Rapport de stage 1A

Binko (SAS Mister Bin)

<u>Intitulé du stage</u> : Stage – Ingénieur Prototypage – Deep Learning dans la structure Binko (SAS Mister Bin)

Elève : Cléa Han, Promo 2021, 1A Alpha

Maître de stage : Romain Pellat

Tuteur école : Stéphane Bourgeois

Période du stage : Du 27 juillet 2022 au 19 août 2022

Remerciements

Je tiens à remercier toutes les personnes qui m'ont soutenue tout au long de mon stage et qui ont contribué à son bon déroulement et à son succès.

Tout d'abord, j'adresse mes remerciements à mon maître de stage, Romain Pellat, président de Binko, pour son accueil et son accompagnement. Grâce à sa confiance j'ai pu m'accomplir totalement dans mes missions et m'épanouir au sein de l'équipe. Il m'a également permis de découvrir le monde de la start-up.

Je remercie également toute l'équipe de Binko, le directeur général Valentin Violleau, ainsi que mes collègues de travail Hamza Madhi et Adrien Pointreau, pour leur accueil et leur aide. Ils m'ont permis de m'épanouir dans mon stage et d'enrichir mon expérience du travail.

Je tiens à remercier ma mentore d'Article 1, Katia Lucas, qui m'a vivement soutenue dans ma recherche de stage et tout au long de mon stage.

Je remercie également mon tuteur de stage, Stéphane Bourgeois, pour son accompagnement, ainsi que sa disponibilité.

Résumé

Ce stage 1A du 27 juin au 19 août m'a permis de mettre à profit des compétences acquises au cours de ma formation à Centrale Marseille et des compétences acquises dans mes activités associatives, notamment celles en lien avec Christopher.ai, l'association d'intelligence artificielle de l'Ecole centrale de Marseille. J'ai travaillé sur la mise en place d'un système de reconnaissance de déchets sans code-barres à l'aide de l'intelligence artificielle, qui constituait donc la ligne directrice de mon stage et dictait toutes mes tâches au cours de ces deux mois.

La base de données constituée et les codes réalisés ont permis de répondre correctement aux principales contraintes et aux attentes de ma mission. Ces livrables sont adaptables et serviront de base à l'avancement du prototypage de la poubelle intelligente développée par Binko.

Abstract

This 1A internship from June 27 to August 19 allowed me to use skills acquired during my first academic year at Centrale Marseille and skills acquired in my associative activities, especially those related to Christopher.ai, the artificial intelligence association of Ecole centrale de Marseille.

I worked on the implementation of a barcode-free waste recognition system using artificial intelligence, which was therefore the guideline of my internship and dictated all my tasks during these two months.

The database built and the codes produced allowed me to correctly answer the main constraints and expectations of my mission. These deliverables are adaptable and will be used as a basis for the prototyping of the smart bin developed by Binko.

Mots-clés

Intelligence artificielle, IA

Déchets sans code-barres

Prototypage

Base de données, BDD

TensorFlow

Deep Learning

PyTorch

CNN (Convolutional Neural Network)

Classification des déchets

AWS (Amazon Web Services)

Glossaire

- Machine Learning : apprentissage automatique en français, technique informatique se basant sur les probabilités afin de permettre aux ordinateurs d'apprendre par eux-mêmes sans programmation explicite.
- Deep Learning : apprentissage profond en français, ensemble de méthodes d'apprentissage automatique permettant aux ordinateurs d'apprendre par eux-mêmes en se basant sur un réseau de neurones artificiels.
- Dataset : un ensemble de données, ici un dataset est un ensemble de données issues d'une BDD.
- GPU (Graphical Processing Unit) : processeur graphique, le GPU est une unité de calcul, destiné aux calculs mathématiques rapides.
- Tensorflow : TensorFlow est une bibliothèque open source en Python destiné au Machine Learning, créée par Google. Elle propose une multitude d'outils pour entraîner et optimiser des réseaux de neurones artificiels.
- PyTorch : PyTorch est une bibliothèque open source en Python pour l'apprentissage machine développée par Meta (ex-Facebook). De la même manière que pour Tensorflow, elle permet également de mettre au point des réseaux de neurones.
- CNN : (Convolutional neural network) technologie permettant le traitement d'images par des réseaux de neurones.
- ResNet (Residual neural network) : ResNet désigne les réseaux résiduels, un type de réseau de Deep Learning.
- MobileNet : un type de réseau de Deep Learning avec une profondeur de réseau caractéristique qui lui est propre.
- Générateur : partie du code d'une IA chargée d'aller chercher et générer les données dans le disque dur quand celles-ci sont nécessaires, pour l'IA.
- Data-augmentation : technique permettant d'augmenter virtuellement la taille d'une base de données via des processus de modification de ces données
- AWS (Amazon Web Services) : division du groupe Amazon, spécialisée dans les services de cloud computing ("l'informatique en nuage") à la demande pour les entreprises et pour les particuliers, leur permettant d'accéder à des serveurs informatiques à distance.

- EC2 (Amazon Elastic Compute Cloud) : service proposé par AWS permettant à ses utilisateurs de louer des serveurs sur lesquels exécuter leurs programmes applicatifs.
- GitHub : plateforme d'hébergement, de versionnage de code et de collaboration autour de projets informatiques.
- RAM (Random Access Memory) : mémoire vive ou mémoire à accès aléatoire, mémoire informatique destinée au stockage d'informations traitées par un appareil.
- Protocole SSH : SSH désigne Secure Shell, c'est un protocole de communication et un programme informatique permettant la connexion à une machine distante (serveur) pour transférer des données et des commandes par liaison sécurisée.
- WinSCP : logiciel utilisant le protocole SFTP ou SFP, permettant la copie sécurisée de données entre l'ordinateur local et une machine distante.
- Adresse IP : numéro d'identification attribué à un appareil connecté au réseau Internet.
- Groupe de sécurité : méthode d'attribution à l'accès aux ressources d'un certain réseau protégé par ce groupe de sécurité.
- Overfitting : désigne le sur-apprentissage en français, en Machine Learning, il arrive lorsque l'algorithme apprend non seulement sur les données de la problématique fournie, mais aussi sur des éléments qui ne sont pas liés au problème traité.
- Fonction de coût, ou loss : elle quantifie l'écart entre la réponse réelle et la réponse théorique.
- Métrique : indicateur d'avancement ou de qualité pour un programme informatique pour en évaluer les performances.
- Accuracy : signifie exactitude, c'est une métrique permettant de mesurer la proximité des mesures, des prédictions à des valeurs spécifiques, des labels.
- TensorBoard : outil informatique permettant de visualiser l'évolution des performances d'un modèle au cours de son entraînement.
- Hyperparamètres : en Machine Learning, un hyperparamètre est un paramètre dont la valeur permet de contrôler le processus d'apprentissage.
- Data-augmentation : technique permettant d'augmenter virtuellement la taille d'une base de données via des processus de modification de ces données;
- Dropout : technique qui est destinée à empêcher le sur-ajustement sur les données d'entraînement en abandonnant des unités dans un réseau de neurones.

Table des matières

Introduction

Face à l'urgence écologique, mon stage de 1A se voulait répondre aux besoins actuels de la société tout en mettant en pratique mes compétences acquises lors de ma première année à Centrale Marseille. Ainsi, parmi les différentes entreprises auprès desquelles j'ai postulées, Binko me permettait de porter un projet écologique tout en sollicitant mes compétences Centraliennes et mes connaissances en intelligence artificielle (notée "IA" par la suite) acquises dans le cadre de mon activité de vice-présidente au sein de Christopher.ai, l'association d'IA de Centrale Marseille. En effet, Binko est une start-up portant un projet d'amélioration du recyclage des déchets se basant sur l'utilisation d'une IA afin de reconnaître les différents déchets sans code-barres.

Le stage dure 8 semaines au sein de leur locaux situés à l'incubateur Makesense au 11 rue Biscornet, 75012 Paris.

L'objectif de mon stage 1A sera de comprendre la structure de l'entreprise, son fonctionnement et ses différents rôles et d'assister l'entreprise dans ses missions et ses difficultés.

