

Rapport_PCD_Aissa_Dinari

par Sameh Malek

Date de l'envoi: 09-mai-2024 08:08PM (UTC+0100)

Nº du document envoyé: 2375346099

Nom du fichier: euse_complet_1_Sign_sans_pdg_tables_figures_et_bibliographie.pdf (11.51M)

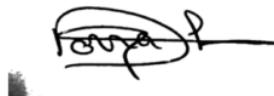
Nombre de mots: 17202

Nombre de caractères: 97902

Appréciations et signature de l'encadrant

Très bon travail

Donna DHOONIB

A handwritten signature in black ink, appearing to read "Donna DHOONIB". The signature is fluid and cursive, with "Donna" and "DHOONIB" being the most distinct parts.

Remerciements

Nous tenons à exprimer notre respect et notre gratitude envers toutes les personnes qui ont joué un rôle essentiel dans la réussite de ce projet. À travers ses différentes étapes, nous avons été assistés par plusieurs personnes dont lesquelles nous n'aurions pas été aussi efficaces. Pour cette raison, nous tenons à remercier toutes les personnes qui ont contribué, directement ou indirectement, à ce travail.

Tout d'abord, nous souhaitons exprimer notre sincère gratitude à notre superviseure, Mme Dorra Dhouib, pour son soutien et ses conseils qui nous ont permis de donner le meilleur de nous-mêmes. Nous tenons également à remercier M. Bacha Othmen pour nous avoir aidés à mieux comprendre le travail demandé et nous avoir donné l'opportunité de mener à bien ce projet.

Nous tenons également à exprimer notre profond respect envers les membres du jury pour nous avoir honorés en examinant et en évaluant notre travail.

Enfin, nous sommes reconnaissants à tous les enseignants de l'École Nationale d'Informatique pour leur aide continue et leur précieuse formation au cours des années d'études.

Glossaire des acronymes

IA *Intelligence Artificielle*

CNN *Convolutional Neural Network*

CV *Computer Vision*

DL *Deep Learning*

IDE *Integrated Development Environment*

ML *Machine Learning*

MO *moelle osseuse*

OD *Object Detection*

Pandas *Python Data Analysis Library*

Pil *Python Imaging Library*

ROI *Region Of Interest*

SCC *Single Cell Classifier*

SI *Segmentation d'Images*

YOLO *You Only Look Once*

Introduction Générale

Les avancées technologiques récentes dans les domaines de l'imagerie et de l'intelligence artificielle offrent de nouvelles possibilités pour automatiser le processus d'analyse des cellules sanguines. Le besoin d'une solution informatique efficace pour identifier, classifier et compter les différents types de cellules présentes dans les images microscopiques obtenues à partir des échantillons de la moelle osseuse est évident. L'objectif principal de ce projet est donc de créer un système qui assiste les biologistes et les experts en hématologie dans leur analyse morphologique des cellules de la moelle osseuse de manière plus efficace et précise.

Le système de classification automatisée des cellules sanguines à partir de frottis de la moelle osseuse proposé dans ce projet se veut être une solution innovante pour les laboratoires d'analyse médicale et les centres hospitaliers spécialisés dans le diagnostic et le traitement des maladies du sang. En utilisant des algorithmes de traitement d'image et des techniques de machine learning, ce système sera en mesure de réaliser une classification rapide et précise des différents types de cellules présentes dans les échantillons de la moelle osseuse. En automatisant une partie du processus d'analyse, il permettra de gagner du temps et d'améliorer la précision des diagnostics, offrant ainsi un outil précieux aux professionnels de la santé.

Le rapport s'articulera autour de quatre chapitres principaux. Tout d'abord, dans le chapitre de l'étude préliminaire et de l'existant, nous examinerons la perspective courante de l'analyse automatisée des cellules sanguines ainsi que les technologies existantes dans le domaine et les défis liés à la segmentation des cellules de la moelle osseuse. Ensuite, le chapitre de l'étude des besoins détaillera les exigences fonctionnelles et non fonctionnelles du système, en mettant l'accent sur les besoins des utilisateurs et les contraintes techniques. Par la suite, le chapitre des expérimentations et technologies explorera les différentes méthodes et technologies utilisées dans le développement du système, en mettant en évidence les expérimentations réalisées dans le cadre de notre projet. Enfin, le chapitre de validation et des tests présentera la mise en œuvre du système, les tests réalisés pour garantir son bon

fonctionnement, ainsi que les résultats obtenus. Ces quatre chapitres seront suivis d'une conclusion synthétisant les principales conclusions du rapport et proposant des pistes de recherche futures.

Chapitre 1

Cadre général du projet

Cette première partie mettra en lumière quelques mots-clés liés au domaine de travail abordé, les applications similaires existantes, leurs limites ainsi que la solution proposée.

1.1 Description du projet

Dans cette section, nous aborderons la problématique ainsi que les objectifs de notre projet.

1.1.1 Problématique

L'examen de la moelle osseuse est un élément essentiel du diagnostic des maladies hématologiques, notamment la leucémie. Cependant, l'analyse manuelle des frottis de moelle osseuse est chronophage et sujette à des variations inter-observateurs, ce qui peut entraîner des diagnostics imprécis ou retardés. Pour améliorer cette situation, il est nécessaire de développer des méthodes automatisées capables de détecter et de classifier les différentes classes de cellules sanguines présentes dans les échantillons de moelle osseuse.

1.1.2 Objectif du projet

L'objectif principal de notre projet de conception et de développement est de développer un modèle basé sur l'intelligence artificielle pour la détection et la classification des cellules sanguines dans les frottis de moelle osseuse. Ce modèle devrait être capable de reconnaître et de quantifier différentes classes de cellules, telles que les blastes, les érythroblastes, les lymphocytes, et d'autres, afin de fournir des informations précises sur la composition cellulaire des échantillons de moelle osseuse. À terme, cette approche permettra d'améliorer la précision du diagnostic de la leucémie et d'autres troubles hématologiques, ainsi que la prise en charge des patients.

1.2 Étude Théorique

1.2.1 Analyse Hématologique et Anatomie des Frottis de Moelle Osseuse

L'examen de la moelle osseuse (MO) joue un rôle important dans le diagnostic et la prise en charge de divers troubles hématologiques, notamment la leucémie. La leucémie est un type de cancer caractérisé par la prolifération anormale de cellules sanguines immatures dans la moelle osseuse, entraînant des perturbations dans la production normale de cellules sanguines [1]. L'identification précise et la classification des cellules sanguines dans les frottis de moelle osseuse sont essentielles pour le diagnostic rapide et précis de la leucémie et d'autres affections connexes.

Dans un frottis de moelle osseuse, différentes classes ou phases de cellules peuvent être observées [2], chacune jouant un rôle crucial dans l'évaluation des conditions hématologiques. Parmi ces classes, on trouve :

TABLE 1.1 – Quelques classes de cellules de moelle osseuse

| Abréviation | Phase/Classe |
|-------------|---|
| ABE | Abnormal eosinophil : Éosinophile anormal |
| ART | Artifact : Artefact |
| BAS | Basophil : Basophile |
| BLA | Blast : Blast |
| EBO | Erythroblast : Érythroblaste |
| EOS | Eosinophil : Éosinophile |
| FGC | Faggott cell : Cellule en fuseau (Faggott) |
| HAC | Hairy cell : Cellule pileuse (Hairy cell) |
| KSC | Smudge cell : Cellule écrasée (Smudge) |
| LYI | Immature lymphocyte : Lymphocyte immature |
| LYT | Lymphocyte : Lymphocyte |
| MMZ | Metamyelocyte : Métamyélocyte |
| MON | Monocyte : Monocyte |
| MYB | Myelocyte : Myélocyte |
| NGB | Band neutrophil : Neutrophile en bande |
| NGS | Segmented neutrophil : Neutrophile segmenté |
| NIF | Not identifiable : Non identifiable |
| OTH | Other cell : Autre cellule |
| PEB | Proerythroblast : Proérythroblaste |
| PLM | Plasma cell : Cellule plasmique |
| PMO | Promyelocyte : Promyélocyte |

En résumé, la compréhension de la structure et de la composition cellulaire de la moelle osseuse est essentielle, même dans le cadre de projets axés sur l'intelligence artificielle. Elle nous donne un aperçu précieux de nos objectifs et de notre portée, orientant nos efforts vers l'intégration de solutions de pointe pour la détection et la classification automatisées. Cette compréhension approfondie enrichit notre approche et souligne l'importance de combler le fossé entre les connaissances biologiques et les avancées en intelligence artificielle et en apprentissage automatique afin de faire progresser les diagnostics hématologiques.

1.2.2 Concepts Clés en IA et Vision par Ordinateur

1.2.2.1 Intelligence artificielle

L'intelligence artificielle (IA) représente le **domaine de l'informatique qui se consacre à** doter les machines de capacités intellectuelles similaires à celles des êtres humains. Dans le contexte de notre projet, l'IA est au cœur de notre démarche pour permettre aux systèmes informatiques de comprendre et d'analyser les données visuelles, telles que les images médicales. Cette capacité à interpréter visuellement leur environnement ouvre la voie à des applications telles que la segmentation d'images et la classification automatique [3], qui sont des éléments essentiels de notre objectif de développer des solutions avancées en vision par ordinateur.

1.2.2.2 La vision par ordinateur

La vision par ordinateur (Computer Vision - CV) représente l'ensemble des techniques permettant l'analyse automatique d'images, notamment la segmentation, la classification et la reconnaissance d'objets, ainsi que la détection de motifs et de structures complexes [4]. Plus précisément, dans le cadre de notre projet, la vision par ordinateur joue un rôle crucial dans l'interprétation et l'analyse des images médicales de cellules sanguines [5]. En combinant des algorithmes sophistiqués avec des techniques d'apprentissage automatique, nous visons à développer des outils capables d'identifier et de classifier avec précision différents types de cellules, facilitant ainsi le diagnostic et la recherche médicale dans le domaine de l'hématologie, tout en utilisant les techniques de CV [6] à savoir la détection d'objet, la segmentation d'images, le seuillage, la classification d'images et la segmentation d'instances.

1.2.2.2.1 Détection d'objet

La détection d'objet (Object Detection - OD) constitue un pilier fondamental de notre projet en vision par ordinateur. Son objectif essentiel est d'opérer une localisation précise et une identification des objets présents au sein des images ou des vidéos. Les algorithmes de détection d'objets font généralement appel à des méthodes d'apprentissage machine ou

d'apprentissage profond pour obtenir des résultats significatifs [7]. Ces derniers permettent d'accomplir la tâche complexe de repérage des cellules sanguines dans les images de frottis de MO avec une exactitude remarquable. En mettant l'accent sur la qualité de cette détection et de cette localisation, nous facilitons grandement l'analyse et le diagnostic médical, ouvrant ainsi de nouvelles perspectives dans le domaine de la santé.

1.2.2.2 Segmentation d'images

Au cœur de notre projet réside la segmentation d'images (Image Segmentation - SI), une méthode complexe mais essentielle dans le domaine de la vision par ordinateur. Cette technique peut être catégorisée en plusieurs approches distinctes : certaines visent à identifier les contours présents dans l'image, d'autres à segmenter des régions spécifiques, tandis que certaines se concentrent sur la détection d'objets ou de formes préétablies [8].

La segmentation d'images repose sur un ensemble de principes et d'algorithmes sophistiqués, allant des approches classiques basées sur la détection de contours et la décomposition de l'image.

Dans le contexte médical, la segmentation revêt une importance capitale en permettant l'identification et l'isolation précises des différentes structures anatomiques présentes dans les images médicales [9]. Dans notre projet axé sur l'analyse des frottis de MO, la segmentation joue un rôle fondamental dans la délimitation précise des cellules sanguines, ainsi que dans la caractérisation de leurs contours et de leurs propriétés morphologiques.

1.2.2.3 Seuillage

Dans le vaste domaine de la vision par ordinateur, le seuillage (ou thresholding en anglais) émerge comme une technique fondamentale, offrant une approche simple mais puissante pour la segmentation et l'analyse d'images. Cette technique repose sur le principe de la binarisation des images, où chaque pixel est classé comme étant soit objet, soit arrière-plan, en fonction d'un seuil prédéfini. [10]

Conceptuellement, le seuillage peut être comparé à un processus de prise de décision binaire, où chaque pixel est évalué en fonction de sa luminosité ou de sa couleur, puis attribué à l'une des deux classes en fonction de sa similarité avec le seuil spécifié [11]. En pratique, plusieurs méthodes de seuillage sont disponibles, allant des approches simples telles que le seuillage global basé sur une valeur fixe, aux méthodes plus avancées telles que le seuillage adaptatif prenant en compte la variabilité locale de l'intensité des pixels. [12]

Dans le cadre de notre projet, le seuillage revêt une importance particulière en tant que

prétraitement essentiel pour la segmentation d'images médicales, telles que les images de frottis de moelle osseuse. En définissant judicieusement les seuils appropriés, nous sommes en mesure de séparer efficacement les cellules sanguines du fond de l'image compliqué [13], facilitant ainsi leur identification et leur analyse ultérieure.

Malgré sa simplicité conceptuelle, le seuillage présente néanmoins des défis pratiques, notamment en ce qui concerne le choix du seuil optimal et sa sensibilité aux variations d'éclairage et de contraste. Cependant, avec une compréhension approfondie de ses principes et de ses applications, le seuillage demeure un outil inestimable dans notre boîte à outils d'analyse d'images, offrant une base solide pour des méthodes de segmentation plus avancées et complètes.

1.2.2.2.4 Classification d'images

La classification d'images est une technique fondamentale de vision par ordinateur axée sur la catégorisation des images en classes ou catégories prédéfinies en fonction de leur contenu visuel. Elle joue un rôle essentiel dans diverses applications, de la diagnostique médicale aux systèmes de conduite autonome [14], en permettant aux machines de discerner et d'interpréter les détails complexes présents dans les données visuelles.

Au cœur de la classification d'images se trouvent les algorithmes d'apprentissage automatique, notamment les modèles d'apprentissage profond, qui permettent de reconnaître les motifs et les caractéristiques au sein des images [15]. Ces modèles sont entraînés sur de vastes ensembles de données surtout de dans le domaine médical [16], où ils apprennent à extraire des caractéristiques discriminantes qui distinguent une classe d'une autre. Grâce à ce processus d'apprentissage, ils deviennent capables d'identifier les motifs, les formes, les textures et les structures caractéristiques de classes ou d'objets spécifiques.

De plus, la classification d'images va au-delà de la simple étiquetage [17] ; elle facilite les processus de prise de décision en fournissant des informations précieuses sur la composition sous-jacente et la nature des données visuelles. Cette capacité offre un potentiel immense dans les applications d'imagerie médicale [18], où la classification précise et opportune des anomalies peut avoir un impact significatif sur la précision diagnostique et les résultats des patients.

En résumé, la classification d'images constitue un pilier de la vision par ordinateur, permettant aux machines de discerner et d'interpréter l'information visuelle avec une précision et une efficacité remarquables. Son intégration dans notre projet souligne son importance dans l'avancement du domaine de l'analyse d'images médicales [19], ouvrant la voie à des outils diagnostiques plus efficaces et à des solutions de soins de santé personnalisées.

1.2.2.5 Segmentation des instances

La segmentation des instances (ou instance segmentation en anglais) est une méthode avancée en vision par ordinateur qui vise à identifier et à délimiter avec précision chaque occurrence individuelle d'objet dans une image [20]. Contrairement à la segmentation sémantique, qui classe chaque pixel de l'image en catégories prédéfinies, la segmentation des instances permet de différencier plusieurs occurrences d'objets de la même classe et de les isoler les unes des autres. [21]

Cette approche repose sur l'utilisation de modèles d'apprentissage profond, tels que les réseaux de neurones convolutifs (CNN) [22], qui apprennent à détecter et à segmenter chaque instance d'objet dans une image. En utilisant des techniques comme le masquage et la localisation précise, la segmentation des instances offre des résultats détaillés et précis, essentiels pour de nombreuses applications en CV. [23]

Dans le cadre de notre projet sur l'analyse des frottis de MO, la segmentation des instances revêt une importance particulière. En permettant une identification précise et individuelle des cellules sanguines et de leurs diverses occurrences dans les images, cette méthode ouvre la voie à des analyses approfondies et à des diagnostics plus précis. Elle permet notamment de distinguer chaque cellule individuelle et de quantifier ses caractéristiques morphologiques [24], ce qui contribue à une meilleure compréhension des échantillons analysés.

1.2.2.3 Apprentissage automatique

L'apprentissage automatique (Machine Learning - ML) est une discipline fondamentale en informatique qui vise à doter les machines de la capacité à apprendre à partir de données et à améliorer leur performance avec l'expérience, de manière similaire au processus d'apprentissage humain. Concrètement, cela implique la construction et l'ajustement de modèles mathématiques et statistiques pour effectuer des tâches spécifiques, telles que la classification, la prédiction ou la reconnaissance de motifs, à partir de données [25].

Parmi les approches les plus répandues en ML, on trouve l'arbre de décision, la régression linéaire, la forêt aléatoire et les réseaux neuronaux artificiels [26]. Ces méthodes visent à extraire des informations significatives à partir des données d'entraînement, afin de construire des modèles capables de généraliser et de produire des prédictions précises sur de nouvelles données [27]. Cependant, les données non structurées sont plus courantes, et c'est dans ce contexte que l'apprentissage profond se révèle plus efficace [28]. Il utilise des réseaux neuronaux artificiels comportant un nombre beaucoup plus élevé de couches que ceux de l'apprentissage automatique, ce qui permet une reconnaissance de caractéristiques

plus complexe.

1.2.2.4 Apprentissage profond

L'apprentissage profond (Deep Learning - DL) est une branche de l'apprentissage automatique [29] qui se concentre sur le développement d'algorithmes appelés réseaux neuronaux artificiels [30], inspirés par les connexions neuronales observées dans le cerveau humain.

1.2.2.4.1 Réseau de neurones

Les réseaux de neurones (ou réseaux neuronaux - Neural Network en anglais) sont des réseaux ou des circuits généralement organisés en couches composées de noeuds interconnectés. Techniquement parlant, les réseaux neuronaux sont un ensemble d'algorithmes conçus pour reconnaître des motifs. Un motif fait référence à la classe ou à l'étiquette d'un signal d'entrée [31].

