CentraleSupélec

Rapport final

Client : M. Luc HAUMONTE
Eaux Cristallines

Reconnaissance de déchets par Transfer Learning et Optimisation d'un CNN

Amine CHERIF HAOUAT
Tom LABIAUSSE
Cyrine NABI
Pierre OLLIVIER
Selim BEN TURKIA

1A 2020-2021

Sommaire

I) Introduction

1) Présentation du client et de l'association Eaux Cristallines

Notre client, Luc HAUMONTE, fonde l'association Eaux Cristallines en 2019 pour lutter contre la pollution marine d'origine humaine et préserver l'environnement. L'association mise sur l'humain, en essayant d'impliquer les citoyens directement, et se tourne vers les outils technologiques pour lutter contre ce fléau, tout en gardant une logique d'économie circulaire et de création de valeur partagée. Parmi les actions entreprises par l'association, le développement d'une application de géolocalisation des déchets, actuellement en phase de bêta-test et soutenue par la Région Sud. Notre projet au sein du pôle Intelligence Artificielle de CentraleSupélec s'inscrit dans la continuité de cette initiative de lutte contre la pollution dans les milieux naturels.

2) Problème posé et bénéfice attendu par le client

Le fond du problème est essentiellement le suivant : les déchets marins se font de plus en plus nombreux, si bien que si rien n'est entrepris d'ici 2050, le nombre de déchets au mètre cube dépassera le nombre de poissons (d'après la fondation Ellen Mac Arthur). L'idée est donc de réussir, à partir d'une photo prise par n'importe quel individu (en respectant évidemment des contraintes de dimensions des photos et une distance adéquate pour pouvoir observer le déchet), à pouvoir **identifier les déchets qui sont présents sur la photo.**

Notre client s'est tourné vers nous car à sa connaissance il n'existe pas d'algorithmes d'analyse d'image accessibles spécifiquement orientés vers la reconnaissance de déchets marins, ou même de déchets en général. Nous avons en effet pu confirmer ce point suite à nos recherches bibliographiques durant la première phase du projet.

Ce que le client attendait de nous, c'est donc que nous puissions identifier la présence ou non de déchets marins sur la photo, mais aussi le type du ou des déchets, selon des catégories définies au préalable et fortement influencées par les catégories déjà présentes dans la base de données. Cette identification se ferait à l'aide d'algorithmes supervisés, qui ont besoin de données labellisées dans chaque catégorie. Cependant, l'absence notable de base de données de déchets marins bien labellisées nous a poussés, avec le client, à reconsidérer le problème : **nous allons donc nous concentrer sur les déchets en général.**

La finalité de ce projet pour le client est de pouvoir intégrer notre solution à l'identification de déchets par l'application que nous avons évoquée précédemment. Elle permettra à ses utilisateurs de prendre en photo des déchets qu'ils peuvent trouver eux-mêmes, et s'ils le souhaitent, d'indiquer la nature du déchet ainsi que des informations complémentaires dessus, afin de pouvoir organiser des sessions de ramassage de déchets intelligentes. Ce côté participatif bénéficiera au client (comme expliqué précédemment), mais aussi à la base de données qui s'en retrouvera plus fournie. Cela pourra donc perfectionner l'algorithme afin qu'à terme, il devienne « autonome », c'est-à-dire qu'il pourra directement identifier les déchets sans informations complémentaires.

II) État de l'art

1) Les bases de données

Cette partie est fondamentale car c'est sur la qualité de la base de données que repose essentiellement la qualité des résultats. C'est l'article **[1]** qui nous a vite orientés sur cette priorité qu'est de trouver une base de données adaptée à notre projet et bien labellisée (l'algorithme étant supervisé). Plusieurs bases de données de déchets ont été trouvées mais la majorité n'est pas utilisable pour plusieurs raisons : inaccessibles, mal labellisées, peu adaptée à des conditions réelles (photos avec un fond blanc uni par exemple)… Les bases de données disponibles et adaptées à notre problème (avec des images de déchets bien labellisées et prises en milieu naturel) sont donc peu nombreuses.

Nous nous sommes finalement orientés vers la base de données TACO **[2]**. TACO est une base donnée open-source des déchets rencontrés dans des environnements naturels ou lieux publics très divers (plage, rue, parc, chemin, forêt, …). Ces images ont été labellisées manuellement dans le but d'entraîner et évaluer les algorithmes de détection d'objets. Le fait que les images soient de bonne qualité et sous licence gratuite fait de la base de données TACO un outil précieux pour l'entraînement d'algorithme d'apprentissage.

Il est cependant important de noter que la quantité d'images est limitée (à peine plus de 1000) ce qui limite la performance de l'algorithme (problème traité plus en profondeur dans la description du travail). De plus, le cadrage assez large de la majorité des photos implique que ces dernières ne contiennent pas uniquement des déchets. Entraîner des algorithmes sur cette base de données leur permettra donc de pouvoir se confronter à différents objets, qu'ils soient des déchets ou non.

2) Pourquoi l'apprentissage profond (Deep Learning) ?

Le Deep Learning est une approche d'apprentissage universelle. Elle peut aussi être retrouvée sous le nom d'Universal Learning puisqu'elle s'applique dans la quasi-totalité des domaines d'apprentissage. L'avantage d'une approche en Deep Learning est l'adaptation au problème que nous rencontrons ainsi que la quantité conséquente de travaux déjà réalisés dans la détection et la classification d'images via cette méthode.

Ce choix du Deep Learning a en particulier été motivé par l'article **[1]** qui a mis en évidence la puissance de cette méthode pour une problématique fortement semblable à la nôtre. Cependant, leur approche a consisté à créer un réseau de neurones « à partir de rien ». Cela semble peu adapté à notre problème car notre réseau devra être entraîné sur des photos en conditions réelles et non pas des photos idéales sur fond blanc où les contours du déchet sont évidents et l'apprentissage facilité. Nous avons donc décidé de nous tourner vers des architectures déjà existantes et pré-entraînées sur des classes similaires à celles que nous devons reconnaître, à savoir des objets sur des photos en couleurs.

3) Le Transfer Learning

Une utilisation populaire des réseaux de neurones est le **Transfer Learning**. Cela consiste à récupérer une architecture existante ainsi qu'une partie des différents poids affectés à chaque couche du réseau pour les « réadapter » au problème que nous souhaitons résoudre. Peu d'articles mentionnent directement le Transfer Learning(il en existe cependant quelques-uns **[3, 4]**), mais les tutoriels OpenClassrooms **[5]** qui nous ont aidés à nous auto-former ont montré le potentiel de ces derniers. Il existe plusieurs manières de faire appel au Transfer Learning (ici adaptées à la classification d'images) :

- **En réentraînant l'intégralité des poids du réseau mais en conservant son architecture initiale.** Cela peut être utile lorsque les *features* (à savoir les traits caractéristiques) des images de la base de données sont différentes de celles extraites par le réseau préentraîné (comme par exemple reconnaître des visages à partir d'un algorithme de détection d'objets). On peut aussi modifier les dernières couches afin de réadapter la sortie au problème, par exemple si la sortie initiale est un vecteur de 1000 éléments alors que nous souhaitons réaliser une classification binaire.
- **En ne réentraînant que les dernières couches du modèle.** C'est une solution peu utilisée car elle suppose de connaître l'effet exact de chaque couche sur les entrées.
- **En ne réentraînant aucune couche et en adaptant la sortie.** Cette solution est très utilisée lorsque les catégories à reconnaître forment un sous-ensemble des catégories déjà reconnaissables par le réseau.

4) Les modèles préexistants

Une fois la voie du **Deep Learning et des CNN (Convultionnal Neural Networks ou réseaux de neurones convolutifs)** choisie, il faut déterminer les algorithmes existants les plus efficaces **[6]**. Cette analyse est fondamentale dans le choix de l'algorithme utilisé pour **effectuer le Transfer Learning**. Nous avons été mis sur cette piste grâce à l'article **[7]** qui détaillait les performances d'algorithmes déjà existants pour la détection de déchets (essentiellement les CNN et d'autres types de *classifiers* moins performants).

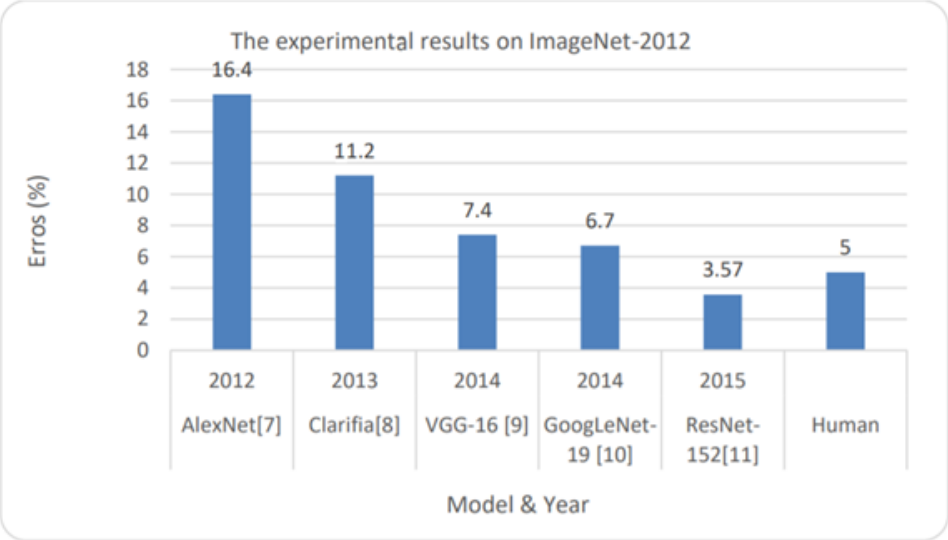


Figure 2.4.1 : Accuracy de différents modèles de Deep Learning entraînés sur ImageNet

Le premier modèle que nous avons analysé est VGG-16 **[8]**, qui est un réseau de neurones convolutif proposé par K. Simonyan et A. Zisserman dans leur article « Very Deep Convolutional Networks for Large-Scale Image Recognition ». Le modèle atteint un taux de précision de 92,7 % sur *ImageNet*[1], une base de données contenant plus de 14 millions d'images de 1000 classes différentes. Il est plus avancé que AlexNet en remplaçant les filtres de grande taille par de multiples filtres de taille 3x3. De plus, VGG16 a été entraîné pendant plusieurs semaines en utilisant un processeur graphique de haute performance (NVIDIA Titan Black GPU).

