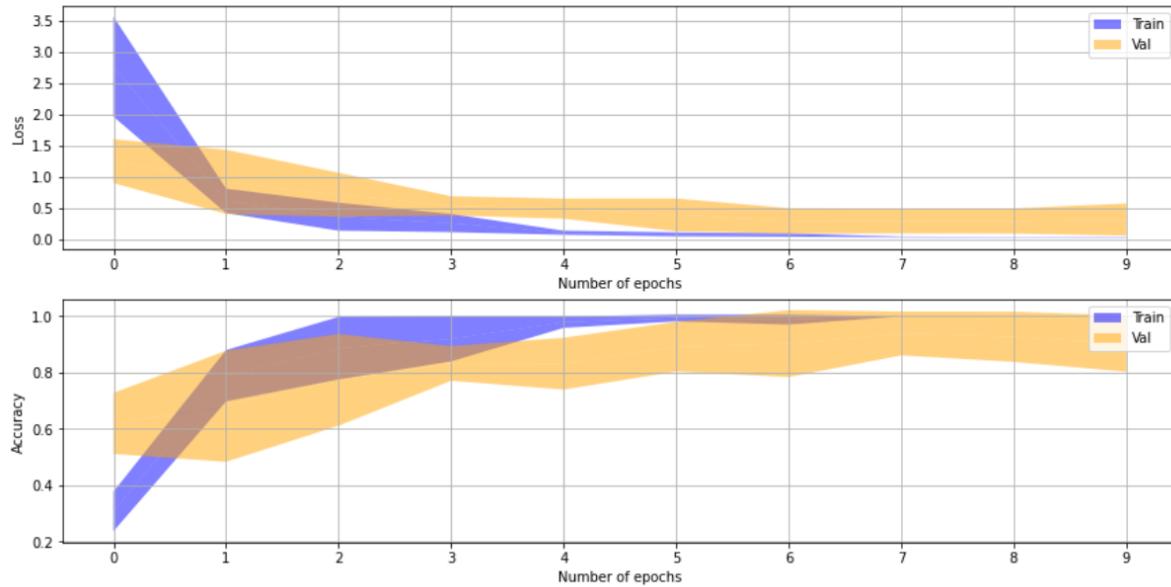
Fonction d'activation: relu

Batch size : 32 il s'agit de la valeur de base du modèle fourni

Nous avons essayé divers hyper paramètres avant d'arriver à ceux-précédemment cités, voici nos différents résultats:

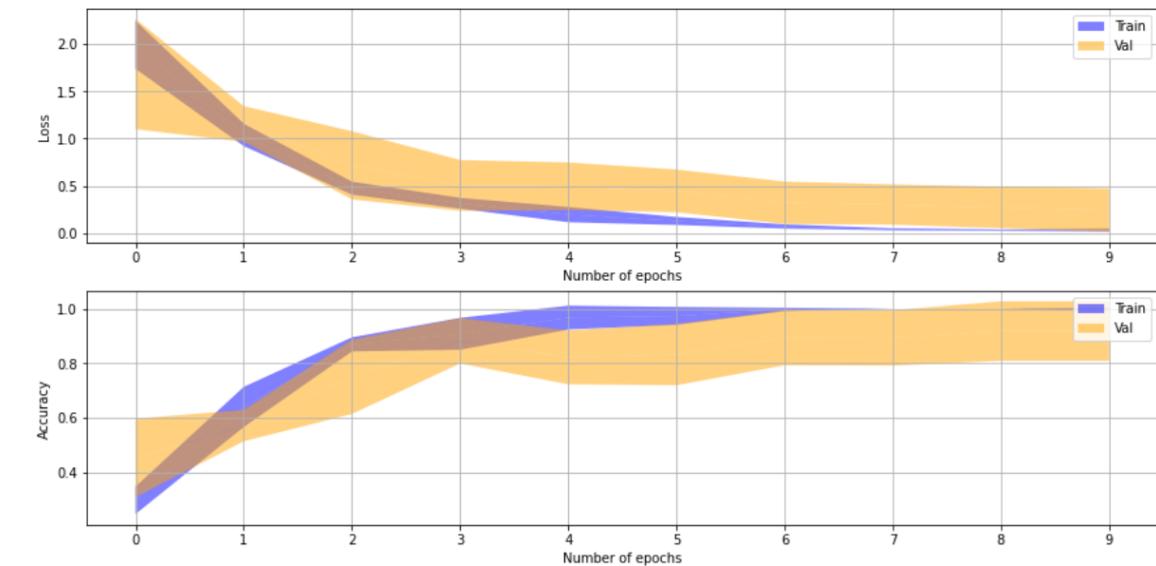
Avec 10 epochs et 1 layer de 250 neurones:

```
Train Loss function: 0.028864263370633126
Train Accuracy: 1.0
-----
Val Loss function: 0.31322699785232544
Val Accuracy: 0.9029411792755127
```



avec 10 epochs et 2 layers de 150 neurones:

```
Train Loss function: 0.03589640874415636
Train Accuracy: 0.9970149278640748
-----
Val Loss function: 0.24664127975702285
Val Accuracy: 0.9176470637321472
```