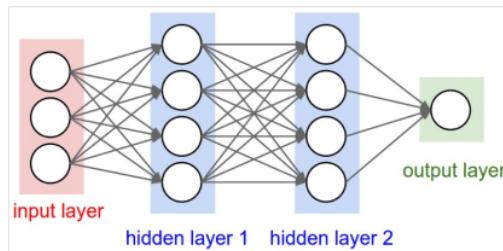


FIGURE 1.1 – Réseau de neurone artificiel [32]

Les couches cachées, où un système de connexions pondérées effectue le traitement réel [33], sont essentielles. Ces couches cachées sont reliées à une couche de sortie. Les motifs d'entrée sont introduits dans le réseau par le biais de la couche d'entrée, qui est connectée à une ou plusieurs couches cachées, où le résultat est généré comme illustré dans la figure 1.1 ci-dessus.

Il existe différents algorithmes d'apprentissage profond, tels que les réseaux neuronaux récurrents (RNN), les réseaux de croyances profondes (DBN) [34], les machines de Boltzmann restreintes (RBM) [35] et les réseaux neuronaux convolutionnels (CNN) [36], qui sont les plus utilisés dans l'analyse d'images visuelles.

1.2.2.4.2 Réseau de neurones convolutif

Le réseau de neurones convolutif (Convolutional Neural Network - CNN) est un algorithme d'apprentissage profond qui prend une image en entrée, attribue de l'importance à différents objets/aspects appelés motifs, et devient capable de les différencier les uns des autres [37]. Les CNN sont privilégiés par rapport à d'autres architectures de réseaux neuronaux pour les raisons suivantes :

- Les réseaux neuronaux convolutionnels nécessitent moins de prétraitement par rapport à d'autres architectures de réseaux neuronaux.
- Le réseau peut toujours être re-entraîné pour améliorer son apprentissage et sa reconnaissance des caractéristiques.
- Les CNN sont efficaces pour lire et capturer des données temporelles ainsi que spatiales à partir d'une image. [38]

L'architecture des CNN se compose d'une couche d'entrée et d'une couche de sortie séparées par trois types de couches : la couche de convolution, la couche de regroupement (ou pooling), et la couche entièrement connectée, comme le montre la figure 1.2

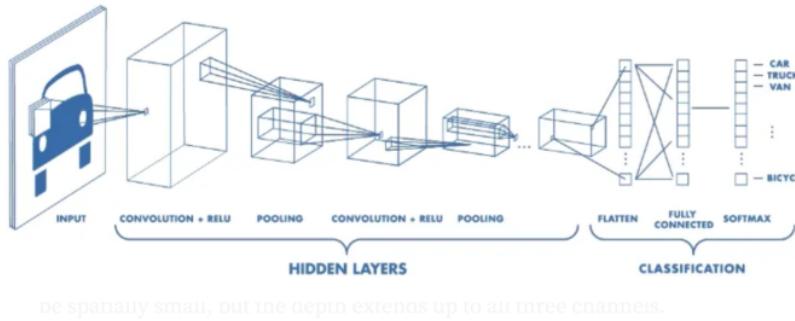


FIGURE 1.2 – Architecture CNN [39]

- Couche de convolution

La couche de convolution constitue le bloc de construction essentiel du CNN. Elle supporte la majeure partie de la charge computationnelle du réseau.

Cette couche réalise un produit scalaire entre deux matrices : l'une contient les paramètres apprenables, également appelés noyaux, et l'autre représente la partie restreinte du champ réceptif. Le noyau est spatialement plus petit que l'image mais plus profond. Ainsi, s'il s'agit d'une image composée de trois canaux (RGB), la hauteur et la largeur du noyau seront spatialement réduites, mais sa profondeur couvrira les trois canaux [40]. La figure

1.3 représente un exemple d'une couche de convolution.

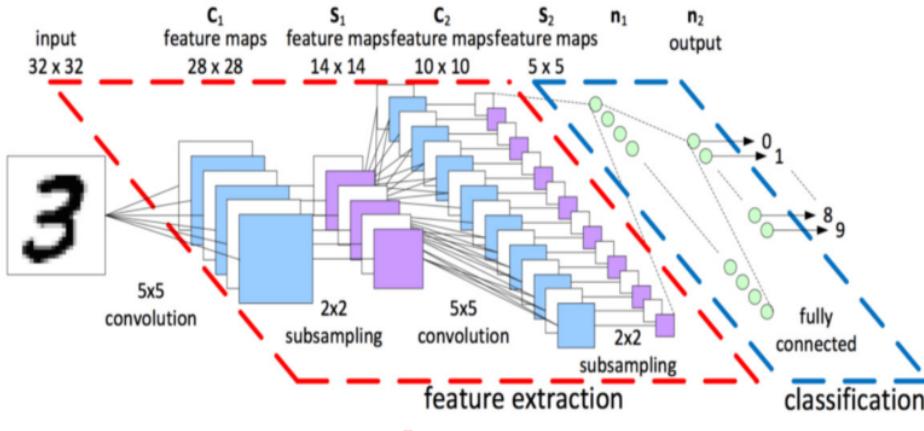


FIGURE 1.3 – Couche de convolution [41]

- La couche de pooling

La couche de pooling est responsable de la réduction de la taille d'une grande image pour résumer la présence moyenne d'une caractéristique.

L'idée est de faire glisser une petite fenêtre pas à pas sur toutes les parties de l'image et de renvoyer la valeur maximale ou moyenne de la fenêtre à chaque étape, comme illustré dans la figure 1.4.

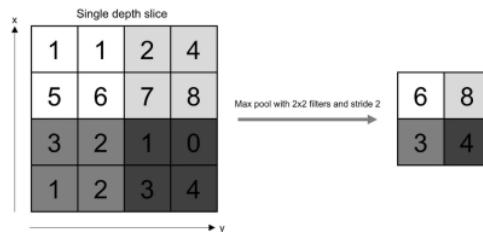


FIGURE 1.4 – Opération de regroupement [42]

- La couche entièrement connectée (ou couche dense)

Le processus CNN débute par la convolution et le "pooling", qui divisent l'image en caractéristiques et les analysent individuellement. Le résultat de cette opération est ensuite transmis à une structure de réseau neuronal entièrement connectée comme le montre la

Figure 1.5. Elle reçoit en entrée la sortie de la dernière couche de convolution, qui est aplatie avant d'être transmise en entrée afin de prendre la décision finale en matière de classification.

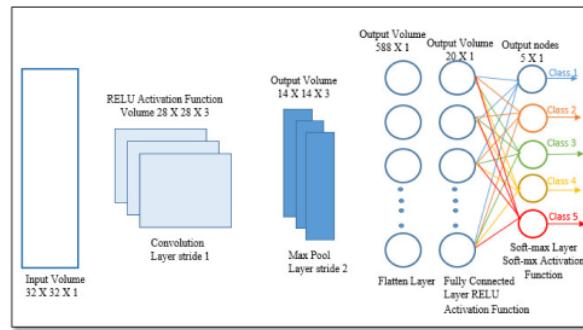


FIGURE 1.5 – Couche dense (entièlement connectée) [43]

- **Défaut des CNNs concernant la localisation d'objet et la segmentation :**

La détection précise d'objets dans des images est une tâche fondamentale en vision par ordinateur, mais elle est souvent entravée par le problème des boîtes englobantes (bounding boxes) [44] dans les CNN. Ce problème se manifeste par la difficulté à prédire avec précision les coordonnées des boîtes englobantes autour des objets d'intérêt, en raison de variations dans leur apparence, leur échelle, leur orientation et leur occlusion.

1.2.2.5 Les concepts et l'architecture de VGG-16

L'algorithme VGG-16 représente une architecture de réseau neuronal convolutif (CNN) profonde, célèbre pour sa structure en couches profondes et homogènes. Conçu par l'équipe de recherche Visual Geometry Group (VGG) [45] à l'Université d'Oxford, ce modèle se distingue par son architecture simple et facile à comprendre.

L'architecture VGG-16 se compose de seize couches de convolution, suivies de trois couches entièrement connectées, et est caractérisée par l'utilisation répétée de petites fenêtres de filtrage de taille 3x3 dans les couches de convolution, associées à des couches de sous-échantillonnage (pooling) de taille 2x2 [46]. Cette structure uniforme permet au modèle de capturer des motifs complexes et des hiérarchies de caractéristiques à différentes échelles dans l'image. Les couches convolutives préservent la résolution spatiale, tandis que les couches de sous-échantillonnage réduisent progressivement la dimension spatiale tout en augmentant la profondeur des caractéristiques extraites.

L'approche VGG-16 utilise des filtres de petite taille répétés dans chaque couche de convolution, ce qui permet au modèle d'apprendre des représentations de caractéristiques

plus discriminantes et invariantes aux translations [47]. Bien que cette approche puisse entraîner un sur-ajustement (overfitting) lorsque de nombreuses couches sont empilées, l'utilisation de couches de sous-échantillonnage contribue à la régularisation du modèle en réduisant la dimensionnalité des données [48]. En fin de compte, cette architecture, comme est montré dans les figures 1.6 et la figure 1.7 robuste et ses représentations de caractéristiques riches en informations ont permis à VGG-16 de devenir une référence dans le domaine de la vision par ordinateur, notamment dans des tâches telles que la classification d'images et la détection d'objets.

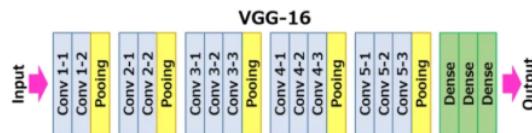


FIGURE 1.6 – Architecture de VGG-16 [49]

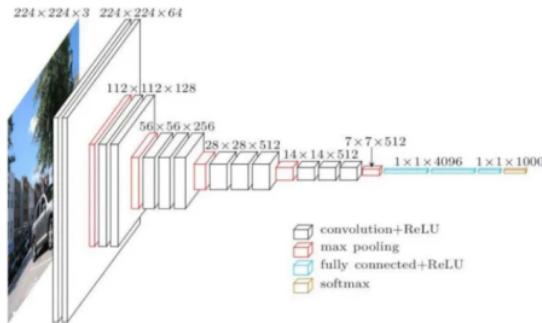


FIGURE 1.7 – Exemple de modèle VGG-16 en 16 couches [50]

1.2.2.6 La détection d'objet et la segmentation avec YOLO

Dans cette étude, nous nous penchons sur l'algorithme You Only Look Once (YOLO) en tant que solution potentielle à au défi de segmentation mentionné dans la partie précédente. YOLO présente une approche innovante qui traite la segmentation des cellules de moelle osseuse comme un problème de régression directe. Contrairement aux méthodes traditionnelles qui utilisent des fenêtres glissantes ou des régions d'intérêt (ROIs), YOLO divise l'image en une grille de cellules et prédit les boîtes englobantes ainsi que les probabilités de classe pour chaque cellule.

YOLO se distingue par sa capacité à prédire les boîtes englobantes et les classes d'objets en une seule passe de réseau, ce qui le rend particulièrement adapté à une segmentation rapide et efficace des cellules de la MO en temps réel. Cette méthode tire également parti de l'exploitation de multiples échelles de prédiction au niveau du réseau, permettant ainsi de prendre en compte les différentes échelles et formes des cellules, renforçant ainsi sa capacité à traiter des données complexes.

Les auteurs de YOLO [51] ont reformulé le problème de la détection d'objets en un problème de régression plutôt qu'un problème de classification. Un réseau neuronal convolutif prédit les boîtes englobantes ainsi que les probabilités de classe pour tous les objets représentés dans une image. Comme cet algorithme identifie les objets et leur positionnement à l'aide de boîtes englobantes en ne regardant l'image qu'une seule fois, ils l'ont nommé You Only Look Once (YOLO). Le CNN fonctionne avec une grande efficacité sur les entrées visuelles pour l'extraction des caractéristiques, car les caractéristiques de bas niveau sont adéquatement propagées des couches convolutionnelles initiales aux couches convolutionnelles ultérieures dans un CNN profond. Ici, le défi réside dans l'identification précise de multiples objets ainsi que de leur position exacte présente dans une seule entrée visuelle. Le partage de paramètres et les multiples filtres sont les deux caractéristiques importantes des CNN, capables de traiter efficacement ce problème de détection d'objets.

Pendant la prédiction des boîtes englobantes, YOLO utilise des "boîtes d'ancre dynamiques" générées par un algorithme de clustering. Ce processus regroupe les boîtes englobantes de référence en clusters et utilise les centroïdes de ces clusters comme boîtes d'ancre. Cela permet aux boîtes d'ancre de mieux s'ajuster à la taille et à la forme des objets détectés [52].

En résumé, dans les sections suivantes de ce rapport, nous explorerons en détail les différents composants et critères nécessaires pour élaborer et déployer avec succès notre modèle de segmentation basé sur YOLO. Nous aborderons notamment les aspects du pré-traitement des données, du choix des architectures de réseau, des techniques d'entraînement et d'évaluation, ainsi que des considérations pratiques pour le déploiement en temps réel. Ces discussions approfondies nous permettront de construire un modèle robuste et efficace pour la segmentation des cellules de moelle osseuse, répondant ainsi aux exigences de notre projet de conception et de développement.

1.2.2.7 Classification sur un modèle MobileNet

Comparé au réseau VGG-16, MobileNet est un réseau léger qui utilise la convolution séparable en profondeur (depthwise separable convolution) pour approfondir le réseau comme

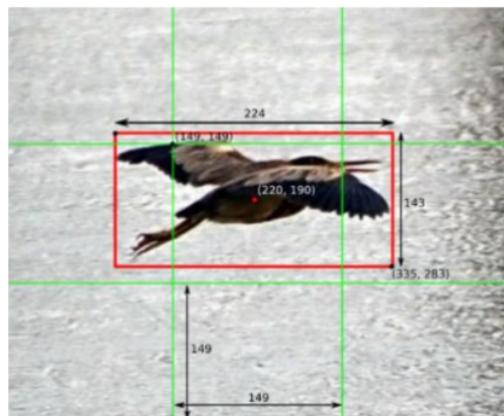


FIGURE 1.8 – Boîte englobante dans YOLO [53]

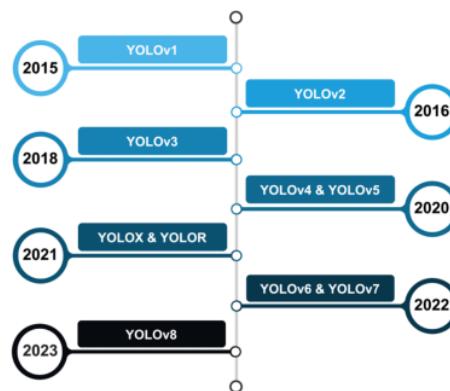


FIGURE 1.9 – Évolution des versions de YOLO [54]

est montré dans les figures 1.10 et 1.11 ci-dessous. Il possède moins de paramètres et une précision de classification plus élevée. [55]

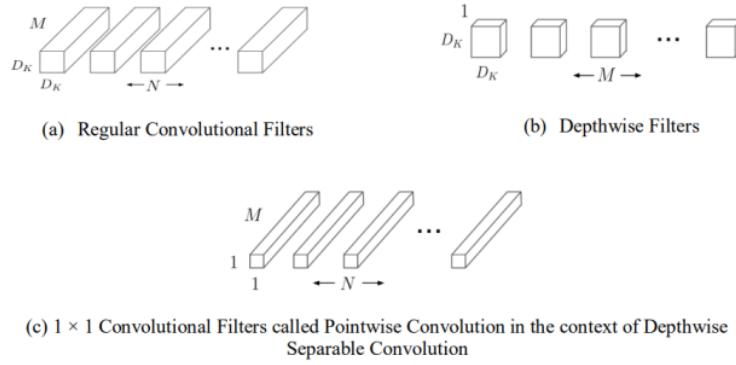


FIGURE 1.10 – Convolution séparable en profondeur [56]

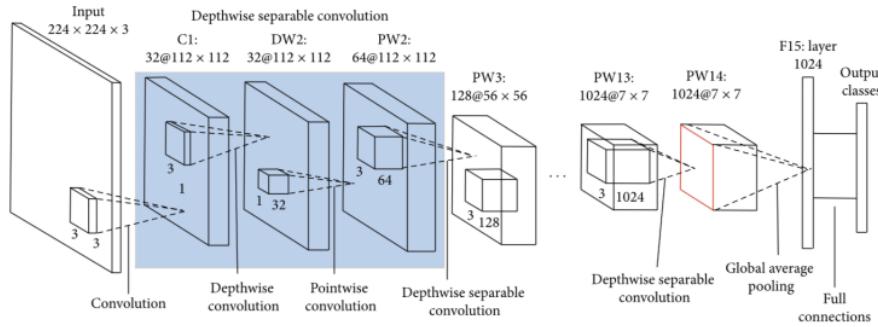


FIGURE 1.11 – Architecter de MobileNet [57]

La classification d'images par ordinateur est essentielle pour remplacer l'interprétation visuelle humaine. Les réseaux de neurones convolutifs (CNN) ont révolutionné ce domaine en permettant l'extraction automatique de caractéristiques à partir des images. Cependant, les modèles traditionnels comme VGG-16 sont complexes et nécessitent beaucoup de ressources, de même pour DenseNet [58]. Pour les applications sur des appareils mobiles avec des ressources limitées, MobileNet offre une solution plus légère. Il utilise des convolutions séparables en profondeur pour réduire le nombre de paramètres tout en maintenant une précision élevée, ce qui le rend idéal pour la classification d'images sur des appareils mobiles.

En particulier, MobileNet a été formé sur le jeu de données ImageNet, ce qui lui permet de capturer une large gamme de caractéristiques visuelles, même avec moins de paramètres.

Dans notre projet, nous avons adopté MobileNet pour classer les cellules de moelle osseuse en raison de sa légèreté et de sa capacité à maintenir des performances élevées sur des images médicales.

1.2.2.8 Apprentissage par transfert

L'apprentissage par transfer (ou le transfert d'apprentissage - Transfer Learning en anglais) est un concept innovant basé sur la conservation des connaissances acquises lors de la résolution d'un problème pour les utiliser ultérieurement dans un problème différent mais connexe. Il met l'accent sur l'utilisation d'un motif pré-entraîné pour renforcer la généralisation dans un autre contexte. Cette approche accélère l'entraînement et améliore les performances de tout modèle d'apprentissage en profondeur, notamment ceux disposant de données relativement restreintes et de ressources informatiques limitées. Le mélange du transfert d'apprentissage avec des sujets de modélisation prédictive, comme dans notre cas, comme nous le verrons plus tard, est très courant.

