UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Méthodes d'apprentissage profond pour la segmentation des défauts dans l'inspection des ponts en béton

Mémoire

Nadir HAMMOUCHE [1]

Ce Mémoire est évalué par le juré composé de:

Présidente du jury : Prof. Rokia Missaoui, [1]
Membre du jury : Prof. Ana-Maria Cretu, [1]
Directeur : Prof. Mohand Said Allili, [1]
Co-directeur : Dr. Jean-François Lapointe, [2]

[1] Université du Québec en Outaouais

[2] Conseil national de recherches Canada

Je dédie ce travail à mes parents.

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude envers M. Mohand Said Allili, pour sa direction, son soutien infaillible et ses conseils avisés tout au long de l'élaboration de ce mémoire. Sa patience, son expertise et son dévouement ont été des éléments essentiels dans la réalisation de ce travail.

Je tiens également à remercier chaleureusement M. Jean-François Lapointe, pour son précieux apport et ses suggestions constructives qui ont enrichi mon travail.

Je souhaite exprimer ma reconnaissance envers l'UQO pour m'avoir offert l'opportunité de poursuivre mes études dans un environnement propice à l'apprentissage et à l'épanouissement personnel et professionnel.

Mes remerciements vont également à mes proches, ma famille et mes amis, pour leur soutien indéfectible, leurs encouragements et leur compréhension tout au long de cette période intense.

Enfin, je tiens à exprimer ma reconnaissance envers toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce travail de recherche.

Je suis profondément reconnaissant de cette expérience enrichissante et des précieuses leçons apprises au cours de ce processus.

Table des matières

Table des figures

Liste des tableaux

Résumé

La maintenance des infrastructures de transport dans un état fonctionnel est une opération cruciale. Les inspections régulières et proactives de ces infrastructures comme les ponts est une stratégie efficace qui permet de repérer les anomalies et les défauts au préalable. Les recherches menées dans ce domaine récemment ont tendance à s'appuyer sur les techniques de l'apprentissage profond et de la vision par ordinateur afin d'améliorer l'efficacité de ce processus. L'objet du travail proposé à travers ce mémoire est centré sur la segmentation pour la détection des défauts dans les images d'inspections de ponts. La première contribution consiste en la construction d'un ensemble de données dédié à la tâche de segmentation des défauts. Il est constitué de 594 images prises lors d'inspections de ponts. Ces images sont annotées avec des masques appartenant à quatre classes de défauts majeurs. Dans la deuxième contribution, nous proposons trois modèles basés sur l'apprentissage multitâche pour la segmentation des défauts. L'apprentissage multitâche permet d'exploiter le contexte d'information de chaque classe de défaut afin de construire un réseau avec une meilleure généralisation et performance. Le premier modèle est un réseau U-Net classique adapté pour la segmentation multi-classe. Le deuxième est un réseau à encodeur partagé avec un mécanisme d'attention. Le dernier est un réseau à encodeur-décodeur partagé qui intègre à la fois le mécanisme d'attention et de la supervision en profondeur. Ces réseaux sont entraînés et testés sur l'ensemble de données construit préalablement et réalisent des scores Dice allant de 60% à 79%.

Mots-clés : *Inspection de pont, apprentissage profond, segmentation sémantique, apprentissage multitâche, mécanisme d'attention, supervision profonde.*

Abstract

Maintenance of transport infrastructure such as bridges is a crucial operation. Thus, regular and proactive inspections represent an effective strategy to prevent defects and damages. Although, inspectors used to detect anomalies by visual assessment of the bridge. Research in this area recently tends to rely on deep learning and computer vision techniques to improve the efficiency of this process. The object of the work proposed through this thesis is about segmentation task for the detection of defects in bridge inspection images. The first contribution of this work is a concrete bridge defect dataset built for semantic segmentation. The dataset contains 594 bridge inspection images annotated with masks that belong to 4 major defect classes. The second contribution is a solution of 3 multitask learning based models for bridge defect segmentation. With multitask learning paradigm, a model will capture the contextual information of all defect classes to boost its task performance and generalization. The first model is a classical U-Net network adapted for multi-class segmentation. The second is an encoder shared network with attention mechanism. The last one is an encoder-decoder shared network with deep supervision and attention mechanism. The experiment results of these models on the bridge defect dataset show that the dice score achieved is between 60% and 79%.

Mots-clés : *Bridge inspection, deep learning, semantic segmentation, multitask learning, attention mechanism, deep supervision*

Chapitre 1

Introduction Générale

1.1 Contexte

1.1.1 Inspection des ponts avec drone

L'inspection des ponts est une tâche cruciale qui rentre dans le volet de la maintenance des infrastructures de transport de chaque pays. Les effets dus aux aléas de la nature (changement de température, humidité, séismes, etc) et les charges appliquées continuellement par les véhicules, réduisent la fiabilité des ponts au fil du temps. D'où, la nécessité d'effectuer régulièrement des opérations de maintenance. Comme le nombre de ponts existant est très important, le budget alloué à cette opération est colossal. Une des approches les plus intéressantes pour réduire les coûts de maintenance est de prévenir les défauts et dommages par le biais d'inspections régulières. Traditionnellement, cette opération est conduite par des inspecteurs qui vont procéder à la vérification à l'œil nu des différentes parties du pont en quêtes de défauts et d'anomalies.

Avec l'avènement des nouvelles technologies, l'utilisation des drones dans la tâche d'inspection présente beaucoup d'avantages tant dans la réduction des coûts budgétaire de l'opération et l'efficacité de l'inspection. En effet, l'utilisation du drone, permettra une accessibilité étendue à des endroits du pont dont l'inspecteur n'a pas accès directement. Cette opération permettra aussi de réduire considérablement sa durée d'exécution. En plus, les drones offriront aussi la possibilité de collecter des données de haute qualité qui peuvent être traitées ultérieurement. En termes de sécurité de l'équipe, les drones permettront de réduire les risques d'accidents parce que l'intervention de l'homme sera réduite d'avantage dans le processus.

Cependant, l'utilisation de drone pour cette opération ne pourra pas remplacer l'expertise d'un inspecteur dans le domaine d'identification des anomalies et de défauts. Dans l'optique d'aider l'inspecteur lors de l'inspection, les techniques

issues du domaine de la vision par ordinateur représentent une solution très prometteuse qui peuvent être appliquées au flux de données collecté par les drones. Les modèles tels que la détection d'objets et la segmentation sémantique ont déjà fait leurs preuves dans d'autres domaines comme la conduite autonome de véhicule. De ce fait, ces techniques peuvent aussi être employées de plusieurs façons différentes dans l'inspection des ponts.

1.1.2 La segmentation des défauts dans les ponts

Dans le processus d'inspection, identifier les anomalies et les défauts présents dans les ponts est le premier objectif. L'expert chargé de cette tâche procède traditionnellement par une vérification visuelle des différents éléments du pont. Cependant, le succès des techniques d'apprentissage profond dans les domaines tels que la conduite autonome suggère, l'idée d'employer et d'adapter ces techniques à cette tâche de détection de défaut. Dans le domaine de la vision par ordinateur, la segmentation sémantique est une tâche qui permettra une détection des défauts avec une précision au niveau pixel. Entre autres, elle permettra ainsi d'assister l'inspecteur dans sa tâche de vérification.

Dans cette optique d'intégrer la segmentation sémantique pour la détection des défauts dans l'inspection de ponts, nous dénotons plusieurs approches pour y parvenir. La première approche intuitive consiste à faire une segmentation sémantique des défauts sans indiquer à quelle classe ils appartiennent. Une autre approche se résume à se concentrer sur un défaut en particulier. Dans la littérature relative à cette problématique, nous trouvons un bon nombre de recherches concluantes qui traitent de la segmentation des fissures. Enfin, l'approche consistant à mettre en évidence les défauts et leurs classes est plus complète comparée aux autres. En effet, les solutions qui adoptent cette approche fournissent des résultats plus détaillés et raffinées. Pour aller encore plus loin, les méthodes qui consistent à distinguer les différentes instances d'une classe de défauts est un défi encore plus poussé. Cette dernière façon de procéder est appelée : Segmentation d'instances pour les défauts.

1.1.3 La segmentation des éléments structurels du pont

L'AASHTO [1] rapporte dans leur manuel pour l'inspection des éléments structurels dans un pont [15] que la cote globale décrivant l'état d'un pont dépend de l'évaluation compréhensive des défauts en fonction de l'élément structurel auquel il appartient dans le pont [5]. L'approche consistant à identifier les différents éléments structurels du pont lors de l'inspection, constitue la première étape cruciale du processus de détection des défauts. En effet, cette étape permet de bien cerner les éléments clés du pont où il faut chercher des défauts bien particuliers. De plus, le fait de distinguer les différents éléments constituants du pont contribuera grandement à l'amélioration de la navigation du drone par le pilote, surtout dans les endroits où

1. American Association of state Highway and Transportation Officials

l'accès est plus difficile.

La solution triviale qui répond au besoin de cette étape est la mise en place d'un modèle de segmentation qui permettra d'associer chaque pixel de l'image à l'élément structurel du pont auquel il appartient. Ajouté à cela aussi le fait que les zones qui ne correspondent pas au pont seront non prises en compte plus tard dans le processus de détection de défauts. Plusieurs types de catégories peuvent être établis au préalable selon la nature des ponts à inspecter. La figure 1.1 ci-dessous montre bien un exemple de segmentation des éléments structurel d'un pont en 4 catégories (arrière plan, béton, acier, structure métallique)

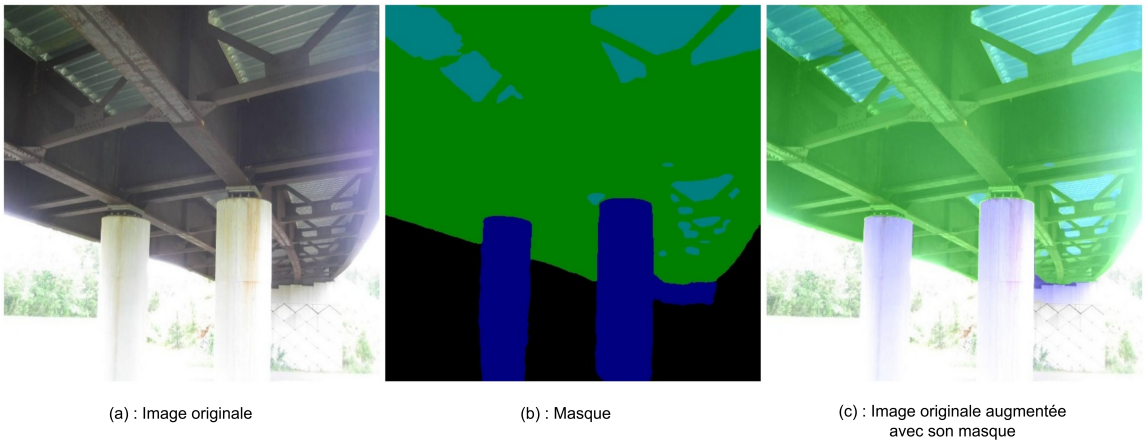


FIGURE 1.1 – Exemple d'image d'un pont avec son masque correspondant à différents éléments structurels segmentés.

1.2 Problématique

Ce travail de recherche s'inscrit comme une partie d'un projet à large spectre qui vise à apporter des solutions basées sur les techniques d'apprentissage profond et de la vision par ordinateur au processus d'inspection de pont. Ces solutions représentent un moyen d'automatiser et de moderniser ce processus tant important dans la maintenance de ce type d'infrastructure. La problématique traitée par le biais de ce travail de recherche est centrée sur la segmentation sémantique dans la détection des défauts lors du processus d'inspection des ponts. La segmentation sémantique étant une technique de haut niveau dans la vision par ordinateur. Contrairement aux techniques de classification d'images et de détection d'objets par boites englobantes, la segmentation sémantique permet de cibler les défauts et les anomalies avec une précision de l'ordre du pixel. Les images d'inspections de ponts se caractérisent avec un certain nombre d'aspects qui peuvent rendre la segmentation des défauts difficile. D'après [16], les aspects suivants sont des facteurs qui causent des difficultés dans la généralisation des modèles d'apprentissage profond relatives à cette tâche :

Non-homogénéité des surfaces en béton : Les surfaces en béton présentes dans les images d'inspections se manifestent avec des textures, des couleurs qui varient

d'un échantillon à un autre. Cela a pour effet d'augmenter la diversité de ces surfaces qui incluent généralement les défauts et les anomalies.

Complexité des classes de défauts : Les défauts présents dans les images se caractérisent avec des grandeurs de dimension variables, des degrés de gravité différents et un champ de localisation assez ouvert dans l'image. En plus, les défauts qui ont des relations de cause à effet peuvent se chevaucher entre eux. Tous ces aspects montrent la complexité des défauts.

Conditions d'acquisition des images : Les conditions d'acquisition des défauts font en sorte d'ajouter une certaine diversité aux défauts. En effet, les détails contenus dans les images d'inspection sont directement liés à la résolution, la profondeur et les angles de capture des caméras lors du processus d'acquisition. Ces détails jouent un rôle primordial dans la détection des défauts.

Annotations des images : La majorité des ensembles de données relatifs aux images d'inspection de ponts incluent des annotations dont la qualité est dépendante de certains facteurs. La subjectivité des annotations est un facteur potentiel qui peut fausser la réalité de ces dernières et ainsi induire en erreur les modèles. De même, la subjectivité peut affecter aussi la précision des annotations, ce qui impactera directement les performances des modèles. D'autre part, il existe des cas où les défauts se chevauchent, ainsi les annotations associées à ces derniers peuvent avoir des répercussions sur la qualité des annotations.

Tous ces aspects représentent des défis que nous voulons aborder à travers ce travail. Pour cela, nous avons opté d'explorer la piste associant l'apprentissage multitâche à la segmentation des défauts dans les images d'inspections de ponts. L'apprentissage multitâche est un paradigme d'apprentissage qui s'inspire du cerveau humain. Il vise à exploiter le transfert d'apprentissage positif entre plusieurs tâches lors de l'entraînement d'un réseau. Cette propriété est importante dans notre problème de segmentation de défauts du fait que les défauts peuvent être cooccurrents et avoir des liens de causalité. L'apprentissage multitâche peut bien représenter ces relations en prédisant de meilleurs masques de segmentation que si on entraînait les différents défauts de manière séparée.

La segmentation des défauts dans ce travail de recherche cible quatre classes de défauts (corrosion, armature exposée, dégradation du béton et l'efflorescence). Ce choix est motivé par le fait que ces classes de défauts partagent des aspects et des caractéristiques communes en termes de forme et de texture. Cela est un atout majeur dans la conception de réseaux de segmentation à apprentissage multitâche. Dans la littérature relative à segmentation de défauts, les cracks représentent une classe très importante en plus de celles déjà citées. Cette classe est très sensible dans le processus d'inspection. Dans le cadre de ce travail, nous n'avons pas inclus cette classe parce que les défauts de cette catégorie se caractérisent par des aspects

très particuliers contrairement aux autres. Les cracks se manifestent souvent sous une forme linéaire et fine, ils peuvent aussi être discontinus. Ces aspects nous ont emmenés à ne pas les inclure dans ce travail orienté apprentissage multitâche, car ils peuvent favoriser un transfert négatif dans l'entrainement de nos modèles et ainsi causer une perte de performance. Cette classe de défauts est traitée exclusivement dans un autre volet de ce même projet.

1.3 Objectifs

Afin de mener ce travail à terme et répondre efficacement et méthodiquement à la problématique posée dans la section 1.1.3, nous avons défini les objectifs à atteindre le long de ce travail. Ces objectifs ont pour rôle de structurer la manière d'approcher la problématique et ainsi permettre de la diviser en sous parties pour mieux cerner ces différents aspects. Les objectifs principaux de ce travail sont :

Recensement des méthodes de segmentation de défauts de ponts en béton : La première étape consiste à faire le tour des méthodes existantes dans ce domaine de recherche. Cette démarche a pour objet de nous permettre d'acquérir des connaissances sur les tendances actuelles des recherches entreprises dans la même lignée que la nôtre. De plus, l'apprentissage multitâche reste une voie assez récente et rarement empruntée pour des fins de segmentation de défauts. Ainsi, à travers cette démarche, nous pourrons recueillir des idées potentielles que nous pourrons exploiter au mieux dans l'élaboration de notre propre solution.

Construction d'une base d'images de segmentation de défauts en béton : Un des aspects le plus important de notre démarche est la mise en place d'un ensemble de données dédié à la tâche de la segmentation des défauts. En effet, cette étape est primordiale, car nous avons constaté qu'il y a un manque de disponibilité et d'accès à ce type de données . En plus, les rares ensembles de données qui incluent des annotations sous forme de masques pour les défauts présents sur les surfaces en béton ne sont pas bien annotés. D'où la nécessité de construire notre propre ensemble de données en suivant une démarche bien établie qui réduirait les biais et les erreurs d'annotation. La qualité de cet ensemble aura une incidence directe sur les performances de la solution qui sera mise place pour répondre à la problématique.

Conception d'architectures basées sur l'apprentissage multitâche pour la segmentation de défauts de béton : Ce point est l'élément central de cette recherche. Il vise dans un premier temps à concevoir différentes architectures basées sur l'apprentissage multitâche pour la segmentation des défauts. La segmentation de chaque défaut est considérée comme étant une tâche à part entière dans le modèle. Ces modèles seront entraînés et évalués sur la base d'images construite. Il est attendu que les modèles saurons tirer profit de la caractéristique fondamentale de l'apprentissage multitâche qui consiste à exploiter les relations existantes entre les tâches afin d'améliorer les performances et la généralisation des modèles. Dans cette

partie, nous nous verrons aussi incorporer des mécanismes avancés de l'apprentissage profond tels que l'attention et la supervision en profondeur dans l'optique de pousser loin les performances de la segmentation des défauts.

1.4 Organisation du document

Ce document retrace le travail effectué dans cette thèse afin de répondre à la problématique posée. Dans une perspective de permettre au lecteur de bien se situer au niveau des différents aspects inclus dans ce document, ce dernier est organisé en cinq parties. La première partie est une introduction à la problématique en mettant en avant le contexte, la problématique et les objectifs à atteindre. La deuxième partie est consacrée à l'état de l'art regroupant les différentes techniques et méthodes qui traitent la même thématique abordée dans cette recherche. Ensuite, la partie donnée est exposée en détails en partant des différentes étapes de la construction de la banque d'images jusqu'aux statistiques relatives à cette dernière. Ensuite, nous avons la partie qui inclue les différents aspects des solutions proposées pour répondre à la problématique de la segmentation des défauts dans les images d'inspections. Enfin, la dernière partie conclut tout le travail effectué en faisant le point sur les résultats obtenus par les solutions proposées en plus de proposer des perspectives d'amélioration pour ces dernières.

Chapitre 2

Revue de littérature

2.1 La segmentation sémantique

2.1.1 Introduction

L'objet principal de ce travail de recherche se rapporte directement à la tâche de segmentation en imagerie numérique. À cet effet, cette section sera consacrée spécifiquement à cette tâche. Elle abordera les concepts clés et de base relatifs à la segmentation.

La segmentation d'image est une tâche très importante dans le domaine de la vision par ordinateur. Elle consiste à partitionner une image en différents segments qui représentent des objets. Cette tâche a connu un succès considérable ces dernières années avec l'incorporation des méthodes et techniques d'apprentissage profond à travers les modèles publiés, et en particulier les réseaux de neurones convolutifs. Tandis que les tâches de classification et la détection d'objet permettent juste l'assignation de label et une localisation d'objets via des boites englobantes, la segmentation opère au niveau de chaque pixel de l'image.

2.1.2 Définition

La segmentation est une tâche de niveau intermédiaire dans la hiérarchie des techniques de la vision par ordinateur. Elle a pour objectif d'identifier les objets présents dans l'image au niveau pixel. Cette tâche traite l'image en assignant à chaque pixel une classe d'objet prédéfinie. Nous distinguons deux types de segmentations comme nous pouvons le voir dans la figure 2.1 :

La segmentation sémantique : Elle consiste précisément à la classification de chaque pixel de l'image d'entrée dans un ensemble de classe d'objet sans faire la différence entre les différentes instances d'une même classe

d'objet.

La segmentation d'instances : Contrairement à la précédente, ce type de segmentation est plus sophistiquée. En plus de la classification des pixels de l'image parmi un ensemble de classe d'objets. Cette dernière fait la différence entre les instances d'un même objet en leur assignant des étiquettes différentes.

FIGURE 2.1 – Comparaison des résultats de la segmentation sémantique et de la segmentation d'instances

2.1.3 Techniques et méthodes traditionnelles

Avant l'avènement des techniques d'apprentissage profond comme une révolution dans le domaine de la vision par ordinateur, la segmentation a longtemps reposé sur des techniques traditionnelles qui extraient les informations utiles des images. Ces techniques sont basées sur des modèles mathématiques et des mécanismes qui permettent de distinguer les régions d'une image en se basant sur leurs caractéristiques communes. Ces caractéristiques sont généralement la couleur, la texture ou la luminosité. Ces méthodes sont assez simples ne nécessitant pas beaucoup de ressources et elles sont faciles à implémenter. Parmi ces techniques, nous citerons :

Méthode de seuillage d'histogrammes

C'est une technique simple qui consiste à classer les pixels d'une image par rapport à un seuil en se basant sur l'histogramme d'intensité de l'image. Il existe deux principales techniques basées sur le seuillage. En premier, la méthode de seuillage globale qui procède à la division de l'image en deux parties : l'avant-plan et l'arrière-plan tout en se référant à un seuil fixé. Cependant, cette technique ne donne pas des résultats acceptables lorsque l'image se caractérise avec un contraste ou une luminosité variable. Pour pallier ce souci, la méthode de seuillage adaptative procède de la même manière que la précédente, à l'exception du seuil qui sera fixé localement selon les régions de l'image.

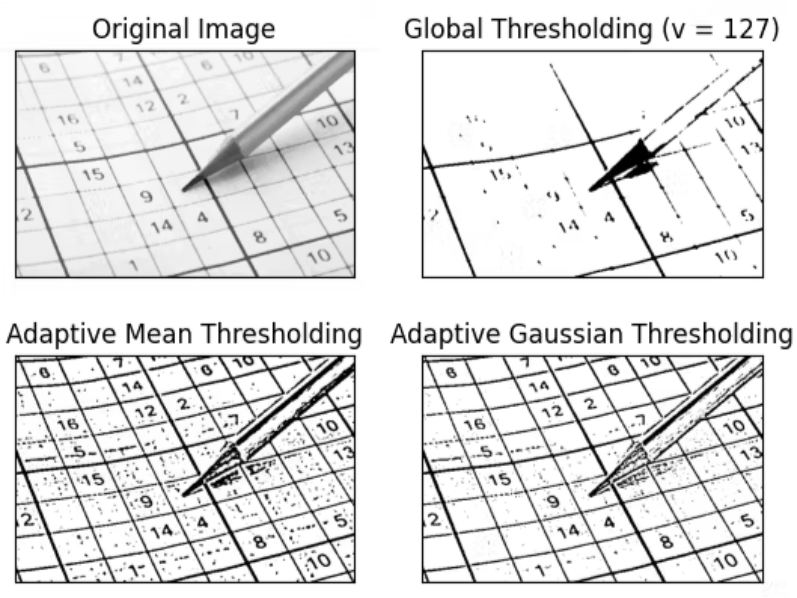


FIGURE 2.2 – Exemples des différentes méthodes de segmentation par seuillage.

Segmentation en régions

Cette approche est souvent utilisée dans le domaine du traitement d'image. Elle consiste à diviser l'image en région par rapport à un critère de similarité sur la couleur, la texture et l'intensité des pixels. Ensuite, une phase de fusion de ces régions succédera pour atteindre un niveau de segmentation désiré. Nous distinguons deux techniques majeurs dans cette catégorie : La segmentation par division et fusion qui procède en divisant l'image en petits blocs jusqu'à satisfaire un critère d'arrêt. Ensuite, les blocs seront fusionnés ensemble de façon à respecter une certaine similarité jusqu'à l'obtention d'un masque final. Cette méthode est efficace et simple, mais elle rencontre des difficultés dans les images complexes avec des chevauchements d'objets. La deuxième méthode s'intitule segmentation par graphes. Elle exploite les fondements de la théorie des graphes dans le but de représenter l'image sous la forme d'un graphe. Les nœuds du graphe sont les pixels de l'image tandis que les arêtes représentent le degré de similarité entre les pixels. Une fois cette représentation construite, la méthode enchaîne en appliquant un algorithme de partitionnement de graphe qui fonctionne en minimisant une fonction de coût. Ce dernier a pour rôle de construire les différents segments de l'image.

Segmentation par détection de contours

Cette approche est aussi issue du domaine du traitement d'image. Elle procède par l'identification des différentes frontières et la séparation de ces dernières de l'arrière-plan de l'image. Cette reconnaissance de frontière est effectuée en soulevant les variations d'intensité et des valeurs de couleurs au niveau des pixels de l'image. Cette approche inclut un ensemble de méthodes bien connues comme la détection de

frontière par filtre de Canny et par filtre de Sobel, en plus de la détection par le Laplacien. D'autres méthodes ont utilisé les contours actifs pour localiser les frontières des objets.

Segmentation par regroupement (clustering)

Cette approche repose sur les algorithmes de partitionnement. Ces derniers procéderont au regroupement des pixels de l'image en groupes en se basant sur des critères de similarité. Parmi les algorithmes les plus répandus de cette approche, nous avons le regroupement par K-moyennes, les méthodes de regroupement hiérarchiques, les modèles de mélanges Gaussiens, les coupes de graphes, etc.

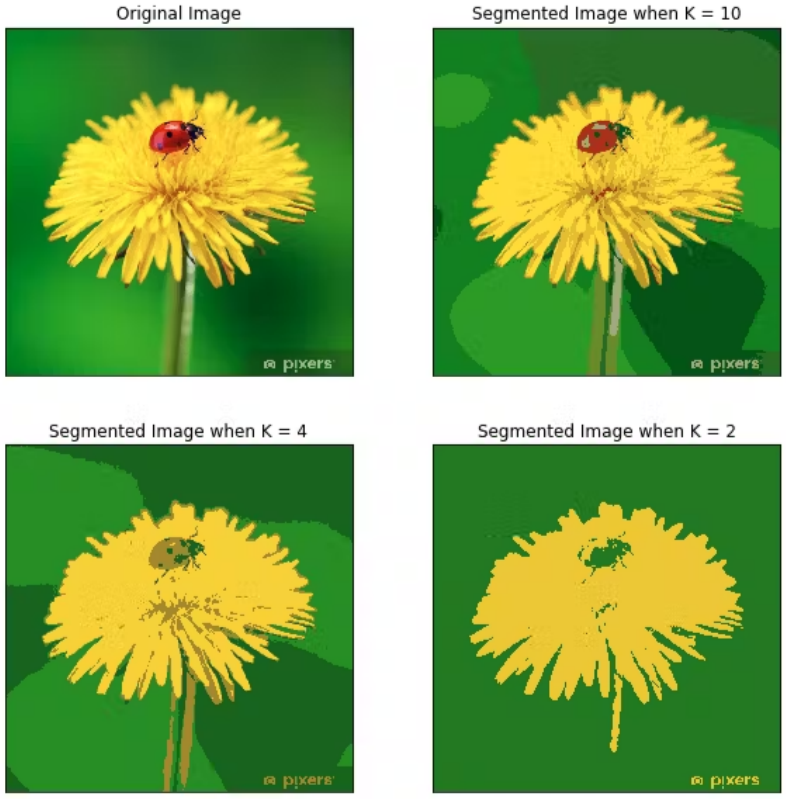


FIGURE 2.3 – Exemple d'une segmentation par partitionnement avec l'algorithme k-moyen.

2.1.4 Techniques basées sur l'apprentissage profond

Ces dernières années, le succès grandissant des techniques d'apprentissage profond dans le domaine de la vision par ordinateur a fait qu'une nouvelle génération de modèles pour la segmentation d'image s'est démarquée par les performances et l'efficacité qu'elle a su apporter.