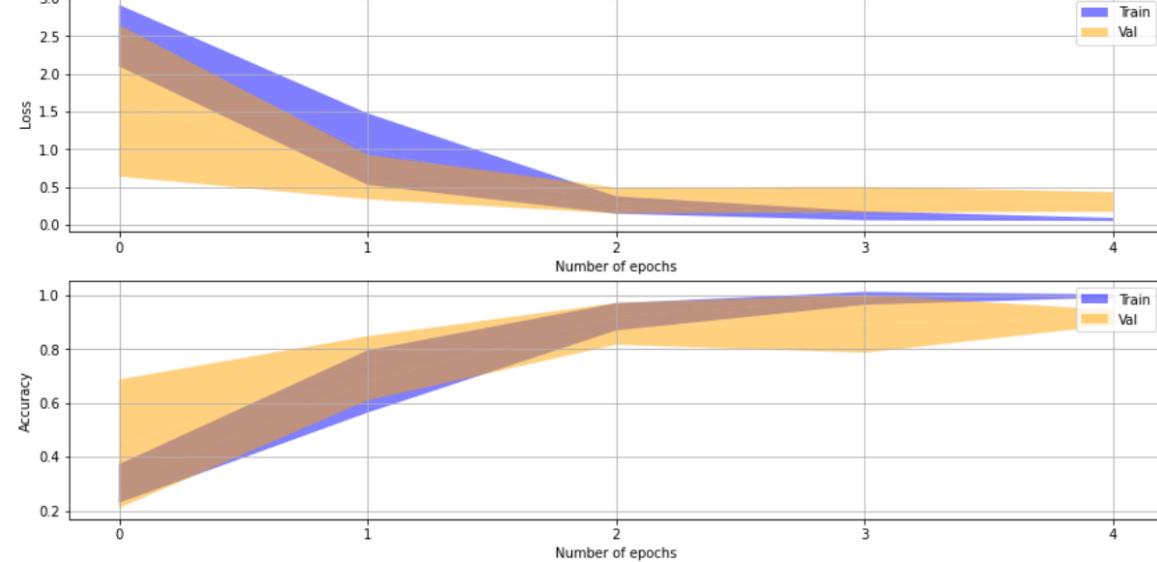
avec 5 epochs et 1 layer de 250 neurones:

Train Loss function: 0.07044398412108421

Train Accuracy: 0.9970149278640748

Val Loss function: 0.30277183949947356

Val Accuracy: 0.9169117689132691



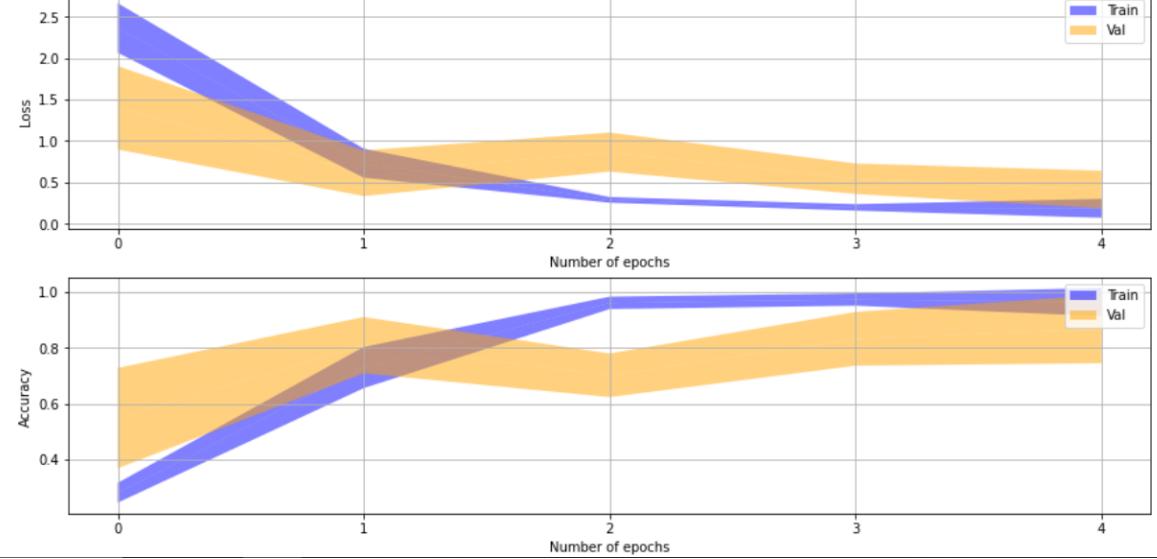
5 epochs et 1 layer de 150 neurones

Train Loss function: 0.18612505346536637

Train Accuracy: 0.9641791105270385

Val Loss function: 0.41297467648983

Val Accuracy: 0.8676470637321472



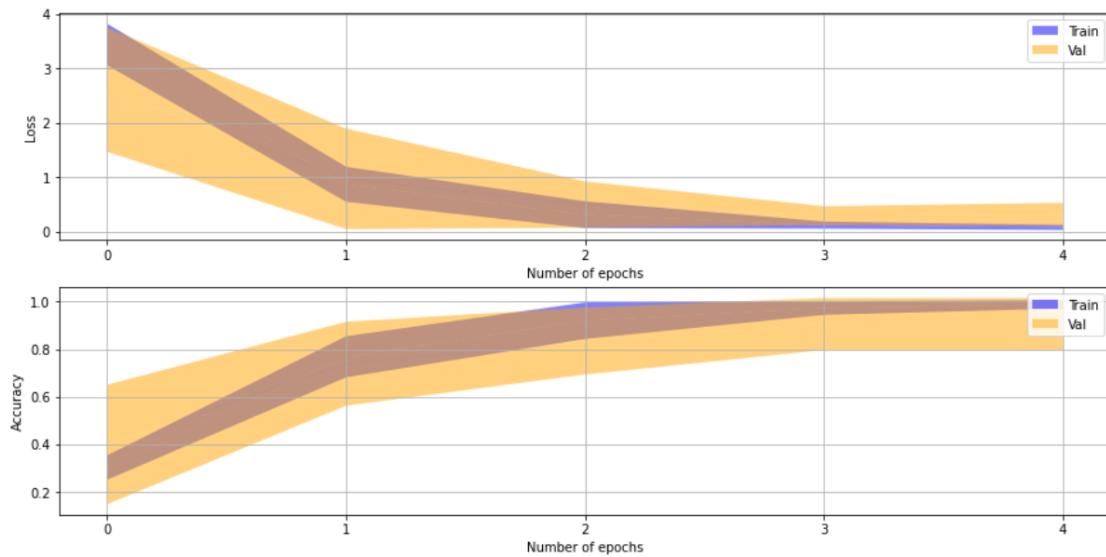
5 epochs et 1 layer de 500 neurones:

Train Loss function: 0.0763882003724575

Train Accuracy: 0.9880596995353699

Val Loss function: 0.3152931064367294

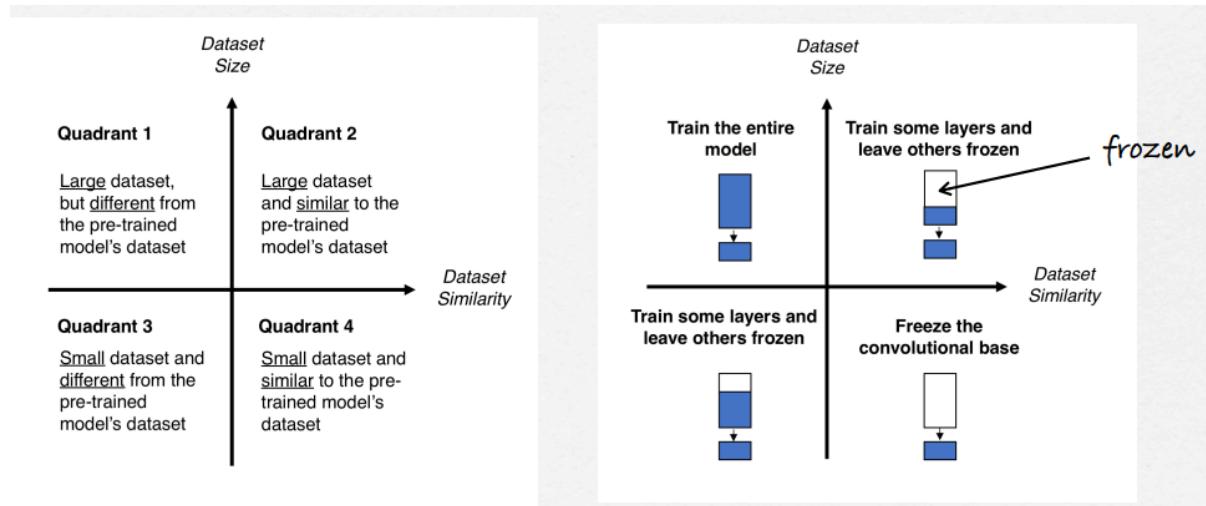
Val Accuracy: 0.9058823585510254



Les deux meilleures performances sont avec un layer de 250 neurones, 5 epoch et 2 layer de 150 neurones, 10 epoch. Les valeurs de l'accuracy et de la Loss function sont plus ou moins équivalent. Cependant dans le cas du layer de 250 neurones, l'intervalle de valeur du validation set (le trait en jaune sur les graphes) est bien moins large que celui du modèle avec 2 layer de 150 neurones et 10 epoch. De plus, ce dernier modèle est un peu plus overfitté que le premier, c'est pourquoi nous avons choisi le modèle avec 1 seul layer de 250 neurones et 5 epoch.

Transfer learning

Nous avons utilisé mobilenetV2 comme modèle de base pour le transfer learning. Nous avons ensuite gelé (freeze) les layers de la base de convolution du modèle de base. En effet, nous avons un petit dataset qui est relativement similaire à celui du dataset du modèle de base, vu que les balles/ballons de sport sont des objets très communs. Comme montré sur ce schéma vu en cours, c'était sans aucun doute la meilleure option, d'autant plus que notre dataset personnel est assez petit (environ 150 images).