[1] https://www.image-net.org/

L'entrée de la première couche est fixée à une image RGB de taille 224x224. L'image passe à travers une multitude de couches, où les filtres sont utilisés avec un champ de faible réceptivité (3x3) qui est la taille minimale pour définir la notion de gauche/droite haut/bas). Dans l'une des configurations, il utilise aussi un filtre convolutif de taille 1x1, qui peut être compris comme une transformation linéaire de la chaîne d'entrée. Le pas d'entrée est de 1 pixel et le remplissage spatial des couches est fait de sorte que la résolution soit conservée durant la convolution, ce qui induit un remplissage de 1 pixel pour les couches de taille 3x3. Le pooling spatial est fait à l'aide de cinq couches max-pooling, qui suivent quelques couches convolutives. Le max-pooling est fait sur une fenêtre de taille 2x2 avec un pas de 2**. La sortie du modèle contient donc 1000 valeurs, chacune correspondant à la probabilité que la photo en entrée contienne une catégorie précise.** Les probabilités somment à 1.

Le modèle contient en tout 16 couches de convolution et de couches *Dense* (toutes les entrées étant connectées à toutes les sorties), d'où le nom VGG16. À noter que nous aurions aussi pu nous intéresser à VGG19 qui contient donc trois couches de plus, mais il est semblable à son prédécesseur.

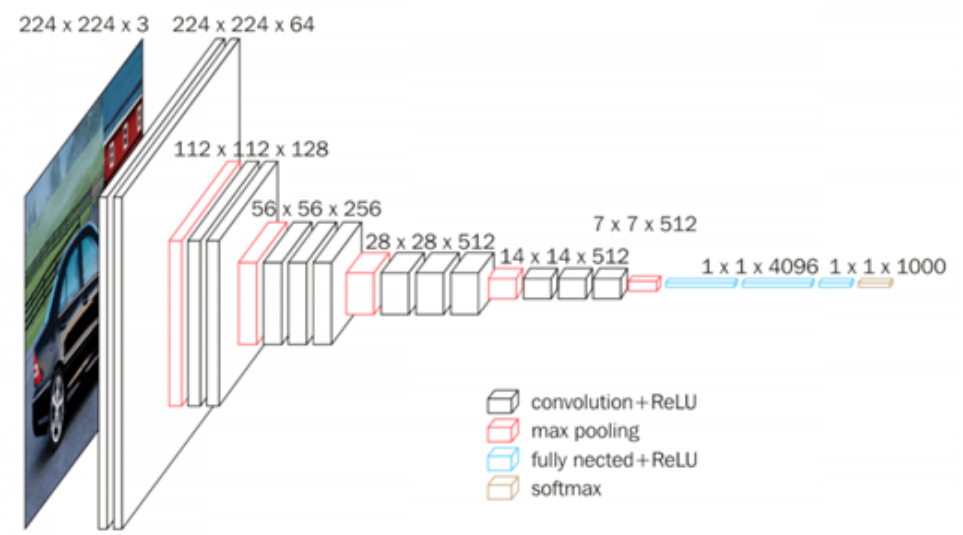


Figure 2.4.2 : Architecture de VGG16

Un second modèle est Residual Network (ResNet) **[9]**, qui a permis d'améliorer les performances de plusieurs applications de visionnage autre que la classification d'image, telles que la détection d'objets et la reconnaissance faciale, en exploitant la puissante capacité de représentation de ResNet.

Depuis que ResNet a bouleversé le domaine du Deep Learning en 2015, plusieurs chercheurs ont approfondi les raisons de son succès, ce qui a abouti à plusieurs raffinements dans son architecture.

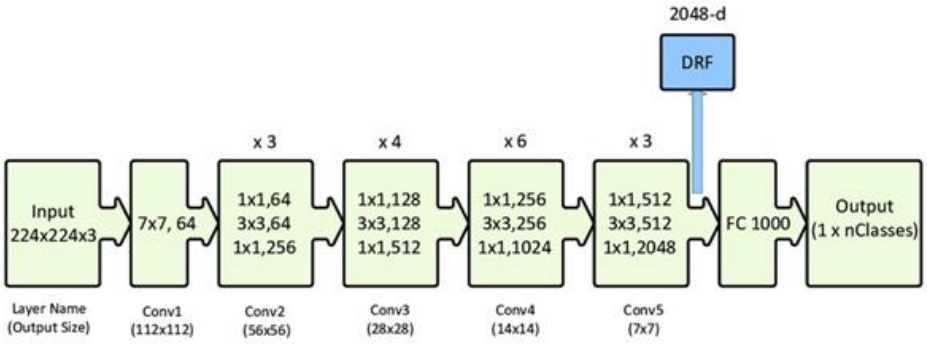


Figure 2.4.3 : Architecture de ResNet-50

Enfin, un troisième modèle est celui d'EfficientNet, qui se décline en plusieurs variantes. En effet, il en existe 7 versions, dont la différence est essentiellement la taille de l'entrée, mais que l'on ne détaillera pas. Ce qui nous importe ici est en effet de repérer quels sont les modèles de références à connaître afin de maximiser nos chances de succès en utilisant une ou plusieurs de leurs architectures dans nos essais de CNN adaptés à notre problème.

5) Le *scaling*

Le *scaling* des CNN **[10]** est utilisé afin d'atteindre une meilleure précision. Par exemple, ResNet peut être « mis en échelle » de ResNet-18 à ResNet-200 en utilisant plus de couches. Cela dit, la procédure de *scaling* des CNN est bien étudiée et inclut aujourd'hui plusieurs méthodes. La plus fréquente est de réaliser le *scaling* les CNN en termes de profondeur ou largeur. Une autre méthode moins utilisée est de faire du *scaling* sur les modèles de résolution d'image. Dans les anciens travaux, on avait tendance à augmenter trois dimensions ; la profondeur, la largeur et la taille de l'image. Il est certes possible de faire varier deux ou trois dimensions arbitrairement, ce qui nécessite une mise en échelle manuelle bien organisée qui peut quand même aboutir à des résultats insuffisants de précision et d'efficacité.

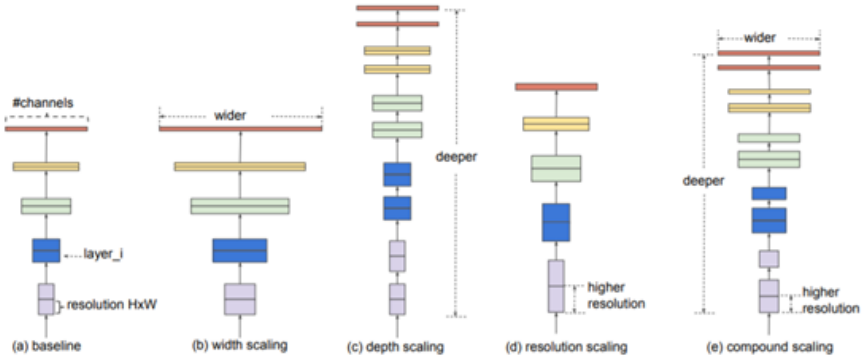


Figure 2.5.1 : Les différents types de scaling

Ce point est fondamental dans la performance de nos algorithmes car il dicte le temps d'apprentissage, d'autant plus que la puissance de calcul dont nous disposons est relativement faible.

6) Le « multi-label classification »

Le **« multi-label classification »** est une méthode qui consiste à déterminer**, pour N classes données, à quelle ou quelles classes appartient notre entrée**. Le document **[11]** détermine plusieurs manières de réaliser ce genre de classification :

- Via une **« chaîne de classification » [12]**, dans laquelle plusieurs classificateurs binaires sont mis en série et où la sortie du classificateur n est l'entrée du classificateur n+1. Cet algorithme a pour but de chercher les corrélations entre les entrées et les sorties, d'où l'importance de l'ordre des classificateurs. Seulement, nos photos contiennent soit 1 déchet à chaque fois, soit des combinaisons de déchets relativement inédites, donc nous avons jugé que chercher des corrélations entre les classes, surtout avec une base de données si peu conséquente, serait peu judicieux. **Cela nous a quand même inspirés à réaliser une succession de classificateurs binaires décorrélés qui aura chacun pour but de détecter la présence ou non d'une classe spécifique.**

- Via une **« multi-class multi-output classification » [11]**, qui consiste à créer des « propriétés » contenant chacune au moins deux classes. C'est comme si nous allions créer 2^n classes où n est le nombre de catégories de déchets, chacune étant de la forme (présence de bouteille ET absence de canette ET présence de cigarette ET présence de polystyrène, …). Cette solution est cependant peu adaptée à notre problème, pour cause la taille de notre base de données, mais surtout le nombre très conséquent de classes disponibles ($2^{\text{nombre de catégories}}$), ce qui nous donnerait autant de classes qu'une architecture comme VGG16 (qui a 1000 classes) pour 5000 fois moins de photos d'entraînement disponibles.