Fondements

Avant d'aborder les bases et les concepts clés de segmentation, nous devons rappeler l'origine de cette tâche qu'est la classification d'images. La classification d'images est considérée comme le point de départ de tous les problèmes relatifs au domaine de la vision par ordinateur. Cette tâche permet de faire le lien entre une image avec une étiquette prédéfinie au préalable. Comme toutes les tâches de la vision, la classification a aussi bénéficié de la révolution et de la démocratisation des techniques d'apprentissage machine et profond ces dernières années. Nous pouvons constater que les modèles de classification d'images les plus aboutis dans le domaine sont ceux qui adoptent des architectures dites à réseaux de neurones profondes et plus précisément les réseaux de convolution.Les réseaux de neurones convolutifs sont un type de réseau de neurones profond qui est composé principalement de deux compartiments comme illustré dans la figure 2.4. Le premier est l'extracteur de traits et d'attributs. Le deuxième, c'est le classificateur. Dans ce type de réseau, nous trouvons généralement quatre types de blocs :

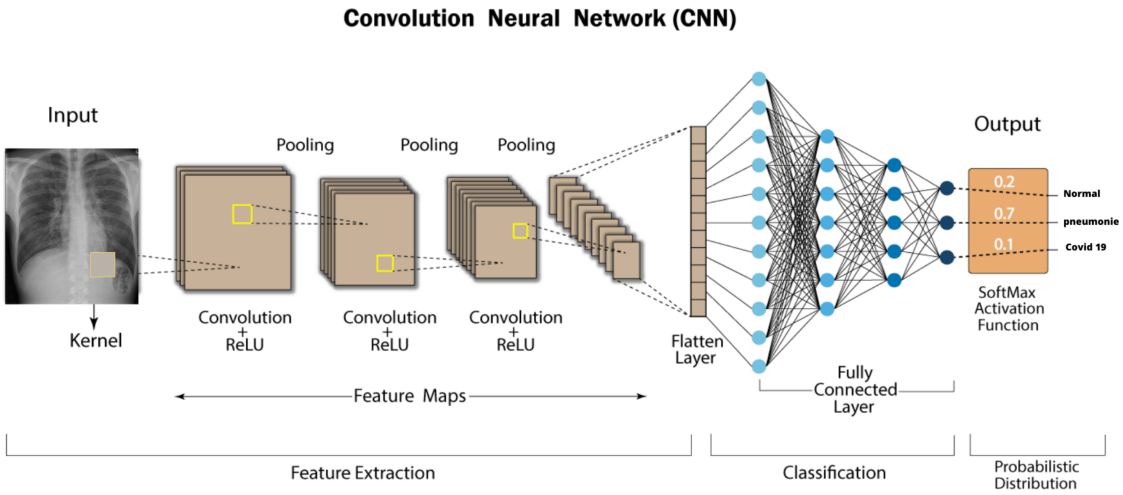


FIGURE 2.4 – Architecture type d'un réseau de neurones convolutif.

Couche de convolution : la convolution est une application d'un filtre sur une représentation de données où nous obtenons une activation des traits et des aspects bien spécifiques de la représentation. Cette couche comporte un ensemble de filtres qui sont entraînables. Ces filtres ont la particularité d'avoir des champs réceptifs restreints par rapport à la représentation d'entrée. En résumé, cette opération permet d'extraire les aspects clés des images de départs qui serviront pour la prédiction finale.

Couche de pooling : Elle a pour rôle de compresser les représentations intermédiaires tout en gardant l'information utile à la prédiction finale. Cela dans le but d'alléger le flux de données qui est traité au niveau du réseau. Cette couche applique en général une des techniques du sous-échantillonnage.

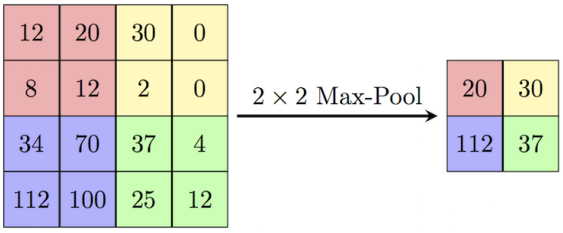


FIGURE 2.5 – Illustration de l'opération de pooling.

Unité Relu : C'est une fonction d'activation, communément appelée unité linéaire rectifiée, dont le rôle est de minimiser le sur-apprentissage du modèle. De ce fait, elle contribue grandement à réaliser des résultats plus précis.

Couche entièrement connectée : C'est une couche qui se caractérise par le fait que ses unités de sorties sont connectées à toutes les unités de la couche qui la précède. Elle se présente comme la couche finale du réseau et prend en entrée une représentation de données aplatie pour calculer la prédiction finale.

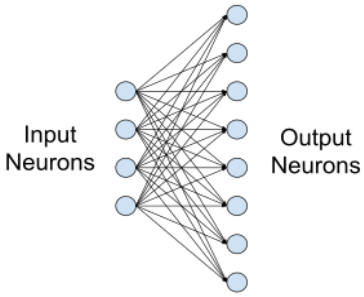


FIGURE 2.6 – Illustration de la structure de la couche entièrement connectée.

Dans la littérature relative à ce champ de recherche, nous avons des classificateurs qui sont la référence dans cette tâche de classification d'Images. Parmi ces réseaux, nous citons : VGG[17], Inception [18], ResNet[13] et EfficientNet [19].

Architecture Encodeur-Décodeur Dans le domaine de l'apprentissage machine et profond, l'architecture de type **Encodeur-Décodeur** est considérée comme un pilier fondamental dans la conception de certains types de modèles. Plus précisément, les domaines du traitement naturel du langage et de la vision par ordinateur sont les champs d'application les plus réputés pour adopter ce type d'architecture dans leurs réseaux. Comme son nom l'indique, cette architecture est constituée de deux modules principaux comme l'illustre bien la figure 2.7. Dans un contexte général, l'encodeur prend en entée une séquence de données pour générer ensuite une représentation

intermédiaire sous une forme de vecteur à taille fixe que l'en nomme "représentation latente". Cette représentation latente est compressée et elle est conçue pour capturer les informations les plus importantes de la séquence d'entrée. Ensuite, le décodeur procède au traitement de cette dernière afin de générer une représentation finale basée principalement sur la représentation latente.

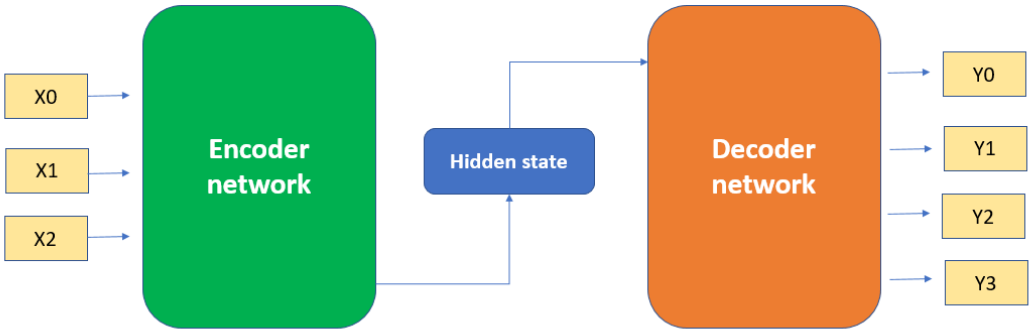


FIGURE 2.7 – Architecture de type Encodeur-Décodeur.

Dans le contexte de la vision et plus précisément de la segmentation, l'encodeur a pour rôle l'extraction des caractéristiques essentielles de l'image d'entrée par le biais d'un ensemble de filtres. Puis, le décodeur génère le masque final correspondant à l'image d'entrée en s'appuyant sur les caractéristiques extraites par l'encodeur.

Parmi les modèles basés sur les techniques d'apprentissage profond qui ont marqué la tâche de segmentation sémantique, nous listons ci-dessous les réseaux qui ont eu un impact considérable dans le domaine :

FCN

En 2014, [1] ont publié un modèle pour la segmentation sémantique qui s'intitule FCN [1]. Le modèle reprend les bases d'un réseau de neurones convolutif classique avec le remplacement de la couche finale entièrement connectée par une couche de sur-échantillonnage. À la base, la couche entièrement connectée est utilisée pour la prédiction des résultats dans la tâche de classification et de détection d'objets à partir des résultats des couches de convolution. Le choix de remplacer cette couche par celle de sur-échantillonnage est motivé par la préservation des dimensions des caractéristiques le long du réseau dans l'optique de produire un masque de segmentation qui

1. Fully Connected Network : Réseau de neurones entièrement connecté

correspond à l'image d'entrée. Par conséquence, l'avantage de cette modification est que l'image d'entrée n'est plus restreinte à une dimension fixe, qui est une contrainte imposée par la couche entièrement connectée.

L'architecture du FCN se distingue par deux compartiments : la partie sous-échantillonnage (Encodeur) suivie de celle de sur-échantillonnage (Décodeur) comme l'illustre bien la figure 2.8.

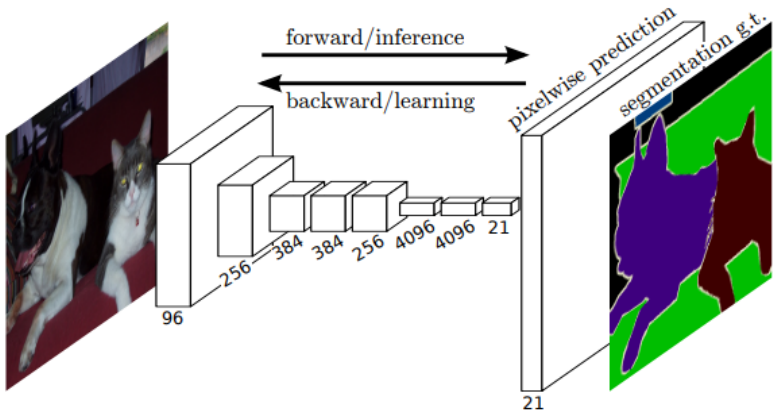


FIGURE 2.8 – Aperçu général de l'architecture du réseau FCN. [1]

La partie encodeur se compose d'une série de couches de convolution et de pooling dont le rôle est l'extraction de caractéristiques complexes en réduisant leur dimension pour une meilleure interprétation. La partie sur-échantillonnage est constituée de couches de convolution transposée dont le rôle est la reconstitution des caractéristiques de dimension plus grande à partir de celles qui précèdent et dont la dimension est plus petite tout en persévérant les informations pertinentes. Dans cette partie, la localisation spatiale des objets s'effectue. Pour atteindre des résultats efficaces dans ce processus, l'ajout des sauts de connexions [2] entre les couches des deux parties contribue à la restauration des informations spatiales perdues au niveau de la première partie du modèle.

U-Net

Le réseau U-Net est proposé par [2] en 2015. Il vient pour apporter des améliorations au modèle FCN [1]. À la base, ce modèle est destiné pour la segmentation d'images biomédicales.

L'architecture de ce modèle est illustrée dans la figure 2.9 et adopte aussi le même paradigme Encodeur-Décodeur. Elle se compose de deux chemins. La première partie (à gauche) est décrite comme un chemin contractant. Elle reprend les mêmes couches que pour un réseau de convolution standard, à savoir une succession

2. Des connexions raccourcis qui permettent d'acheminer des données entre des couches non adjacentes dans un réseau de neurones

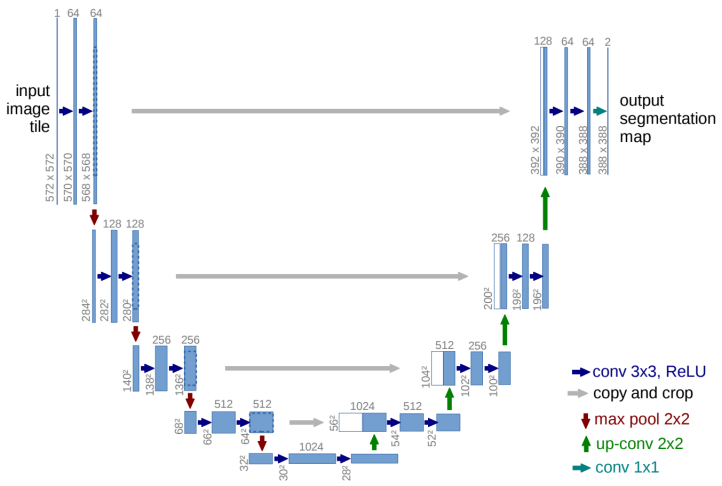


FIGURE 2.9 – L'architecture du réseau U-Net. [2]

de couche de convolution suivie d'une unité linéaire rectifiée (ReLU) [3] et de l'opération de max pooling. L'objectif de cette partie reste l'extraction des caractéristiques pertinentes qui constituent le contexte de l'image.

La partie expansive (à droite) se compose principalement de couches de convolution disposées d'une manière ascendante pour augmenter la résolution des caractéristiques au fil du processus. Comme dans [1], la localisation des formes de haute résolution se fait à partir de la combinaison (concaténation) des caractéristiques de chaque niveau dans les deux parties respectivement via des sauts de connexion. Le résultat de cette combinaison sera ensuite traité par une série de couches de convolution qui a pour but d'apprendre à construire des caractéristiques très précises [2].

L'autre particularité de ce réseau est que dans la partie expansive, les caractéristiques ont un nombre important de canaux. Par conséquence, le réseau propage les informations sur le contexte dans les couches de haute résolution. Ce qui induit que la partie expansive est plus au moins symétrique à la partie contractante et d'où une architecture sous forme d'un "U" [2].

DeepLab

Le DeepLab est un modèle à code source ouvert conçu pour la segmentation sémantique par l'équipe de recherche des laboratoires de Google. Ce réseau adopte aussi le paradigme architectural de type Encoder-Décodeur.

Parmi les défis auxquels le FCN [1] fait face, la réduction de dimension des caractéristiques lors du processus de convolution et pooling qui engendre une perte

3. Une fonction mathématique utilisée souvent dans les réseaux de neurones comme fonction d'activation.

d'information capitale. Les masques de segmentation produits sont d'une résolution réduite et les frontières entre les objets sont confuses. La famille DeepLab s'est penché sur le défi précédant en introduisant une nouvelle technique de convolution *" Convolution atrous"* qui est un module qui permet une segmentation d'objet à différente échelle de grandeur s'intitulant *"Atrous Spatial Pyramid Pooling"*.

La convolution atrous : Des solutions présentées par [2, 1], la convolution transposée est le moyen employé dans la partie décodage du réseau afin de sur-échantillonner les caractéristiques. Cependant, cette technique requiert une capacité de calcul et de mémoire importante. La solution proposée pour palier à ce problème dans [3] est la convolution atrous. Cette technique illustrée dans la figure 2.10 permet le calcul des caractéristiques avec une résolution spatiale que l'on désire [3]. La convolution atrous suggère d'étendre le champ de vision des filtres pour la capture d'un contexte plus large sans augmenter le nombre de paramètres ni la complexité de calcul. Cela est possible via l'insertion de zéros dans les filtres conventionnels, d'où l'appellation *"atrous"* .

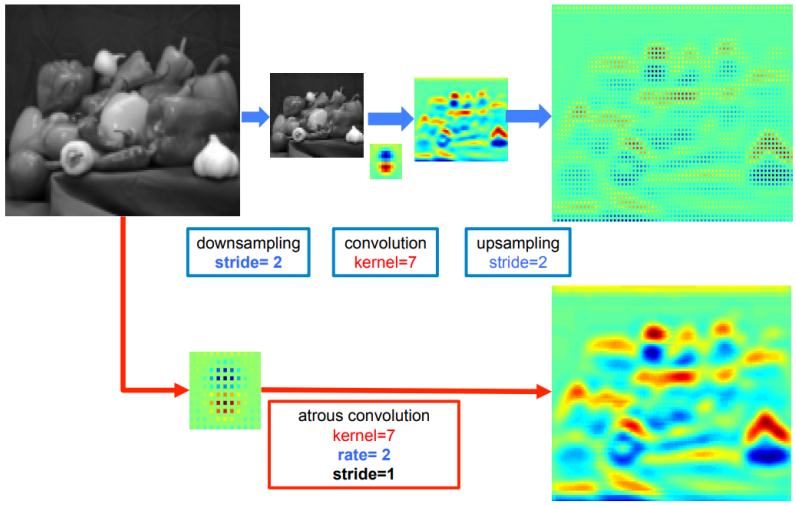


FIGURE 2.10 – Illustration de l'opération de convolution atrous sur une image 2D.[3].

Atrous Spatial Pyramid Pooling : Ce module s'inspire du travail de [20]. Le but étant d'améliorer la reconnaissance d'objets de la même classe, mais qui diffère en termes d'échelle de grandeur. L'idée est d'appliquer à la caractéristique d'entrée une multitude d'opérations de convolution atrous avec des paramètres d'échantillonnage différents comme illustré dans la figure 2.11. Les résultats de chaque opération seront ensuite fusionnés pour former une nouvelle caractéristique qui contribuera à une meilleure appréhension d'objet d'une même classe sous différentes échelles de grandeur.

DeepLabv1 : La première version DeepLabv1 [3] apporte des améliorations aux nombreux défauts des modèles déjà existants et plus spécifiquement le FCN. Pour palier au problème de la perte d'information lors du processus de sous-échantillonnage,

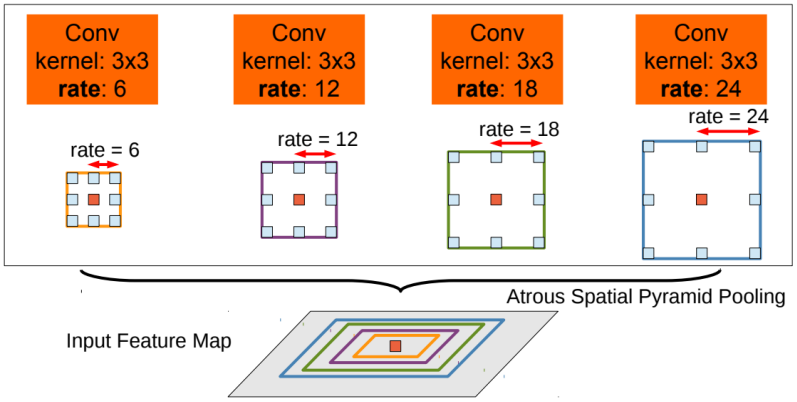


FIGURE 2.11 – Illustration du processus effectué par le module "atrous spatial pyramid pooling"[3].

les dernières couches de max pooling du décodeur ont été remplacées par les couches de convolutions atrous. De plus, ce modèle intègre un champ aléatoire conditionnel entièrement connecté afin de produire des masques de segmentation avec des détails très raffinés. En résumé, le DeepLabv1 prend une image d'entrée et la passe à travers les couches conventionnel d'un réseau profond de convolution suivi d'une ou deux couches de convolutions atrous. La caractéristique résultante sera ensuite surdimensionnée par le biais de l'opération d'interpolation bilinéaire. Enfin, le champ aléatoire conditionnel entièrement connecté procédera à l'affinement des résultats de segmentation comme illustré dans la figure 2.12.

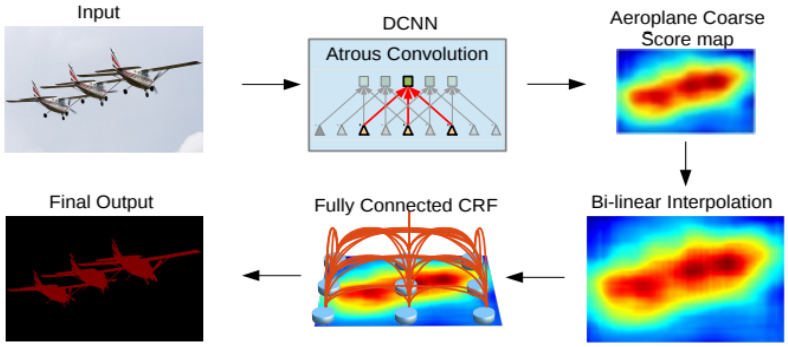


FIGURE 2.12 – Illustration des pricipales étapes consittuantes du modèle DeepLabv1 [3].

DeepLabv2 : Pour améliorer l'architecture du modèle précédent, il fallait se concentrer sur le défi relatif à la gestion d'un objet de la même classe qui peut se trouver sous différentes échelles de grandeur dans l'image. La solution est donc l'intégration du module *"atrous spatial pyramid pooling"* dans l'architecture antérieure. Pour cela, après l'extraction des caractéristiques via la partie encodage, une multitude d'opérations de convolution atrous avec différentes fréquences seront appliquées sur la

caractéristique. Les résultats seront ensuite fusionnés par le biais du module ASPP [4] tel qu'illustré à la figure 2.11.

DeepLabv3 : La version DeepLabv3 reprend les bases concernant l'utilisation de la convolution atrous qui permet l'ajustement des champs réceptifs des filtres et un contrôle sur la résolution désirée des caractéristiques [4]. Le modèle adopte une architecture de type Encodeur-décodeur qui s'appuie sur la convolution atrous mise en place en cascade pour la capture d'un contexte à différentes échelles. Ensuite, le module *"atrous spatial pyramid pooling"* proposé dans [3] est amélioré en lui incorporant des caractéristiques à contexte global pour un traitement plus efficace comme illustré dans la figure 2.13. Toutes les améliorations apportées dans ce modèle font en sorte que ce dernier surpasse les deux versions précédentes.

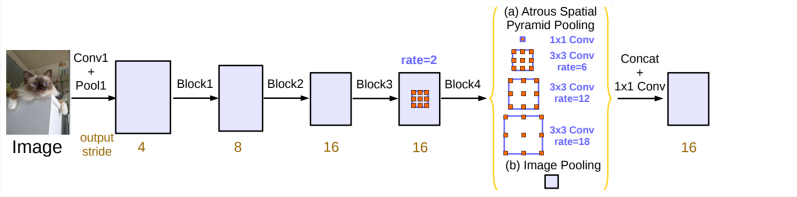


FIGURE 2.13 – Illustration de l'emploi du module ASPP amélioré par une caractéristques à contexte global dans le réseau DeepLabv3 [4].

DeepLabv3+ : Ce réseau est proposé par [21], il reprend le modèle précédent [4] avec l'ajout d'un décodeur plus efficace. Ce dernier a pour rôle d'affiner au mieux les résultats de la segmentation et plus précisément au niveau des frontières des objets. L'architecture de ce modèle est illustrée dans la figure 2.14.

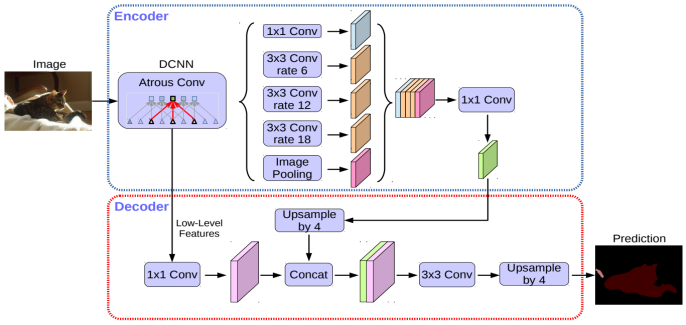


FIGURE 2.14 – Architecture du modèle DeepLabv3+ .

2.1.5 Métriques d'évaluation

Pour évaluer les performances d'un modèle, nous avons besoin de métriques qui interprètent au mieux les résultats de ce dernier. Dans la littérature relative à l'apprentissage profond et la vision par ordinateur, nous retrouvons une panoplie de métriques qui diffèrent entre elles. Dans le cas de la segmentation, nous énumérons dans ce qui suit les métriques les plus utilisées et les plus adaptées pour cette tâche.

4. Atrous Spatial Pyramid Pooling

Précision

Cette mesure est souvent utilisée dans le domaine de reconnaissance de formes et de la segmentation. Elle permet de quantifier la proportion de pixels correctement classés comme positifs (vrais positifs) par rapport à l'ensemble des pixels classés comme positifs (vrais positifs et faux positifs). L'équation de cette mesure est donnée comme suit :

$$Prcision = \frac{VP}{VP + FP}$$

Rappel

Cette mesure (aussi appelé sensibilité ou taux de vrais positifs) est fréquemment associée à la précision par conjonction afin de mesurer les performances des modèles de segmentation avec pertinence. Cette mesure reflète la capacité du modèle à détecter correctement tous les pixels appartenant à la classe cible dans l'image, par rapport à la vérité de référence. L'équation de cette mesure est donnée comme suit :

$$Rappel = \frac{VP}{VP + FN}$$

Exactitude

C'est une métrique assez simple et basique. Elle est le résultat du ratio des pixels correctement classifiés par le modèle par rapport au nombre total de pixels composant l'image. Mathématiquement, La métrique se définit comme suit :

$$Exactitude = \frac{VP + VN}{VP + VN + FP + FN}$$

Néanmoins, cette métrique présente un inconvénient dans le cas où notre modèle est confronté à des données qui comporte des classes qui domine en termes de pixel dans l'image. Pour exemple, si nous avons un jeu de données avec la classe "arrière-plan" qui occupe 90% de l'image, alors l'évaluation de cette métrique sera élevée et cela ne reflète pas la réalité des autres classes.

Intersection sur Union

L'intersection sur union (ou bien **l'indice de Jaccard**) est une métrique très utilisée dans la tâche de segmentation dans le domaine de la vision par ordinateur. Cette métrique vise à quantifier le degré de chevauchement (Similarité) entre la région prédite par le modèle et celle de la réalité terrain. La formule mathématique de IoU [5] est définie comme suit :

$$IoU = \frac{Aire\ de\ chevauchement}{Aire\ d'union}$$

5. Intersection over Union

Dans le cas d'une segmentation avec plusieurs classes d'objets, la valeur globale de l'IoU est calculée en prenant la moyenne de la métrique pour chaque classe. Contrairement à la métrique précédente, l'IoU permet une interprétation plus effective des résultats.

Intersection sur Union pondérée

C'est une extension de la métrique précédente IoU, elle a pour objectif de mieux quantifier des modèles qui sont entraînés sur des jeux de donnée qui présentent un déséquilibre en termes de classe. Au lieu de faire la moyenne des IoU de chaque classe, cette métrique suggère d'associer à chacune d'elle une fréquence qui représente le taux d'apparition de cette classe dans le jeu de données en entier. Ensuite, la valeur finale de la métrique sera la moyenne pondérée des IoU de chaque classe.

Coefficient de Dice

Communément appelé aussi l'indice de Sørensen-Dice, cet indice est utilisé dans la segmentation afin de mesurer le degré de similarité entre le masque prédit A par le modèle et le masque de la réalité terrain B. Ce coefficient est très intéressant, car il est sensible à la fois à la précision et au rappel de la segmentation. Il prend en compte à la fois les faux positifs et les faux négatifs, offrant ainsi une évaluation équilibrée de la qualité de la segmentation. L'équation de calcul de cet indice est donnée comme suit :

$$Dice = \frac{2\times \mid A \cap B \mid}{\mid A \cup B \mid}$$

2.1.6 Fonctions de perte

Dans le jargon des modèles et des réseaux de neurones, la fonction de perte est cruciale dans le processus d'optimisation des paramètres du réseau lors de la phase d'entraînement. Le choix de cette fonction dépend aussi de la tâche pour laquelle le réseau est conçu. Dans cette section, nous aborderons les différents types de fonctions de perte les plus utilisées pour la tâche de segmentation.

Fonction d'entropie croisée

À la base, c'est une fonction qui mesure la différence entre deux distributions de probabilité d'une variable aléatoire ou d'un ensemble d'événements. L'entropie croisée est construite sur l'idée d'entropie qui est issue de la théorie de l'information. Elle agit sur les pixels individuellement en calculant l'entropie croisée entre les valeurs prédites par le modèle et celles de la réalité terrain en appliquant la formule suivante :

$$H(P,Q) = -\sum_{i \in Classes} P(i) \log Q(i)$$

P : la distribution de probabilité de la réalité terrain des classes pour le pixel.
Q : la distribution de probabilité prédite par le modèle des classes pour le pixel.

