



**PROJET CARREFOUR
CHALLENGE DE RECONNAISSANCE
D'IMAGES**

Projet de MAP569

6 avril 2021

Paul MATHIVON, Victor SAILLANT, Geoffrey SAUNOIS
promotion 2018



TABLE DES MATIÈRES

1 Présentation du problème	4
1.1 Présentation	4
1.2 Cadre retenu	5
1.2.1 Calcul des labels	5
1.2.2 Détermination de E_{min}	6
1.3 Pre-processing des images	7
2 Modèles retenus	10
2.1 Transfer Learning	10
2.2 VGG-16	10
2.3 DenseNet161	11
2.4 ResNet50	11
3 Résultats	13
3.1 Comparaisons des modèles	13
3.2 Test du réseau sur des images en dehors du set	13
3.3 Pistes d'améliorations vers l'objectif initial	15
3.3.1 Enregistrement des sorties de la base	15
3.3.2 Frigo connecté et data augmentation	16
4 Conclusion	18
Références	19

INTRODUCTION

Ce travail de reconnaissance d'image est proposé par l'entreprise Carrefour, en collaboration avec le cours de *Machine Learning II* de l'École polytechnique, identifié *MAP569*.¹

Le but de ce projet était d'arriver à reconnaître, à partir de la photo d'un produit prise par un consommateur, le produit du catalogue Carrefour associé. Pour mener à bien cette tâche, nous disposions de la base de données regroupant des photos de tous les produits proposés à la vente, chaque produits étant pris sous différents angles.

Le principal défi de ce challenge était la taille très conséquente de la base de données, qui contenait presque **100Go d'images** pour un total de 22000 références. Ceci impose de mettre en place des architectures de données pertinentes, capables de supporter un nombre de données important et d'évoluer à grande échelle.

Une des applications les plus prometteuses de ce projet serait dans la création d'une application pour **frigos connectés**, où une caméra à l'intérieur du frigo serait capable de reconnaître les aliments encore présents dans le frigo, et ainsi de **proposer au consommateur une liste de courses personnalisée**.

1. Les codes développés à l'occasion de ce challenge sont disponibles dans le dossier git en ligne du projet : github.com/bison-fute/Carrefour-Image-Recognition-Challenge.

1

PRÉSENTATION DU PROBLÈME

1.1 PRÉSENTATION

L'objectif dans ce challenge était de créer un modèle pouvant reconnaître, à partir de la photo d'un produit prise par un consommateur, le produit du catalogue Carrefour associé, comme représenté en figure 1.

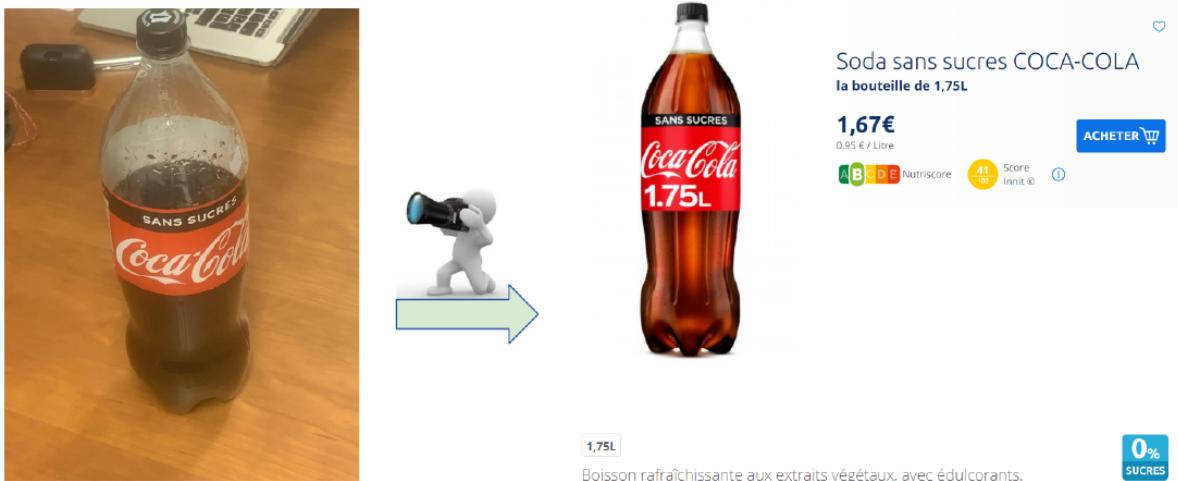


FIGURE 1 – Exemple de reconnaissance d'image

Cependant, nous nous sommes rapidement rendu compte que les ressources à notre disposition n'étaient pas suffisantes pour l'entraînement d'une telle architecture sur une quantité d'images aussi imposante que celle fournie (plus de 110Go d'images). En particulier, il nous était difficile de satisfaire en même temps les deux contraintes de **précision** (identifier un produit exact à partir de sa photo) et d'**échelle** (arriver à bien cette tache pour les 22000 produits de la base).

Afin de travailler sur l'ensemble des données fournies, nous avons donc choisi d'alléger la première contrainte, en regroupant les articles par **catégories**. La problématique business devenait alors la suivante : à partir d'une photo d'un produit Carrefour (par exemple la bouteille de Coca-Cola représentée en figure 1), réussir à identifier à quelle catégorie il appartient ("boissons sucrées à base de plantes" dans ce cas), ceci afin de **proposer au consommateur des produits similaires de la même catégorie** : Pepsi, Oasis, Fanta, Orangina.

1.2 CADRE RETENU

1.2.1 • CALCUL DES LABELS

Afin de traiter ce problème, nous avions également accès aux méta-données associées à chaque produit du catalogue. En plus des code-barres (faisant office d'identifiants pour les produits, et permettant de relier ces informations aux images correspondantes), ces méta-données contenait également les catégories auxquelles appartenaient chaque produit.