III) Description du travail réalisé

	Décembre	Janvier	Février	Mars	Avril	Mai
Documentation						
Recherche de bases de données						
Structuration de la base de données						
Formation aux réseaux de neurones						
Première tentative d'implémentation de CNN						
Prise en main de VGG16 et Transfer Learning						
Sélection des modèles et résultats obtenus						
Tests d'autres architectures						
Interprétation des prédictions avec LIME						

1) La documentation (décembre-janvier)

Sans trop rentrer dans les détails (traités dans l'état de l'art), la documentation a été un aspect majeur de ce projet. En effet, il fallait nécessairement s'inspirer de solutions déjà existantes à des problèmes qui étaient relativement proches du nôtre.

La tâche a donc été réalisée par l'ensemble du groupe : chacun a, de son côté, lu une certaine quantité de rapports, publications scientifiques, … avant de mettre en commun ces informations lors de nos premières réunions projet. La phase de documentation « intensive » a **duré 1 mois environ**, mais elle a été complétée par plusieurs documentations brèves et ponctuelles lorsque nous faisions face à de nouveaux concepts que nous ne maîtrisions pas.

Au terme de cette phase de documentation, nous avons décidé de nous tourner vers des CNN (Réseaux de Neurones Convolutifs) afin de répondre à la demande du client qui est d'identifier et de catégoriser des déchets.

2) Recherche de bases de données (février)

Une des tâches essentielles en IA est la recherche et la structuration d'une base de données. C'est pourquoi nous avons pris 2 mois pour **rechercher des bases de données (un peu plus d'1 mois)** et de les **mettre en forme (un peu moins d'1 mois)**.

La première étape, qui est la recherche de base de données, aura été réalisée par l'ensemble du groupe pour maximiser les chances d'en obtenir une, et s'est soldée par l'obtention de la base de données *TACO Dataset* **[1]** qui contenait exactement **1500 images de déchets** ainsi que leur labellisation et même leur détourage. Nous avons tenté d'obtenir d'autres bases de données, mais sans franc succès (pas de réponses des organismes qui disaient pouvoir en fournir une si elle était utilisée à des fins de recherche, base de données contenant des photos non labellisées, …). En avril,

nous avons réussi à mettre la main sur la base de données TIDY **[11]** contenant **200 images environ et qui est venue compléter la nôtre**. L'ensemble du groupe a aussi récolté des photos de masques car ils n'existent pas dans les bases de données que nous avons trouvées et que ce sont des déchets d'actualité. Cela a demandé à chacun un fort investissement personnel et nous avons pu collecter au total **182 photos de masques**. Cependant, il n'était pas réalisable d'enrichir notre base de données dans plusieurs catégories de cette manière étant donné les moyens humains disponibles et le temps à notre disposition. De plus, bien que la base de données soit indispensable à notre projet, la constituer n'est pas réellement l'objectif d'un projet mené au sein du pôle IA. Ainsi, comme nous l'a conseillé Wassila Ouerdane, nous avons lancé un appel au sein du pôle IA pour inciter nos camarades à nous aider à enrichir notre base de données en nous envoyant des photos de déchets. Malheureusement, nous n'avons pas reçu une seule photo suite à cette demande, ce qui témoigne bien de la rareté des bases de données spécifiques de nos jours.

Revenons sur ce point en particulier, car il est central dans le déroulement du projet. En effet, comme annoncé ci-dessus, **une base de données bien labellisée et bien fournie est essentielle en IA**, d'autant plus lorsque nous utilisions des CNN, et nous n'avons malheureusement pas bénéficié d'une base de données assez conséquente pour réaliser un CNN d'une efficacité optimale. **Si ce projet venait à être réutilisé et/ou complété, il est impératif d'obtenir une base de données plus conséquente nécessaire à l'entraînement du CNN.**

Le **livrable** est donc un **dossier contenant 15 batchs de photos** ainsi qu'**une base de données SQL** qui recense le ou les déchets sur chaque photo, leur « supercatégorie », leur catégorie ainsi que les coordonnées de leur contour (inutilisées dans notre approche). Au final **la base de données contient environ 1700 images pour plus de 4500 déchets labellisés.**

3) Structuration de la base de données (mars)

Une fois la base de données trouvée, il a fallu la mettre en forme. C'est pourquoi nous avons écrit une série de codes Python qui permettent de **télécharger la base de données et répartir les photos** dans les différents batchs (pour qu'ils soient équilibrés) ainsi que de **détourer les déchets**. Nous avons aussi conçu une interface Tkinter permettant d'**ajouter de nouvelles photos à la base de données** sans modifier manuellement cette dernière. Cette tâche aura donc duré **un peu moins d'un mois**.

Figure 3.3.1 : Aperçu de l'interface Tkinter permettant d'ajouter des photos à la base de données

Les moyens techniques mis en œuvre ici sont essentiellement les **connaissances du langage Python** et des modules **Tkinter** et **CSV**. **Le livrable est donc un ensemble de fichiers Python** contenant l'ensemble du code rédigé.

4) Formation aux réseaux de neurones (avril)

Cette phase est essentielle car nous avons commencé le projet avec très peu voire aucune connaissance en IA. **Durant le début du mois d'avril**, l'ensemble du groupe s'est formé en réseaux de neurones grâce aux formations dispensées par l'association *Automatants* sur l'IA et les réseaux de neurones.

Le cœur de la formation a été **l'initiation au module Tensorflow** de Python, spécialisé dans les réseaux de neurones. Il contient les fonctions nécessaires à la création d'un réseau de neurones « from scratch » ainsi qu'un accès facile à une multitude de réseaux pré-entraînés (VGG, EfficientNet, …)

Il n'y a pas de livrable à proprement parler à l'issue de cette phase, si ce n'est l'acquisition des connaissances et moyens techniques nécessaires à la poursuite du projet.

5) Première tentative d'implémentation de CNN (avril)

Une fois les formations terminées, les premières tentatives de CNN ont eu lieu. Le groupe s'est mis d'accord sur la **création de plusieurs CNN de classification binaire**, chacun ayant pour but de détecter si oui ou non un déchet d'une classe donnée est présent sur une image donnée. En effet, comme annoncé précédemment, la **base de données est beaucoup trop limitée en contenu pour envisager un « multi-label classifier »**, comme expliqué dans l'état de l'art. **La solution de plusieurs CNN mis en série a donc été retenue**.

Il a d'abord fallu adapter la lecture des données à Google Colab. D'abord, il était très peu viable de devoir téléverser l'intégralité de nos photos ainsi que la base de données à chaque nouvelle session, ce qui nous a poussés à créer un Google Drive commun (dont les identifiants sont disponibles) duquel nous pouvions directement importer les images.

Il a ensuite fallu mettre les images au format attendu à l'entrée du réseau, à savoir un array contenant des arrays de taille 800x800x3 (images de 800x800 pixels en format RGB), et un array d'entiers valant 0 ou 1 indiquant si l'image correspondante contient le déchet souhaité (1) ou non (0).

Les premières tentatives ont donc été réalisées à partir d'un réseau créé « from scratch ». Celui-ci était inspiré des réseaux de neurones en reconnaissance d'image proposés par Automatants lors des formations. Cependant, nous ne rentrerons pas dans les détails des résultats car ils ont été peu fructueux.

Il n'y a donc pas de livrable associé à cette tâche car il n'était pas pertinent de sauvegarder le modèle obtenu. Cependant, grâce à cette étape, nous avons pu bien nous faire la main sur le code de création et de modification de réseaux de neurones, et comprendre le rôle de chaque ligne de code plutôt que de faire des copier/coller de lignes de codes déjà existantes.

6) Prise en main de VGG16 et Transfer Learning (avril-mai)

Une fois la piste des CNN « from scratch » essayée, nous avons décidé de tenter d'utiliser les réseaux de plus grosse envergure déjà existants afin de réaliser du **Transfer Learning**. Le premier réseau sur lequel nous nous sommes penchés est VGG-16, car il est déjà particulièrement orienté vers la détection d'objets, et même de certaines classes de déchets que nous cherchions à détecter. Il y a d'abord eu toute une phase de remaniement des données, pour deux raisons :

1) VGG-16 prend en entrée des images de dimension 224x224x3. C'est pourquoi, au lieu de les redimensionner à chaque chargement depuis le Drive, nous avons directement téléversé les images dans ce format-ci dans le Drive. Point important : **toujours privilégier les photos en format carré** pour les réseaux de neurones de reconnaissance d'images. En effet, s'il faut réaliser du Transfer Learning sur des réseaux déjà préentraînés, l'écrasante majorité d'entre eux (VGG16, VGG19, EfficentNet, Resnet, …) prennent des entrées carrées. Si cependant ce n'est que l'architecture du réseau qui intéresse et non pas les poids d'entraînement du réseau, le format des photos importe moins car le format de l'entrée est alors modifiable.

2) Pour une classe donnée, le rapport « nombre images contenant cette catégorie »/« nombre total d'images » était trop faible (de l'ordre de 20 % seulement pour les catégories les plus fournies) : par exemple, **pour 1500 images, à peine 300 contenaient des bouteilles.** Nous garderons l'exemple des bouteilles pour la suite de ce rapport. Enlever des images ne contenant pas la catégorie bouteille était inconcevable vu la taille déjà réduite de notre base de données. C'est pourquoi **nous avons doublé le nombre de photos contenant des bouteilles en ajoutant le symétrique de chaque photo contenant une bouteille.** Cela permettait de passer à un **ratio de photos contenant des bouteilles de l'ordre de 40 %**, ce qui est significativement plus équilibré.

Nous avons donc tenté plusieurs combinaisons possibles en ce qui concerne les couches à réentraîner ou non. Cependant, après discussion avec Léo Milecki, nous avons conclu qu'il fallait soit totalement réentraîner le réseau, soit ne réentraîner que la couche finale de classification.