Finalement, le résultat global de la fonction sera la moyenne des quantités précédentes de chaque pixel. Cependant, en suivant ce mécanisme, l'importance d'apprentissage correspondant à chaque pixel dans l'image est similaire. Or, lorsque nous sommes face à des jeux de données non équilibrés, cela présentera un problème pour l'entraînement qui sera dominé par les classes les plus représentées. À ce propos, [1] s'est penché sur la possibilité de pondérer cette fonction pour y remédier à ce problème. Pendant de temps, [2] ont proposé une méthode de pondération pour la fonction de perte qui stipule d'attribuer des poids élevés au niveau des pixels constituants les frontières d'objets. Cela a permis au U-Net de segmenter les cellules d'une manière à pouvoir les discerner individuellement dans les masques de segmentation.

La fonction de Dice

C'est une fonction de perte qui est construite sur la base des coefficients de Sørensen-Dice. Elle a été proposée pour la première fois par [22] dans leurs travaux concernant la segmentation des images médicales volumétriques. Cette fonction quantifie le degré de chevauchement entre les masques de la réalité terrain et celui prédit. Elle prend ces valeurs dans l'intervalle $[0, 1]$ et l'objectif est de la maximiser. Dans la pratique, cette quantité est calculée pour chaque classe, ensuite la moyenne de toutes ces classes est prise pour être la valeur globale de la fonction de perte.

2.1.7 Conclusion

Cette section fait office d'introduction au domaine de la segmentation sémantique d'image. En effet, dans l'optique de bien aborder le contexte de la segmentation dans l'inspection des ponts en béton, il est judicieux de poser les fondements de cette tâche de la vision par ordinateur. Dans un premier temps, nous avons posé la définition concrète de cette tâche. Ensuite, nous avons rapporté les détails pertinents des différentes architectures et modèles qui ont eu des succès considérables dans ce domaine durant cette dernière décennie. Enfin, nous avons mis l'accent sur les différentes métriques d'évaluation et fonctions de perte qui sont adoptées dans cette tâche de segmentation. La section suivante sera consacrée aux différents travaux réalisés dans le contexte de la segmentation associée à la tâche d'inspection de ponts en béton.

2.2 La segmentation dans l'inspection de ponts

2.2.1 Introduction

L'inspection de ponts est une tâche importante dans le processus de maintenance des infrastructures civiles d'un pays. Récemment, cette tâche tend de plus en plus à être automatisée et à gagner en performance en termes du temps et de la qualité de l'inspection. Cela est rendu possible grâce aux différentes avancées technologiques réalisées dans les domaines de la robotique et de la vision par ordinateur. En effet, l'utilisation des drones dans cette tâche est une tendance très prisée, car cela permet un accès aisé à toutes les zones du pont et une réduction considérable du temps de l'inspection. Par ailleurs, les modules de la vision par ordinateur tels que les détecteurs d'objets et les modèles de segmentation permettent une inspection pointue en mettant en évidence les différentes anomalies présentes au niveau des ponts.

Pour la tâche de segmentation, elle peut être intégrée dans le processus d'inspection de ponts dans différents contextes. La première approche se situe dans un contexte d'un système d'inspection totalement automatisé. Ce contexte entièrement automatisé, où la présence humaine n'est pas requise, mettra la tâche de segmentation au centre du système, à l'instar du rôle qu'elle occupe déjà dans le domaine de la conduite autonome de véhicules. Les deux principaux volets relatifs aux drones seront concernés : la navigation et l'identification de défauts seront ainsi assurés via les modules de segmentation sémantique très performants. Une autre approche est aussi très prometteuse dans le cas d'un contexte d'un système semi-automatisé. La présence humaine dans la boucle est requise dans ce cas et la segmentation servira comme assistante pour le pilotage au pilote et comme moyen d'aide à l'identification d'anomalies à l'inspecteur.

Dans cette section, nous nous intéresserons à la segmentation dans la tâche d'inspection des ponts. Au début, nous discuterons des volets de l'inspection dans lesquels la segmentation joue un rôle important. Ensuite, nous ferons le tour des travaux majeurs réalisés dans ce domaine. Enfin, nous conclurons avec les défis rencontrés dans cette approche.

2.2.2 Travaux existants

Segmentation des défauts

La segmentation est une tâche de la vision par ordinateur qui a connu une avancée très importante dans la dernière décennie grâce aux techniques et méthodes de l'apprentissage profond. Ces dernières années, les recherches et les travaux entrepris dans le domaine de l'inspection des ponts se sont penchés de plus en plus sur les modèles de segmentation. L'emploi de ces modèles de segmentation pour une détection de défauts au niveau pixel constitue un défi prometteur.

Parmi les recherches entreprises dans ce domaine, les travaux de [23, 24] témoignent de la réussite de l'utilisation du réseau HRNet [25] dans la détection et la segmentation des défauts de surfaces. Le réseau HRNet est un réseau de neurones convolutif dont la particularité est de lier les convolutions de résolution hautes à celle de basse résolution de manière parallèle. Rubio et al. [26] et Dunget al. [27] ont réussi à utiliser le réseau FCN [1] pour la segmentation des anomalies se trouvant sur le tablier [6] des ponts (délaminage, armature exposée et les fissures de béton). Pan et al. [28] ont développé le réseau PipeUNet qui reprend le réseau U-Net [2] et le mécanisme d'attention pour segmenter les défauts des conduits d'égouts (fissures, infiltration, décalage des jointures, etc.). Wang et Cheng [29] ont proposé le réseau DilaSegCRF qui reprend le réseau PSPNet [30] pour la segmentation des défauts des conduits d'égouts. Shi et al. [31] ont contribué à la segmentation des corrosions métalliques et des fissures en s'appuyant sur le réseau U-Net ([2] . Pour la segmentation d'instances, nous avons rapporté que d'après [5], le réseau Mask R-CNN [32] est utilisé pour la segmentation des fissures sur les ponts en béton [33], et des défauts de façades [34].

Segmentation des éléments structurels

Dans le domaine de l'inspection automatique ou semi-automatique des ponts, la reconnaissance des différentes parties du pont est une tâche centrale de ce processus. En effet, la segmentation est une approche qui permettra de répondre au mieux à cette problématique. Cependant, d'après [5], les travaux et recherches effectuées dans ce champ n'ont pas atteint des résultats satisfaisants tant attendus en comparaison aux autres domaines. Cela est dû principalement à la complexité de la tâche en elle-même comparée à la segmentation des défauts. Plus encore, le manque de disponibilité des banques d'images d'inspection des ponts est un facteur qui frein le développement de cette tâche.

Dans la littérature relative à cette tâche, le travail de Yeum et al. [35] propose une méthode basée sur un réseau de neurone convolutif pour la localisation et la classification des régions d'intérêts dans les images d'inspection. Narazaki et al. [36] ont contribué dans les méthodes de segmentation pour l'automatisation de l'inspection des ponts en proposant une solution de segmentation sémantique des éléments d'un pont. La méthode combine la classification de scène et la segmentation traditionnelle pour des résultats consistants. Bianchi et al.[37] a utilisé le modèle SSD v2 [38] pour la détection des éléments que l'on trouve le plus souvent au-dessous des ponts métalliques. Karim et al. [39], s'est appuyé sur la technique de transfert d'apprentissage pour entraîner le réseau Mask R-CNN pour une segmentation multi-classes des éléments structurels des ponts. D'autre part, Zhang et al. [40] ont mené une étude qui démontre que HRNetV2-W32 est le réseau le plus convenable pour l'extraction des caractéristiques profondes relatives à la segmentation multi-classe des éléments de structure de ponts. Yu et al. [41] ont mené une investigation sur la segmentation et la reconnaissance des éléments des ponts. Ils se sont appuyé sur le modèle de segmentation d'instance Mask R-CNN qu'ils ont entraîné sur un ensemble

6. C'est une structure porteuse qui supporte les charges du trafic routier

de données de 800 images.

Approche par apprentissage multitâche

L'apprentissage multitâche est un sous-domaine de l'apprentissage machine. Ce paradigme d'apprentissage se fait en parallèle sur des tâches liées entre elles. Il exploite pour cela au mieux les aspects communs et différents de ces dernières afin de permettre au modèle de réaliser une meilleure généralisation comparé à un apprentissage individuel pour chaque tâche. Ainsi, chaque caractéristique apprise pour une tâche peut contribuer à aider l'autre tâche [42].

Dans les travaux reportés dans les deux sections précédentes, chaque tâche est traitée indépendamment de l'autre. Zhang et al.[5] ont exploré une autre approche qui adopte l'apprentissage multitâche pour les tâches de segmentation des éléments structurels et celle de la segmentation des défauts pour les images d'inspections. En effet, les auteurs ont noté la corrélation existante entre les éléments structurels et le type de défauts qui s'y trouve dessus. Cela a fait en sorte de les motiver afin de mettre en place un réseau 2.15 basé sur un apprentissage multitâche qui était entraîné sur une banque d'images contenant à la fois des annotations sur les éléments structurels et des défauts (corrosion). Les résultats obtenus ont démontré l'avantage de cette approche vis-à-vis des approches standards ou chaque tâche est traitée par un réseau indépendant.

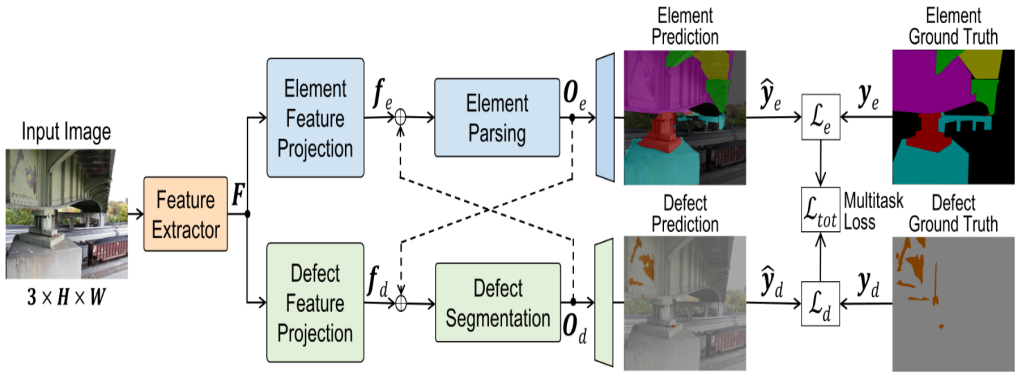


FIGURE 2.15 – Architecture du réseau à apprentissage multitâche [5].

Le tableau ci-après est un récapitulatif qui résume l'ensemble des méthodes de segmentation abordées dans cette section :

TABLE 2.1 – Tableau récapitulatif des méthodes et modèles de segmentation pour les tâches d'inspections de pont

	Modèle	Année	Type d'annotation	Type de tâche
Segmentation de défauts	HRNet[23, 24]	2021	Défauts de surface	Seg sémantique + Détection
	FCN[26]	2019	Armature exposée + Délaminages	Seg sémantique
	FCN[27]	2019	fissures	Seg sémantique
	PipeUNet[28]	2020	Défauts des conduits d'égout	Seg sémantique
	DilaSegCRF[29]	2020	Défauts des conduits d'égout	Seg sémantique
	VGG-UNet[31]	2021	Corrosion + fissures	Seg sémantique
	Mask R-CNN[33]	2020	Fissures	Seg d'instance
	Mask R-CNN[34]	2022	Défauts de façade	Seg d'instance
Seg. éléments	Multi-scale CNN[36]	2020	Éléments du pont	Seg sémantique
	SSD v2[37]	2021	Éléments des ponts en métal	Seg sémantique
	Mask R-CNN[39]	2022	Éléments du pont	Seg sémantique + Détection
	Mask R-CNN[41]	2022	Éléments du pont	Seg d'instance
MTL	Multitask Deep learning[5]	2022	Corrosion + Éléments du pont	Seg sémantique

2.2.3 Conclusion

Dans cette section, nous avons fait le tour des travaux et contributions réalisées dans le cadre des tâches de segmentations de défauts et d'éléments structurels dans l'inspection des ponts. Nous avons constaté que ce domaine d'inspection est de plus en plus convoité ces dernières années par des approches basées sur la segmentation sémantique et d'instances. Cela dans l'objectif d'atteindre des résultats précis qui permettront d'automatiser le processus d'inspection dans sa globalité. Néanmoins, il reste beaucoup de défis à franchir avant d'atteindre cet objectif. Le manque de données accessibles et annotées est un obstacle majeur dans cette quête comme il a été mentionné dans [5]. Ensuite, il faut adapter les modèles et réseaux de segmentation de pointe actuelle aux caractéristiques complexes de ces tâches en explorant différentes approches telles que celle par l'apprentissage multitâche [5].

Dans la partie suivante, nous allons aborder la méthodologie du projet de recherche. Nous mettrons en lumière les différents aspects de l'approche que nous avons adoptée durant ce projet. Cette section permettra au lecteur de bien cerner le but visé par notre approche en justifiant au mieux notre démarche tout le long de ce travail afin de mieux répondre à la problématique posée au départ.

Chapitre 3

Ensemble de données

3.1 Introduction

Ce chapitre se consacre au processus de construction d'un ensemble de données dédié à la tâche de la segmentation des défauts dans les images d'inspection de ponts en béton. Ce processus est constitué d'un ensemble d'étapes bien définies que nous aborderons en détails par la suite. En raison du manque de disponibilité et d'accès à des banques d'images d'inspection de pont, nous avons fixé pour objectif premier de construire une base d'images annotées pour la segmentation des défauts au niveau des ponts. Cette démarche débute par la phase d'acquisition des images puis s'enchaîne avec l'annotation de ces dernières. Une fois l'ensemble des images annoté, la répartition de ces dernières en sous ensemble est capitale. La dernière étape consiste à tester un réseau de segmentation sur notre ensemble de donnée afin d'évaluer sa qualité.

3.2 Acquisition et annotation

Dans cette section, nous décrivons les différentes étapes de la création d'une base d'images d'inspection de pont pour la segmentation des défauts.

3.2.1 Source d'images

L'ensemble de données CODEBRIM [6] constitue la principale source d'images pour notre base d'images. CODEBRIM [6] intègre une collection d'images qui regroupe cinq catégories de défauts que nous retrouvons sur les surfaces en béton des ponts (fissures, efflorescence, corrosion, barre exposée et éclatement). Les images sont issues de 30 ponts différents. Ils sont choisis en fonction des détériorations présentes, étendues des défauts, leur sévérité et la diversité des surfaces mises en avant. Les images ont été prises dans des conditions météorologiques variables avec des multi-

FIGURE 3.1 – Échantillon d'images de la base d'image CODEBRIM [6].

tudes de caméras. Une partie des images est acquise par drone, car elles concernent des zones non accessibles à un humain.

Le choix d'opter pour des images de l'ensemble de données CODEBRIM est motivé principalement par le fait que ces images contiennent les cinq principaux défauts tant recherchés lors de l'inspection de ponts sur les surfaces en béton. En plus, les images sont en haute définition, ce qui permet d'avoir des informations de qualité, à savoir les détails de la texture des défauts qui constitue un atout important pour la tâche de segmentation qui opère au niveau pixel.

Nous avons sélectionné 593 images à partir de la base d'images de CODEBRIM pour construire notre ensemble de données. Cette sélection est conduite en se basant sur la diversité et la variété des défauts présents dans les images. L'aspect de la profondeur des prises de vues des images a aussi joué un rôle dans cette sélection. Nous avons distingué deux catégories d'images selon la profondeur de leur champ de vision comme le montre la figure 3.2. D'une part, les images avec une profondeur de champ lointaine qui incluent des vues d'ensemble des ponts, par conséquence les défauts sont de taille très réduite. D'autre part, nous avons des images avec des profondeurs de champ proche des zones d'intérêts avec une vue très focalisée sur les défauts, ce qui permet d'avoir des défauts avec des grandes tailles proportionnellement à l'image. Cette différence de champ de profondeur des images constituera un défi pour les modèles et réseaux de segmentation qui seront amenés à traiter des défauts dans des conditions différentes.

FIGURE 3.2 – (a) : Exemple d'images avec un champ de profondeur proche.
(b) : Images avec un champ de profondeur lointain

3.2.2 Processus d'annotation

La tâche d'annotation est une étape cruciale dans le processus de construction de notre ensemble de données pour la segmentation. Cette étape consiste à la construction des masques de segmentations pour les défauts convoités lors de l'inspection de ponts en béton. Ces masques représenteront la vérité terrain de notre ensemble de données. Comme les images déjà sélectionnées incluent cinq catégories de défauts (taches de corrosion, fissures, efflorescence, dégradation de béton et barres exposées), nous allons nous focaliser pour commencer sur 4 catégories pour notre ensemble de données, à savoir :

Corrosion : survient suite au processus d'oxydation des structures métalliques présentes dans le béton des ponts. Dans le contexte d'inspection, la présence de ces taches signifie que les structures métalliques se sont affaiblies ou bien, il y a une dégradation du béton, ce qui permettrait l'exposition des barres métalliques à l'humidité. Dans tous les cas, cela constitue un danger.

Efflorescence : c'est revêtement blanc qui résulte du phénomène de migration des minéraux (sels) à la surface du béton.

Dégradation de béton : se manifeste par un affaiblissement de la structure du béton avec des formes de désintégration.

Barre exposée : apparition des barres en métal constituant l'armature de la structure en béton. Ce défaut survient toujours avec une dégradation

du béton.

(a) (b)

FIGURE 3.3 – (a) : image contenant à la fois de la corrosion et des barres exposées ainsi qu'une dégradation du béton. (b) : Image contenant une efflorescence.

CVAT : outils d'annotation

CVAT est un outil d'annotation libre, à code source ouvert qui existe en deux versions (web et locale). Il intègre une panoplie d'outils pour l'annotation de données liées au domaine de la vision par ordinateur. Dans notre cas, il est l'outil idéal pour créer des masques de segmentation de façon interactive et les modifier au niveau pixel. CVAT intègre aussi un outil pour la génération semi-assistée des masques qui se base sur le modèle *Segment Anything* [43]

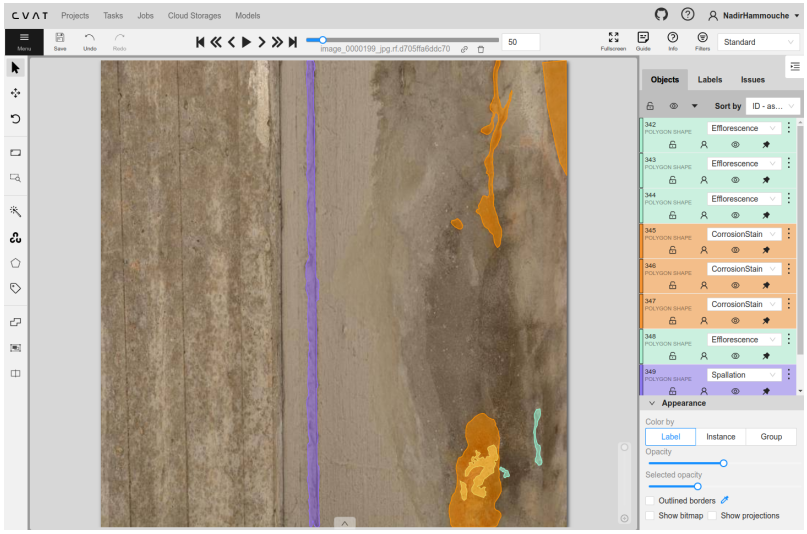


FIGURE 3.4 – Capture d'écran de l'environement d'annotation CVAT.

Dans un premier temps, nous avons chargé les images à annoter dans la plateforme web qui inclue un espace de stockage. Puis, nous avons défini les classes des défauts à annoter comme le montre la figure 3.5. Ensuite, nous avons procédé

à l'annotation de chaque image en binôme[1].En résumé, cette étape s'est déroulée sous deux passes. La première consistait à créer des masques pour chaque image de notre banque de données à partir de rien. Et la deuxième venait juste après pour la vérification et le raffinement des masques.

FIGURE 3.5 – Étape de création des classes de défauts dans CVAT.

Format des annotations

Une fois le processus d'annotation des images terminé, nous avons procédé à l'étape de génération des annotations pour que notre ensemble de donnée soit utilisable. Pour cela, nous avons opté pour le format d'exportation MS COCO pour exporter nos masques. Notre choix est motivé par le fait que ce format permet une prise en charge d'annotations pour différentes tâches de la vision pour ordinateur (détection d'objets, segmentation sémantique et d'instance). De plus, le stockage et l'encodage des masques pour notre cas se font sous forme de polygones. Ce qui offre la possibilité d'avoir des masques qui se chevauchent sans perte d'information pour une segmentation multiclasse. Enfin, les annotations sont stockées dans un fichier JSON simple et facilement interprétable comme l'illustre la figure 3.6.

```
{
  "info"          : info,
  "images"        : [image],
  "annotations"   : [annotation],
  "licenses"      : [license],
}

info{
  "year"          : int,
  "version"       : str,
  "description"   : str,
  "contributor"   : str,
  "url"           : str,
  "date_created"  : datetime,
}

image{
  "id"            : int,
  "width"         : int,
  "height"        : int,
  "file_name"     : str,
  "license"       : int,
  "flickr_url"    : str,
  "coco_url"      : str,
  "date_captured" : datetime,
}

license{
  "id"            : int,
  "name"          : str,
  "url"           : str,
}
```

FIGURE 3.6 – Le format JSON d'annotation COCO.

1. Annotation par deux personnes pour réduire la subjectivité et augmenter la précision des masques.

3.3 Répartition des données

Après le processus d'annotation des images de notre ensemble de données, il faut procéder à la répartition des images et de leurs annotations en trois sous-ensembles avec des proportions bien définies. Pour cette étape, nous avons choisi de répartir 75% des images dans l'ensemble d'entraînement et 12,5% pour l'ensemble de test et de validation.

3.3.1 Description de l'approche

Pour que nos sous-ensembles de données soient cohérents et équilibrés, nous avons mis en place une approche de répartition qui se base sur la similarité qui existe entre les images 3.7. Pour commencer, nous avons chargé les images de notre ensemble pour les redimensionner ensuite sous la résolution (300×300). Puis, nous avons généré des prolongements d'image pour chacune. Cette transformation consiste à aplatir une structure en deux dimensions pour obtenir une structure à une dimension. Ensuite, nous avons procédé à la réduction de dimension de ces prolongements pour avoir des coordonnées à deux composantes que nous pourrions projeter sur un plan à deux dimensions. Une fois cette étape terminée, nous avons appliqué une méthode de répartition de données pour construire des groupes homogène qui regroupe les points qui sont proches entre eux en termes de distance. Cela correspond à la similarité entre les images de départ. Enfin, nous avons parcouru chaque groupe construit précédemment pour affecter son contenu en suivant les proportions initiales (entraînement : 75%, test : 12,5% et validation : 12,5%). À la fin, nous avons obtenu un ensemble de données exploitable qui comprend les sous-ensembles d'entraînement, validation et de test qui sont à la fois cohérents et équilibrés.

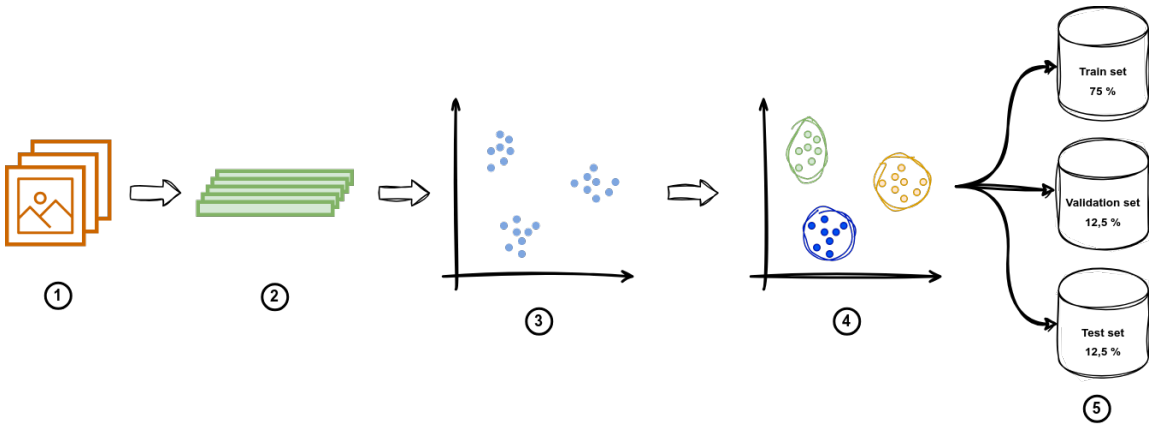


FIGURE 3.7 – Étapes du processus de répartition des images. (1) : Chargement des images avec une résolution fixe (300×300) ; (2) : Génération des prolongements d'images ; (3) : Réduction de dimension ; (4) : Construction des groupes (clustering) ; (5) : Répartition finale de chaque groupe selon les proportions fixées.

3.3.2 Outil d'exploration de données : Fiftyone

Pour accomplir ce processus de répartition des images, nous avons opté pour une bibliothèque python à code source ouvert qui s'intitule fiftyone [7]. Cette bibliothèque est développée par Voxel51 et offre une panoplie de fonctionnalités pour l'exploration des données. Elle permet de simplifier le processus de conception de modèles et leur évaluation. En plus, des équipes peuvent facilement collaborer sur des projets via cette plateforme. La plateforme est composée de 3 principales composantes : la librairie python, l'application de visualisation et le cerveau. La librairie python offre une collection de structure de données qui sert à l'exploration et la manipulation des ensembles de données. L'application est une interface graphique qui facilite la visualisation des données de manière intuitive et interactive comme le montre la figure 3.8. Le cerveau est une librairie python qui inclue une collection d'outils d'apprentissage machine qui servent à l'extraction des statistiques et des caractéristiques pertinentes des données.

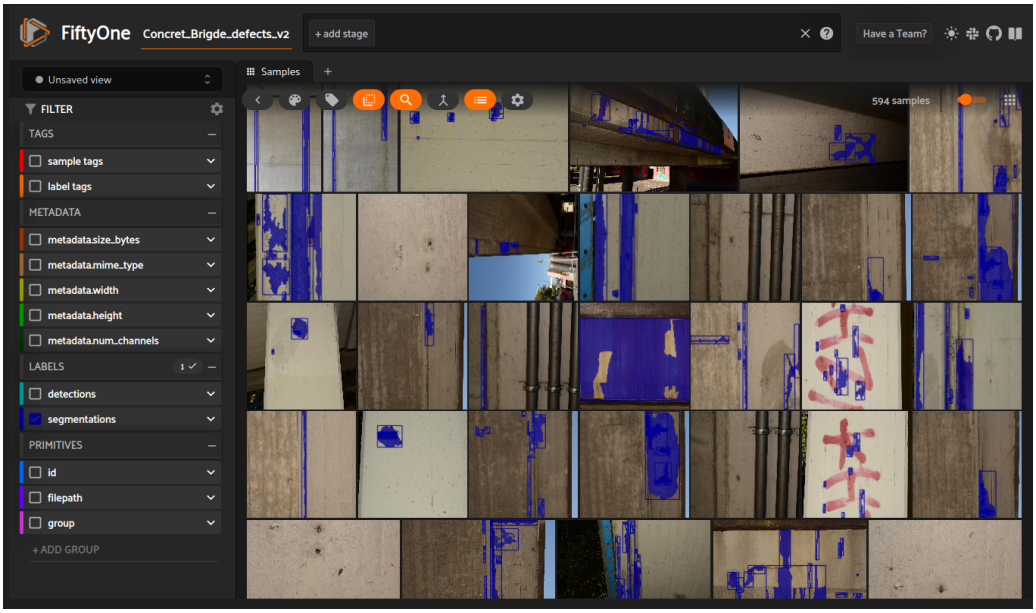


FIGURE 3.8 – Aperçu de l'ensemble de données avec annotations via l'interface de fiftyone [7].

Fonctionnalités de fiftyone :

- Une interface interactive pour la visualisation des données et leurs annotations ;
- Débogage des modèles et des réseaux ;
- Intégration des librairies les plus connues de l'apprentissage profond et machine comme TensorFlow et PyTorch ;
- Outils de visualisation des statistiques et des données comme les histogrammes, matrices de confusion,... etc.

