Rapport - Projet de Programmation

Détection automatique de pigments par modèle d'apprentissage profond

DUMERC Alex
LACHAUD Lucien
GRENIER Florian
GARNIER Anatole
NAIT SADI Samir

21 octobre 2024

Encadrants : Marie Beurton-Aimar, Bruno Dutailly M1 informatique - 2nd semestre 2023-2024

Table des matières

Remerciements

Pour commencer, nous tenons à exprimer notre profonde gratitude à nos encadrants, **Marie Beurton-Aimar** et **Bruno Du-tailly**, pour leur accompagnement et leurs conseils précieux tout au long de ce projet. Leur expertise et leur soutien ont permis de mieux appréhender les enjeux du projet et d'explorer avec efficacité le domaine du machine learning, qui était jusqu'alors relativement nouveau pour nous.

Introduction

Le temple d'Apollon à Delphes est l'un des sites archéologiques les plus emblématiques de la Grèce antique, qui continue de révéler ses secrets grâce aux avancées technologiques contemporaines. Au cours d'une récente campagne de numérisation par photogrammétrie, des traces de pigments autrefois imperceptibles à l'œil nu ont été découvertes sur les blocs qui constituent le plafond du temple. Cependant, la détection automatique de ces traces est entravée par la présence de faux positifs, tels que des lichens, des moisissures et la pollution environnementale.

Face à ce défi, dans un contexte marqué par l'essor des modèles d'apprentissage automatique, l'utilisation de techniques de segmentation d'image apparaît comme une approche essentielle et incontournable. Ces techniques offrent en effet la possibilité de détecter de manière fiable et précise les subtiles traces de pigments, tout en minimisant les faux positifs.

Le projet sera présenté autour de plusieurs étapes cruciales. Dans un premier temps, on définira clairement les objectifs attendus, visant notamment à développer un modèle de segmentation d'image efficace pour détecter les traces de pigments.

Ensuite, on s'appuiera sur l'état perçu du projet précédent pour identifier ce qui a été implémenté auparavant, les pistes d'amélioration et les points à renforcer que l'on érigera dans une partie d'analyse des besoins.

Puis, on se concentrera sur la recherche et l'implémentation de plusieurs modèles d'apprentissage automatique pour la segmentation d'image. Cette étape comprendra l'évaluation de diverses architectures de réseaux neuronaux, telles que U-Net, MSU-Net, DeepLab, et Mask R-CNN, parmi d'autres, pour déterminer celle qui convient le mieux au contexte spécifique de détection des traces de pigments sur les blocs du temple d'Apollon à Delphes.

Chaque modèle sera entraîné, testé et évalué sur un jeu de données généré. Les évaluations des performances seront mises en avant par les mesures de la matrice de confusion, la précision et les courbes de perte.

Suite à l'évaluation des différents modèles, une comparaison approfondie de leurs performances sera réalisée. Cela permettra de déterminer le modèle le plus adapté aux besoins du projet en termes de précision, de sensibilité à la détection des traces de pigments et de réduction des faux positifs.

Vous pouvez à partir de ce lien Github consulter l'ensemble du projet :

Lien Github : https://gitlab.emi.u-bordeaux.fr/lulachaud/unettraining_temple.git

Chapitre 1

Analyse

Ce chapitre d'introduction offre l'opportunité d'examiner le sujet en profondeur, de clarifier nos objectifs et d'évaluer les ressources disponibles. À partir de cette analyse, un plan d'action clair sera établi pour entamer ce projet.

1.1 Contexte

Pour contextualiser, le projet a commencé l'année précédente par d'autres élèves. Ils ont laissé un projet avec de bonnes idées, mais des résultats peu concluants. Le défi est donc de mettre en œuvre des démarches pour comprendre et améliorer les résultats proposés par ces élèves.

1.2 Cadre de travail

Ce projet bénéficie de l'encadrement de deux entités d'expertise renommées : Le LaBRI [1] et Archéovision [2] .

Le LaBRI (Laboratoire Bordelais de Recherche en Informatique), affilié à l'Université de Bordeaux et au CNRS, est un laboratoire qui mène des recherches dans divers domaines de l'informatique, y compris l'intelligence artificielle, la bio-informatique, les systèmes distribués, la sécurité informatique.

Archéovision est une structure spécialisée dans l'ingénierie et la recherche en archéologie numérique, elle propose des solutions innovantes pour la numérisation, la modélisation 3D et la visualisation du patrimoine archéologique.

Ce projet sera accompagné tout le long par Marie Beurton-Aimar, chercheuse au LaBRI, ainsi que par Bruno Dutailly, membre de l'équipe d'Archéovision. M. Dutailly est renommé pour son expertise en archéologie et en vision par ordinateur, particulièrement dans le contexte de l'archéologie. Leur accompagnement est indispensable pour orienter et mener à bien notre projet.

1.3 Objectif

L'objectif principal de ce projet est de fournir des outils de prédiction pour la reconstitution précise du temple d'Apollon à Delphes dans son état d'origine. Pour ce faire, le projet se concentrera sur la détection des pigments imperceptibles par des apprentissages automatiques, notamment ceux présents sur les blocs du plafond du temple. Ces pigments fournissent des indications cruciales sur la distribution des blocs à l'époque de la construction du temple, ce qui est essentiel pour une reconstitution précise de sa structure.

Dans ce contexte, notre objectif est de comparer et d'implémenter différents modèles de réseaux de neurones pour la segmentation d'image, afin de trouver le modèle le plus adapté à la détection des traces de pigments. Il vise à développer un modèle efficace qui permettra une détection fiable et précise des pigments tout en minimisant les faux positifs. Pour ce faire, un certain seuil de validation est défini par les archéologues et ne doit pas être dépassé, en garantissant que chaque modèle produise le moins de faux positifs possible.

1. `https://www.labri.fr/`
2. `https://archeovision.cnrs.fr/`

1.4 Différentes segmentations

Les modèles d'apprentissage profond qui seront implémentés utilisent différentes techniques de segmentation d'image afin de localiser les pixels imperceptibles.

La segmentation d'image est une technique de vision par ordinateur qui divise une image numérique en groupes de pixels distincts, également appelés patchs. Cette technique est utilisée pour aider à détecter des objets dans les images et à effectuer d'autres tâches liées à la vision par ordinateur. Grâce à la segmentation d'image, on peut interpréter et analyser des images de manière plus efficace, ce qui trouve des applications dans de nombreux domaines, de la médecine à la conduite autonome.

La segmentation d'image peut être réalisée selon deux principales approches : la segmentation sémantique et la segmentation d'instance.

1.4.1 Approche de segmentation par sémantique

La segmentation sémantique attribue une classe sémantique à chaque pixel de l'image, sans distinction entre objets individuels ou ensembles. Elle traite tous les pixels comme des ensembles et ne différencie pas les instances multiples d'une même classe.

FIGURE 1.1 – Illustration de la segmentation sémantique

1.4.2 Approche de segmentation par instance

La segmentation d'instance vise à isoler chaque instance d'objet distinct dans une image. Elle génère des masques de segmentation précis pour chaque objet, permettant ainsi de les différencier même lorsqu'ils se touchent ou se chevauchent. Par exemple, dans la même image de rue, la segmentation d'instance serait capable de distinguer chaque voiture individuelle et de générer un masque de segmentation spécifique pour chacune. Dans l'exemple si dessous, elle est capable de détacher plusieurs personnes d'un groupe.

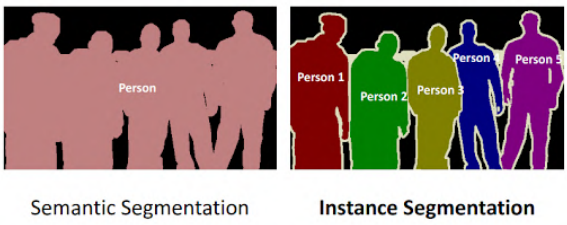


FIGURE 1.2 – différence entre la ségmentation sémantique et la ségmentation par instance

1.5 Etat de l'art

Dans cette section, les ressources mises à notre disposition depuis le début de notre projet seront examinés. Il est important de rappeler que le projet n'a pas démarré à partir de zéro, puisqu'il est repris d'un travail entamé l'année précédente. Par conséquent, il existe un ensemble d'outils existants qui ont aidés à progresser. L'objectif ici est de dresser une liste exhaustive des outils disponibles pour le projet.

1.5.1 Étiqueteuse pour la vérité terrain

L'étiquetage de la vérité terrain est une étape cruciale pour entraîner et évaluer efficacement notre modèle de *machine learning*. Cette étape implique la confirmation manuelle de la présence de

pigments bleus sur les images fournies. L'étiqueteur fournit des outils précis permettant d'annoter avec précision les emplacements de ces pigments sur les photos. Cette annotation manuelle garantit que le modèle dispose de données fiables et précises pour apprendre à détecter les pigments bleus dans les images.

1.5.2 Générateur du jeu de données

Le programme comporte un générateur de données, ou plutôt un splitter d'images. Ce module prend en entrée un jeu d'images ainsi que le jeu de masques comportant la vérité terrain (format JSON). Il effectue des découpes (patchs) d'une taille personnalisable afin d'obtenir plusieurs images et masques à partir d'une.

1.5.3 Modèle d'apprentissage profond

Modèle d'apprentissage par segmentation sémantique

Ce modèle n'est pas implémenté dans l'existant mais apparaît dans le compte rendu des développeurs de l'application. Il s'agit du modèle U-net.

Modèle d'apprentissage par segmentation d'instance

Ce modèle est quant à lui implémenté. Le modèle Mask R-CNN est une architecture de réseau de neurones convolutif. Ce modèle a pour principe de détecter des boîtes d'objets dans les images et affiner dans un deuxième temps ses intuitions grâce aux opérations *RoiPool* (Region of Interest Pooling).

1.5.4 Fonctions de Perte

Pour espérer obtenir de bons résultats et aiguillée le modèle sur la détection des pigments, il existe de nombreuses fonctions de perte adéquate au problème. Elles permettent d'encourager le modèle à améliorer l'*overlap* entre les zones prédites et les zones réelles, essentiel pour obtenir des résultats de segmentation de haute précision.

1.5.5 Évaluation des modèles

Une fonction d'évaluation est présente dans le projet permettant la création d'un fichier de compte rendu avec un graphe évaluant l'évolution de la loss ainsi qu'une matrice de confusion utile pour interpréter la précision du modèle actuel.

1.6 Besoin fonctionnel

Dans cette partie, on va analyser ce dont on aura besoin pour notre projet. La constatation de l'existant et la considération de toutes les possibilités qu'il apporte au projet, on était faites. On va désormais lister ce qui permettra de répondre à l'objectif que l'on s'est fixé. Pour clarifier, un besoin fonctionnel définit les actions spécifiques que le système doit être capable d'exécuter pour répondre aux attentes des utilisateurs. Il s'agit essentiellement de ce que le système doit faire, comme les fonctionnalités et les comportements directement liés aux tâches qu'il doit accomplir.

1.6.1 Générer le jeu de données

Une fois que les couples (image-masque) ont été réalisés, le générateur de jeu de données doit constituer un ensemble de couples en fonction d'un pourcentage de couples positifs (couples où un pigment est présent dans la sous-image). Ce pourcentage devra être passé en paramètre, pour que l'utilisateur puisse le choisir. L'utilisateur spécifiera aussi le dossier de destination où seront placées les sous-images et les masques en fin de traitement.

1.6.2 Entraîner un modèle

Le programme doit permettre à l'utilisateur d'entraîner le modèle avec les images et les paramètres de son choix. Les données pourront au préalable être formatées avec le module de génération de jeu de données et les paramètres sont le nombre d'époques (epochs) et la taille des lots (batch

size). Le modèle ainsi entraîné devra être enregistré sur le disque à un emplacement décidé par l'utilisateur.

1.6.3 Augmentation des données

Le programme doit permettre d'augmenter de manière efficace et automatisée un ensemble de données d'images et de masques en appliquant diverses transformations telles que la rotation, le retournement horizontal, la propagation de couleur et l'ajustement de la chaleur des images.

1.6.4 Évaluation de la similarité des masques

On doit pouvoir évaluer et mesurer la précision de nos modèles d'apprentissage, donc connaître la similarité entre les masques prédits et les masques de références.

1.6.5 Création de la fonction objectif

Le programme doit pouvoir utiliser une fonction objective adaptée, telle que la fonction de perte Dice. En utilisant la fonction de perte Dice, le programme doit évaluer la similitude entre les masques prédits par le modèle et les masques de référence.

1.6.6 Évaluation des performances des modèles

Le programme doit pouvoir évaluer les performances des modèles sur un ensemble de données de test afin de mesurer les précisions, leurs rappels et d'autres métriques de performance.

La matrice de confusion consiste à évaluer la performance d'un modèle de segmentation d'image en comparant les prédictions du modèle avec les annotations réelles, permettant ainsi d'analyser les résultats obtenus en termes de vrais positifs, faux positifs, vrais négatifs et faux négatifs.