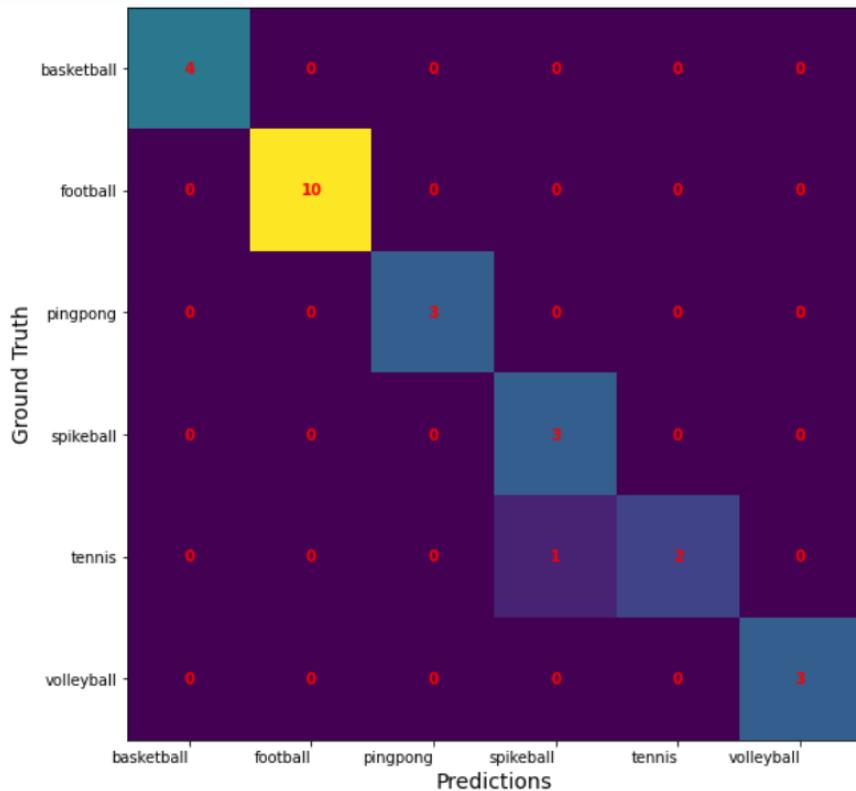
Nous avons utilisé le transfert learning car c'était le meilleur moyen d'obtenir rapidement un modèle assez performant. En effet, au vu du nombre assez faible de photo dans notre dataset sans l'utilisation du transfert learning, nous aurions eu un

modèle très mauvais et très probablement overfitté. De plus, les ballons de sport étant des objets très communs, nous étions pratiquement sûrs qu'ils seraient présents dans imageNet.

Résultats

a) Matrice de confusion

Voici la matrice de confusion de notre modèle :



b) F-score obtenus pour chaque classe

Voici les fscores obtenus lors de l'évaluation du test set.

```
F-score basketball: 1.0
F-score football: 1.0
F-score pingpong: 1.0
F-score spikeball: 0.8571428571428571
F-score tennis: 0.8
F-score volleyball: 1.0

F1 score global: 0.942857142857143
```

Nous observons que les fscores sont vraiment bons (proches de 1). Une balle de tennis a été confondue comme une balle de spikeball. C'est en effet le souci principal que notre modèle rencontre. Le spikeball étant moins connu, il doit certainement être moins contenu dans le modèle imageNet. De plus, les balles de spikeball et de tennis se ressemblent énormément. Nous allons revenir plus en détails sur ce cas lors du point e).

c) Résultats sur le test set

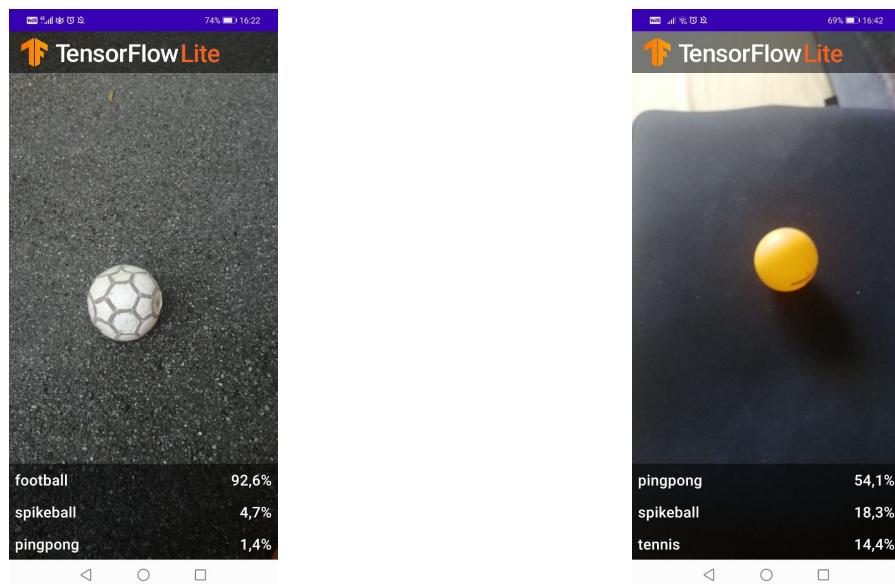
Dans le cas de notre modèle, la loss function est bien plus faible dans le cadre du train set que dans celui du test set. En effet, elle passe de 0.07 à 0.3 ce qui correspond environ à une multiplication par 4. C'est, cependant, plutôt normal car nous avons très peu de valeurs et donc il y a un peu d'apprentissage par coeur. De plus, la valeur de la loss function est