3.3.3 Méthode de réduction de dimension : UMAP

Dans la phase de réduction de dimension citée dans les paragraphes précédents, nous avons employé la méthode UMAP [2] [44]. Cette méthode de réduction de dimension est une technique très sophistiquée qui permet générer des coordonnées en deux dimensions dans notre cas à partir des prolongements d'image. Cette technique a la capacité de préserver la structure locale et globale de l'ensemble des prolongements. Ce qui la rend efficace pour des données très larges. Dans notre approche pour la répartition de notre base d'images, cette étape se positionne comme une étape de prétraitement à la phase de regroupement des images en groupes. UMAP est reconnue comme une méthode qui améliore les résultats de regroupement des données.

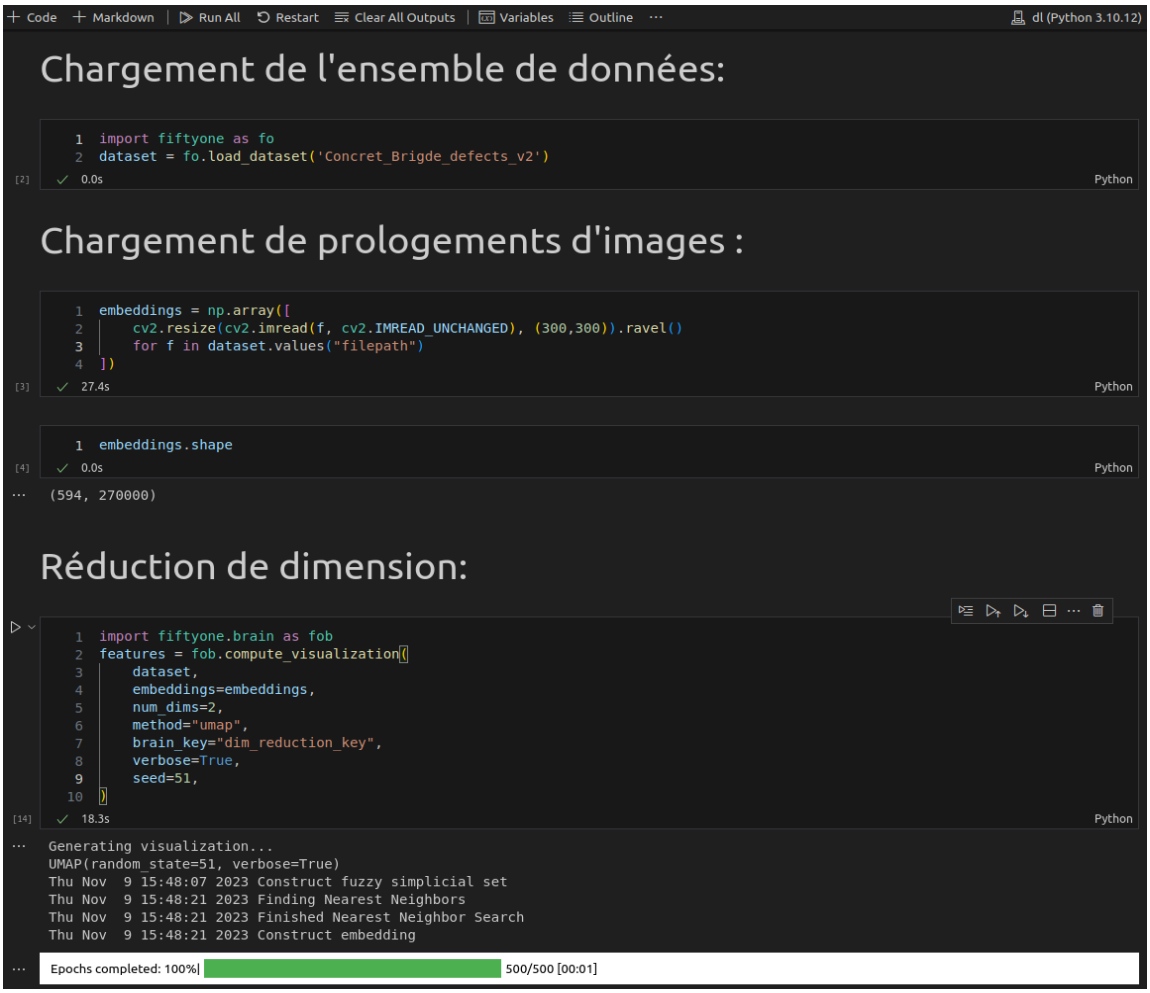
Principe de fonctionnement : La méthode UMAP procède dans un premier temps par la construction de l'ensemble simplicial flou [3] correspondant aux données de départ dans un espace à haute dimension. Ensuite survient la phase d'optimisation. Elle consiste à la minimisation de la fonction d'entropie croisée entre les ensembles simpliciaux flous correspondants aux représentations à haute dimension et celle à dimension réduite respectivement. Ce processus s'achève par la production d'une représentation des données à dimension réduite.

Avantages : La technique UMAP se distingue avec une meilleure évolutivité sur de large quantité de données avec des temps de traitements très optimisés comparée à la méthode t-SNE [45]. En plus, UMAP offre une flexibilité aux utilisateurs dans l'ajustement des paramètres de la méthode pour équilibrer la préservation entre la structure locale et globale des données.

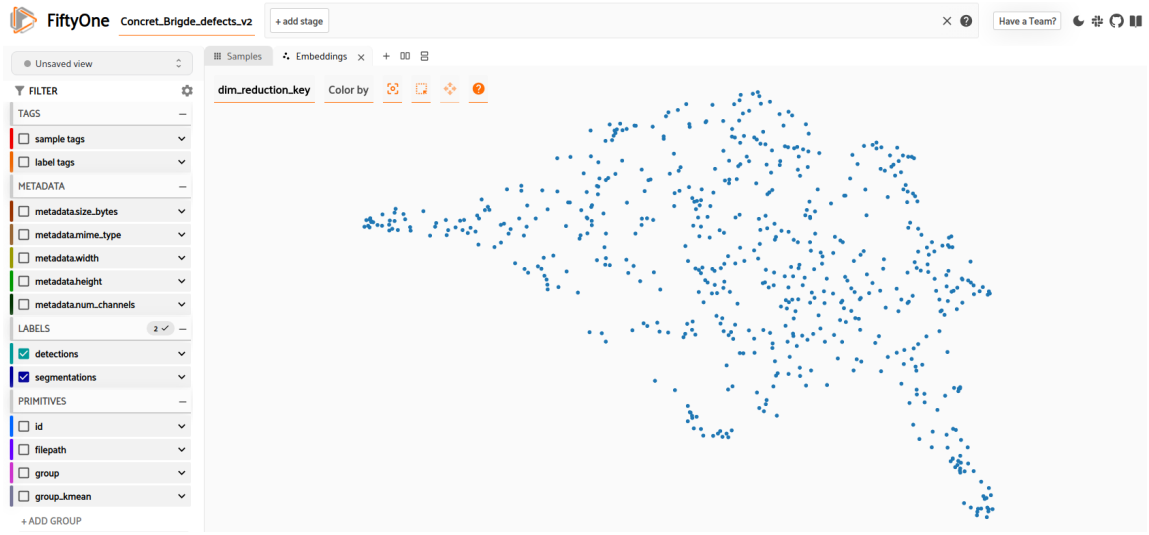
Code et implémentation : Afin d'appliquer cette technique de réduction de dimension sur les prolongements d'images préalablement chargés, nous avons fait appel à la fonction compute_visualization() qui est incluse dans la librairie fiftyone.brain. Le code qui décrit les différentes étapes de cette phase est illustré à la figure 3.9a. Après l'exécution du code de la figure 3.9a, nous avons obtenu la visualisation des coordonnées sur un plan en deux dimensions de nos images comme le montre la figure 3.9b.

2. Acronyme en anglais : Uniform Manifold Approximation and Projection

3. Une construction mathématique qui permet de présenter les relations entre des points de données à la fois dans un espace à haute dimension et dans celui à basse dimension. Il se caractérise sous la forme d'un graphe pondéré avec des relations probabilistes qui lient les points de données dans l'ensemble



(a) Illustration des étapes de la phase de réduction de dimension par code python.



(b) Graphe des projections d'images sur un plan 2D après la réduction de dimension.

FIGURE 3.9 – Code et résultats de la phase de réduction de dimension

3.3.4 Répartition des données en groupes avec K-Moyennes

Une fois le processus de réduction de dimension soit achevé, l'étape de regroupement des images en sous groupes homogènes doit prendre place. Pour cette étape en particulier, nous avons choisi d'opter pour l'algorithme des K-moyennes. L'objectif de cette étape est de mettre les images similaires et qui partagent un certain nombre d'aspects dans les mêmes groupes afin de faire une répartition équilibrée de ces dernières sur nos ensembles d'entrainement, test et de validation.

Description de l'algorithme k-moyen : L'algorithme k-moyen nécessite avant tout de fixer le nombre de sous-groupes à construire à partir du jeu de données. Ensuite, un processus itératif prendra place jusqu'à atteindre un certain point de convergence. Les étapes suivantes résument le déroulement de l'algorithme :

1. Sélectionner K points de l'ensemble de données et les assigner comme les centroïdes des groupes de départs ;
2. Affecter chaque point de l'ensemble de données à un sous-groupe qui a le centroïde le plus proche de lui. La distance euclidienne est plus souvent utilisée pour calculer les distances entre les points.
3. Mise à jour des centroïdes de chaque groupe en se basant sur la moyenne des points qui appartient à ce dernier.
4. Répéter l'étape 2 et 3 jusqu'à ce que les centroïdes des groupes ne changent pas significativement.

La somme des moyennes des carrés entre les données et les centroïdes est minimisée tout le long du processus itératif jusqu'à convergence. Le choix du nombre de sous groupe à construire est très crucial pour l'algorithme. Cependant, il existe des méthodes qui permettent de choisir ce paramètre de façon optimale.

Code et implémentation : Dans notre approche, l'algorithme k-moyen est utilisé sur les coordonnées en deux dimensions réduites précédemment qui correspondent aux images de notre ensemble de données. La libraire python scikit-learn inclut une panoplie d'algorithmes de regroupement de données. Nous avons utilisé la version du k-moyen qui est proposé dans cette librairie.

Après plusieurs expériences, nous avons constaté visuellement via les graphes de projection que le nombre de sous groupe qui correspond au mieux pour notre ensemble d'images est 38.

La figure 3.10 illustre bien le code correspondant à cette étape. La figure 3.11, nous montre les différents groupes construits à la fin de l'exécution de l'algorithme.

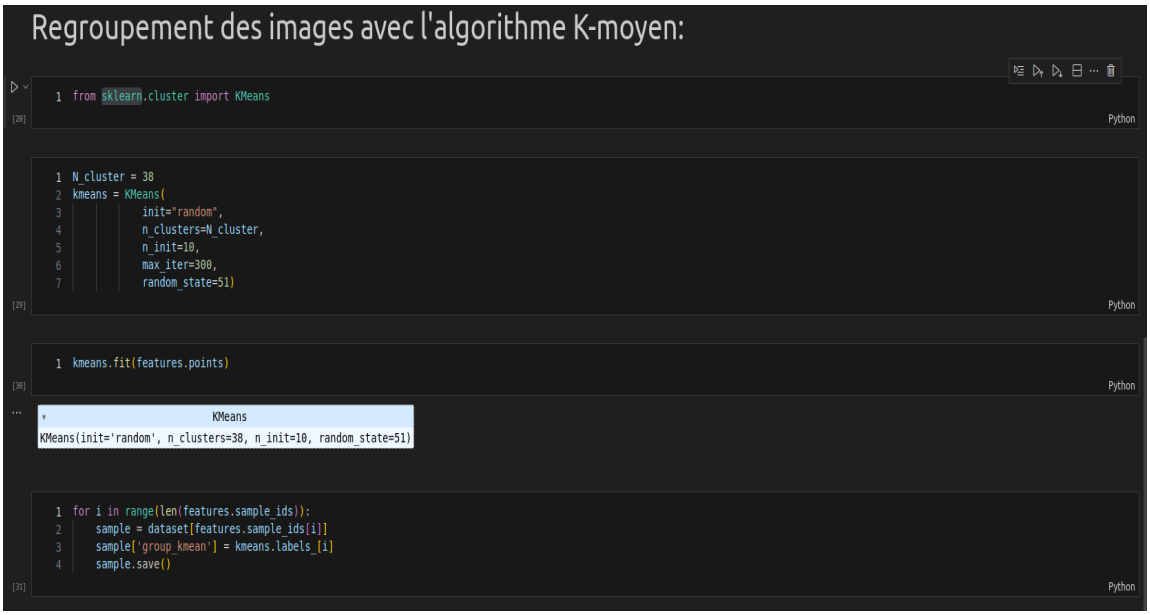


FIGURE 3.10 – Illustration des étapes de regroupement via K-moyen.

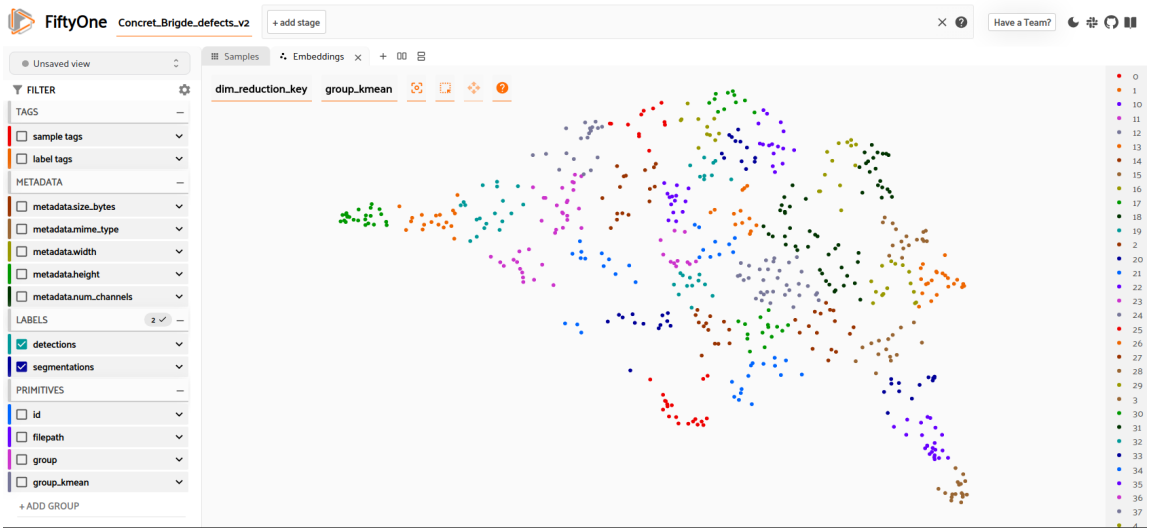


FIGURE 3.11 – Graphe des projections d'images sur un plan 2D après regroupement k-moyen.

3.4 Les statistiques de l'ensemble de données

Une fois notre ensemble d'images annoté, nous présenterons dans cette section quelques statistiques basiques qui caractérisent la nature de notre ensemble de données. Nous entamerons par l'analyse des distributions des instances de défaut ainsi que des images par catégorie de défaut pour avoir une idée sur l'équilibre inter-classe des défauts dans notre ensemble de données. Ensuite, nous enchaînerons avec des statistiques sur les relations binaires entre classes avec la matrice de corrélation des défauts. Nous terminerons par la mise en évidence des cartes de chaleur pour chaque classe de défaut afin d'avoir une idée sur la concentration et la localisation des défauts dans notre ensemble de données.

3.4.1 Distribution des instances d'objet par classe de défaut

Dans les graphes de la figure 3.12, nous notons clairement que la classe corrosion est la classe qui comprend le plus d'instances d'objets. Ensuite, nous avons les classes d'efflorescence et des barres exposées qui détiennent à approximativement les mêmes proportions à savoir 25,7% et 21,9% respectivement. Enfin, nous avons la classe dégradations de béton avec la plus petite proportion de 18,3% . Ces chiffres traduisent la réalité terrain des instances de défauts par classes de notre ensemble d'images.

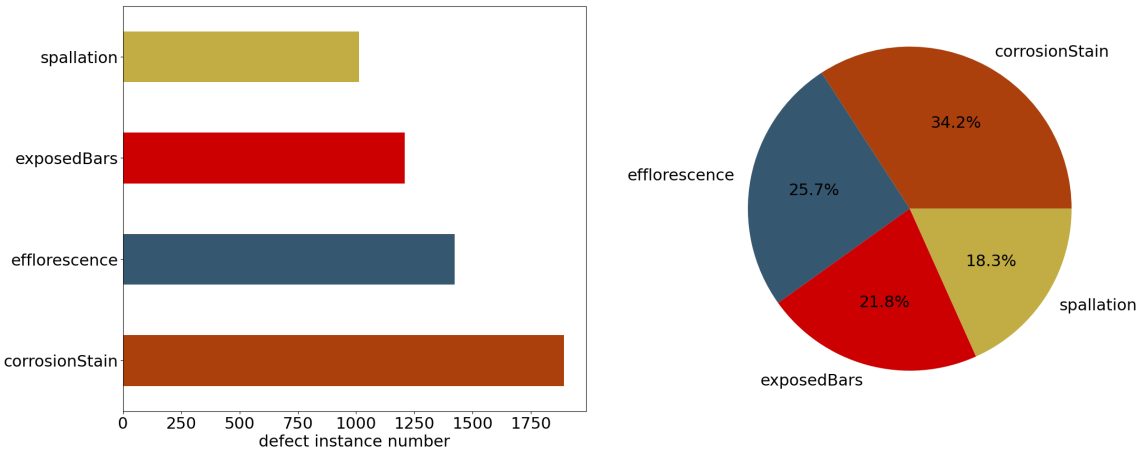


FIGURE 3.12 – Distribution des instances de défauts par classe

3.4.2 Distribution des images par classes de défaut

L'histogramme de la figure 3.13 traduit la distribution des images par catégorie de défaut. Nous notons clairement que les classes corrosion et dégradation de béton sont les plus représentées dans l'ensemble de nos images. Ensuite, nous avons les classes efflorescences et barre exposée qui vient juste derrière.

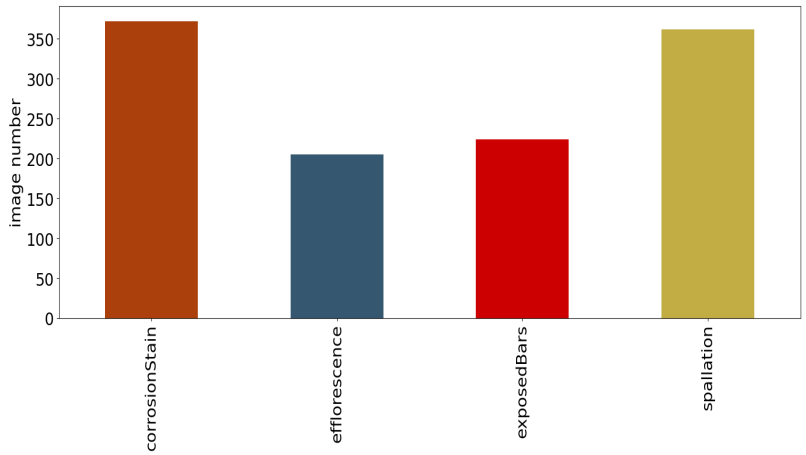


FIGURE 3.13 – Histograme de répartition des images par catégorie de défaut.

À travers les deux analyses statistiques précédentes. Nous constatons que les classes de défauts sont réparties en suivant un certain équilibre qui est déjà présent dans la réalité terrain de nos images d'inspection.

3.4.3 Matrice de corrélation

Le tableau 3.1 représente la matrices de corrélation qui donne la nature des relations entre les classes de défauts dans notre ensemble de données. La matrice de corrélation est calculée en tenant compte des nombres d'instances de défaut par images.

TABLE 3.1 – Matrice de corrélation des défauts

	Corrosion	Efflorescence	Barre exposée	Dégradation béton
Corrosion	1.0	0.08	0.69	0.29
Efflorescence	0.08	1.0	0.004	-0.056
Barre exposée	0.69	0.004	1.0	0.37
Dégradation béton	0.298	-0.056	0.37	1.0

D'après les résultats du tableau 3.1, nous constatons une forte corrélation en la classe de corrosion et celle des barres exposées. Cela est dû au fait que dans la majorité des images d'inspection de notre base, la présence de la corrosion est le résultat direct du phénomène d'oxydation des barres exposées. D'autre part, nous avons aussi une corrélation considérable entre les cas des dégradations de béton et l'apparition des barres exposées. Comme les barres métalliques sont cachées à la base dans le béton, alors leur présence est la conséquence logique de la dégradation du béton.

3.4.4 Carte de température par classe de défauts

Dans la figure 3.14, nous pouvons avoir une idée sur la concentration des défauts en termes d'emplacement sur les images. Cette information est très importante lors de la conception des modèles et des réseaux de segmentation. En effet, elle pourra considérablement contribuer au choix des extracteurs de caractéristiques visuelles des images.

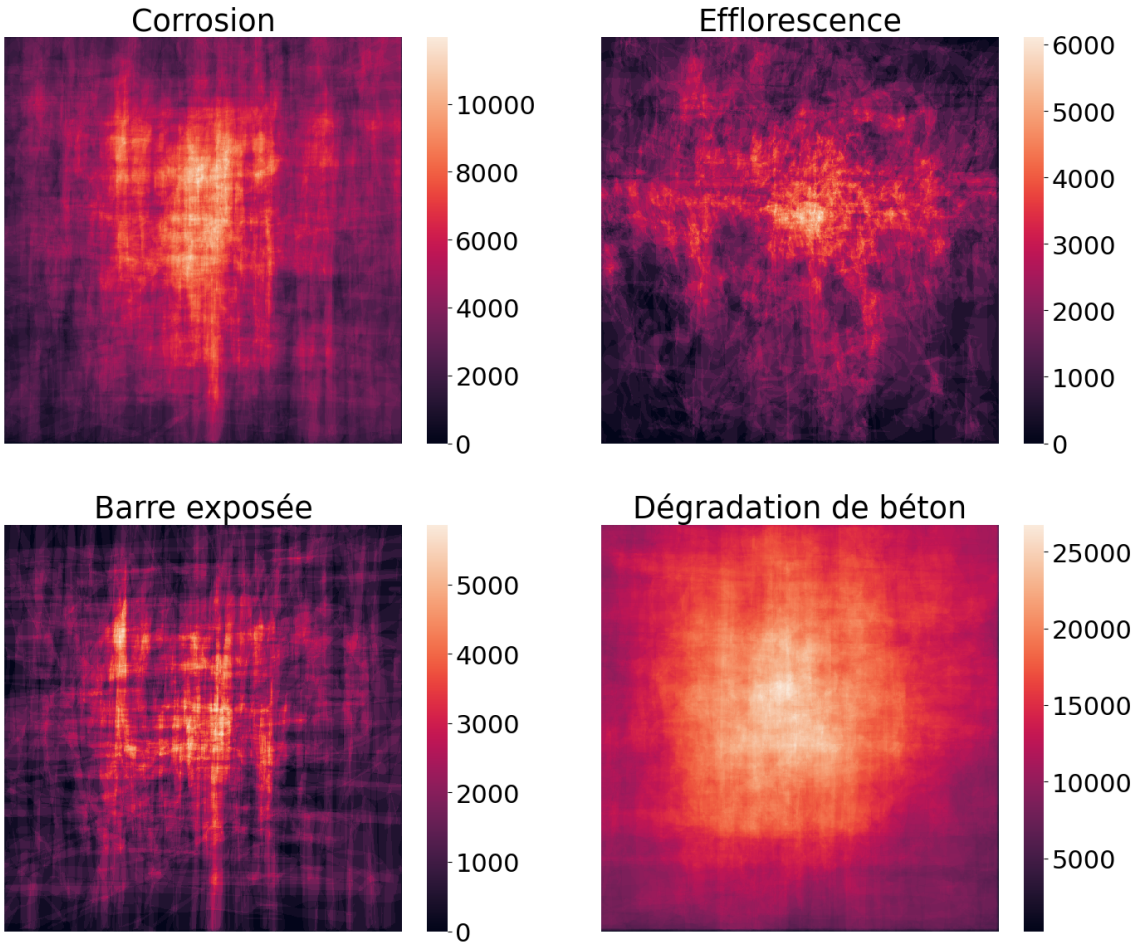


FIGURE 3.14 – Cartes des températures par classes de défaut.

3.5 Conclusion

Au terme de ce chapitre, l'étape qui consiste à la mise en place d'un ensemble de donnée pour la tâche de la segmentation des défauts présents au niveau des ponts en béton est un objectif atteint. Ce processus est achevé au bout d'un certain nombre d'étapes qui sont décrites en détails dans cette section. Maintenant que nous disposons d'une banque d'images et de leurs masques respectifs pour la segmentation des défauts présents sur les ponts en béton, nous aborderons la partie centrale de notre travail de recherche. Elle concerne les solutions pour la problématique posée sur les segmentations des défauts que nous aborderons avec plus de détails dans le chapitre suivant. Ainsi, cet ensemble de données annoté servira pour l'entraînement des modèles et des réseaux qui seront abordés ultérieurement.

Chapitre 4

Méthodologie

4.1 Introduction

Dans ce présent chapitre, nous aborderons les différents aspects de la solution que nous avons proposée afin de répondre à la problématique de la segmentation des défauts dans le processus d'inspection de ponts en béton. En premier temps, nous ferons la lumière sur les concepts clés et fondamentaux avec lesquels nous avons élaboré notre solution. Ensuite, nous exposerons en détails les déférents aspects de la solution. Cette dernière est composée de trois parties. La première approche est classique et naïve, elle se présente sous la forme d'un réseau de segmentation multi-classe des défauts. La deuxième approche aborde le problème sous l'angle de l'apprentissage multitâche. En effet, la segmentation de chaque défaut est considérée comme une tâche à part entière. Dans cette approche, nous proposerons deux architectures orientées apprentissage multitâche : Architecture à encodeur partagé et celle à encodeur-décodeur partagés. Ces deux architectures sont dotées aussi de mécanismes qui permettent d'améliorer les résultats comme l'attention et la supervision en profondeur.

4.2 Concepts de base

4.2.1 Apprentissage multitâche

L'apprentissage multitâche est un paradigme d'apprentissage dans le domaine de l'apprentissage automatique. D'après [8], ce paradigme suggère que le modèle soit entraîné simultanément sur des données issues de multiples tâches différentes. Ainsi le modèle apprendra des représentations partagées et communes aux multiples tâches. Ces représentations partagées améliorent l'efficacité des données, ce qui permettra ainsi d'atteindre une vitesse d'apprentissage plus rapide pour les tâches. Tous ces effets pallieront les limites traditionnelles que l'en rencontre dans l'apprentissage profond, à savoir la limite de la quantité de données disponible et la nécessité d'avoir des ressources de calculs importante.

L'apprentissage multitâche reflète le processus d'apprentissage du cerveau humain comparé à l'apprentissage monotâche. La force de ce procédé réside dans le fait qu'un modèle peut exploiter positivement les connaissances qu'il a apprises pour une tâche donnée dans l'apprentissage d'autres tâches différentes. C'est ce que l'en nomme par le transfert positif. Néanmoins, ce paradigme d'apprentissage présente aussi des difficultés qui ne sont présentes dans l'apprentissage monotâche. Ces difficultés se manifestent principalement lorsqu'un modèle améliore les performances d'une tâche particulière pendant que pour les autres tâches les performances se détériorent. Dans ce cas, les tâches ont des besoins conflictuels. Ce phénomène s'appelle le transfert négatif.

L'enjeu majeur dans la conception des architectures à apprentissage multitâches est la minimisation du transfert négatif entre les tâches. Il existe des approches dans cette optique qui consistent à concevoir des parties plus spécifiques du modèle en vue des tâches, en plus du mécanisme d'attention qui s'avère utile dans ce cas. Cependant, il faut toujours garder à l'esprit que le but d'une architecture à apprentissage multitâche est d'exploiter au maximum les aspects communs entre les tâches. Pour ça, le réseau doit permettre au flux de donnée d'atteindre les parties communes entre les tâches de sorte à maximiser le transfert positif entre les tâches.