1.6.7 Création de la vérité terrain pour les nouvelles photos

La vérité terrain implique, à partir de nouvelles photos, à examiner attentivement les photos et identifier les zones contenant des pigments bleus, puis utiliser les outils d'annotation appropriés pour marquer précisément les emplacements des pigments bleus, et comparer visuellement les annotations avec la photo originale pour vérifier la qualité des images.

1.6.8 Analyse du voisinage RGB

Examiner les pixels voisins pour chaque pixel détecté afin de vérifier s'ils répondent à certaines conditions basées sur la couleur.

1.6.9 Comparaison de la performance entre les modèles d'apprentissage

Évaluer et comparer la performance des différents modèles d'apprentissage utilisés, en analysant leurs résultats de segmentation sur un ensemble de données spécifique.

1.6.10 Suppression des masques noirs

L'utilisateur doit avoir la possibilité de supprimer un pourcentage spécifié de masques entièrement noirs afin de nettoyer l'ensemble de ses données.

1.6.11 Éroder les masques d'images

Le programme doit pouvoir appliquer une opération d'érosion aux masques d'images pour affiner les contours des composantes connexes.

1.6.12 Visualisation des résultats

L'utilisateur doit disposer de fonctionnalités permettant de visualiser les comparaisons entre les masques réels, les masques prédits et les images originales. De plus, il doit avoir la possibilité d'afficher la matrice de confusion pour évaluer les performances du modèle de segmentation.

1.7 Besoin non fonctionnel

Un besoin non fonctionnel, souvent aussi appelé exigence non fonctionnelle, désigne une caractéristique ou une condition que doit respecter un système pour garantir sa fiabilité, sa facilité d'utilisation, son efficacité ou tout autre attribut qualitatif essentiel à son fonctionnement, en dehors des tâches spécifiques qu'il doit accomplir. Ces besoins spécifient comment le système doit se comporter et ils se distinguent des besoins fonctionnels qui définissent ce que le système doit faire.

1.7.1 Disponibilité

Il est nécessaire que le système soit opérationnel et accessible lorsque l'utilisateur à le code devant lui. Ainsi, le souhait est de garantir que le système soit accessible via un terminal pour toute la durée de son utilisation.

1.7.2 Capacité

Le système doit être capable de gérer une charge de travail spécifique déterminée par le système du client. Le système sera exécuté sur une machine personnelle, sans hébergement central, ce qui nécessite que notre code soit optimisé pour fonctionner efficacement sur des configurations variées.

1.7.3 Environnemental

Le projet doit minimiser son impact environnemental, en dépit des exigences élevées en performances de calcul. Il est essentiel d'optimiser la consommation d'énergie tout en atteignant les objectifs de performance, même si cette dimension n'est pas la priorité immédiate en raison des contraintes de délai.

1.7.4 Maintenabilité

Il est crucial que le système soit facile à maintenir, à modifier ou à étendre. La maintenabilité est amélioré par une organisation claire des paquets et une meilleure lisibilité du code, adressant ainsi les défauts de lisibilité du code hérité et préparant le système pour une maintenance future aisée.

1.7.5 Recouvrabilité

Bien que le code ne soit pas hébergé, ce qui réduit les risques de problèmes serveur, le système doit pouvoir se récupérer rapidement de toute défaillance logicielle. Il est impératif de mettre en place des mécanismes permettant une restauration rapide et efficace en cas de problème.

1.7.6 Évolutivité

L'évolutivité, bien que n'étant pas une priorité absolue pour notre projet, reste un aspect qu'il faut envisager d'explorer, notamment la possibilité d'implémenter CUDA pour décharger le processeur en cas de nécessité. Cette approche pourrait offrir une gestion plus efficace des ressources systèmes, adaptant ainsi notre application aux variations potentielles de charge.

1.7.7 Fiabilité

La fiabilité constitue un aspect crucial de notre projet, et il sera prévu de l'assurer par le biais de plusieurs méthodes. Tout d'abord, l'utilisation de techniques telles que la validation croisée, qui permettra de garantir la stabilité et la précision de nos résultats. De plus, une généralisation efficace sera mise en œuvre afin de répondre aux exigences spécifiques de notre projet. En combinant ces approches, nous visons à construire un système robuste et fiable, capable de produire des résultats cohérents et précis dans diverses situations.

1.7.8 Sécurité

La sécurité, bien que souvent critique dans de nombreux projets, n'est pas une priorité dans le cadre de notre initiative. Le programme est conçu pour être utilisé par un utilisateur qui gère et traite ses propres images en local. Comme l'accès au logiciel est limité à l'utilisateur unique et que celui-ci traite uniquement ses propres données, il n'y a pas de nécessité de mettre en œuvre des mesures de sécurité supplémentaires. Cette spécificité permet de concentrer les ressources sur d'autres aspects critiques du projet, sans devoir renforcer inutilement la sécurité du logiciel.

1.7.9 Utilisabilité

En définitive, notre produit s'adresse à un public averti, familiarisé avec l'utilisation de logiciels similaires. Toutefois, pour faciliter sa prise en main, il y aura un fichier README inclus, qui sera détaillé et qui expliquera comment générer les masques indiquant les pigments bleus, offrant ainsi une aide supplémentaire en cas de besoin. Cependant, il est important de noter que notre interface utilisateur restera basique, sous forme de terminal brut, ce qui peut présenter une courbe d'apprentissage plus abrupte pour les utilisateurs non-initiés.

1.8 Outils choisis

Python

Python est un langage de programmation utilisé pour le développement de logiciels et d'applications. Il est largement utilisé pour sa simplicité et sa polyvalence, avec des bibliothèques étendues. Python est facilement intégrable dans notre projet.

PyTorch

PyTorch est une bibliothèque open source de machine learning utilisée pour le développement de modèles d'apprentissage automatique en utilisant des réseaux de neurones profonds. PyTorch offre une flexibilité et une facilité d'utilisation, en particulier pour la recherche en apprentissage profond.

Numpy

NumPy est une bibliothèque en Python utilisée pour le traitement et l'analyse des données numériques. PyTorch s'intègre parfaitement avec NumPy. De nombreuses opérations sur les tableaux NumPy peuvent être utilisées directement avec les tenseurs PyTorch, facilitant ainsi la manipulation des données pendant l'entraînement et l'évaluation du modèle. NumPy est optimisé pour les calculs vectoriels et matriciels, ce qui est important pour les opérations intensives en calculs lors de l'entraînement et de l'évaluation des modèles

OpenCV

OpenCV est une bibliothèque open source utilisé pour le traitement d'images et la vision par ordinateur. OpenCV permet de lire des images à partir de fichiers sur le disque, ainsi que d'écrire des images dans différents formats de fichier.Il effectue des opérations de traitement d'images telles que la conversion des couleurs, le redimensionnement, la mise à l'échelle, la rotation, le flou. Il est aussi utilisé pour appliquer une opération d'érosion à une image en niveaux de gris.

matplotlib

matplotlib est une bibliothèque de visualisation de données en Python utilisée pour créer des graphiques et des visualisations interactives de haute qualité. Elle offre des fonctionnalités étendues pour la génération de graphiques en deux dimensions, y compris des graphiques linéaires, des histogrammes, des nuages de points, et des images. Sa capacité à intégrer avec des bibliothèques de calcul numérique comme NumPy la rend indispensable pour la visualisation des résultats en machine learning.

Pillow

Pillow, la bibliothèque Python Imaging Library (PIL Fork), permet la manipulation et le traitement d'images dans divers formats. Elle est utilisée pour la lecture, l'écriture et le traitement d'images, permettant des opérations telles que le redimensionnement, la rotation, et l'ajustement des couleurs, ce qui est crucial pour la préparation et l'augmentation des ensembles de données d'image en apprentissage automatique.

tqdm

tqdm est une bibliothèque qui permet d'afficher des barres de progression dynamiques dans les boucles. Elle est utilisée pour fournir une indication visuelle de la progression des processus longs comme l'entraînement de modèles ou le traitement de grandes ensembles de données, contribuant à une meilleure gestion du temps et à l'efficacité du développement.

wandb

wandb, ou Weights & Biases, est une plateforme pour le suivi des expériences en *machine learning*. Elle permet de suivre automatiquement tous les aspects de l'entraînement d'un modèle, y compris les hyperparamètres, les métriques, les sorties, et même les médias associés. Cette intégration est essentielle pour le débogage, l'optimisation des modèles, et la reproduction des résultats.

imgaug

imgaug est une bibliothèque puissante pour l'augmentation des images en *machine learning*. Elle permet de transformer les images de manière aléatoire et réaliste, ce qui augmente la diversité des données sans nécessiter de collecte supplémentaire. Cette diversité est cruciale pour la généralisation et l'efficacité des modèles d'apprentissage profond.

Chapitre 2

Conception

Pour atteindre notre objectif, divers modèles de segmentation ont été développés, avec leurs avantages et leurs limites. Dans cette étude, l'illustration de plusieurs architectures de réseaux de neurones supervisés sera présenté, ce qui permettra, à terme, de pouvoir sélectionner le modèle le plus appropriés pour ce sujet.

2.1 Les modèles de réseaux profonds

2.1.1 Modèle MaskRCNN

Pour atteindre leur objectif, le projet précédent avait choisi d'utiliser le modèle MaskRCNN [4]. Le fonctionnement de Mask R-CNN repose sur une architecture de réseau de neurones convolutifs (CNN) conçue pour la segmentation d'instance.

Tout d'abord, il utilise un réseau de neurones convolutifs pré-entraîné, comme ResNet, pour extraire des caractéristiques visuelles de l'image d'entrée. Ensuite, un réseau de propositions de régions identifie les zones de l'image susceptibles de contenir des objets, en utilisant des boîtes englobantes. Ces régions d'intérêt sont ensuite examinées de plus près pour extraire des informations spécifiques à chaque objet.

Le modèle comprend également des parties spécialisées pour détecter et segmenter les objets. La partie de détection prédit les classes des objets détectés ainsi que les boîtes englobantes qui les entourent, tandis que la partie de segmentation prédit des masques pixel par pixel pour chaque objet détecté. Ces masques indiquent où se trouvent les objets dans l'image si dessous.

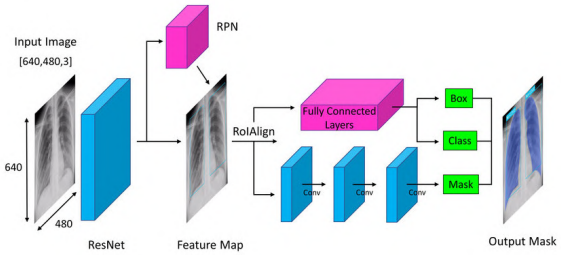


FIGURE 2.1 – Illustration du modèle MaskRCNN

2.1.2 Modèle DeepLabV3

Le premier modèle implémenté dans ce projet est le modèle DeepLabV3 [1]. Ce modèle de segmentation sémantique est largement utilisé pour la segmentation précise des objets dans les images, et il est particulièrement réputé pour sa capacité à segmenter des objets de petite taille avec une grande précision. Cette spécificité a conduit à examiner l'adéquation de DeepLabV3 avec les besoins de notre projet. En effet, la détection d'objets de petite taille est essentielle pour identifier les pigments bleus, qui sont souvent de taille minime par rapport aux dimensions globales des images utilisées en entrée. L'emploi de DeepLabV3 a été motivé par sa renommée dans le domaine de la segmentation sémantique et par sa performance confirmée dans la détection de détails fins, alignant

ainsi ses capacités avec les exigences spécifiques de notre projet pour une détection efficace et précise des pigments.

DeepLabV3 utilise des réseaux de neurones convolutifs profonds pour segmenter les images en attribuant une classe à chaque pixel. Il utilise des réseaux de neurones convolutifs profonds, tels que ResNet-101, pour extraire des caractéristiques de bas niveau de l'image.

La dernière couche de classification quand à elle est remplacée par une convolution 2D avec un seul canal de sortie, adaptée à la segmentation binaire. Pour obtenir des prédictions binaires pour chaque pixel de l'image, le modèle applique une fonction d'activation sigmoïde sur la sortie. En utilisant des convolutions à différents taux de dilatation, DeepLabV3 capture des informations contextuelles à différentes échelles, ce qui améliore la précision de la segmentation.

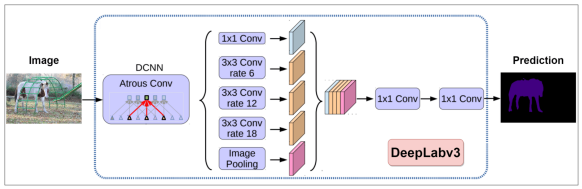


FIGURE 2.2 – Illustration du modèle DeepLabV3

2.1.3 Modèle U-Net

L'illustration du prochain modèle porte sur le modèle U-Net [11]. Ce modèle est une architecture de réseau de neurones convolutionnels (CNN) reconnue dans le domaine de la vision par ordinateur, en particulier pour la segmentation sémantique des images. Reconnu pour ses performances dans la segmentation d'images médicales, il serait intéressant qu'il soit applicable à notre projet. Il se compose d'un chemin contractant (encodeur) et d'un chemin expansif (décodeur). L'encodeur réduit progressivement la résolution spatiale de l'image tout en augmentant le nombre de caractéristiques, tandis que le décodeur restaure la résolution spatiale tout en réduisant le nombre de caractéristiques.