Les catégories sont **organisées sous forme d'arbre** : une catégorie principale contient plusieurs catégories secondaires, qui elles-mêmes contiennent plusieurs catégories tertiaires, etc... En itérant sur toutes les données, nous avons obtenus un arbre similaire à celui représenté en figure 2. Ainsi, plutôt que d'essayer de relier une photo au produit exact correspondant, ce qui posait problème comme vu en section 1.1, la première approche que nous avons considérée a été de regrouper les produits en fonction de la **catégorie** la plus fine à laquelle ils appartenaient (les "feuilles" de l'arbre, en rouge sur la figure 2).

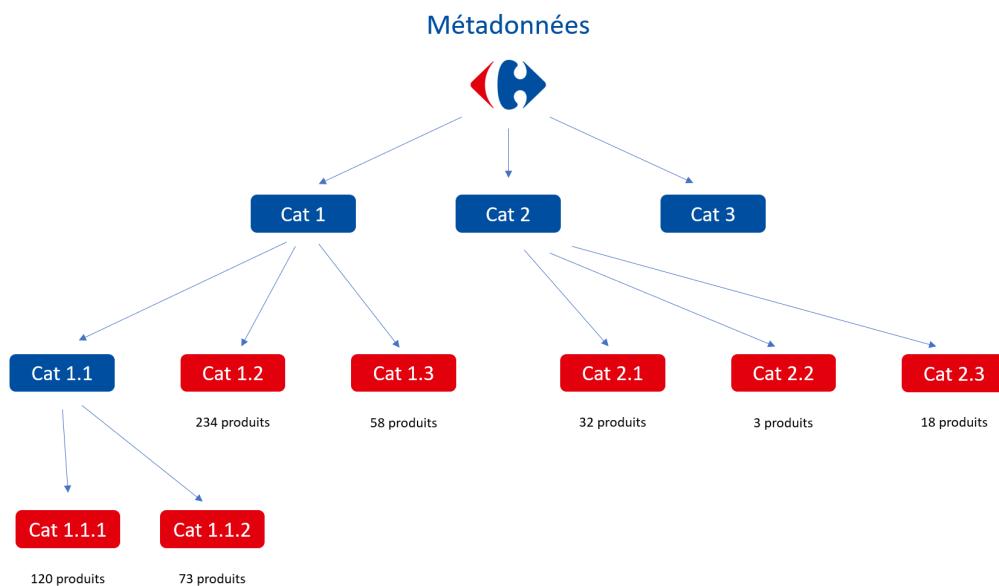


FIGURE 2 – Labels avant pré-processing

Cependant, cette approche avait un défaut certain : parmi ces catégories "feuilles" de l'arbre, les effectifs de produits étaient très inégalement répartis. En effet, si certaines catégories regroupaient plusieurs centaines de références, d'autres n'en contenaient que deux ou trois (comme par exemple la **catégorie 2.2** en figure 2). Ce défaut de répartition était extrêmement problématique, car il n'était pas possible d'entraîner un réseau de neurones à reconnaître ces catégories avec aussi peu d'exemples.

Pour palier à ce problème, nous avons décidé de **fixer un effectif minimal** E_{min} de produits

par catégorie. En ce qui concerne les catégories ne vérifiant pas ce critère, nous avons fait le choix de les fusionner avec les catégories "soeurs" et de les regrouper dans la catégorie parent. Un exemple de réorganisation est représenté en figure 3, les labels retenus pour la classification étant les catégories représentées en rouge. Cette manipulation nous a permis de *lisser* les effectifs par catégories de produits, permettant ainsi de réduire le biais entre les différentes classes.

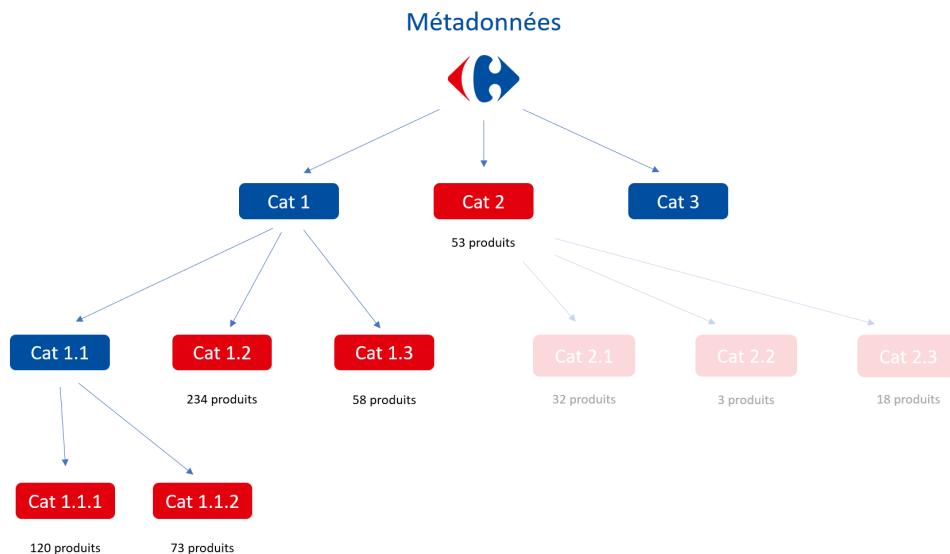


FIGURE 3 – Labels après transformations

1.2.2 • DÉTERMINATION DE E_{min}

Nous avons donc introduit un nouveau paramètre : le nombre de produits minimal retenu pour définir la taille seuil moyenne des labels, E_{min} . Pour ce problème, il était donc judicieux d'identifier **la bonne granularité** à utiliser. Intuitivement, si E_{min} augmente, le nombre de catégories diminue, car on force l'agrégation des catégories faiblement peuplées en super catégories. Il sera dès lors plus simple d'obtenir une bonne précision sur ces catégories, mais ces dernières seront très générales : "glaces", "tout pour des fêtes réussies", "viandes et charcuterie", "légumes"... De plus certaines catégories, par manque d'effectif, seront même agrégées dans la catégorie "racine" de l'arbre.