vraiment très faible dans le cas du train set. Nous nous attendions donc à avoir une loss function plus élevée dans le cas du test set.

Nous retrouvons également ce phénomène pour l'accuracy. En effet, elle passe de 0.997 dans le train set à 0.917 dans le test set. Cela correspond à une perte d'environ 10%. Cependant, il faut tenir en compte le fait que l'accuracy est très élevée dans les deux cas. C'est donc plutôt normal que l'accuracy baisse dans le cas du test set. Ces résultats, même étant moins bons que ceux lors du train set, sont quand même très bons.

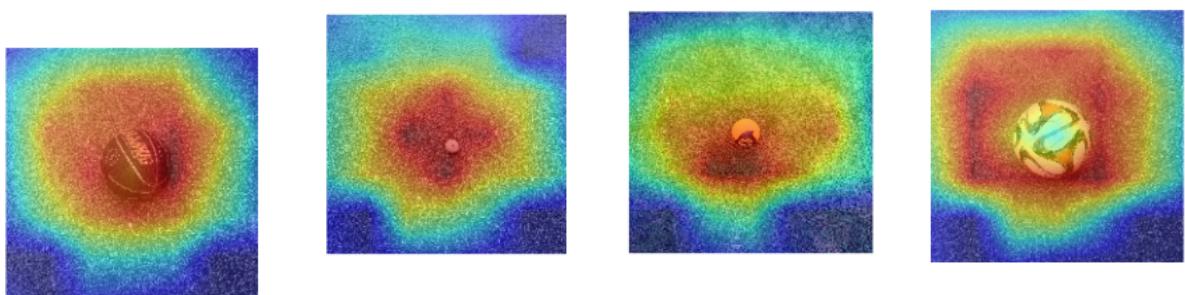
avec 5 epochs et 1 layer de 250 neurones:

```
Train Loss function: 0.07044398412108421
Train Accuracy: 0.9970149278640748
-----
Val Loss function: 0.30277183949947356
Val Accuracy: 0.9169117689132691
```

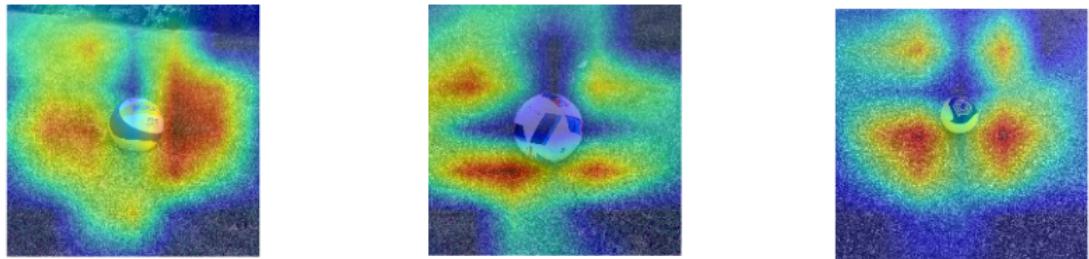


Lorsque nous avons essayé les prédictions de notre modèle sur nos téléphones, nous avons été agréablement surpris des résultats. En effet, dans la plupart des cas, les résultats sont corrects. Ces derniers oscillent entre 90% et 50% lorsque la balle se trouve sur un fond neutre. Nous discuterons de la précisions de nos résultats sur des fonds différents lors des points e) et g).