Le chemin contractant consiste en l'application répétée de deux convolutions 3x3, chacune suivie d'une unité linéaire rectifiée (ReLU) . Cette opération permet de capturer les caractéristiques de bas niveau de l'image tout en réduisant sa résolution spatiale. Après chaque paire de convolutions, une opération de *max pooling* est généralement effectuée pour réduire la taille de l'image tout en préservant les caractéristiques importantes.

Le chemin expansif (ou décodeur) de U-Net utilise des opérations de déconvolution (ou up-convolution) pour restaurer la résolution spatiale de l'image tout en réduisant le nombre de caractéristiques. Contrairement au *max pooling* qui effectue une réduction, la déconvolution augmente la taille de l'image en insérant des zéros entre les pixels et en appliquant des filtres pour rétablir la résolution spatiale perdue pendant l'étape de contraction. Cela permet au décodeur de reconstruire l'image segmentée à partir des caractéristiques extraites par l'encodeur, en conservant les informations contextuelles importantes.

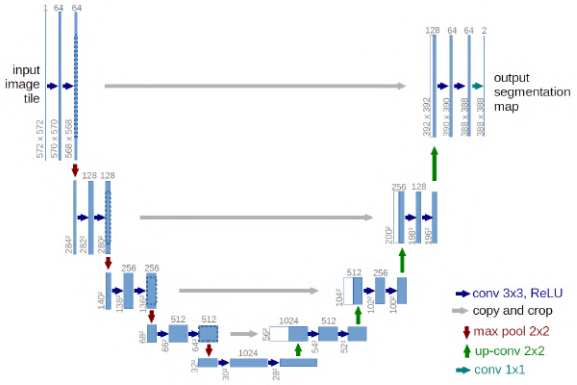


FIGURE 2.3 – Architecture du modèle U-Net

2.1.4 Modèle MSU-Net

Afin d'étoffer les recherches, il a été décidé d'implémenté le modèle MSU-Net [14]. De son nom, Multi-Scale U-Net est une architecture de réseau de neurones convolutionnels (CNN) conçu pour la segmentation sémantique d'image, précisément une segmentation par généralisation.

Variante de l'architecture U-Net, le modèle MSU-Net constitue des blocs multi-échelles composés de séquences de convolutions spécifiques (conv-3-1) au lieu de convolutions traditionnelles afin de capturer des informations à plusieurs échelles dans l'image, ce qui lui permet de mieux percevoir les détails fins et les structures de différentes tailles.

L'architecture de MSU-Net comporte des couches d'encodage pour capturer les caractéristiques de l'image à différentes échelles et des couches de décodage pour restaurer la résolution spatiale à l'aide d'opérations d'*upsampling*.

Les caractéristiques du modèle sont les suivantes. Il utilise des blocs de convolution spécifiques (conv-3-1) qui comprennent des convolutions de noyaux de taille 3x3 suivies de couches de normalisation et d'activation ReLU. Ces blocs sont efficaces pour extraire des caractéristiques à différentes échelles spatiales.

L'architecture utilise des opérations d'upsampling pour agrandir la résolution spatiale des cartes d'activation vers les niveaux supérieurs du réseau, permettant ainsi une reconstruction précise des détails spatiaux. La couche de sortie est une convolution 1x1 qui réduit le nombre de canaux à celui de la sortie souhaitée, généralement pour la segmentation binaire.

Le MSU-Net intègre des connexions résiduelles dans le chemin de l'encodeur vers le décodeur, facilitant ainsi le flux d'informations à différentes échelles spatiales et aidant à prévenir le problème de disparition des gradients.

Le nombre de filtres dans les couches convolutives varie progressivement, généralement en doublant ou en réduisant de moitié à chaque niveau. Cela permet de capturer des caractéristiques à différentes échelles spatiales tout en réduisant graduellement la résolution spatiale.

Les activations ReLU sont utilisées après chaque convolution pour introduire la non-linéarité dans le réseau et faciliter l'apprentissage de représentations complexes.

L'implémentation de ce réseau n'est pas anodine. En effet, il présente plusieurs avantages dans notre projet. Inspiré de l'architecture de U-Net, MSU-Net possède déjà tous les avantages que U-Net possède dans le domaine de la segmentation d'images. De plus, il capture efficacement les informations à différentes échelles spatiales dans l'image, ce qui lui permet de segmenter des ensembles de pixel de tailles variées avec précision. Il peut être utilisé et être efficace avec une segmentation sur de grandes images telles que nous avons, et facilement être modifié en testant différentes tailles de convolutions et différents nombres de couche pour convenir à nos attentes.

L'architecture du modèle MSU-Net est expliqué par le schéma suivant :

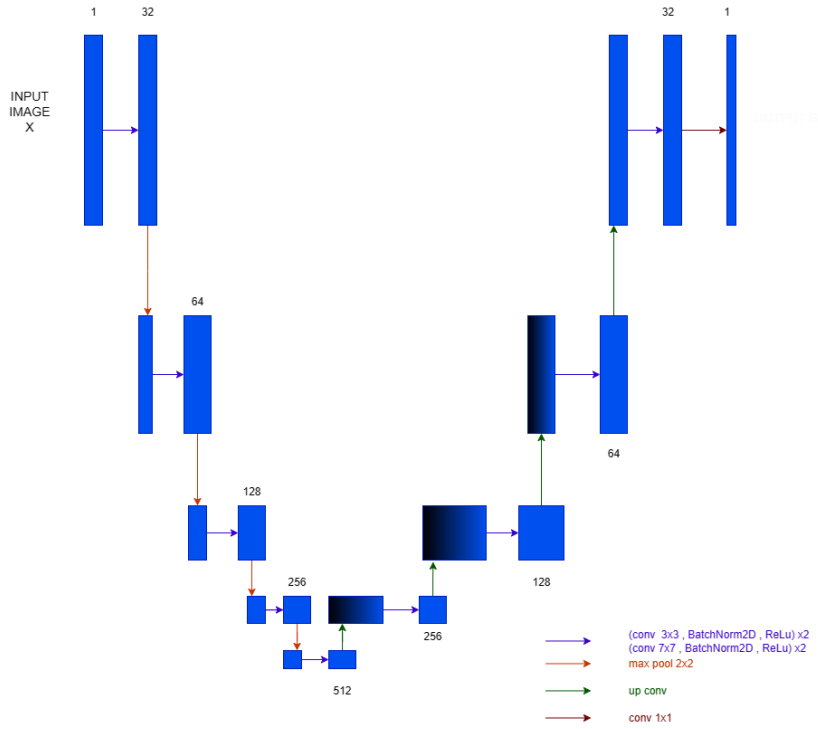


FIGURE 2.4 – Architecture du modèle MSU-Net

La figure suivante présente les différents blocs de convolution à multi-échelle avec différents kernels tels que 3x3 et 7x7

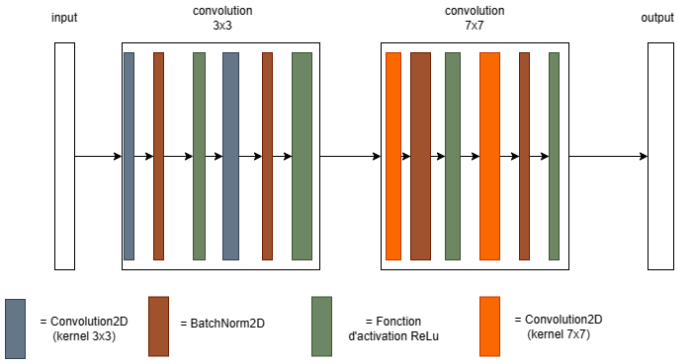
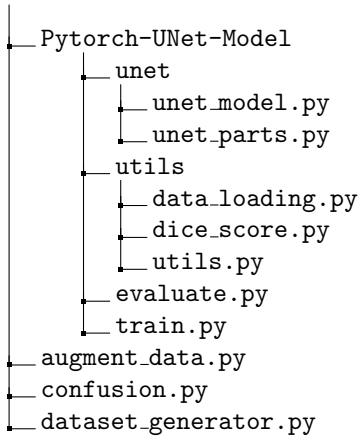


FIGURE 2.5 – Architecture des blocs de convolution

Les blocs de convolutions représentent l'effet des actions réalisés par les flèches violettes.

2.2 Structure Général

Pour assurer une gestion efficace du projet, il est divisé en plusieurs modules distincts. Dans cette optique, ci-dessous, il est présenté l'architecture générale du projet, sous forme d'une arborescence, détaillant les différents modules qui le composent.



2.3 Fonction objectif

La fonction objectif, plus communément appelé fonction de coût , est un outil mathématique ayant une grande influence dans le processus d'optimisation des modèles d'apprentissage supervisé. En effet, son rôle est de réaliser une indication précise sur la qualité des prédictions faites par le modèle et ainsi pouvoir ajuster les paramètres utilisés par le modèle. Afin de garantir la convergence vers des modèles optimaux et une prise de décision prédictive efficace, il est important de choisir la fonction la plus adéquate au problème.

2.3.1 Cross-Entropy Loss

La *Cross-Entropy Loss* [3] est une métrique couramment utilisée pour les tâches de classification, y compris la segmentation sémantique, où l'objectif est de minimiser la distance entre les distributions de probabilités prédites et les étiquettes réelles. Pour un pixel spécifique, la *cross-entropy loss* est définie comme suit :

$$L = -\sum_{c=1}^{C} y_{o,c} \log(p_{o,c})$$

où C est le nombre de classes, $y_{o,c}$ est un indicateur binaire (0 ou 1) si la classe c est la véritable classe pour l'observation o, et $p_{o,c}$ est la probabilité prédite que l'observation o appartient à la classe c. Cette fonction de perte pénalise les prédictions qui sont confiantes et incorrectes avec une pénalité plus élevée imposée pour un écart plus important entre la prédiction et la vérité terrain.

2.3.2 Dice Loss

La *Dice Loss* [9], ou perte de Sørensen-Dice, est une fonction de perte couramment employée dans les tâches de segmentation d'images, notamment lorsque les classes présentent un déséquilibre significatif. Cette mesure évalue la similarité entre les ensembles prédits par le modèle et les vérités terrain, ce qui la rend particulièrement pertinente pour les domaines où la précision de la segmentation est cruciale, comme en imagerie médicale.

Le coefficient de Dice, sur lequel se base cette fonction de perte, est défini par la formule suivante :

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.1}$$

où $|X \cap Y|$ représente le nombre de pixels correctement classés (vrais positifs), $|X|$ est le nombre total de pixels prédits comme appartenant à la classe d'intérêt, et $|Y|$ est le nombre total de pixels réellement appartenant à la classe dans la vérité terrain.

La *Dice Loss* est alors exprimée par l'équation :

$$\text{Dice Loss} = 1 - \frac{2\sum_{i=1}^{N} p_i g_i + \epsilon}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} g_i + \epsilon} \tag{2.2}$$

dans laquelle p_i et g_i indiquent les valeurs prédites et les étiquettes réelles pour les N pixels, respectivement. Le terme ϵ est une petite constante ajoutée pour éviter la division par zéro et garantir la stabilité numérique.

En optimisant cette fonction de perte, on encourage le modèle à améliorer l'overlap entre les zones prédites et les zones réelles, essentiel pour obtenir des résultats de segmentation de haute précision.

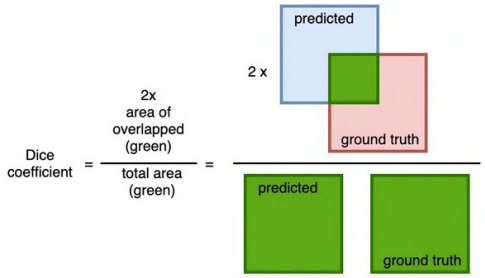


FIGURE 2.6 – Fonction Dice Loss

2.3.3 Tversky Loss

La *Tversky Loss* [16] est une adaptation de la *Dice Loss* qui permet un contrôle plus fin sur le rappel et la précision, ce qui est particulièrement utile dans les cas de déséquilibre des classes. Elle est définie comme suit :

$$L = 1 - \frac{\sum_{i=1}^{N} p_i g_i + \epsilon}{\sum_{i=1}^{N} p_i g_i + \alpha \sum_{i=1}^{N} p_i (1 - g_i) + \beta \sum_{i=1}^{N} (1 - p_i) g_i + \epsilon}$$

où p_i et g_i représentent les prédictions et les vérités terrain pour chaque pixel i, respectivement, α et β sont des paramètres qui contrôlent le compromis entre faux positifs et faux négatifs, et ϵ est un petit nombre ajouté pour éviter la division par zéro. Cette formulation permet aux chercheurs de mettre plus d'accent sur les faux positifs ou les faux négatifs selon les besoins du projet.