D'un autre côté, si E_{min} diminue, le nombre de catégories augmente, car trop peu de produits seront imposés par catégories, et beaucoup de catégories feuilles avec un faible effectif seront conservées. L'évolution du nombre de catégories retenues en fonction de E_{min} est représenté à gauche en figure 4. Celle-ci est comme attendue décroissante, en forme d'hyperbole.

Cependant, contrairement à ce que l'on pensait initialement, agréger d'avantage les catégories en augmentant E_{min} n'a pas tendance à diminuer mais plutôt à augmenter l'écart type des effectifs des produits à l'intérieur des catégories. Or un écart type trop élevé est également problématique pour l'entraînement d'un classificateur, car les catégories très peuplées seraient

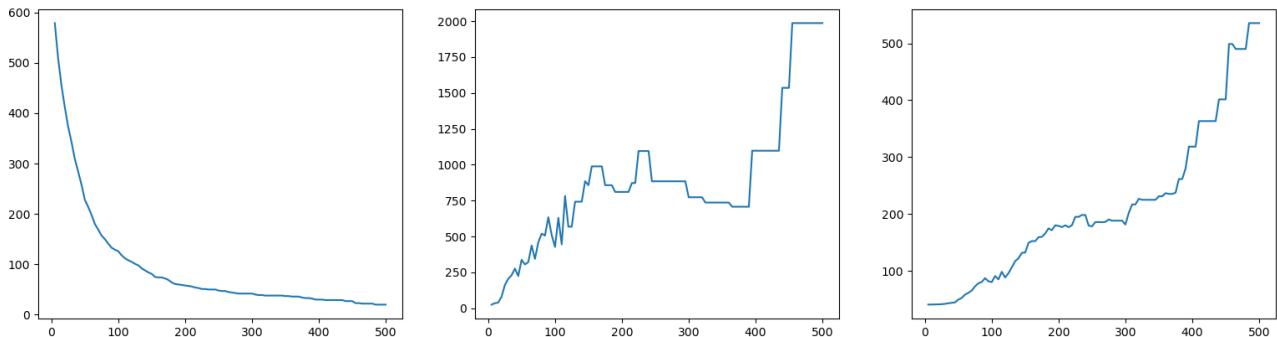


FIGURE 4 – Étude de l'évolution des paramètres en faisant varier E_{min} de 1 à 500. À gauche : le nombre de catégories différentes obtenues. Au milieu : le nombre de produits dans la catégorie "racine". Enfin à droite : l'écart-type des effectifs de produits dans les catégories.

avantageées par rapport aux catégories moins peuplées, ce qui entraînerait un biais dans la classification.

Au terme de cette étude, **nous avons finalement choisi de retenir $E_{min} = 50$** , d'une part parce que cette valeur réalisait un bon compromis effectif minimal / nombre de catégories (228 catégories dans ce cas), et d'autre part car l'écart-type pour cette valeur reste assez faible ($\sigma = 49.6$), comme on peut le voir sur la figure 4 à droite. L'histogramme de répartition des effectifs des produits à l'intérieur des catégories est représenté en figure 5.

Ce pré-traitement des données est implémenté dans les fichiers : `meta_data_preprocessing.py` contenant les classes respectives au nettoyage et à cette tâche, et dans `meta_data_main.py`, fichier exécutant les étapes successives requises.².

La détermination précise de ce paramètre ne se limite cependant pas à l'aspect purement data de ce challenge. Une étude marketing doit être réalisée afin d'englober toutes les contraintes du problème et de Carrefour. Cette étude, dépassant le cadre de data, est laissée aux équipes de l'industriel partenaire.

1.3 PRE-PROCESSING DES IMAGES

Un autre aspect important de ce problème était de parvenir à reconnaître un produit peu importe l'angle de prise de vue, la luminosité de la pièce, ou encore le bruit de l'image. Pour permettre au modèle de reconnaître un produit malgré ces facteurs aléatoires, nous avons mis en place un ensemble de **transformations aléatoires** (rotations, effets de perspective, translations, changements de couleurs...) appliquées à notre set d'images à chaque itération dans nos modèles de réseaux de neurones. Un aperçu de ces transformations aléatoires est représenté en figure 6.

2. Tous les fichiers auxquels nous faisons référence dans ce rapport sont disponibles dans le dossier git en ligne du projet : github.com/bison-fute/Carrefour-Image-Recognition-Challenge