Notre discussion avec lui nous a aussi permis de mettre le doigt sur un problème dont nous étions peu conscients. En effet, lorsque nous voulions réentraîner VGG16, nous avons instinctivement transformé les images en arrays tridimensionnels contenant des valeurs entre 0 et 255. Cependant, le preprocessing des images à l'entrée de VGG est différent. **En effet, VGG16, contrairement à la coutume de traiter des valeurs entre 0 et 255, centre l'entrée avant de la traiter.** Cette subtilité a des conséquences limitées lorsque nous réentraînons tout le réseau VGG16, mais est fondamentale si nous désirons réentraîner seulement une partie du réseau. C'est à ce jour la méthode que nous continuons d'utiliser, et voici quelques résultats :

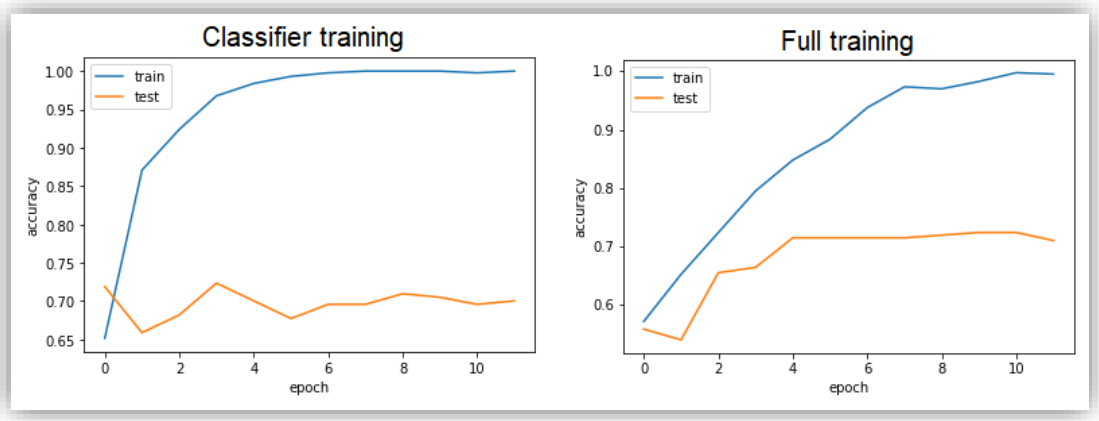


Figure 3.6.1 : Évolution de l'accuracy en Transfer Learning sur la catégorie « Bottle » avec VGG16

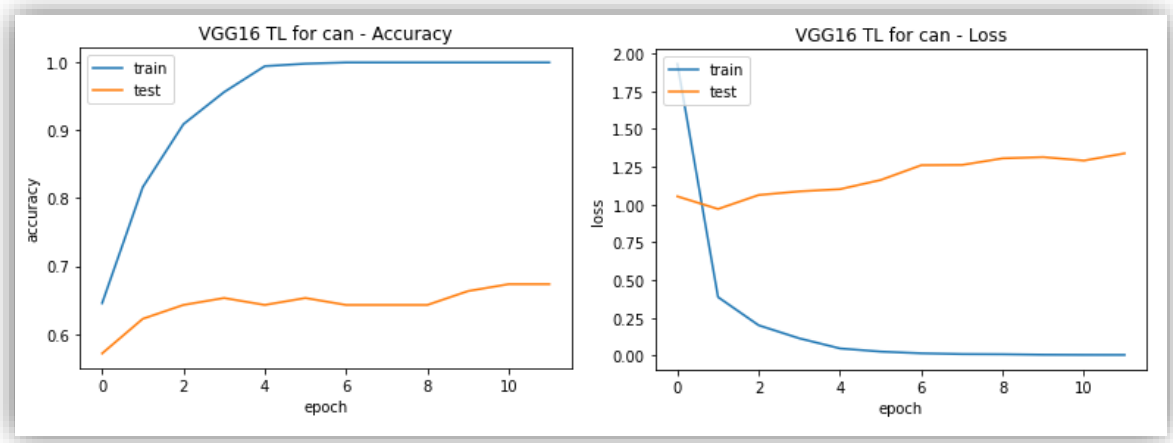


Figure 3.6.2 : Évolution de l'accuracy et de la loss en Full Learning sur la catégorie Can avec VGG-16

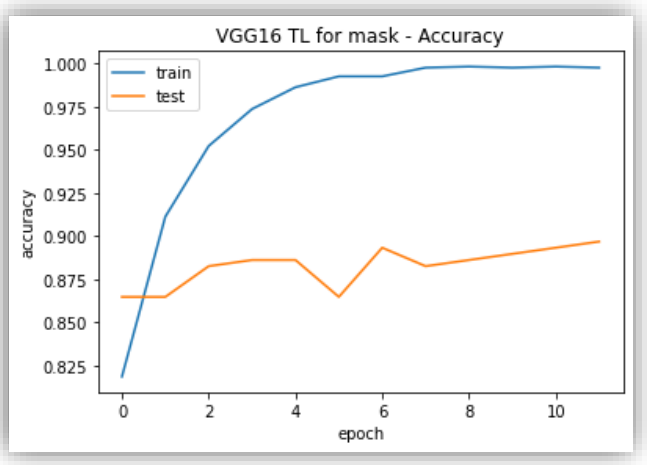


Figure 3.6.3 : Évolution de l'accuracy en Full Learning sur la catégorie Mask avec VGG-16

Chaque catégorie de déchets nécessitant un modèle, nous avons décidé de nous répartir les catégories afin que chacune soit traitée par un membre de l'équipe.

Le premier point à relever dans ces résultats est la **légère différence d'*accuracy* selon que nous entraînons seulement la dernière couche de VGG** qui propose en sortie la probabilité d'appartenance à la classe souhaitée, **ou que nous réentraînons tout le réseau**, comme montré en **Figure 3.6.1**. En effet, en ne réentraînant que la dernière couche, le souci est le nombre très limité de paramètres entraînables, faisant rapidement entrer le réseau en phase de surapprentissage, tandis que la précision sur les données d'entraînement en *« Full Learning »* (à savoir en réentraînant tout le réseau) augmente beaucoup moins vite au vu des 14 millions de paramètres à entraîner. Sur les conseils de Jean-Philippe Poli et Wassila Ouerdane, nous avons, pour la suite du projet, décidé de privilégier l'entraînement complet du réseau.

Le second point est toujours autour du **surapprentissage**. En effet, dans la majorité de nos essais, nous avions affaire au surapprentissage dès les premières *epochs* d'entraînement. Ce phénomène est rapidement reconnaissable, car il est fortement lié à une croissance de la fonction de *loss* (ou fonction de perte) des données de test alors que la *loss* des données d'entraînement est quasiment nulle comme on peut le voir sur la **Figure 3.6.2**. Ce souci est essentiellement dû à deux choses :

1) Pour les canettes par exemple, la proportion de photos contenant cette catégorie est très faible, même après « dédoublement » des photos contenant des canettes. Nous avons donc résolu ce problème en rajoutant une option permettant d'avoir autant de photos contenant la catégorie souhaitée que de photos n'en contenant pas, et ce en retirant des données d'apprentissage et de test des photos ne contenant pas de canettes, jusqu'à obtenir le « 50/50 ».

2) La résolution du problème ci-dessus mène à un second souci, qui est que la base de données ne contient pas assez de photos pour correctement entraîner les réseaux de neurones, problème d'autant plus exacerbé lorsque nous retirions des photos d'entraînement pour atteindre le « 50/50 ».

Le dernier point, et sans doute le plus important, est **la différence énorme de performances entre les différentes classes**. En effet :

1) La catégorie « bottle » par exemple a une précision sur les données test d'environ 70 % (**Figure 3.6.1**), ce qui est correct, mais qui n'est pas suffisant, d'autant plus qu'un pourcentage de précision aussi haut s'explique simplement par la faible proportion de faux positifs : si le modèle fait peu d'erreurs sur les faux positifs, la proportion initiale faible de photos contenant une bouteille suffit à obtenir un pourcentage de précision relativement élevé sans pour autant obtenir un algorithme performant.

2) À l'inverse, la catégorie « mask » atteint une précision proche des 90 % (**Figure 3.6.3**) alors que la proportion de photos contenant un masque est sensiblement la même que celles contenant une bouteille.

Cette différence très importante de précision reflète un paramètre fondamental dans la performance d'un réseau de neurones : la **qualité des photos**.

Figure 3.6.4 : Photo de masque prise par l'équipe du projet

Figure 3.6.5 : Photo de bouteille issue de la base de données

Comme nous pouvons le remarquer ci-dessus, la différence de qualité entre les deux photos est importante : celle de la **Figure 3.6.4** est prise d'assez proche et le masque reste facilement distinguable même après passage au format 224x224. De l'autre côté, sur la **Figure 3.6.5**, la bouteille est décentrée, et à peine reconnaissable à l'œil nu après passage au format 224x224 (et pourtant cette photo-ci est loin d'être la pire du lot). Une solution serait de rogner la photo de bouteille avant de la passer au format souhaité afin que la bouteille soit plus distinguable, mais cela fait soit appel à un algorithme de segmentation d'images en amont pour localiser le déchet mais qui demanderait d'autres compétences techniques considérables, soit à un traitement manuel photo par photo qui serait long et fastidieux.

La conclusion à tirer est donc claire, la base de données dans son état actuel est insuffisante pour réaliser des algorithmes très performants. Pour l'améliorer, il faudrait :

1) D'abord qu'elle soit **plus fournie**, car reconnaître une dizaine ou une quinzaine de catégories (comme souhaité par le client) est une tâche difficile lorsque la totalité de notre base de données ne contient pas plus de 2000 photos, **surtout quand nous comparons notre base de**

données à celle sur laquelle a été entraîné VGG16 : 14 millions de photos pour 1000 classes, soit en moyenne 14000 photos par classe (contre au maximum 436 photos pour la catégorie « plastic bag » dans notre base de données).