d) Analyse du grad-cam

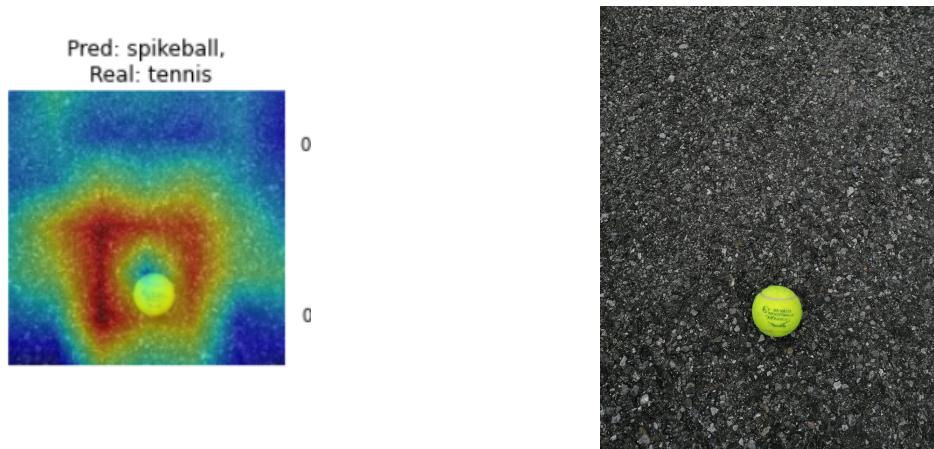


Ci-dessus, nous observons la grad cam des objets ayant été prédits correctement. Nous voyons directement que le ballon est à chaque fois au centre de la zone bien rouge. Le modèle a donc bien détecté des caractéristiques liées aux ballons afin de déterminer leur type.



Nous remarquons cependant que sur certaines images, la partie rouge foncé n'est pas exactement sur la balles. Cela est plutôt surprenant car nous avons fait attention à avoir un fond qui est commun à tous les types de ballons (soit de l'herbe, soit du goudron). C'est un point faible de notre modèle car ce dernier trouve des informations pertinentes alors que nous, humains, n'en voyons pas. Dans un cas réel, nous aurions modifié ce modèle afin d'éviter d'arriver dans ce genre de cas.

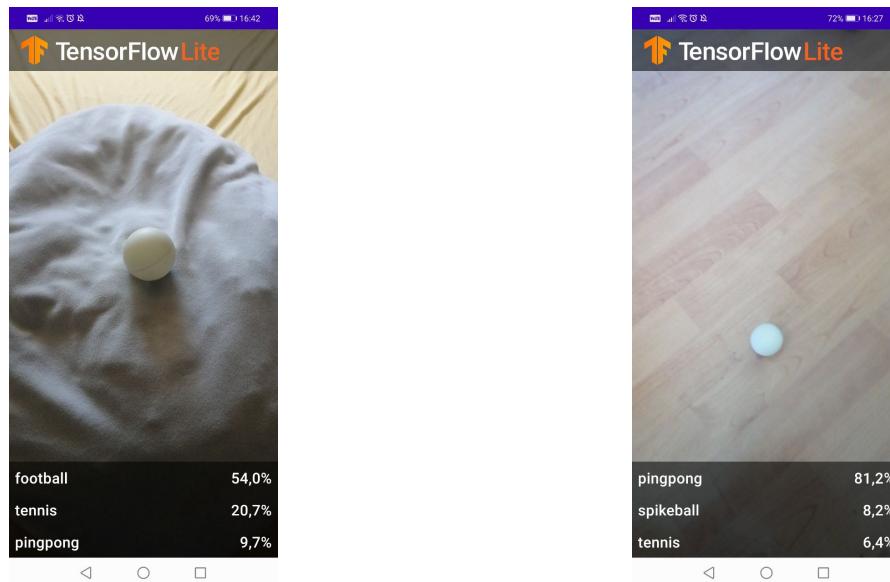
e) Exemples d'images mal classées



Ci-dessus, il s'agit de la seule image de base qui a été mal prédite lors de l'évaluation du test set. Il s'agit d'une balle de tennis mais le système l'a classifiée en tant que balle de spikeball. Cela n'est pas si étonnant que cela car les deux types de balles sont plutôt similaires car elles sont de mêmes tailles et sont les deux jaunes. De plus, nous voyons que la partie rouge de la grad cam n'est pas du tout sur la balle mais plutôt tout autour de cette dernière. Comme nous l'avons expliqué au point d), le modèle a malheureusement trouvé des caractéristiques qui n'en sont en réalité pas.



Ci-dessus, nous pouvons voir un échec de prédiction depuis la version sur téléphone. Il s'agit d'un ballon de basket qui a été prédit en tant que balle de ping pong. Le ballon de basket étant très usé et assez lisse, l'appareil a eu du mal à bien voir les petites rainures sur ce dernier. Il est donc assez compréhensible qu'il soit mal prédit, car il ressemble beaucoup à une balle de ping pong orange. De plus, il est très dur pour le modèle de savoir si nous sommes à une dizaine de centimètre de l'objet ou à environ 1m. Donc la taille du ballon n'a pas un si grand impact que cela.