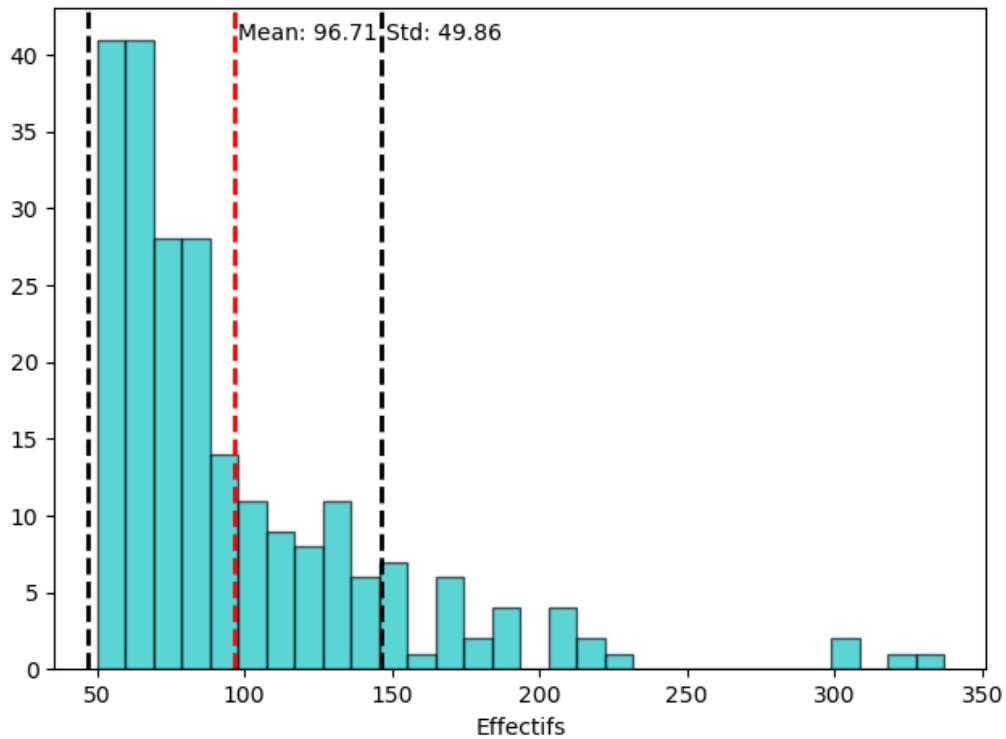


FIGURE 5 – Histogramme de répartition des effectifs des produits à l'intérieur des catégories pour $E_{min} = 50$.

Enfin, pour préparer la tache de classification avec la reprise d'architectures pré-entraînées (voir la section 2.1 *Transfer Learning*), nous avons choisi de normaliser toutes les images, en les transformant au format 224×224 pixels, qui est le format classique pour la reconnaissance et classification d'images.



FIGURE 6 – Exemples de transformations aléatoires sur le set d'images part5

2

MODÈLES RETENUS

2.1 TRANSFER LEARNING

Afin d'aborder ce challenge, nous avons choisi d'adopter une approche par **transfer learning**. Le transfer learning est une méthode qui consiste à reprendre un réseau pré-entraîné afin de l'adapter à notre tâche de classification spécifique, et ainsi s'épargner beaucoup de temps d'entraînement.

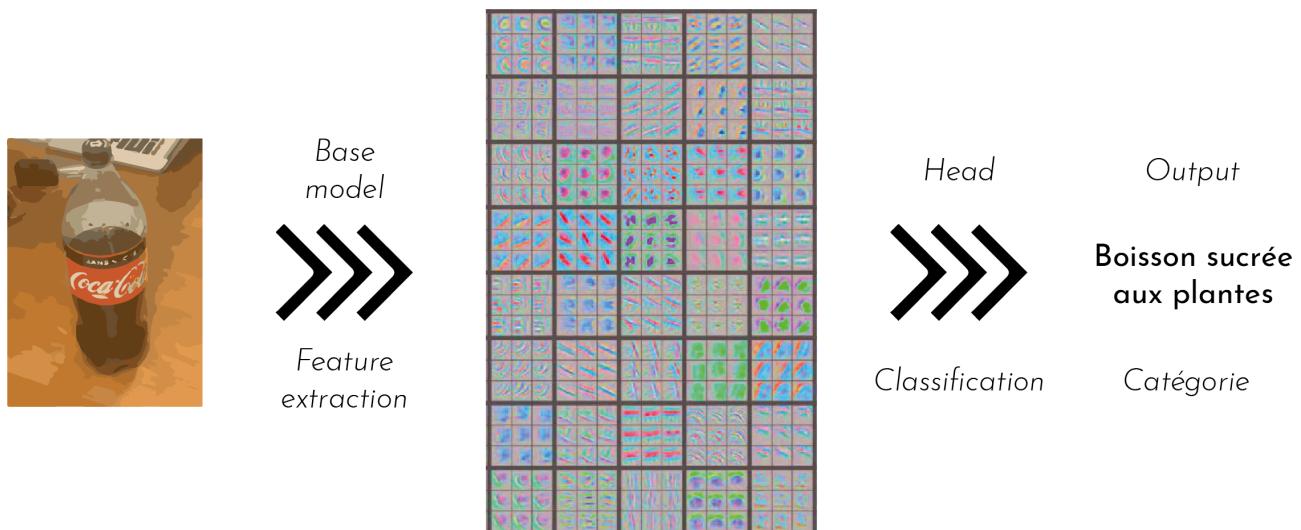


FIGURE 7 – Schéma de fonctionnement du transfert learning

Le principe du transfer learning est de reprendre uniquement la **base** du réseau pré-entraîné et d'ajouter à sa sortie une **head** (tête du réseau) qui aura pour rôle de classifier les images à partir de **features** extraites par la **base**. On présente dans les parties suivantes trois architectures typiques adaptées à la problématique du challenge : VGG-16, DenseNet161 et ResNet50. Nous présenterons en section 3.1 la comparaison des performances de ces trois architectures.

2.2 VGG-16

VGG16 est un modèle de réseau neuronal convolutif proposé par l'équipe Visual Geometry Group de l'Université d'Oxford [1]. Le modèle atteint une précision de 92,7 % dans le test top-5 d'ImageNet, qui est un ensemble de données de plus de 14 millions d'images appartenant à

1000 classes. C'est l'un des célèbres modèles soumis à l'ILSVRC-2014. Il utilise massivement des convolutions avec des filtres de tailles 3x3. VGG16 a été entraîné pendant des semaines et a utilisé des GPU NVIDIA Titan Black.

2.3 DENSENET161

Le DenseNet connecte chaque couche à toutes les autres couches de manière prospective. Alors que les réseaux convolutifs traditionnels à L couches ont L connexions - une entre chaque couche et la couche suivante - le réseau d'intérêt réseau possède $L(L+1)/2$ connexions directes. Pour chaque couche, les cartes de caractéristiques de toutes les couches précédentes sont utilisées comme entrées, et ses propres cartes de caractéristiques sont utilisées comme entrées dans toutes les couches suivantes. Les réseaux denses présentent plusieurs avantages convaincants : ils améliorent la propagation des paramètres par descente de gradient, encouragent la réutilisation des caractéristiques et réduisent considérablement le nombre de paramètres. Les DenseNets apportent des améliorations significatives par rapport à l'état de l'art sur la plupart d'entre elles, tout en nécessitant moins de calculs pour atteindre des performances élevées. Cette architecture a été établie et pré-entraînée sur ImageNet par Huang en 2016 [2].