Motivation

Lors de la construction de l'ensemble d'image pour l'inspection des défauts, nous avons recensé quatre classes de défauts (corrosion, barres exposées, dégradation de béton et efflorescence). Après avoir exploré ces données, nous avons constaté que les instances de ces classes de défauts partageaient des aspects communs entre eux. En d'autres termes, nous avons observé qu'il y avait une relation de cause à effet entre ces classes de défauts. Par exemple, la dégradation de béton a pour conséquence directe l'apparition des barres exposées qui est le plus souvent accompagnée par la corrosion. Ceci est expliqué par le fait que la corrosion se forme lorsqu'il y a une partie métallique qui est exposée à l'humidité. Toutes ces relations existantes entre

les classes de défauts représentent un potentiel pour une architecture à apprentissage multitâche qui sera capable de bien exploiter ces aspects communs afin d'achever des meilleurs résultats dans la segmentation des différents défauts.

D'autre part, l'origine de cette approche à apprentissage multitâche prend racine du fonctionnement du cerveau humain qui a la capacité d'apprendre plusieurs tâches tout en exploitant les connaissances communes à ces dernières. De même, pour la problématique traitée dans ce travail de recherche, mettre en place une seule architecture qui sera capable de segmenter les quatre classes de défaut est un défi majeur. À cet effet, l'apprentissage multitâche est la clé qui permet d'avoir une seule architecture optimisée pour les quatre défauts de sorte à avoir des résultats qui sont au moins similaires à une architecture spécifique à chaque défaut. Cet aspect est crucial dans le processus du déploiement du modèle.

Types d'architectures :

Dans le processus de conception de modèles à architecture partagée entre différentes tâches, il est important de prendre en considération les facteurs qui définissent la manière de partitionner les paramètres du réseau sur les parties partagées et celles spécifiques aux tâches. En d'autres termes, savoir répartir les filtres de convolution du modèle entre les ensembles de filtres partagés et spécifiques est la clé pour concevoir une architecture efficace qui saura exploiter au mieux les vertus de l'apprentissage multitâches. Dans la littérature relative à ce champ de recherche, les architectures les plus performantes se caractérisèrent avec un certain équilibre dans la répartition des parties partagées et spécifiques de ces dernières relativement au domaine d'information des tâches concernées.

Traditionnellement, les architectures orientées apprentissage multitâches sont réparties en deux grandes familles :

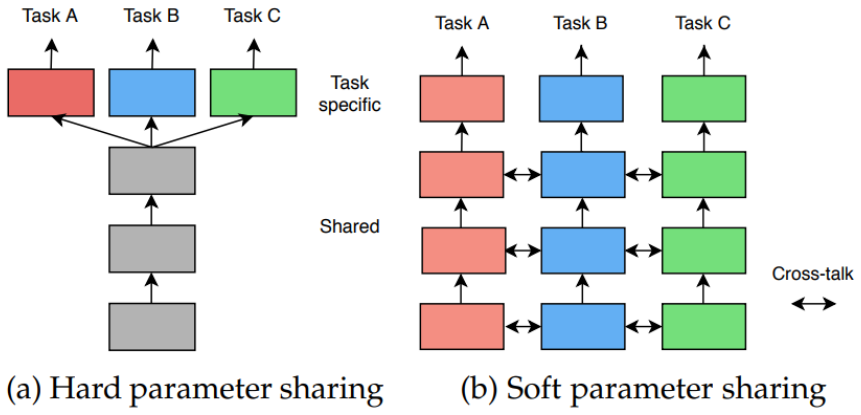


FIGURE 4.1 – Les deux familles d'architectures à apprentissage multitâche.[8]

Architectures à hard parameter sharing : qui se distinguent par le fait que les paramètres du réseau sont répartis en deux sous-ensembles. Les paramètres partagés sont communs pour toutes les tâches et sont responsables directement aux représentations communes entre les tâches. D'autre part, les paramètres dédiés et spécifiques à chaque tâche. L'architecture à encodeur partagé est la plus répandue dans cette classe.

Architecture à encodeur partagé Cette architecture se distingue par un encodeur commun à toutes les tâches. L'apprentissage des aspects commun entre les différentes tâches se fait au niveau de cet encodeur. Elle se caractérise par le fait d'avoir deux catégories de paramètres : les paramètres partagés et les paramètres propres à chaque tâche. La figure 4.2 proposé par [9] l'incarne bien. Le réseau TCDCN est constitué d'un encodeur commun à toutes les tâches qui intègre une série de couches de convolution. Les représentations résultantes de cet encodeur seront ensuite acheminées vers des branches spécifiques pour chaque tâche.

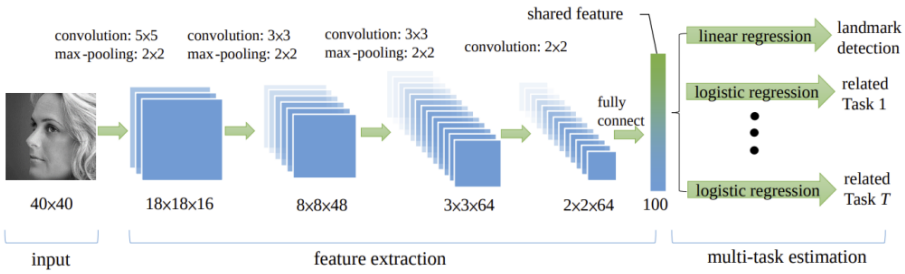


FIGURE 4.2 – Architecture du réseau TCDCN [9].

Architecture à soft parameter sharing : Dans cette catégorie, chaque tâche possède son propre ensemble de paramètres. Pour permettre une interaction entre les tâches, un mécanisme de partage de caractéristiques est mis en place entre les branches de chaque tâche comme un canal de communication.

Architecture à canal de communication Cette architecture se caractérise par des parties bien distinctes pour chaque tâche. Ces parties sont reliées entre elles via un canal intermédiaire dont le rôle est d'assurer une communication et un échange d'information entre les différentes tâches. Ainsi, cette communication permet pour chaque tâche de tirer profit des informations issues d'autre tâche afin d'améliorer au mieux les performances de chaque tâche. Dans ce type de configuration, le réseau global a une seule catégorie de paramètres qui sont propres pour chaque tâche. Dans la figure 4.3, nous avons un exemple de réseau qui adopte ce type d'architecture. Le réseau NDDR-CNN [10] dans la figure 4.3 est constitué d'un ensemble de sous-réseaux distincts propres à chaque tâche. Ces sous-réseaux communiquent entre eux au niveau de chaque couche de sorte que les représentations de chacune seront concaténées les une avec les autres. Ensuite, ils lui appliquent une convolution 1×1.

Le résultat est une représentation qui inclue des informations issues des deux tâches et qui seront acheminées vers les couches suivantes respectives de chaque sous-réseau.

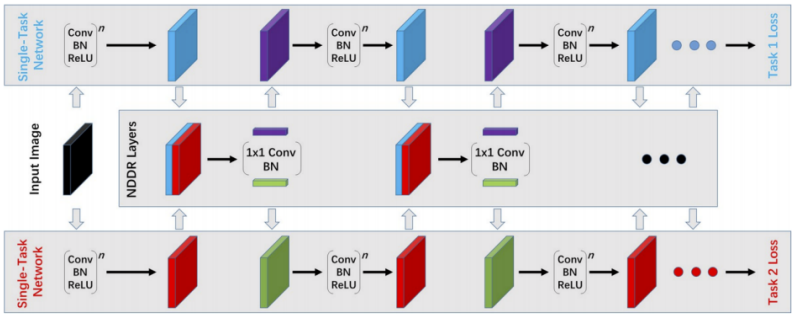


FIGURE 4.3 – Architecture du réseau NDDR-CNN [10].

Stratégies d'optimisation

Après avoir discuté les architectures des modèles à apprentissage multitâches, nous aborderons l'aspect relatif à l'entraînement de ce type d'architecture qui est le processus d'optimisation. En effet, les réseaux à apprentissage multitâche sont amenés à apprendre des différentes tâches simultanément. L'objectif est de veiller à garder un équilibre entre les tâches lors de la phase d'entraînement du réseau, de sorte qu'il n'y a pas une tâche qui domine plus sur la mise à jour des paramètres. D'après l'article de [8], il existe plusieurs stratégies d'optimisation pour ce type de techniques. Avant de parler de ces stratégies, il faut rappeler que pour ce type de réseau, nous avons une fonction de perte du réseau dans son ensemble. Elle est calculée en se basant sur des fonctions de perte pour chaque tâche du réseau comme dans l'équation 4.1.

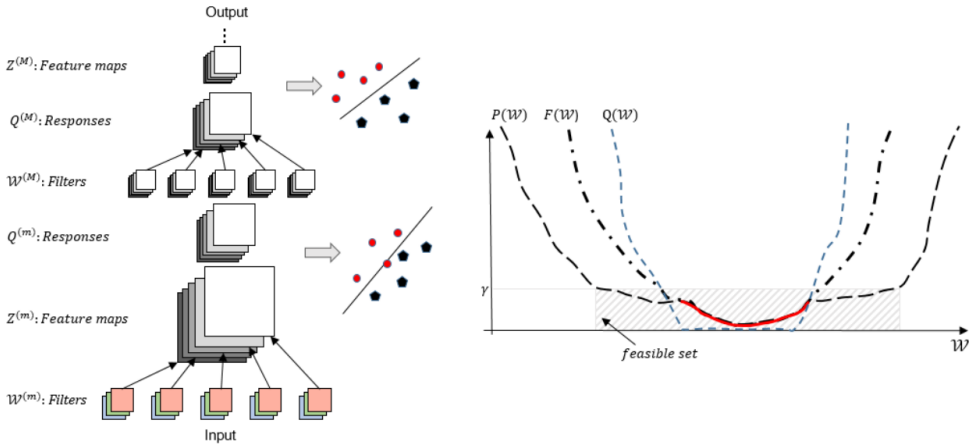
$$Loss_{globale} = \sum \omega_i \cdot Loss_i \tag{4.1}$$

En résumé, les stratégies d'optimisation visent dans un premier lieu à trouver le meilleur moyen de pondérer chaque fonction de perte propre à une tâche afin d'avoir une fonction globale équilibrée lors de l'entraînement. La première approche est proposée par [46], elle consiste à exploiter l'incertitude calculée par la fonction de vraisemblance entre les distributions relatives à la réalité terrain et la prédiction. La deuxième approche s'appuie sur la vitesse d'apprentissage de chaque tâche. Si la vitesse d'apprentissage d'une tâche diminue, alors le poids de sa fonction de perte doit augmenter pour réajuster l'équilibre. Il y a aussi une autre stratégie qui ressemble à la précédente et elle s'appuie sur une pondération en suivant les performances de chaque tâche pendant l'entraînement. D'autre part, nous avons aussi une autre stratégie qui est centrée sur le concept de récompense pour chaque tâche. Enfin, la dernière est une stratégie qui utilise la moyenne géométrique entre les fonctions de pertes spécifiques pour calculer la fonction de perte du réseau.

4.2.2 Supervision en Profondeur

La contribution proposée à travers ce travail de recherche se base principalement sur l'apprentissage multitâche. Les réseaux de segmentations basés sur ce paradigme d'apprentissage se distinguent par des architectures profondes et complexes. L'idée d'incorporer la technique de la supervision en profondeur dans ce travail s'avère très prometteuse en vue de répondre au problème classique de disparition du gradient lors du processus de la rétro-propagation. Ce problème est souvent rencontré au niveau des architectures profondes et complexes.

La supervision en profondeur est appliquée pour la première fois en 2015 par [11] dans la classification des images. Ensuite, elle a été largement appliquée dans le domaine de la vision par ordinateur et plus précisément dans le domaine de la segmentation des images biomédicales.



(a) DSN illustration (b) functions

FIGURE 4.4 – Ilustration d'incorporation de la supervision en profondeur dans le réseau DSN [11].

Cette technique permet au réseau d'avoir des supervisions auxiliaires au niveau de ces couches intermédiaires. Techniquement, chaque couche intermédiaire se verra associée une fonction de perte. Ainsi, le processus de rétro-propagation du gradient ne va pas s'enclencher seulement à partir de la couche de sortie, mais les couche intermédiaires aussi vont contribuer à ce dernier. Il existe un consensus selon lequel la supervision en profondeur peut améliorer les performances du réseau en permettant au réseau l'apprentissage de caractéristiques plus discriminatives. Cela a pour effet de rendre le modèle plus profond en empêchant ainsi le problème de la disparition du gradient de survenir. L'autre avantage aussi de cette technique est qu'elle permet au modèle de converger plus rapidement avec une quantité de données d'entraînement limitée [11].

4.2.3 Mécanisme d'attention

Le mécanisme d'attention est une technique qui prend son origine du fonctionnement du système visuel de l'homme. Ce dernier a pour capacité d'analyser et de comprendre efficacement des scènes complexes. Cela est dû au fait qu'il ressort les éléments saillants et importants de la scène en ignorant les parties moins importantes. Ce mécanisme est ensuite adopté dans le domaine de la vision par ordinateur avec succès.

Le mécanisme d'attention est considéré comme un système de sélection dynamique qui procède à la pondération flexible des caractéristiques en s'appuyant sur l'importance de la donnée d'entrée [47]. Dans la littérature relative à ce mécanisme, il existe plusieurs variantes avec des utilisations plus spécifiques à chacune. Dans le cadre de notre travail de recherche, nous nous sommes intéressés au module d'attention à bloc de convolution : CBAM [12]. Ce module est constitué de deux sous module d'attention : le module d'attention spatiale et le module d'attention à canal.

Module d'attention à canal

Ce module se présente comme un réseau de convolution qui a pour but de produire une caractéristique d'attention focalisée sur les canaux de la donnée d'entrée. Ce processus est rendu possible en exploitant les relations existantes entre les canaux de la donnée d'entrée. Le module se concentre à mettre en relief ce qu'est-ce qui est important sémantiquement dans la représentation d'entrée [12].

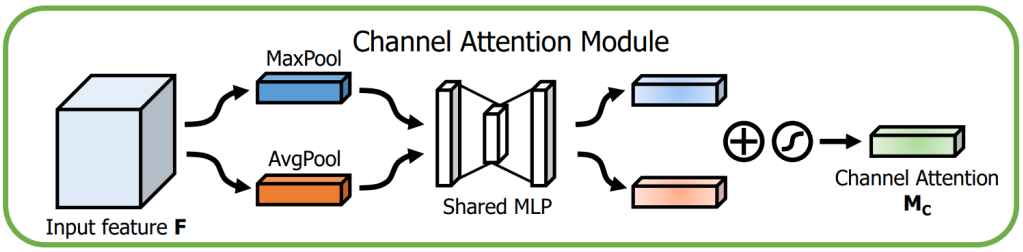


FIGURE 4.5 – Architecture du module d'attention à canal [12].

Le module procède en premier par le calcul du regroupement moyen et maximum de l'entrée pour former deux vecteurs (descripteurs). Ces derniers seront ensuite acheminés dans un réseau de neurones multicouche commun qui générera à son tour deux autres vecteurs respectifs. Les deux vecteurs résultats seront sommés pour servir d'entrée pour une fonction d'activation qui générera la caractéristique finale. Ce processus est décrit entièrement dans la figure 4.5.

Module d'attention spatiale

Le but de ce module est la génération d'une caractéristique qui contient les informations spatiales importantes de la représentation de départ. Contrairement

au module précédent, il procède en exploitant les informations inter-spatiales de la représentation d'entrée. Ce module vise à répondre à la question d'où se situent les zones les plus importantes de la caractéristique.

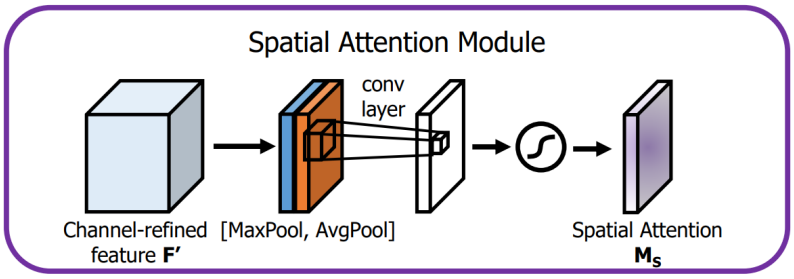


FIGURE 4.6 – Architecture du module d'attention spatiale [12].

Ce module commence par appliquer à la représentation d'entrée un regroupement moyen et maximum le long de l'axe des canaux. Ces deux opérations ont pour but de mettre en évidence les parties importantes de la donnée d'entrée. Les représentations résultantes seront ensuite concaténées et acheminées vers une couche de convolution. Enfin, la caractéristique spatiale est générée. L'architecture globale de ce module est bien décrite dans la figure 4.6.

Module d'attention à bloc de convolution

Dans l'optique de mettre en évidence à la fois les canaux et les régions qui sont plus informatives d'une représentation, [12] ont proposé un module d'attention à bloc de convolution. Ce module d'attention est constitué de deux sous-modules : le module d'attention à canaux et le module d'attention spatiale. Ces derniers sont disposés en série afin de tirer profit d'une meilleure efficacité d'inférence et ils ont la propriété d'êtres complémentaires. En effet, le Module CBAM permet à la fois de se concentrer sur les canaux et zones les plus informatives de la représentation. En plus d'être léger, il peut s'intégrer dans des réseaux de neurones plus complexes sans contrainte. L'architecture globale de ce module est décrite dans la figure 4.7.

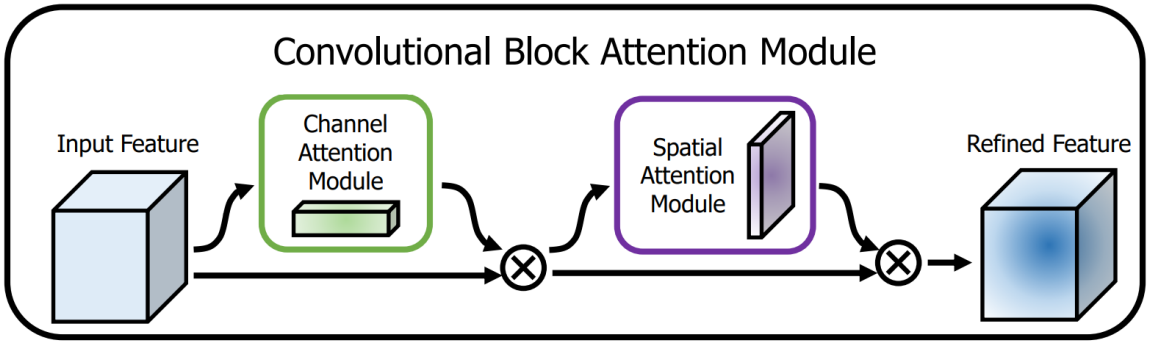


FIGURE 4.7 – Architecture du module d'attention à bloc de convolution [12].

4.3 Solutions proposées

4.3.1 Réseau U-Net classique

Architecture

Cette première solution présentée dans cette partie est une approche classique pour une segmentation de type multi-classe. Elle se présente sous forme d'un réseau de segmentation U-Net classique [2] avec une disposition de type encodeur-décodeur sous forme d'un 'U'. Nous avons ajusté sa couche de sortie afin qu'il génère une collection de masques qui correspondent aux défauts traités. En d'autres termes, ce réseau prendra en entrée une image d'inspection et produira des masques pour chaque classe de défauts. Il faut mentionner aussi que la fonction d'activation qui permet d'avoir les masques est de type sigmoïde.

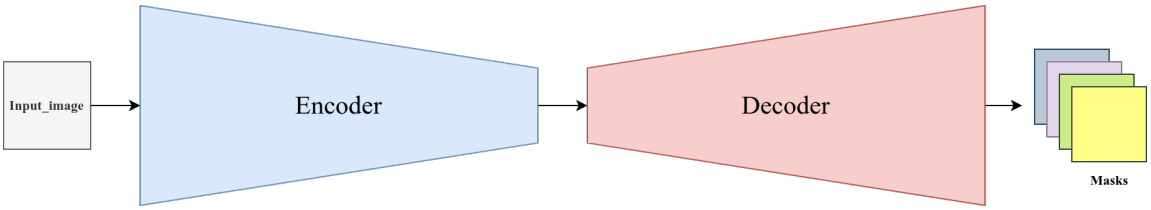


FIGURE 4.8 – Apérçu de l'architecture globale.

D'un point de vue architectural et conceptuel, ce modèle est aussi considéré comme un cas particulier de réseau qui adopte le paradigme d'apprentissage multitâche. En effet, la globalité du réseau est partagée entre les différents défauts qui peuvent être vus comme des tâches. Ainsi, ce modèle a pour particularité de ne pas inclure de partie de son architecture qui est totalement dédiée à une tâche particulière.

Implémentation :

Comme mentionné dans la section précédente, cette solution propose un réseau classique de type U-Net. Pour implémenter une telle architecture, nous avons opté pour la libraire Segmentation-models [48] développée entièrement en PyTorch et qui propose une panoplie de modèles destinés pour la tâche de segmentation. Cette libraire a pour particularité d'offrir des modèles avec des paramètres pré-entraînés qui favorisent une meilleure et rapide convergence pour la phase d'entraînement. L'encodeur de notre réseau reprend les bases du ResNet50 [13] pré-entraîné sur ImageNet avec cinq niveaux de profondeur. Il comporte 23 millions de paramètres entraînables que nous avons gelé pour le processus d'entraînement. La figure 4.9 illustre le bout de code utilisé afin de construire le réseau dans sa globalité. Le nombre de masques à produire par le modèle est fixé par le paramètre 'classes'. La fonction de perte est, elle aussi, paramétrée via l'argument 'activation'.

```python
model = smp.Unet(
    encoder_name="resnet50",        # choose encoder, e.g. mobilenet_v2 or efficientnet-b7
    encoder_weights="imagenet",     # use `imagenet` pre-trained weights for encoder initialization
    in_channels=3,                  # model input channels (1 for gray-scale images, 3 for RGB, etc.)
    classes=4,                      # model output channels (number of classes in your dataset)
    activation='sigmoid'
)
```

FIGURE 4.9 – Code d'implémentation du réseau U-Net.

4.3.2 Réseau à apprentissage multitâche avec encodeur partagé

Architecture

Cette architecture suit le modèle encodeur-décodeur et s'inspire plus précisément du réseau U-Net [2]. Elle se distingue par ces deux parties, une partie commune aux tâches et une deuxième spécifique à chaque tâche. La première partie de cette architecture se caractérise avec un encodeur commun pour l'ensemble des tâches. En effet, mettre en place un encodeur commun pour toutes les tâches permet une extraction des caractéristiques génériques des défauts. Ce choix est motivé par les relations existantes entre les défauts que nous avons constatés dans la réalité terrain des données. Ces relations ont le potentiel de permettre un transfert positif d'apprentissage entre les tâches dans le but d'améliorer l'efficacité d'extraction des caractéristiques génériques des défauts au niveau de l'encodeur. Par conséquence, le réseau dans sa globalité gagnera en performance. La deuxième partie du réseau est constituée d'un décodeur pour chaque tâche. En effet, nous avons opté pour l'idée de dédier un décodeur propre à chaque tâche dans le but de permettre un apprentissage plus spécifique dans cette partie. Comme ce réseau adopte l'architecture du célèbre réseau U-Net, il faut mentionner que les deux parties (encodeur commun et décodeurs dédiés) sont reliées par des connexions résiduelles. Le schéma de la figure 4.10 décrit en détail les différents composants de cette architecture orientée apprentissage multitâche.

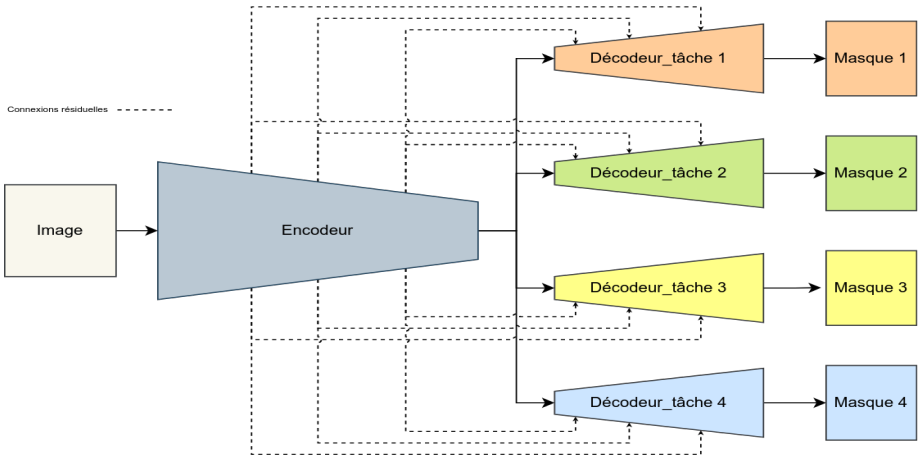


FIGURE 4.10 – Architecture à encodeur partagé.

Implémentation

Comme mentionné dans la section précédente, le réseau de segmentation à encodeur partagé proposé dans le cadre de ce travail de recherche adopte un style architectural similaire au réseau U-Net [2]. L'encodeur partagé est construit sur la base de l'architecture ResNet50 [13] avec quatre niveaux de profondeur, pré-entraîné sur l'ensemble de donnée ImageNet. Nous avons opté d'élaguer le dernier niveau de la version d'origine de l'encodeur afin d'avoir un modèle qui s'aligne sur les limites de la mémoire graphique disponible. D'autre part, l'efficacité de l'encodeur n'est pas impactée par cette modification d'après les tests effectuée préalablement. La partie décodeur de chaque tâche est implémentée en suivant le même schéma que l'encodeur ResNet50 avec quatre blocs constituant le chemin extensif du modèle. Chaque bloc est constitué d'une succession d'opérations : la convolution transposée, deux blocs de convolution et enfin un module d'attention à bloc de convolution (CBAM)[12]. Un dernier bloc de convolution s'ajoute à la fin dont le rôle est de générer le masque final de la tâche concernée. Le lien qui lie l'encodeur à chaque décodeur se présente sous forme d'un ensemble de connexions résiduelles entre chaque bloc de même niveau dans les deux parties. La figure 4.11 montre les différents aspects que nous venons de décrire.

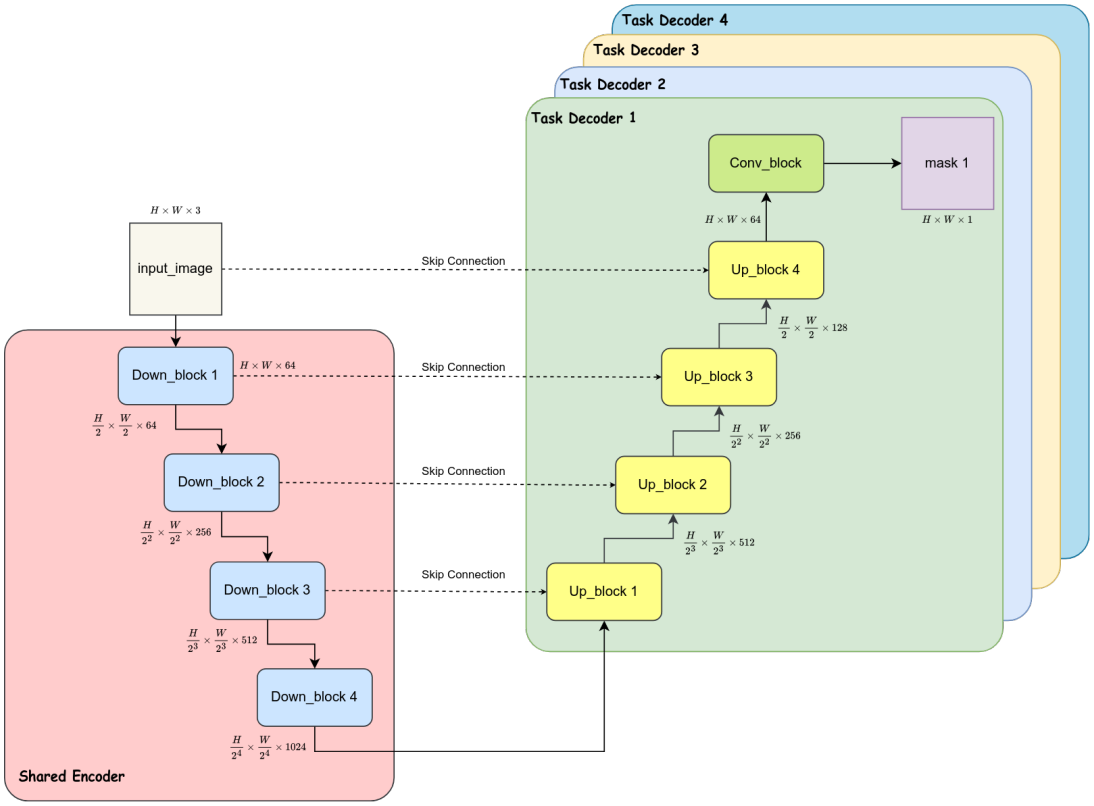


FIGURE 4.11 – Architecture détaillée du réseau à encodeur partagé.

