UNIVERSITÉ DU QUÉBEC À MONTRÉAL

Rapport TP2 PRÉSENTÉ À :

JEAN MASSARDI

Dans le cadre du cours:

Gestion et analyse de données - INF5081

Par :
Alexandre Billereau : BILA14069808
Dalil Touati : TOUD89080108
Maha Ghozzi : GHOM04529705
Maroua Bach : BACM25549904

12-08-2022

1. Introduction:

Pour répondre à la problématique de ce TP, qui est de distinguer 10 espèces d'animaux de personnes déguisées en fursuit , nous avons entraîné plusieurs réseaux de neurones type CNN. Pour ce faire, nous avons tenté le maximum de paramètres et d'optimisations tant sur le modèle des couches de notre CNN que nos paramètres d'entraînement ou encore dans le choix du bon traitement des images. Nous allons dans ce document rapporter au mieux les expériences réussies ou ratées de notre résolution du problème.

2. Les différents paramètres et architectures:

Dans un premier temps, nous avons hésité sur la façon dont nous allions traiter le cas des fursuits au vu de la différence de quantité entre les différentes databases. Une première solution a été de faire de l'augmentation de données sur ce dataset mais celà revenait à le faire que sur un seul dataset et donc potentiellement provoquer une perte de performance pour les animaux. Nous avons finalement décidé de récupérer 200 images de chaque dossier et de passer le tout à 2000 images grâce à la librairie Augmentator qui permet d'effectuer des rotations, des zooms aléatoires sur les images ou encore de retourner les images.

Par rapport à la taille des images, nous avons décidé de les normaliser en 128 par 128 pour éviter que certaines soient trop lourdes en mémoire et pour s'assurer de l'uniformité des données.

Pour le choix des batchs, nous nous sommes adaptés à la limite de mémoire imposées par Google collab qui restait le plus pratique pour nous afin de nous coordonner sur les avancées et les modifications futures. Nous avions donc des batchs de 15 pour un total de 1174 itérations et 13 epochs qui a été le plus efficace après avoir tenté avec 5 , 30 puis 15. Quant au choix du learning rate, nous n'avons pas pu trouver de valeur adéquate donc nous avons laissé la valeur par défaut.

Après avoir essayé l'optimiseur RMSProp dont les résultats au vu de notre manque de test concernant le learning rate, n'ont pas été convaincants. Nous avons choisi d'utiliser Adam pour sa facilité à paramétrer et son efficacité sur le type de données.

Premiere model

```
[ ]  num_classes = 11

     model = tf.keras.Sequential([
       #On met les images entre 0 et 1
       tf.keras.layers.Rescaling(1./255),
       #fin 0 a 1

       #todo couche de convolution avec activation expo
       tf.keras.layers.Conv2D(32, 3, activation='relu'),
       tf.keras.layers.MaxPooling2D(),
       tf.keras.layers.Conv2D(32, 3, activation='relu'),
       tf.keras.layers.MaxPooling2D(),

       tf.keras.layers.Flatten(),
       tf.keras.layers.Dense(128, activation='relu'),
       tf.keras.layers.Dense(num_classes)
     ])
```

Le premier modèle fut un test simple de CNN basique comme on le trouve dans la documentation de tensorflow.

```python
num_classes = 11

model = tf.keras.Sequential([
  #On met les images entre 0 et 1
  tf.keras.layers.Rescaling(1./255),
  #fin 0 a 1

  #augmentation des données
  tf.keras.layers.RandomFlip("horizontal",
                             #par puissance de 2
                             input_shape=(256,
                             256,
                             3)
  ),
  tf.keras.layers.RandomRotation(0.1),
  tf.keras.layers.RandomZoom(0.1),
  #fin augmentation des données

  #todo couche de convolution avec activation expo
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),

  #Principe de l'abandon
  layers.Dropout(0.2),
  #fin abandon

  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes)
])
```

Par la suite, pour le second modèle, nous avons essayé d’affiner l'apprentissage en faisant de l'augmentation directement dans le réseau de neurones avec des couches d'augmentation et de dropout. La couche de dropout a été particulièrement efficace pour éviter l’overfitting avec un pourcentage de 20% qui plus tard passera à 25%. Nous n'avions à ce moment pas la bonne méthode. En effet notre dataSet n'augmente que la classe fursuit et donne de mauvais résultats sur l'entraînement. Alors nous avons décidé de faire l’augmentation de données en amont voir troisième modèle.

Troisième model

```
num_classes = 11

model = tf.keras.Sequential([
  #On met les images entre 0 et 1
  tf.keras.layers.Rescaling(1./255),
  #fin 0 a 1

  #todo couche de convolution avec activation expo
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),

  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),

  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),


  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  #Principe de l'abandon
  layers.Dropout(0.2),
  #fin abandon

  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes),
])
```

Le troisième modèle ci-dessus lui fut une des première réussite en terme “d’accuracy” un premier 40% fut atteint. La joie de fou quand on donnait une photo de vache et que le réseau sortait “muca”. En effet après l’augmentation

en amont et non dans le CNN et l'ajout d'une couche de convolution + relu et de pooling fut un bon début.

Combinaison du quatrième et cinqième essaie

```
num_classes = 11

model = tf.keras.Sequential([
  #On met les images entre 0 et 1
  tf.keras.layers.Rescaling(1./255),
  #fin 0 a 1

  #todo couche de convolution avec activation expo
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Dropout(0.25),

  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Dropout(0.25),


  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Dropout(0.25),


  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  #Principe de l'abandon
  layers.Dropout(0.25),
  #fin abandon

  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes, activation="softmax"),
])
```

Enfin pour finir, le quatrième modèle fut de jouer avec les paramètres en essayant la fonction d'activation exponentiel dans les couches cachées. Ce fut un échec, un retour a 30% d'accuracy. De plus, nous avions rajouté des couches de drop-out après chaque couche de convolution. Finalement le cinquième/sixième (en vérité) modèle fut le bon une accuracy de 62% le einstein de ses grand frère CNN. Ce modèle était en fait simplement des couches de convolution et de pooling supplémentaires , les dropout a 0.05 de

plus que les précédents et la fonction d’activation SoftMax sur la couche de sortie.

3. Conclusion:

En conclusion, après nos efforts de test et les longues heures d'entraînement, nous avons dépassé nos espérances de 60% avec notre dernier réseau de neurones qui a atteint les 62%. Nous pouvons dire que nous sommes satisfaits de notre performance et de voir enfin un résultat encourageant.