2.4 RESNET50

Les Resnets sont une sous-catégorie des réseaux de neurones convolutionnels, appelés Residual Networks (réseaux résiduels). Ces réseaux ont la particularité d'être très profonds : par exemple le réseau VGG présenté en 2.2 possède une architecture à 16 couches, alors ResNet50 [3] déploie un architecture de 50 couches. De plus, les réseaux de neurones résiduels introduisent des connexions *résiduelles* entre les couches, ce qui signifie qu'une couche, à la place de prendre en entrée uniquement la sortie de la couche précédente, prend en fait les sorties des n_{res} couches précédentes, où n_{res} est un paramètre fixé du réseau.

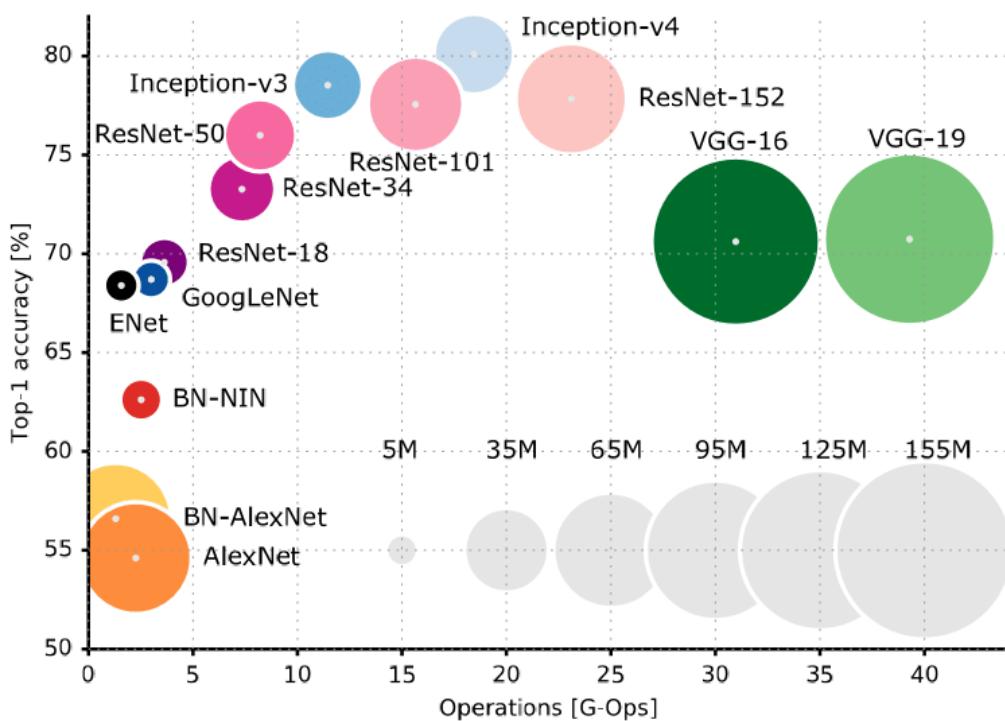


FIGURE 8 – Comparatif des performances de différents réseaux sur le dataset Imagenet

3 RÉSULTATS

3.1 COMPARAISONS DES MODÈLES

Nous abordons ici la présentation de nos résultats. Le dataset complet se compose des trois dossiers parents **part1**, **part2** et **part5**. Les trois dossiers totalisant plus de 110Go de données, nous avons fait le choix de n'utiliser que les données contenues dans **part5**, dossier de **15Go** **mieux adapté à la puissance de calcul** que nos avions à disposition.

La majorité des calculs ont été réalisés sur une machine avec la configuration suivante : DELL Inspiron 15 5000, CPU Intel Core i7-6500U, RAM 16 Go, GPU Intel HD Graphics 520. Pour quatre époques d'entraînement, chacun des modèles prenait en moyenne deux heures pour s'entraîner. Le code source pour l'exécution de ces modèles est disponible dans le fichier `main.py`³.

Modèle	VGG-16	DenseNet161	ResNet50
Précision sur l'ensemble d'entraînement	0.37	0.45	0.39
Précision sur l'ensemble de test	0.36	0.45	0.36

La courbe d'apprentissage du modèle DenseNet161 est représenté en figure 9. On peut notamment voir que la courbe d'erreur continue de chuter ce qui signifie qu'on aurait pu continuer l'entraînement pour améliorer les résultats. D'autre part, on vérifie que l'erreur d'entraînement est quasiment égale à l'erreur de validation ce qui confirme que le modèle n'a pas sur-appris l'ensemble de données (pas d'overfitting).

Le modèle retenu pour cette étude est donc le **réseau pré-entraîné DenseNet161** sur le dataset d'ImageNet, affichant une précision de 0.45 sur notre ensemble de test du dossier de données **part5**. Une information supplémentaire intéressante à connaître est le rang moyen de la véritable catégorie cible parmi les catégories les plus probables données par l'architecture mise en place. La figure 10 rend compte la distribution du rang de la catégorie cible prédit par le modèle.