4.3.3 Réseau à apprentissage multitâche avec encodeur-décodeur partagés

Architecture

L'architecture de la solution proposée dans cette partie adopte aussi le modèle encodeur-décodeur orienté aussi apprentissage multitâche. Néanmoins, les tâches qui seront traitées par ce réseau se verront partager à la fois l'encodeur et décodeur contrairement à la solution précédente. Comme nous pouvons le constater dans la figure 4.12, ce réseau est composé d'un encodeur qui est directement relié à un encodeur. Les deux parties reprennent la même architecture classique du réseau U-Net [2] avec un chemin qui rétrécit et l'autre qui est de nature expansif en plus des connexions résiduelles qui font le lien entre les deux. Nous avons opté de mettre le décodeur comme une partie partagée pour les différentes tâches dans l'optique d'explorer si ce dernier peut bénéficier du transfert positif entre les tâches lors de la phase d'apprentissage. Ainsi, nous aurons à la fois une architecture plus légère et efficace que celle précédente vu qu'elle possède juste un seul décodeur comparé à la précédente qui associe un décodeur par tâche. Afin de générer des masques respectifs à chaque tâche, nous avons ajouté des blocs dédiés pour chaque tâche composés d'une opération de convolution simple.

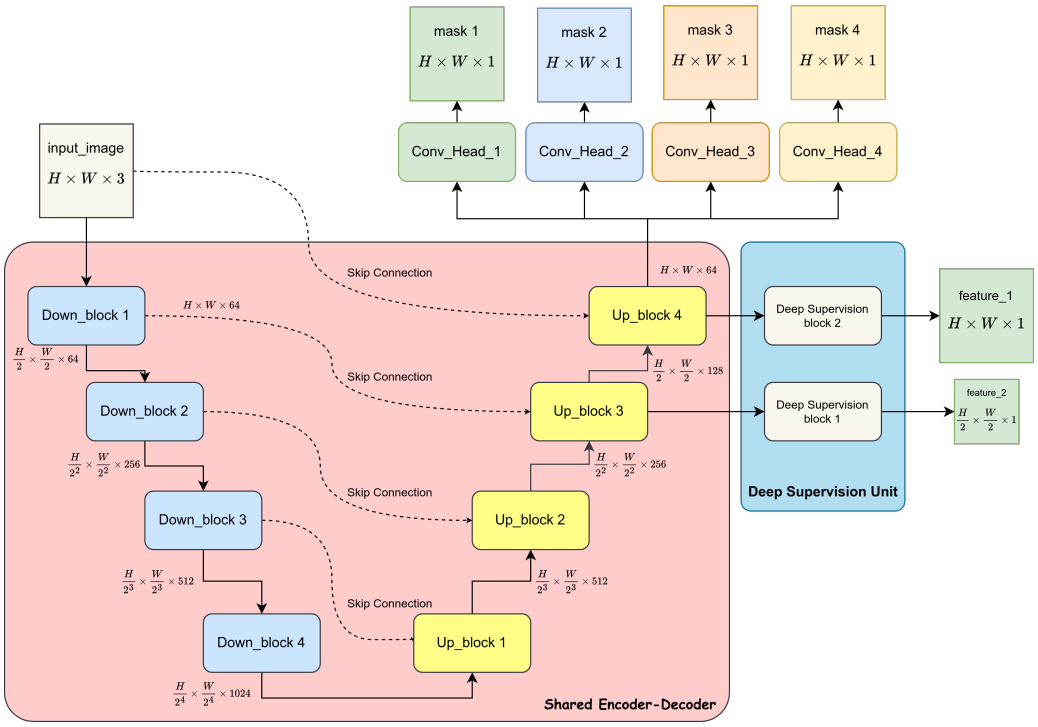


FIGURE 4.12 – Architecture encodeur-décodeur partagé avec supervision en profondeur.

Ce réseau se distingue aussi avec une unité spécialement conçue pour prendre en charge une supervision en profondeur lors de la phase d'entraînement. Cette unité

est composée de deux blocs de convolutions qui vient directement se brancher aux deux derniers blocs du décodeur. Le rôle de chaque bloc est de générer une caractéristique intermédiaire propre à chaque bloc du décodeur en prenant en entrée les caractéristiques résultantes de ces blocs au niveau du décodeur. Les représentations produites par cette unité serviront exclusivement dans la phase d'entraînement au calcul de la fonction de perte du réseau. Cela impactera ensuite le processus de la rétro-propagation en permettant ainsi d'amplifier le flux du gradient directement à partir de ces deux couches intermédiaires. Ce phénomène a pour effet d'améliorer l'apprentissage.

Implémentation

Comme mentionné précédemment, ce réseau reprend les mêmes bases d'implémentation que la solution précédente au niveau de l'encodeur. Ce dernier se présente sous une version de quatre niveaux du réseau ResNet [13]. La partie décodeur est, elle aussi, constituée de quatre blocs qui s'ajustent avec l'encodeur. Chaque bloc du décodeur intègre un module d'attention à bloc de convolution CBAM [12]. Nous avons aussi mis en place des parties dédiées pour chaque tâche à la sortie du décodeur. Chaque partie est composée d'une couche de convolution à laquelle nous avons associé une fonction d'activation de type sigmoïde. Enfin, L'unité spécialisée dans la supervision en profondeur est constituée de deux blocs qui sont directement reliés aux sorties des deux derniers blocs de l'encodeur. Chacun est composé d'une couche de convolution suivie d'une fonction d'activation sigmoïde.

Dans ce qui suit, nous distinguons deux phases d'expérimentations. La première partie est consacrée aux expérimentations réalisées sur les données d'entraînement à la fin de sa construction afin d'évaluer sa qualité. La deuxième partie est réservée aux expérimentations liées aux trois architectures proposées dans la section précédente.

4.4 Expérimentation 1

Dans cette section, nous rapporterons les résultats obtenus lors de la phase d'entraînement et de test du réseau de segmentation U-Net [2] sur l'ensemble de donnée que nous avons construit. Ces expérimentations ont été menées pour chaque classe de défaut indépendamment des autres dans le but d'évaluer la qualité de notre ensemble de données. Nous avons choisi différents encodeurs pré-entraînés sur l'ensemble de données ImageNet [14]. Cela nous a permis de faire un transfert d'apprentissage au réseau U-Net sur notre ensemble de données.

4.4.1 Données

Les expérimentations concernent l'ensemble d'images d'inspection que nous avons mis en place dans les étapes précédentes. Il est constitué de trois sous ensembles partitionnés de manière équilibrée et consistante. L'ensemble d'entraînement comprend 450 images et les ensembles de validation et test en comptent 72 chacun. L'histogramme de la figure 4.13 illustre bien cette répartition.

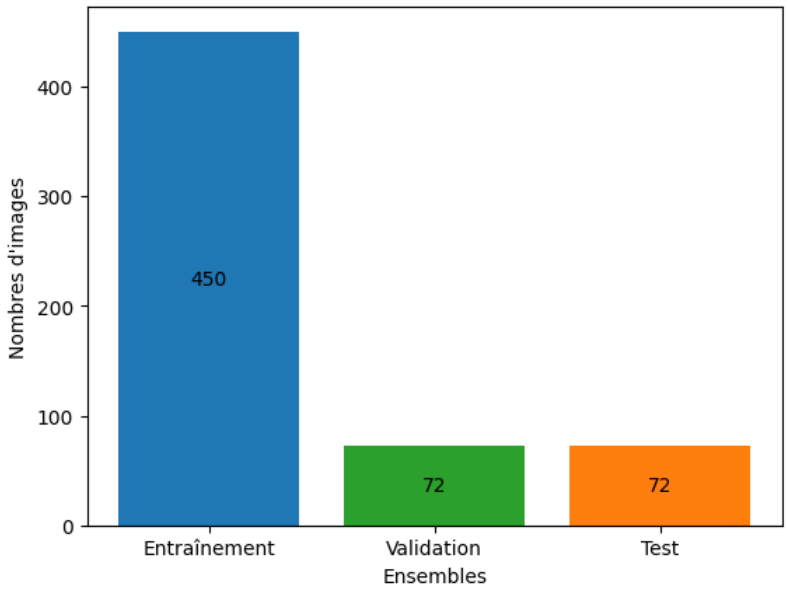


FIGURE 4.13 – Histogramme de la répartition des images sur les ensembles d'entraînement , validation et de test.

4.4.2 Modèle de segmentation

Le choix du réseau de segmentation U-Net[2] dans un premier temps se justifie par le fait qu'il a réalisé ces preuves dans le domaine avec des performances très louables. Comme nous l'avons bien rapporté dans la section 2.1.4, le réseau U-Net est constitué principalement de deux parties qui sont l'encodeur et le décodeur. Dans les expérimentations que nous avons menées, nous avons choisi de varier le type

d'encodeur du réseau. En résumé, nous avons six versions du réseau U-Net avec les encodeurs suivants pré-entraînés sur ImageNet :

— ResNet [13].
— DenseNet [49].
— Xception [50].
— MobileNet [51].
— EffecientNet [19].
— Mix Transformer [52].

Les paramètres des encodeurs seront figés lors de la phase d'entraînement des modèles dans le but d'exploiter au mieux les paramètres appris par les encodeurs préalablement. Pour créer contexte commun d'évaluation pour tous les modèles, nous avons fixé le nombre d'itérations de la phase d'entraînement à 100.

4.4.3 Code et implémentation

Nous avons eu recours à la libraire PyTorch pour mettre en place notre réseau U-Net et l'évaluer sur notre ensemble de données. PyTorch est une librairie python à code source ouvert développée par Meta pour le domaine de l'apprentissage machine et profond. Elle offre des fonctions et des outils qui permettent de concevoir, entrainer, évaluer et déployer un modèle d'apprentissage profond. Nous avons aussi utilisé la librairie segmentation-models-pytorch [48]. Cette librairie est basée sur Pytorch et elle est conçue pour la segmentation d'image. Elle offre une collection de 9 réseaux de segmentation avec une combinaison de 124 encodeurs. En plus, elle inclut aussi des métriques d'évaluation et des fonctions de perte dédiées à la tâche de segmentation.

Le code de la figure 4.14, illustre comment mettre en place un réseau U-Net avec un encodeur de type ResNet pré-entraîné sur ImageNet.

```python
import segmentation_models_pytorch as smp

model = smp.Unet(
    encoder_name="resnet34",        # Choisir l'encodeur du réseau
    encoder_weights="imagenet",     # Utiliser `imagenet` pour les paramètres pre-entraînés
    in_channels=1,
    classes=3,
)
```

FIGURE 4.14 – Exemple de code pour instancier le réseau U-Net avec un encodeur de type ResNet [13] pre-entraîné sur ImageNet [14].

4.4.4 Résultats

Dans cette section, nous présenterons les résultats quantitatifs et qualitatifs des expérimentations décrites précédemment. Les résultats sont organisés par classes de défaut de notre ensemble de données.

Corrosion

Les graphes de la figure 4.15 représentent l'évolution des courbes de la fonction de perte 4.15a et du coefficient Dice 4.15b lors de la phase d'entrainement sur les donnes d'entraînement et de validation :

Le tableau 4.1 rapporte l'ensemble des résultats quantitatifs des meilleures versions des modèles sauvegardés lors de la phase d'entraînement. Ces résultats concernent les ensembles de validation et de test.

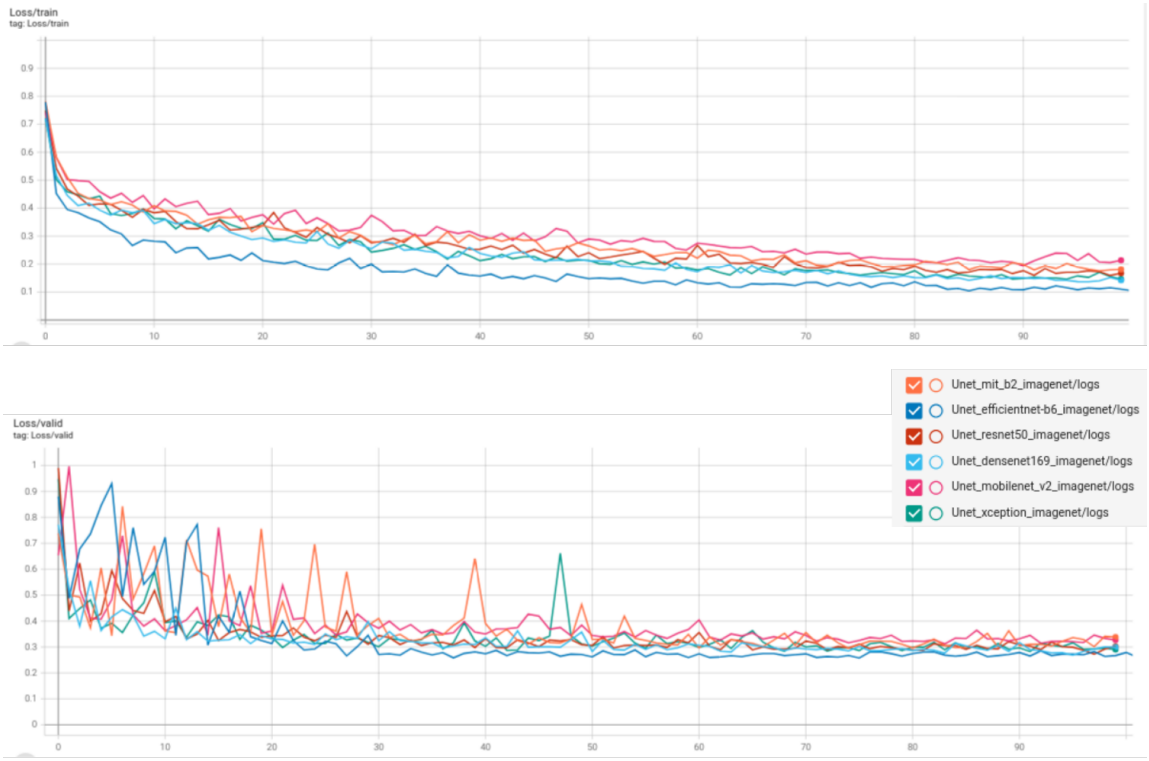
TABLE 4.1 – Résultats quantitatifs des modèles sur la classe corrosion

	Validation					Test				
Encodeur	Loss	Acc	Dice	Prec	Recall	Loss	Acc	Dice	Prec	Recall
ResNet[13]	0.33	0.98	0.67	0.78	0.60	0.31	0.96	0.69	0.87	0.57
DenseNet[49]	0.31	0.97	0.68	0.73	0.66	0.30	0.96	0.69	0.85	0.59
Xception[50]	0.33	0.97	0.67	0.77	0.60	0.30	0.96	0.69	0.827	0.60
MobileNet[51]	0.36	0.97	0.63	0.65	0.63	0.31	0.96	0.68	0.78	0.63
EffecientNet[19]	0.34	0.97	0.65	0.7	0.64	**0.27**	0.96	**0.72**	0.85	0.64
MixTrans[52]	0.36	0.97	0.63	0.65	0.63	0.28	0.96	0.71	0.8	0.64

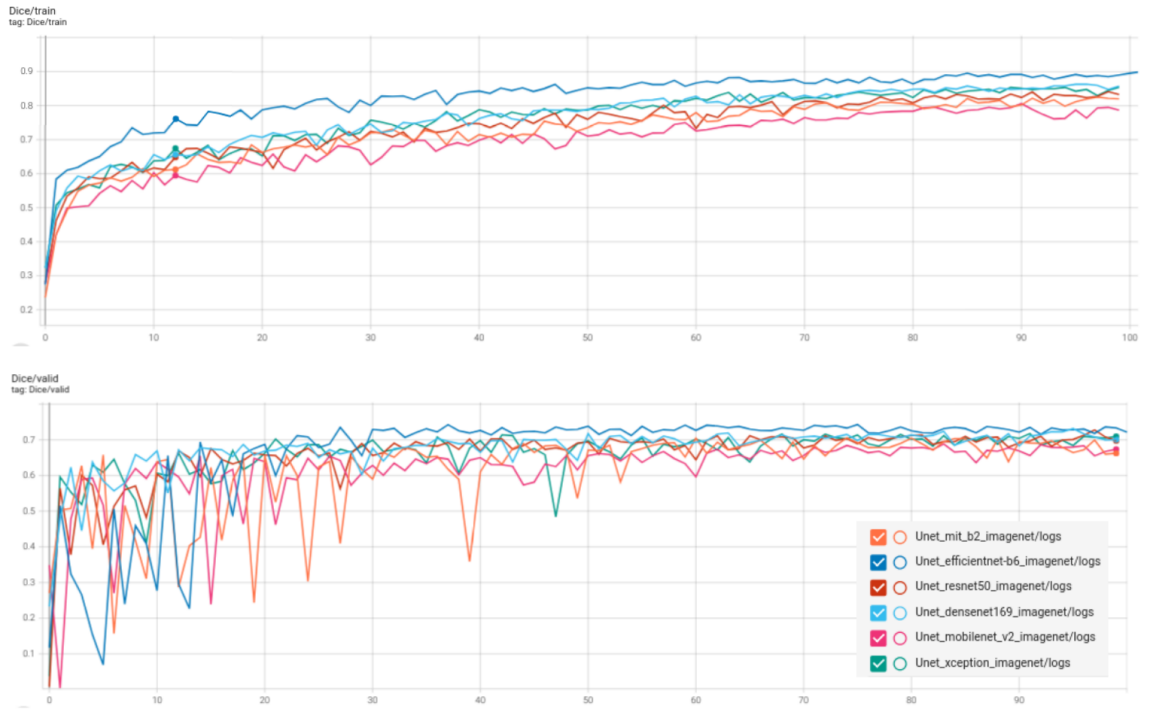
Discussion des résultats : Les graphes de la fonction de perte de la figure 4.15a montre que les modèles testés convergent tous après un certain nombre d'itérations. Néanmoins, nous constatons que les courbes de la fonction de perte subissent des perturbations et des variations inhabituelles dans les premières itérations sur l'ensemble de validation seulement. Pour les courbes d'évolution du coefficient dice 4.15b, nous constatons aussi une progression standard avec une convergence et une stabilisation les dernières itérations autour de la valeur de 70%.

Les résultats du tableau 4.1, montrent clairement que le modèle U-Net avec l'encodeur EffecientNet [19] réalise le meilleur coefficient dice sur l'ensemble de test avec un score de 72%. Pour les modèles restants, nous observons qu'il n'y a pas un grand écart de performance entre eux.

Évaluation qualitative : La figure 4.16 montre un aperçu des prédictions qui sont faites par les six modèles sur un ensemble d'images :



(a) Les courbes de la fonction de perte.



(b) Évolution des courbes du coefficient Dice.

FIGURE 4.15 – Les Courbes de la phase d'entraînement

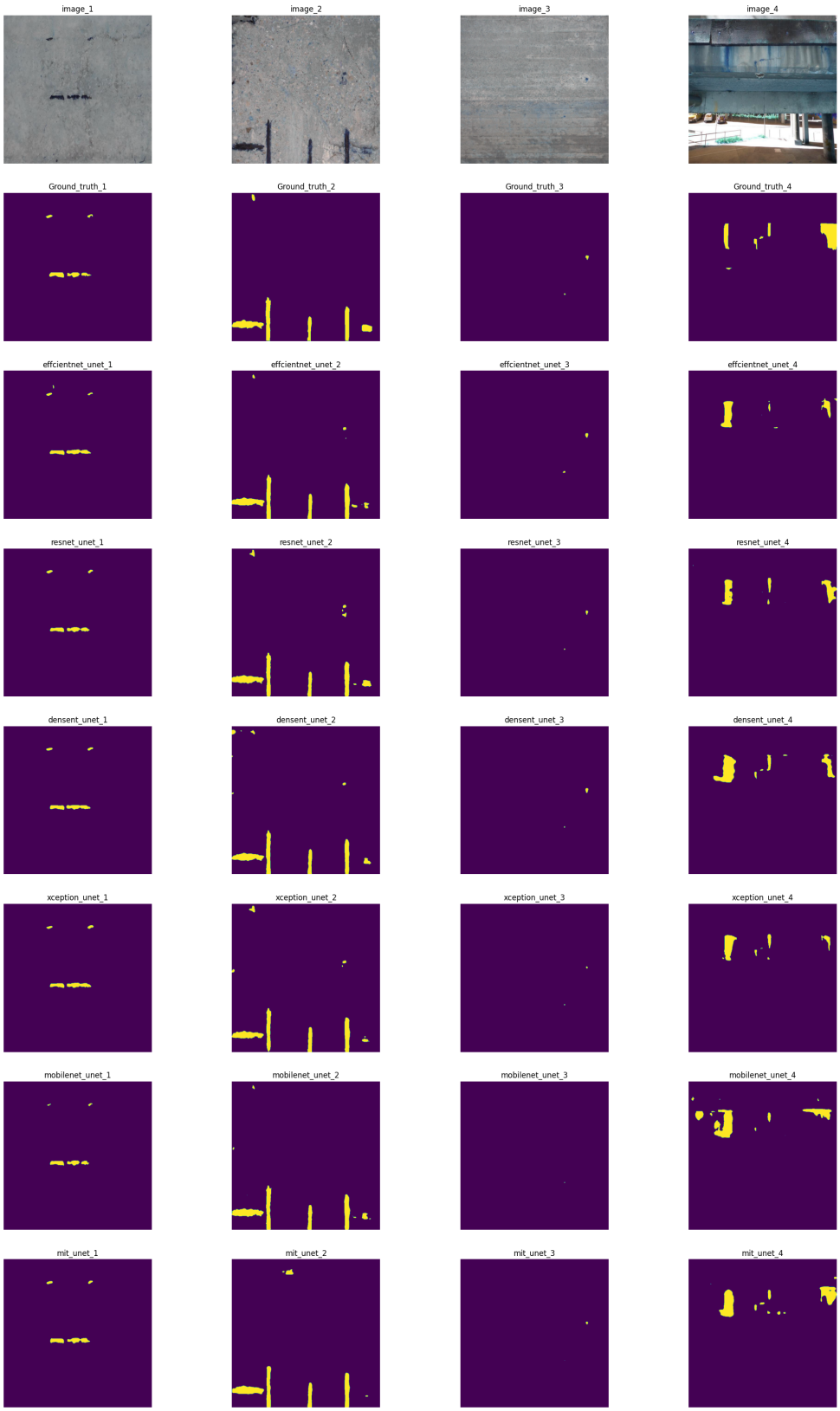


FIGURE 4.16 – Aperçu des résultats des prédictions des six modèles sur des images de l'ensemble de test.

Efflorescence

(a) Les courbes de la fonction de perte.

(b) Évolution des courbes du coefficient Dice.

FIGURE 4.17 – Les Courbes de la phase d'entraînement

Les graphes de la figure 4.17 représentent l'évolution des courbes de la fonction de perte 4.17a et du coefficient Dice 4.17b lors de la phase d'entrainement sur les donnes d'entraînement et de validation.

Le tableau 4.2 rapporte l'ensemble des résultats quantitatifs obtenus lors des expérimentations. Ces résultats sont réalisés par les meilleures versions sauvegardées des modèles testés lors de la phase d'entraînement. Ces résultats concernent les ensembles de validation et de test.

TABLE 4.2 – Résultats quantitatifs des modèles sur la classe efflorescence.

	Validation					Test				
Encodeur	Loss	Acc	Dice	Prec	Recall	Loss	Acc	Dice	Prec	Recall
ResNet[13]	0.39	0.97	0.60	0.74	0.53	0.62	0.98	0.37	0.54	0.30
DenseNet[49]	0.42	0.97	0.57	0.66	0.52	0.63	0.98	0.36	0.41	0.40
Xception[50]	0.34	0.97	0.65	0.72	0.64	0.63	0.98	0.37	0.50	0.30
MobileNet[51]	0.46	0.97	0.53	0.68	0.48	0.74	0.98	0.25	0.32	0.21
EffecientNet[19]	0.33	0.98	0.66	0.76	0.61	0.67	0.98	0.32	0.41	0.30
MixTrans[52]	0.31	0.98	0.68	0.73	0.65	0.66	0.98	0.33	0.5	0.34

Discussion des résultats : Les expérimentations sur la classe de défaut "efflorescence" nous ont permis de constater que les six modèles testés ont convergé. Ce constat est établi en s'appuyant sur l'évolution des courbes de la fonction de perte de la figure 4.17a qui atteignent une certaine stabilité dans les dernières itérations. Dans l'autre côté, l'évolution des courbes du coefficient Dice suit une trajectoire classique. Une première partie où nous notons une progression puis un autre phase qui se résume par une stabilisation des valeurs.

Dans le tableau 4.2, nous constatons que les modèles avec des encodeurs : efficientNet [19] et Mix Transformer [52] ont réalisé les meilleurs scores du coefficient dice dans l'ensemble de validation. Les autres modèles suivent la même tendance avec des scores qui se situent dans un voisinage de 60%. Par contre, les expérimentations menées sur l'ensemble de test montrent une baisse de résultats comparés à ceux de l'ensemble de validation. Cette baisse de résultats est aussi constatée dans les résultats de l'évaluation qualitative comme le montre la figure 4.18.

