**UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS**

# DÉTECTION ET SEGMENTATION DE DÉFAUTS POUR LES INFRASTRUCTURES DE TRANSPORT EN UTILISANT L'IMAGERIE PAR DRONES ET L'APPRENTISSAGE PROFOND

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN SCIENCES ET TECHNOLOGIES DE L'INFORMATION

PAR

**LOUCIF HEBBACHE**

FÉVRIER 2023

Ce mémoire a été évalué par un jury composé des personnes suivantes :

Pr. Larbi Talbi (UQO) ..... Président du jury

Pre. Ana-Maria Cretu (UQO) ..... Membre du jury

Pr. Mohand Said Allili (UQO) ..... Directeur de recherche

Dr. Jean-François Lapointe (CNRC) ..... Codirecteur de recherche

Mémoire accepté le : 03-03-2023

## ***Remerciements***

*Ce mémoire est le fruit de longues heures de recherche et de travail. Tout d'abord, nous tenons à remercier ALLAH Tout-Puissant de nous avoir aidé à accomplir ce travail.*

*J'exprime mes plus sincères remerciements à mon directeur de thèse, le prof. Mohand Said Allili, pour son encadrement et sa confiance accordée. Sa présence et ses remarques furent d'une aide précieuse. Je remercie aussi chaleureusement mon codirecteur de recherche M. Jean-François Lapointe pour son encouragement et ses apports enrichissants.*

*Merci aux membres du jury de l'Université du Québec en Outaouais pour leur participation à l'évaluation de ce mémoire.*

# Table des matières

# Liste des figures

# Liste des tableaux

# Liste des abréviations, sigles et acronymes

**AI** Artificial Intelligence

**AR** Augmented Reality

**AUC** Area Under the ROC Curve

**DL** Deep Learning

**DETR** DEtection TRansformer

**FCN** Fully Convolutional Network

**GPS** Global Positioning System

**Grad-CAM** Gradient-weighted Class Activation Map

**LIDAR** Light Detection And Ranging

**mAP** Mean Average Precision

**NLP** Natural Language Processing

**R-CNN** Region-Based Convolutional Neural Network

**ROC** Receiver Operating Characteristic

**SSD** Single Shot MultiBox Detector

**YOLO** You Only Look Once

**UAV** Unmanned Aerial Vehicle

**ViT** Vision Transformer

# Résumé

Les infrastructures de transport telles que les ponts, atteignent un jour ou l'autre la fin de leur durée de vie en raison du vieillissement, de l'utilisation accrue et de l'impact climatique défavorable. Une inspection régulière est indispensable pour maintenir la sécurité du public qui les utilise ainsi que l'intégrité de ces structures. Jusqu'à récemment, ces inspections sont entièrement manuelles et se basent sur l'inspection visuelle pour détecter les défauts. Avec le développement de l'apprentissage profond, l'inspection à l'aide de drones équipés de caméras, a été envisagée comme une approche alternative prometteuse qui permet un accès plus efficace aux structures avec moins de risques.

Dans ce mémoire, nous proposons une méthodologie permettant de classer, détecter, et éventuellement segmenter, les défauts sur les images acquises par drone, en utilisant l'apprentissage profond pour les infrastructures de transport, et en particulier les ponts. En premier lieu, une étude comparative des algorithmes de classification a été faite pour sélectionner les architectures les plus appropriées, en particulier ResNet, Inception, EfficientNet et leurs variantes. Une nouvelle méthode combinant la détection et la saillance visuelle, baptisée SMDD-NET, a été proposée comme notre principale contribution dans le cadre de la détection de défaut, et qui a donné des résultats très prometteurs. Par la suite, des architectures telles que U-Net, FCN et leurs variantes ont été explorées pour la segmentation de défauts .

**Mots clés** : *Pont · Inspection · AI . Intelligence Artificielle · DL . Apprentissage Profond · Détection de défauts . Drone.*

# Abstract

Transportation infrastructures, such as bridges, are, one day or another, reaching the end of their useful lifespan due to ageing, increased use and adverse climate impact. Periodic inspections are then essential to maintain the safety of the public who use them as well as the integrity of theses structures. Until recently, these inspections have been done entirely manually using primarily visual inspection to detect defects on the structure. With the development of deep learning, inspection using drones equipped with cameras has been considered as a promising alternative approach that allows more efficient access to structures with less risk.