Présentation de l'Entreprise

BINKO est une start-up fondée par Romain Pellat et Valentin Violleau domiciliée à Versailles (78000), en juin 2021 dont le secteur d'activité correspond à la fabrication d'articles métalliques ménagers. De manière plus concrète, elle propose une solution intelligente pour la gestion des déchets en améliorant le processus de recyclage : une poubelle connectée, qui porte le même nom. Cette poubelle rend le recyclage plus simple, plus économique, et plus efficace. Elle reconnaît, broie et tri les déchets automatiquement. En effet, le tri à la source permet d'envoyer les déchets directement au centre de recyclage sans passer par le centre de tri. Ceci permet alors aux mairies de faire de sérieuses économies tout en recyclant 3 fois plus de déchets. De plus, le projet porté par Binko permet d'accompagner la population dans les démarches de tri et de recyclage qui sont et seront actées par la loi. Ce projet assiste le tri et le recyclage pour que les déchets soient valorisés de manière optimale, afin d'éviter au plus l'impossibilité de trier ou de recycler pour des raisons économiques.

Portée par la French Tech Tremplin, Binko a actuellement ses bureaux à l'incubateur Makesense, à Paris, où j'ai pu réaliser mon stage de deux mois. Elle a également sa place à l'Electrolab de Nanterre, qui abrite ses projets électroniques. Des municipalités comme Versailles (78000), Saint-Paul-en-Pareds (85500) et Sainte-Hermine (85210) ont déjà

adressé leurs lettres d'intention à Binko pour manifester leur soutien et leur intérêt pour la poubelle connectée qui sera déployée par Binko à la fin de son processus de prototypage de la poubelle Binko qui est prévue pour avril 2023.

Binko est soutenue par l'incubateur Station F, la ville de Paris et le groupe GRDF. La start-up est également lauréate de Start'in ESS[1] 2020-2021 et lauréate de "Act for Impact"[2].

Sa composition

Binko est composée tout d'abord de deux cofondateurs : Romain Pellat, le dirigeant de la start-up et Valentin Violleau, le directeur général; ils sont tous deux issus de la même promotion d'école de commerce ESSCA, diplômés en 2019. La start-up accueille également 3 stagiaires dont moi : Hamza Madhi, un étudiant en Master à Polytechnique qui s'occupe de la partie mécanique du projet, Adrien Pointreau, un étudiant en Master à Télécom Paris qui s'occupe de la gestion de la base de données (notée "BDD" par la suite) de l'ensemble du projet Binko, puis moi, qui m'occupe de la partie IA du projet.

Son marché

La start-up a pour but de servir les municipalités, qui sont ses clients. Les municipalités achètent les Binkos et les installent gratuitement chez les particuliers, qui sont donc les utilisateurs finaux. Cette solution est alors gratuite pour les particuliers, car elle est entièrement financée par la taxe d'enlèvement des ordures ménagères.

Étant donné que Binko permet de faire un lien direct entre ses utilisateurs finaux et les centres de recyclage, leurs concurrents principaux sont les centres de tri, qui effectuent le relais intermédiaire.

Sa situation économique actuelle

Binko est actuellement en phase de prototypage, de recherche et développement. Ainsi elle s'appuie sur un fond initial personnel et a fait appel à une campagne de financement participatif, Ulule, qui a permis à la start-up de faire découvrir son projet auprès de contributeurs dès novembre 2021 en récoltant 9 467€ pour un objectif initial à 6 000€. Sa campagne s'est terminée mi-décembre 2021. L'entreprise participe également à de nombreux appels à projets et de concours pour start-ups innovantes, contribuant à la

[1] Dispositif mis en place par la ville de Paris qui accompagne de jeunes entrepreneurs porteurs de projets qui veulent s'engager, entreprendre et innover dans le domaine de l'Économie Sociale et Solidaire.

[2] label créé par BNP Paribas pour soutenir les entrepreneurs innovants en vue d'avoir un impact positif sur la société ou l'environnement.

transition écologique et sociale, lui permettant de grandir en nom et de récolter ses plus importants fonds pour la conception de sa Binko.

L'objectif est de créer un produit industrialisable final. Pour cela, en plus des démarches légales et administratives, l'entreprise passe dans plusieurs phases qui sont le prototypage, le passage au banc d'essai pour le respect des normes de sécurité et des normes européennes, puis la conception pour réaliser des moules à injection plastique et l'assemblage de l'électrique, de la mécanique et de la mécatronique du produit auprès de bureaux d'études. Tout cela demande un coût élevé estimé à 380 000€ dont 40 000 à 50 000€ réservés à la création du moule à injection plastique qui lancera le début de l'industrialisation de la poubelle Binko. D'autre part, le travail des parties électriques, mécaniques et mécatroniques du projet sont destinés aux bureaux d'étude moyennant 80€ par heure de travail.

En plus des capitaux obtenus sur les concours et prix pour start-ups innovantes auxquelles Binko participe, cet apport financier permet de contribuer de manière optimale au développement du prototype de la poubelle connectée, et notamment à son développement technologique actuel qui est concentré sur le prototypage mécanique et le prototypage de l'IA et de la BDD derrière l'intelligence de la poubelle. L'entreprise y alloue un budget d'environ 41 000€ en attribuant ce travail à des stagiaires, car cela lui permet de maximiser le travail en interne afin de réduire les coûts externes qui seront liés à l'appel de bureaux d'études plus tard dans son processus d'industrialisation. La fin de cette première étape de prototypage numérique de Binko est prévue pour décembre 2022.

Son projet et ses objectifs

Binko développe une poubelle connectée qui permet de reconnaître les déchets qui lui sont montrés par l'intermédiaire d'une caméra suivant une technologie que la start-up souhaiterait breveter. Il y a deux cas d'utilisation de la caméra principalement : soit la caméra intégrée reconnaît le code-barre d'un produit, soit il n'y a pas de code-barres.

Dans le cas où la caméra reconnaît un code-barre sur le déchet, Binko catégorise le produit, qui est ensuite broyé ou bien compacté, puis il est trié vers le bon bac. Cette partie du projet a déjà été avancée par la start-up avant mon arrivée, grâce à une technologie de texte de code barre et de QR code, développée avec leur partenaire Google France. Ce qui a permis à l'entreprise de se constituer une base de données de 819 120 produits pour identifier un maximum de déchets, avec code-barres.

Dans le cas où le déchet ne possède pas de code-barre, la classification du produit se base sur une intelligence artificielle qui a pour but de réussir à catégoriser et différencier les déchets sans code-barres du quotidien français. On m'a alors confiée l'amorçage de cette partie nécessitant de l'IA. On m'a dès lors confiée la responsabilité de débuter cette partie et d'en fournir les bases afin de contribuer au prototypage de l'IA de la poubelle intelligente.

Ensuite, lorsque le bac est plein, il est collecté puis envoyé directement au centre de recyclage, ce qui permet une revalorisation du produit entamé.

Ses performances par rapport à une poubelle classique

Chaque Binko est composée de deux matériaux, l'un en matière naturelle biosourcée, l'autre en matière recyclable non toxique. Elle est donc fabriquée à partir de nos déchets comme matière première. D'autre part, la fabrication de la Binko relève d'une démarche éco-responsable: elle est développée et fabriquée à 97% en France avec l'aide de bureaux d'études partenaires et d'un industriel français spécialisé dans l'injection. De plus, elle sera désignée de telle sorte à apporter également un meilleur confort à ses utilisateurs, car elle sera constituée de bacs hermétiques, inodores et solides.

Ce processus automatisé de recyclage simplifie la vie de ses utilisateurs en leur permettant de sortir trois fois moins leur poubelles, car le bac plein est à sortir une à deux fois par mois, contrairement à deux fois par semaine environ pour une poubelle classique grâce au compactage des déchets. En effet, Binko permet de réduire jusqu'à 12 fois la taille des déchets grâce au compresseur et réduit les erreurs de tri grâce à sa technologie. Elle permet donc une meilleure revalorisation de nos emballages plastiques, qui étaient revalorisés à 20% avec une poubelle classique, contre 89% avec Binko.

Ainsi la poubelle Binko permet de diviser les coûts de recyclage par 6, tout en restant attractif pour les utilisateurs et financée par nos impôts: la taxe d'enlèvement des ordures ménagères.

Présence et vie

Ma place dans leur incubateur - Makesense

Binko étant hébergée par l'incubateur Makesense, toute l'équipe travaille dans leur espace de coworking. Mon cadre de travail a dû être également dicté par l'environnement de travail de Makesense. Ainsi pour chaque nouvel arrivant à l'espace, aussi bien pour les

nouveaux stagiaires des start-ups incubés chez eux ou pour les nouveaux employés de Makesense, j'ai dû suivre une présentation et une formation d'une heure pour connaître l'esprit et les règles de vie dans l'incubateur.