Évaluation qualitative : La figure 4.18 montre un aperçu des prédictions qui sont faites par les six modèles sur un ensemble d'images :

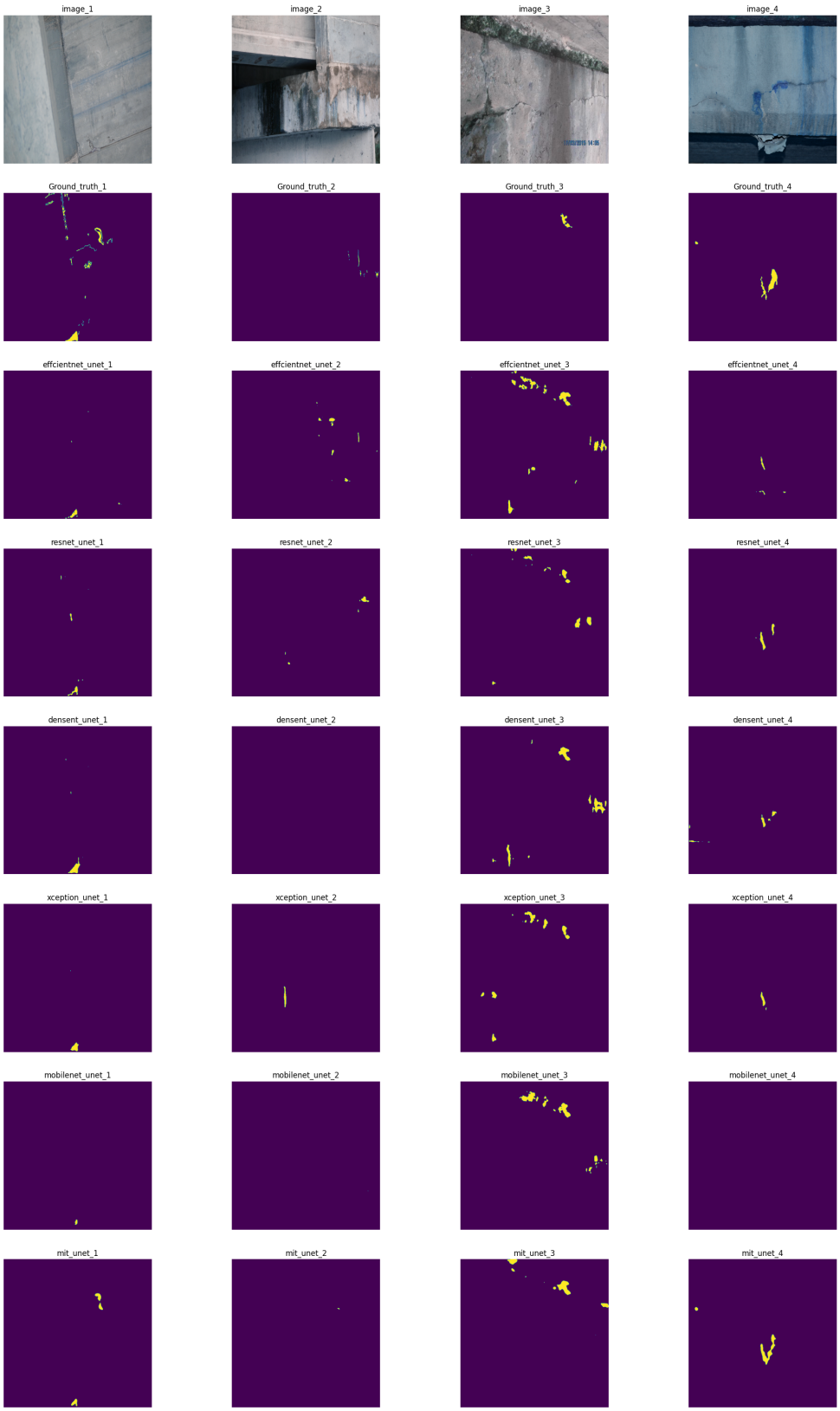
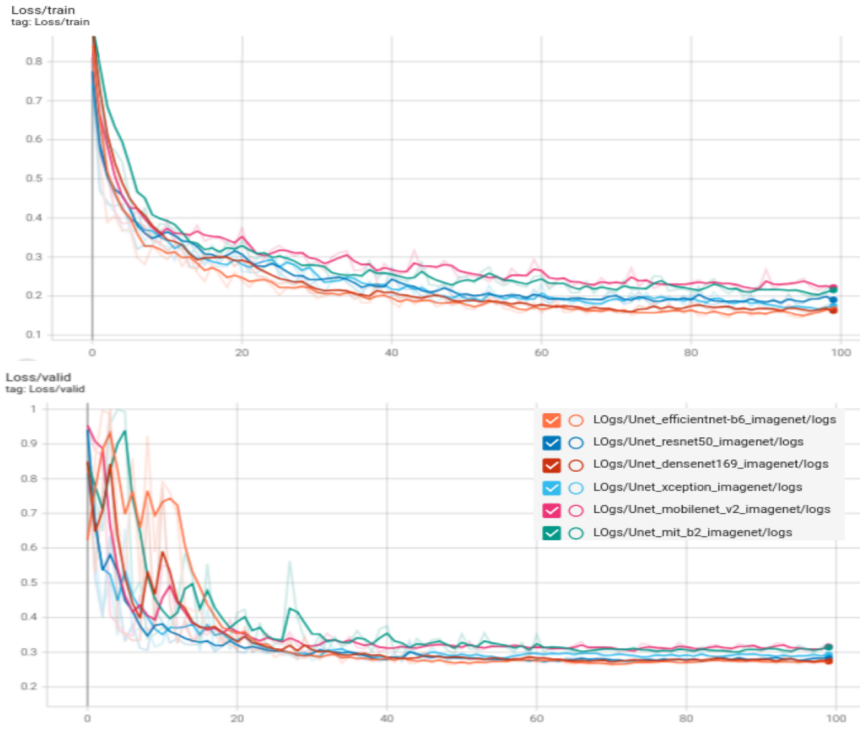
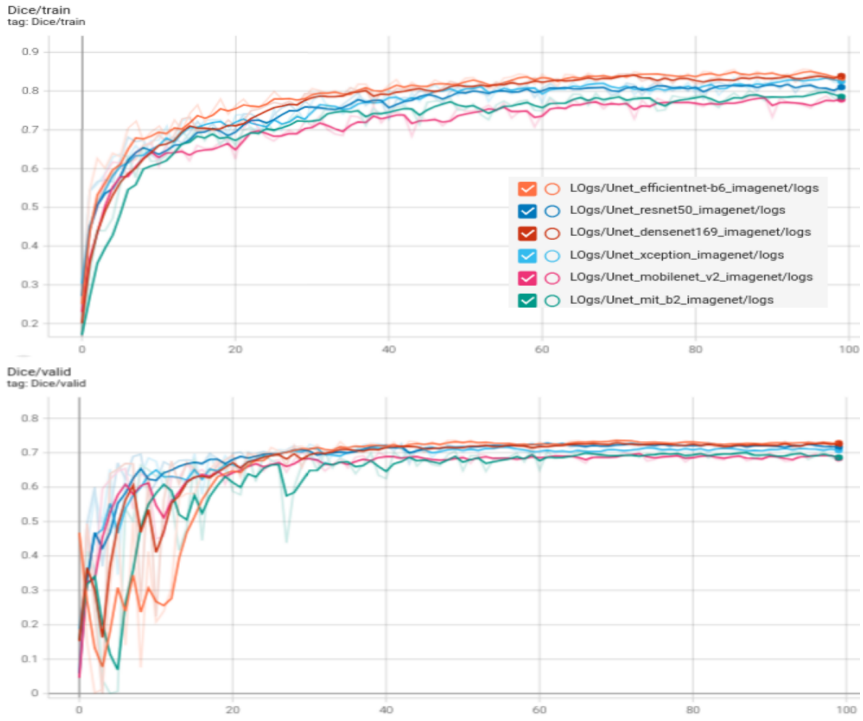


FIGURE 4.18 – Aperçu des résultats des prédictions des six modèles sur des images de l'ensemble de test.

Barres exposées



(a) Les courbes de la fonction de perte.



(b) Évolution des courbes du coefficient Dice.

FIGURE 4.19 – Les Courbes de la phase d'entraînement

Les graphes de la figure 4.19 représentent l'évolution des courbes de la fonction de perte 4.19a et du coefficient Dice 4.19b lors de la phase d'entrainement sur les donnes d'entraînement et de validation.

Le tableau 4.3 rapporte l'ensemble des résultats quantitatifs des meilleures versions des modèles sauvegardés lors de la phase d'entraînement. Ces résultats concernent les ensembles de validation et de test.

TABLE 4.3 – Résultats quantitatifs des modèles sur la classe Barre exposée.

	Validation					Test				
Encodeur	Loss	Acc	Dice	Prec	Recall	Loss	Acc	Dice	Prec	Recall
ResNet[13]	0.30	0.98	0.69	0.68	0.74	0.31	0.98	0.68	0.69	0.70
DenseNet[49]	0.30	0.98	0.7	0.66	0.75	0.32	0.98	0.68	0.67	0.70
Xception[50]	0.31	0.98	0.68	0.68	0.71	0.33	0.98	0.67	0.66	0.69
MobileNet[51]	0.32	0.97	0.67	0.63	0.73	0.35	0.98	0.64	0.62	0.68
EffecientNet[19]	0.30	0.98	0.69	0.68	0.73	0.28	0.98	0.71	0.69	0.77
MixTrans[52]	0.31	0.98	0.68	0.68	0.69	0.31	0.98	0.69	0.68	0.71

Discussion des résultats : Les résultats des expérimentations menées sur la classe de "barre exposée" montrent que les six modèles testés convergent lors de l'entraînement. En effet, les courbes de la fonction de perte 4.19a montrent clairement que les modèles se stabilisent dans les dernières itérations du processus d'entraînement. Dans l'autre côté, l'évolution des courbes du coefficient Dice 4.19b suit un comportement similaire qui reflète la convergence des modèles. Dans un premier temps, le coefficient dice augmente progressivement jusqu'à atteindre une certaine stabilité autour de 80 % pour l'ensemble d'entraînement et 70 % pour l'ensemble de validation.

Concernant le tableau 4.3, les six modèles ont réalisé des résultats qui sont proches l'un des autres en termes du coefficient de dice. Les résultats varient de 67 % à 70 % pour l'ensemble de validation tandis que ceux de l'ensemble de test se situent entre 64 % et 71 %.

Évaluation qualitative : La figure 4.20 montre un aperçu des prédictions qui sont faites par les six modèles sur un ensemble d'images :

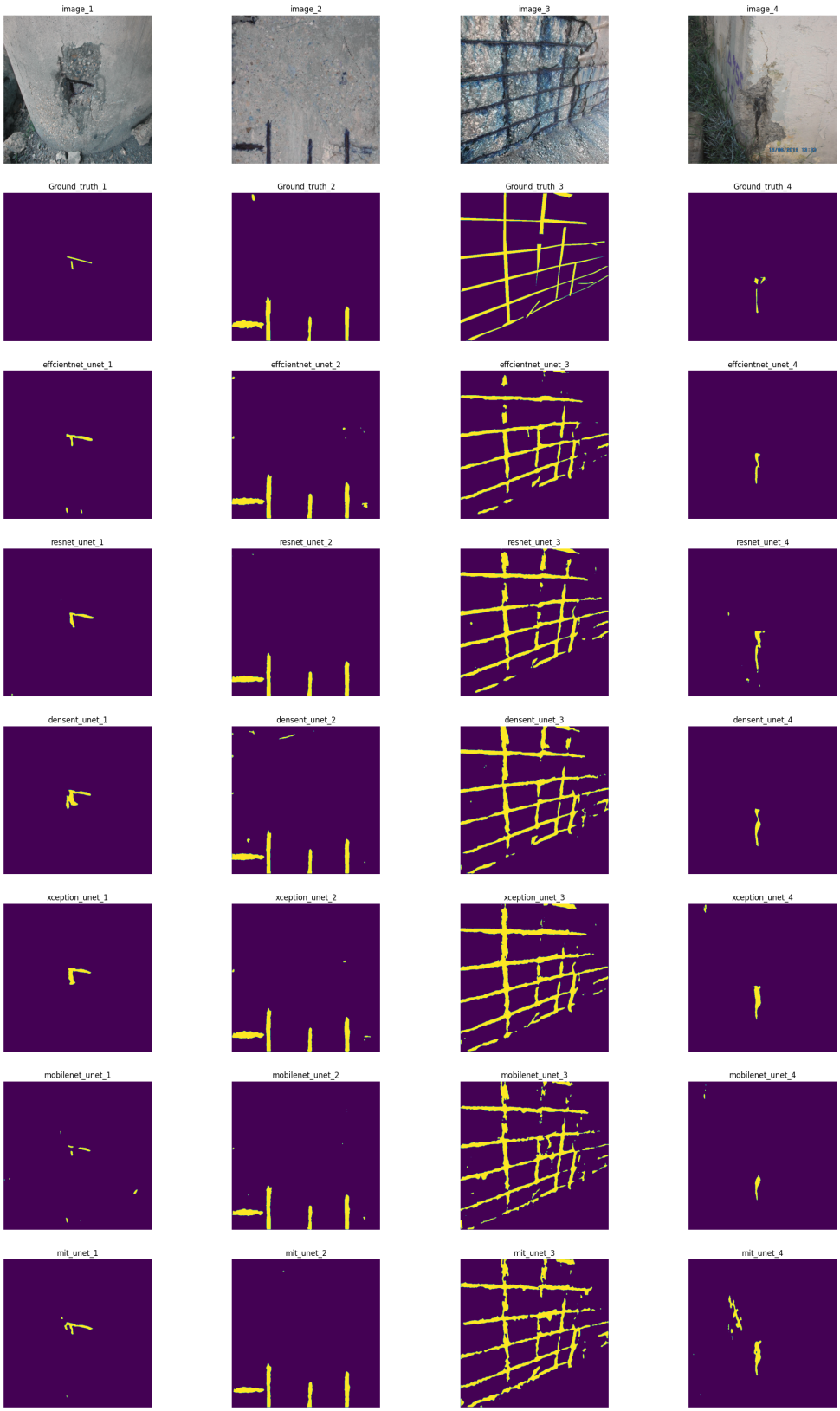
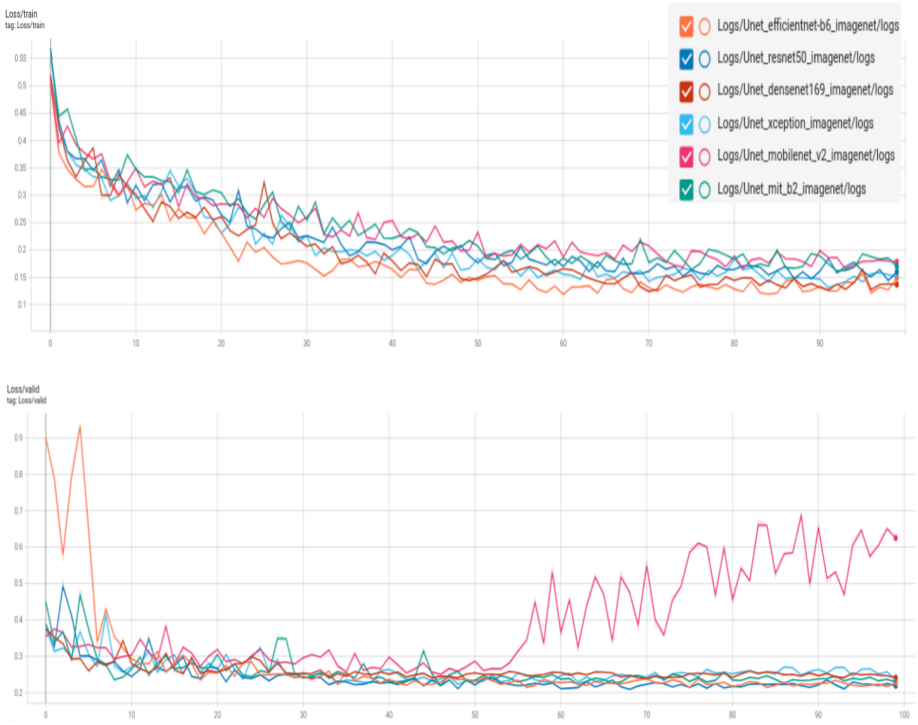
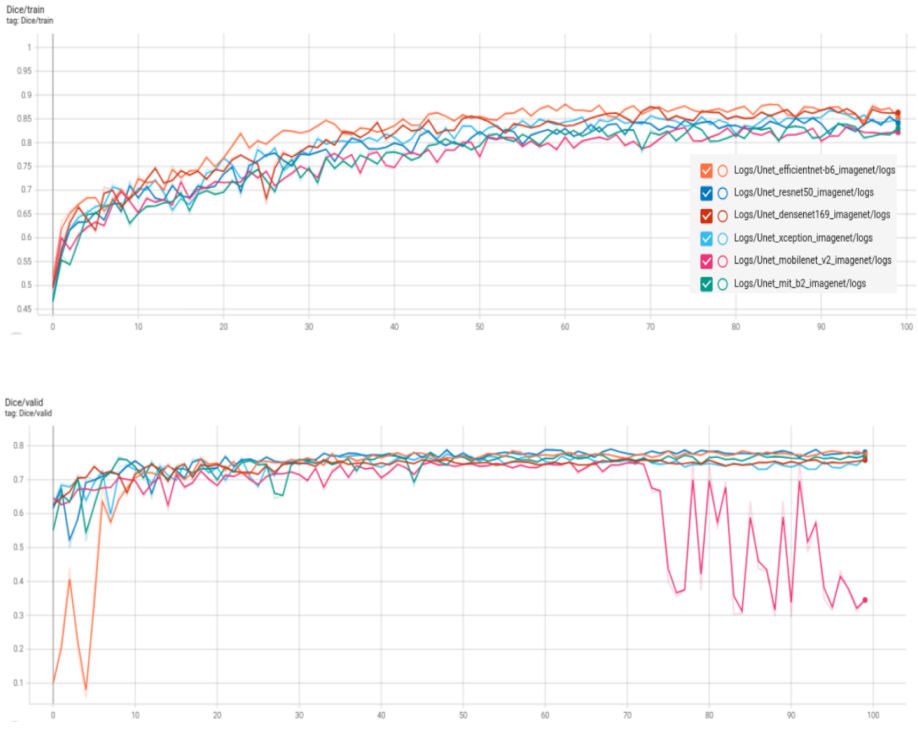


FIGURE 4.20 – Aperçu des résultats des prédictions des six modèles sur des images de l'ensemble de test.

Dégradation du béton



(a) Les courbes de la fonction de perte.



(b) Évolution des courbes du coefficient Dice.

FIGURE 4.21 – Les Courbes de la phase d'entraînement

Les graphes de la figure 4.21 représentent l'évolution des courbes de la fonction de perte 4.21a et du coefficient Dice 4.21b lors de la phase d'entraînement sur les donnes d'entraînement et de validation.

Le tableau 4.4 rapporte l'ensemble des résultats quantitatifs des meilleures versions des modèles sauvegardés lors de la phase d'entraînement. Ces résultats concernent les ensembles de validation et de test.

TABLE 4.4 – Résultats quantitatifs des modèles sur la classe Dégradation de béton.

	Validation					Test				
Encodeur	Loss	Acc	Dice	Prec	Recall	Loss	Acc	Dice	Prec	Recall
ResNet[13]	0.27	0.94	0.73	0.81	0.67	0.23	0.90	0.76	0.81	0.72
DenseNet[49]	0.28	0.93	0.71	0.75	0.70	0.22	0.90	0.77	0.77	0.77
Xception[50]	0.28	0.93	0.71	0.79	0.65	0.23	0.91	0.76	0.82	0.72
MobileNet[51]	0.48	0.84	0.57	0.50	0.71	0.38	0.84	0.67	0.61	0.76
EffecientNet[19]	0.25	0.94	0.75	0.79	0.72	0.20	0.91	0.79	0.78	0.80
MixTrans[52]	0.25	0.94	0.74	0.84	0.68	0.22	0.90	0.78	0.77	0.79

Discussion des résultats : Pour la classe "Dégradation de béton", nous avons constaté que les modèles ont tous convergé sauf un. Le modèle en question est le U-Net avec l'encodeur MobileNet[51]. Comme nous pouvons le voir dans la figure 4.21a, la courbe de sa fonction de perte augmente de façon significative dans les dernières itérations alors qu'elle est supposée se stabiliser. Le même constat est observé au niveau de la courbe du coefficient de dice. Dans la figure 4.21b, nous notons que dans la progression des courbes du dice pour les données d'entraînement, tous les modèles ont eu une trajectoire standard avec une stabilisation vers la fin de processus. Or, pour les données de l'ensemble de validation, nous constatons la non-convergence du même modèle. Pour les restants, nous observons une stabilisation des courbes vers un de dice de 75%.

Les résultats du tableau 4.4 montre que le modèle U-Net avec l'encodeur MobileNet[51] a réalisé le plus faible résultat. Pour les restants, le modèle U-Net avec EffecientNet[19] est le celui avec la meilleure performance, tandis les autres sont aussi assez proches de lui.

Évaluation qualitative : La figure 4.22 montre un aperçu des prédictions qui sont faites par les six modèles sur un ensemble d'images :

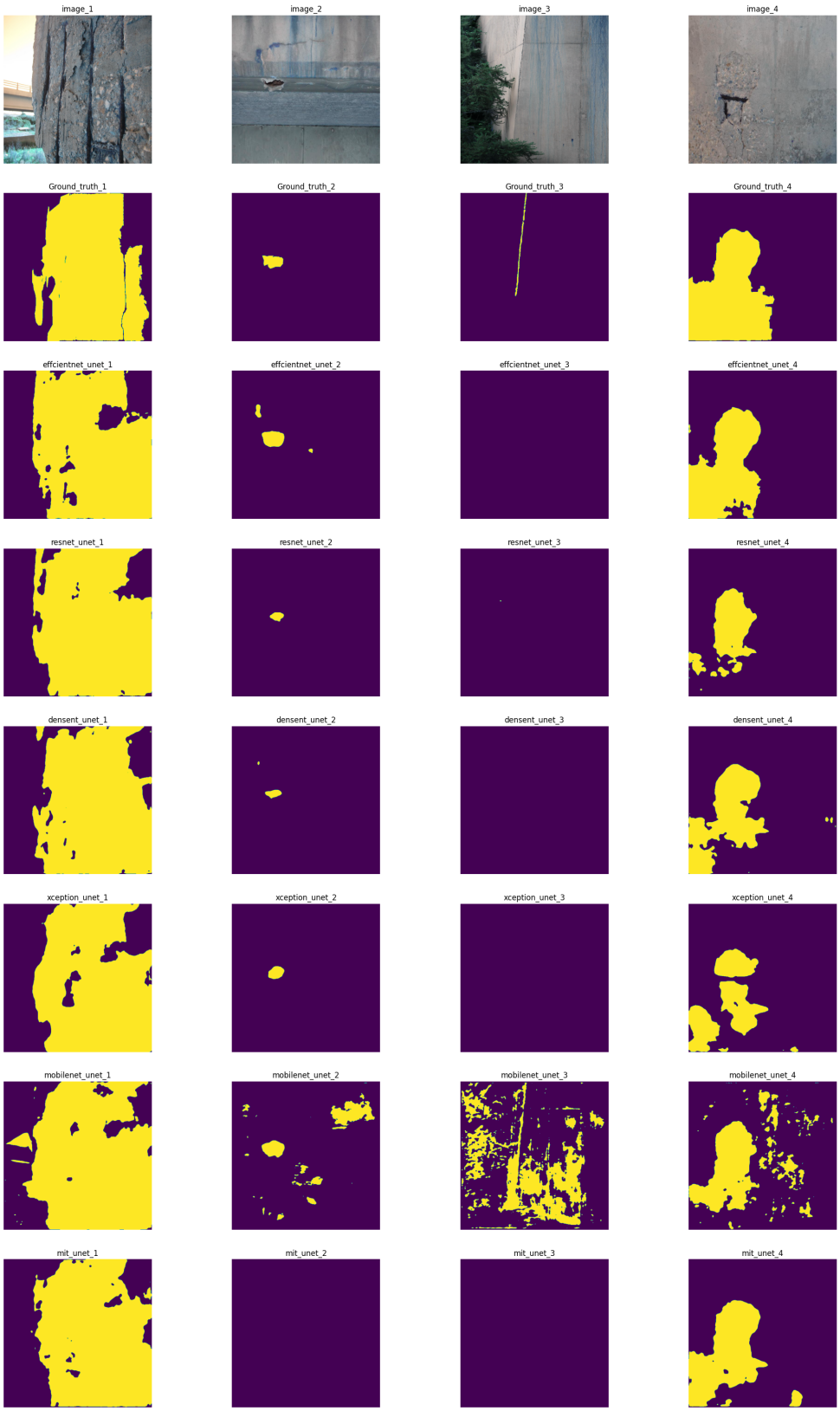


FIGURE 4.22 – Aperçu des résultats des prédictions des six modèles sur des images de l'ensemble de test.

4.5 Expérimentation 2

Cette deuxième phase d'expérimentations concerne les trois architectures à apprentissage multitâche proposées pour répondre à la problématique de la segmentation des défauts dans le processus d'inspection des ponts en béton. Elle retrace les détails clés des expérimentations avant d'exposer les résultats quantitatifs et qualitatifs obtenus par les modèles.

4.5.1 Entraînement

La phase d'entraînement de ces réseaux s'est effectuée sur l'ensemble de données que nous avons construit auparavant. Pour rappel, ce dernier comporte des images d'inspection de ponts avec des masques pour quatre classes de défauts. La partie encodeur du réseau est gelée afin de ne pas subir d'apprentissage vu qu'elle a été pré-entraînée sur ImageNet. Cela est jugé suffisant, car cet aspect lui confère une efficacité reconnue dans l'extraction des caractéristiques génériques des objets. En ce qui concerne la partie décodeur, elle sera affectée et sujette aux modifications et à l'apprentissage de cette phase d'entraînement.

Nous avons fixé le nombre d'itérations à 200. Ce choix se justifie par le fait que dans la majorité des expérimentations que nous avons menées préalablement, les modèles ont tendance à atteindre leurs points de convergence en moins de 200 itérations.

Pour le paramètre **batch-size**, nous avons opté de le fixer à 8. Cette valeur est un compromis entre le fait de s'aligner aux contraintes matérielles de la station, plus précisément la mémoire graphique disponible, et la vitesse d'apprentissage ainsi que la qualité de la généralisation du modèle.

Comme ces réseaux adoptent le paradigme d'apprentissage multitâche, une itération s'effectue sur l'ensemble des défauts, sans chevauchement et de manière successive, comme illustré dans la figure 4.23. Cette stratégie est simple, néanmoins elle garantit qu'il n'y aura moins de conflit et divergence entre les tâches lors de ce processus d'apprentissage.

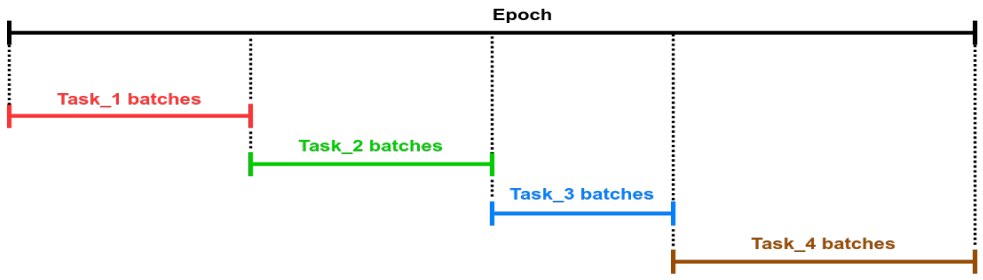


FIGURE 4.23 – Illustration de la stratégie d'entraînement au niveau d'une seule itération.

4.5.2 Augmentation de données

Dans cette phase d'entraînement, nous avons inclus une étape importante dans le pipeline d'acheminement des données. Cette dernière est l'augmentation de donnée. Elle a pour rôle d'augmenter notre ensemble d'entraînement afin d'amener de la diversité dans les données au modèle pour qu'il puisse atteindre une meilleure généralisation. Cette étape est composée de deux opérations qui s'appliquent successivement de la même manière à l'image et son masque :

Retournement Horizontal : Bascule l'image sur son axe horizontal avec une probabilité d'occurrence :$p = 0.5$

Rotation : Procède à une rotation de l'image avec un angle compris dans $[0, 10]$ avec une probabilité d'occurrence :$p = 0.5$

4.5.3 Fonction de perte

C'est un élément central dans le processus d'apprentissage d'un réseau. En effet, cette fonction permet d'évaluer la qualité de l'apprentissage du modèle à chaque itération et ensuite le processus de la rétropropagation du gradient mis à jour les paramètres du modèle en partant du résultat de cette dernière.

Architecture à encodeur partagé

Dans le cas de cette architecture, nous avons opté pour une fonction de perte combinée. Cette fonction de perte est une somme de deux fonctions : la fonction Dice et l'entropie croisé binaire. L'équation 4.2 illustre cette fonction de perte.

$$F_{perte}(Y, \hat{Y}) = F_{ecb}(Y, \hat{Y}) + F_{dice}(Y, \hat{Y}) \tag{4.2}$$

Dans l'équation 4.2, Y est le masque de la réalité terrain, et $\hat{Y}$ est le masque prédit par le modèle.