La dernière image mal prédite est une balle de ping pong sur un coussin rond. Ici, la balle est confondue avec un ballon de football mais cela est très clairement dû à la présence du coussin. Nous pensons que le modèle n'a pas vraiment su si l'objet dont il devait trouver le type était la balle ou le coussin rond. A droite nous voyons la même balle de ping pong sur un fond neutre et la prédiction du modèle est plus que correct. (81% pour pingpong)

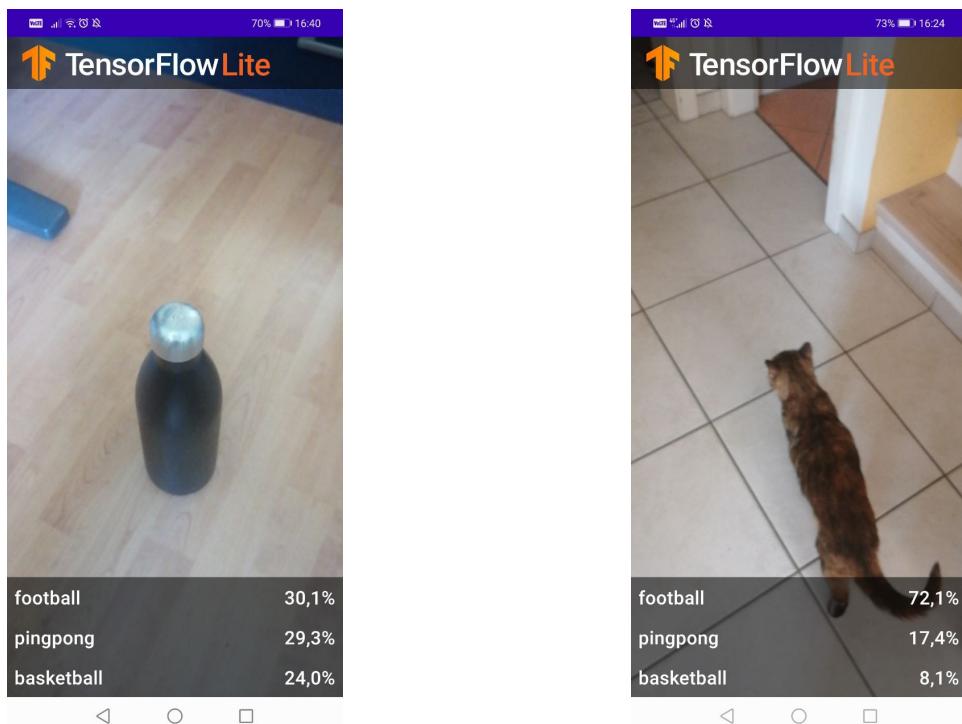
f) Comment améliorer notre dataset ?

Nous pourrions ajouter plus d'images. Le nombre d'images pour le type football par exemple (40 au lieu de 10-15) permet de travailler bien plus facilement. En effet, notre test set bénéficierait beaucoup d'un peu plus de données. Certains sports ont par exemple uniquement 3-4 photos dans le test set, ce qui n'est pas vraiment suffisant pour une bonne évaluation.

De plus, faire des photos des différents types de balles à différentes distance (ici, nous avons toujours gardé environ 1m de distance) permettrait sans doute d'avoir de meilleures performances lors de tests réels avec l'application sur smartphone par exemple.

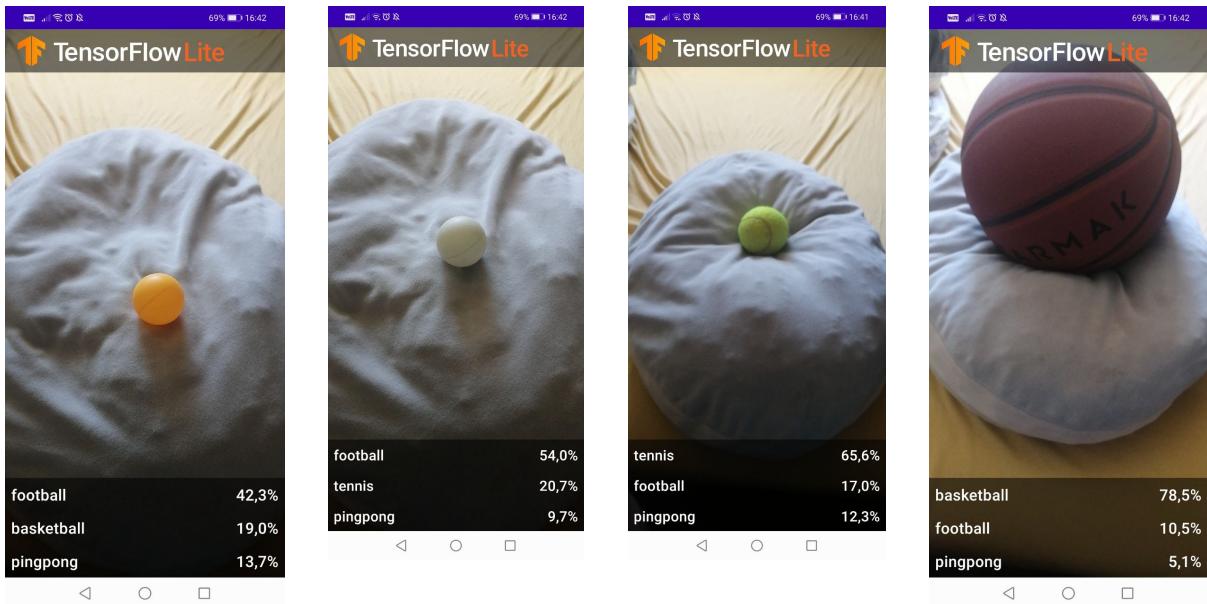
g) Quelles classes sont confuses ?