En effet, Makesense est une entreprise qui supporte des start-ups soutenant des causes sociales ou écologiques, mais aussi divers projets soutenant la transition sociale et écologique, à travers le monde en étant présente à travers différents pays du monde, passant de l'Amérique du Sud à l'Asie.

Le lieu est principalement composé d'un espace de convivialité composé d'une cuisine ouverte et d'un salon, puis d'un open-space où des bureaux sont mis à disposition pour pratiquer le "flex office", c'est-à-dire que chacun prend le bureau à disposition qu'il souhaite. Makesense met également à disposition plusieurs salles de réunion disponibles sur réservation et des box d'isolation acoustique permettant de passer des appels ou assister à des visioconférences. C'est un lieu de vie collective où tout le monde contribue à son bon fonctionnement en participant collectivement aux tâches et corvées de l'espace commun s'il y en a besoin.

À l'issue de la formation, j'ai eu également accès aux plateformes de communication de Makesense qui utilise principalement l'application *Slack*, une application de messagerie pour les entreprises, mettant en contact les différents collaborateurs. Cette application permet de nous tenir au courant des événements ou des activités à venir, d'échanger des informations, demander de l'aide et de réserver certains espaces de l'open space. Tout le personnel, y compris les incubés et leurs salariés sont invités à participer aux événements de convivialité organisés par Makesense.

Ma place chez Binko

Je suis chez Binko en tant que stagiaire non rémunérée. Néanmoins, mon tuteur, qui est le dirigeant, m'accorde beaucoup de confiance et de responsabilités pour la mission qu'il m'a assignée. En effet, je suis non seulement responsable de la partie IA du projet, mais je suis aussi une force de proposition quant aux décisions stratégiques prises par la start-up, me permettant de contribuer activement au développement du projet dans sa globalité.

Nous formons une équipe de travail assez jeune : notre âge va de 21 ans à 25 ans. Les cofondateurs portent une certaine importance à l'âge de leurs futurs employés et stagiaires car ils estiment que cela permet de contribuer à une meilleure entente entre les collaborateurs, ainsi il y aura une meilleure ambiance de travail et un meilleur épanouissement.

Communication

L'entreprise a besoin que les salariés puissent communiquer correctement entre eux sur différents sujets. Binko a opté pour *Slack*, principal outil destiné à la communication interne pour l'ensemble des collaborateurs.

Slack permet d'échanger des informations sur l'avancement de missions et du projet sur différents canaux destinés à différentes parties du projet, de convoquer les participants aux réunions, de déposer des comptes rendus, ainsi que de prévenir en cas de modification des heures de travail ou de congés.

D'autre part, les cofondateurs de Binko utilisent également *Asana*, un gestionnaire de communication d'équipe, pour gérer la liste de tâches qu'ils ont, afin de tenir un carnet de bord de l'ensemble des tâches, quel que soit le domaine, synchronisé sur une plateforme et disponible en temps réel. Les stagiaires de Binko sont libres d'utiliser cette plateforme. Personnellement, j'utilise cette plateforme afin d'indiquer de manière transparente et concise mes tâches individuelles et ce que je prévois de faire dans le cadre de la mission qui m'a été attribuée vis-à-vis de mon tuteur.

Organisation

L'entreprise a adopté une organisation libre : aucun contrôle particulier n'est fait pour vérifier la présence des employés et des stagiaires, les absences pour raisons diverses sont signalées via un message sur *Slack*. Les locaux de Makesense sont ouverts de 9h à 18h. Une durée journalière de 7h est demandée à chaque stagiaire, avec une possibilité de reporter ses heures de travail en cas de coupure éventuelle.

La hiérarchie de Binko laisse une grande latitude à ses stagiaires en échange du sérieux et de l'assiduité de ceux-ci. En effet, les stagiaires sont libres d'aller au travail en présentiel ou en télétravail après en avoir prévenu leur tuteur, sans aucun contrôle sur leur présence, sous réserve de leur disponibilité sur leur horaires de travail dans la journée que ça soit sur *Slack* ou en personne s'ils sont au bureau, sauf s'ils préviennent de leur absence sur *Slack*.

Il y a également une attente de résultat de la part du tuteur. Des réunions sur l'avancement de leur mission peuvent se faire tous les jours ou bien dès que cela devient nécessaire. Les réunions sont souvent spontanées et proviennent soit d'une demande du tuteur soit des stagiaires. Elles peuvent durer de 30 min à 3h en fonction du sujet abordé. En effet, lors du début de mon stage, mes réunions avec mon tuteur pouvaient durer jusqu'à 3h afin de saisir au mieux ses exigences et la mission qu'il me confiait. Puis, après la prise en

main de la mission, je tenais mon tuteur au courant de l'avancé de mon travail une fois par semaine au minimum en lui exposant ce qui était prévu par la suite. Au cours de la semaine, mon tuteur se tenait disponible pour répondre à mes interrogations éventuelles ou à m'épauler dans les tâches secondaires que je pouvais lui confier concernant ma mission. D'autre part, mon tuteur est soucieux de l'épanouissement de ses stagiaires et donc prend également le temps sur ces temps de réunion de me demander mon avis quant à l'accompagnement qu'il m'apporte en tant que tuteur de stage.

Concernant la santé et la sécurité au travail, mon stage est un travail sédentaire dans un bureau, ce qui peut engendrer certains risques de santé et donc à une démotivation générale dans le travail, ainsi qu'une baisse de productivité, ce qui coûterait à la start-up. Néanmoins, mon tuteur s'intéresse beaucoup au bien-être de ses employés, car il prend très au sérieux mon état de santé au travail. D'autre part, il est toujours disponible pour parler de la manière de me manager, afin que je me sente épanouie au sein de l'équipe.

Période d'autonomie

Du 1 août au 14 août, mon tuteur a pris son congé annuel. Ainsi, j'ai eu une période d'autonomie de deux semaines. Sur cette période, il ne m'est plus possible de joindre mon tuteur sur *Slack*. Néanmoins, en cas d'urgence ou de nécessité, mon tuteur reste disponible sur son téléphone personnel dont il m'a laissé le numéro.

Avant ses vacances, j'ai eu une réunion avec mon tuteur pour faire un point sur l'avancement global du stage et de la mission, puis de lui parler de mes prévisions pour ces deux semaines d'autonomie, afin de pouvoir être sur la même longueur d'ondes et effacer toutes zones d'ombres ou interrogations.

Mon stage

Contexte et objectifs

Je suis intervenue dans cette start-up lors de leur premier prototypage technologique, en mettant en place le début de la partie intelligence artificielle de leur projet. On m'a confiée la responsabilité de débuter la partie IA du projet, et j'ai pu travailler en collaboration avec d'autres stagiaires qui eux travaillaient sur la partie mécanique et sur la base de données de Binko. Ma partie s'intéressait donc principalement à la détection de déchets ne possédant pas de code-barres, et ceci serait réalisée grâce à l'IA.

À terme, la classification par IA de déchets du quotidien sans code-barres devrait être dans un premier temps déployée sur une application mobile, dont la sortie de la première version est prévue pour décembre 2022. Cette application permettrait à l'utilisateur de prendre en photo son déchet, puis il serait classifié et catégorisé par l'IA implémentée dans l'application, permettant d'indiquer à l'utilisateur comment recycler son déchet, et par le même biais d'alimenter la BDD de l'entreprise pour une amélioration future de l'application. Cet objectif a pour but de répondre à des attentes d'investisseurs et des nombreuses municipalités qui lui ont adressé des lettres d'intention, et ainsi de se développer en tant que start-up et de gagner en crédibilité auprès d'un plus grand public sur la durée.

Tout d'abord j'ai effectué un travail de bibliographie et de recensement dans une base de donnée pour répertorier les déchets sans code-barre existants, puis j'ai mis en place l'entraînement d'une IA avec la bibliothèque Tensorflow puis avec la bibliothèque PyTorch, à l'aide de BDD libres de droits en ligne, afin de pouvoir donner une première version de codes. D'autre part, j'ai pu accompagner mon tuteur dans des prises de décisions et dans la réflexion autour de la croissance de la start-up et dans le développement stratégique de leur produit.

