Rapport IA - Python

> Notre problème est d'identifier les différentes races de chien à partir d'images données. Notre intelligence artificielle reposera donc sur sa capacité à identifier les différences races sur un échantillon donnée.

1 - Préparation de nos données

Notre dataset est composée de plusieurs dossiers de races de chiens comportant eux même plusieurs centaines d'image de chiens. L'intégralité de ces images est au format jpg. Le dataset de départ comportait plusieurs anomalies qui risqueraient de compromettre la fiabilité de notre modèle. Nous avons fait le nécessaire pour remédier à ce problème en supprimant les images très peu visible ou trop flou. Les dossiers des races de chiens ont également été renommés avec un nom plus simple et non des chiffres comme c'était le cas dans le dataset de départ. Après avoir traité ces données bruts, nous avons redimensionner toutes les images en une taille standard, ce qui est essentiel ensuite pour le bon fonctionnement de notre modèle.

2 - Le fractionnement de nos données

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=8)
```

- D'abord que représente ces données ? Ce sont des données d'entrées et de sorties que nous donneront ensuite à notre modèle pour apprendre et tester. Les X sont les données d'entrée et le y les données de sortie (voir documentation pour plus de détail). On sépare notre jeu de donnée pour que le modèle s'accorde aux données d'entrainement et de test. On effectue cette partie afin d'éviter l'apprentissage "par coeur".
- Concernant le paramètre de random_state, nous avons décider de lui donner 8. La valeur en elle-même n'est pas importante, le plus important est qu'il faut lui donner un entier positif afin de représenter une meme sortie pour chaque appel de fonction. Cela évite de fractionner aléatoirement nos valeurs à chaque fois. Sans cela, nos tests ne serait pas reproductibles et donc pas représentable.
- Comme paramètre de test_size nous avons choisi de mettre 0.20. En effet, en général on découpe nos données en 80% d'apprentissage et 20% de test.

3 - Le choix de notre modèle

- Pour commencer, sommes nous dans le cas d'une classification ou d'une régression ? Concernant notre intelligence artificielle c'est un problème de classification. En effet, si l'on veut déterminer si une photo de chiens appartient à tel ou tel race de chiens, on effectue une classification puisque le type de sortie est une catégorie, ici une race de chien
- Concernant le model en lui-même, nous avons crée un modèle séquentiel avec l'aide de tensorflow. A ce modèle, nous avons appliqué plusieurs couches avec des tailles de filtres différents afin d'obtenir de meilleurs résultats. Nous avons remarqué que lors de l'ajout de chaque couche, si on effectue un remplissage (padding="Same") plutot qu'un remplissage (padding="valid"), la pahse d'apprentissage est beaucoup plus rapide et les résultats assez bons.
- Pour le deuxième model que nous avons crée, nous avons décidé d'ajouter l'opération MaxPooling2D() après chaque filtre ajouter. Auparavant, nous avions remarqué que cette opération joué un rôle important dans l'apprentissage de notre modèle 1. Ce second model diffère du premier que nous avons crée, en effet, l'accuracy du second model est d'autant plus élevé, et le temps d'apprentissage est fortement réduit (divisé par 10 pour chaque epochs à comparé du model 1), mais les résultats ne sont pas aussi bons que notre premier model.

4 - La compilation de notre modèle

```
model.compile(  optimizer=Adam(learning_rate=0.0001),
                                loss='categorical_crossentropy',
                                metrics=['accuracy'])

history = model.fit_generator(datagen.flow(X_train,y_train,
batch_size=nb_images), epochs = epochs, validation_data = (X_test,y_test))
```

- Nous avons choisi comme optimizer Adam, car c'est avec celui-ci que nous avons eu les meilleurs résultats après avoir testé d'autre optimizer. L'optimizer a pour rôle de corriger l'algorithme de notre IA. Le but de notre optimizer n'est pas d'avoir une erreur nulle mais d'avoir l'erreur la plus faible possible. Comme fonction de minimisation nous avons testé 'categorical_crossentropy', 'binary_crossentropy' et nous nous sommes aperçu que la fonction la plus fiable est 'categorical_crossentropy'. Nous obtenons de biens meilleurs résultats après plusieurs epochs, ce qui parrait logique car nous sommes dans le cas de catégories de races.

5 - Les résultats et graphiques

Nous pouvons constater que notre IA répartit ses ressources dans ses entraînements et donc on peut extrapoler le fait que plus nous ferons “d’epoch” plus notre modèle sera entraîné et qu’en conséquence nos résultats seront plus précis. Le but étant d’avoir des tests précis rapidement nous permettant, dans la situation idyllique finale, de ne plus entraîner notre IA et d’obtenir une quasi certitude lors des tests. Pour l'instant, nos résultats ne sont pas assez poussés mais nous avons tout de même obtenons des résultats plutôt convaincants allant jusqu'à 60 % de précision, avec seulement quelques epochs. Sur une plage de test de 36 images, notre intelligence artificielle a déjà réussi à reconnaitre la race de 20 chiens.