2.3.4 Jaccard Loss (IoU Loss)

La *Jaccard Loss* [5], également connue sous le nom d'*Intersection over Union (IoU) Loss*, est une fonction de perte basée sur l'évaluation de la similarité entre les ensembles prédits et les ensembles de vérités terrain. Elle est particulièrement utile pour les problèmes de segmentation sémantique en raison de sa robustesse face aux variations de taille des objets. La *Jaccard Loss* est définie comme suit :

$$L = 1 - \frac{\sum_{i=1}^{N} p_i g_i}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} g_i - \sum_{i=1}^{N} p_i g_i + \epsilon}$$

Ou de manière plus visuelle :

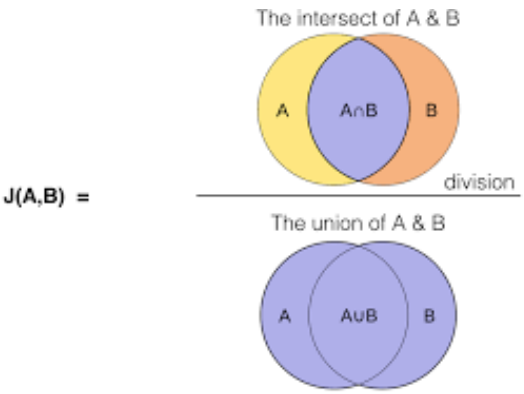


FIGURE 2.7 – Fonction Jaccard Loss

où p_i et g_i sont les prédictions et les vérités terrain pour chaque pixel i, et ϵ est ajouté pour la stabilité numérique. Cette métrique quantifie la proportion de l'intersection sur l'union des prédictions et des étiquettes réelles, fournissant une indication claire de la précision globale des prédictions.

2.4 Traitement de jeu de données pour la segmentation d'image

Il existe un processus essentiel afin de pouvoir entraîner et évaluer le modèle, celui de la manipulation et de la préparation des données. En effet, les modèles d'apprentissage ont besoin d'être entraînés sur des nombreux ensembles de données diversifiés.

Ainsi, le traitement des données intervient à plusieurs niveaux pour garantir la qualité et la pertinence de ces ensembles de données. Il s'agit de charger les données et d'effectuer des opérations de prétraitement pour nettoyer et normaliser les données brutes, éliminant ainsi les incohérences qui pourraient affecter négativement les performances des modèles. Il comprend également la génération de jeux de données équilibrés, garantissant une représentation adéquate de toutes les classes d'objets à segmenter.

2.4.1 Chargement et Pré-traitement des données

La phase de chargement et de pré-traitement des données dans le contexte de l'entraînement des modèles de segmentation d'images joue un rôle essentiel en préparant les données brutes pour l'apprentissage automatique. Ce processus commence par l'extraction des images à partir de fichiers spécifiés, en identifiant automatiquement le format approprié et en les chargeant en mémoire. En parallèle, une analyse des masques associés aux images est effectuée pour déterminer les valeurs uniques présentes, offrant ainsi un aperçu de la distribution des classes dans les données. En ce qui concerne le pré-traitement des données, des opérations standard, telles que le redimensionnement des images et des masques pour assurer une taille uniforme, ainsi que la normalisation des valeurs de pixel pour garantir une plage de valeurs cohérente, sont appliquées. Ces étapes visent à uniformiser et à optimiser les données en vue de l'entraînement des modèles.

2.4.2 Génération et Augmentation de jeux de données

La disponibilité de jeux de données de haute qualité et représentatifs est essentielle pour entraîner des modèles de machine learning performants. Cependant, dans de nombreux cas, les données réelles peuvent être limitées en quantité ou ne pas être parfaitement équilibrées, ce qui peut affecter les performances du modèle. Pour remédier à cela, diverses techniques de génération et d'augmentation de données peuvent être utilisées.

La génération de données implique la création de nouvelles instances de données synthétiques qui sont similaires aux données réelles mais qui augmentent la diversité du jeu de données. Cela peut être réalisé en utilisant des techniques telles que la génération de données basée sur des modèles probabilistes, la modification de données existantes, ou l'ajout de données synthétiques générées par des algorithmes de génération de données spécialisés.

L'augmentation de données, d'autre part, consiste à appliquer des transformations aux données existantes pour créer de nouvelles instances tout en préservant les étiquettes de classe d'origine. Ces transformations peuvent inclure la rotation, le redimensionnement, le recadrage, le changement de perspective, ou l'application de filtres. L'objectif de l'augmentation de données est d'augmenter la variabilité des données disponibles pour l'entraînement du modèle et d'améliorer sa robustesse aux variations dans les données réelles.

En combinant la génération et l'augmentation de données, il est possible d'enrichir considérablement un jeu de données et d'améliorer les performances du modèle en fournissant une représentation plus complète de l'espace des données. Cependant, il est important de noter que ces techniques doivent être utilisées avec précaution pour éviter le surajustement et garantir que les données générées sont fidèles à la distribution des données réelles.

2.5 Entraînement d'un modèle

L'entraînement d'un modèle de machine learning, en particulier dans le domaine du deep learning, est un processus itératif et intensif qui vise à optimiser les paramètres du modèle pour minimiser une fonction de perte donnée. Ce processus commence par la préparation des données, qui doivent être nettoyées, normalisées et parfois augmentées pour permettre au modèle d'apprendre efficacement à partir d'elles. Une fois les données prêtes, elles sont divisées en ensembles d'entraînement, de validation et de test.

Le cœur de l'entraînement implique de présenter les données d'entraînement au modèle, qui effectue des prédictions basées sur ses paramètres actuels. Ces prédictions sont ensuite comparées aux résultats attendus à l'aide de la fonction de perte. L'erreur calculée est utilisée pour ajuster les paramètres du modèle, généralement via un algorithme d'optimisation tel que la descente de gradient stochastique ou ses variantes (comme Adam ou RMSprop). Ce processus est répété sur plusieurs époques, chaque époque représentant une itération complète sur l'ensemble des données d'entraînement.

Tout au long de l'entraînement, la performance du modèle est régulièrement évaluée sur l'ensemble de validation pour surveiller les problèmes tels que le sur-apprentissage, où le modèle performe bien sûr les données d'entraînement mais pauvrement sur les données non vues (validation/test). Ajuster les hyperparamètres, appliquer des techniques comme la régularisation et le drop-out peut aider à contrôler le sur-apprentissage. Finalement, après plusieurs cycles d'entraînement et de validation, le modèle est testé sur l'ensemble de test pour évaluer sa performance globale, ce qui fournit une mesure de la qualité de l'apprentissage.

2.6 Évaluation d'un modèle

L'évaluation d'un modèle de *machine learning* est une étape critique pour déterminer sa performance et sa capacité à généraliser sur des données non vues. Cette étape implique généralement de soumettre le modèle à des données de test, qui ne sont pas utilisées pendant l'entraînement ou la validation, afin d'évaluer sa capacité à faire des prédictions précises sur de nouvelles instances.

Pour évaluer un modèle, différentes mesures de performance peuvent être utilisées en fonction de la nature du problème et des objectifs spécifiques. Parmi les mesures couramment utilisées, on trouve l'exactitude, la précision (accuracy), le rappel (recall), le score F1, l'aire sous la courbe ROC (AUC-ROC), et l'aire sous la courbe PR (AUC-PR). Ces mesures fournissent une vue d'ensemble de la capacité du modèle à classer correctement les instances dans différentes catégories, en tenant compte à la fois des faux positifs et des faux négatifs.

Les formules de précision, de rappel, et de F1-score sont des mesures essentielles dans l'évaluation des performances des modèles de classification, notamment dans le domaine de l'apprentissage automatique et de la vision par ordinateur.

La précision d'un modèle est définie comme le ratio des vrais positifs (TP), c'est-à-dire les éléments correctement identifiés par le modèle, par rapport à la somme des vrais positifs et des faux positifs (FP), qui représentent les éléments incorrectement identifiés comme faisant partie de la classe d'intérêt. Ainsi, la précision mesure la proportion d'éléments prédits comme appartenant à la classe d'intérêt qui le sont effectivement :

$$\text{Précision} = \frac{TP}{TP + FP} \tag{2.3}$$

Le rappel, quant à lui, est défini comme le ratio des vrais positifs (TP) par rapport à la somme des vrais positifs et des faux négatifs (FN). Les faux négatifs représentent les éléments réels de la classe d'intérêt qui ont été incorrectement classés comme n'appartenant pas à cette classe. Le rappel évalue donc la capacité du modèle à identifier tous les éléments réels de la classe d'intérêt :

$$\text{Rappel} = \frac{TP}{TP + FN} \tag{2.4}$$

Le F1-score est une mesure qui combine à la fois la précision et le rappel en une seule valeur. Il est défini comme la moyenne harmonique de la précision et du rappel :

$$\text{F1-score} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \tag{2.5}$$

Ces mesures sont cruciales, car elles permettent d'appréhender différentes dimensions de la performance du modèle : la précision évalue sa capacité à éviter les faux positifs, tandis que le rappel

évalue sa capacité à détecter tous les éléments d'intérêt, même au risque d'incorporer certains faux positifs. En combinant ces deux mesures, il est possible d'avoir une vision plus complète et nuancée de la performance d'un modèle de classification.

En plus des mesures de performance, il est également important d'analyser les erreurs spécifiques faites par le modèle pour identifier les cas où il performe bien et ceux où il échoue. Cela peut impliquer l'examen des exemples mal classés, la visualisation des prédictions du modèle sur des données de test, et l'utilisation de matrices de confusion pour comprendre les tendances de classification incorrectes.

En résumé, l'évaluation d'un modèle de *machine learning* est un processus multidimensionnel qui nécessite une analyse approfondie de ses performances à travers différentes mesures et une compréhension de ses erreurs. Une évaluation complète permet de prendre des décisions éclairées sur l'adéquation du modèle pour une tâche donnée et de guider les efforts d'amélioration continue.

Chapitre 3

Réalisation

Dans ce chapitre, l'ensemble du travail pratique sera réalisé pour mener notre projet à bien, en s'appuyant sur les fondations posées dans les chapitres de conception et de stratégie. Les étapes de mise en œuvre technique y sont détaillées, les choix technologiques effectués et la manière dont les défis techniques sont confrontés et surmontés. L'objectif est de présenter un aperçu exhaustif des actions concrètes entreprises, depuis l'initialisation du projet jusqu'à sa finalisation. Mais aussi, il y sera illustré comment les théories et les plans établis précédemment ont été traduits en actions pratiques, soulignant les ajustements nécessaires pour adapter le projet aux réalités rencontrées. Ce chapitre est crucial pour apprécier la cohérence entre la théorie et la pratique et pour évaluer l'efficacité des méthodes appliquées dans la réalisation de nos objectifs.

3.1 Modèle choisi

Dans le processus de sélection du modèle le plus adapté à notre tâche de segmentation d'images, plusieurs architectures de réseaux neuronaux ont été exploré. Parmi celles-ci, le modèle U-Net s'est démarqué comme étant le plus performant, offrant des résultats significativement supérieurs aux autres architectures testées. Cette décision découle d'une évaluation approfondie des performances de chaque modèle, prenant en compte des critères tels que la précision, la sensibilité et la spécificité de la segmentation. Les résultats obtenus avec le modèle U-Net ont démontré sa capacité à fournir des prédictions précises et cohérentes, ce qui en fait le choix optimal pour notre projet de segmentation d'images. En conséquence, le modèle U-Net est retenu comme pilier central de notre approche de segmentation, ainsi, elle permet de poursuivre les travaux avec confiance et efficacité.

3.2 Jeux de données utilisés

Au cours de ce projet, une évolution significative du jeu de données est observé, élément central de notre initiative. Les données constituent l'essence même du *machine learning* et revêtent une importance capitale pour l'entraînement de notre modèle. Nous décrirons ci-dessous comment le jeu de données a évolué au fil de notre progression.

3.2.1 Jeu de données initial

Au début de ce projet, un jeu de données composé d'images de certaines pierres du temple de Delphes a été fourni. Ces images incluaient des exemplaires avec et sans pigments bleus. Certaines de ces images présentaient également des zones floues qui ont été traiter.

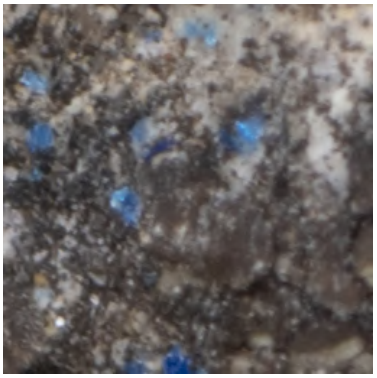


FIGURE 3.1 – Points bleus

FIGURE 3.2 – Masque des points bleus

FIGURE 3.3 – Image floue

Ce jeu de données était accompagné des masques correspondant à la vérité terrain, créés grâce à une étiqueteuse web.

FIGURE 3.4 – Masque des points bleus associé

Chaque masque est associé à une image spécifique du jeu de données et indique les positions des pigments bleus. Les zones en blanc sur les masques représentent l'emplacement des pigments bleus sur les images, lesquelles peuvent contenir ou non des pigments.