Base de données de déchets sans code-barres

Avant de commencer à coder la classification de déchets sans code-barres avec une IA, il a fallu effectuer un travail de recherche et de réflexion justement sur les déchets sans code-barres. En effet, ce travail permet d'amorcer la mise en place de l'IA, nous permettant de réfléchir aux nombres de "catégories" de déchets à considérer lors de leur classification. Ainsi, dans un premier temps, j'ai dû réfléchir à une BDD représentative et pertinente pour chercher à quantifier et qualifier les déchets sans code-barres de manière optimale.

En effet, étant dans une première étape de prototypage dont l'objectif est de sortir une première version du code, nous nous sommes intéressés principalement aux déchets les plus courants du quotidien français afin de mieux cibler le public visé par Binko. De plus, cette BDD de déchets sans code-barres doit prendre en compte le fait que les déchets peuvent être eux même composés de plusieurs matériaux différents recyclables, donc la BDD doit prendre en compte les différents états dans lesquels ils seront jetés par l'utilisateur. Par exemple, pour une bouteille de lait classique, il y a le film plastique entourant la bouteille, le couvercle, l'opercule et la bouteille de lait en elle-même. Selon la bouteille, si le film plastique l'entourant est toujours là, il pourra être détecté par le système de code-barres, ce qui permettra de savoir quelle est sa composition initiale et de donner des instructions en fonction de la marque de la bouteille. Cependant, si le film plastique a été enlevé par le consommateur c'est là que l'IA peut intervenir pour classifier la bouteille selon

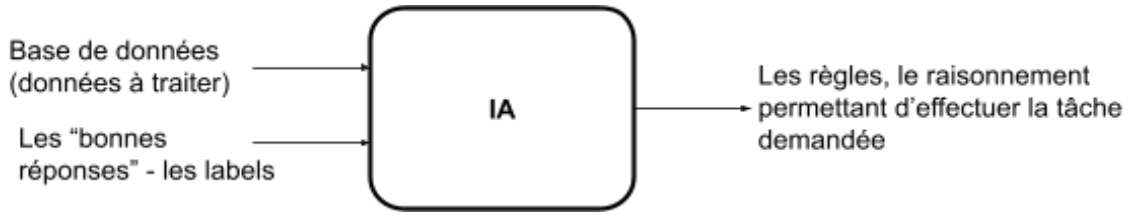
sa matière, ou si l'opercule ou le bouchon est resté sur la bouteille, à travers une photo que l'utilisateur enverra à l'application mobile.

Pour cela, j'ai réalisé une bibliographie sur les déchets des ménages français, afin d'en extraire des données sur les déchets sans code-barres :

Le document [1], dont les références sont dans la partie bibliographie, permet de fournir des données sur les déchets ménagers français par matériaux. D'autre part, lorsqu'on s'intéresse aux déchets municipaux en France [2], les déchets sans code-barres les plus souvent jetés sont ceux correspondants à des déchets destinés au recyclage. Au-delà des déchets recyclables, il y a beaucoup de déchets sans code-barres courants de la vie qui ne sont pas recyclables, voire même qui nécessitent des traitements spéciaux, comme les piles par exemple, ceux-ci sont également à prendre en compte dans la BDD selon le document [3]. Toutes ces données sur les déchets récoltés sont plus ou moins bien recyclées ou traitées d'une certaine manière par les ménages d'après le document [4], ainsi la BDD doit prendre en compte ces comportements afin d'obtenir une BDD la plus optimale possible pour la première version. En effet, les ménages ont tendance à jeter les bouteilles en plastique de type PET qui sont transparentes au même endroit que les bouteilles en plastique opaque. Malheureusement c'est une mauvaise croyance que les ménages ont car le recyclage des bouteilles PET est différent des bouteilles plastiques opaques. Ainsi, en suivant cet exemple, il faut prendre en compte de manière distincte les bouteilles en plastique PET et les bouteilles en plastiques opaques dans la BDD. Ainsi, à cette étape de la bibliographie j'ai pu établir une première BDD sous la forme d'un tableau Excel recensant tous les déchets sans code-barres du quotidien français. Afin d'affiner cette BDD, j'ai pu la compléter avec les données de l'application mobile *Guide du tri* de Citéo [5] qui a été vivement recommandée par mon tuteur de stage.

Ainsi, dans le cadre de mon stage qui se concentre principalement à l'élaboration d'une IA, ce travail de constitution d'une BDD se voulait concise, optimale et représentative et a été réalisé lors de ma première semaine de stage. J'ai recensé dans cette BDD environ 1500 types de déchets sans code-barres. Cette BDD n'est pas figée et est utilisée non seulement comme base pour réflexion dans la conception de la première version de l'IA que je vais coder, mais aussi pour le stagiaire s'occupant de la BDD générale de l'entreprise.

Fonctionnement globale de l'IA



L'IA est un ensemble de neurones à qui on va donner tous les moyens de résoudre la problématique, par le biais d'un entraînement. Dans notre cas, notre IA doit résoudre un problème de classification. Pour cela, il faut l'entraîner à réaliser de la classification. Ainsi, il faut lui fournir une base de données qui est représentative des objets à classifier par l'IA, ici des images de déchets, puis étiqueter les images de déchets qui représenteront le label des catégories finales dans lesquelles on veut classifier nos déchets.

Selon un certain modèle de réseau de neurones adapté au problème de classification, notre IA va s'entraîner sur toutes ces données qu'on lui donne en entrée puis il fournira en sortie une "réponse", la prédiction de l'IA, qui est représentée généralement sous la forme d'un vecteur mais interprétable par le suite afin de répondre de manière pertinente à la problématique. L'IA est donc capable de trouver de lui-même le raisonnement nécessaire pour effectuer la classification.

Par exemple, pour la problématique de Binko, si on ne voulait trier que des bouteilles en verre des bouteilles en plastique, il faudrait alors préparer une BDD d'images de bouteilles en verre et de bouteilles en plastique, puis pour chaque image les labelliser "bouteille en plastique" ou "bouteille en verre". Puis lorsque l'IA serait entraînée, on lui fournirait une image nouvelle de bouteille en verre par exemple, elle nous renverrait un vecteur à deux valeurs correspondant aux labels définis précédemment, où les valeurs sont interprétables comme des probabilités. La valeur la plus grande du vecteur de sortie est située à un indice du vecteur correspondant à un des deux labels. Si l'IA est bien entraînée, dans cet exemple l'IA devrait alors nous indiquer que l'image de déchet correspond à une bouteille en verre. Puis cette donnée de sortie de l'IA serait traitée par un autre système de labellisation générale traitée par mon co-stagiaire qui s'occuperait d'y attribuer une consigne de recyclage.

Elaboration technique de l'IA : mise au point de sa structure et de son entraînement

En effet, ma mission consiste concrètement à établir une base de données d'images sur laquelle l'IA va s'entraîner, ensuite coder l'IA dont la structure sera détaillée dans les paragraphes suivants, puis de l'améliorer en jouant sur ses hyperparamètres, afin de la rendre la plus efficace pour résoudre le problème de classification selon un critère pertinent qui sera à définir et à choisir judicieusement.

Base de données

On suppose que l'utilisateur prend en photo ses déchets sous un certain cadrage: le déchet occupe la majorité de l'espace de la photo, de telle sorte à ce qu'on n'ait pas besoin de découper l'image du fond et pour qu'on puisse négliger le fond de l'image, le tout sous une luminosité correcte permettant de distinguer le déchet. En effet, chercher à segmenter le déchet de son fond pourrait ajouter de potentielles erreurs de segmentation et complexifierait inutilement l'IA, sachant que l'image est déjà supposée correctement cadrée. De plus, il n'y a toujours qu'un seul déchet par photo, pour éviter de faire de la "séparation" de déchets sur une même photo, car ce n'est pas une priorité pour la première version du code d'IA. Ainsi, pour différencier cette BDD de déchets à fournir pour l'entraînement de l'IA de la BDD créée sur Excel et expliquée précédemment, à partir de maintenant, ce que j'appelle BDD désigne la base de données d'images de déchets fournie à mon IA, et donc logiquement pour son entraînement. Cette BDD devra donc contenir des images de déchets cadrés selon les critères susmentionnés.

CNN et classes

Nous sommes dans une problématique de classification, ce qui signifie que nous allons nous intéresser aux modèles de réseaux de neurones CNN (Convolutional Neural Network), qui sont des systèmes de traitements d'images utilisés en Deep Learning. Un CNN nous permet donc de classifier des images dans des "classes", qui ici représentent les labels des déchets respectifs qu'on traite. On se concentre donc sur un modèle de CNN qui nous permettra de classifier au mieux toutes les classes des déchets qu'on aura prédéfini. En effet, de manière théorique, un CNN sera efficace quel que soit le nombre de classes qu'on lui impose, tant qu'il y a une base de données d'images assez diversifiée et grande dans chaque classe de déchet.