Dans le cadre de notre projet de classification des cellules de la moelle osseuse à l'aide du modèle MobileNet et de la segmentation des images de frottis de moelle osseuse à l'aide de YOLOv8, il est crucial de souligner l'importance du transfert d'apprentissage, notamment dans le contexte d'un pipeline d'apprentissage automatique. Ce processus permet d'utiliser les connaissances préalablement acquises lors de la classification des cellules pour améliorer la précision de la segmentation des images, et vice versa, dans un workflow intégré d'apprentissage automatique. En utilisant des modèles pré-entraînés pour ces tâches, nous pouvons bénéficier de la généralisation des motifs appris, ce qui est essentiel lorsque les données sont limitées et que les ressources informatiques sont restreintes.

1.3 Étude de l'existant

Après cette brève analyse des concepts et techniques de base pour le développement de notre système, une étude des solutions existantes semble cruciale. Nous commencerons d'abord par mentionner quelques applications similaires de détection des cellules de la MO, puis présenter leurs limites, ce qui nous amènera à proposer notre propre solution et à expliquer notre méthodologie de travail.

1.3.1 Détection de la leucémie promyélocyttaire aiguë dans la moelle osseuse avec l'apprentissage profond sans annotation

Dans cette recherche [59], l'analyse microscopique optique des frottis sanguins et des aspirations de MO par un hématologue est cruciale pour diagnostiquer la leucémie aiguë. Cependant, cette méthode est chronophage et sujette à des erreurs humaines, surtout dans les contextes sans autres méthodes de diagnostic. L'intégration de l'hématopathologie computationnelle automatisée dans les processus cliniques offre une solution pour améliorer l'efficacité des services et réduire les erreurs. Cependant, le manque d'annotations suffisantes des caractéristiques morphologiques cellulaires constitue un obstacle majeur. Les chercheurs ont développé une méthode ayant pour architecture celle illustrée dans la figure 1.12, basée sur l'apprentissage par instance multiple pour l'identification des leucocytes (Multiple Instance Learning for Leukocyte Identification - MILLIE), en utilisant les étiquettes de diagnostic des patients pour former des modèles faiblement supervisés capables de détecter différents types de leucémies aiguës. MILLIE permet une analyse automatique fiable des frottis sanguins avec une supervision minimale. MILLIE est capable de différencier la leucémie lymphoblastique aiguë de la leucémie myéloblastique dans les frottis sanguins avec une précision exceptionnelle (AUC $0,94 \pm 0,04$), représentant ainsi une solution viable pour accélérer les processus cliniques nécessitant une évaluation de la microscopie des frottis sanguins.

1.3.2 Algorithme amélioré YOLOv7 pour la détection des cellules de moelle osseuse

La détection et la classification des cellules de MO sont essentielles en hématologie. Pour remédier à la faible précision des modèles existants, une amélioration nommée YOLOv7-CTA est proposée. Elle intègre le module CoTLAN qui est très similaire au module ELAN [60] pour une meilleure sensibilité aux détails, le module CoordAtt [61] pour une attention accrue aux caractéristiques des petites cibles, et une méthode de clustering K-means++ pour des boîtes d'ancre plus adaptées. Des résultats expérimentaux démontrent une précision moyenne globale de 88,6%, surpassant les modèles Faster R-CNN, YOLOv5l et YOLOv7, une amélioration de 12,9%, 8,3% et 6,7% respectivement ce qui confirme l'efficacité et la supériorité du modèle YOLOv7-CTA dans les tâches de détection des cellules de MO. La figure 1.13 illustre le cadre de détection de leur méthode proposée.

1.3.3 Limites des solutions présentées

Notre projet se distingue par son caractère unique et son intérêt remarquable, malgré l'absence de solutions existantes satisfaisantes. Cette singularité renforce sa pertinence et sa valeur dans le domaine. En effet, les modèles et applications actuellement disponibles se heurtent à plusieurs problématiques majeures. Tout d'abord, bon nombre de ces solutions

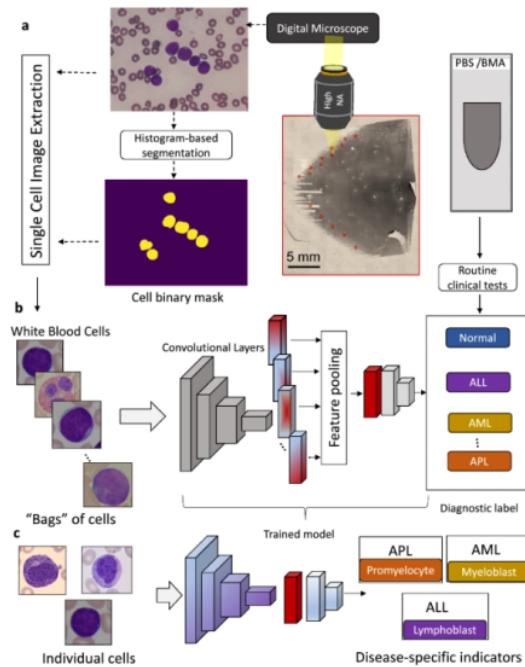


FIGURE 1.12 – Classification avec la méthode MILLIE

sont commerciales propriété, ce qui signifie qu'elles ne sont pas accessibles gratuitement, rendant ainsi leur utilisation prohibitive pour de nombreux chercheurs ou organisations ayant des ressources limitées. De plus, même lorsque ces solutions sont disponibles, elles sont souvent accompagnées de coûts prohibitifs, les rendant inaccessibles pour de nombreuses petites équipes de recherche ou institutions. De plus, certaines de ces applications existantes sont notoirement complexes, nécessitant une expertise technique considérable pour leur utilisation efficace. Cela crée une barrière à l'entrée pour de nombreux utilisateurs potentiels, limitant ainsi leur adoption et leur impact dans la communauté de recherche. En résumé, l'absence de solutions existantes satisfaisantes, combinée aux défis associés à l'utilisation des modèles et applications disponibles, met en lumière l'importance et la nécessité de notre projet, qui vise à proposer une solution accessible, efficace et conviviale pour les chercheurs et les praticiens du domaine.

1.4 Méthodologie et solution proposée

Notre objectif est de développer un système de classification des cellules de la moelle osseuse (MO) qui répond aux exigences de notre projet en utilisant des algorithmes d'ap-

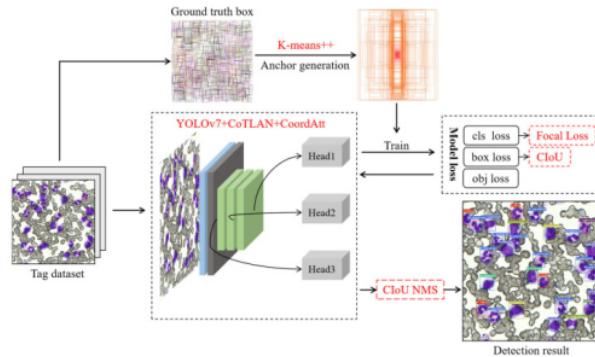


FIGURE 1.13 – Détection avec un modèle YOLOv7-CTA amélioré [62]

apprentissage supervisé. Il est indéniable que les applications de détection d'objets sont des applications de traitement d'image complexes, surtout s'il s'agit d'une détection en temps réel. Nous avons exploré plusieurs approches, notamment les réseaux de neurones convolutifs (CNN), YOLO (You Only Look Once) [63], SAM (Segment Anything Model) [64], etc., pour concevoir et exécuter nos modèles de classification. Notre solution s'est inspirée des systèmes existants et nous avons utilisé les approches présentées dans l'étude préliminaire. En ce qui concerne notre méthodologie de travail, nous avons suivi un modèle itératif qui consiste à répéter les phases de mise en œuvre, de test et de maintenance jusqu'à ce que les exigences de notre projet soient satisfaites. Nous avons également déployé deux modèles : le premier utilisant CNN pour classifier les cellules de la moelle osseuse, et le second un modèle de détection pour segmenter les images de frottis de MO. La figure 1.14 décrit le mieux le modèle itératif que nous avons suivi.

- **Pipeline d'IA pour la Classification et la Détection Cellulaire :**

Les pipelines d'apprentissage automatique sont essentiels pour accélérer, réutiliser et déployer des modèles d'IA efficacement. Inspirés par les pratiques du génie logiciel, ces pipelines simplifient désormais le processus de test et de déploiement, permettant une mise en production rapide et fiable [65]. Dans notre PCD sur la "classification et la segmentation des cellules de frottis de moelle osseuse (MO)" en IA, nous avons utilisé un pipeline pour développer deux modèles d'IA. Le premier, formé sur un ensemble de données Kaggle, identifie les cellules individuelles, tandis que le second, combinant les avantages de MobileNet et de YOLOv8, détecte les cellules de moelle osseuse dans les images de frottis. Cette approche nous a permis de surmonter les défis de disponibilité de données et de déployer efficacement nos modèles.



FIGURE 1.14 – Méthodologie de travail - Modèle itératif

1.5 conclusion

Pour clôturer cette étude introductory, nous avons examiné en détail les aspects fondamentaux de notre projet, notamment en présentant la biologie sous-jacente des échantillons de moelle osseuse et en expliquant comment l'intelligence artificielle jouera en sa faveur. Nous avons défini les objectifs et la portée de notre travail, tout en exposant notre méthodologie et la solution proposée pour les atteindre. Nous avons constaté que, malgré l'absence de solutions existantes satisfaisantes dans ce domaine, notre projet se démarque par son caractère unique et son importance remarquable. En répondant à un besoin crucial, notre solution se distingue par son accessibilité, son efficacité et son caractère innovant. Dans les prochains chapitres, nous approfondirons notre analyse en explorant les aspects techniques, les résultats obtenus, et en envisageant les perspectives futures de notre travail.

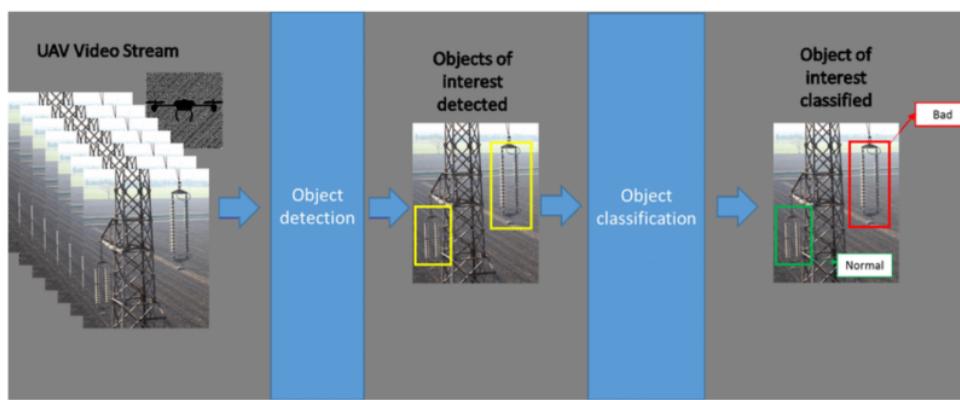


FIGURE 1.15 – Exemple d'un ML pipeline [66]

Chapitre 2

Analyse des besoins

2.1 Expression des besoins

2.1.1 Identification des acteurs

Dans le cas de ce projet, l'application s'adresse à deux types d'acteurs à savoir :

- • Biogiste
- • Administrateur

2.1.2 Identification des besoins

2.1.2.1 Besoins fonctionnels

Les exigences fonctionnelles de notre projet se concentrent sur le développement d'une interface utilisateur conviviale destinée aux biologistes. Cette interface leur permettra de visualiser les données analysées à partir des images d'échantillon fournies par le biais de notre modèle d'IA pré-entraîné, offrant ainsi une précision élevée dans les résultats, soit par le biais d'un glisser-déposer soit par téléchargement au serveur (téléversement). De plus, les biologistes pourront annoter certaines cellules dans les résultats de la sortie. Enfin, notre application leur offrira la possibilité de signaler toute anomalie survenue, afin de fournir un retour d'information essentiel pour améliorer le système.

2.1.2.2 Besoins non fonctionnels

Les exigences non fonctionnelles principales de notre application sont résumées comme suit :

1. – Performances : Le système doit être capable de traiter un grand volume d'images de frottis de la moelle osseuse de manière efficace et rapide et le temps de réponse pour la classification des cellules sanguines doit être raisonnable afin de ne pas retarder le processus d'analyse.

2. – Précision : Le système doit fournir des résultats de classification précis et fiables, en minimisant les erreurs de classification et la sensibilité et la spécificité du système doivent être élevées pour garantir la pertinence clinique des résultats.
3. – Sécurité : Les accès au système doivent être contrôlés par des mécanismes d'authentification et d'autorisation robustes pour empêcher les utilisations non autorisées tout en garantissant la confidentialité et l'intégrité des données des patients, en respectant les normes de sécurité et de protection des données en vigueur.
4. – Évolutivité : Le système doit être conçu pour pouvoir évoluer et s'adapter à l'augmentation du volume de données et aux évolutions technologiques futures et il doit être possible d'ajouter de nouvelles fonctionnalités et de mettre à jour le système sans perturber son fonctionnement.
5. – Facilité d'utilisation : Les processus de chargement des images, de visualisation des résultats et d'annotation des cellules doivent être simples et intuitifs et l'interface utilisateur du système doit être conviviale et intuitive, permettant aux biologistes et aux techniciens de laboratoire de naviguer facilement dans le système et d'accéder aux fonctionnalités pertinentes.
6. – Maintenance et support : Le système doit être facile à maintenir et à dépanner, avec des outils de surveillance et de journalisation pour détecter et résoudre les incidents rapidement.

2.2 Spécifications des besoins

Dans cette partie, nous exprimerons les besoins fonctionnels de l'application d'analyse de cellules d'un frottis de moelle osseuse que nous avons exposé dans la section précédente sous forme de diagrammes UML pour mieux comprendre et structurer les interactions entre notre application et le biologiste (utilisateur).

2.2.1 Diagramme de cas d'utilisation

Nous commencerons par un **diagramme de cas d'utilisation** (voir figure 2.1), dans lequel nous définirons le comportement attendu de notre système, ainsi que les acteurs qui interagiront avec celui-ci.

Comme le montre la figure 2.1 ci-dessous, le premier acteur, qui est le biologiste, peut fournir une image d'un frottis de MO et visualiser les données de classification, une fois que l'image soit segmentée, en obtenant l'image segmentée, l'image annotée et le rapport de classification. Il pourrait aussi signaler les erreurs pour améliorer la performance tout en étant authentifié et autorisé par le système.

L'administrateur, en tant que deuxième acteur, est en mesure de surveiller le système et l'améliorer en fonction des signaux des utilisateurs.

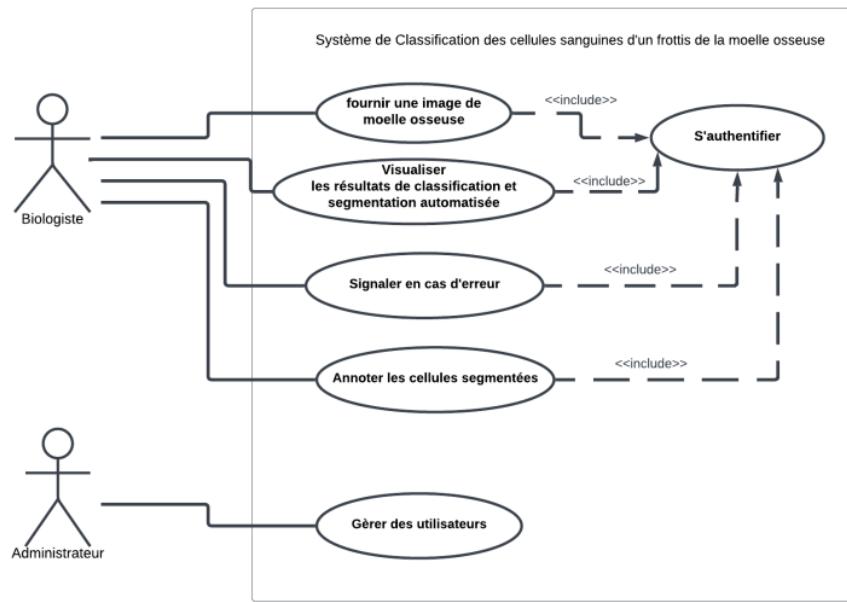


FIGURE 2.1 – Diagramme des cas d'utilisation global.

2.2.2 Diagrammes de séquences

2.2.2.1 Scénario 1 : Visualisation des données de segmentation

le biologiste soumet une image d'échantillon téléchargée au système. Ensuite, ce dernier analyse l'image, marquant le deuxième message, et retourne les données de visualisation au biologiste pour interprétation, marquant le troisième message dans la séquence d'actions.

2.2.2.2 Scénario 2 : Signaler en cas d'erreur ou anomalie

Ce diagramme de séquence représente l'interaction pour le cas d'utilisation "Signaler en cas d'erreur", où le biologiste signale une erreur lors de l'analyse de l'image segmentée générée par le modèle d'IA déployé. L'interaction commence lorsque le biologiste détecte une erreur et signale l'anomalie au système. Le système traite ensuite le signalement de l'erreur et génère une notification pour les équipes de développement ou de maintenance, montrant ainsi la séquence d'actions entre le biologiste et le système.

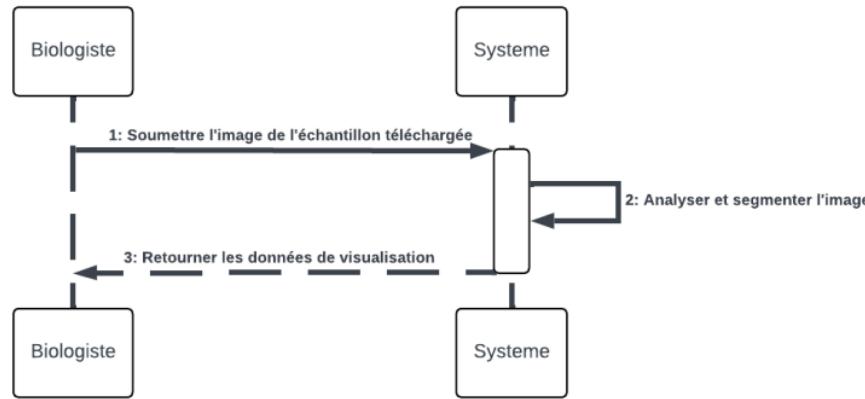


FIGURE 2.2 – Diagramme de séquence du premier scénario.