3.2 TEST DU RÉSEAU SUR DES IMAGES EN DEHORS DU SET

Afin de tester le modèle final, l'initiative a été prise d'utiliser en entrées des photos de produits issus de la grande distribution avec un smartphone. L'objectif de cette étude est de **vérifier**

3. Tous les fichiers auxquels nous faisons référence dans ce rapport sont disponibles dans le dossier git en ligne du projet : github.com/bison-fute/Carrefour-Image-Recognition-Challenge

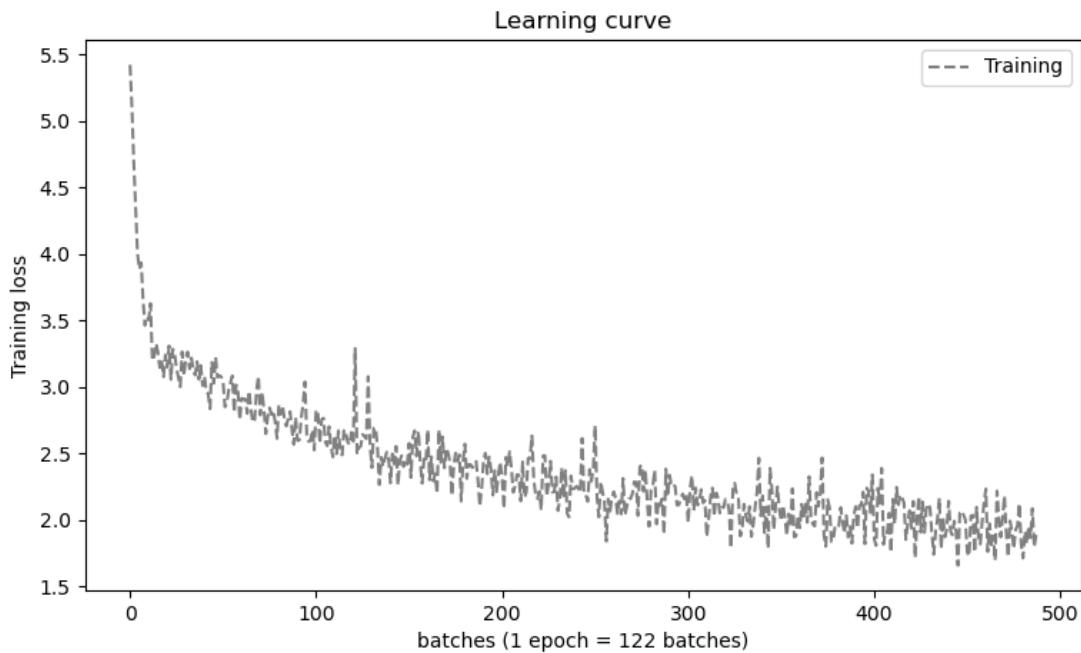


FIGURE 9 – Courbe d'apprentissage de l'architecture Denseet161

si l'architecture obtient les mêmes performances avec des images du monde réel. Elle s'inscrit dans une des applications potentielles évoquées lors de la présentation du projet de reconnaissance d'image : le frigo connecté, capable de lister automatiquement les produits en son sein et les produits manquants à acheter.

Les photos extérieures amenées sont représentées en figure 11. Elles appartiennent à différentes catégories présentées dans le dossier Part5 sur lequel le réseau a été entraîné : "Lentilles et Légumes secs", "riz", "Pâtes, Riz et Féculents".

Les résultats obtenus, essayant de prédire les classes et catégories de ces produits à l'aide de notre réseau entraîné sur les données Carrefour, sont assez mauvais. Cette mauvaise qualité était prévisible : l'ensemble d'entraînement possède des images détournées et sans fond, le réseau est donc incapable d'isoler une image sur un fond non-transparent. Pour se placer dans un contexte plus proche de l'ensemble d'entraînement, les images ont ensuite été détournées. Les résultats sont reproduits en figure 12.

Parmi les images considérées, seul de jus de fruit de la marque Tropicana parvient à être identifié correctement, et dix produits sur douze n'ont aucune prédiction juste parmi leurs trois premiers labels prédits. Ces résultats montrent que malgré les transformations aléatoires appliquées aux images en entrée, le modèle n'a pas réussi à se généraliser à tout type de photos.

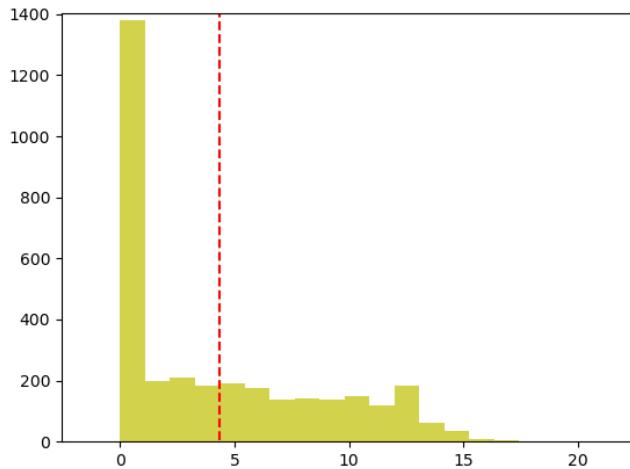


FIGURE 10 – Histogramme du rang de la vraie catégorie parmi les prédictions en sortie du modèle, en rouge la valeur moyenne.