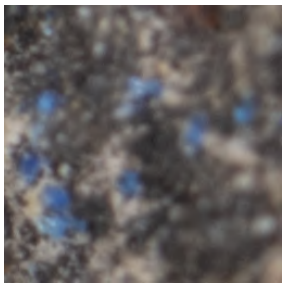


FIGURE 3.5 – Points bleus

FIGURE 3.6 – Masque des points bleus

Lors de l'évaluation de ce jeu de données, certaines images avaient été mal étiquetées, ce qui a affecté négativement les performances de notre modèle. Sur l'image ci-dessous, on constate que la prédiction a détecté des pigments alors que la vérité terrain ne les avait pas identifiés.

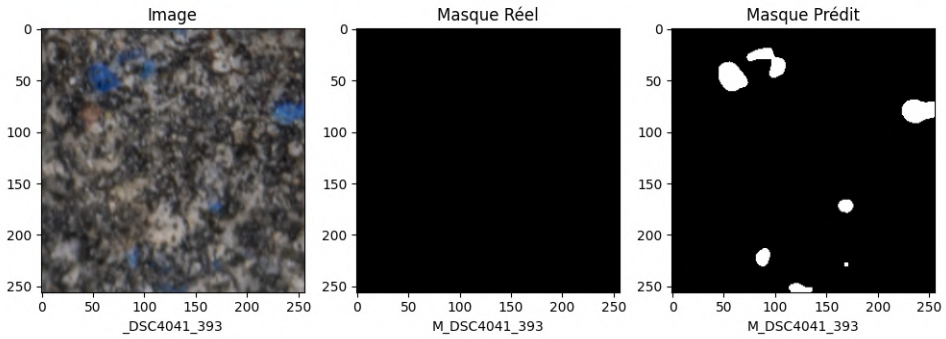


FIGURE 3.7 – Erreur du jeu de données

3.2.2 Étiquetage de nouvelles images

Plus tard dans le projet, notre encadrant, **Bruno Dutailly**, a partagé un nouveau jeu d'images, provenant de différentes pierres, présentant une teinte plus chaude due à des conditions météorologiques plus ensoleillées lors de la prise des photos précédentes.

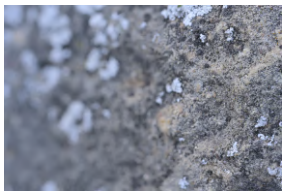


FIGURE 3.8 – Photo du premier jeu de données

FIGURE 3.9 – Photo du second jeu de données

Ce nouveau jeu d'images n'était pas étiqueté, ce qui nous a conduit à réaliser cette tâche nous-mêmes. Lors de cette opération, les limites de l'ergonomie de l'étiqueteuse ont été observées, dont les défauts seront abordés ultérieurement dans le rapport. De plus, il était facile de pouvoir manquer des parties de l'image lors du processus d'étiquetage. Toutefois, l'ajout de ces images au jeu de données demeure important et intéressant pour améliorer la compréhension du modèle utilisé pour l'apprentissage.

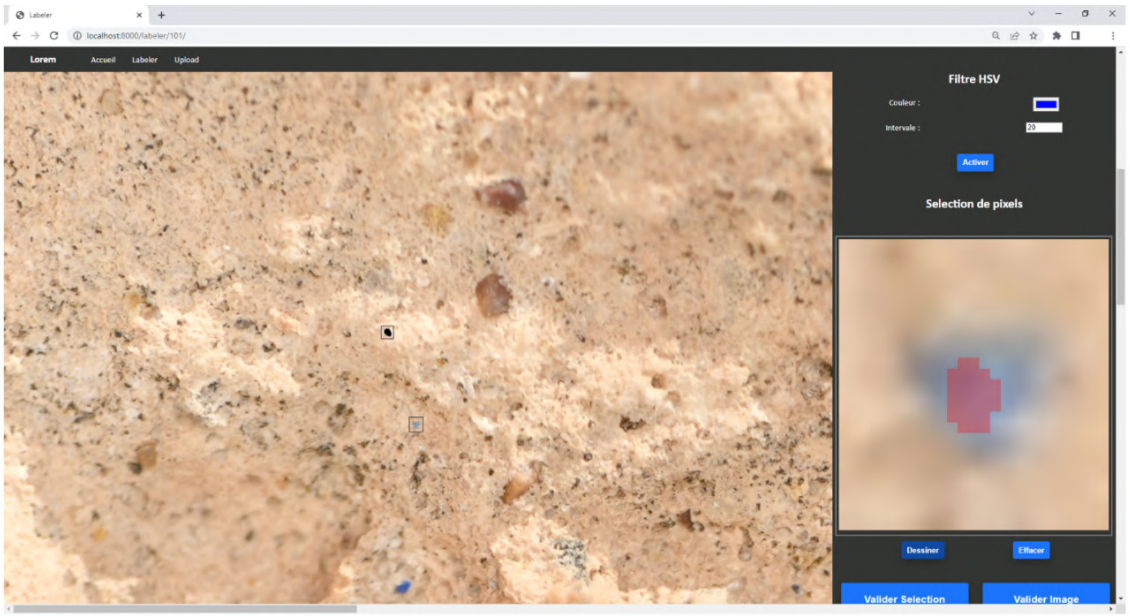


FIGURE 3.10 – Étiqueteuse d'image

3.2.3 Augmentation du jeu de données disponible

Malgré la disponibilité de deux jeux d'images, il est pertinent d'augmenter ce jeu de données en mettant en œuvre différents algorithmes.

Algorithme de rotation d'images

Pour élargir le jeu de données, il a été initialement d'envisagé d'effectuer des rotations aléatoires de 45, -45, 90 ou -90 degrés sur un nombre aléatoire d'images. Cependant, ces rotations ont entraîné des coupures d'images et l'apparition de zones noires, impactant négativement les résultats. Une tentative de zoom pour éliminer les bordures noires a également conduit à une perte de données supplémentaire.

Par la suite, nous avons opté pour des rotations aléatoires entre 90 et -90 degrés, et 180 ou -180 degrés en ajoutant aléatoirement un effet miroir à l'image. Cette approche a permis de conserver l'intégralité des données de l'image tout en les modifiant.

Algorithme de réchauffement d'images

La différence de température entre les deux jeux de données disponibles a incités à explorer l'idée de manipuler la température des images. Cependant, la prudence prudents est exigés, car une augmentation excessive de la température pouvait altérer la teinte bleue en la rapprochant de la couleur verte ou en la mêlant aux autres couleurs, compliquant ainsi la détection des pigments. les valeurs des composantes HSV (teinte, saturation et valeur) ont été ajustées de manière aléatoire dans des plages appropriées, avec des variations contrôlées. La plage que nous avons choisie se situe entre -30 et plus 30 uniquement sur la composante de teinte.

3.3 Entraînement et Évaluation du modèle

Le modèle a été entraîné sur le jeu de données augmenté, en utilisant un réseau de neurones UNET adapté à la tâche de segmentation d'images. La fonction de perte choisie était la fonction de perte Dice, qui mesure la similitude entre les prédictions de segmentation et les segmentations de vérité terrain.La valeur de la *batch size* est 20 et le taux d'apprentissage choisi est $1 * 10^{-5}$. Le taux d'apprentissage a été réduit de manière dynamique pendant l'entraînement en utilisant une stratégie de plateau. Les performances du modèle ont été évaluées sur un ensemble de données de validation indépendant. La métrique d'évaluation utilisée était le score Dice Le modèle a obtenu un score Dice de 0,95 sur l'ensemble de données de validation, ce qui indique une performance élevée dans la segmentation des images. L'évaluation utilisé aussi est les résultats de la matrice de confusion où une précision de 0.83 et un rappel de 0.93 ont été observés.

3.4 Erodage des points bleus

Au début de notre projet, à travers de multiples essais, il est apparu que notre modèle d'apprentissage détectait plus efficacement les petits pigments que les gros. Cette particularité s'explique principalement par les défis associés à l'annotation des petits pigments, où les marges d'erreur tendent à être plus larges. Face à cette observation, il a été recommandé d'appliquer une technique d'érosion aux pixels entourant les pigments.

L'érosion est une opération morphologique qui réduit les régions dans une image en se basant sur les composantes connexes des pixels. En pratique, cette technique implique l'érosion autour d'un pigment sur un rayon défini, créant ainsi une marge nette qui sépare la fin du pigment du reste de l'image. Cela assure que la zone sélectionnée contienne exclusivement le pigment et élimine les erreurs potentielles dues à des débordements lors de la définition de la vérité terrain, ou à des similitudes de couleur minimes entre le pigment et la matière avoisinante de la pierre.

Pour réaliser cette opération, la librairie *opencv2* a été utilisée, elle offre les outils nécessaires pour mettre en œuvre efficacement des opérations d'érosion tout en ajustant précisément les paramètres selon nos besoins spécifiques. Ce processus a contribué à améliorer la précision de notre modèle en affinant la détection des pigments, permettant ainsi une meilleure généralisation des résultats obtenus.

3.5 Proportion masques noirs

La proportion du masque noir [12, 18, 8] dans le jeu de données revêt une importance cruciale. Le masque noir dans ce contexte fait référence aux parties de l'image où aucun pigment n'est présent ou détecté. Cette proportion influence directement la capacité du modèle à généraliser et à apprendre efficacement à reconnaître les pigments.

1. **Balancement du jeu de données :** Une proportion équilibrée de masque noir et de masque non noir est essentielle pour éviter un biais dans l'apprentissage du modèle. Un déséquilibre peut conduire le modèle à privilégier les classes majoritaires (zones pigmentées), au détriment de l'apprentissage des zones sans pigment et peut conduire à un sur ajustement du modèle ou un sous ajustement du modèle.
2. **Impact sur la performance :** Si la proportion du masque noir est trop faible, le modèle peut avoir du mal à identifier les zones sans pigment, ce qui peut entraîner des faux positifs ou une détection insatisfaisante des pigments. À l'inverse, une proportion trop élevée peut réduire la diversité des exemples d'apprentissage et nuire à la généralisation du modèle.

Recommandations pour la proportion du masque noir :

— **Idéalement, maintenir un équilibre** où la proportion du masque noir représente environ 50% à 70% du jeu de données total, selon la complexité du problème et de la proportion de zones pigmentés dans le réel.
— **Analyse statistique préliminaire :** Avant l'entraînement du modèle, il est recommandé d'effectuer une analyse statistique du jeu de données pour évaluer la répartition du masque noir par rapport aux zones pigmentées.Dans notre cas, il y a 1.519% de patchs ayant des pigments.
— **Augmentation de données :** Si le jeu de données présente un déséquilibre significatif, des techniques d'augmentation de données peuvent être utilisées pour synthétiser des exemples supplémentaires de masque noir, améliorant ainsi la diversité des données d'entraînement.

3.6 Fonction de perte retenue

Dans le processus d'évaluation des différentes fonctions de perte pour notre tâche de segmentation sémantique, nous avons procédé à une analyse de leurs performances respectives. Cependant, sur la base des conseils avisés de nos encadrants et des résultats significativement supérieurs obtenus lors de l'expérimentation avec la *Dice Loss*, cette fonction a été privilégiée pour la suite de notre projet. Cette décision a été motivée par la constatation que les performances obtenues avec la *Dice Loss* étaient largement satisfaisantes et répondaient aux exigences de notre tâche de segmentation. Ainsi, plutôt que de poursuivre les évaluations comparatives, nous avons jugé opportun de consacrer notre attention et nos ressources à d'autres aspects du projet nécessitant des développements et des optimisations supplémentaires.

La *Dice Loss*, également connue sous le nom de coefficient de similarité de *Dice*, est une métrique bien adaptée aux tâches de segmentation où la détection précise des objets est essentielle. Elle mesure la similarité entre deux ensembles en calculant le rapport entre deux fois l'intersection de ces ensembles et la somme de leurs cardinaux. Formellement, la *Dice Loss* est définie comme suit :

$$L = 1 - \frac{2 \times \sum_{i=1}^{N} p_i g_i + \epsilon}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} g_i + \epsilon}$$

où p_i et g_i représentent respectivement les prédictions et les vérités terrain pour chaque pixel i, et ϵ est un terme d'ajustement pour éviter la division par zéro.

Notre choix de la *Dice Loss* s'est appuyé sur plusieurs facteurs. Tout d'abord, elle présente une sensibilité élevée aux petites variations dans les prédictions, ce qui est crucial pour notre tâche de détection de pigments bleus dans des images de microscopie où les détails fins sont significatifs. De plus, la *Dice Loss* est capable de gérer efficacement les déséquilibres de classe, un aspect important étant donné que les pigments bleus peuvent être rares et dispersés dans les images.

Enfin, l'expérimentation sur notre jeu de données a révélé que la *Dice Loss* produisait des résultats supérieurs en termes de précision et de rappel par rapport aux autres fonctions de perte évaluées. Cette performance accrue a conforté la décision de retenir la *Dice Loss* comme fonction de perte principale pour notre modèle de segmentation sémantique.

3.7 Résultat

L'approche initiale de l'utilisation du réseau Mask R-CNN dans le cadre du projet était basée sur sa présence dans le projet précèdent. Cependant, après une analyse approfondie, il est devenu évident que l'utilisation de Mask R-CNN pour la segmentation sémantique n'était pas logique, car cette architecture est spécifiquement conçue pour la segmentation d'instances, qui distingue et segmente chaque instance d'objet individuellement, tandis que notre tâche impliquait la segmentation sémantique, où l'objectif était de classifier chaque pixel de l'image en tant que pigment bleu ou non.