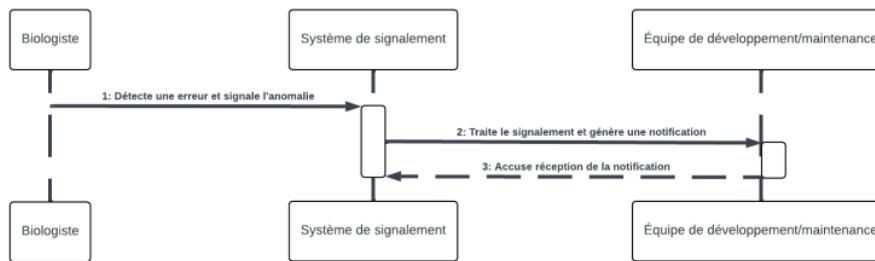


FIGURE 2.3 – Diagramme de séquence du deuxième scénario.

2.2.2.3 Scénario 3 : Authentification

Ce diagramme de séquence représente l'interaction pour le cas d'utilisation "S'authentifier", où le biologiste se connecte à son compte existant ou crée un nouveau compte s'il n'en a pas. L'interaction commence par la tentative d'authentification du biologiste sur le système. Ensuite, le système vérifie les informations d'identification fournies par le biologiste et autorise l'accès au compte existant ou crée un nouveau compte si nécessaire. Le diagramme illustre la séquence d'actions entre le biologiste et le système pour l'authentification.

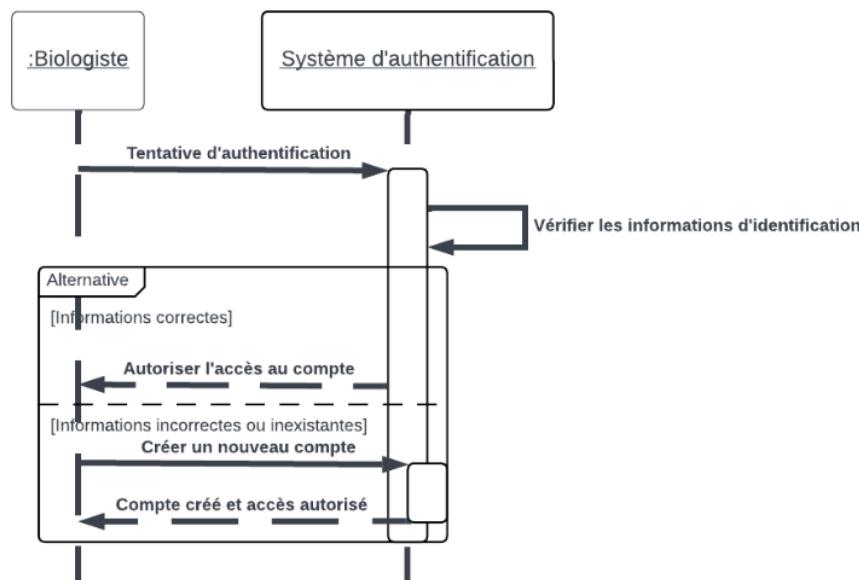


FIGURE 2.4 – Diagramme de séquence du troisième scénario.

2.3 Conclusion

Dans ce chapitre, nous avons réalisé l'analyse des besoins dans le processus de développement du projet. En examinant les spécifications détaillées, y compris le diagramme de cas d'utilisation et les scénarios, nous avons établi une base solide pour la conception et l'implémentation du système. Les informations recueillies dans cette section guideront les étapes suivantes du projet, en assurant une compréhension claire des exigences et des objectifs à atteindre.

Chapitre 3

Développement des Modèles d'Apprentissage Automatique pour la classification et la segmentation d'images

Les expériences ont montré l'efficacité des approches d'apprentissage profond dans les tâches de détection d'objets et de classification d'images, notamment dans le domaine médical.

Dans ce chapitre, nous détaillerons les étapes pour construire un système intelligent, comme mentionné précédemment dans la section de la solution proposée. Notre projet utilise une chaîne de traitement comprenant deux modèles distincts, MobileNet (ou CNN) et YOLO : le premier pour identifier les cellules dans les images microscopiques et le second pour classifier les cellules identifiées.

Nous expliquerons également comment ces deux modèles communiquent au sein du pipeline pour former un système intégré.

Notre projet, avant d'être déployé, se décompose en trois étapes :

- Pré-traitement des données
- Configuration des modèles
- Entraînement des modèles

Il est important de noter que chaque section couvrira les spécificités de deux modèles, en tenant compte de leurs architectures, besoins et objectifs distincts.

3.1 Prétraitement des données

Le prétraitement des données (Data pre-processing) est essentiel dans les tâches de détection d'objets basées sur l'apprentissage profond. Souvent considérée comme la plus critique, cette étape prépare les données pour une interprétation efficace par les modèles. Dans ce chapitre, nous explorerons en détail les étapes de prétraitement des données pour notre projet de détection d'objets.

Étant donné que les modèles CNN et YOLO ont des architectures et des besoins distincts, le prétraitement des données pour chaque modèle diffère. C'est donc l'étape qui varie le plus d'un modèle à l'autre. Nous aborderons le prétraitement des données pour chaque modèle individuellement, en mettant en évidence les différences importantes entre les deux et en montrant comment adapter les techniques en fonction de leurs exigences spécifiques.

3.1.1 Prétraitement des Données CNN

Dans cette sous-section, nous aborderons le prétraitement des données spécifique au modèle CNN. Nous détaillerons les étapes nécessaires pour préparer les données à être utilisées avec ce modèle.

- Données d'Entrée

Nous utiliserons un ensemble de données provenant de Kaggle, une plateforme en ligne connue pour héberger des ensembles de données de divers domaines. Cet ensemble est composé de plus de 170 000 images de cellules moelleuses couvrant 21 classes différentes (voir figure 3.1). Ces images sont annotées par des experts, provenant des frottis de moelle osseuse de 945 patients. Ils constitueront notre matière première pour la classification des cellules. L'acquisition d'images a été réalisée à l'aide d'un microscope en champ clair avec un grossissement de 40x et une immersion dans l'huile.

La figure 3.1 présente les abréviations, les noms complets et le nombre de données pour chaque classe dans notre jeu de données.

3.1.1.1 Nettoyage des Données CNN

Nous avons d'abord commencé par le nettoyage des données. En se basant sur nos hypothèses et recherches dans le domaine de l'hématologie, nous avons décidé de réduire le nombre d'exemples par classe afin de mieux équilibrer notre ensemble de données. Cette démarche était justifiée par le besoin de rendre notre jeu de données plus représentatif de

| Abréviation | Signification | Nombre d'images |
|-------------|----------------------|-----------------|
| ABE | Abnormal eosinophil | 8 |
| ART | Artefact | 19630 |
| BAS | Basophil | 441 |
| BLA | Blast | 11973 |
| EBO | Erythroblast | 27395 |
| EOS | Eosinophil | 5883 |
| FGC | Faggott cell | 47 |
| HAC | Hairy cell | 409 |
| KSC | Smudge cell | 42 |
| LYI | Immature lymphocyte | 65 |
| LYT | Lymphocyte | 26242 |
| MNZ | Metamyelocyte | 3055 |
| MON | Monocyte | 4040 |
| MYB | Myelocyte | 6557 |
| NGB | Band neutrophil | 9968 |
| NGS | Segmented neutrophil | 29424 |
| NIF | Not identifiable | 3538 |
| OTH | Other cell | 294 |
| PEB | Proerythroblast | 2740 |
| PLM | Plasma cell | 7629 |
| PMO | Promyelocyte | 11994 |
| Total | | 171374 |

FIGURE 3.1 – Ensemble de données (Dataset) avec 21 classes

la réalité et plus pertinent pour notre objectif ultérieur.

En éliminant certaines classes avec très peu d'exemples par rapport aux autres et en fusionnant des classes similaires, nous avons rationalisé la variabilité de notre ensemble de données. Cette pratique est courante dans le domaine de la classification d'images, notamment en médecine, où la qualité et la pertinence des données sont essentielles pour obtenir des résultats précis.

En réduisant le nombre d'exemples par classe, nous avons également contribué à améliorer l'équilibre de notre ensemble de données, ce qui permettra à notre modèle de effectuer des généralisations plus solides et d'éviter d'être biaisé vers les classes sur-représentées.

Ces changements rendront notre ensemble de données plus représentatif de notre problème et permettront à notre modèle d'apprentissage automatique de mieux appréhender les caractéristiques discriminantes des différentes classes, améliorant ainsi ses performances lors de la classification d'images hématologiques.

À la fin du processus, notre ensemble de données comprenait 41373 images et un total de 13 classes, représentée dans la Figure 3.2 sous forme de table de classes.

3.1.1.2 Réduction de Dimension

Nous avons redimensionné les images en (224x224) pixels et nous avons vérifié que chaque image avait uniquement trois canaux (224x224x3) pour correspondre aux dimensions d'entrée requises par le modèle CNN (voir Figure 3.3). Cette standardisation était nécessaire car les images originales avaient des tailles de (250x250) pixels, et certaines

| Abréviation | Signification | Nombre d'images |
|-------------|------------------|-----------------|
| ART | Artefact | 3000 |
| BLA | Blast | 3000 |
| EBO | Erythroblast | 3000 |
| EOS | Eosinophil | 3000 |
| LYT | Lymphocyte | 3000 |
| MMZ | Metamyelocyte | 3055 |
| MON | Monocyte | 4040 |
| MYB | Myelocyte | 3000 |
| NGS | Band neutrophil | 4000 |
| NIF | Not identifiable | 3538 |
| PEB | Proerythroblast | 2740 |
| PLM | Plasma cell | 3000 |
| PMO | Promyelocyte | 3000 |
| Total | . | 41373 |

FIGURE 3.2 – Table des 13 classes (ensemble de données réduit)

images comprenaient un canal alpha supplémentaire qui indique la transparence des pixels. En supprimant le canal alpha, nous avons assuré la cohérence des données d'entrée et la compatibilité avec l'architecture MobileNet.

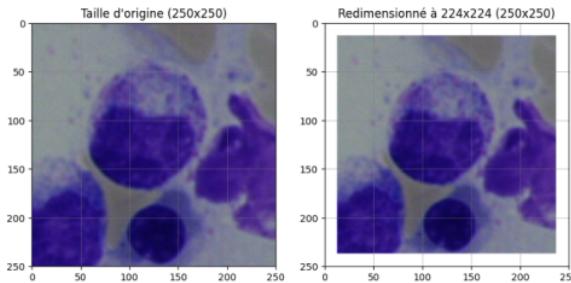


FIGURE 3.3 – Une image originale et sa version redimensionnée

3.1.1.3 Étiquetage des Données

Pour entraîner un modèle CNN, les données d'entraînement doivent être étiquetées dans un format que le modèle peut apprendre et mettre en œuvre. Cette étape, appelée étiquetage des données, consiste à associer chaque image à une étiquette qui représente la classe à laquelle elle appartient.

Nous avons utilisé des scripts Python pour associer chaque image à une étiquette correspondant à sa classe. Voici le processus que nous avons suivi :

Collecte des Images : Nous avons commencé par collecter les chemins d'accès des images et leurs étiquettes associées à partir du répertoire de données. Nous avons utilisé une fonction `collect_images` qui parcourt les répertoires de classes et leurs sous-répertoires, récupérant les chemins d'accès des images et extraire leurs étiquettes à partir du nom de l'image ou de son répertoire , et associer cette étiquette à l'image correspondante dans une Dataframe.

Création d'une Dataframe : En utilisant les chemins d'accès et les étiquettes collectés, nous avons créé une Dataframe avec deux colonnes : "images" contenant les chemins d'accès des images et "labels" contenant les étiquettes des classes. Cette datafram nous a fourni une structure organisée pour nos données d'entraînement.

3.1.1.4 Partitionnement des Données

En apprentissage automatique, il est essentiel de diviser l'ensemble de données en ensembles d'entraînement et de test. L'ensemble d'entraînement est utilisé pour entraîner le modèle, tandis que l'ensemble de test est utilisé pour évaluer ses performances. Cette répartition garantit que les performances du modèle peuvent être évaluées sur des données non vues, fournissant une mesure plus précise de ses capacités de généralisation.

La répartition des données est importante pour s'assurer que l'ensemble d'entraînement et celui de test sont représentatifs de l'ensemble de données global. Cela permet d'éviter le surajustement, où le modèle apprend à mémoriser les données d'entraînement plutôt qu'à généraliser à partir d'elles. En assurant un équilibre adéquat entre les deux ensembles, nous pouvons obtenir des mesures de performance fiables et évaluer plus efficacement l'efficacité du modèle.

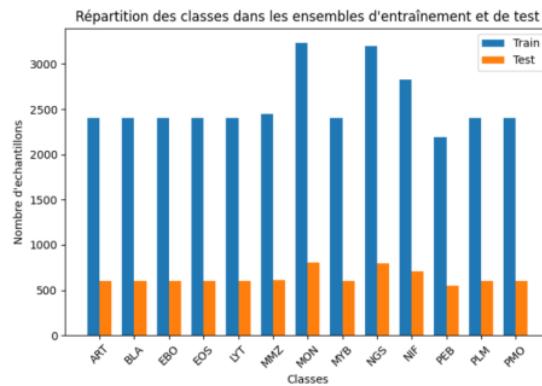


FIGURE 3.4 – Répartition des Classes dans l'Ensemble d'Entraînement et de Test

Ce graphique présente la répartition des classes dans les ensembles d'entraînement et

de test. Chaque barre représente le nombre d'images par classe, avec les classes organisées verticalement sur l'axe des ordonnées."Nous avons adopté une approche de répartition des données en 80% pour l'ensemble d'entraînement et 20% pour l'ensemble de test, conformément à la norme, après avoir testé autres répartitions et constaté que celle-ci donnait les meilleurs résultats.

3.1.1.5 Augmentation des données

La "data augmentation" est une technique largement utilisée pour augmenter la taille et la diversité des ensembles de données d'entraînement, ce qui peut améliorer les performances des modèles de réseaux de neurones et prévient le surapprentissage en introduisant une variabilité supplémentaire dans l'ensemble de données d'entraînement, ce qui aide le modèle à s'adapter aux données de test. Nous avons appliquée différentes transformations aux images de notre ensemble de données pour introduire des variations et enrichir la variabilité des données.

Une performance médiocre du modèle est obtenue lorsqu'il est confronté à des données qu'il n'a pas vu pendant l'entraînement, phénomène connu sous le nom de **surapprentissage**, ou surajustement (overfitting en anglais). Il se produit lorsqu'un modèle d'apprentissage machine s'adapte trop étroitement aux données d'entraînement, au point de ne pas généraliser correctement aux données de test ou à de nouvelles données. Cela se traduit par une performance médiocre du modèle lorsqu'il est confronté à des données qu'il n'a pas vu pendant l'entraînement.

- Sélection minutieuse des techniques

Les techniques d'augmentation des données ont été choisies avec soin pour correspondre au mieux à la nature des images microscopiques de cellules que nous traitons. Par exemple, la rotation simule la rotation de la lame du microscope, tandis que le zoom simule les variations d'échelle observées lors de la mise au point sur différentes parties de l'échantillon. La variation de la luminosité simule différentes conditions d'éclairage rencontrées lors de la capture des images. Nous avons également expérimenté plusieurs techniques via un processus itératif pour trouver celles qui donnent les meilleurs résultats.

- Techniques de Data Augmentation

Nous avons utilisé les techniques suivantes pour amplifier nos données :

1. Rotation aléatoire : Nous avons effectué des rotations aléatoires sur les images dans une plage spécifiée pour simuler différentes orientations.

2. Translation horizontale et verticale : Nous avons déplacé les images horizontalement et verticalement de manière aléatoire pour simuler des changements de position.
3. Miroir horizontal : Nous avons appliqué une symétrie horizontale aux images pour augmenter la variation.
4. Zoom aléatoire : Nous avons effectué un zoom aléatoire sur les images pour simuler des variations d'échelle.
5. Luminosité aléatoire : Nous avons ajusté aléatoirement la luminosité des images pour simuler différentes conditions d'éclairage.
6. Rescale. Nous avons normalisé les valeurs des pixels en les divisant par 255 pour mettre les valeurs des pixels dans la plage [0, 1].
7. Décalage de canal : Nous avons déplacé aléatoirement les canaux de couleur des images pour simuler des variations de couleur.

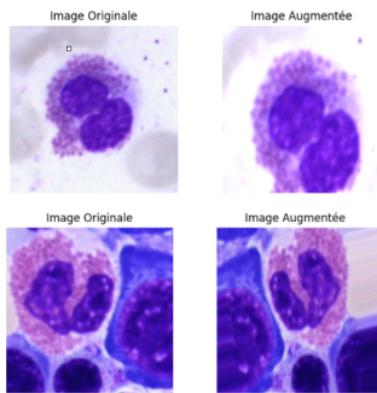


FIGURE 3.5 – Comparaison entre les images originales et les images augmentées. Les deux premières colonnes montrent les images originales, suivies des images augmentées avec les techniques de zoom/rotation et de changement de luminosité.

3.1.1.6 Conclusion

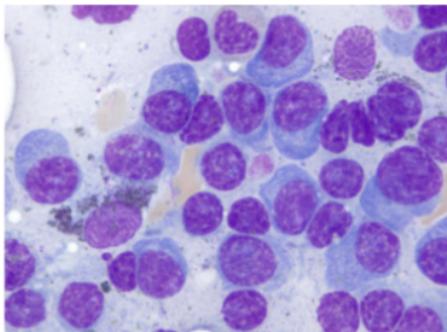
Dans cette section, nous avons abordé les techniques de pré-traitement. Il nous reste maintenant à configurer le modèle et à le passer à l'entraînement.

3.1.2 Prétraitement des données YOLOv8

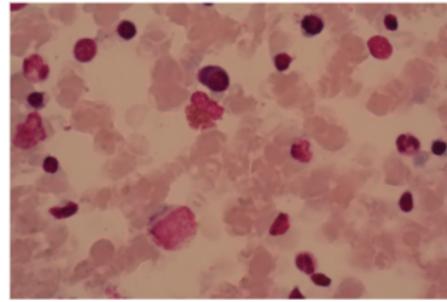
Dans cette sous-section, nous abordons le prétraitement des données spécifiquement adapté à YOLO. Le traitement des données pour YOLO diffère considérablement en raison de sa méthode unique de gestion des boîtes englobantes et des annotations. Nous

couvrirons en détail ces aspects spécifiques, essentiels pour la préparation efficace des données utilisées par ce modèle.

La collecte de données a combiné des images internet et des prélèvements directs en centres de laboratoire. Nous avons fait face à un défi notable : le manque de disponibilité en ligne d'images microscopiques annotées de moelle osseuse. Cette approche mixte a été essentielle pour rassembler un ensemble de données suffisamment diversifié pour notre application.