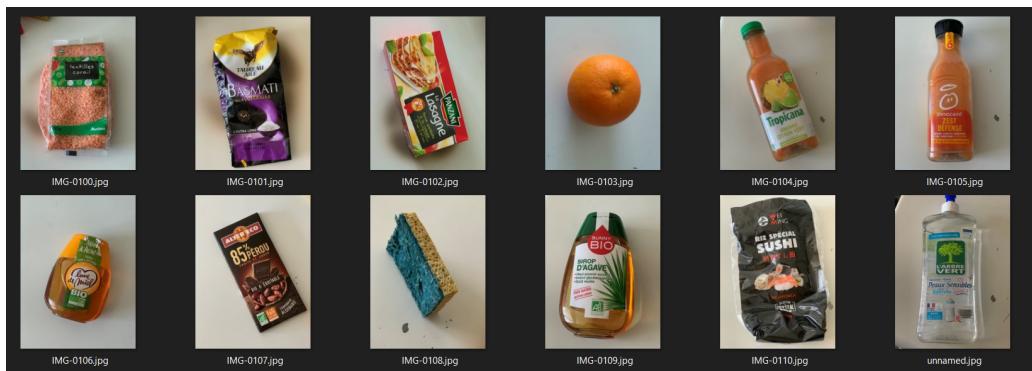


FIGURE 11 – Photos des produits considérés

3.3 PISTES D'AMÉLIORATIONS VERS L'OBJECTIF INITIAL

3.3.1 • ENREGISTREMENT DES SORTIES DE LA BASE

Dans les trois modèles présentés plus haut, nous passons à chaque époque l'ensemble des données à travers tout le réseau de neurones pendant la phase d'entraînement. En transfer learning, nous n'entraînons que la **head** du réseau tandis que les poids de tout le reste du modèle (**la base**) restent figés. **Le passage des données dans la base est extrêmement coûteux** (beaucoup d'opérations effectuées afin d'extraire les caractéristiques des images). Les images en entrée sont toujours identiques, celle-ci effectuera toujours les mêmes calculs et les features extraits seront donc toujours les mêmes.

Une optimisation simple est dès lors possible : pré-calculer les features extraits par le modèle en faisant passer une première fois toutes les images dans la **base**, puis en stockant les features

obtenues. Ainsi, pendant l’entraînement, il ne reste plus qu’à faire passer ces données pré-calculées à travers la `head` uniquement, ce qui **permet d’économiser beaucoup de temps de calcul, et donc d’entraîner le modèle sur plus d’époques**.

Cependant, cette optimisation n’est à priori plus possible dès lors que l’on fait subir des transformations aléatoires (rotations, translations, changements de couleurs...) aux images en entrée, puisque ces transformations changeront justement les outputs de la `base`. Comme vu en section 1.3, ces transformations étaient nécessaires à notre modèle, celui-ci ambitionnant de reconnaître des produits peu importe les caractéristiques de la prise de vue. De plus, ayant finalement effectué relativement peu d’époques (4 pour chaque modèle), cette optimisation aurait été superflue.

Néanmoins, dans la perspective de l’entraînement d’un réseau plus précis, capable de reconnaître un produit exact à partir de sa photo, il serait envisageable d’appliquer à chaque image de la base de données un certain nombre de transformations (par exemple 5 transformations à chaque image), et d’effectuer le pré calcul évoqué précédemment sur chacune des images obtenues. On serait alors capable d’entraîner le modèle sur un grand nombre d’époques, tout en conservant sa capacité à reconnaître des images peu importe les conditions de prise de vue, comme vu en section 3.2.

3.3.2 • FRIGO CONNECTÉ ET DATA AUGMENTATION

Une solution commune pour généraliser un réseau dédié à l’analyse d’image est *l’augmentation de données*. Le principe est proche de celui des transformations aléatoires appliquées aux images dans ce travail. Une batterie de transformations plus élaborées est appliquée à chaque image utilisée en entrée : des déformations non-uniformes de l’image pour reproduire un paquet chiffonné, des gradients de luminosité et contraste, reproduisant les effets d’éclairage présents sur une vraie photo. Toutes les sorties de ces transformations sont conservées pour l’entraînement : une image en entrée devient associée à n nouvelles images copies modifiées. Ce processus **augmente artificiellement la taille du dataset** et permet au réseau d’être plus général et robuste.

Il s’agit également de prendre en compte le cas plus général des images non-détourées, images obtenus dans le cas pratique du réfrigérateur. A cette fin, on peut ajouter des fonds aléatoires aux images utilisées en entrée. Ceci afin que le réseau apprenne lui-même à localiser la partie de l’image associée au produit cible.

Ces développements sont intéressants et prometteurs. Ils sont massivement utilisés pour des applications de Computer Vision. Notons toutefois que dans le cadre de ce projet, les capacités de calculs à disposition (ordinateurs personnels de particuliers) sont trop maigres pour se permettre d’augmenter la taille du dataset sans mener à des calculs trop longs.

