ARN Practical Work 5

Rémi Ançay & Lucas Charbonnier

# Introduction

Notre projet consiste en la création d’un modèle de reconnaissance d’images permettant de différencier un petit échantillon des boissons disponibles au Chillout de la HEIG-VD.

## Problématique

Le Chillout doit trier les récipients des boissons en fonction de la marque. Certains fournisseurs demandent une consigne lors de l’achat de boissons dans des conteneurs en verres. Il est donc important que les récipients soient triés afin que le Chillout ait un minimum de pertes dues à ces consignes. Chaque semaine, les bénévoles doivent trier le verre et faire attention à séparer les bouteilles sur lesquelles il y a une consigne.

Couplé à un système de tri automatisé, notre modèle permettrait de réduire considérablement le travail des bénévoles du Chillout en séparant automatiquement le verre qui doit être jeté du verre qui doit être retourné.

Ce projet consiste en la création d’une POC (Proof Of Concept) pour voir s’il est possible de concevoir un tel modèle.

# Collecte des données

## Choix des boissons

Étant donné la grande quantité de boissons disponibles au Chillout, nous avons décidé de limiter notre projet à 5 d’entre-elles :

* El Tony Mate (bouteille en verre blanc, 33cl)
* Hoegaarden (bouteille en verre brun, 25cl)
* Brewdog Hazy Jane (canette, 5dl)
* Fizzy Myrtille (bouteille semi-opaque, 35cl)
* Swaf (bouteille en verre brun, 33cl)

Nous aurions pu collecter des données sur un plus grand nombre de boissons mais cela aurait pris beaucoup de temps et notre but est principalement de faire une proof-of-concept plutôt qu’un modèle exhaustif. Nous avons choisi un ensemble de boissons ayant des caractéristiques permettant facilement de les identifier. Typiquement, la Fizzy a un système d’ouverture unique, la Brewdog et la seule boisson sous forme de canette et la Hoegaarden est une bouteille un peu plus petite que les autres.

## Collecte des données

Il est possible d’augmenter les données avec divers techniques (rotations, inversions horizontale/verticale, application de filtres de couleur, zooms) mais il n’est pas possible de créer une information n’existant pas ; nous avons donc fait attention à prendre des photos des boissons sous tous les angles possibles.

En pratique, nous avons pris 10 angles différents par boisson :

* 2 photos de chacun des quatre « côtés » (rotation de 90°)
* Une photo de dessus
* Une photo de dessous

Par exemple, pour « El Tony Mate » nous avons pris les photos suivantes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

De plus, nous avons également pris le même ensemble de photos mais à un endroit différent (à l’extérieur), dans le but de dissuader le modèle de reconnaître des caractéristiques appartenant au décor, plutôt qu’aux boissons elles-mêmes.

Nous disposons donc de 20 photos pour chacune des 5 boissons, soit un total de 100 photos. Toutes les photos ont été prises sur le même appareil, un Samsung Galaxy A72.

# Préparation des données

## Pré-traitements des images

Nos images brutes ont une résolution de 3472x3472 pixels. Elles sont donc très volumineuses, c’est pourquoi nous avons commencé par les redimensionner en 224x224 pixels. Cela permet de diminuer le volume, le temps de traitement et également de les préparer pour être ingérées par le modèle puisque c’est la dimension attendue par MobileNetV2.

Nous avons également appliqué un « Rescaling » afin de normaliser les valeurs des pixels entre 0 et 1, au lieu de 0 et 255.

## Augmentation des données d’entraînement

Afin d’améliorer la capacité de généralisation du modèle, nous avons utilisé un certain nombre d’augmentations provenant de la bibliothèque Keras. Les augmentations sont appliquées aléatoirement, pas systématiquement à toutes les images. Voici les augmentations que nous avons utilisées :

* Des rotations entre -180 et 180°
* Changement de contraste
* Zooms divers

Cela nous permet de tripler la taille du dataset d’entraînement. Nous avons remarqué empiriquement qu’aller plus loin que 2 à 3 fois la taille du dataset original ne permettait pas une meilleure généralisation et que cela ne valait pas le temps d’entraînement supplémentaire.

# Création du modèle

Afin d’éviter des temps d’entraînements extrêmement longs, nous avons pris une approche de « transfer learning ». Nous avons utilisé le modèle MobileNetV2 entrainé sur le dataset ImageNet et avons gelé toutes les couches du modèle avant d’ajouter deux couches cachées « Dense ».

Pour évaluer la capacité de généralisation et la performance du modèle lors de la recherche des meilleurs hyperparamètres, nous avons utilisé une validation croisée à 5 plus (5-Folds CV).