(a) Image en ligne



(b) Image prélevée d'un laboratoire local

FIGURE 3.6 – Images collectées pour la phase de prétraitement

3.1.2.1 Annotation des Données YOLO

Contrairement aux classificateurs d'images tels que MobileNet (Single Cell Classifier - SCC), l'annotation des données pour des modèles de détection d'objets comme YOLO implique une approche distincte.

- **Qu'est-ce que l'annotation des données en détection d'objets ?**

L'annotation des données pour la détection d'objets consiste à identifier visuellement et à marquer les zones d'intérêt sur les images. Cela permet au modèle d'apprendre où se trouvent précisément les objets qu'il doit détecter.

3.1.2.1.1 Boîtes englobantes et zones d'intérêt

Dans ce cadre, les zones d'intérêt (Region Of Interest - ROI) sont marquées à l'aide de boîtes englobantes (voir Figure 3.7). Ces boîtes délimitent visuellement les objets cibles dans les images. Pour notre projet, les zones d'intérêt sont les cellules sanguines. L'objectif est de distinguer ces cellules du fond et d'autres cellules non pertinentes.

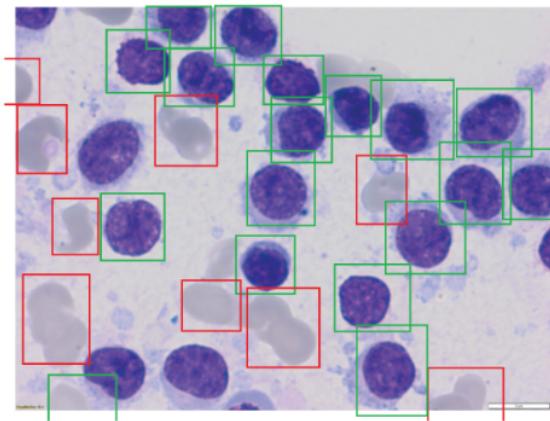


FIGURE 3.7 – Visualisation des zones d'intérêts en vert, et les zone non pertinentes en rouge

3.1.2.1.2 Manque de données annotées et pratique courante

Aucun ensemble de données pré-annoté n'était disponible pour nos besoins spécifiques, nous avons donc dû procéder à une annotation manuelle. Cette pratique est courante dans le domaine de la segmentation d'images médicales, où les données spécifiques sont souvent rares.

3.1.2.1.3 Formats d'Annotation

Pour annoter nos images, nous avons utilisé deux formats largement utilisés dans le domaine du Computer Vision : COCO JSON et YOLO(.txt). Après avoir défini ces formats, nous expliquerons pourquoi nous les avons choisis pour notre travail de annotation d'images.

3.1.2.1.4 COCO JSON (Common Objects in Context)

Le format COCO JSON est largement utilisé pour annoter les images dans des ensembles de données contenant des objets dans différents contextes. Dans ce format, chaque annotation est représentée par un objet JSON qui contient des informations telles que les coordonnées des boîtes englobantes, les catégories d'objets et les identifiants d'image. Ce format est polyvalent et largement pris en charge par de nombreux outils d'annotation et de traitement d'images.



FIGURE 3.8 – Exemple d'annotation au format COCO JSON

Dans cette figure, nous présentons un exemple d'annotation au format COCO JSON. Chaque annotation est représentée par un objet JSON avec plusieurs attributs :

- **id** : L'identifiant unique de l'annotation.
 - **image_id** : L'identifiant de l'image à laquelle l'annotation est associée.
 - **category_id** : L'identifiant de la catégorie de l'objet annoté.
 - **segmentation** : Les coordonnées de segmentation de l'objet.
 - **area** : L'aire de l'objet en pixels.
 - **bbox** : Les coordonnées de la boîte englobante de l'objet (x, y, largeur, hauteur).

3.1.2.1.5 YOLO.(txt)

Le format YOLO(.txt) est spécifique à l'outil de détection d'objets YOLO. Dans ce format, chaque ligne du fichier texte correspond à une annotation et contient les informations suivantes : classe de l'objet, coordonnées de la boîte englobante (centre x, centre y, largeur, hauteur). Ce format est simple et efficace pour les tâches de détection d'objets.

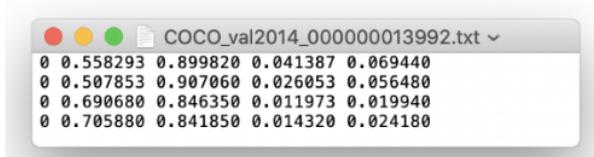


FIGURE 3.9 – Exemple d'un fichier yolo(.txt)



FIGURE 3.10 – Format de la boîte englobante dans le réseau YOLO

3.1.2.1.6 Outils utilisés pour l'annotation

Dans cette partie, nous avons utilisé deux formats et donc deux outils : Roboflow et Mokesense.ia. Cela est dû au fait que YOLO n'accepte que les annotations au format YOLO (.txt), et il n'est pas possible d'exporter directement ce format à partir des outils d'annotations manuelles. Cependant, il est possible d'exporter ces annotations dans le format cocojson, et dans Roboflow, nous importons les fichiers cocojson et les exportons ensuite au format YOLO.

Roboflow est une plateforme en ligne complète qui permet d'annoter et de préparer des données pour l'apprentissage automatique. Elle offre diverses fonctionnalités, y compris des outils d'annotation et techniques de pré-procussions dans le domaine de détection d'objets , ainsi que des fonctionnalités de conversion vers différents formats, notamment le

format YOLO (.txt).

Makesense.ia, quant à lui, est une autre plateforme d'annotation axée sur la précision et la facilité d'utilisation. Elle offre des outils sophistiqués pour annoter les données. Makesense.ia permet également d'exporter les annotations dans le format cocojson.

Nous avons d'abord utilisé Makesense.ia pour délimiter les zones d'intérêt sur nos images en traçant des contours (voir Figure 3.11), puis nous les avons exportées au format COCOJson (voir Figure 3.12).

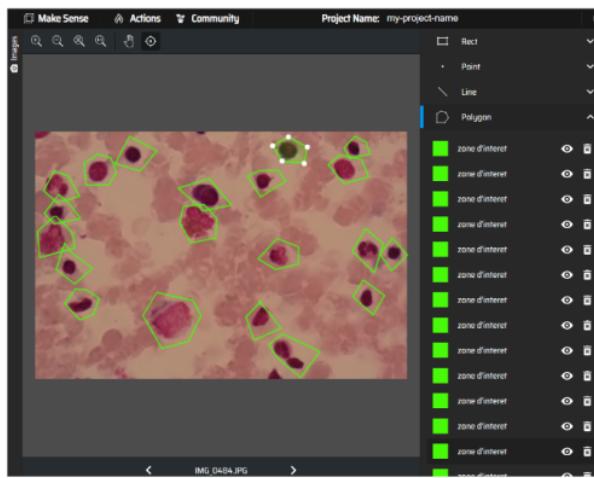


FIGURE 3.11 – Interface de Makesense.ai montrant les polygones dessinés autour de nos zones d'intérêt (ROI)

Une fois nos annotations exportées, . Nous avons donc importé les images ainsi que leurs fichiers cocojson associés dans Roboflow qui offre à la fois la conversion en format YOLO et les étapes de prétraitement supplémentaires. Nous aborderons en détail ces étapes dans la prochaine section.

Dans cette représentation, nous observons l'interface de prétraitement de Roboflow, présentant nos images sources accompagnées de leurs annotations, prêtes à être intégrées dans le pipeline de prétraitement.

3.1.2.2 Répartition entraînement/test

Comme évoqué dans le CNN, nous devons diviser nos données en ensembles d'entraînement et de validation. Nous avons réparti nos 8 images initiales en 6 pour l'entraînement et

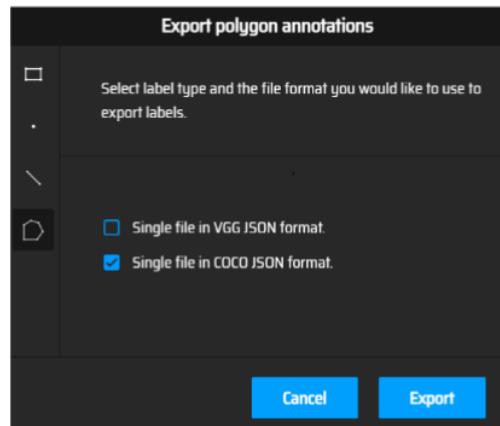


FIGURE 3.12 – l’exportation des zones délimitées de nos ensemble d’images au format cocojson dans Makesense.ia.

2 pour la validation (nous avons opté pour une approche 80/20). Nous n’avons pas affecté d’images à l’ensemble de test en raison du nombre limité d’annotations disponibles. Ces chiffres augmenteront ultérieurement grâce à l’augmentation que nous aborderons.

3.1.2.3 Redimensionnement des données et orientation automatique

Ici, nous avons redimensionné les images en 640x640 pour les adapter à l’entrée de notre modèle YOLO, car elles présentaient des résolutions variables. Nous avons également appliqué une **orientation automatique** (illustrée dans Figure 3.13) pour préserver les coordonnées des boîtes englobantes lorsque les données sont augmentées par rotation ou d’autres transformations.

3.1.2.4 Augmentation des Données

Comme précédemment pour le CNN, nous avons opté pour les techniques d’augmentation de données proposées par Roboflow.

Pour modéliser notre problème, nous avons utilisé des rotations et des techniques de modification de la luminosité pour simuler différents microscopes. Nous avons également appliqué des transformations en niveaux de gris, des rotations et des modifications de teinte (voir Figure 3.14) pour renforcer la capacité du modèle à segmenter les zones d’intérêt (ROI) de manière plus robuste. Ces choix sont justifiés dans le contexte de la microscopie et de la détection d’objets (OD), visant à augmenter la taille de nos données dans des cas



FIGURE 3.13 – Affichage de deux images côte à côté, l'une avec l'orientation automatique désactivée (qui a entraîné un décalage des boîtes englobantes lors de l'application de la rotation) et la suivante avec l'orientation automatique appliquée

où elles sont limitées.

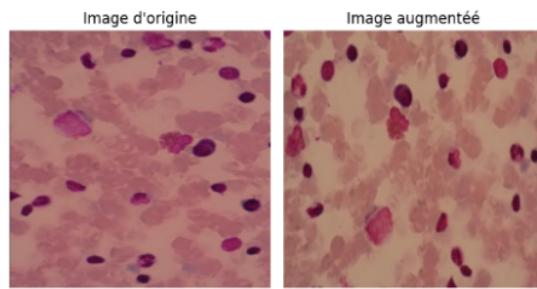


FIGURE 3.14 – Visualisation de changement de teint, image originale à gauche et à droite sa version teintée.

Grâce à l'augmentation, nous avons obtenu jusqu'à 20 images (18 pour l'entraînement et 2 pour la validation). Nous avons privilégié l'entraînement ici en raison du nombre limité d'exemples.

Maintenant que les étapes de prétraitement sont terminées, nous devons simplement exporter et convertir notre ensemble de données.

3.1.2.5 Exportation et creation du fichier .yaml

Roboflow nous permet d'exporter nos données dans le format YOLOv8 (voir Fig. 3.15).

Ainsi, maintenant, notre ensemble de données est exporté sous forme de deux dossiers : "training" (entraînement) et "validation", chacun contenant des dossiers "images" et "labels". Chaque image est associée à son fichier yolo.(txt) correspondant. En plus, un fichier data.yaml est créé (voir Fig. ??) contenant des informations sur cette structure de projet. Ce fichier yaml sera alimenté au modèle lors de la phase d'entraînement afin qu'il sache comment naviguer dans le projet et obtenir les images avec leurs étiquettes correspondantes, le nombre de classes (nc) est défini à 1, car il s'agit d'une seule classe : la zone d'intérêt (ROI) (cellule sanguine).

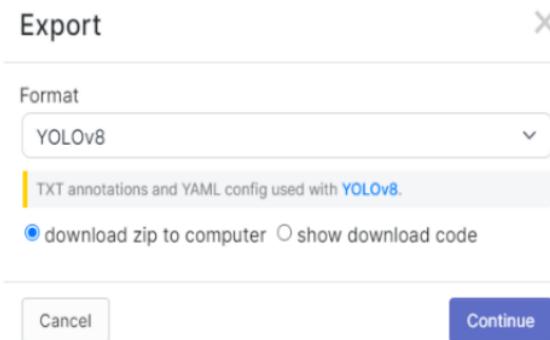


FIGURE 3.15 – Interface Roboflow montrant l'exportation des données au format YOLOv8

3.1.2.6 Conclusion

En conclusion ,nous avons annoté manuellement nos données, en définissant nos zones d'intérêt (ROI) à l'aide de boîtes englobantes pour les cellules sanguines. Nous avons utilisé les formats COCO JSON et YOLO (.txt) pour nos annotations, avec les outils Roboflow et Makesense.ai.

Nous avons appliqué des techniques d'augmentation de données pour augmenter la taille de notre ensemble de données et améliorer la robustesse de notre modèle de détection. Après avoir exporté nos données dans le format YOLOv8 avec un fichier data.yaml, nous sommes prêts à configurer le modèle.

3.2 Configuration des modèles

3.2.1 Configuration du modèle YOLO

3.2.1.1 Configuration de l'environnement

Nous avons d'abord téléchargé notre ensemble de données annotées au format YOLO. Comme Kaggle ne dispose pas de la bibliothèque Ultralytics pré-installée, nous l'avons installée. Cette bibliothèque nous a permis de télécharger le modèle YOLO pré-entraîné qui a été entraîné sur le jeu de données COCO comprenant 328 000 images annotées avec différentes classes d'objets (voir Figure 3.16). Le jeu de données COCO comprend une vaste collection d'images contenant des objets courants.



FIGURE 3.16 – Exemple d'images du coco dataset avec leurs annotation

3.2.1.2 Configuration des rappels

YOLO ajuste automatiquement et dynamiquement ses paramètres tels que le taux d'apprentissage, surveille ses métriques et met en œuvre un mécanisme d'arrêt précoce, ainsi que le choix de l'optimiseur, etc., dont nous traiterons dans la section suivante.

3.2.1.3 Compilation du modèle

Pour YOLO, la compilation du modèle est effectuée lors de l'appel de la commande d'entraînement.

3.2.2 Configuration du modèle MobileNet

3.2.2.1 Mise en place de l'environnement

Nous avons d'abord téléversé les données prétraitées sur la plateforme Kaggle, puis importé les bibliothèques nécessaires de science des données qui sont préinstallées dans cet environnement.

Nous avons téléchargé le modèle MobileNet pré-entraîné grâce à l'API TensorFlow Keras. Ce modèle a été entraîné sur le jeu de données ImageNet, composé de plus d'1,2 million d'images réparties en 1 000 classes, notre travail de configuration du modèle consistera principalement à trois choses : choisir quelles couches figer ou dégeler, ajuster correctement les hyperparamètres et déterminer quelles couches personnalisées ajouter. Nous avons sélectionné ces couches et hyperparamètres au cours d'un processus itératif d'expérimentation et d'évaluation afin d'optimiser les performances de notre modèle pour notre tâche spécifique.

Nous avons ajouté une couche dense de sortie personnalisée. Cette couche était composée de 13 nœuds, correspondant au nombre de classes dans notre ensemble de données. Chaque nœud de la couche dense représente une classe, permettant au modèle de produire une distribution de probabilité sur les 13 classes, pour cela.

La fonction Softmax est couramment utilisée dans la classification d'images pour convertir les scores bruts en probabilités normalisées sur les différentes classes. Elle est généralement associée à une couche dense, où elle transforme la sortie en une distribution de probabilités, facilitant ainsi l'interprétation des prédictions du modèle.

Lorsqu'une image traverse le réseau, la classe prédite est déterminée par le nœud avec la probabilité la plus élevée à la sortie du modèle.

Dans notre étude, nous avons également pris en compte la similarité entre les classes de notre ensemble de données, composé de 13 types de cellules très similaires. En effet, cette similitude augmente le risque de surapprentissage, où le modèle pourrait se fier excessivement à des caractéristiques spécifiques de chaque classe pour effectuer ses prédictions.

Pour contrer ce problème, nous avons ajouté une couche de dropout à notre architecture modèle. Cette couche de dropout, placée avant la couche dense, désactive aléatoirement un pourcentage des neurones pendant l'entraînement. Cette stratégie permet d'encourager le réseau à apprendre des représentations plus robustes et généralisables des classes, en évitant qu'il ne dépende excessivement de certaines caractéristiques spécifiques à chaque classe.

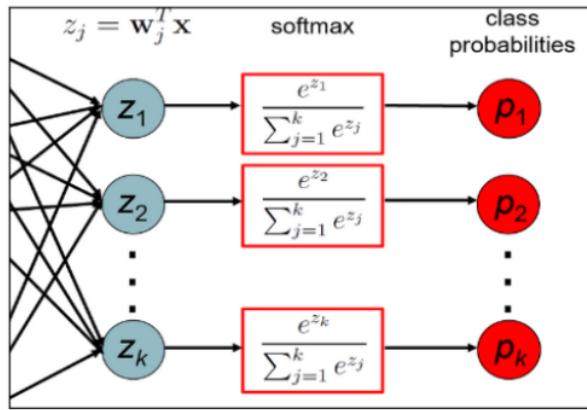


FIGURE 3.17 – illustration d'une couche dense avec k nœuds, suivi d'une activation softmax. Dans notre cas, $k = 13$, correspondant au nombre de classes dans notre ensemble de données.

Dans notre cas, nous avons configuré le dropout avec $p = 0.5$ comme paramètre , ce qui signifie qu'à chaque itération, la moitié des neurones sont désactivés de manière aléatoire.

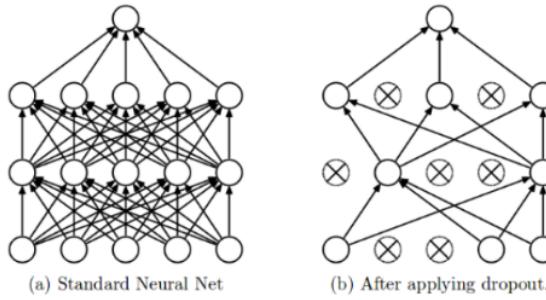


FIGURE 3.18 – Illustration du fonctionnement de l'abandon (dropout)

3.2.2.2 Application de la régularisation L2

Toujours dans le contexte de la similarité entre classes de notre jeu de donnée, il est important d'empêcher le modèle de se suradapter à des caractéristiques spécifiques.

Par exemple, considérons le noyau des cellules, qui est une caractéristique importante pour distinguer les différents types de cellules. Cependant, si le modèle se base fortement

sur les caractéristiques du noyau, comme sa taille ou sa forme, il risque de se suradapter à ces caractéristiques et de peiner à généraliser à de nouvelles données.