Comme mentionné dans l'introduction, les classes tennis et spikeball ont été quelque peu confondues. C'est d'ailleurs la seule prédiction fausse lors de l'évaluation du test set (une balle de tennis prédite en tant que balle de spikeBall). Nous ne sommes pas du tout surpris qu'une telle prédiction puisse se produire. En effet, la taille et la couleur étant à ce point similaires (la principale différence entre ces deux balles est leur matière mais ce n'est pas visible sur les images), la confusion est très facile. Même un humain, qui ne connaît pas le spikeball, pourrait penser que cette balle est une balle de tennis.



Ci-dessus, nous avons deux images n'appartenant pas du tout à notre dataset. Il s'agit d'une gourde et d'un chat. Dans les deux cas, le type prédict était football, même si dans le cas de la bouteille le système était bien plus hésitant.

Cela est sûrement dû au fait que nous avons une grosse majorité de photo de ballon de football dans notre dataset. Cela résulte sûrement en une prédiction un peu abusive de cette catégorie.



Nous avons par la suite tenté de modifier le background de nos balles afin de voir si la prédiction était affectée. Comme discuté au point e), la balle de ping pong sur un gros coussin rond a causé une mauvaise prédiction.

Lorsque nous mettons la balle sur un coussin, cela a un effet négatif uniquement lorsqu'il s'agit d'une balle de ping pong. Nous supposons que les autres balles, étant plus grosse, sont bien moins affectée par le coussin.



Nous avons également fait un test où nous mettons différents types de balle au même endroit. Nous avons d'abord été plutôt surpris car le modèle a réussi à trouver tous les types de balles présent dans l'image. Mais en réfléchissant, c'est plutôt normal car il retrouve toutes les caractéristiques requises afin d'identifier chaque type de balle. Nous constatons également que le pourcentage de précisions de chaque type est environ équivalent, ce qui est encore un point positif.

Conclusions

Pour commencer, nous avons rencontrés diverses difficultés lors de ce dernier laboratoire. Par exemple lors de l'augmentation du dataset, il nous fallait redémarrer le kernel si nous voulions retenter l'augmentation. Ce qui a rendu le processus plus long. L'entraînement des différents modèles était également très long (parfois plus de 10 minutes), il a donc été assez difficile de trouver un très bon modèle. Pour finir sur les divers problèmes, la librairie tensorflow n'est pas encore facilement compatible avec les ordinateurs ARM, ce qui a également causé quelques soucis.

Nous sommes globalement très satisfaits de notre modèle. Nous avons même dépassé nos espérances. En effet, en choisissant de faire nos dataset avec uniquement nos photos, nous pensions avoir un modèle final bien moins performant, surtout sur de nouvelles balles non présentes dans le dataset de train.

Afin d'expliquer des résultats aussi bons, il faut prendre en compte le fait que nous avons choisi une problématique assez "simple". En effet, tous les types de ballons étant assez différents les uns des autres (mis à part les balles de tennis et de spikeball) et ce sont des objets du quotidien. Il était donc assez simple de prendre nos photos et d'en faire un dataset. De plus, le modèle de base d'imageNet doit avoir été entraîné sur un nombre conséquent de ballons car, encore une fois, il s'agit d'objets communs du quotidien.

Même si nous trouvons que notre modèle est bon, nous sommes tout à fait conscients de ses limites. Premièrement, notre modèle est tout de même assez susceptible au changement de background. Comme montré dans nos résultats, les classes prédites peuvent être totalement fausses (ex: football à plus de 50% alors qu'il s'agit d'une balle de ping pong). Pour améliorer ce point, il nous faudrait certainement plus d'image et principalement des images avec des backgrounds plus variés. Il faudrait également prendre des photos à différentes distances. Même si l'augmentation des données sert à ça, nous pensons que de vraies données pourraient aider encore plus sur ce point.

Secondement, il faudrait plus uniformiser notre dataset d'images. Actuellement nous avons environ entre deux et trois fois plus d'images de football que des autres classes et si nous essayons de catégoriser une image correspondant à aucun type de balle, le modèle va souvent la prédire comme une balle de football.

Nous pensons qu'avec toutes les améliorations proposées, notre modèle pourrait encore s'améliorer et être moins perturbé quand nous demandons de prédire le type d'une balle sur une image moins conventionnelle. De plus avec un peu plus de temps, nous aurions sûrement pu trouver un modèle plus performant (en changeant le nombre de layers et le nombre de neurones par couche). Il aurait été possible de lancer un notebook une nuit avec une grande liste d'hyper paramètres à tester afin de choisir plus précisément lesdits hyper paramètres.

Pour finir, nous sommes très contents d'avoir pu expérimenter la création complète d'une application de classification d'objet. En effet, être passé au travers de l'entièreté du processus, de la création du dataset aux tests finaux sur smartphone, nous a permis d'avoir un meilleur aperçu du travail derrière la création des applications de transfer learning. De plus, nous avons grandement apprécié réaliser une application très concrète, directement testable dans la vie de tous les jours.