En effet, l'IA ne doit pas juste reconnaître la catégorie "bouteille" par exemple, car la bouteille en verre et la bouteille en plastique de lait ne se recyclent pas de la même manière. On définit donc le terme de "classe", une catégorie dans laquelle les déchets à considérer vont être classifiés. Ainsi, la BDD établie sur Excel précédemment permet de réfléchir au

nombre de classes à prendre en compte pour notre IA. De plus, on a vu que théoriquement le codage du CNN sera efficace quel que soit le nombre de classes considérées.

Par conséquent, par souci de simplicité et par contrainte de temps, j'ai décidé d'établir les codes d'IA en ne prenant en compte que 10 classes de déchets dans le cadre de mon stage, sachant qu'il sera directement adaptable pour un plus grand nombre de classes après mon stage, à condition d'avoir assez d'images dans la BDD pour entraîner correctement l'IA. Sachant cela, j'ai pu établir avec l'approbation de mon tuteur les 10 classes de déchets, en anglais, sans code-barres à considérer dans l'élaboration des IA dans le cadre de mon stage : **battery, cardboard, clothes, food, glass, mask, metal, paper, PET, shoes**. Ces 10 classes permettent de représenter les principaux déchets sans code-barres qui peuvent exister au sein d'un ménage français et ce sont également des objets dont les BDD d'images sont plutôt accessibles en ligne ce qui permet de concentrer mon stage sur l'IA au lieu de passer du temps à chercher des BDD ou d'en créer moi-même.

En effet, mon tuteur est en train de travailler sur un système de caméras permettant de prendre en photo un déchet sous différents angles sur un fond avec une luminosité optimale. Cette technologie n'était pas encore finie, je ne pourrai pas utiliser sa base de données afin d'entraîner mon IA dans le temps imparti de mon stage.

Récolter et nettoyer la BDD

Afin de constituer cette BDD d'entraînement, j'ai choisi de prendre des images libres de droit disponibles sur le site *Kaggle*, un site dédié à la communauté de Machine Learning et à la Data Science. Toutes les références des BDD sélectionnées ont été indiquées dans la partie bibliographie du rapport de stage, correspondant au [6]. En effet, ce site permet de récolter un grand nombre d'images pour la BDD d'entraînement, car pour une classification de 10 classes, il faut environ entre 500 à 1000 images par classe pour avoir une bonne représentation de chaque déchet, c'est-à-dire chaque classe considérée, afin d'avoir cette fameuse BDD grande et diversifiée.

Sachant qu'une BDD bien constituée et diversifiée est un facteur essentiel au bon entraînement de l'IA, c'est à cette étape de mon stage que j'ai pu demander à mon tuteur de m'épauler dans le travail de nettoyage des BDD récoltées sur *Kaggle*. En effet, certaines images ne sont pas assez réalistes : certaines images étaient des illustrations de déchet ou bien d'autres contenaient des filigranes ou des bandes publicitaires dessus. Ainsi, l'objectif était d'avoir une BDD la plus "propre" et réaliste possible en éliminant ces images indésirables, dès la fin de la deuxième semaine de stage.

Stratégie de codage de l'IA

Les librairies en Python les plus courantes pour coder de l'IA sont TensorFlow et PyTorch. Avant mon stage, j'ai pu me familiariser avec TensorFlow, notamment dans le cadre de mes activités associatives à Christopher.ai. Cependant, il y a une problématique à anticiper dans le cadre du codage de l'IA :

À terme, Binko va récolter de nouvelles photos de déchets prises par les utilisateurs ainsi l'IA utilisée devra prendre en compte les nouvelles photos pour affiner ses résultats. D'autre part, la première version de l'IA ne classifie pas uniquement 10 classes, car dès que mon IA sera au point avec les 10 classes dans le cadre de mon stage, il sera possible de l'adapter facilement pour de plus nombreuses classes. Toutes ces procédures de "mise à jour" demande de réentraîner l'IA sur la nouvelle base de données mise à jour afin de prendre en compte les nouvelles images et/ou les nouvelles classes. Or, un entraînement d'IA demande un temps de calcul énorme selon la taille de la BDD et le matériel informatique utilisé. Dans le cadre de mon stage, j'ai pu avoir accès à des GPU en ligne (voir le paragraphe de prise en main d'AWS), qui permettent d'effectuer des calculs accélérés, appropriés à l'entraînement d'IA. Cependant, malgré la mise à disposition de ces processeurs graphiques, l'entraînement de l'IA pour 10 classes prenait environ 5 heures, ce qui n'est pas négligeable.

De plus, il faut penser sur le long terme, lorsqu'il y aura plus de 10 classes, sachant que j'avais recensé auparavant environ 1500 classes de déchets sans code-barres classifiables idéalement par l'IA. Ainsi, l'élaboration de mon code doit prendre en compte l'évolution de la BDD. Pour parvenir à pallier ce problème, j'ai proposé à mon tuteur d'envisager la technique du fine-tuning, qui est une technique utilisée en Deep Learning permettant de prendre en compte les nouvelles données dans la BDD afin de réaliser un entraînement nouveau uniquement sur les nouvelles données afin de réduire considérablement le temps d'entraînement par rapport à l'entraînement sur toute la BDD comprenant les nouvelles photos. Cependant, cette méthode est implémentable en PyTorch et non pas en TensorFlow. D'autre part, PyTorch est également une librairie pratique permettant à Binko plus tard de pouvoir plus personnaliser l'entraînement de son IA contrairement à TensorFlow qui est plus adapté pour entraîner une IA sur un modèle précis choisi.

Ainsi pour allier efficacité et utilité, j'ai décidé de coder dans un premier temps une IA en TensorFlow utilisant un modèle de CNN appelé "MobileNet". Puis parallèlement, j'ai appris en autodidacte à coder une IA sous la librairie PyTorch, afin de coder après une IA en PyTorch utilisant le modèle de CNN appelé "ResNet". En effet, cette stratégie me permet alors non seulement de comparer deux modèles de CNN dédiés à la problématique de

classification, mais aussi de fournir à Binko différentes versions d'IA à utiliser en fonction de leurs performances après optimisation (voir le paragraphe optimisation).

En effet, ResNet, est censé nous permettre d'optimiser plus efficacement la classification d'images qu'un MobileNet, car le ResNet est connu pour faire "progresser" plus rapidement l'IA. Néanmoins le MobiletNet reste tout aussi efficace à sa propre manière. C'est pour cela que j'ai implémenté le ResNet sur le code en PyTorch qui servira sûrement sur le long terme. Le code TensorFlow me sert donc d'alternative de secours au cas où mon apprentissage autonome de PyTorch se passe mal et que je n'arrive pas à implémenter de code sous PyTorch.

D'autres alternatives de codage et de solutions possibles étaient possibles pour répondre à la problématique que m'a adressée mon tuteur. Cependant elles n'étaient pas réalisables dans le temps imparti de mon stage ou bien en raison d'un manque de matériel informatique nécessaire. J'ai pu détailler ces alternatives possibles et des détails sur leur implémentation dans un document écrit que j'ai laissé à mon tuteur à la fin de mon stage.

L'entraînement d'une IA

Pour réaliser le code d'une IA, il faut donc principalement écrire le code permettant de l'entraîner, dans un premier temps.

Structure d'un entraînement d'IA

Quelque soit la librairie Python utilisée, PyTorch ou TensorFlow, l'entraînement d'une IA se code selon une même structure. L'IA s'entraîne à classifier sur une partie de la BDD, qui correspond au dataset d'entraînement, puis s'évalue sur une partie de la BDD qu'elle n'aura jamais vu dans son entraînement, ce qui correspond au dataset de validation. En effet, par analogie on pourrait prendre l'exemple d'un élève centralien à qui on voudrait lui enseigner une nouvelle matière. Pour parvenir à la maîtrise de cette matière, il faut qu'il aille en cours et en TD pour s'entraîner, comme l'IA. Puis pour vérifier le bon apprentissage de l'élève, celui-ci passe des contrôles continus pour évaluer ses compétences de manière objective. En effet, ces contrôles sont censés différencier l'élève qui a appris par cœur bêtement sans comprendre de l'élève qui a compris la logique et les méthodes de la matière tout au long de l'entraînement global. C'est pour cela que dans l'entraînement de l'IA, il y a une partie d'évaluation, qu'on appelle communément la "validation" afin de vérifier si l'IA n'a pas appris bêtement par coeur les labels de chaque images pour les classifier, en lui demandant de classifier les images de la validation qu'il n'aura donc jamais vu lors de sa phase d'entraînement. Ainsi, l'entraînement d'une IA entière est composé d'un entraînement à proprement parler et de validations.