En appliquant la régularisation L2, nous pénalisons les poids excessivement grands associés à des caractéristiques spécifiques telles que le noyau.

Cela encourage le modèle à considérer un éventail plus large de caractéristiques, au-delà du seul noyau, pour effectuer ses prédictions.

Mathématiquement, la régularisation L2 ajoute un terme de pénalité proportionnel au carré de la magnitude des poids à la fonction de perte.

La fonction de perte régularisée $E_{\text{régularisation}}(w)$ est définie comme suit :

$$E_{\text{régularisation}}(w) = E(w) + \frac{\lambda}{2} \|w\|^2$$

Où :

- $E(w)$ est la fonction de perte originale.
- $\frac{\lambda}{2} \|w\|^2$ est le terme de régularisation L2.
- λ est le paramètre de régularisation.
- $\|w\|^2$ est la norme L2 au carré du vecteur de poids.

En minimisant cette fonction de perte régularisée lors de l'entraînement, l'optimiseur est incité à trouver des poids qui minimisent à la fois la perte originale et la pénalité de régularisation. Ainsi, le modèle apprend des représentations plus robustes, réduisant le risque de surapprentissage et améliorant sa capacité à classifier avec précision des types de cellules similaires.

3.2.2.3 Déblocage et blocage des couches

En CNN, les couches plus profondes capturent des détails de plus en plus spécifiques dans les données. Par exemple, les premières couches d'un modèle préentraîné peuvent apprendre des caractéristiques de base telles que la couleur et les bords, tandis que les couches plus profondes se concentrent sur des caractéristiques plus complexes et spécifiques aux données. C'est pourquoi il est courant de geler les premières couches et de débloquer les dernières lors du fine-tuning d'un modèle CNN. En gelant les premières couches, nous conservons les représentations de caractéristiques de bas niveau apprises lors de la préformation, tandis que le déblocage des dernières couches permet à notre modèle de capturer des caractéristiques plus spécifiques à notre ensemble de données.

Après de multiples itérations et expérimentations, nous avons observé que la stratégie consistant à débloquer les six dernières couches offre les meilleures performances pour notre jeu de données sur les cellules, parallèlement, nous gelons les autres couches pour maintenir les caractéristiques générales apprises lors du pré-entraînement.

3.2.2.4 Fonction de rappel

Nous avons mis en place une fonction de rappel personnalisée à l'aide de la bibliothèque `tensorflow.keras.callbacks`.

Les fonctions de rappel sont invoquées pendant l'entraînement pour effectuer des actions spécifiques à des moments prédéterminés, tels que la fin d'une époque d'entraînement ou après chaque itération, ils surveillent les paramètres du modèle, tels que les fonctions de perte, et peuvent ajuster des paramètres tels que le taux d'apprentissage en fonction de l'évolution des performances du modèle (voir Figure 3.19).

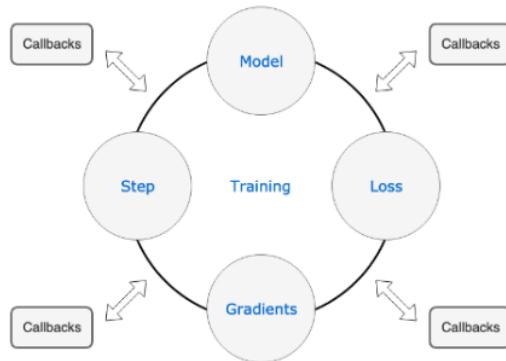


FIGURE 3.19 – Interaction des callbacks au model dans sa phase d'entraînement

Dans notre cas, avec notre fonction de rappel personnalisé, nous avons décidé de surveiller d'abord la précision d'entraînement. Si cette précision atteint un seuil désiré, nous surveillons ensuite la précision de validation. Cette surveillance nous permet d'ajuster dynamiquement le taux d'apprentissage si les performances ne s'améliorent pas après un certain nombre d'époques.

De plus, nous avons implémenté le rappel d'arrêt anticipé (early stopping), ce qui signifie arrêter l'entraînement si les métriques surveillées ne s'améliorent pas en fonction du nombre d'époques passées.

La Figure 3.20 illustre les erreurs d'entraînement et de validation en fonction du nombre d'époques. L'erreur d'entraînement diminue de manière constante à mesure que le nombre

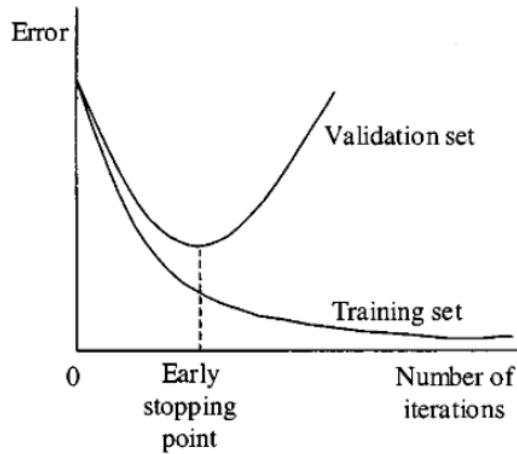


FIGURE 3.20 – Schéma illustrant le mécanisme d'arrêt anticipé

d'époques augmente, ce qui indique que le modèle apprend à partir des données d'entraînement. Cependant, l'erreur de validation diminue initialement mais atteint ensuite son minimum et commence à augmenter, suggérant que les performances du modèle sur des données non vues commencent à se détériorer. C'est un phénomène de surapprentissage.

3.2.2.5 Compilation du modèle

Une fois que l'architecture du modèle est définie, la prochaine étape est de le compiler. La compilation du modèle consiste à définir l'optimiseur, la fonction de perte et les métriques à utiliser pendant l'entraînement.

Pour notre modèle, nous avons utilisé l'optimiseur Adamax avec un taux d'apprentissage initial de 0.01. Adamax est une variante de l'optimiseur Adam qui a montré de bonnes performances dans nos expériences comparatives avec d'autres optimiseurs. Les optimiseurs sont des algorithmes qui ajustent les poids du réseau de neurones lors de l'entraînement à chaque époques afin de minimiser la fonction de perte.

Nous avons également choisi la fonction de perte de l'entropie catégorielle, adaptée à notre tâche de segmentation multi-classe. Cette fonction de perte est couramment utilisée dans les problèmes de classification multi-classe.

Il est à noter que nous avons testé plusieurs optimiseurs et avons constaté que Adamax offrait les meilleures performances pour notre modèle.

Nous avons initialement utilisé un taux d'apprentissage relativement élevé afin d'accélérer la convergence du modèle vers le minimum de la fonction de perte. Cela a permis à notre modèle de progresser rapidement dans l'espace des paramètres pendant les premières étapes de l'entraînement. Toutefois, grâce à notre callback personnalisé, ce taux d'apprentissage est réduit dynamiquement en fonction des métriques surveillées. Ainsi, même avec un taux d'apprentissage initial élevé, notre modèle est capable d'apprendre de manière robuste et de s'adapter aux variations des données d'entraînement tout en évitant le surapprentissage.

Une fois que le modèle est compilé, il est prêt à passer à la phase d'entraînement.

3.3 Entraînement de deux modèles

3.3.1 Entraînement de YOLO

Pendant la phase d'entraînement, plusieurs étapes sont effectuées. Le modèle extrait d'abord les caractéristiques pertinentes des images à l'aide d'un réseau de neurones convolutifs. Ensuite, il prédit les boîtes englobantes pour chaque objet détecté, dans notre cas les zones d'intérêt, et compare ces prédictions aux annotations réelles (voir Figure 3.21) pour calculer la perte. L'optimiseur ajuste alors les poids du réseau pour minimiser cette perte, ce qui améliore les performances de détection.

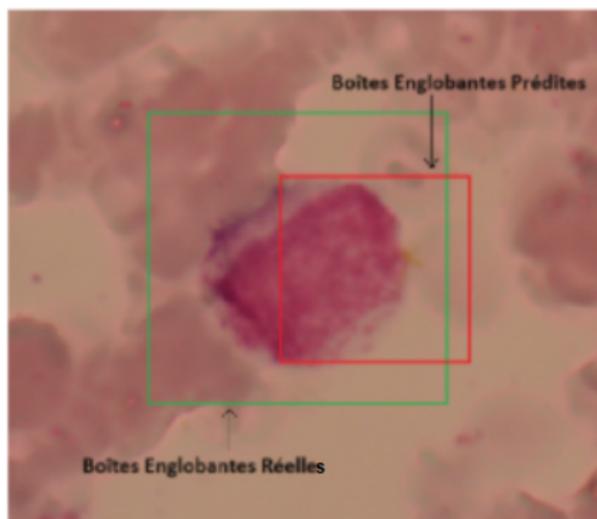


FIGURE 3.21 – Boîtes englobantes prédictives vs réelle

Nous lançons l'entraînement avec la commande `model.train()`, en passant par paramètres le chemin vers le fichier `data.yaml`, Nombre d'époques, device = 'cuda' pour acceleration GPU, et le paramètre `single_cls` est défini sur True pour indiquer le travail avec une seule classe. Après l'exécution de la commande, le modèle définit ses hyperparamètres comme spécifié dans la section de configuration. L'optimiseur AdamW est utilisé avec un taux d'apprentissage initial de 0.002. Le modèle est entraîné sur 200 époques sans mécanisme d'arrêt anticipé, avec une taille de lot de 4 et des images redimensionnées à 640 pixels.

Nous connectons ensuite le modèle à la plateforme Weights and Biases(W&B) en utilisant notre clé API.

Weights and Biases(W&B) est une plateforme de suivi des expériences en apprentissage automatique qui permet de visualiser et de comparer les performances des modèles, d'analyser les résultats et de collaborer efficacement sur les projets. L'interface affiche diverses métriques et visualisations, fournissant des informations sur les performances du modèle pendant l'entraînement.

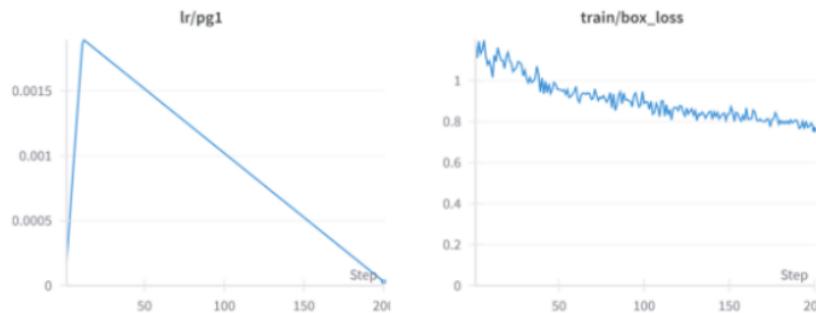


FIGURE 3.22 – Ajustement dynamique du taux d'apprentissage et de la perte d'entraînement pour les boîtes englobantes

La Figure 3.22 illustre l'ajustement dynamique du taux d'apprentissage par époque lors de l'entraînement du modèle YOLO, suivi à l'aide de la plateforme Weights & Biases.

Le graphique montre qu'au fur et à mesure de l'entraînement, le taux d'apprentissage est ajusté dynamiquement pour optimiser les performances du modèle. Initialement, le taux d'apprentissage est relativement élevé pour permettre une convergence rapide. Au fur et à mesure que l'entraînement progresse, le taux d'apprentissage diminue progressivement pour affiner les paramètres du modèle et stabiliser le processus d'entraînement.

Simultanément, la fonction de perte d'entraînement pour les boîtes englobantes (loss training/box) diminue régulièrement au fil des époques. Cela indique que le modèle apprend efficacement à partir des données d'entraînement, comme le montre la réduction des erreurs de prédiction.

La perte d'entraînement pour les boîtes englobantes (loss training/box) mesure l'écart entre les boîtes englobantes prédites et les boîtes englobantes réelles dans les images d'entraînement. Plus précisément, elle évalue la précision de la localisation des objets détectés. Une perte plus faible indique une meilleure précision de localisation des objets.



FIGURE 3.23 – Evolution de la perte de validation pour les boîtes englobantes

Cette figure (Fig. 3.23) montre l'ajustement dynamique du taux d'apprentissage par époque et la perte de validation pour les boîtes englobantes (loss validation/box) lors de l'entraînement du modèle YOLO. Comme pour la perte d'entraînement, la perte de validation diminue progressivement au fil des époques, indiquant une amélioration des performances du modèle sur les données de validation.

Cela signifie que le modèle est capable de détecter des zones d'intérêt non vues auparavant, ce qui est essentiel pour sa capacité à généraliser et à fournir des résultats précis sur de nouvelles données. Cependant, il convient de noter que cette métrique pourrait ne pas être aussi fiable que prévu en raison du nombre limité d'exemples dans notre ensemble de données de validation. Nous aborderons cette limitation plus en détail dans le chapitre d'évaluation de performance pour ce modèle.

En conclusion, les graphiques générés par Weights and Biases ont montré une amélioration constante des performances du modèle au fil du temps. Après un entraînement de seulement 20 minutes pour 200 époques, démontrant ainsi son efficacité en matière d'entraînement grâce à son architecture optimisée, le modèle est prêt à être évalué.

3.3.2 Entraînement de MobileNet

3.3.2.0.1 Initiation de l'entraînement

Nous avons initié l'entraînement du modèle MobileNet en utilisant la commande `model.fit()`, passant en arguments les données d'entraînement prétraitées et une fonction de rappel personnalisée.

ainsi qu'un objet CSVLogger pour enregistrer les métriques d'entraînement dans un fichier CSV. Cela nous permet de récupérer les données pendant l'entraînement, car elles ne sont disponibles que pour la session en cours (voir figure 3.24).

Le nombre d'époques a été fixé à 12, avec une taille de lot (batch size) de 32.

La "taille de lot" représente le nombre d'échantillons qui sont propagés à travers le réseau de neurones avant que les poids ne soient mis à jour. Une taille de lot plus grande peut conduire à une convergence plus rapide, mais il est plus coûteux en termes de mémoire et de temps de calcul. D'autre part, une petite taille de lot peut être moins coûteuse mais peut nécessiter plus d'itérations pour converger.

Nous avons choisi une taille de lot en fonction de nos expériences préliminaires. Et avons constaté que la taille de lot de 32 nous permettait d'obtenir une convergence rapide sans sacrifier trop de ressources computationnelles.

| epoch | acc | loss | lr | val_acc | val_loss |
|-------|----------|----------|----------|----------|----------|
| 0 | 0.712188 | 0.923783 | 5.00E-05 | 0.7348 | 0.865529 |
| 1 | 0.746875 | 0.845359 | 5.00E-05 | 0.733946 | 0.870366 |
| 2 | 0.739688 | 0.853802 | 5.00E-05 | 0.7348 | 0.869381 |
| 3 | 0.734375 | 0.872552 | 5.00E-05 | 0.7348 | 0.818775 |
| 4 | 0.735 | 0.817328 | 5.00E-05 | 0.744981 | 0.782691 |
| 5 | 0.738125 | 0.813451 | 5.00E-05 | 0.743201 | 0.75689 |

FIGURE 3.24 – Exemple du contenu d'un fichier CSV comprenant le nombre d'époques, les fonctions de perte, et les précisions d'entraînement et de validation par époque

3.3.2.0.2 Durée de l'entraînement et coût computationnel

Nous avons observé que chaque époque a pris environ 15 minutes à s'exécuter, malgré l'utilisation de l'accélération GPU performante de Kaggle. Ainsi, l'ensemble de l'entraînement, sur 12 époques, a duré environ 3 heures au total.

Cette durée relativement longue souligne la nature coûteuse en termes de ressources computationnelles des réseaux de neurones convolutifs. Bien que l'accélération GPU puisse réduire le temps nécessaire pour entraîner un modèle, les CNNs restent des modèles complexes qui nécessitent des calculs intensifs, surtout lorsqu'ils sont appliqués à des tâches complexes telles que la segmentation d'images.

3.3.2.0.3 Évolution des métriques d'entraînement et de validation

Les Figures 3.25 et 3.26 illustrent l'évolution des métriques d'entraînement et de validation pendant le processus d'apprentissage.

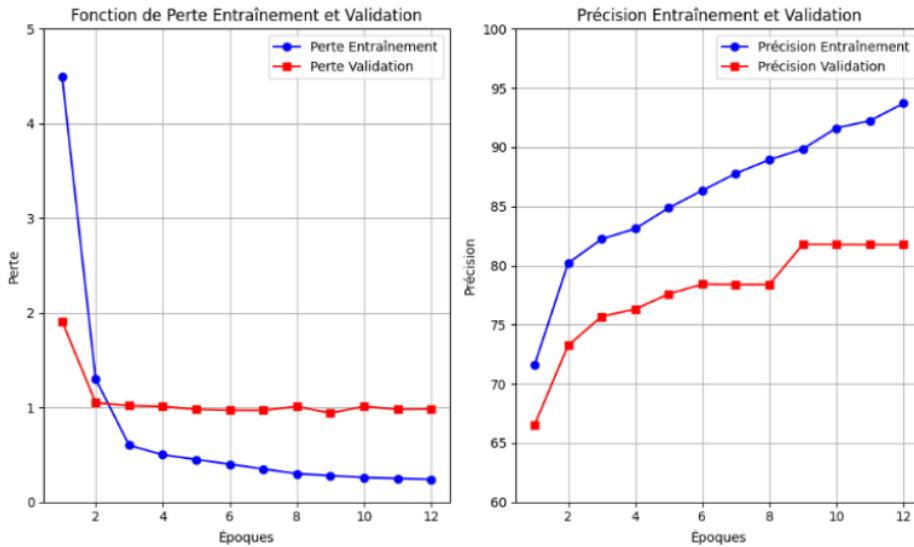


FIGURE 3.25 – Courbes de pertes et de précision par époque pour le modèle MobileNet

La Figure 3.25 montre les courbes de perte et de précision pour les données d'entraînement et de validation par époque. On peut observer que la perte diminue progressivement

tandis que la précision augmente au fil des époques pour les deux ensembles de données.

Cependant, il est intéressant de noter que la précision de validation peut commencer à stagner ou à diminuer après un certain nombre d'époques, ce qui peut indiquer un possible surapprentissage du modèle.