2) Que les photos soient **de meilleure qualité**, à savoir prises à une **distance raisonnable du déchet**, que celui-ci soit **théoriquement reconnaissable et classifiable par un être humain**, et que la photo ne contienne **pas trop de déchets qui se superposent les uns aux autres**.

Le **livrable est donc le Jupyter Notebook** « **Projet_TDD_CNN.ipynb** » contenant toutes les importations de modules, la procédure pour se connecter à un Google Drive qui doit contenir la base de données, la mise en forme des données ainsi que les différents réseaux que nous avons mis en place, prêts à être entraînés. Les réseaux entraînés qui nous ont donné les meilleurs résultats de chaque catégorie ont été sauvegardés au format .h5, et sont directement réutilisables via le Jupyter Notebook « **TDD_Classifier.ipynb** ».

7) Sélection des modèles et résultats obtenus

Afin de sélectionner les meilleurs modèles possibles pour chaque catégorie de déchets, nous avons dû tester plusieurs paramétrages possibles de VGG-16.

Tout d'abord, comme nous l'avait expliqué Léo Milecki, nous avons eu le choix entre un entraînement complet du réseau de neurones *(Full Learning)* ou un entraînement des couches hautes de classification uniquement *(Transfer Learning)*. Dans cette dernière configuration, la partie du réseau non entraînée sur nos données conserve le paramétrage issu de l'entrainement de VGG-16 sur *ImageNet*.

Nous avons ensuite testé nos modèles avec plusieurs *learning rate (lr)*. Cette valeur numérique contrôlant la « vitesse d'apprentissage » du réseau, nous avons testé principalement deux valeurs : une haute $lr = 25 \times 10^{-6}$ et une basse $lr = 3 \times 10^{-6}$.

Pour limiter les phénomènes de surapprentissage et de classificateurs triviaux (qui renvoient toujours 0 ou 1 indépendamment de l'image), tous les modèles ont été entraînés en équilibrant les données d'entraînement et de test pour que 50 % de chacune d'entre elles contiennent un déchet correspondant à la catégorie, et 50 % n'en contiennent pas. Enfin, pour chaque catégorie, les images sélectionnées ont été partitionnées en deux ensembles : 80 % d'entre elles servent à entraîner le modèle, et 20 % à l'évaluer. L'entraînement des CNN a été réalisé sur 12 époques puisque nous avons constaté que les performances sur les données de test atteignaient déjà leur maximum pour ce nombre d'époques quelles que soient les catégories. Nous risquons donc de favoriser le surapprentissage au-delà d'une douzaine d'époques étant donné notre base de données.

Le tableau de la **Figure 3.7.1** détaille les résultats obtenus avec les différents paramétrages par le biais de l'*Accuracy.* Cette valeur estimant les performances d'un réseau est calculée sur les données de test grâce à la formule donnée ci-dessous, avec VP les vrais positifs, VN les vrais négatifs, FP les faux positifs et FN les faux négatifs.

$$Accuracy = \frac{VP + VN}{N_{tot}} = 1 - \frac{FP + FN}{N_{tot}}$$

Accuracy	lr = 0,000003		lr = 0,000025			
Categories	Transfer	Full	Transfer		Full	
BOTTLE	0,63	0,68	**0,71**	0,69	0,70	0,66
CAN	0,65	0,68	**0,72**	0,71	0,60	0,69
CUP	0,66	**0,71**	0,65	0,70	0,70	0,70
CARTON	0,70	0,70	0,68	0,64	**0,73**	0,71
CIGARETTE	0,80	0,74	0,78	0,77	0,74	**0,81**
MASK	0,79	0,82	0,78	0,81	0,85	**0,86**
PAPER	0,60	0,59	0,63	**0,68**	0,54	0,63
PLASTIC BAG	0,61	0,60	**0,64**	0,61	0,61	0,60
PLASTIC CONTAINER AND OTHER	0,69	**0,70**	0,64	0,68	**0,70**	**0,70**
STYROFOAM	0,76	0,74	0,80	0,80	**0,82**	0,80

Figure 3.7.1 : Comparaison des Accuracy obtenues avec différents paramétrages de VGG-16

Les résultats de la **Figure 3.7.1** mettent aussi en évidence le caractère aléatoire de l'entraînement. En effet, pour la valeur $lr = 25 \times 10^{-6}$, nous avons réalisé deux fois l'entraînement dans les mêmes conditions. Les résultats obtenus diffèrent (par exemple, pour la catégorie *can*, nous avons obtenu 60 % puis 69 %), ce qui nous a poussés à conserver le meilleur des deux modèles obtenus.

De manière générale, entraîner un réseau de neurones complet sans passer par du *Transfer Learning* se révèle légèrement plus efficace que passer par du *Transfer Learning*. Cependant, les résultats varient beaucoup en fonction de la catégorie concernée.

Nous avons pu tester 10 catégories différentes. Cela peut sembler peu comparé à la diversité des déchets qu'il est possible de trouver dans la nature, mais nous avons été très fortement limités par le nombre d'images présentes dans chaque catégorie de notre base de données. La **Figure 3.7.2** donne le nombre d'images dont nous disposions pour chaque catégorie.

Categories	#Photos
BOTTLE	271
CAN	162
CUP	101
CARTON	145
CIGARETTE	259
MASK	182
PAPER	220
PLASTIC BAG	436
PLASTIC CONTAINER AND OTHER	309
STYROFOAM	64

Figure 3.7.2 : Nombre de photos disponibles par catégories de déchet

Ainsi, en ne conservant que ces 10 catégories, nous avons pu isoler les modèles les plus performants donnés par les cases colorées en vert sur la **Figure 3.7.1**. Ces meilleures performances sont regroupées dans la **Figure 3.7.3**.

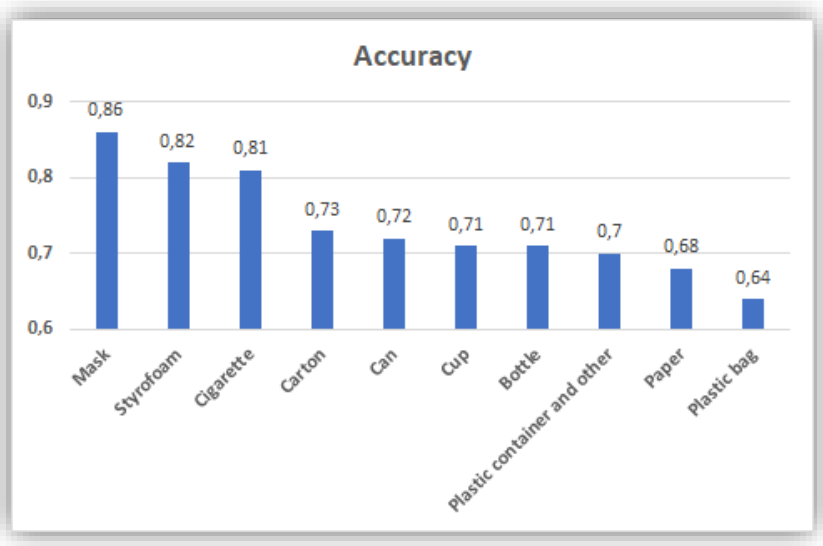
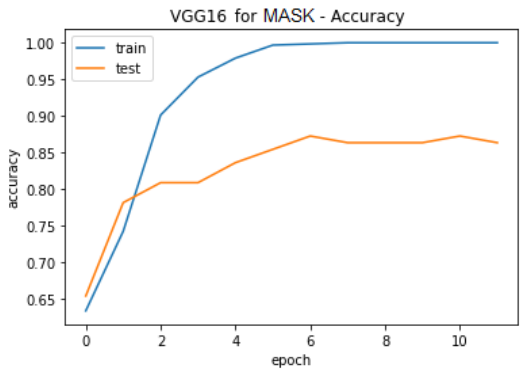
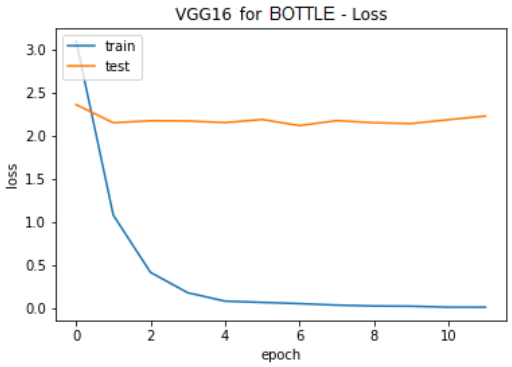


Figure 3.7.3 : Performances des meilleurs modèles obtenus pour chaque catégorie

Il est important de noter que les performances présentées en **Figure 3.7.4** sont biaisés par la faible taille de notre base de données. Comme nous avons pu le constater en la parcourant, cette dernière a été constituée par peu de personnes différentes. Ainsi, la diversité des environnements et des prises de vue est relativement limitée. Cela limite fortement la capacité du réseau à généraliser et donc à obtenir les mêmes performances sur des photos non-issues de la base de données. Nous avons en effet constaté cet effet avec des prédictions réalisées sur des images récupérées via *Google Images.* Cependant, cette seconde méthode d'évaluation est difficilement généralisable et nous aurait demandé beaucoup trop de temps pour la rendre réellement instructive. Encore une fois, la taille limitée de la base de données est au cœur de la problématique.

