Report 01

Majurca - PFEE

Abstract

Ceci est le premier rapport de nos progrès sur la tâche de classification d'images pour le tri des déchets plastiques qui nous a été confiée par la société Majurca. Étant donné qu'il s'agit du premier rapport, il couvrira tout ce qui a été fait depuis le début du projet. Ceci sera fait de manière chronologique. Pour donner un aperçu de ce qui a été fait jusqu'à présent; Nous avons d'abord choisi le modèle CNN pré-entraîné VGG16 pour l'apprentissage par transfert. Puis, comme notre premier jeu de données était un sous-ensemble de sept catégories contenant chacune 10 images, nous avons rapidement été confrontés à des problèmes de surajustement. Pour combattre ce problème, nous avons utilisé la régularisation en ajoutant une couche de Dropout supplémentaire au modèle VGG16. Cela a permis une augmentation significative de la précision de validation (détaillée dans les sections suivantes). À ce stade, nous avons eu accès à l'ensemble complet de données d'environ 64 000 images. En testant le modèle précédent sur le jeu de données complet, nous avons observé une disparité significative entre la précision obtenue sur le jeu de données initial et celle obtenue sur le jeu de données complet. Après une inspection plus approfondie des données, nous avons conclu que cela était dû à la prévalence plus élevée d'images prises sous différents angles et avec différentes luminosités. Pour l'instant, nous avons entraîné le même modèle VGG16 modifié sur un sous-ensemble d'images présentant toutes des angles et des luminosités similaires, et nous avons obtenu une précision de validation moyenne de 89%. Actuellement, nous testons plusieurs méthodes afin d'exploiter une plus grande partie de l'ensemble des données. Par exemple, en empilant des modèles individuels entraînés sur des images ayant toutes le même angle et la même luminosité ou en passant à un autre modèle pré-entraîné capable d'atteindre une plus grande précision malgré cette disparité.

Christopher Diamana Lutete
Jose A. Henriquez Roa

September 30, 2021

1 Transfer Learning

Après avoir reçu le jeu de données initial, nous avons d'abord essayé d'entraîner plusieurs modèles pré-entraînés, comme c'est habituellement le cas pour les tâches de classification d'images. Parmi ceux qui ont été testés, le modèle qui a donné les meilleurs résultats pour le jeu de données donné était le VGG16.
Le VGG16 est un CNN (Convolutional Neural Network) dont l'architecture est relativement simple. À ce jour, il est l'un des CNN les plus performants dans la tâche de classification d'images sur le jeu de données Imagenet, qui est un jeu de donnée contenant 1000 catégories avec un total de 1 341 167 images. Sans aucune modification, nous avons pu atteindre une précision de validation de base de 74% en moyenne, comme le montre l'image suivante qui représente sa courbe d'apprentissage sur 300 époques.

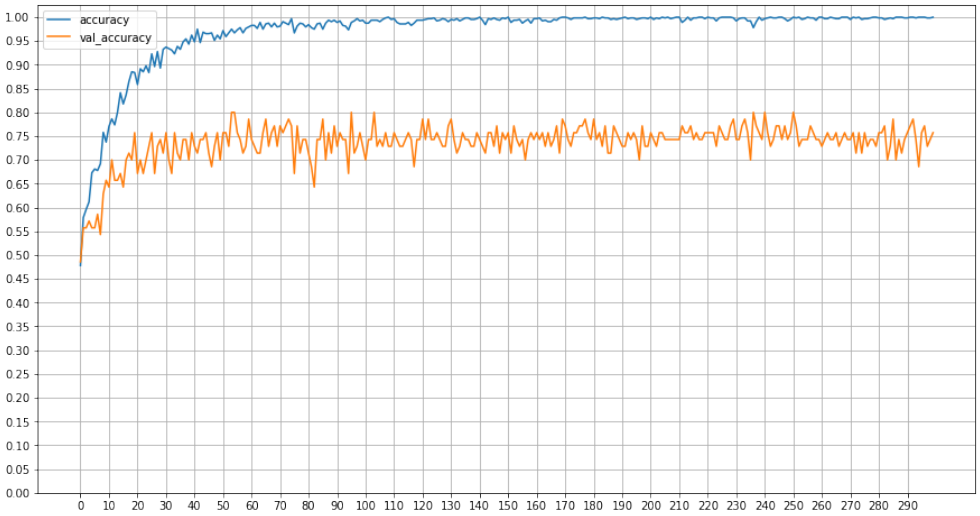

Figure 1: Learning curve of model 01 over 300 epochs

Comme on peut le constater, il y avait un écart considérable entre la précision obtenue sur l'ensemble d'apprentissage et celle obtenue sur l'ensemble de validation. Cela signifie que le modèle sur-exploitait les données d'apprentissage et n'était pas en mesure de généraliser à des données qu'il n'avait pas vues auparavant. Pour cela, nous avons utilisé la régularisation.

