Rapport Projet Apprentissage Progond
Classification d'Image : Sport

Groupe L12-07

Wail ANWAR
Martin BONIOL
Luka CHAVOIT
Octave CHAMPY

Département Sciences du Numérique - Filière Logiciel - Deuxième année
2021-2022

Table des matières

1 Introduction

1.1 Quelques infos pour commencer

L'objectif de ce projet est de mettre en oeuvre un algorithme d'apprentissage profond, depuis la constitution d'une Base de Données (BD) d'entrainement, jusqu'à l'analyse finale des résultats.

Le groupe pour ce projet est constitué de 4 étudiants de la filière Logiciel de l'ENSEEIHT : Wail ANWAR, Martin BONIOL, Luka CHAVOIT et Octave CHAMPY.

Ici vous trouverez le lien vers le GitHub pour accéder à la BD associée au sujet choisit :

https://github.com/MartinBoniol/DeepLearnig_Project

1.2 Choix du Sujet

Nous souhaitons, lors de ce projet de classification d'images, réaliser un algorithme d'apprentissage profond permettant la reconnaissance du sport présent sur une image. Ce projet se limitera à la reconnaissance de seulement 5 sports (Foot, Basket, Hockey sur glace, Tennis et Rugby) pour certaines raisons de temps. Le travail reste cependant très simple si nous voulons faire la même chose avec d'autres sports, il suffit d'augmenter la Base de Donnée en rajoutant une classe par sport que l'on veut ajouter.

Pour notre sujet nous avons alors décidé de créer 5 classes respectivement aux sports choisis. Chaque classe est constituée de 200 images, cette quantité d'image nous semble suffisante et satisfaisante afin de construire une Base Donnée pour un tel projet. Cependant il n'est pas impossible que durant le déroulement du projet, certaines modifications pourront être apportée à notre Base de Donnée.

2 Base de Donnée

1. Comme indiqué précédement la Base de Donnée est constituée de 5 classes, qui sont elles même constituées de 200 images. Ces images ont été récupérées sur Google Image sans utilisées de technique particulière.

2. L'annotation des données a été réalisée de façon automatique graçe à l'utilisation d'un script (cf le code *annotation.sh* disponible dans le GitHub). En effet le lancement de ce script permet de convertir chaque image, peu importe son nom ou son format, et la renommer $NomClasse\_NumeroImage.jpg$.

Par exemple voici ci dessous les images nommées $Tennis\_65.jpg$ de la classe Tennis et $Hockey\_36.jpg$ de la classe Hockey :

FIGURE 1 – Image de Tennis : $Tennis\_65.jpg$

FIGURE 2 – Image de Hockey : $Hockey\_36.jpg$

3. Notre méthodologie de partitionnement de nos images en 3 sets ($Apprentissage$, $Validation$ et $Test$) consiste à répartir aléatoirement les images de chaques classes dans ces sets de tel sorte qu'il doit se trouver : 70% de chaque classe dans le set $Apprentissage$ et 15% de chaque classe dans les sets $Validation$ et $Test$.
   Pour réaliser cela nous avons rédigé un script permettant cette répartition aléatoire et organisée des classes (cf $Classification.sh$ dans le GitHub). Ainsi l'automatisation est réalisée avec un script qui tire aléatoirement les indices des images qui à mettre dans $Test$ et $Validation$, puis les images restantes iront dans $Apprentissage$.

3 Attente du Projet

La difficulté de ce projet est assez variable en fonction de ce que l'on peut retrouver sur les images. En effet certains angles montrant uniquement un jouer d'un sport sans ballon, sans raquette ou sans équipement particulier peut pet-être porter à confusion. De même si uniquement les terrains apparaissent, en plan large ceux ci peuvent etre assez similaire entre les differents sports.
Nous espérons tout de même ne pas rencontrer trop de difficultés. En effet la confusion la plus probable risque d'apparaitre entre le Foot et Rugby, mais le Tennis, le Hockey et le Basket restent néanmoins assez distinctif.
De plus les images choisis pour l'instant ne présentent pas le problème du joueur seul à l'image, cependant nous pourront par la suite, si le projet se passe bien, améliorer notre travail pour y ajouter cette difficulté.

4 Chargement des Données

Le script de chargement des données est disponible sur le GitHub (ainsi que dans l'archive déposé sur moodle).
Nous avons adapté le modèle disponible sur moodle à notre projet. Cela affiche bien les différentes images de notre label.