Les graphiques ci-dessous présentent quelques déroulements d'entraînement de CNN via l'évolution de l'*Accuracy* et/ou de la *Loss* (fonction de coût cherchant à être minimisée par le réseau) :





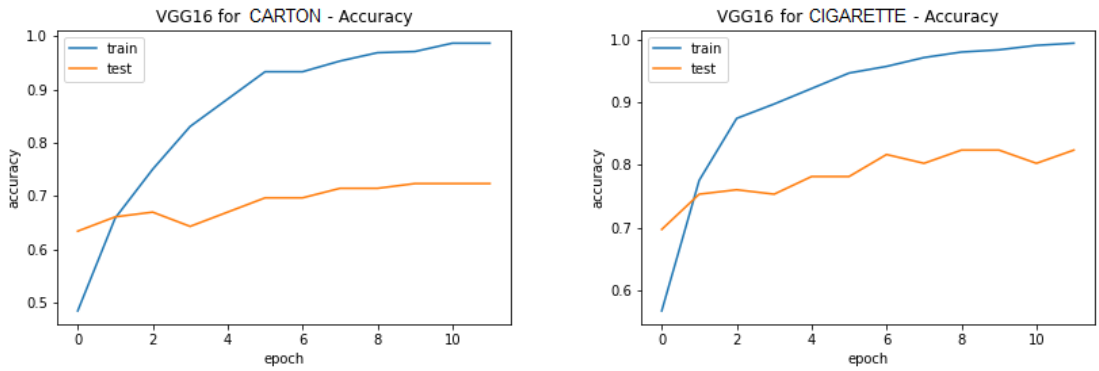


Figure 3.7.4 : Quelques déroulements d'entraînement des CNN

Nous aurions pu entraîner les modèles sur plus d'époques (plus d'itérations), cependant les résultats n'auraient pas forcément été meilleurs. Comme on peut le voir, à partir d'un certain point, le réseau de neurones fait du surapprentissage (il apprend par cœur les données d'entraînement), ce qui résulte en une baisse des performances (stagnation puis diminution de l'*accuracy*, et augmentation de la *loss*).

Enfin, les modèles renvoient un nombre compris entre 0 et 1. Nous choisissons d'interpréter un résultat supérieur à 0,65 comme la présence sûre du déchet concerné, un résultat inférieur à 0,5 comme l'absence sûre dudit déchet, et un résultat intermédiaire comme une incertitude. De plus, l'utilisateur peut décider de considérer cette zone d'incertitude comme une zone de rejet et donc d'interpréter une prédiction supérieure à 0,65 par une présence du déchet et inférieur à 0,65 par une absence. Cela produit un système qui privilégie les vrais positifs et limite les faux positifs. À l'inverse, si le souhait est plutôt de limiter les faux négatifs, l'utilisateur peut plutôt choisir comme seuil de décision 0,50. Nous avons décidé de fixer des seuils communs pour tous les CNN de classification puisque la trop faible précision de ces derniers nous empêche d'affiner raisonnablement ces valeurs pour chaque modèle. Enrichir la base de données permettrait ici encore d'affiner le traitement des prédictions fournies par les réseaux de neurones.

8) Tests d'autres architectures (mai)

Lors de la rencontre avec Léo Milecki, ce dernier nous a conseillé de tester d'autres architectures de CNN déjà existantes afin de comparer leurs différentes performances sur notre problème. L'utilisation de Tensorflow s'est alors révélée très avantageuse puisque nous n'avons pas eu besoin de remettre en place la totalité du processus de préparation des images et de transfert learning mais juste à adapter ce que nous avions déjà instauré pour VGG-16. Comme nous l'a recommandé Léo Milecki, nous avons testé les deux architectures ResNet-50 et EfficientNet-B3. Le premier est sensiblement le plus gros avec environ 23 millions de paramètres contre 14 millions pour VGG-16. EfficientNet-B3 lui ne présente qu'environ 11 millions de paramètres.

Comme déjà observé avec VGG-16, les résultats obtenus en entraînant la totalité des nouveaux réseaux et pas seulement les couches hautes de classification sont légèrement meilleurs. Cependant, nous n'avons pas obtenu de résultats significativement meilleurs avec ResNet-50 et EfficientNet-B3 que ceux que nous avions déjà avec VGG-16. À titre d'exemple, les graphiques ci-dessous présentent les résultats obtenus en classification binaire sur la catégorie « Bottle » pour les trois modèles entraînés entièrement :

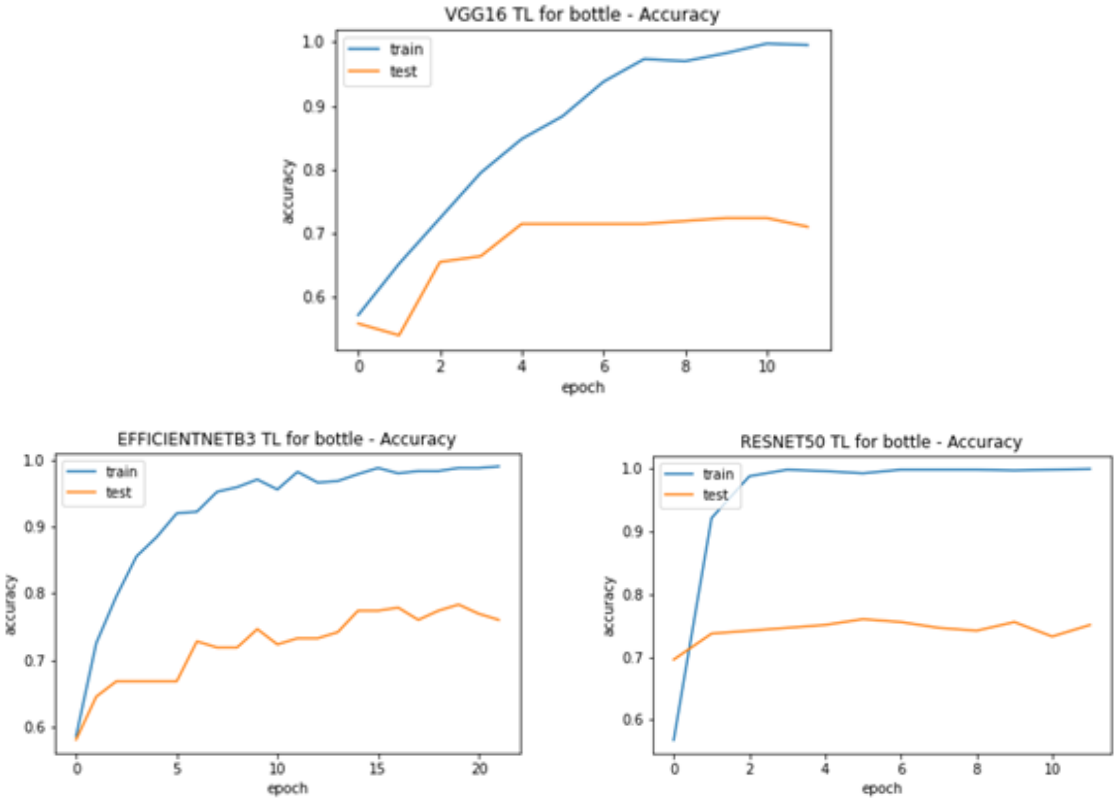


Figure 3.8.1 : Comparaison des performances en Full Learning des architectures VGG-16, ResNet-50 et EfficientNet-B3 sur la catégorie « Bottle »

Après avoir réalisé ces essais de nouvelles architectures, nous avons décidé de conserver l'architecture de VGG-16 pour notre projet. D'une part, il n'était pas nécessaire au vu des résultats obtenus avec un réseau plus important comme ResNet-50 de changer pour un modèle plus complexe. D'autre part, nous avons acquis au fil du projet une plus grande connaissance sur VGG-16 que sur tout autre réseau comme EfficientNet-B3 qui nous a posé des problèmes techniques liés aux versions utilisées de Tensorflow.

Les tests réalisés nous auront tout de même permis de confirmer notre hypothèse selon laquelle le principal facteur limitant des résultats obtenus lors de ce projet fut bien la base de données à notre disposition. En effet, les réseaux testés ont déjà démontré leurs grandes performances lorsqu'ils sont combinés à du transfer learning et entraînés sur plusieurs milliers d'échantillons. Rappelons que notre base de données contient un peu moins de 2000 images pour une vingtaine de catégories significatives (c'est-à-dire qui contiennent au moins 60-80 images). La catégorie la plus fournie est la catégorie « plastic bag » avec seulement 436 photos. Ces chiffres contrastent fortement avec ceux de l'univers original de VGG16. En effet, VGG16 est entraîné sur plus de 14 millions d'images issues de la base de données « ImageNet » (sur laquelle nous n'avons pas eu la possibilité de mettre la main) pour 1000 classes, soit en moyenne 14000 images par classe. Il est aussi important de noter que même en s'entraînant sur 14 millions d'images, VGG16 a beaucoup de mal à reconnaître les images de bouteilles de notre base de données : ce modèle renvoie dans l'ordre décroissant la probabilité que la photo représente une classe donnée, et (sans rentrer dans les détails), lorsqu'on lui faisait reconnaître une bouteille, il ne plaçait que très rarement cette catégorie dans le top 3 des catégories reconnues.

9) Interprétation des prédictions avec LIME

Après avoir entraîné nos différents CNN de classification binaire, nous les avons testés sur quelques nouvelles photos (autres que celles de la base de données). Nous avons alors constaté que pour certaines photos que nous ne considérions pas comme « difficiles » pour nos réseaux, les résultats fournis pouvaient être très fortement erronés. Ainsi, comme l'illustre la **Figure 3.9.1**, certains réseaux peuvent parfois être très « confiants » dans leurs erreurs.