2 Regularization

La régularisation est un terme utilisé pour désigner plusieurs méthodes utilisées pour lutter contre l'overfiting. Un modèle surajusté est un modèle qui mémorise les données d'apprentissage au lieu de les généraliser afin de pouvoir effectuer la même tâche sur des données qu'il n'a pas vues. L'overfitting se produit souvent lorsque le modèle est trop complexe pour la tâche donnée (un modèle qui a trop de paramètres entraînables). La régularisation implique soit de simplifier le modèle en modifiant son architecture, soit de le contraindre d'une manière ou d'une autre afin de réduire la probabilité qu'il mémorise les données d'apprentissage.
Dans notre cas, ayant utilisé l'apprentissage par transfert, la suppression de couches du modèle VGG16 présentait un inconvénient majeur : cela signifiait que les paramètres pré-entraînés qu'il avait appris pour la tâche imagenet seraient perdus. Nous avons donc opté pour la méthode consis-

tant à contraindre le modèle pendant la phase d'apprentissage. La méthode particulière que nous avons utilisée est appelée Dropout. La méthode de Dropout consiste à désactiver l'apprentissage pour une partie aléatoire du réseau à chaque époque de la formation. Cela rend difficile pour le réseau de mémoriser certains aspects des données d'apprentissage en le forçant à "regarder l'image entière" en quelque sorte. Après avoir ajouté une couche de régularisation, nous avons pu obtenir une précision de validation moyenne de 87%.

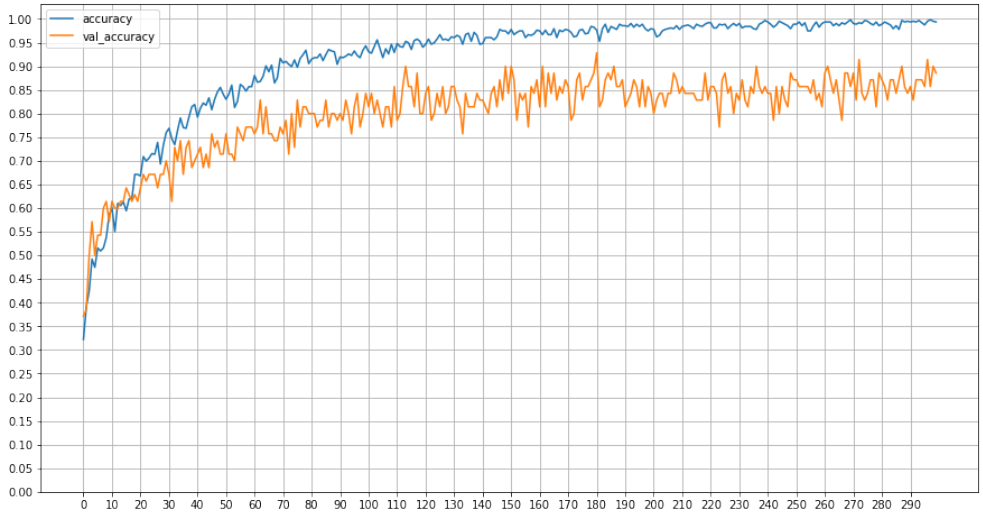

Figure 2: Learning curve of model 01 over 300 epochs

3 Complete Dataset

Jusqu'à présent, nous avions travaillé sur un sous-ensemble du jeu de données . Après avoir eu accès au jeu de données complet et avoir entraîné notre modèle sur celui-ci. Nous avons découvert que la précission a diminué de manière significative. En poursuivant notre investigation, nous avons remarqué que le type d'images du jeu de données avait considérablement changé. Dans l'ensemble, nous avons découvert quatre types d'images.

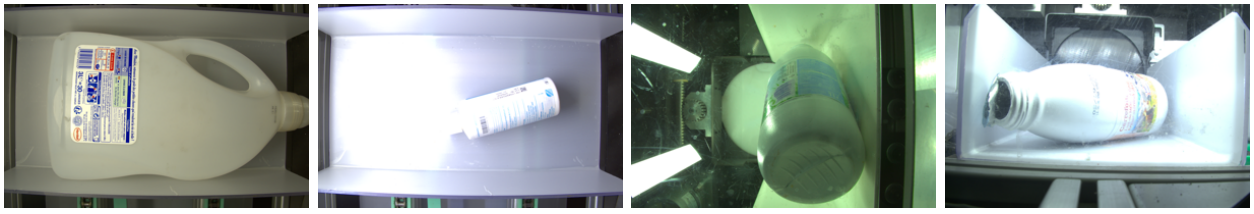

Figure 3: Different types of images

Images prises du haut et avec un éclairage uniforme, images prises du haut et avec un éclairage plus élevé. Images prises horizontalement depuis le côté et images prises verticalement depuis le côté. Vu que le traitement de cette disparité dans les données nécessitera un travail supplémentaire nous avons premièrement testé le précédent modèle VGG16 sur un sous-ensemble d'images prises sur le côté et horizontalement et avons réussi à obtenir une précision de validation moyenne de 89%. Et à ce stade, nous testons actuellement plusieurs méthodes afin d'exploiter le plus de données possible de l'ensemble de données.

4 Final model architecture

Au moment de la rédaction de ce rapport, l'architecture actuelle du modèle que nous utilisons est celle d'un VGG16 modifié, avec quatre couches supplémentaires. Deux de ces couches sont des couches Dense, et les deux autres sont une couche de Pooling et une de Dropout (comme détaillé précédemment). L'architecture complète est la suivante, où Conv2D fait référence aux couches convolutionnelles et MaxPooling2D et GlobalAveragePooling2D font référence aux couches de pooling.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 512)               262656
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 3591
=================================================================
Total params: 14,980,935
Trainable params: 14,980,935
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Modified VGG16 Model