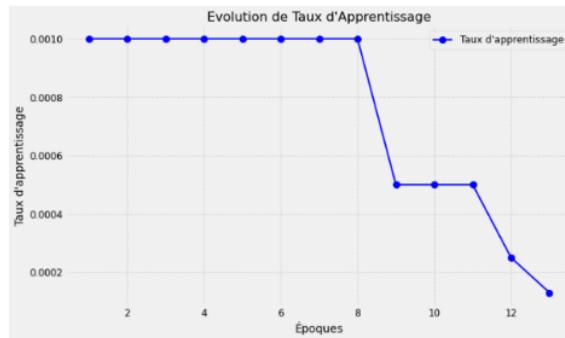


FIGURE 3.26 – Evolution du pas d'apprentissage (graphe obtenu grâce au fichier CSV qui journalise l'évolution de LR)

La Figure 3.26 ci-dessous présente l'évolution du taux d'apprentissage au cours des époques. On peut voir que le taux d'apprentissage diminue progressivement au fur et à mesure que l'entraînement progresse. Cette décroissance du taux d'apprentissage permet au modèle de converger de manière plus stable vers un minimum de perte. Il est important de noter que le taux d'apprentissage change lorsque la valeur surveillée ne s'améliore pas, grâce au callback que nous avons mis en place pour ajuster dynamiquement les métriques surveillées.

On observe qu'à partir de la 6ème époque que la valeur surveillée, la précision de validation, n'a pas montré d'amélioration pendant deux époques consécutives. En réponse à cela, le taux d'apprentissage a été réduit, et nous avons constaté une amélioration de la précision de validation dès l'époque suivante, démontrant ainsi l'importance de la fonction de rappel dans l'optimisation de l'entraînement du modèle.

3.3.2.1 Conclusion

L'entraînement du modèle MobileNet a été réalisé avec succès sur 12 époques, en utilisant des données d'entraînement prétraitées et des techniques de régularisation, d'optimisation et de gestion du taux d'apprentissage. Les métriques d'entraînement et de validation

ont montré une amélioration progressive, tandis que le taux d'apprentissage a été ajusté dynamiquement grâce à notre callback personnalisé.

Avec l'entraînement terminé, le modèle est maintenant prêt à être évalué sur un ensemble de données de test pour évaluer sa performance en généralisation.

3.4 Conclusion

En conclusion de ce chapitre, nous avons exploré en détail la configuration et l'entraînement de deux modèles de détection d'objets : YOLO et MobileNet. Nous avons décrit en profondeur les processus de configuration, de compilation et de mise en œuvre de ces modèles, mettant en évidence les stratégies spécifiques utilisées pour optimiser leurs performances. Les résultats de l'entraînement, illustrés par des métriques et des graphiques pertinents, ont montré une amélioration constante au fil du temps pour les deux modèles.

Dans le prochain chapitre, "Test et Validation", nous évaluerons ces modèles et discuterons du déploiement de notre application grâce à un pipeline d'apprentissage automatique. Nous utiliserons le modèle MobileNet entraîné comme composant clé, intégré dans un pipeline YOLOv8 pour segmenter les images de frottis de moelle osseuse. Ce chapitre nous permettra de tirer des conclusions sur l'efficacité et la pertinence de nos modèles dans un contexte pratique, tout en explorant les défis et les opportunités liés à leur déploiement dans un environnement réel.

Chapitre 4

Tests et Validation

Pendant ce chapitre, nous présenterons notre environnement de travail. Ensuite, nous aborderons l'évaluation des performances des deux modèles avant de traiter le déploiement et l'intégration dans notre pipeline. Enfin, nous évoquerons quelques-unes des difficultés rencontrées dans la poursuite de nos objectifs.

4.1 Environnement de travail

Dans cette section, nous présenterons un aperçu des environnements matériels et logiciels utilisés pour mener à bien ce projet.

4.1.1 Environnement Matériel

Initialement, nous avons utilisé nos propres ressources informatiques pour nous familiariser avec le développement de ce modèle dans ses premières phases. De plus, l'utilisation des notebooks Kaggle nous a facilité la phase d'entraînement des deux modèles séparés (MobileNet et YOLOv8), en particulier MobileNet, malgré les limitations. Nous avons ensuite choisi Google Colab comme environnement de travail principal pour le reste du projet en raison de sa compatibilité étroite avec Gradio. Cette décision s'est avérée particulièrement pertinente car Gradio offre une intégration fluide avec Google Colab, permettant ainsi une utilisation efficace de ses fonctionnalités avancées, telles que le mode de partage. Cette synergie entre Google Colab et Gradio a grandement facilité le développement et le déploiement de notre pipeline. En effet, cette combinaison nous a permis de créer une interface utilisateur interactive pour notre modèle, accessible même sur des appareils mobiles, ce qui a considérablement amélioré l'accessibilité et l'expérience utilisateur de notre application. Nous examinerons cela en détails dans la phase de déploiement.

| Composant | Unité de travail 1 | Unité de travail 2 |
|-----------------|-------------------------------------|-----------------------------------|
| Processeur | AMD Ryzen 7 4800H 2.90GHz / 4.20GHz | Intel i5-10300H 2.50GHz / 4.50GHz |
| Mémoire vive | 16.00 Go | 8.00 Go |
| Carte graphique | NVIDIA GTX 1660-ti (Max-Q Design) | NVIDIA GTX 1650-ti |

TABLE 4.1 – Spécifications des unités de travail

4.1.2 Environnement Logiciel

Dans ce projet, on a eu recours à :

- **Langage de Programmation**

Python : Un langage de programmation de haut niveau généralement utilisé pour l'analyse de données, l'apprentissage automatique, la construction de modèles d'IA, etc.

- **Bibliothèques :**

- **TensorFlow** : Une bibliothèque open-source d'apprentissage automatique développée par Google, utilisée pour la construction et l'entraînement de modèles d'apprentissage automatique, y compris les réseaux de neurones.
- **Numpy** : Une bibliothèque Python utilisée pour effectuer des calculs numériques, en particulier pour travailler avec des tableaux multidimensionnels et des matrices.
- **PIL (Python Imaging Library)** : Une bibliothèque Python qui ajoute des fonctionnalités de traitement d'images à Python, permettant de charger, manipuler et sauvegarder différents formats d'images.
- **SciKit Learn** : Une bibliothèque Python open-source qui offre des outils simples et efficaces pour l'analyse de données et l'apprentissage automatique, y compris la classification, la régression, le clustering, etc.
- **Ultralytics** : Une bibliothèque Python spécialisée dans la vision par ordinateur et la détection d'objets, offrant des outils pour entraîner et évaluer des modèles de détection d'objets.
- **Matplotlib** : Une bibliothèque Python utilisée pour créer des visualisations statiques, telles que des graphiques, des diagrammes et des parcelles (plots).
- **Open-CV (Open Source Computer Vision Library)** : Une bibliothèque open-

source de vision par ordinateur et de traitement d'images, principalement utilisée pour effectuer des tâches telles que la détection d'objets, la reconnaissance faciale, etc.

- **IDE** : Notre environnement de développement intégré (IDE) était un notebook Colab ou un notebook Kaggle, qui sont des environnements interactifs pour écrire et exécuter du code Python.

4.2 Évaluation des Performance des modèles

Une fois l'entraînement terminé, les deux modèles étaient prêts à être exportés et testés sur nos données locales. Nous avons évalué les modèles un par un en utilisant les matrices de confusion et les rapports de classification chaque fois que possible, en commençant par MobileNet puis YOLOv8.

4.2.1 Évaluation de la Performance de MobileNet

4.2.1.1 Introduction

Dans cette section, nous évaluerons les performances du modèle MobileNet. L'évaluation des performances d'un modèle est importante pour comprendre sa capacité à généraliser à de nouvelles données. Dans cette étude, nous avons utilisé un ensemble de données de test distinct de l'ensemble d'entraînement.

4.2.1.2 Définition des métriques

Avant de présenter les résultats de l'évaluation, il est important de définir les métriques de performance que nous utiliserons :

4.2.1.2.1 Précision

La précision mesure la proportion d'instances positives correctement prédictes (vrais positifs) parmi toutes les instances prédictes comme positives par le modèle. Mathématiquement, elle est définie comme :

$$\text{Précision} = \frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Positifs}}$$

4.2.1.2.2 F1-score

Le F1-score est une mesure qui combine à la fois la précision et le rappel du modèle. Il est calculé comme la moyenne harmonique de la précision et du rappel, donnant une mesure de la précision et de la complétude du modèle. Il est défini comme :

$$\text{F1-Score} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

4.2.1.2.3 Rappel (Recall)

Le rappel mesure la capacité du modèle à retrouver tous les exemples positifs. Il est calculé comme le rapport entre le nombre d'exemples positifs correctement prédits et le nombre total d'exemples positifs dans les données.

$$\text{Rappel} = \frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Négatifs}}$$

4.2.1.3 Matrice de confusion

La matrice de confusion est une table qui compare les vraies valeurs des échantillons dans un ensemble de données avec les prédictions faites par le modèle. Elle met en évidence les vrais positifs, qui se trouvent sur la diagonale principale de la matrice, ce qui indique les classifications correctes pour chaque classe.

4.2.1.4 Analyse des résultats

Nous commençons par présenter la matrice de confusion (Fig. 4.1). En effet, l'analyse de la matrice de confusion nous fournit des informations sur les performances du modèle. Les éléments diagonaux représentent le nombre d'échantillons correctement classés pour chaque classe. Des valeurs plus élevées indiquent une meilleure performance. Par exemple, la classe "EOS" (éosinophiles) a une précision élevée avec 574 échantillons correctement classés sur 600, tandis que la classe "PMO" (promyélocytes) a une précision légèrement plus faible avec 450 échantillons correctement classés sur 600.

Les éléments non diagonaux de la matrice de confusion représentent les erreurs de classification. En examinant ces valeurs, nous pouvons identifier les classes qui sont fréquemment

confondues entre elles. Par exemple, la classe "NGS" est souvent confondue avec la classe "MMZ", ce qui indique une certaine similitude dans les caractéristiques des images.

Dans l'ensemble, la matrice de confusion montre que le modèle a de bonnes performances pour la plupart des classes, mais peut bénéficier d'une amélioration dans la distinction entre certaines classes similaires.

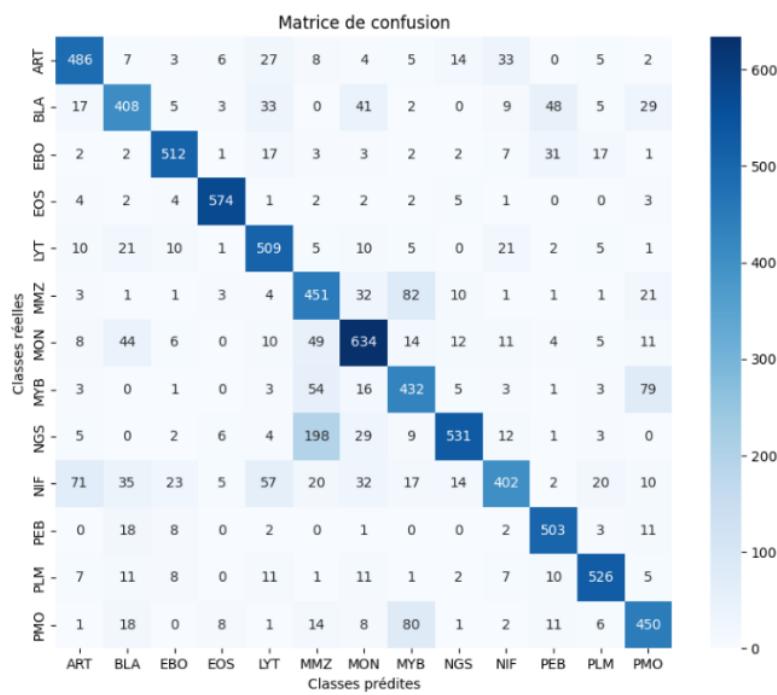


FIGURE 4.1 – Matrice de confusion

• Rapport de classification :

L'analyse du rapport de classification nous donne un aperçu des performances du modèle pour chaque classe. Chaque classe est évaluée en termes de précision, de rappel et de score F1.

Précision : "ART" a une précision de 0.79, ce qui signifie que 79% des échantillons classés comme artefacts sont réellement des artefacts.

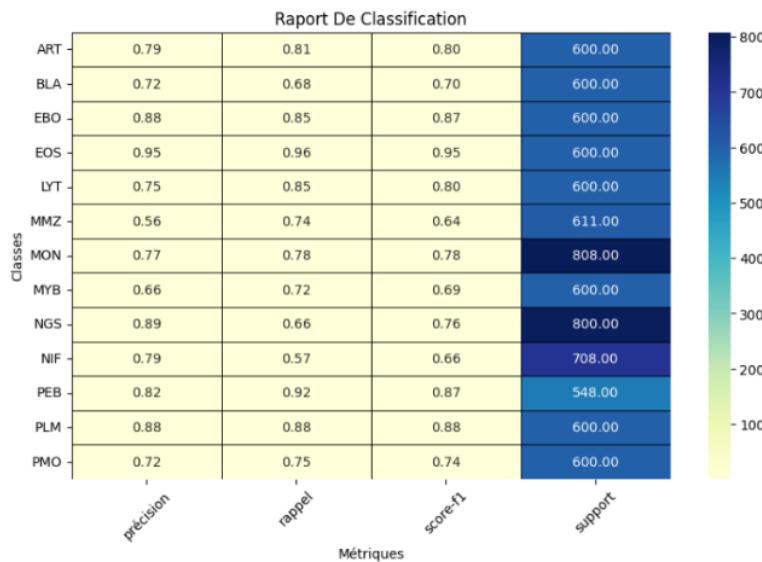


FIGURE 4.2 – Rapport de classification

Rappel : Le rappel représente la proportion d'échantillons correctement classés pour une classe spécifique parmi tous les échantillons réels de cette classe. Par exemple, la classe "EOS" a un rappel de 0.96, ce qui signifie que 96% des éosinophiles ont été correctement identifiés par le modèle.

****score F1**** : Le Score F1 est une mesure combinée de la précision et du rappel, donnant une indication de l'exactitude globale du modèle. Pour notre modèle "MMZ", ayant une précision de 0.56 et un rappel de 0.74, le Score F1 est de 0.64. Cela indique que le modèle atteint un équilibre entre la précision et le rappel pour la classe "MMZ", avec une performance globale de 64%.

En regardant ces mesures pour chaque classe, nous pouvons voir que le modèle a généralement de bonnes performances, mais il y a des variations entre les classes. Par exemple, les classes "EOS" et "PLM" ont des scores F1 élevés, tandis que les classes "LYT" et "MMZ" ont des scores F1 plus bas.

En conclusion, l'évaluation de notre modèle montre des performances globalement satisfaisantes, avec une précision moyenne de 78%. Les métriques de précision, de rappel et de score F1 fournissent des informations détaillées sur la capacité du modèle à distinguer entre les différentes classes de cellules. Bien que certaines classes présentent des perfor-

mances plus élevées que d'autres, il existe des opportunités d'amélioration, notamment en explorant des stratégies telles que l'ajustement des hyperparamètres, l'exploration d'architectures de réseaux neuronaux plus complexes et l'enrichissement des données et des caractéristiques.

4.2.2 Évaluation de la Performance du modèle YOLOv8

L'évaluation des performances du modèle YOLO a été un défi, principalement en raison du manque de jeux de données pré-annotés disponibles. Nous avons été obligés de prioriser la majorité des données que nous avions pour l'entraînement, dans le but que le modèle soit capable de généraliser sur des données non vues. En conséquence, nous disposions de très peu d'exemples pour l'ensemble de validation.

Pour évaluer les performances du modèle dans de telles conditions, nous avons opté pour l'observation des résultats de segmentation. Cette approche nous a permis d'avoir un aperçu direct de la capacité du modèle (voir Figure 4.3) malgré le manque de données de validation.

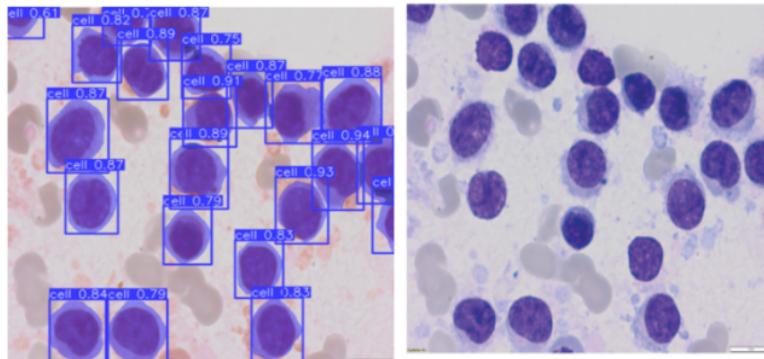


FIGURE 4.3 – Le résultat de la segmentation d'image par le modèle YOLO, avec un score de confiance associé à chaque objet détecté. On constate que le modèle a réussi à détecter toutes les cellules cibles dans notre image, fournissant ainsi une indication de ses performances.

Le paramètre de confiance permet d'éliminer certains objets mal détectés en ajustant la confiance en dessus de l'attribut de confiance de ces objets.

On peut constater que malgré les défis liés au manque de données annotées et au nombre limité d'exemples dans notre ensemble de validation, le modèle YOLO a démontré une performance satisfaisante lors de l'inférence sur des données non vues, confirmant ainsi

son utilité pour notre projet.

4.3 intégration des modèles via pipeline

4.3.1 Introduction au Pipeline d'Apprentissage Automatique (ML)

Le pipeline est amorcé par l'envoi de l'image microscopique au modèle YOLO. Tout d'abord, l'image est redimensionnée à 640x640 pixels pour correspondre à la taille d'entrée requise par YOLO. Il performe inférence sur l'image et retourne un objet de détection contenant la visualisation de l'image segmentée ainsi que tous les attributs, tels que les coordonnées de chaque boîte englobante prédite et son centre.

4.3.2 Détection et Localisation des Cellules

Une fois les boîtes englobantes prédites, chaque boîte est localisée et délimitée par un carré autour de son centre de gravité, maintenant ainsi un rapport hauteur/largeur de 1 :1. Cette étape est importante pour conserver la cohérence dimensionnelle des images afin de correspondre à la taille d'entrée de MobileNet. En préservant cette proportion, les caractéristiques des cellules ne sont pas altérées lors du redimensionnement, ce qui garantit que MobileNet peut extraire efficacement les informations pertinentes pour la classification.

4.3.3 Prétraitement pour l'Inférence de MobileNet

les images extraites subissent un prétraitement pour les préparer à l'inférence de **MobileNet**. Les images sont normalisées et ensuite redimensionnées en (224x224).

4.3.4 Inférence de MobileNet et Annotation de Classe

Les images prétraitées sont ensuite introduites dans **MobileNet** il prédit alors l'étiquette de classe pour chaque image de cellule.