Ci dessous quelques photos de notre BD, chargé par le code python :

FIGURE 3 – Images de la BD (Hockey et Tennis)

FIGURE 4 – Images de la BD (Basket)

5 Présentation du modèle

5.1 Réseau convolutif de base

Nous avons essayé d'améliorer notre modèle avec la technique du transfert learning en utilisant le modèle VGG-16. Cependant celui qui reste le plus performant est **l'approche par réseau convolutif de base**.

Notre réseau alterne tout d'abord les couches de convolutions et de Max Pooling afin de diviser à chaque fois la dimension des tenseurs par 2. La première couche compte 32 filtres de convolutions, la deuxième 64, la troisème 96 et la 4e 128. Ces 4 premières couches ont pour activation "relu" et utilisent la dimension 64x64x3 des images.
On met ensuite à plat le tenseur pour permettre de le connecter à une couche dense. On ajoute donc une couche dense à 512 neuronnes d'activation "sigmoid" puis une couche de sortie de taille 5 correspondant aux 5 labels et d'activation "softmax".

Pour l'entrainement du modèle, on a utilisé différents hyperparamètres. La fonction de coût utilisée est la $sparse\_categorical\_crossentropy$ avec comme metrics "accuracy". On lance ensuite l'entrainement avec nos données placées auparavant dans des tenseurs et avec des epochs et un $batch\_size$ de taille 10.
Cependant, nous faisons face à du suraprentissage. Le problème que nous essayons de résoudre pendant l'entrainement consiste à établir 450 000 paramètres avec environ 2500 exemles.
On trouve une $val\_accuracy$ partant de 0.5 jusqu'à 0.75 et une loss qui diminue fortement ce qui reste correct avec du suraprentissage.

5.2 Correction du surapprentissage

Pour lutter contre ce surapprentissage, on utilise la technique de l'augmentatiion de la base de données.
On introduit un objet ImagedataGenerator qui va appliquer de nouvelles transformations aux images de notre base de donées.

Ci dessous l'objet introduit avec ses différents paramètres :

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    rescale=1.0/255.0,
    horizontal_flip=True)
```

FIGURE 5 – Objet ImageDataGenerator et ses paramètres

On recrée ensuite le même modèle avec les mêmes couches et leurs paramètres.
Cependant pour l'entrainement, on utilise cette fois une $sparse\_categorical\_accuracy$ et le modèle est maintenant lancé avec le nouvel objet $train\_datagen$. Pour le suraprentissage, on prend des epochs de taille 100.
On trouve alors une meilleur accuracy qui peut aller jusqu'à 0.85 environ.

6 Analyse de nos résultats

6.1 Analyse quantitative

6.1.1 Précision globale et par classe

Pour bien analyser le modèle et comprendre ses erreurs il faut tout d'abord regarder la précision pour chaque classe puis la précision globale. On a alors récuperer la précision, le rappel et le f1-score pour chaque classe.

Voici notre tableau des précisions.

La précision globale est de 70.5%

Classe	Précision	Rappel	F1-score
Rugby	0.87	0.60	0.71
Tennis	0.90	0.90	0.90
Foot	0.57	0.89	0.69
Hockey	0.93	1.00	0.97
Basket	0.87	0.87	0.87

FIGURE 6 – Precision, rappel et f1-score

On remarque alors qu'on obtient une précision globale de 70.5%. Le tableau nous donne des bonnes précisions pour à peu près tout les sports sauf le foot où la précision est seulement de 57%. La précision permet de répondre à cette question : Sur tous les enregistrements positifs prédits, combien sont réellement positifs ? Le rappel permet de répondre à : Sur tous les enregistrements positifs, combien ont été correctement prédits ? L'indice de rappel est quant à lui plus faible pour le rugby avec une valeur de 0.60.

6.1.2 Matrice de confusion

La matrice de confusion est un élément très important dans l'analyse de nos résultats car elle va pouvoir donner plus de précision sur la performance de notre modèle.

La matrice de confusion est en quelque sorte un résumé des résultats de prédiction pour notre problème de classification. Elle va comparer les données réelles pour une variable cible à celles prédites par un modèle. Les prédictions justes et fausses sont révélées et réparties par classe, ce qui permet de les comparer avec des valeurs définies.

Ci dessous, on peut voir notre matrice de confusion pour l'ensemble de validation avec le meilleur modèle qu'on a.