In this thesis, we propose a methodology for classifying, detecting, and possibly segmenting defects in images acquired by drones, using deep learning for transportation infrastructure, particularly bridges. Firstly, a comparative study of classification algorithms was done to select the most suitable architectures, particularly ResNet, Inception, EfficientNet, and their variants. A new method combining detection and visual saliency, called SMDD-NET, was proposed as our main contribution in the context of defect detection, which gave very promising results. Subsequently, architectures such as U-Net, FCN, and their variants were explored for defect segmentation.

**Keywords** : *Bridge · Inspection · AI . Artificial Intelligence · DL . Deep Learning · Defect Detection . Drone.*

# Chapitre 1

# Introduction Générale

## 1.1 Mise en contexte et problématique

Les ponts sont des infrastructures critiques des réseaux routier et ferroviaire. Lorsqu'un pont subit une dégradation de l'un de ses éléments composants, cela compromet la sécurité du public et réduit l'efficacité du réseau routier. Les récentes tragédies causées par les effondrements des ponts I-35W et Morandi [60, 12] illustrent l'importance de mener des inspections minutieuses et régulières sur ce type de structure. Le TPSGC (Travaux publics et Services gouvernementaux Canada) a mis en place une politique globale sur l'inspection et l'évaluation des ponts appartenant au gouvernement fédéral, qui comprend des dispositions voulant que tous ces ponts soient inspectés par des personnes qualifiées à intervalles réguliers, tel que décrit dans le Manuel d'Inspection des Ponts ( MIP)[20]. Ces inspections ont été menées jusqu'à présent par des inspecteurs, en procédant soit visuellement (l'oeil humain), ou bien manuellement (par contact) via un outil comme un marteau. Toutefois, ceci est très coûteux en terme de temps, d'efforts et d'argent, sans oublier le danger lié à l'inspection d'endroits qui sont difficilement accessibles.

Ces dernières années, les véhicules aériens sans pilote (UAV), aussi connus sous le nom de drones ou systèmes d'aéronefs télépilotés (SATP), sont devenus l'une des technologies les plus prometteuses dans les applications civiles. De nos jours, grâce aux avancées technologiques de la conception des drones, leur utilisation a considérablement augmenté en permettant l'accès à des endroits éloignés sans l'intervention directe d'un opérateur humain à bord. Les drones sont donc utilisés dans toute une gamme d'applications,

telles que la télédétection, l'assistance dans les situations d'urgence, l'inspection des infrastructures, les systèmes logistiques, la photographie professionnelle, les systèmes de pulvérisation pour l'agriculture de précision, entre autres, et bien d'autres. Pour améliorer l'efficacité du guidage des drones dans ces applications, ils sont généralement équipés de systèmes de positionnement par satellites ou GNSS (Géolocalisation et Navigation par Système de Satellites globaux (GPS, Galileo, etc.)) et de capteurs extéroceptifs comme le LIDAR, des caméras qui leur permettent d'explorer et de cartographier leur environnement.

Avec le développement récent de l'apprentissage profond (DL), il est devenu possible d'exploiter la puissance des algorithmes d'intelligence artificielle (AI) pour utiliser les drones à des fins de navigation ou d'inspection. La navigation consiste à doter les drones de la capacité de comprendre l'environnement visuel et d'améliorer leur autonomie, leur efficacité et leur sécurité. Tandis que l'inspection consiste à capturer et enregistrer l'état actuel de la structure sous forme de vidéos et d'images qui seront analysées en temps réel ou différé pour déterminer les défauts en utilisant les techniques de vision par ordinateur : la classification, la localisation et la segmentation d'objet. Avec la réalité augmentée (AR) de l'autre côté, cela permettra un contrôle humain à distance interactif des drones en générant un contenu AR aligné avec l'environnement capturé. Le contenu AR permettra à un expert ou à une équipe de techniciens de contrôler à distance et dynamiquement des drones pour effectuer des actions spécifiques ou se comporter d'une manière spécifique.