À partir d'une BDD, l'IA va apprendre à classifier en saisissant les "points communs" et leur "différences" entre les images de déchets, associés à leur labels respectifs. Ainsi, dans le code d'une IA, il y a un fichier correspondant au générateur pour préparer les images. Le générateur permet de charger les images de la BDD, pour chaque dataset, quelque soit la manière dont elles sont stockées et rangées, afin de les réunir en paquet d'images, qu'on appelle "batch". En effet, pour que l'IA puisse saisir les "points communs" lui permettant d'apprendre à classifier les photos de déchets, on lui montre les images d'un dataset par paquets, et l'IA va "étudier" toutes les images du paquet en simultanée, paquet par paquet. Puis, on va réitérer le processus plusieurs fois en mélangeant les images des paquets. Cette itération de la visualisation des paquets d'image d'un dataset correspond à une "epoch". L'epoch correspond à un seul apprentissage sur toutes les données du dataset, donc lors d'une epoch, l'IA va "apprendre" par paquets d'images, sur toutes les images du dataset, c'est-à-dire que tous les batchs du dataset passent une fois dans le modèle du réseau. Puis ces paquets d'images seront mélangés pour la prochaine epoch.

Après avoir généré des batchs d'images pour chaque phases de l'entraînement global de l'IA, c'est-à-dire de l'entraînement et de la validation, il faut coder le modèle de réseau de CNN adopté. En effet, ces images vont passer par un modèle de CNN : des filtres, des couches de neurones agencés d'une certaine manière caractéristique au modèle adopté. Ce traitement des images est ce qui est caractéristique des CNN et nécessite des paramètres pour paramétrer le modèle et déterminer la manière dont le modèle va fonctionner vis-à-vis de la problématique traitée et de la BDD utilisée. Il y a donc également un fichier correspondant au modèle de réseau dans le code de l'IA et un fichier recensant les paramètres utilisés.

De plus, il faut pouvoir évaluer les performances de l'IA aussi bien pendant son entraînement global à chaque epoch par exemple qu'à la fin de son entraînement ou lors de son application à une image de déchet nouvelle qu'on aimerait que l'IA classifie. Pour cela, il faut définir une métrique pour évaluer les performances de notre IA. Généralement, il faut implémenter une fonction de coût, permettant de calculer l'écart d'erreur entre la réponse prédite par l'IA et la bonne réponse indiquée par le label. Puis, on peut choisir une métrique pour évaluer les performances de l'IA. La métrique choisie est l'accuracy. Ces critères d'évaluation peuvent donner une indication sur l'avancement de l'apprentissage de l'IA lors de son entraînement global après chaque epoch par exemple. Puis contrairement au code en TensorFlow, le code en PyTorch nécessite en plus de coder la manière dont l'IA va s'entraîner sur le modèle de CNN choisi. Ainsi, tous les éléments explicités dans ce paragraphe sont réunis dans un fichier d'entraînement.

À la fin de son entraînement, on peut éventuellement faire passer un ultime test pour mesurer les performances réelles de l'IA en condition réelle, ce qui par analogie équivaut au partiel pour l'élève. Son test se fera donc logiquement sur les images prévus à cet effet dans un dataset de test. En effet, le test doit se faire sur un dataset à part que le réseau n'a jamais vu, de la même manière qu'on prend un dataset de validation à part du dataset d'entraînement, pour éviter la "triche" de l'IA. Le test est facultatif, donc pour plus de simplicité, je ne l'ai pas implémentée dans mes codes dans le cadre de mon stage. En effet, si la validation nous suffit pour juger des performances au cours de l'évolution de l'IA, le test est optionnel.

Ainsi, pour le code TensorFlow et le code PyTorch, chacun est composé d'un dossier **src** contenant:

- un générateur : **generator.py**
- un modèle : **model.py**
- la structure de l'entraînement mené : **train.py**
- des paramètres auxquels les autres fichiers vont faire appel : **parameters.py**
- un fichier recensant toutes les librairies nécessaires pour les installer d'un seul coup de commande avec : *pip install -r requirements.txt* : **requirements.txt**
- un fichier **traitement.py** qui permet de réunir des codes auxiliaires aux codes destinés à l'IA

Au préalable, la BDD utilisée pour l'entraînement a été stockée en local dans mon ordinateur et elle a été rangée dans des dossiers portant le nom du label des images qui la contiennent.

Conception logique des codes TensorFlow et PyTorch

L'explication qui suit est valable pour le code en TensorFlow et pour le code en PyTorch.

Pour coder le générateur, j'ai défini une fonction qui génère de batchs pour l'entraînement et la validation et qui prennent en paramètre le chemin local menant au dossier contenant la BDD d'entraînement. À partir de cette BDD, la fonction va récolter tous les chemins de chaque image logée dans chaque dossier associé à leur classe, puis elle va séparer les chemins selon une proportion prédéfinie indiquée dans le fichier de paramètres pour constituer le dataset d'entraînement et le dataset de validation, à partir de la BDD initiale. Récolter le chemin des images permet d'en extraire le label de chaque image puisqu'il est indiqué dans l'adresse du chemin d'accès de l'image. Puis pour chaque dataset, le générateur va créer des batchs d'images de taille définie également dans le fichier de

paramètres. Ce batch est remélangé aléatoirement à chaque epoch. Le générateur retourne à la fin des batchs d'images chargées et les labels associés à chaque images chargées.

Le label retourné est sous le format d'un vecteur de probabilités où chaque indice correspond à un label. Ainsi, le générateur qui fournit les "bonnes réponses" va donc désigner les labels par des vecteurs de 0 avec un 1 à l'indice correspondant au bon label.

Puis ce générateur va faire passer ces images préparées dans le modèle choisi qui a été codé selon la structure du MobileNet d'après [7] si le code est en TensorFlow ou ResNet d'après [8] si le code est en PyTorch. Les modèles ont été codés filtres par filtres, et blocs par blocs pour les neurones en utilisant les fonctions des librairies utilisées (TensorFlow, PyTorch) en s'inspirant de la structure expliquée dans les documents [7] et [8]. Cette partie est donc située dans le fichier model.py.

Puis dans dans le fichier d'entraînement train.py, on y indique la manière dont l'entraînement sera mené, en fournissant le générateur et le modèle préalablement codés. Pour TensorFlow, une fonction est déjà destinée à l'entraînement direct de l'IA en lui fournissant les paramètres, ainsi que notre fonction de coût et notre métrique. Cependant en PyTorch, cette fonction n'existe pas dans la librairie, ainsi c'est ce qui la démarque de TensorFlow. En effet, comme expliqué auparavant, sur PyTorch j'ai dû donc indiquer à l'IA explicitement toutes les étapes par lesquelles l'IA doit passer pour effectuer cet entraînement, dans les détails près. C'est pour cela que PyTorch est plus maniable pour des codes utiles à long terme car elles permettent de personnaliser au maximum l'IA créée.

Dans ce même fichier, la fonction de coût est une fonction définie par les librairies respectives qui permet d'évaluer l'écart entre la prédiction de l'IA lors de son entraînement et son label réel. Puis j'ai pu coder une métrique adaptée à la forme de la problématique. En effet, l'IA renvoie une prédiction en sortie de ses réseaux de neurones sous la forme d'un vecteur de probabilités. J'ai donc codé la métrique d'accuracy donnant à chaque epoch le pourcentage d'exactitude, ce qui correspond à la moyenne d'exactitude des évaluations par l'IA de chaque image de chaque batch dans l'epoch concernée.

À l'issue de l'entraînement, on récolte un fichier correspondant à l'IA entraîné selon nos paramètres et sur notre BDD. D'autre part, j'ai intégré dans mon code global un outil de visualisation de métrique et de fonction de coût tout au long de l'entraînement de l'IA. Cet outil s'appelle TensorBoard, et il permet de fournir en plus de l'IA entraînée, des fichiers récoltants les valeurs de l'accuracy et de la fonction de coût à chaque epoch, afin de pouvoir en analyser l'évolution. Ces données me permettent alors de prendre des décisions sur les modifications de hyperparamètres des IA afin de l'optimiser et de le rendre plus performant.