À la suite de classification , on revient aux coordonnées des boîtes englobantes de l'image d'origine pour annoter chaque boîte avec l'étiquette de classe prédite. Ce processus itératif se poursuit jusqu'à ce que toutes les cellules de l'image soient annotées(voir Figure 4.4). Nous avons également enregistré le nombre d'instances de chaque classe pour chaque cellule.

4.3.5 Conclusion

Dans cette sous-section, nous avons introduit le concept de pipeline d'apprentissage automatique utilisé dans notre projet. Ce pipeline comprend plusieurs étapes essentielles, notamment la détection et la localisation des cellules, le prétraitement pour l'inférence de

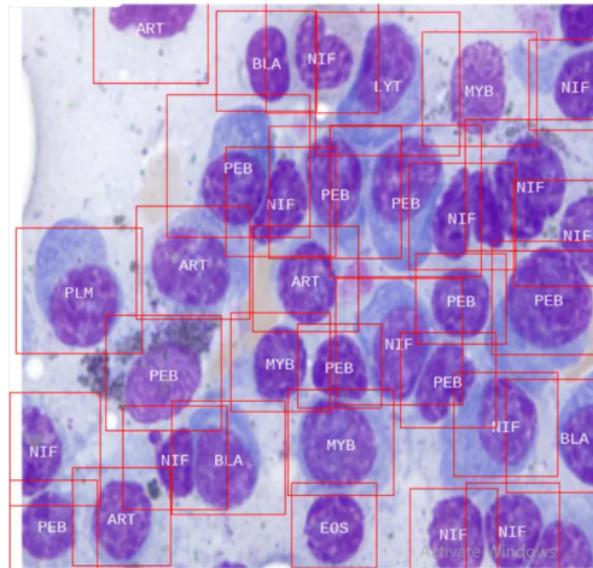


FIGURE 4.4 – Segmentation des cellules et annotation des classes

MobileNet, l'inférence de MobileNet pour l'annotation de classe, et enfin l'annotation des cellules avec les étiquettes prédites. Grâce à cette approche itérative, nous parvenons à annoter efficacement chaque cellule de l'image d'origine avec les classes correspondantes. En combinant ces différentes étapes, notre pipeline permet une analyse précise et détaillée des images de frottis de MO.

4.4 Déploiement du pipeline

4.4.1 Introduction au Déploiement

Le déploiement d'un modèle d'apprentissage automatique consiste à le rendre accessible et utilisable par d'autres utilisateurs. Avec la montée en puissance des plateformes en ligne, il est devenu plus facile de partager et de distribuer des modèles d'apprentissage automatique, permettant ainsi à un large public d'en bénéficier. Dans notre projet, nous avons choisi de déployer notre modèle via Gradio en raison de sa simplicité d'utilisation et de sa compatibilité multiplateforme.

4.4.2 Gradio

C'est une bibliothèque open-source qui permet de déployer facilement des modèles d'apprentissage automatique et d'interagir avec eux via une interface utilisateur graphique.

4.4.3 Environnement de travail

Nous avons utilisé Google Colab comme environnement de travail pour notre projet, car il offre une intégration transparente avec Gradio. Cette combinaison nous a permis de développer et de déployer notre pipeline de manière efficace.

4.4.4 Caractéristiques

Nous avons utilisé le même mécanisme de pipeline décrit dans le chapitre précédent pour notre déploiement sur Gradio. Notre ordinateur agit comme un serveur, permettant aux biologistes de télécharger leurs images et d'interagir avec notre modèle via une interface graphique.

4.4.5 Utilisation de l'interface Gradio

Lorsque l'utilisateur télécharge son image sur l'application, il peut ajuster les paramètres tels que la confiance et IoU avant d'obtenir la segmentation.

L'interface propose 4 boutons : recevoir l'image segmentée, obtenir la classification, signaler et obtenir le rapport de classification.

L'utilisateur peut expérimenter avec différents scores de confiance jusqu'à obtenir le résultat désiré. A ce stade, l'image annotée est prête à être récupérée dès que l'utilisateur appuie sur le bouton de récupération des annotations. Cette approche est conçue pour optimiser les performances, car l'obtention des annotations est plus coûteuse en termes de ressources que la segmentation.

Pour cette démonstration, nous allons présenter l'application en cours d'exécution sur un appareil mobile afin de mettre en évidence la capacité multiplateforme de Gradio. Notre machine locale agit comme un serveur et héberge l'application grâce au lien public accessible généré par Gradio.

Cette interface (illustrée dans la Figure 4.5) offre une expérience utilisateur conviviale et intuitive pour la segmentation d'images.

Une image téléchargée est présentée aux côtés de sa version segmentée (voir Figure 4.6), avec des scores de confiance pour chaque cellule segmentée. Remarquez comment une cellule de moelle osseuse a été incorrectement identifiée. L'ajout de barres de confiance



FIGURE 4.5 – L'interface de l'application Gradio

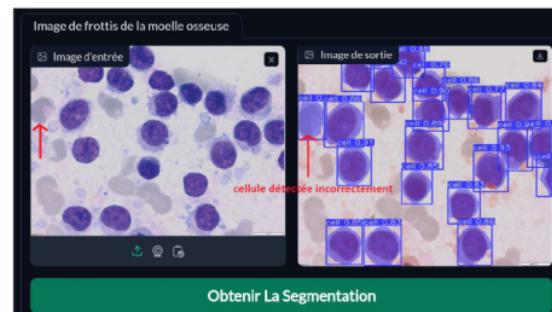


FIGURE 4.6 – Acquisition de l'image segmentatée à partir de l'interface

permet de rectifier de telles erreurs.

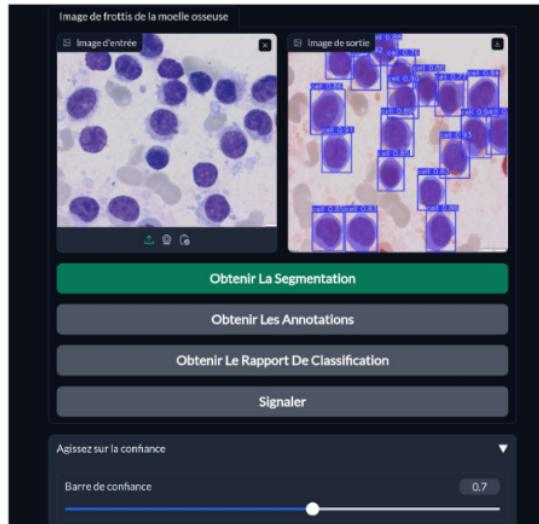


FIGURE 4.7 – Intégration de la barre de confiance dans l’interface Gradio

Après ajustement du score de confiance, la cellule mal identifiée a été corrigée, démontrant la capacité de l’utilisateur à influencer le processus de segmentation. la Figure 4.7 l’illustre.

L’image annotée (voir Fig. 4.8) présente une classification détaillée pour chaque classe de cellules détectées, offrant une vue précise de la segmentation.

Le rapport de classification fournit des informations détaillées sur le nombre d’instances par type de classe, permettant à l’utilisateur de mieux comprendre la répartition des classes dans l’image segmentée (voir Figure 4.9).

Chaque image de sortie peut être téléchargée par l’utilisateur pour une consultation en haute résolution. La Figure 4.10 illustre un rapport de classification exporté.

- **Login :** Notre application Gradio intègre la gestion des utilisateurs via Hugging Face Spaces, offrant une authentification robuste et des fonctionnalités de contrôle d’accès (Fig. 4.11). Les utilisateurs peuvent se connecter en toute sécurité avec leurs comptes Hugging Face, éliminant ainsi la nécessité de gérer manuellement les utilisateurs. Grâce à cette intégration, l’expérience utilisateur est sécurisée et fluide.

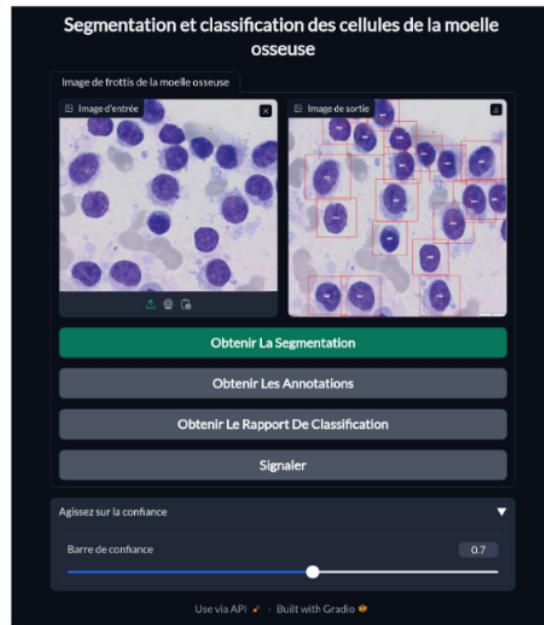


FIGURE 4.8 – Récupération de l'image annotée

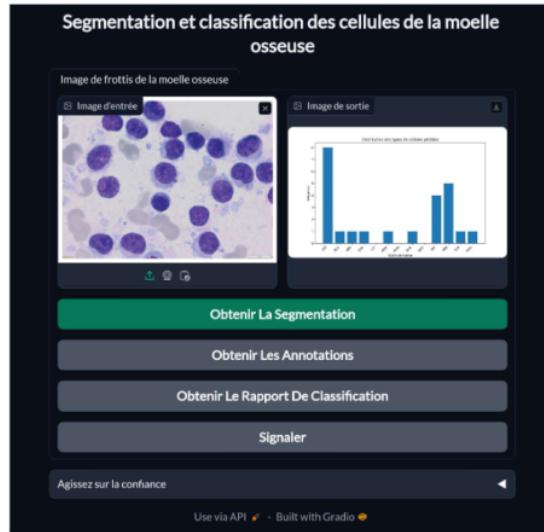


FIGURE 4.9 – Obtention du rapport de classification en utilisant le bouton dédié

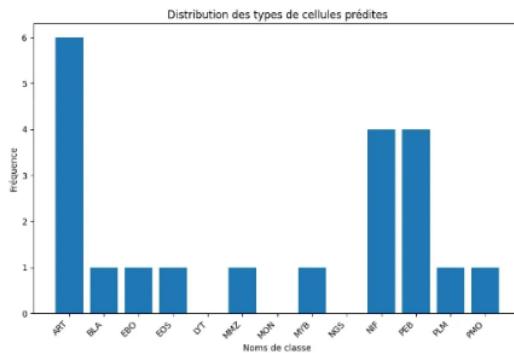


FIGURE 4.10 – Rapport de classification téléchargé depuis Gradio (Haute Résolution)

Hugging Face, intégré à Gradio, fournit une plateforme complète pour développer et déployer des modèles d'intelligence artificielle, y compris des fonctionnalités avancées telles que la gestion des utilisateurs et l'authentification pour les applications Gradio.

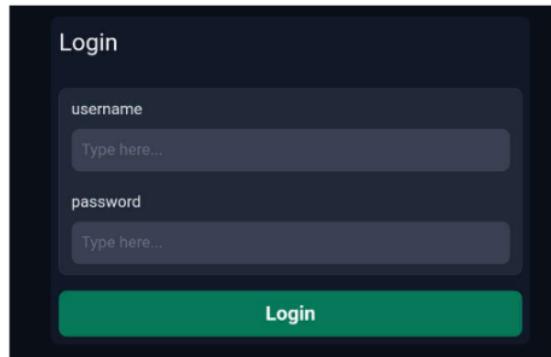


FIGURE 4.11 – Écran d'identification

En résumé, notre expérience avec Gradio a démontré que sa flexibilité facilite le déploiement de pipelines complexes, tels que la segmentation et la classification d'images. Malgré la sophistication des modèles utilisés, nous avons réussi à les exécuter sur des appareils mobiles, ce qui témoigne de leur portabilité. La convivialité de Gradio nous permet d'ajouter de nouvelles fonctionnalités en continu, assurant ainsi que notre application reste adaptable aux besoins changeants des utilisateurs. L'intégration de la barre de confiance a également amélioré l'expérience utilisateur en offrant une interaction plus fluide et in-

tuitive. En choisissant cette méthode de déploiement en ligne, nous rejoignons la tendance actuelle en faveur de la simplicité et de l'accessibilité dans les applications d'apprentissage automatique.

4.5 Interprétation des résultats trouvées

Malgré les défis posés par la taille restreinte de notre équipe, composée de seulement deux individus, nous avons réussi à intégrer avec succès les deux modèles via un pipeline et à les déployer. Par conséquent, nous avons réalisé le déploiement d'une solution qui produit des résultats respectables.

4.6 Défis Rencontrés

- ● **Problèmes de compatibilité, notamment avec TensorFlow :** Nous avons rencontré des difficultés de compatibilité, en particulier avec TensorFlow. Certains des modèles que nous avons expérimentés n'étaient pas intégrables dans notre pipeline en raison de problèmes de compatibilité. Ces problèmes se sont manifestés sur différentes plateformes telles que Kaggle, Google Colab et notre machine locale. Parfois, nous étions contraints de travailler avec un modèle spécifique dans un environnement particulier, par exemple VGG-16, en raison de sa lourdeur et de son besoin d'un GPU rapide.
- ● **Limitations de l'utilisation du GPU sur Kaggle et Google Colab :** Les plates-formes Kaggle et Google Colab offrent l'utilisation du GPU pour un temps limité et proposent des plans payants. Cette limitation a eu un impact sur notre capacité à expérimenter et à entraîner des modèles, car nous devions travailler dans des délais stricts et parfois ajuster notre approche en fonction de ces contraintes.
- ● **Absence de jeux de données pré-annotés pour les cellules sanguines :** Nous avons également été confrontés au défi de l'absence de jeux de données pré-annotés pour les cellules sanguines. Cette lacune nous a obligés à consacrer du temps et des ressources supplémentaires à l'annotation manuelle des données, ce qui a retardé le processus d'entraînement des modèles.
- ● **Exploration de différentes approches et modèles complexes :** Nous avons consacré du temps à l'exploration de différentes approches et modèles qui présentent des architectures complexes. Cette exploration nous a demandé de nous familiariser avec des concepts avancés et a parfois conduit à des impasses lors de l'adaptation de ces modèles à notre pipeline.

- ● **Limitations de l'hébergement gratuit de Gradio :** En ce qui concerne la phase de déploiement, nous avons utilisé Gradio pour héberger notre application. Cependant, nous avons rencontré des limitations liées à l'hébergement gratuit, qui offre une durée limitée de seulement 72 heures. Cette contrainte a rendu notre application moins accessible à long terme, ce qui a nécessité des ajustements dans notre stratégie de déploiement.
- ● **Absence de frottis de MO complets :** Un défi supplémentaire auquel nous avons été confrontés est l'absence de frottis de moelle osseuse complets disponibles pour travailler. Nous avions envisagé d'implémenter un mécanisme permettant à notre système de balayer un frottis complet et de le parcourir, détectant et classant les cellules, afin de fournir une analyse quantitative de la moelle osseuse complète d'une personne. Cela aurait permis de fournir des informations sur des maladies telles que la leucémie et d'autres pathologies hématologiques. Malheureusement, l'absence de telles ressources nous a limités dans notre capacité à explorer cette approche et à développer un système de diagnostic plus complet. Habituellement, de telles initiatives sont menées en collaboration avec de grandes entreprises d'intelligence artificielle et des centres d'analyse médicale en biologie, qui disposent des ressources et des données nécessaires pour entreprendre de telles études.

4.7 Conclusion

Dans ce chapitre, nous avons présenté en détail le processus de test et de validation de notre pipeline d'apprentissage automatique. Nous avons décrit l'environnement de travail utilisé, évalué les performances des modèles YOLOv8 et MobileNet, et examiné les défis rencontrés lors de l'intégration et du déploiement de ces modèles. En combinant ces analyses, nous avons démontré l'efficacité et la robustesse de notre approche dans la segmentation des images de frottis de MO.

Conclusion et perspectives

Dans un monde en constante évolution, les avancées technologiques, notamment dans le domaine de l'intelligence artificielle et de l'apprentissage automatique, transforment de manière significative notre quotidien, y compris dans le secteur de la médecine. Cependant, ces progrès sont souvent le résultat d'efforts colossaux déployés par de grandes entreprises, avec des ressources considérables et des équipes de recherche étendues. Pour les équipes de développement plus modestes, l'accès à ce domaine peut sembler décourageant en raison des barrières à l'entrée élevées, des ressources limitées et de la complexité des collaborations avec les institutions médicales. En conséquence, il est difficile pour les acteurs plus petits de rivaliser sur un pied d'égalité avec les grandes entreprises.

Pourtant, malgré ces obstacles apparents, il est important de reconnaître que chaque grande percée commence par un premier pas. Face aux défis du domaine médical et à la domination des grandes entreprises disposant de budgets colossaux et d'équipes de grande envergure, nous n'avons pas baissé les bras. Nous avons choisi, ne serait-ce qu'un peu, de contribuer à ce domaine crucial. Nous avons ainsi entrepris de développer le meilleur modèle possible avec les ressources dont nous disposions.

Dans ce rapport, nous avons d'abord présenté un aperçu du projet, en spécifiant la problématique, l'objectif ainsi que la solution proposée. Ensuite, nous avons défini les termes techniques nécessaires au déploiement de tout modèle de détection d'objets. Par la suite, dans le deuxième chapitre, nous avons explicité les interactions entre le système et ses utilisateurs à l'aide de quelques diagrammes. Dans le troisième chapitre, nous avons expliqué chaque étape nécessaire pour développer notre classifieur de frottis de MO. Enfin, nous avons présenté notre environnement de travail ainsi que les tests que nous avons effectués pour choisir l'algorithme de segmentation et de classification qui correspond le mieux aux exigences de notre projet. En conclusion, nous avons mentionné quelques défis rencontrés lors de la réalisation de ce projet.

Cependant, il reste toujours place à l'amélioration. Pour les recherches futures, il serait intéressant d'entraîner le modèle sur des ensembles de données plus importants et sur davantage d'images d'étalement. De plus, les modèles Vision Transformers semblent prometteurs et nous avons hâte de pouvoir éventuellement les utiliser.

Pour repousser encore plus les limites, un modèle capable de reconnaître et de produire plusieurs images d'étalement segmentées et annotées simultanément serait une idée stimulante mais intéressante.

Rapport_PCD_Aissa_Dinari

RAPPORT DE SIMILITUDE



CORRESPOND À TOUTES LES SOURCES (SEULE LA SOURCE SÉLECTIONNÉE EST IMPRIMÉE)

3%

★ Submitted to University of Carthage

Copie de l'étudiant

Exclude les citations Arret

Exclude la Arret

bibliographie

Exclude les Arret

correspondances