```
from sklearn.metrics import confusion_matrix

y_pred = model.predict(x_val/255)
y_max = np.argmax(y_pred, axis=1)

labels = ['Rugby', 'Tennis', 'Foot', 'Hockey', 'Basket']
cm = confusion_matrix(y_val, y_max)

print(cm)
```

```
[[26  0  3  0  1]
 [ 1 27  2  0  0]
 [ 5  3 22  0  0]
 [ 2  0  0 27  1]
 [ 2  0  1  0 27]]
```

FIGURE 7 – Matrice de confusion pour nos 5 classes

La matrice de confusion se lit tout d'abord sur la diagonale. En effet la valeur des éléments diagonaux indique si chaque classe a bien été prédite correctement. On le voit bien à travers notre matrice de confusion où les élements diagonaux révèlent une bonne prédiction des classes, environ 26 ou 27 sur 30.

Cependant on remarque qu'il existe certaines confusions entre différentes classes. Par exemple, l'élement à la 3ème ligne de la 1er colonne nous montre bien que le modèle peut parfois confondre des images de foot avec celle du rugby et inversement à la 1er ligne 3eme colonne.

Cette matrice nous montre aussi qu'il y a rarement voire jamais de confusion avec le Hockey. Cela peut s'expliquer par un environnement de jeu très différent des autres sports mais aussi dans la posture et la tenue des joueurs des hockey.

6.2 Analyse qualitative

6.2.1 Exemples négatifs tirés de la base de test

Nous avons ensuite récupéré les différents cas d'erreurs pour essayer de comprendre au mieux ce que l'algorithme d'apprentissage prédit mal.

FIGURE 8 – Différents cas d'erreurs

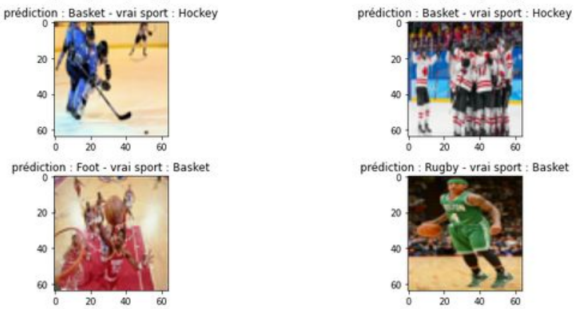

FIGURE 9 – Différents cas d'erreurs

On remarque bien que l'apprentissage est biaisé lorsque l'image comporte un filet. Il se trompe donc facilement entre le foot et le tennis. Les images où sont représentées plusieurs personnes trompent aussi l'algorithme ce qui explique les confusions entre le rugby et le foot.

6.2.2 Analyse des erreurs et réflexions sur les pistes d'amélioration

Le modèle que l'on a utilisé permet d'avoir une accuracy max de 86% ce qui est correcte pour une base de données d'environ 2000 images. Certaines erreurs peuvent s'expliquer par la posture des joueurs trompant ainsi le modèle. Les images où l'on ne voit ni les poteaux ni le ballon sont aussi mal prédites par l'algorithme qui n'arrive pas bien à détecter le sport. Les photos d'équipes peuvent aussi tromper l'algorithme.

Nous pouvons améliorer nos prédictions en prenant un modèle plus performant. On a voulu commencer la solution de transfert learning avec le reseau VGG-16. On peut aussi améliorer notre modèle en ajouter beaucoup plus d'images. 2000 images ne suffisent pas pour avoir un modèle qui prédit à environ 95%. Nous pouvons aussi améliorer le cas où l'image ne représente qu'un seul joueur en analysant les différentes postures pour éviter au maximum les confusionS.

7 Conclusion

Ce projet nous a permis grâce aux connaissances acquises en TP de classifier 5 sports grâce à une base de donnée d'environ 2 000 images. Nous obtenant des résultats cohérents qui nous permettent d'obtenir avec notre meilleur modèle par réseau convolutif de base une accuracy de 86%. Nous avons tout de même essayé une autre méthode par transfert learning avec le réseau VGG-16 mais sans résultats cohérents.
Cependant, il persiste certaines confusions principalement entre le foot et le rugby comme nous le pensions dès le début du projet.

Une manière d'améliorer notre projet serait d'analyser les différentes postures des joueurs pendant un match. En effet, un basketteur ou un joueur de hockey n'auront pas les mêmes mouvements. De même pour le joueur de rugby qui sera souvent au sol contrairement au joueur de foot(sauf Neymar). Cela évitera les confusions et améliora les performance du modèle.