Pour trouver une architecture plus adaptée à notre tâche, des expérimentations avec plusieurs architectures de réseaux de segmentation sémantique ont été entrepris, notamment DeepLab V3, MSUNet et U-Net. Après évaluation, U-Net produisait les meilleurs résultats pour ce cas d'utilisation spécifique. Forts de cette constatation, nous avons approfondi notre exploration en cherchant des méthodes d'entraînement supplémentaires pour améliorer les performances du modèle.

Dans cette optique, nous avons exploré une méthode utilisant la précision mixte avec une amplification automatique (Mixed Precision Training), combinée à un scaling des images à une valeur unitaire et l'utilisation d'un score Dice pour la validation et l'entraînement. Cette approche a permis de détecter et de prévenir efficacement le surapprentissage du modèle. Les résultats obtenus ont été prometteurs, démontrant une amélioration significative de la capacité du modèle à généraliser aux données de test.

De plus, afin d'évaluer plus précisément les performances du modèle, nous avons développé un code permettant de générer une matrice de confusion en fonction des labels de pigment bleu. Cette matrice a fourni une vue détaillée des prédictions du modèle par rapport aux classes réelles, confirmant la qualité des résultats obtenus.

En outre, lors de l'analyse des modèles formés sur différents ensembles de données d'entraînement, nous avons observé un phénomène de surapprentissage sur certains modèles, en particulier lorsque la proportion de masque noir dans le jeu de données d'entraînement dépassait 50%. Cette observation souligne l'importance de maintenir un équilibre dans la répartition des classes lors de la constitution du jeu de données d'entraînement pour éviter le sur-apprentissage.

3.7.1 résultat entraînement d'un jeu de données possédant 20% de masques négatifs et 80% de masques positifs

Configuration de l'entraînement :
— *epoch* :1000
— *batch size* : 20
— *scale* : 1
— amplification automatique : activé
— *learning rate* : de base : $1 * 10^{-5}$

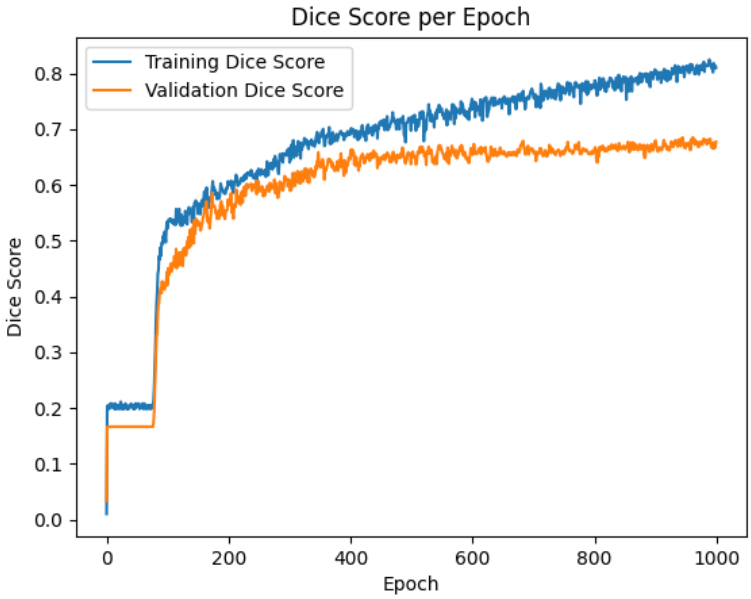


FIGURE 3.11 – Graphique de l'évolution du Dice Score de validation et d'entraînement par epoch

Le graphique 3.11 montre que le score de dés d'entraînement augmente constamment au cours de la formation, atteignant un pic de 0,98 à l'époque 1000. Cela indique que le modèle apprend à segmenter les images avec de plus en plus de précision au fil du temps.

Le score de dés de validation, en revanche, augmente plus lentement et atteint un pic de 0,95 à l'époque 800. Cela indique que le modèle commence à sur-adapter les données d'entraînement après l'époque 800. En d'autres termes, le modèle apprend à segmenter les images d'entraînement très précisément, mais il ne généralise pas bien aux nouvelles images après l'époque 800.
L'évaluation a été faite sur le modèle sauvegardé avant que le surapprentissage soit observé.

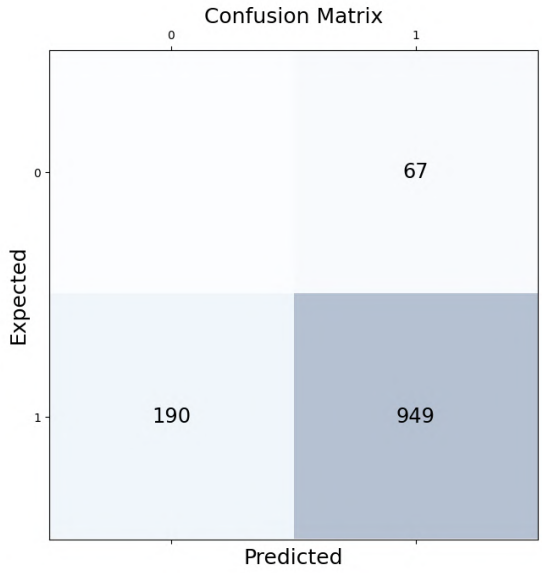


FIGURE 3.12 – Matrice de confusion

Précision : La précision est la proportion de prédictions positives qui sont réellement positives. Dans ce cas, la précision est de 949/(949 + 67) = 0.955. Cela signifie que le modèle prédit correctement les images positives dans 95,5% des cas.

Rappel : Le rappel est la proportion de classe réelle positive qui est correctement identifiée comme telle. Dans ce cas, le rappel est de 949/(949 + 190) = 0.832. Cela signifie que le modèle identifie correctement 83,2% des images positives.

F1-score : Le F1-score est une mesure moyenne de la précision et du rappel. Il est calculé en prenant la moyenne harmonique de la précision et du rappel. Dans ce cas, le F1-score est de 2 * (0.955 * 0.832) / (0.955 + 0.832) = 0.887. Cela signifie que le modèle a une performance globale moyenne de 88,7% dans la classification des images.
D'après la matrice de confusion et les métriques d'évaluation, on peut conclure que le modèle a une performance élevée dans la classification des images. La précision et le rappel sont tous deux élevés, et le F1-score indique une performance globale moyenne solide.
Cependant, il est important de noter que le modèle a un nombre élevé de faux négatifs (FN = 190). Ce qui peut s'expliquer par le nombre de patchs où il y a une composante connexe représentant une zone pigmenté qui est coupée 3.13 et non centrée.
Un aspect potentiellement contraignant pour les archéologues réside dans la découverte que de nombreux pigments bleus issus de la vérité terrain ne sont pas prévus par le modèle. Cette situation n'est pas systématique lorsqu'on examine les faux négatifs identifiés par notre programme d'évaluation.

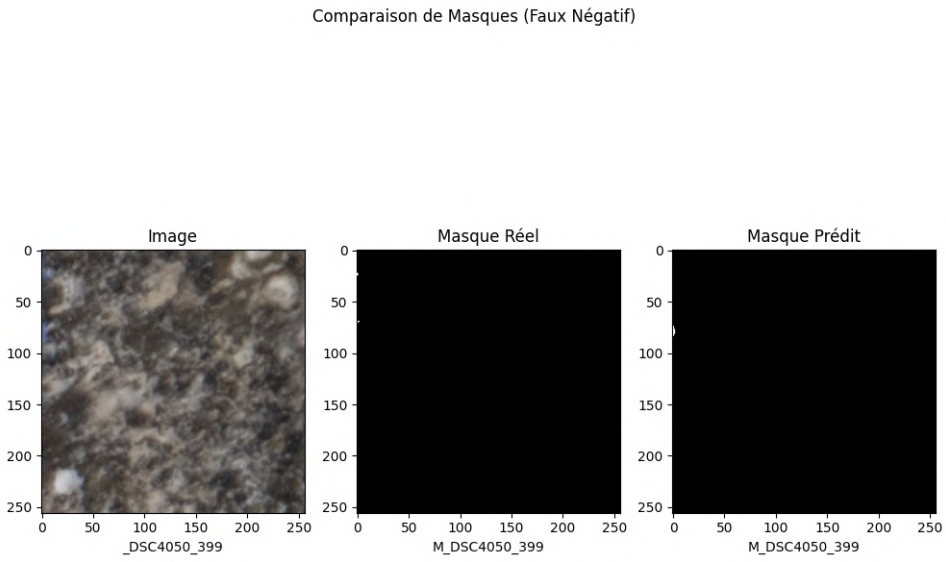


FIGURE 3.13 – Comparaison d'un patch où un faux négatif a été observé

3.7.2 résultat entraînement d'un jeu de données possédant 80% de masques négatifs et 20% de masques positifs

Configuration de l'entraînement :
— *epoch* :1000
— *batch size* : 20
— *scale* : 1
— amplification automatique : activé
— *learning rate* : de base : $1 * 10^{-5}$

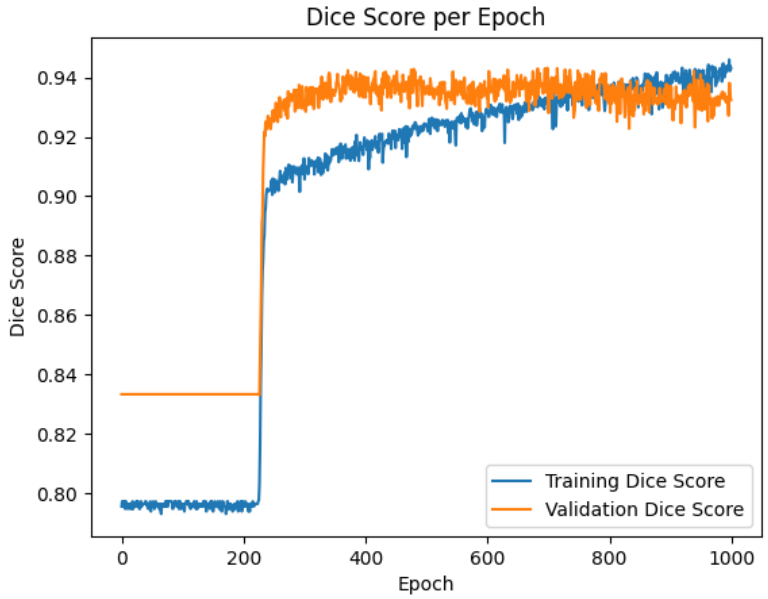


FIGURE 3.14 – Graphique de l'évolution du Dice Score de validation et d'entraînement par epoch

Le graphique 3.14 montre que le score de dés d'entraînement augmente constamment au cours de la formation, atteignant un pic de 0,94 à l'époque 1000. Cela indique que le modèle apprend à segmenter les images avec de plus en plus de précision au fil du temps.

Le score de dés de validation, en revanche, augmente plus lentement et atteint un pic de 0,92 à l'époque 600. Cela indique que le modèle commence à sur-adapter les données d'entraînement après l'époque 600.
L'évaluation a été faite sur le modèle sauvegardé avant que le surapprentissage soit observé.

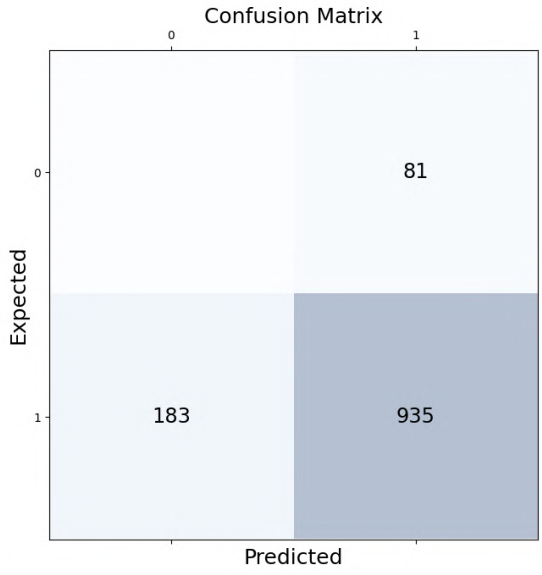


FIGURE 3.15 – Matrice de confusion

Précision : La précision est la proportion de prédictions positives qui sont réellement positives. Dans ce cas, la précision est de 935/(935 + 81) = 0.922. Cela signifie que le modèle prédit correctement les images positives dans 92,2% des cas.

Rappel : Le rappel est la proportion de classe réelle positive qui est correctement identifiée comme telle. Dans ce cas, le rappel est de 935/(935 + 183) = 0.834. Cela signifie que le modèle identifie correctement 83,4% des images positives.