Puis à partir de l'IA entraînée, j'ai codé pour les codes de chaque librairie un fichier **predict.py** permettant de classifier n'importe quelles images utilisant un fichier d'IA entraîné, afin que mon tuteur aie un code prêt pour classifier de nouvelles images, s'il veut tester une IA entraînée.

Prise en main d'AWS

Afin de faire tourner les entraînements des IA que j'ai codés afin d'en estimer les performances puis d'en faire l'optimisation successive par la suite, j'ai dû apprendre à les faire tourner sur un serveur à distance. En effet, mon ordinateur n'est pas équipé de processeurs graphiques permettant d'effectuer des calculs accélérés nécessaires pour les entraînements d'IA. Ainsi, Binko m'a mis à disposition dès août un compte AWS (Amazon Web Services) rechargé en crédits afin de pouvoir utiliser des serveurs en ligne d'AWS disposant de GPU.

Cependant, la plateforme d'AWS n'est pas très intuitive à utiliser. Ainsi j'ai dû me former en autodidacte sur les différentes manières d'utiliser AWS à bon escient dans le cadre de mon stage.

J'ai donc principalement appris à exploiter des GPU, car c'est mon objectif principal, afin de faire tourner mes entraînements. Pour cela, j'ai dû me former à utiliser EC2. Dans ce service d'AWS, j'ai dû créer des instances, dites "P", désignant les GPU, permettant le calcul "accéléré". J'ai réservé à distance un serveur de taille proportionnelle au problème traité. Ici un entraînement pour la classification de déchets me permet de prendre un GPU avec 16 Gio de RAM. Afin d'optimiser le temps de calculs et d'optimiser de manière général mon travail d'optimisation, j'ai ouvert 4 instances pour faire tourner en parallèle des entraînements des codes en TensorFlow et en PyTorch avec différents hyperparamètres, afin d'accélérer le processus d'optimisation.

En effet, le choix du GPU a été réfléchi, car un GPU dit "classique" possède une RAM de 8 Gio, et des programmes nécessitant de travailler sur 6 To de données voire plus ne nécessitent qu'au maximum un GPU avec 24 Gio de RAM. La BDD utilisée faisait moins d'1 Go de taille, donc les calculs nécessaires à l'entraînement sur 1 Go de BDD pouvaient être supportés par le GPU choisi. Cette réflexion sur le choix du GPU permet également de souligner l'impact écologique que peut avoir le numérique. En effet, faire tourner des GPU de tailles largement supérieures à ce qui est nécessaire gaspille beaucoup d'énergie inutilement.

Après avoir appris à utiliser les services d'EC2, j'ai appris quelques notions complémentaires : l'utilisation de *WinSCP* et du protocole SSH. En effet, pour faire tourner les entraînements sur les différents serveurs, il faut au préalable avoir transmis les codes ainsi que la BDD associée pour lancer l'entraînement. Ce transfert de données s'est fait par le biais de *WinSCP* que j'ai appris à maîtriser tout au long de mon stage. Afin de lancer les entraînements sur les serveurs à distance d'AWS depuis mon ordinateur, j'ai dû renforcer mes connaissances sur le protocole SSH, afin d'établir des connexions depuis mon terminal de commandes avec le serveur correspondant. Pour réaliser ces connexions, il faut au préalable sécuriser la connexion en ligne, donc j'ai appris à manipuler des clés d'accès ainsi que des groupes de sécurité permettant de contrôler les différentes adresses IP d'utilisateurs se connectant au serveur concerné.

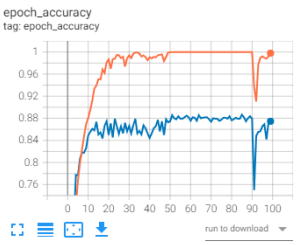
Pour le bon fonctionnement des entraînements, j'ai appris à résoudre des bugs spécifiques aux serveurs en ligne, comme les incompatibilités de versions entre les modules de Python définis par défaut dans le serveur à distance et ceux qui ont été utilisés sur mon ordinateur en local pour coder les IA.

Sur demande de mon tuteur, j'ai pu compiler mes connaissances acquises au cours de mon apprentissage en autodidacte des services d'AWS, notamment de l'utilisation d'EC2, sur un document visuel, qui servira de tutoriels aux futurs stagiaires et aux dirigeants de Binko.

Optimisation des IA : analyse des résultats

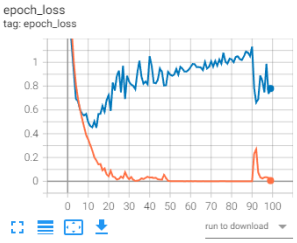
Tout d'abord, l'optimisation d'une IA se fait en menant des expériences successives qui consistent à faire varier les hyperparamètres et refaire tourner des entraînements pour en évaluer les performances à l'aide de TensorBoard. L'optimisation est nécessaire pour éviter principalement un phénomène : l'overfitting. C'est un problème très courant pour la classification et c'est ce problème que j'ai résolu au cours de mon stage après avoir établi les codes des IA.

Concernant l'optimisation des IA, j'ai également écrit un autre document détaillé pour mon tuteur afin d'expliquer en profondeur les nuances et les manières d'optimiser les IA utilisées.



Reconnaître un overfitting :

Il y a overfitting lorsque la validation ne suit plus l'entraînement qui est, on suppose, en apprentissage (donc qui croît en terme d'accuracy par exemple). Ce qui se manifeste par une rupture de la

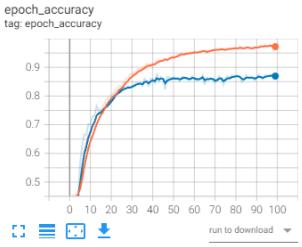


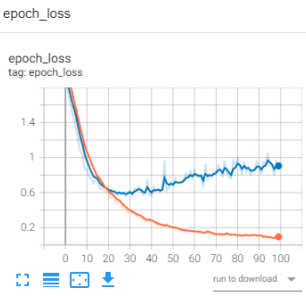
courbe d'accuracy de validation pendant l'entraînement. Cette rupture témoigne du moment où le modèle a décidé d'apprendre par cœur.

Par exemple ici, sur une des nombreuses expériences que j'ai réalisées, les courbes nous indiquent qu'à partir de la 13e epoch environ, la validation ne suit plus en accuracy, et la fonction de coût, appelée loss, de la validation augmente de nouveau significativement. Par conséquent, l'écart entre les bonnes réponses et les prédictions en validation augmente, l'IA n'apprend plus.

(légende : courbe en orange correspond aux valeurs de la phase d'entraînement et la courbe en bleue correspond aux valeurs de la phase de validation)

Comment voir que l'overfitting a été combattu ?





L'overfitting est combattu lorsque la tendance de la validation suit la tendance de l'entraînement et quel que soit l'écart qu'il y a entre l'entraînement et la validation. En effet, cet écart témoigne uniquement d'une tendance à overfitter. Dès lors que la validation ne suit plus l'allure de l'entraînement, elle commence à baisser en accuracy par exemple (ou augmenter en loss) on peut alors dire qu'il y a overfitting.

Qu'en est-il de réduire l'écart ? Bien évidemment pour améliorer le modèle, il faut que l'écart soit le plus petit possible, car plus cet écart est petit, par définition, plus notre validation progresse, et notre réseau s'améliore. Si on se retrouve dans le cas où on veut désormais réduire cet écart, c'est qu'on a officiellement combattu l'overfitting, mais qu'il faut remonter le modèle en termes de performances, en réduisant cet écart justement.

Par exemple, ces courbes issues de l'une de mes expériences avec le dossier montre que l'overfitting est pratiquement presque parti, mais l'écart entre la validation et l'entraînement peut être amélioré.

Quelles sont les méthodes pour combattre l'overfitting ?