## Paramètres finaux

### Données, pré-traitements, augmentations

* 100 images, dont 20% réservées pour le test
* 80 images d’entraînement, redimensionnées en 224x224 et augmentées deux fois pour un total de 240 images d’entraînement au total.
* Augmentations : rotations, changements de contraste et zoom aléatoire

### Paramètres du modèle

* 2 couches cachées composées de 20 et 10 neurones
* 20% de dropout après chaque couche cachée
* 40 epochs
* Validation croisée à 5 folds

## Problèmes rencontrés

### Problème de labélisation des données d’entraînement augmentées

Au début, nous obtenions de très mauvais résultats lors de l’entraînement de notre modèle. Les scores d’entraînement et de validation étaient aléatoires et ne s’amélioraient pas du tout au fil des epochs.

En fait, les données d’entraînement **augmentées** étaient mal labélisées car nous utilisions incorrectement une fonction Numpy permettant de répéter les labels pour les exemples augmentés. Une fois cela corrigé, l’entraînement du modèle fonctionnait beaucoup mieux et nous avons pu commencer à rechercher les meilleurs hyperparamètres.

### « Accuracy » de validation meilleure que l’entraînement

En essayant de rechercher les meilleurs paramètres, nous avons remarqué plusieurs fois que les scores sur les données de validation dépassaient fréquemment les scores sur les données d’entraînement. Cela se produit quand les exemples de validation sont « trop simples » par rapport aux données d’entraînement.

En l’occurrence, nous avions une distribution similaire du nombre de classes et des types d’angles des photos entre les données d’entraînement et de validation, donc il ne s’agissait pas d’un déséquilibre entre les données d’entraînement et de validation.

Notre meilleure hypothèse est que cela est dû à la grande quantité d’images augmentées. Nous avons utilisé des augmentations assez complexes, combinant parfois un zoom et une rotation, ce qui a pour effet de rendre la prédiction plus difficile et de forcer le modèle à mieux généraliser. En d’autres termes, nos augmentations rendaient les prédictions plus difficiles sur les données d’entraînement que sur les données de validation.

### Temps d’entraînement

Notre modèle prend un grand nombre d’epochs pour atteindre de bons résultats. De plus, nous avons un assez grand nombre d’images d’entraînements (~300 avec les augmentations). Le modèle prend donc un certain temps à être entrainé (entre 7 et 15 minutes pour faire toutes les validations croisées) et cela a rendu fastidieux la recherche des meilleurs hyperparamètres. Nous n’avons pas utilisé d’algorithme de recherche mais si nous l’avions fait, la recherche aurait pris au moins 1 nuit entière.

# Résultats obtenus

## Entraînement du modèle, matrice de confusion et scores

Une image contenant Tracé, ligne, capture d’écran, texte

Description générée automatiquement

Voici les graphiques qui présentent les performances du modèle sur les folds d’entrainement et de validation durant l’entrainement.

Une image contenant capture d’écran, texte, diagramme, Rectangle

Description générée automatiquement

Après réentrainement du modèle avec l’intégralité des images, voici la matrice de confusion des données de test. (La seul erreur ici est une photo prise depuis le dessous de la bouteille. Nous expliquons plus en détails cela dans la partie « Donnée de test » qui viens ensuite.)

Le modèle obtient donc un F1 score de 0.9555 et une accuracy de 0.95 se qui démontre que notre modèle est très bon.

## Analyse avec GradCam++

Après analyse des images générée avec GradCam++, nous pouvons voir plus en détails certains problèmes.

Déjà, on comprend pourquoi les photos prise depuis le dessus ou le dessous ont du mal à être classifiée. En effet le modèle accorde beaucoup d’importance au fond derrière la bouteille et non à elle comme le montre les images 1 et 2 suivantes.

Cependant, le modèle se concentre bien sur la bouteille dans la plupart des cas, car elle est souvent au centre de la partie la plus activée de la photo comme le montre l’image 3

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Image 1 | Image 2 | Image 3 |

## Données de test

### Images de dessus/dessous

Notre modèle final obtient un très bon score sur quasiment toutes les données de test, à l’exception des images de dessus/dessous des boissons. En effet, même pour un humain il n’est pas aisé de reconnaître de quelle boisson il s’agit si on ne voit que le dessous de la bouteille et notre modèle a beaucoup de peine avec ces cas spécifiques.

Voici un exemple typique de ce problème. Ici, le modèle prédit la boisson Tony Mate, au lieu de Fizzy.



Cependant, même un être humain aurait de la peine à identifier duquel de quelle boisson il s’agit.

## Test grandeur nature avec l’application

Une fois le modèle déployé sur l’application, nous sommes allés au Chillout pour faire un test grandeur nature. Nous obtenons de bons résultats lorsque l’appareil est proche du sujet (autour des 90% de précision) mais, au-delà d’un mètre, le modèle détecte assez mal, en particulier si le fond n’est pas uniforme.

En résumé, notre modèle fonctionne bien de près mais mal de loin.

# Conclusion