F1-score : Le F1-score est une mesure moyenne de la précision et du rappel. Il est calculé en prenant la moyenne harmonique de la précision et du rappel. Dans ce cas, le F1-score est de 2 * (0.922 * 0.834) / (0.922 + 0.834) = 0.877. Cela signifie que le modèle a une performance globale moyenne de 87,7% dans la classification des images.
D'après la matrice de confusion et les métriques d'évaluation, on peut conclure que le modèle a une performance élevée dans la classification des images. La précision et le rappel sont tous deux élevés, et le F1-score indique une performance globale moyenne solide.
Mais ces résultats 3.15 paraissent moindres que les premiers 3.12

Chapitre 4

Perspectives futures a considérer

4.1 Interface Graphique

4.1.1 Interface Web

Concernant l'interface web de l'étiqueteuse d'images, notre équipe n'a pas eu l'occasion de développer cette composante durant la phase actuelle du projet. L'interface a été principalement utilisée pour enregistrer des vérités terrain. Il a été observé que sa conception manque d'ergonomie, présentant des défis en termes d'utilisation efficace. Plusieurs améliorations pourraient être envisagées afin d'optimiser son fonctionnement et d'améliorer l'expérience utilisateur.

4.1.2 Amélioration de la fonctionnalité de zoom

Il est primordial d'implémenter une fonctionnalité permettant de zoomer directement sur l'image au sein de l'interface, indépendamment du zoom du navigateur web. L'utilisation actuelle du zoom du navigateur se révèle peu pratique, car elle affecte la dimension globale de l'interface, compliquant ainsi l'utilisation. Cette modification faciliterait une analyse plus précise des images sans altérer l'ergonomie générale de l'interface utilisateur.

Implémentation d'une mini-carte

Pour améliorer la navigation au sein des images, l'ajout d'une mini-carte est recommandé. Cette fonctionnalité permettrait aux utilisateurs de se situer précisément dans l'image, un atout crucial pour ne pas omettre des détails importants tels que la présence de pigments bleus. Actuellement, les seuls repères disponibles sont les barres de défilement du navigateur, qui manquent de précision et ne facilitent pas une navigation efficace dans les images.

Un bon compromis ?

Enfin, pour parfaire l'expérience utilisateur, une approche combinée pourrait être envisagée. Cela consisterait à intégrer un mécanisme de zoom sur l'image, adapté à sa taille, ainsi qu'un découpage de l'image en petits espaces bien délimités, rendus cliquables sur une mini-carte. Cette approche offrirait aux utilisateurs un bon compromis entre la visualisation globale de l'image et la navigation détaillée dans ses différents éléments. De plus, elle permettrait aux utilisateurs de scroller si nécessaire, notamment dans le cas où un pigment chevaucherait deux découpages.

Panels d'outils

Une autre amélioration envisageable serait l'ajout de divers outils dans l'interface, offrant ainsi une palette fonctionnelle plus riche. Parmi ces outils, on pourrait inclure un outil de remplissage par rayon, permettant à l'utilisateur de cliquer au centre d'un point puis à une certaine distance pour créer un rayon. Le cercle ainsi formé serait instantanément rempli. De plus, un outil de remplissage pour des formes autres que circulaires pourrait être envisagé, offrant une flexibilité supplémentaire. Alternativement, un outil similaire à une pipette pourrait être développé, permettant à l'utilisateur de remplir tous les pixels environnants en fonction de la couleur sélectionnée sur le point choisi. Dans ce cas, le processus s'arrêterait dès qu'une différence de couleur trop élevée serait détectée. Ces outils supplémentaires offriraient une plus grande variété de fonctionnalités et faciliteraient ainsi le processus d'étiquetage des images.

Interface épurée

Dans un souci d'amélioration du confort d'utilisation, une refonte de l'interface pourrait être envisagée. Cela inclurait l'utilisation d'un cadre pour encadrer l'image, la conception d'une boîte à outils pour accueillir les différents outils, ainsi que le choix d'un jeu de couleurs plus attrayant. Bien que cette tâche nécessite un effort conséquent, les bénéfices qu'elle apporte sont relativement limités. Néanmoins, travailler dans un environnement épuré peut contribuer à une expérience utilisateur plus sereine et favoriser une meilleure concentration et efficacité dans le travail.

4.1.3 Interface Terminal pour la prédiction de pigments

Développement logiciel

Il serait intéressant de développer un logiciel pour la gestion de l'entraînement des données ainsi que l'évaluation des modèles de *machine learning*, éliminant ainsi la nécessité d'utiliser un terminal. Cette évolution vise à rendre le projet plus accessible aux utilisateurs moins expérimentés.

Gestion des fichiers

La gestion des fichiers serait intégrée à une interface graphique, permettant la navigation par un explorateur de fichiers ou le glisser-déposer (drag and drop). Cette approche, non seulement esthétique, faciliterait considérablement les opérations si les chemins des jeux de données varient. De surcroît, lors de l'importation des images, le système pourrait gérer automatiquement les cas où certains masques seraient manquants, en important toutes les autres images si le nombre de masques absents est faible, évitant ainsi les ajustements manuels. Un système de mise en évidence des images affectées serait également envisageable, permettant de générer manuellement les vérités terrain nécessaires pour compléter le jeu de données. Actuellement, les masques sont associés aux images par le biais des noms de fichiers, ce qui faciliterai l'implémentation pour le repérage des masques manquants.

Visualisation des résultats

Le logiciel proposé pourrait intégrer une fonctionnalité permettant la visualisation des résultats dans un onglet spécifiquement dédié, ce qui faciliterait considérablement l'interprétation des données, surtout pour les utilisateurs novices, tout en optimisant la gestion du temps. Il serait judicieux d'ajouter des interfaces permettant la visualisation des jeux de données utilisés précédemment, la sélection des modèles à appliquer, ainsi que la navigation à travers les masques générés. La capacité de comparer visuellement les masques produits avec les images originales et les masques de vérité terrain est essentielle. Une fonctionnalité de surbrillance superposant l'image, le masque obtenu, et le masque de vérité terrain pourrait identifier les pigments bleus non détectés lors de l'étiquetage initial ou les erreurs de prédiction du modèle.

4.1.4 Modularité du logiciel

La sélection des modèles vu précédemment aurait un intérêt d'adaptabilité. Cette adaptabilité du logiciel pourrait le rendre applicable dans d'autres domaines, en permettant une variabilité des modèles et des jeux de données utilisés, ce qui étendrait significativement l'objectif initial du projet. Une architecture logicielle modulaire permettrait une intégration aisée de nouvelles fonctionnalités et une adaptation rapide à différentes configurations de données et de modèles. Cette modularité renforcerait la flexibilité du logiciel, le rendant ainsi adaptable à une large gamme d'applications, tout en facilitant sa maintenance et son évolution à long terme.

4.2 Augmentation du jeu de données

Dans le cadre de ce projet, le nombre de données utilisées était relativement limité, ce qui a incité à explorer des méthodes pour augmenter le jeu de données disponible. Cette démarche a été partiellement réalisée par l'implémentation d'algorithmes qui appliquent des transformations géométriques, telles que des rotations aléatoires, sur les *patchs* existants ainsi que sur leurs masques correspondants. De plus, un algorithme de manipulation de la chaleur de l'image a été élaboré afin de diversifier les nuances de bleu présentes dans les données. Cette approche vise à enrichir la variabilité des exemples d'entraînement, permettant ainsi au modèle d'apprendre à reconnaître différents types de pigments bleus, et non pas seulement ceux présents dans un seul type d'image.

4.2.1 Acquisition de davantage d'images de terrain

Pour enrichir notre jeu de données, la stratégie la plus directe consiste à intégrer un plus grand nombre d'images des roches du temple. Les images actuellement à notre disposition ont été sélectionnées par des chercheurs qui ont jugé probable la présence de pigments sur ces pierres spécifiques. L'ajout de nouvelles images, même dépourvues de pigments bleus, contribuerait significativement à l'efficacité de l'entraînement du modèle en augmentant la diversité et le volume du jeu de données.

4.2.2 De nouveaux algorithmes pour augmenter le jeu de données existant

Nous avons déjà expérimenté l'application de filtres et de rotations sur les images existantes pour augmenter notre jeu de données. Il serait judicieux d'explorer d'autres méthodes d'augmentation pour enrichir davantage le jeu de données. Bien que ces images modifiées ne soient pas entièrement nouvelles, elles offrent au modèle l'opportunité de s'adapter à de nouveaux paramètres visuels. Nous envisageons donc de développer de nouveaux algorithmes, par exemple en découpant des images et en les combinant de manière créative pour analyser l'impact sur les performances du modèle.

4.2.3 Intégration d'une API de banque d'images

L'incorporation d'une API de banque d'images pourrait s'avérer bénéfique pour l'entraînement de notre modèle. Bien qu'incertains de la pertinence directe de ces images pour notre projet spécifique, il est plausible de trouver des jeux de données contenant des exemples de pigments bleus ou de petits agrégats de pixels colorés. Notre objectif étant l'identification précise de pigments bleus, l'accès à une telle diversité d'images pourrait enrichir significativement notre jeu de données. Cette flexibilité permettrait également d'adapter le logiciel à différents modèles et applications, en sélectionnant des modèles appropriés pour d'autres types d'analyse d'images. Idéalement, notre interface graphique inclurait une fonctionnalité permettant de visualiser et de rechercher des jeux de données disponibles, avec une barre de recherche pour filtrer les données selon leur pertinence et leur efficacité, offrant ainsi une adaptabilité accrue à l'utilisateur en fonction de ses besoins spécifiques.

4.3 Pré-entraînement du modèle

4.3.1 Utiliser les entraînements effectués sur notre jeu de données

Dans notre projet, il est déjà possible de conserver les résultats obtenus des entraînements précédents pour pré-entraîner notre modèle. Toutefois, l'accès à cette fonctionnalité n'est pas particulièrement intuitif actuellement. Dans la perspective de développer un logiciel doté d'une interface graphique, il serait judicieux de rendre cette option facilement accessible. Cela permettrait d'exploiter pleinement les capacités de pré-entraînement du modèle, favorisant ainsi l'obtention de meilleurs résultats lors des prédictions futures. En intégrant cette fonctionnalité de manière transparente et efficace, nous pourrions non seulement optimiser le processus d'apprentissage du modèle mais également améliorer l'expérience utilisateur en simplifiant l'interaction avec le système.

4.3.2 Utiliser des modèles pré-entraînés

Certains modèles de segmentation offrent des options incluant des configurations déjà pré-entraînées. Jusqu'à présent, ils n'ont pas été utilisés, redoutant une possible détérioration de nos résultats, car le pré-entraînement de ces modèles ne serait pas nécessairement adapté à notre contexte spécifique. Toutefois, il pourrait être bénéfique de réexaminer cette approche. Approfondir l'étude des modèles pré-entraînés disponibles pourrait révéler des opportunités inattendues d'améliorer notre précision ou de réduire le temps nécessaire pour atteindre une efficacité optimale, notamment en adaptant ces modèles à notre cas d'utilisation spécifique. En explorant ces possibilités, nous pourrions potentiellement enrichir notre stratégie de modélisation tout en conservant une adaptabilité élevée aux caractéristiques uniques de notre projet.

4.4 Recherche de modèles plus pertinents ou performants

Lors de ce projet, parmi les modèles de segmentation les plus reconnus, ceux qui semblaient les plus adaptés à nos besoins ont été sélectionnés. Toutefois, compte tenu des exigences spécifiques de

notre application, il est plausible que d'autres possibilités, peut-être moins connues ou émergentes, pourraient offrir des performances équivalentes ou supérieures à celles des solutions actuelles. Dans cette optique, il serait judicieux d'explorer et d'intégrer de nouveaux modèles, continuant ainsi la quête d'optimisation tant en termes de temps de calcul que de qualité des résultats. Cette démarche s'inscrit également dans nos objectifs de maintenance continue du code et d'amélioration de l'expérience utilisateur. En intégrant régulièrement des innovations dans notre plateforme, on peut s'assurer non seulement de maintenir le système à la pointe de la technologie, mais aussi assurer une adaptation agile aux évolutions futures du domaine.

4.5 Intégration complète dans un logiciel unique ?

Pour renforcer la cohérence de notre projet et améliorer son accessibilité, une interface web de l'étiqueteuse directement dans notre logiciel principal peut être intégrée. Cette centralisation offrirait une expérience utilisateur plus fluide et unifiée, en regroupant tous les éléments nécessaires dans une seule et même plateforme. Cette approche pourrait non seulement simplifier l'utilisation du logiciel, mais également améliorer la gestion globale des différentes fonctionnalités et données.

4.6 Ou optimisation des calculs via le cloud computing ?

Considérant la puissance de calcul importante requise par notre application, une alternative pourrait consister à héberger l'ensemble du système sur le web et à dédier les traitements à des superordinateurs via le *cloud computing*. Cette solution permettrait de réduire significativement les temps de traitement, en tirant parti de ressources matérielles supérieures et potentiellement plus économiques à grande échelle. En optimisant davantage notre code, nous pourrions ainsi réduire les coûts tout en améliorant les performances du système. Avec une puissance accrue à disposition, il serait possible d'augmenter le nombre d'epochs durant l'entraînement, visant ainsi une précision optimale des prédictions.