Dans un premier temps, il faut modifier les hyperparamètres du modèle de réseau utilisé et du générateur. Concrètement, il faut agir sur la taille des batchs d'images ainsi que

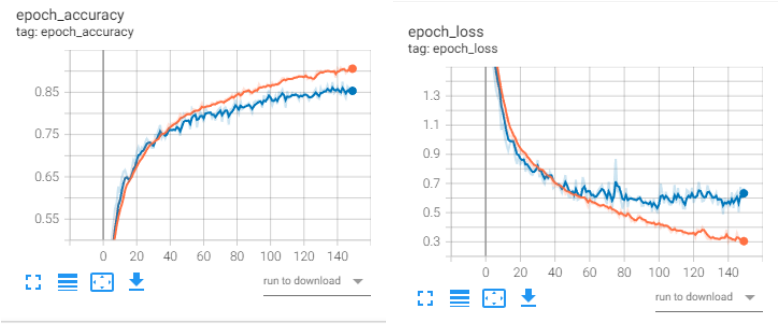
sur le nombre d'epochs. Plus l'IA voit d'images par batchs, plus il sera apte à saisir le plus de points communs par epoch. Plus on a d'epochs, plus l'IA s'entraînera longtemps et pourra récolter de "connaissances" lors de son entraînement.

Au niveau du modèle de réseau de CNN, en fonction du nombre d'images à disposition dans notre BDD, il faut adapter le nombre de neurones par couches de réseau de neurones. En effet, moins il y a de neurones, plus les "connaissances" récoltées par l'IA seront concises mais risquent de ne pas être assez nombreuses pour classifier de manière optimale. Cependant, plus il y a de neurones et plus l'IA récoltera d'informations, mais cela allonge le temps d'exécution pour chaque entraînement lancé et il y a un risque que l'IA apprenne un surplus de "points communs" obsolètes. Il faut donc trouver un juste milieu.

Puis après avoir réglé les hyperparamètres du code d'IA, on peut envisager de rajouter de la data-augmentation et des couches de Dropout pour empêcher l'IA "d'apprendre par cœur" au maximum au cours de son entraînement. En effet, la data-augmentation permet d'aider l'IA ici à reconnaître le fait que les déchets sont invariants par rotations, car quel que soit l'angle sous lequel on prend en photo un déchet, il appartient toujours à la même classe. En l'occurrence, pour ce qui est des déchets, on va appliquer des rotations aléatoires aux images générées, afin d'implanter l'invariant par rotation. Le Dropout permet à l'IA de voir s'il a bien compris la "logique" de la classification afin d'éviter au plus la "triche" lors de la validation.

Après 48 expériences menées en août pour optimiser mes deux codes d'IA, ce qui équivaut à 240 heures d'entraînement, je suis arrivée à la fin de mon stage avec des codes optimisés qui ne présentent plus d'overfitting et qui peuvent atteindre la meilleure accuracy possible.

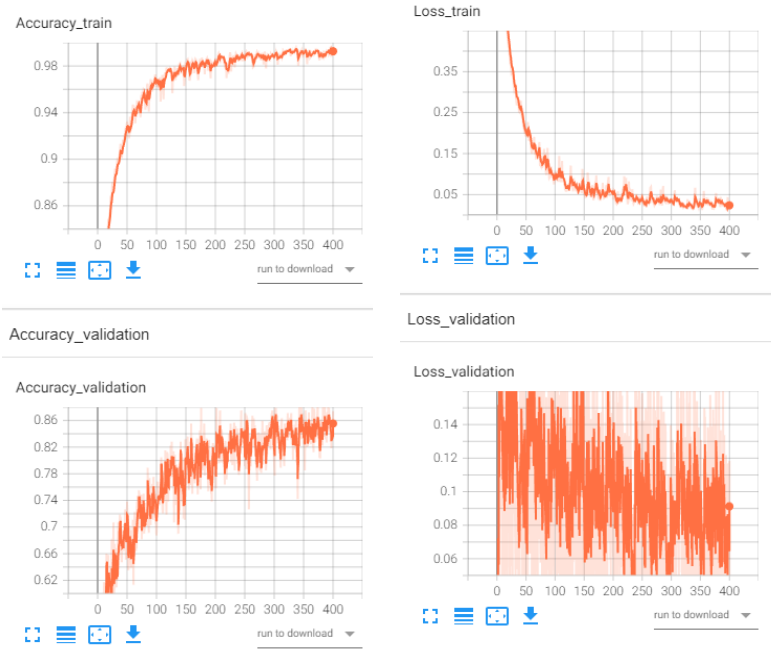
- Pour TensorFlow :



(légende : courbe en orange correspond aux valeurs de la phase d'entraînement et la courbe en bleue correspond aux valeurs de la phase de validation)

L’overfitting a été combattu et l’entraînement a été arrêté à la 150e epoch pour éviter l’apparition de l’overfitting si on continuait l'entraînement de cette IA trop longtemps.

- Pour PyTorch :



L’overfitting n’apparaît toujours pas après un entraînement sur 400 epochs, ce qui équivaut à un entraînement de 6h sur des GPUs d’AWS, ainsi avec plus de temps j’aurai pu lancer une autre expérience avec un nombre plus important d’epochs.

La loss de validation a une allure chahutée qui peut s’expliquer en raison de la manière technique dont a été codée le code en PyTorch. Néanmoins son allure est visible et correspond aux attentes de l’optimisation.

Grâce à mes fichiers predict.py, j’ai pu testé les IA optimisées entraînées sur des images de déchets aléatoires sur internet et le classement est correct. Cependant, elles ont clairement plus de mal à classifier des images de déchets qui s’éloignent du style des images sur lesquelles elles se sont entraînées auparavant. En effet, d’où l’importance d’établir une BDD diversifiée et grande pour avoir des IA les plus généralistes et performantes possibles.

Mes réalisations dans l’entreprise

Pour pouvoir communiquer clairement et quantifier précisément mes résultats tout au long de mon stage, j’ai rédigé un document explicatif très détaillé pour expliquer les principes de bases d’une IA, le fonctionnement et les étapes de mon travail, avec une liste de pistes pouvant être plus approfondies ou envisagées à l’issue de mon travail de deux mois. J’ai également laissé des tutoriels sur AWS et sur l’utilisation de WinSCP, ainsi qu’un document pour expliquer les notions et les méthodes de l’optimisation d’une IA. J’ai également transmis les codes que j’ai réalisés dans le cadre de mon stage sur le GitHub de l’entreprise, afin qu’ils soient accessibles à tous les membres et futurs membres de l’entreprise.

D’autre part, j’ai également pu contribuer à des choix stratégiques commerciaux et de développement pour Binko. J’ai pu découvrir le fonctionnement d’une start-up et la manière dont elle se développe. J’ai pu contribuer de manière créative à la participation de Binko dans ses appels à projets en participant à la création de contenus artistiques de communication. J’ai également assisté mon tuteur dans certaines tâches administratives.

Conclusion

J’ai réussi à proposer une solution optimale et viable pour la problématique qui m’était posée au début de mon stage. J’ai pu fournir non seulement une solution technique mais j’ai également fourni des ressources pour que mon tuteur et d’autres membres de Binko puissent reprendre et continuer le travail que j’ai entamé dans cette partie IA du projet de l’entreprise. Mes solutions ont été approuvées et très appréciées par mon tuteur qui m’a vivement félicitée.

Au niveau personnel, cette expérience de stage m’a donnée l'occasion de découvrir le monde du travail, notamment celui des ingénieurs du numérique, celui dans lequel j’aspire à travailler après mes études. J’ai appris des connaissances pratiques et techniques également grâce aux prises d’initiatives qu'impliquent cette mission à la problématique large et globale que m’a proposée mon tuteur de stage. J’ai également découvert l’envers du décors dans la création et la gestion d’une entreprise et notamment d’une start-up en accompagnant mon tuteur qui est président de Binko.

Bibliographie

[1] CITÉO, Rapport annuel 2020/2021, “*Plus on sait, mieux on fait” saison 2*, juin 2021

[2] ADEME, *Chiffres-clés Déchets, l’essentiel 2020*, septembre 2020.

[3] ADEME, *Que faire de ses déchets*, Septembre 2019

[4] OCDE, Études de l'OCDE sur la politique de l'environnement et le comportement des ménages, *Politique de l'environnement et comportement des ménages,* 2011

[5] CITÉO, Application mobile *Guide du tri*, 2021

[6] Kaggle, consultées entre le 06/07/2022 et le 12/07/2022, https://www.kaggle.com/datasets/arkadiyhacks/drinking-waste-classification, https://www.kaggle.com/datasets/mostafaabla/garbage-classification, https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification

[7] Analytics Vidhya sur Medium.com, modèle de Mobilenet consulté pendant le mois d’août 2022, https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470

[8] Analytics Vidhya, modèle de ResNet, consulté pendant le mois d’août 2022, https://www.analyticsvidhya.com/blog/2021/08/how-to-code-your-resnet-from-scratch-in-tensorflow/