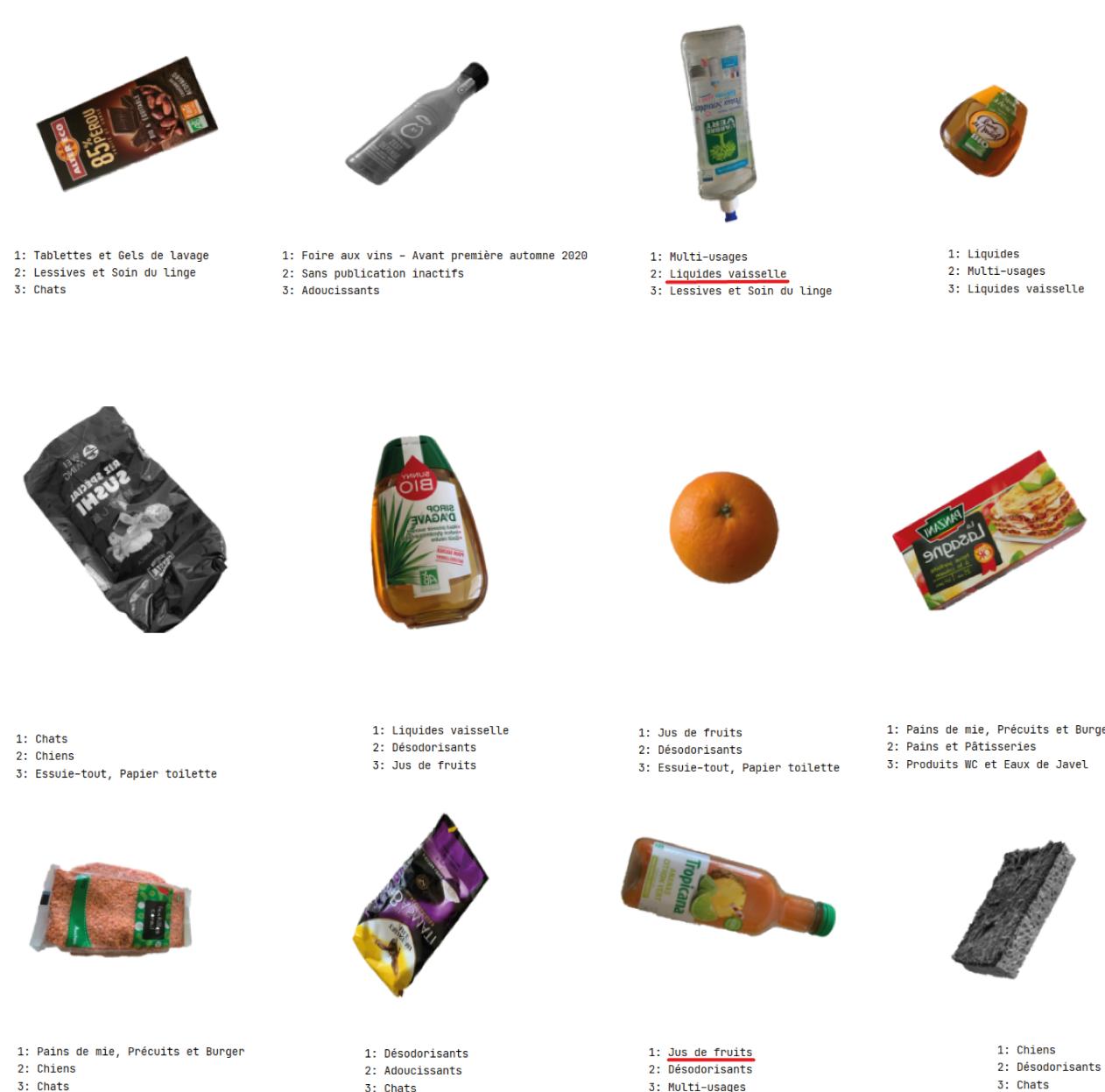


FIGURE 12 – Résultats obtenus sur les images hors du set

4

CONCLUSION

Ce projet de reconnaissance d'image a été mené conjointement avec Carrefour dans le cadre de notre cours *MAP569*. Le but de ce projet était d'arriver à reconnaître à partir d'une photo d'un produit prise par un consommateur le produit du catalogue Carrefour associé. Ce premier objectif a été identifié comme irréalisable dans le cadre d'une tâche d'apprentissage supervisé à partir de l'ensemble de données à disposition. Pour répondre à celui-ci, une méthodologie plus directe peut plus facilement être mise en place grâce au code-barres des produits : l'utilisateur peut simplement *scanner* ce code et identifier le produit grâce à une API de reconnaissance de code-barres.

Afin de tirer profit des données utilisées pour ce challenge, un deuxième objectif a été défini : celui de prédire les produits similaires à partir de l'image, la vision d'un produit. Ainsi, l'application business devient celle de, à partir d'une image d'un produit, *proposer un produit similaire ou un substitut au consommateur*. Dès lors, l'objectif était de définir comment savoir ce qu'est un produit similaire : est-ce un produit de la même couleur, un produit du même industriel, le même type de produit, etc. Pour ce faire, l'utilisation est faite des métadonnées des produits de Carrefour, on choisit **d'agrégner les produits par catégories similaires** : "viandes", "produits transformés à base de viande", etc.

À ce stade, il s'agit alors de mener une étude data de la problématique : nous cherchons à prédire une catégorie d'un produit. La catégorie doit comporter suffisamment de produits pour que l'entraînement et la prédiction par un méthode de machine learning soit rentable, tout en identifiant des catégories **d'une granularité adaptée** et suffisamment fines : des produits très similaires enclins à être utiles au consommateur. Le pré-processing des données a donc représenté une part majoritaire du travail sur ce challenge et un algorithme de pré-processing a été défini à cette fin. La taille minimale moyenne par catégorie E_{min} est laissée en paramètre et peut être ajustée par les équipes à l'interface data et marketing du partenaire de ce challenge.

La quantité de données a été difficile à traiter avec les ordinateurs de particuliers à notre disposition. Le traitement d'un tel amas aurait été plus adapté avec des serveurs et des services de cloud computing qui n'étaient malheureusement pas disponibles. Afin de palier à ce problème, le choix a été fait de **se concentrer sur une partie restreinte de l'ensemble de données**, suffisamment représentatif et varié pour être étendu à plus de données.

Finalement, le projet soulevait initialement deux *use cases* associés à la reconnaissance de produit du partenaire : le frigo-connecté et la détermination de substituts. Pour le premier, les résultats ne sont pas encore convaincants. Ils pourraient être améliorés par une augmentation de données intelligente et une plus grande puissance de calcul. En revanche, pour le second cas d'application, les performances affichées sont très encourageantes : la précision affichée est de 0.45, soit près d'un produit sur deux correctement substitué. Ce chiffre est à mettre en regard

avec le nombre de classes en sortie, 240, pour lequel un classifieur naïf aléatoire présenterait **une précision de 0.004**. On montre également que la catégorie cible est **en moyenne parmi les 5 premières catégories prédictes** par le modèle, ce qui est également encourageant au regard des applications de recommandation citées plus haut.

RÉFÉRENCES

- [1] Karen SIMONYAN et Andrew ZISSEMAN : Very deep convolutional networks for large-scale image recognition, 2015.
- [2] Gao HUANG, Zhuang LIU et Kilian Q. WEINBERGER : Densely connected convolutional networks. CoRR, abs/1608.06993, 2016.
- [3] Kaiming HE, Xiangyu ZHANG, Shaoqing REN et Jian SUN : Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.