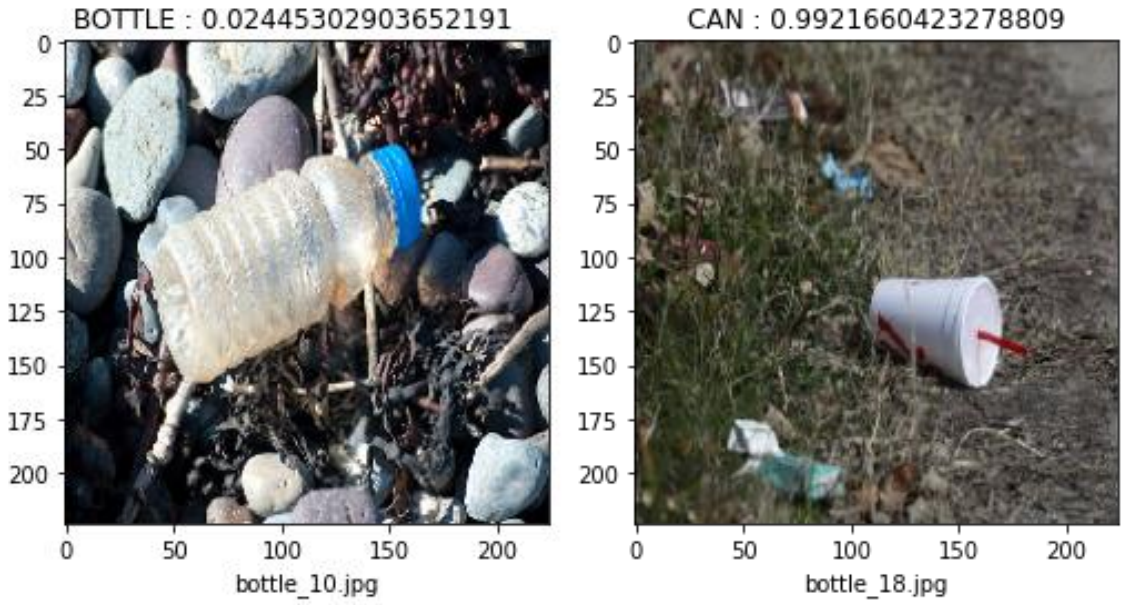


Figure 3.9.1 : Exemple d'un faux négatif avec le CNN « Bottle » (gauche) et d'un faux positif avec le CNN « Can » (droite).

Pour tenter de mieux comprendre les raisons de ces prédictions fausses, Jean-Philippe Poli nous a conseillé de nous renseigner sur la librairie python LIME (Local Interpretable Model-Agnostic Explanations). Cette librairie permet en particulier d'analyser le résultat de classifications réalisées par un CNN en identifiant les zones de l'image ayant fortement pesé dans la décision. Nous avons donc décidé de mettre en œuvre cette analyse que nous avons dû légèrement adapter. En effet, « l'explication » fournie par LIME consiste en un coloriage appelé « masque » des pixels d'importance sur l'image d'origine qui est dans notre cas pré-traitée pour répondre aux exigences de VGG-16. Afin de visualiser plus facilement ces pixels d'importance, nous avons plutôt préféré appliquer ce masque sur les images d'origine.

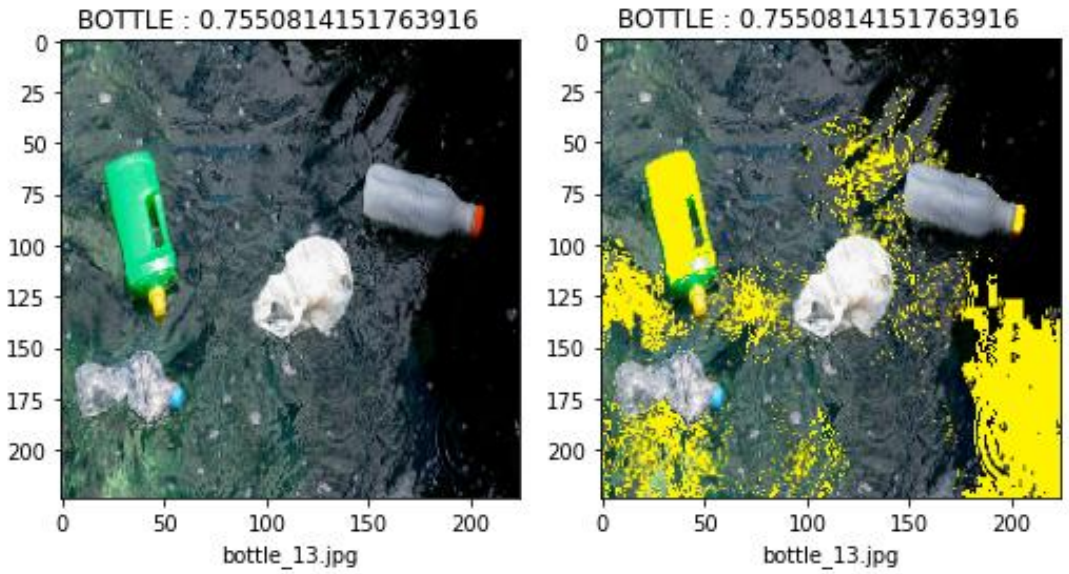


Figure 3.9.2 : Exemple avec le CNN « Bottle » de prédiction simple (gauche) et avec masque appliqué sur l'image sans pré-traitement (droite)

Une fois ces détails techniques réglés, nous avons pu identifier différentes causes de fausses prédictions de nos CNN. Tout d'abord, la ressemblance entre les objets à reconnaître pose de grandes difficultés. En analysant le faux positif « Can » de la **Figure 3.9.1** avec LIME, nous sommes maintenant assurés que le CNN « Can » a du mal à faire la différence entre un gobelet (« Cup ») qu'on aperçoit sur l'image et une canette. On peut voir sur la **figure 3.9.3** que le masque fourni par LIME recouvre principalement le gobelet, c'est bien lui qui attire l'attention du réseau. Cependant, nous tenons absolument à différencier ces deux catégories proches puisque qu'un gobelet généralement en carton ou en plastique a un impact bien différent sur l'environnement de celui d'une canette en aluminium. De même, une confusion entre un sac plastique et un masque chirurgical est également fréquente du fait de leur aspect proche.

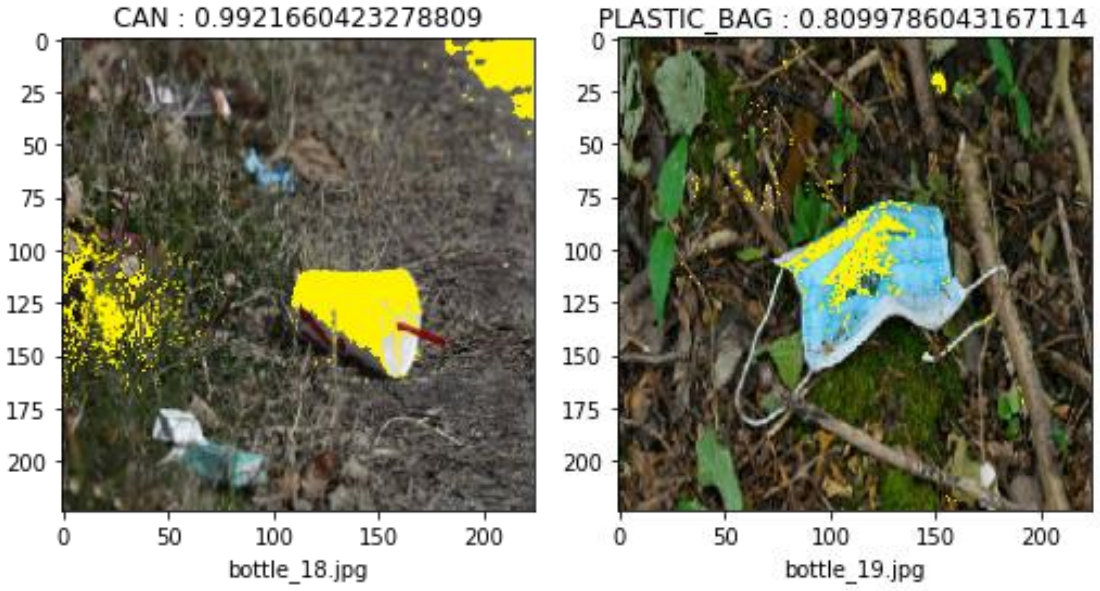


Figure 3.9.3: Exemple de confusion pour les CNN « Bottle » (gauche) et « Plastic bag » (droite)

Les CNN éprouvent également des difficultés à différencier le « bruit » présent sur une photo d'un déchet réel. Les masques fournis par LIME présentés en **Figure 3.9.2 et 3.9.3** en témoignent bien. Ce que nous appelons ici « bruit » n'est rien d'autre que le fond de la photo. Ce problème est particulièrement important dans notre projet puisque les déchets sont capturés dans leur environnement naturel drastiquement différent d'un fond blanc de laboratoire par exemple. Les détails autres que le déchet sur une photo sont parfois tellement importants que ce dernier n'est presque pas pris en compte par le CNN qui aboutit donc à une prédiction infondée et très probablement fausse. Cela est en particulier le cas lorsque l'image n'est pas très bien cadrée ou de maigre qualité. C'est cependant un problème de fond puisque dans le cadre du projet de notre client, des photos contenant plusieurs déchets comme celle de la **Figure 3.9.4** sont amenées à être analysées. On remarque alors qu'en fixant un seuil à 50 %, le sac plastique ci-dessous ne serait pas détecté par le CNN « Plastic Bag ».