Comme le dénote bien l'équation 4.3, l'entropie croisée binaire a été choisie dans l'optique de mesurer la divergence entre la distribution de probabilité des pixels réels y_i et celle des pixels prédits $\hat{y_i}$. N étant le nombre de pixels total dans le masque. Lorsque les prédictions du modèle correspondent exactement aux valeurs réelles, la cross-entropie binaire est minimale.

$$F_{ecb} = -\frac{1}{N}\sum_{i=1}^{N} y_i . \log(\hat{y_i}) + (1 - y_i) . \log(1 - \hat{y_i}) \tag{4.3}$$

De même pour la dice, cette fonction a pour but de mesurer le degré de similitude des régions segmentées. L'équation 4.4 illustre bien cette fonction.

$$F_{dice}(Y, \hat{Y}) = \frac{2 \times Y \cup \hat{Y}}{Y \cap \hat{Y}} \tag{4.4}$$

Architecture à encodeur-décodeur partagés

Comme l'architecture présentée dans 4.3.3 comporte une unité dédiée à la supervision en profondeur, la fonction de perte associée à ce réseau est sophistiquée comparée à la précédente vu qu'elle prend en compte aussi les deux supervisions auxiliaires. Au niveau de l'unité spécialisée pour la supervision en profondeur, les deux blocs en question se verront associé une fonction de perte intermédiaire dont l'équation est illustrée dans 4.5. Cette fonction prend en entrée deux arguments, le premier $\hat{Y}_{ds}$ est la caractéristique sortante du bloc de la supervision en profondeur et le deuxième Y est le masque de la réalité du terrain redimensionné de sorte à avoir les mêmes dimensions que le premier argument.

$$F_{perte\_intermidiare}(\hat{Y}_{ds}, Y) = F_{ecb}(\hat{Y}_{ds}, Y) + F_{dice}(\hat{Y}_{ds}, Y) \tag{4.5}$$

La fonction de perte générale est la somme des deux fonctions de perte intermédiaires avec la fonction de perte standard calculée avec le masque final prédit et le masque de la réalité du terrain. La figure ci-dessous illustre bien la fonction de perte globe du modèle.

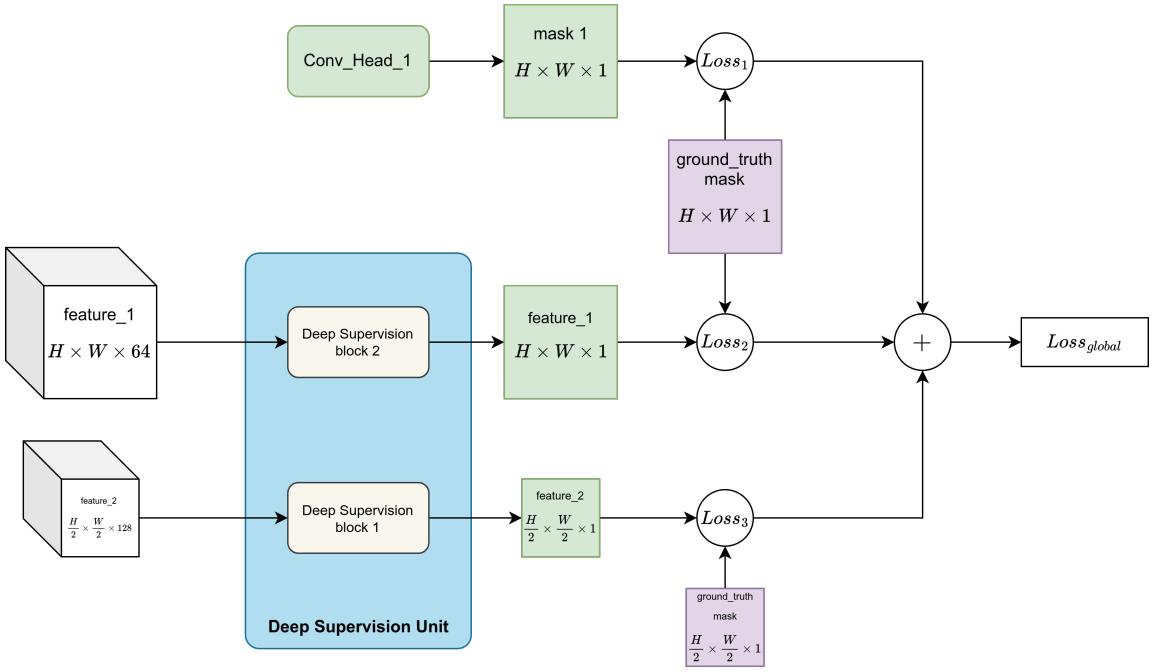


FIGURE 4.24 – Diagramme illustrant le calcul de la fonction de perte globale du réseau.

4.5.4 Algorithme d'optimisation

Le choix d'un algorithme d'optimisation du gradient est cruciale pour l'entraînement des modèles proposés via ce travail de recherche. Le rôle d'un tel algorithme est de minimiser la fonction de perte lors de la phase d'entraînement. Lors des expérimentations sur les réseaux proposés, nous avons choisi d'employer l'optimiseur Adam [53]. Cet algorithme est très populaire dans le domaine de l'apprentissage profond et combine à la fois les avantages des optimiseurs : AdaGrad [54] et RMSProp.

```
# define the optimzer and learning scheduler:
optimizer = torch.optim.Adam(myModel.parameters(), lr=1e-3)
scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer=optimizer,
                                                   gamma=0.98)
```

FIGURE 4.25 – Code d'implémentation de l'algorithme d'optimisation et de son ordonnanceur.

Nous avons fixé le taux d'apprentissage initial à 0.001, et nous avons défini un ordonnanceur qui ajustera la valeur de ce taux au fil des 200 itérations. L'ordonnanceur adopte une stratégie qui consiste à diminuer le taux d'apprentissage au fil des itérations d'entraînement en suivant une fonction exponentielle comme le montre le graphe de la figure 4.26.

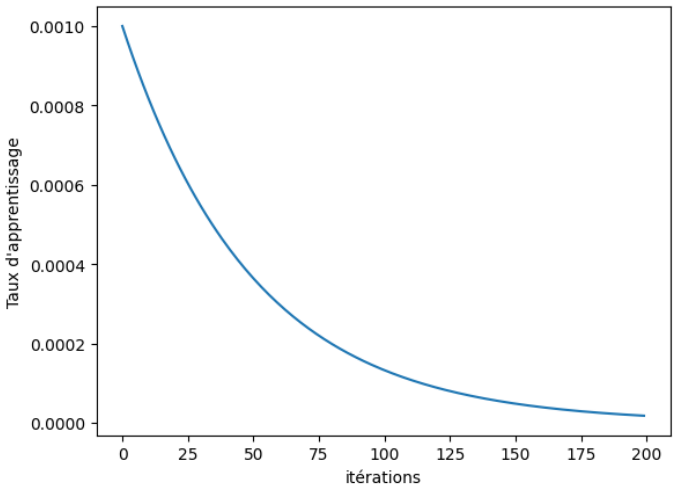


FIGURE 4.26 – Progression du taux d'apprentissage en fonction des itérations.

4.5.5 Spécifications matérielles

Afin de reproduire les mêmes conditions d'expérimentation pour les architectures proposées dans ce travail de recherche, nous énumérons les détails et caractéristiques logiciels et matérielles de la station sur laquelle les réseaux se sont entraînés :

TABLE 4.5 – Caractéristiques matérielles et logicielle de la station d'expérimentation.

Système d'exploitation :	Ubuntu 22.04 LTS
Processeur de calcul :	AMD ryzen threadripper pro 5955wx 16-cores
Processur graphique :	2 × NVIDIA RTX A6000
Mémoire :	256 Gio
Version python :	3.11.4

4.5.6 Résultats

Dans cette section, nous aborderons les résultats quantitatifs et qualitatifs des expérimentations conduites sur les modèles exposés précédemment.

Résultats quantitatifs

Le tableau 4.6 résume en détails les résultats obtenus lors des expérimentations menées sur les 3 architectures proposées dans ce travail.

TABLE 4.6 – Résultats quantitatifs des expérimentations sur les modèles.

		Corrosion		Barre exposée		Dégrad béton		Efflorescence	
Modèle	Données	Loss	Dice	Loss	Dice	Loss	Dice	Loss	Dice
U-Net[2]	train	0.12	0.91	0.14	0.88	0.09	0.95	0.19	0.84
	valid	0.46	0.7	0.51	0.69	0.53	0.81	0.68	0.6
	test	0.58	**0.68**	0.43	**0.73**	0.71	0.76	0.81	0.34
Encodeur partagé	train	0.08	0.93	0.07	0.94	0.05	0.97	0.13	0.89
	valid	0.43	0.72	0.68	0.69	0.49	0.83	0.50	0.60
	test	0.67	0.60	0.57	0.70	0.71	**0.79**	0.62	0.50
Encodeur-décodeur partagés	train	1.77	0.93	2.62	0.87	0.92	0.97	2.64	0.82
	valid	2.45	0.69	3.4	0.67	1.73	0.81	2.75	0.65
	test	2.27	0.66	3.55	0.67	2.05	0.76	2.87	**0.52**

Afin de bien illustrer la qualité de la généralisation de chaque architecture sur les données de validation et de test, l'histogramme de la figure 4.27 résume bien cet aspect.

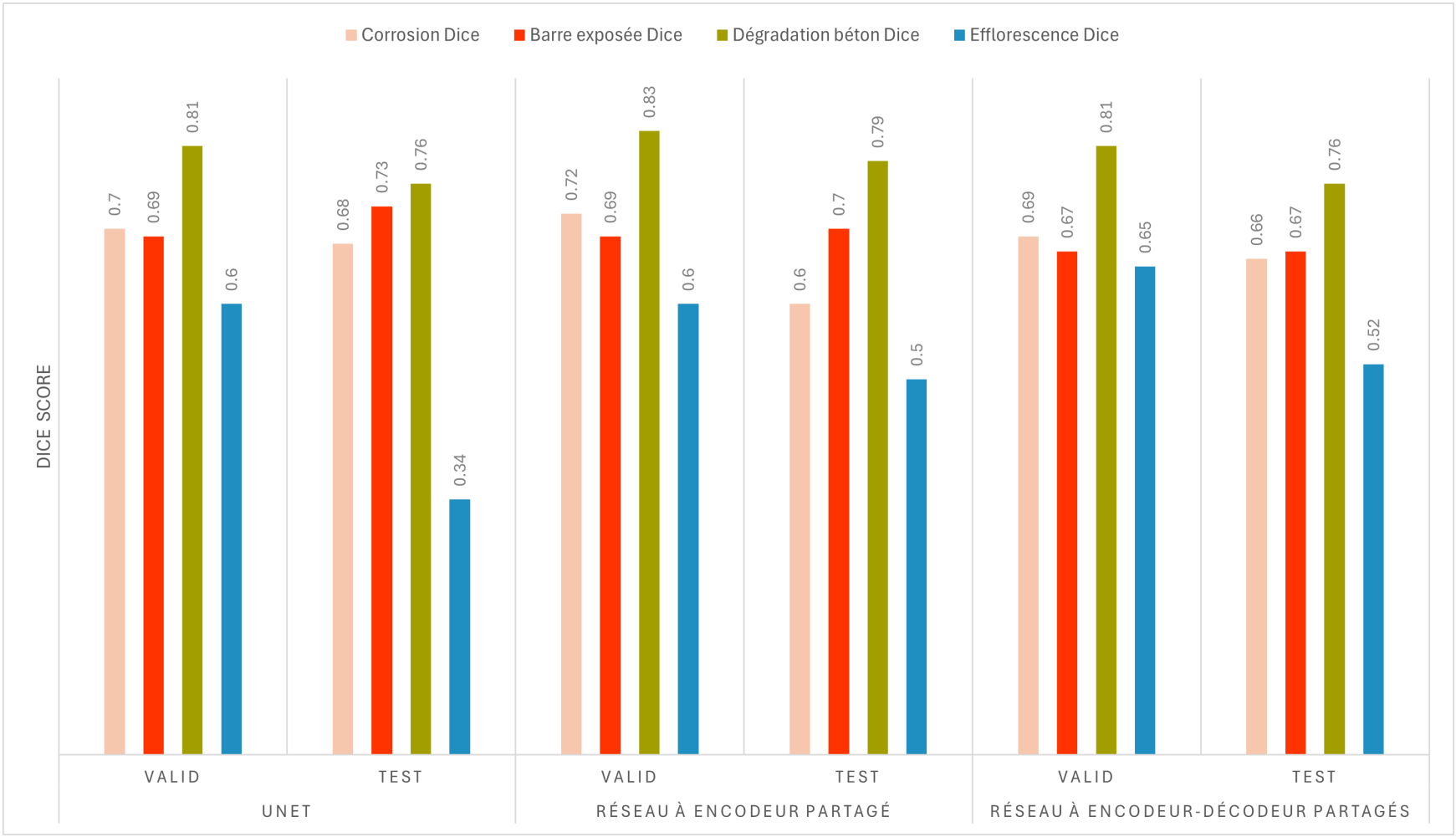


FIGURE 4.27 – Histogramme des résultats Dice de chaque architecture par classe de défauts sur les données de test et de validation

Résultats qualitatifs

Cette section montre les résultats d'inférences des modèles sur un ensemble d'images afin d'avoir une idée globale sur la qualité des masques générés.

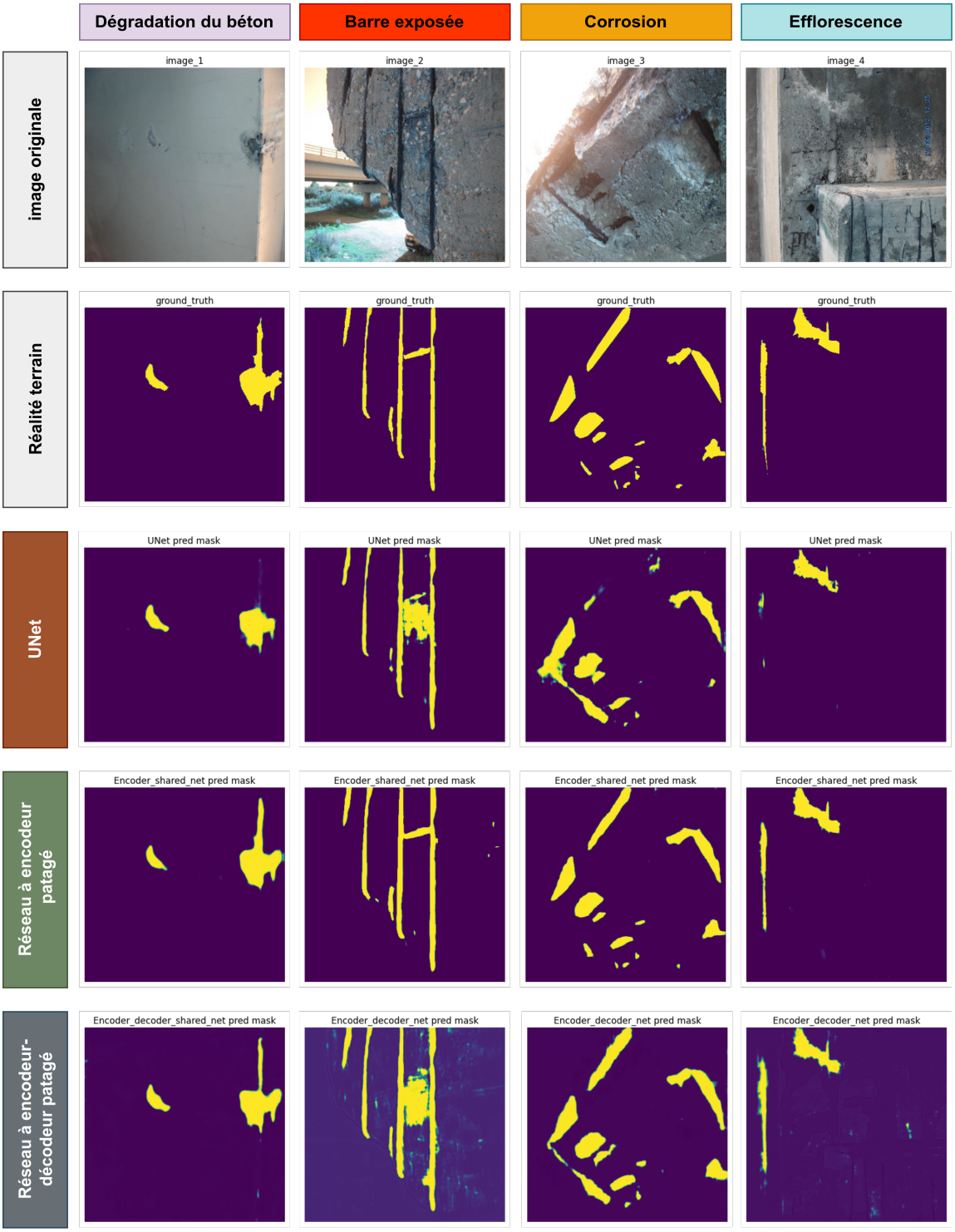


FIGURE 4.28 – Résultats des prédictions des 3 architectures sur 4 images.

Discussion

D'après les résultats des expérimentations menées sur nos trois architectures, l'ensemble des architectures atteignent le point de convergence lors du processus d'apprentissage. C'est un indicateur que les trois réseaux ont capturé l'ensemble des aspects caractéristiques et les relations présentes dans l'ensemble d'entraînement. Nous constatons aussi que pour les classes corrosion et barre exposée, les réseaux ont obtenu des résultats assez similaires dans un intervalle allant de 0.6 à 0.7. Concernant la classe de dégradation du béton, les modèles ont performé très bien à l'unanimité comme nous pouvons le voir dans la figure 4.27. Enfin, nous constatons que les réseaux n'ont pas su bien se généraliser par rapport à la classe efflorescence dans les données de test. Cela est probablement dû à la faible proportion d'images de cette classe présente dans l'ensemble d'entraînement relativement aux autres. Ainsi, les modèles n'ont pas assez appris pour bien se généraliser. En plus de ça, nous soupçonnons aussi le fait que les masques de cette classe ne capturent pas un ensemble suffisant d'aspects caractéristique de cette dernière pour une meilleure discrimination lors du processus d'apprentissage.

Chapitre 5

Conclusion Générale et perspectives

Ce travail de recherche est une partie qui s'inscrit dans un projet portant sur un système d'inspection de ponts avec drone. La problématique traitée dans ce travail concerne la mise en place d'une méthode d'apprentissage profond, à savoir la segmentation sémantique dans la détection des défauts pour ce processus d'inspection.

Pour répondre à ce problème, nous avons mis en place dans un premier temps une base de 594 images d'inspections annotées pour la segmentation de quatre défauts majeurs. Cette base d'image est une étape fondamentale pour la tâche de segmentation sémantique vu le manque de disponibilité de ce type de donnée. Ensuite, nous avons proposé trois architectures de réseaux de segmentation avec une approche orientée apprentissage multitâche. Ainsi, la segmentation de chaque classe de défaut sera considérée comme une tâche à part entière. Le premier réseau est un U-Net classique adapté pour une segmentation multi-classe. Le deuxième et le dernier se caractérisent par une architecture de type encodeur-décodeur. L'un se distingue par un encodeur partagé entre les tâches et l'autre avec les deux parties encodeur et décodeur qui sont partagées par les tâches. En plus, ces architectures intègrent des mécanismes d'apprentissage profond comme l'attention et la supervision en profondeur qui permettent d'améliorer les performances.

Les expérimentations réalisées sur les modèles proposés et l'ensemble d'images construit ont confirmé l'hypothèse de départ de cette recherche qui stipule que l'apprentissage multitâche peut s'avérer utile pour cette tâche de segmentation des défauts. Les résultats obtenus par les modèles montrent que ces derniers ont su bien se généraliser pour la plupart des défauts avec un score dice allant de 60% à 79% sur les données de test, à l'exception de la classe efflorescence.

Dans l'optique d'améliorer la solution proposée à travers ce travail, nous avons dressé un ensemble de pistes potentielles à explorer dans les futurs travaux :

Raffinage des annotations de l'ensemble de données : Cette opération est primordiale après les résultats obtenus dans la classe efflorescence. En tenant compte des analyses faites sur les instances de cette classe de défauts, nous estimons qu'il y a matière à améliorer les annotations de cette classe. Ainsi, nous obtiendrons des masques consistants qui capturent l'essentiel des aspects caractéristiques de cette classe.

Augmenter la base d'image : Cette approche est centrée sur les données. Dans la littérature relative à l'apprentissage profond, l'augmentation des données permet de pousser les performances des réseaux. Dans le cas de cette base d'image, il existe deux approches pour ce processus. La première est une approche naïve qui consiste à ajouter davantage d'images d'inspection de pont et ensuite les annoter. La deuxième consiste à générer une autre version de la base actuelle en divisant les images et leurs masques correspondants en plusieurs parties pour constituer une base plus large.

Repenser la conception architecturale des modèles proposés : Comme les modèles proposés adoptent l'apprentissage multitâche, cette piste consiste à modifier la structure de ces derniers afin de réduire le transfert d'apprentissage négatif entre les tâches. Dans le cas de cette solution, cette approche peut se concrétiser via la mise en place d'une ligne de décodage dédiée à la tâche de segmentation de l'efflorescence. Ainsi, la ligne pourra apprendre exclusivement des instances de la classe efflorescence et ne subira pas l'effet de la rétro-propagation engendrée par les données des autres défauts.

Mettre en place une fonction de perte dynamique : C'est une approche qui vise à mieux arbitrer le processus d'apprentissage multitâche du modèle. Cette fonction de perte dynamique permettra d'éviter le phénomène d'hégémonie d'une tâche sur les autres lors de l'apprentissage. Ce phénomène peut survenir lorsqu'il y a un déséquilibre entre les données dédiées pour chaque tâche. Ainsi, les paramètres du modèle auront plus tendance à être affectés par les données de la classe dominante.

Bibliographie

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation, 2015.

[3] 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, 2017.

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[5] Chenyu Zhang, Muhammad Monjurul Karim, and Ruwen Qin. A multitask deep learning model for parsing bridge elements and segmenting defect in bridge inspection images, 2022.

[6] Martin Mundt, Sagnik Majumder, Sreenivas Murali, Panagiotis Panetsos, and Visvanathan Ramesh. Meta-learning convolutional neural architectures for multi-target concrete defect classification with the concrete defect bridge image dataset. *CoRR*, abs/1904.08486, 2019.

[7] B. E. Moore and J. J. Corso. Fiftyone. *GitHub. Note : https ://github.com/voxel51/fiftyone*, 2020.

[8] Michael Crawshaw. Multi-task learning with deep neural networks : A survey, 2020.

[9] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 94–108, Cham, 2014. Springer International Publishing.

[10] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. Nddr-cnn : Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3200–3209, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society.

[11] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets, 2014.

[12] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM : Convolutional Block Attention Module, July 2018. arXiv :1807.06521 [cs].

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[15] American Association of State Highway and Transportation Officials (AASHTO). Manual for Bridge Element Inspection, 2019.

[16] Dariush Amirkhani, Mohand Saïd Allili, Loucif Hebbache, Nadir Hammouche, and Jean-François Lapointe. Visual concrete bridge defect classification and detection using deep learning : A systematic review. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–23, 2024.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[19] Mingxing Tan and Quoc V. Le. Efficientnet : Rethinking model scaling for convolutional neural networks, 2020.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision – ECCV 2014*, pages 346–361. Springer International Publishing, 2014.

[21] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[22] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net : Fully convolutional neural networks for volumetric medical image segmentation, 2016.

[23] Dmitry A. Yudin, Vasily Adeshkin, Alexandr V. Dolzhenko, Alexandr Polyakov, and Andrey E. Naumov. Roof Defect Segmentation on Aerial Images Using Neural Networks. In Boris Kryzhanovsky, Witali Dunin-Barkowski, Vladimir Redko, and Yury Tiumentsev, editors, *Advances in Neural Computation, Machine Learning, and Cognitive Research IV*, pages 175–183, Cham, 2021. Springer International Publishing.

[24] Fityanul Akhyar, Chih-Yang Lin, and Gugan S. Kathiresan. A Beneficial Dual Transformation Approach for Deep Learning Networks Used in Steel Surface Defect Detection. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*, ICMR '21, pages 619–622, New York, NY, USA, September 2021. Association for Computing Machinery.

[25] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition, 2020.

[26] Juan Jose Rubio, Takahiro Kashiwa, Teera Laiteerapong, Wenlong Deng, Kohei Nagai, Sergio Escalera, Kotaro Nakayama, Yutaka Matsuo, and Helmut Pren-

dinger. Multi-class structural damage segmentation using fully convolutional networks. *Computers in Industry*, 112 :103121, November 2019.

[27] Cao Vu Dung and Le Duc Anh. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction*, 99 :52–58, March 2019.

[28] Gang Pan, Yaoxian Zheng, Shuai Guo, and Yaozhi Lv. Automatic sewer pipe defect semantic segmentation based on improved U-Net. *Automation in Construction*, 119 :103383, November 2020.

[29] Mingzhu Wang and Jack C. P. Cheng. A unified convolutional neural network integrated with conditional random field for pipe defect segmentation. *Computer-Aided Civil and Infrastructure Engineering*, 35(2) :162–177, 2020.

[30] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network, April 2017. arXiv :1612.01105 [cs].

[31] Jiyuan Shi, Ji Dang, Mida Cui, Rongzhi Zuo, Kazuhiro Shimizu, Akira Tsunoda, and Yasuhiro Suzuki. Improvement of Damage Segmentation Based on Pixel-Level Data Balance Using VGG-Unet. *Applied Sciences*, 11(2) :518, January 2021.

[32] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN, January 2018. arXiv :1703.06870 [cs].

[33] Yonas Zewdu Ayele, Mostafa Aliyari, David Griffiths, and Enrique Lopez Droguett. Automatic Crack Segmentation for UAV-Assisted Bridge Inspection. *Energies*, 13(23) :6250, January 2020. Number : 23 Publisher : Multidisciplinary Digital Publishing Institute.

[34] Jiajun Li, Qian Wang, Jun Ma, and Jingjing Guo. Multi-defect segmentation from façade images using balanced copy–paste method. *Computer-Aided Civil and Infrastructure Engineering*, 37(11) :1434–1449, 2022.

[35] Chul Min Yeum, Jongseong Choi, and Shirley J Dyke. Automated region-of-interest localization and classification for vision-based visual assessment of civil infrastructure. *Structural Health Monitoring*, 18(3) :675–689, May 2019. Publisher : SAGE Publications.

[36] Yasutaka Narazaki, Vedhus Hoskere, Tu A. Hoang, Yozo Fujino, Akito Sakurai, and Billie F. Spencer Jr. Vision-based automated bridge component recognition with high-level scene consistency. *Computer-Aided Civil and Infrastructure Engineering*, 35(5) :465–482, 2020.

[37] Eric Bianchi, Amos Lynn Abbott, Pratap Tokekar, and Matthew Hebdon. COCO-Bridge : Structural Detail Data Set for Bridge Inspections. *Journal of Computing in Civil Engineering*, 35(3) :04021003, May 2021. Publisher : American Society of Civil Engineers.

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD : Single Shot MultiBox Detector. volume 9905, pages 21–37. 2016. arXiv :1512.02325 [cs].

[39] Muhammad Monjurul Karim, Ruwen Qin, Genda Chen, and Zhaozheng Yin. A semi-supervised self-training method to develop assistive intelligence for seg-

menting multiclass bridge elements from inspection videos. *Structural Health Monitoring*, 21(3) :835–852, May 2022. Publisher : SAGE Publications.

[40] Chenyu Zhang, Muhammad Monjurul Karim, Zhaozheng Yin, and Ruwen Qin. A deep neural network for multiclass bridge element parsing in inspection image analysis, 2022.

[41] Weilei Yu and Mayuko Nishio. Multilevel structural components detection and segmentation toward computer vision-based bridge inspection. *Sensors*, 22(9), 2022.

[42] Rich Caruana. Multitask Learning. *Machine Learning*, 28(1) :41–75, July 1997.

[43] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[44] Leland McInnes, John Healy, and James Melville. Umap : Uniform manifold approximation and projection for dimension reduction, 2020.

[45] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86) :2579–2605, 2008.

[46] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2018.

[47] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R. Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention Mechanisms in Computer Vision : A Survey. *Computational Visual Media*, 8(3) :331–368, September 2022. arXiv :2111.07624 [cs].

[48] Pavel Iakubovskii. Segmentation models pytorch. https://github.com/qubvel/segmentation_models.pytorch, 2019.

[49] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[50] François Chollet. Xception : Deep learning with depthwise separable convolutions, 2017.

[51] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets : Efficient convolutional neural networks for mobile vision applications, 2017.

[52] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer : Simple and efficient design for semantic segmentation with transformers, 2021.

[53] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization, 2017.

[54] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61) :2121–2159, 2011.