4.7 Perspectives prioritaires d'amélioration

Au cours de ce projet, nous avons exploré diverses pistes d'amélioration visant à affiner et à élargir notre projet. Nous avons réalisé que notre simple modèle d'apprentissage pouvait être étendu bien au-delà des limites de la simple reconnaissance des pigments, offrant ainsi un potentiel d'évolution considérable. Dans cette section, nous avons discuté des différentes pistes d'amélioration, mais nous estimons que l'ergonomie et la centralisation du projet sont des aspects cruciaux à ce stade. Que ce soit en ligne ou en local, l'amélioration de ces aspects constitue notre priorité.

4.8 Améliorations externes

Dans cette partie, nous explorons des options externes qui pourraient améliorer les performances de notre modèle ou apporter une aide supplémentaire au projet de manière indirecte.

4.8.1 Les photos

Les images du jeu de données que nous avons reçues pourraient être améliorées sous plusieurs aspects que nous détaillerons ci-après.

Profondeur de champs

La profondeur de champ représente un élément déterminant pour l'efficacité de l'apprentissage de nos modèles. Comme mentionné précédemment, certaines images ou sections d'images présentent un flou qui peut entraver le processus d'apprentissage automatique, ainsi que le travail des chercheurs, en réduisant la quantité d'informations exploitables. Un accroissement de la profondeur de champ, bien que complexe à obtenir avec un objectif macro, permettrait de capter une gamme d'informations plus étendue sur les images. Cela contribuerait à une meilleure précision dans la détection et la classification des éléments présents dans nos images, en minimisant les pertes d'informations dues au flou.

Qualité des images

Bien que les photos soient de grande taille, nous avons constaté lors de l'étiquetage que, en raison de la petite taille des pigments, la résolution nécessaire pour annoter correctement les images est insuffisante après application du zoom. Améliorer la qualité des images représenterait un investissement coûteux pour les archéologues, étant donné le coût élevé des appareils photo à haute résolution. Par ailleurs, l'utilisation d'images de plus grande taille pourrait également augmenter nos besoins en puissance de calcul, ce qui amène à la section suivante.

4.8.2 La puissance de calcul

Dans le domaine du *machine learning*, la puissance de calcul est une composante essentielle. L'entraînement de modèles est en effet une tâche très gourmande en ressources, nécessaire pour obtenir de bons résultats. Le nombre d'époques et la complexité du modèle ont un impact significatif sur les ressources requises pour l'apprentissage. Voici quelques idées qui auraient pu aider :

Infrastructures plus puissantes

Dans le cadre de notre projet, nous avons pu lancer des tâches d'apprentissage sur de nombreuses époques avec un jeu de données d'une taille négligeable par rapport à des ensembles de données utilisés de nos jours pour des tâches d'apprentissage. Malgré l'utilisation des machines de calcul mises à notre disposition au CREMI, nous savons qu'il existe des machines dédiées au calcul avec une puissance bien supérieure à celles qui ont été allouées. Si le projet est amené à changer d'échelle, il serait envisageable d'utiliser des machines plus puissantes et performantes pour nos tâches d'entraînement, dans le but de travailler plus rapidement et de rendre nos prédictions plus précises en entraînant notre modèle sur un plus grand volume de données et sur un plus grand nombre d'époques.

Calcul parallélisé sur plusieurs machines

L'utilisation du calcul parallélisé est une stratégie performante pour optimiser les ressources et réduire le temps d'entraînement des modèles de deep learning. En distribuant le processus d'apprentissage sur plusieurs machines, il est possible de gérer de manière plus efficace des volumes de données importants et de complexifier les architectures de modèles sans être limité par les capacités d'une seule unité de traitement.

Le calcul parallélisé peut se faire de plusieurs manières, notamment par la parallélisation des données où chaque machine traite une fraction du jeu de données, ou par la parallélisation des modèles où chaque machine prend en charge une partie du modèle. Cette dernière technique est particulièrement utile pour les réseaux de neurones profonds qui nécessitent une grande capacité de calcul pour l'entraînement.

En utilisant des *frameworks* adaptés comme TensorFlow ou PyTorch, qui supportent nativement le calcul distribué, on peut facilement mettre en œuvre cette approche. Ces *frameworks* permettent de synchroniser les opérations et de gérer efficacement la communication entre les différents nœuds du réseau, assurant ainsi que le modèle final soit correctement intégré et que les performances d'apprentissage ne soient pas compromises.

Par ailleurs, cette approche permet de réduire significativement les coûts opérationnels en permettant l'utilisation de machines moins performantes mais plus nombreuses, réparties peut-être géographiquement, pour accomplir des tâches qui auraient nécessité autrement des super-calculateurs très coûteux. En résumé, le calcul parallélisé ouvre la voie à une démocratisation plus large de l'usage de techniques avancées de *machine learning*, en rendant l'accès à la puissance de calcul haute performance plus accessible et économiquement viable. Ce qui dans notre cas serait une solution viable et envisageable.

4.8.3 Discussion sur les différentes propositions

Dans cette section, nous avons exploré différentes options d'amélioration pour notre projet. Cependant, il est important de noter que chaque amélioration potentielle doit être évaluée en fonction de sa pertinence par rapport à nos objectifs et de son coût en termes de ressources, qu'elles soient matérielles, énergétiques ou temporelles.

Tout d'abord, en ce qui concerne les images du jeu de données, l'amélioration de la profondeur de champ et la qualité des images pourraient potentiellement améliorer la précision de notre modèle. Cependant, ces améliorations nécessitent des investissements importants en termes de matériel et de temps, et il est important de peser ces coûts par rapport aux bénéfices potentiels en termes de performance du modèle.

De même, en ce qui concerne la puissance de calcul, l'utilisation d'infrastructures plus puissantes ou le calcul parallélisé sur plusieurs machines peuvent offrir des gains significatifs en termes de vitesse d'entraînement et de capacité à traiter de grandes quantités de données. Cependant, ces approches nécessitent également des ressources importantes et doivent être évaluées en fonction de leur rentabilité et de leur pertinence par rapport aux objectifs du projet.

Dans l'ensemble, notre approche consiste à rechercher des améliorations incrémentielles qui offrent le meilleur rapport coût-efficacité en termes de ressources investies par rapport aux gains en performance du modèle. Bien que la recherche de la précision absolue soit importante, il est également crucial de garder à l'esprit que notre objectif principal est de détecter les pigments dans les images, et que des améliorations mineures de la qualité de l'image ou de la puissance de calcul peuvent ne pas se traduire par une amélioration significative des performances du modèle dans ce contexte spécifique.

En fin de compte, notre approche est guidée par une recherche du juste équilibre entre l'amélioration des performances du modèle et la gestion prudente des ressources disponibles. Cela nécessite une évaluation de chaque proposition d'amélioration en fonction de ses avantages potentiels et de ses coûts, afin de garantir que nos efforts sont dirigés là où ils auront le plus grand impact sur la qualité et l'efficacité de notre modèle de prédiction.

Conclusion

Pour conclure, ce projet a permis de rentrer en détail dans le monde du *machine learning* et des difficultés auxquelles on peut être confrontées lors d'une élaboration de modèles de prédiction. Au cours de ce projet, nous avons été confrontés à plusieurs défis, notamment la gestion et le traitement d'un jeu de données complexe et hétérogène. L'ajout d'un nouveau jeu d'images avec des conditions d'éclairage différentes a nécessité une adaptation de notre stratégie d'annotation et de pré-traitement, soulignant l'importance de la flexibilité dans la gestion des données.

Le projet que nous rendons aujourd'hui est non seulement pleinement fonctionnel, mais également nettement plus performant que lors de ses premières phases. Cette amélioration substantielle est le résultat de plusieurs ajustements clés au fil du développement. Premièrement, l'adoption d'une fonction de perte appropriée a significativement affiné la précision de notre modèle. De plus, le choix du modèle de réseau de neurones, spécifiquement adapté à nos besoins, a également joué un rôle crucial dans l'augmentation des performances et l'ajout du second jeu de données qui a permit une meilleur variété d'images lors de l'entraînement. Enfin, le temps qui a été alloué pour se concentrer sur la partie machine learning a permis une exploration plus profonde et une optimisation accrue de nos méthodes, conduisant à des résultats supérieurs à ceux initialement obtenus.

De plus, nous avons exploré diverses techniques d'augmentation de données comme la rotation et le réchauffement des images pour enrichir notre modèle sans nécessiter de données supplémentaires. Bien que ces techniques aient présenté leurs propres défis, notamment en termes de maintien de la qualité des données, elles ont également offert des perspectives précieuses sur comment améliorer la robustesse des modèles de machine learning face à des variations non anticipées.

L'expérimentation avec différentes architectures de réseaux de neurones et l'implémentation de fonctions de perte spécifiques comme la *Dice Loss*ont permis de mieux comprendre comment chaque composant d'un réseau peut influencer les performances générales du modèle. Cette expérience a été enrichissante, nous offrant une vision pratique des théories qu'ont été étudiées.

Les infrastructures de calcul utilisées ont également montré qu'au-delà des compétences techniques, la gestion des ressources matérielles est cruciale pour mener à bien des projets d'apprentissage profond. Cela nous a ouvert les yeux sur l'importance de l'aspect technique de l'informatique dans la recherche en intelligence artificielle.

En regardant vers l'avenir, ce projet a posé les bases de recherches plus approfondies. L'expérience acquise incite à explorer de nouvelles méthodes d'apprentissage semi-supervisé ou non supervisé, qui pourraient permettre de surmonter certaines des limitations observées, notamment dans le traitement de grandes quantités de données non étiquetées.

En somme, bien que ce projet ait été par moments exigeant, il a été extrêmement enrichissant. Il a élargi notre compréhension non seulement des techniques spécifiques, mais aussi des enjeux pratiques et théoriques du domaine de l'intelligence artificielle.

Bibliographie

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4) :834–848, 2017.

[2] Guillaume Chhor, Cristian Bartolome Aramburu, and Ianis Bougdal-Lambert. Satellite image segmentation for building detection using u-net. *Web : http ://cs229. stanford. edu/proj2017/final-reports/5243715. pdf*, 2017.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[5] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2) :37–50, 1912.

[6] Shruti Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–7, 2020.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco : Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[8] Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, Levente Lanczi, Elizabeth Gerstner, Marc-André Weber, Tal Arbel, Brian B. Avants, Nicholas Ayache, Patricia Buendia, D. Louis Collins, Nicolas Cordier, Jason J. Corso, Antonio Criminisi, Tilak Das, Hervé Delingette, Çağatay Demiralp, Christopher R. Durst, Michel Dojat, Senan Doyle, Joana Festa, Florence Forbes, Ezequiel Geremia, Ben Glocker, Polina Golland, Xiaotao Guo, Andac Hamamci, Khan M. Iftekharuddin, Raj Jena, Nigel M. John, Ender Konukoglu, Danial Lashkari, José António Mariz, Raphael Meier, Sérgio Pereira, Doina Precup, Stephen J. Price, Tammy Riklin Raviv, Syed M. S. Reza, Michael Ryan, Duygu Sarikaya, Lawrence Schwartz, Hoo-Chang Shin, Jamie Shotton, Carlos A. Silva, Nuno Sousa, Nagesh K. Subbanna, Gabor Szekely, Thomas J. Taylor, Owen M. Thomas, Nicholas J. Tustison, Gozde Unal, Flor Vasseur, Max Wintermark, Dong Hye Ye, Liang Zhao, Binsheng Zhao, Darko Zikic, Marcel Prastawa, Mauricio Reyes, and Koen Van Leemput. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10) :1993–2024, 2015.

[9] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net : Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.

[10] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning : A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7) :3523–3542, 2022.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015 : 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[13] Hyunseok Seo, Charles Huang, Maxime Bassenne, Ruoxiu Xiao, and Lei Xing. Modified u-net (mu-net) with incorporation of object-dependent high level features for improved liver and

liver-tumor segmentation in ct images. *IEEE transactions on medical imaging*, 39(5) :1316–1325, 2019.

[14] Run Su, Deyun Zhang, Jinhuai Liu, and Chuandong Cheng. Msu-net : Multi-scale u-net for 2d medical image segmentation. *Frontiers in Genetics*, 12 :639930, 2021.

[15] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In M. Jorge Cardoso, Tal Arbel, Gustavo Carneiro, Tanveer Syeda-Mahmood, João Manuel R.S. Tavares, Mehdi Moradi, Andrew Bradley, Hayit Greenspan, João Paulo Papa, Anant Madabhushi, Jacinto C. Nascimento, Jaime S. Cardoso, Vasileios Belagiannis, and Zhi Lu, editors, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, Cham, 2017. Springer International Publishing.

[16] Amos Tversky. Features of similarity. *Psychological review*, 84(4) :327, 1977.

[17] William S Vincent. *Django for Beginners : Build websites with Python and Django*. WelcomeToCode, 2022.

[18] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser : Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7) :3142–3155, 2017.

Sources annexes

Voici la liste exhaustive des sources qui ont servi dans ce projet

Images utilisées pour les illustrations

1.1 source : `https://www.mathworks.com`

1.2 source : `https://www.folio3.ai`

2.6 source : `https://i.stack.imgur`

2.7 source : `https://medium.com/data-science-bootcamp/`

2.1 source : `https://www.researchgate.net`

2.2 source : `https://learnopencv.com/`

Les images de masques et de pierres sont *tirés du jeu de données du projet*