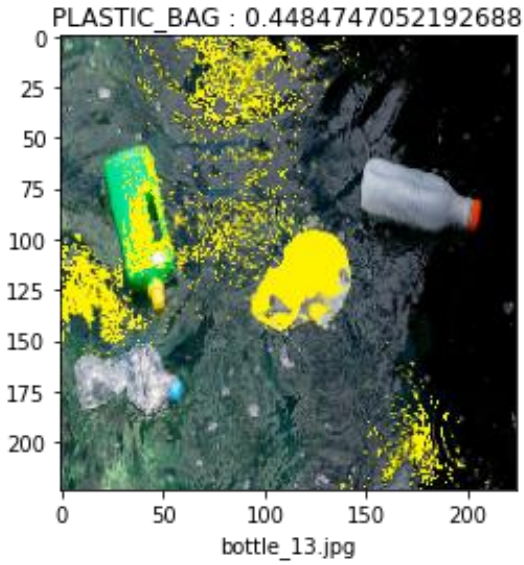


Figure 3.9.4 : Exemple de perturbation du CNN par d'autres éléments de la photo

Les interprétations obtenues avec LIME nous ont aussi permis de mettre en évidence de belles réussites de nos CNN. La **Figure 3.9.5** en présente quelques-unes. Cela démontre donc bien que pour des photos de bonne qualité, nos réseaux de neurones parviennent tout à fait à réaliser des prédictions correctes avec certitude.

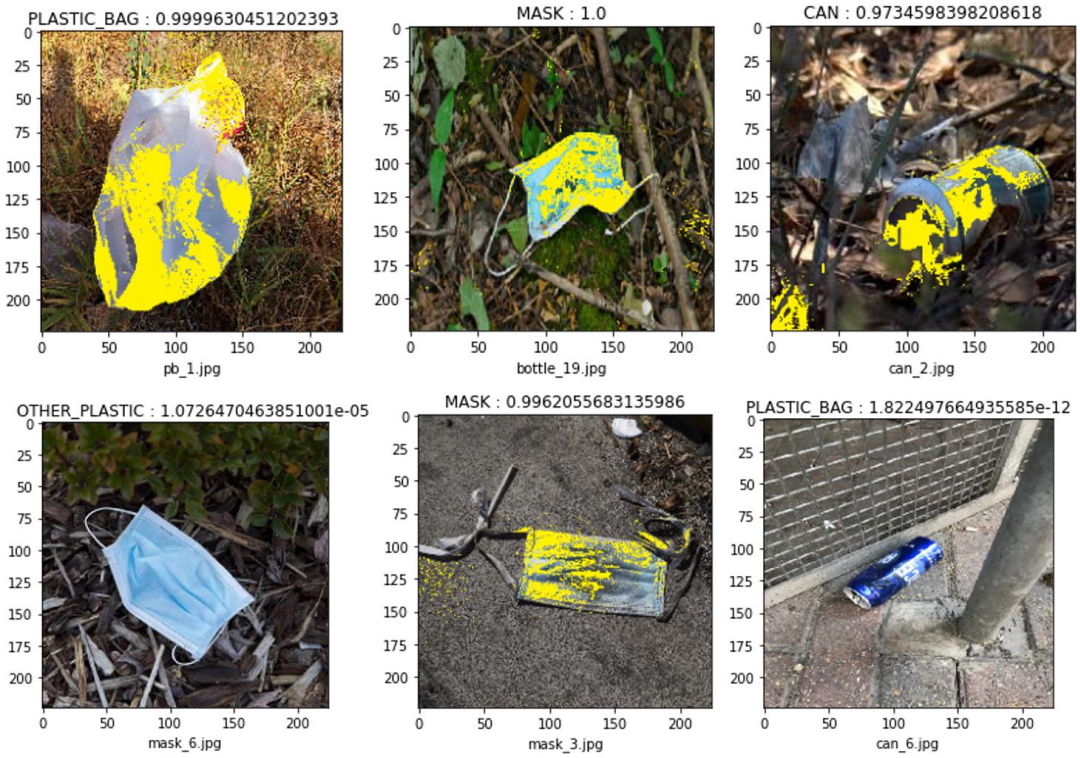


Figure 3.9.5 : Quelques réussites des CNN de classification

Rappelons aussi que cette analyse des prédictions grâce à LIME est difficilement applicable à grande échelle car une seule image peut prendre entre 30 et 45 secondes pour être analysée. Ainsi, nous n'avons bien sûr pas pu mener ces analyses sur des centaines d'échantillons.

IV) Conclusion

Nous voulons tout d'abord insister sur le fait que le projet nous a réellement permis de monter en compétence dans diverses notions d'intelligence artificielle : la manipulation des CNN, la prise en main de la bibliothèque Tensorflow et de la librairie LIME en Python ainsi que l'exploitation d'un réseau tel que VGG-16 combiné au Transfert Learning. Néanmoins, il nous semble nécessaire de pointer du doigt quelques difficultés que nous avons rencontrées lors du déroulement du projet et donc donner quelques pistes de réflexion pour une suite possible de ce même projet.

En premier lieu, et comme nous l'avons précédemment évoqué, la qualité de la base de données est l'un des facteurs clés les plus importants, si ce n'est *le* plus important, pour la réussite d'un tel projet. Les outils d'intelligence artificielle ne sont en effet pas « magiques », la détection et la classification d'image utilisant les CNN étant étroitement liées aux données d'entraînement. Plus la base est fournie et les images de bonne qualité, plus les résultats seront probants. Nous avons été limités en termes de nombre d'images car nous n'avons trouvé qu'une seule base de données conséquente de déchets en libre accès mais aucune regroupant des déchets marins spécifiquement. Nous ne pouvons que conseiller au client de continuer à constituer une base de données de déchets, en veillant à ce que les images prises soient de bonne qualité comme évoqué plus haut dans notre rapport.

Élargir la base de données permettrait également d'affiner les catégories de déchets puisque les photos pourraient être séparées en un plus grand nombre de sous-ensembles de tailles suffisantes. Nous avons été très optimistes en début de projet, en espérant inclure dans nos algorithmes au moins une douzaine de catégories mais nous nous sommes finalement résignés à n'en inclure que dix car trop de catégories héritées de la base de données TACO ne contenaient pas assez de photos pour pouvoir entraîner un CNN. Sur ce point, il nous est difficile de conclure à ce stade sur le type de catégories à privilégier. La meilleure approche, selon nous, serait de renseigner un maximum d'informations lors de la constitution de la base de données pour ensuite pouvoir tester plusieurs combinaisons de catégories et voir laquelle donne les meilleurs résultats et permet une différenciation optimale par les réseaux de neurones.

Un autre point que nous voulons aborder est la gestion de ladite base de données. La solution que nous avons choisie était d'utiliser une base SQL puis de charger les photos en ligne sur un Drive associé à une adresse électronique dédiée au projet pour pouvoir l'exploiter sur Google Colaboratory, notre environnement de travail. Ce système n'est en réalité pas la chose la plus pratique car le processus est souvent pénible et peu fluide. Par exemple, charger les photos à chaque fois peut prendre beaucoup de temps, surtout si celles-ci sont de bonne qualité. Nous avons heureusement réussi à mener notre projet ainsi puisque la taille de notre base de données était raisonnable pour un tel système (65 Mo). Nous conseillons cependant de chercher une solution plus optimale que la nôtre pour poursuivre ce projet, notamment en cherchant à gérer la base de données en ligne. Comme suggéré par Léo Milecki, il serait également avantageux d'utiliser un *Data Loader* pour mieux gérer l'importation et l'utilisation des images dans le code Python.

Après avoir analysé les différentes performances obtenues avec nos réseaux de neurones, nous en sommes venus à la conclusion que les CNN de classification binaire mis en place sont opérationnels pour des images de bonne qualité. L'élargissement de cette zone de fonctionnement ne peut se faire qu'en enrichissant la base de données d'entraînement. Il faut à la fois plus de photos de bonne qualité pour affiner la différenciation des CNN entre les catégories proches et des photos moins soignées pour améliorer la flexibilité des CNN et leur adaptation à l'environnement extérieur aux déchets et à de nouveaux types de photos.

V) Bibliographie

[1] M. Fulton, J. Hong, M. J. Islam and J. Sattar, "Robotic Detection of Marine Litter Using Deep Visual Detection Models," *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5752-5758, doi: 10.1109/ICRA.2019.8793975.

[2] Base de données TACO (Trash Annotations In Context) : http://tacodataset.org/

[3] https://www.sciencedirect.com/science/article/abs/pii/S016974392100037X?dgcid=rss_sd_all

[4] Rawat, W. and Wang, Z., 2017. Deep convolutional neural networks for image classification: A comprehensive review. Neural computation, 29(9), pp.2352–2449.

[5] https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5097666-tp-implementez-votre-premier-reseau-de-neurones-avec keras, “Classez et segmentez des données visuelles”

[6] Md. Zahangir Alom, Tarek M. Taha, Christopher Yacopcic, Stefan Westberg, Paheding Sidike, Mst Shamina Nasrin, Brian C. Van Essen, Abdul A. S. Awwal, Vijayan K. Asari : The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. CoRR abs/1803.01164 (2018)

[7] Ashley Chung, Sean Kim, Ethan Kwok, Michael Ryan, Erika Tan, Ryan Gamadia, “Cloud Computed Machine Learning Based, Real-Time Litter Detection using Micro-UAVSurveillance", 27 juillet 2018

[8] https://neurohive.io/en/popular-networks/vgg16/, “VGG16 - Convolutional Network for Classification and Detection”

[9] https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[10] Tan M, Le Q V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Porceedings of the 36th International conference on machine learning, 2019 : 6105-6114.

[11] https://scikit-learn.org/stable/modules/multiclass.html#multilabel-classification, “Multiclass and multioutput algorithms”

[12] Read J., Pfahringer B., Holmes G., Frank E. (2009) Classifier Chains for Multi-label Classification. Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2009. Lecture Notes in Computer Science, vol 5782. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04174-7_17

[13] Base de données TIDY : https://github.com/gale31/TIDY

VI) Table des figures