Pour la mission d'inspection des infrastructures en utilisant le drone, les algorithmes d'apprentissage profond sont devenus l'état de l'art des méthodes d'inspection en raison de leur approche d'apprentissage de bout en bout sans avoir recours à l'ingénierie des caractéristiques (*Feature Engineering*). Parmi ces algorithmes, les réseaux de neurones convolutifs (CNN) sont les plus utilisés pour la classification des images dans la vision par ordinateur grâce au succès des architectures profondes proposées dans la dernière décennie telles que VGGNet [74], GoogleNet [79], ResNet [30], DenseNet [34] et InceptionV4 [78] pour n'en nommer que quelques unes. Les CNNs sont construits avec la capacité d'extraire des caractéristiques uniques à partir de l'image, qui décrivent une représentation significative des objets, ce qui va aider à leur classification. Inspiré de ces avancées, la classification et la détection automatiques des défauts en béton sont

devenues un domaine actif de recherche au cours des dernières années [48], [58], [77]. Les premières méthodes se concentraient principalement sur la classification de défaut, grâce à la disponibilité de réseaux de neurones profonds pré-entraînés et à l'utilisation de l'apprentissage par transfert pour classifier divers défauts dans les images [56], [72]. Récemment, la détection et la segmentation de défauts en béton ont gagné un intérêt croissant dans la communauté grâce à la disponibilité des jeux de données d'entraînement, et à la performance croissante des méthodes de détection d'objet (par exemple, SSD, Fast-RCNN, YOLO) [21], [89].

Cependant, plusieurs défis existent pour atteindre une classification/détection précise et automatisée des défauts dans les structures en béton :

– **Non-homogénéité des surfaces en béton :** en fonction des matières premières utilisées, des finitions et de l'exposition aux éléments de la nature, il peut exister une grande variété de couleurs et de textures des surfaces en béton (figure 1.1). De plus, les surfaces en béton peuvent contenir des peintures, des graffitis et des pièces métalliques qui augmentent la variabilité de la classe arrière-plan (background).

FIGURE 1.1 – Exemple de textures des surfaces en béton [3].

– **Variabilité des classes de défauts en béton :** les défauts peuvent avoir différentes tailles et intensités, et peuvent aussi se situer dans des différents emplacements dans l'image, ce qui augmente la difficulté de leur détection. Le même défaut peut être constitué de différentes parties et différents types de défauts peuvent se chevaucher au même endroit (figure 1.2). En fait, comme ilsera présenté dans le chapitre d'état de l'art, certaines classes de défauts sont fortement corrélées, ce

qui peut justifier leur co-occurrence. Par exemple, la détection d'une oxydation de métaux peut être souvent un indice de la présence de corrosion, de délaminage ou d'écaillage. On appelle un tel phénomène où plusieurs classes de défauts peuvent se chevaucher, un problème de classification de défauts multi-classe multi-étiquette (*multi-label*).

(a) (b) (c)

FIGURE 1.2 – Un exemple de défauts qui se chevauchent au même endroit : (a) Écaillage ; (b) Corrosion ; (c) Barres exposées [58].

- **Conditions d'acquisition des images de défauts :** en fonction de sa taille et de sa localisation dans l'infrastructure, un défaut peut être capturé entièrement ou partiellement dans une image. De plus, le changement des conditions météorologiques, éclairage, occlusion et méthodologies/dispositifs d'acquisition d'images peut augmenter la variabilité des classes de défaut. Toutes ces conditions peuvent affecter la précision de la classification/détection du modèle (figure 1.3).
- **Annotation d'image et génération de jeu de données :** la précision de l'annotation peut être affectée par plusieurs facteurs. Par exemple, lorsqu'un défaut est discontinu, il peut être considéré comme multiple instances de la même classe de défauts (figure 1.4). De même, la même image peut contenir différentes classes de défaut au même endroit. Cela peut induire une certaine subjectivité dans le processus d'annotation où certains défauts pourraient être ignorés ou mal étiquetés, surtout lors de la création de masques de segmentation.

Pour relever certains de ces défis, plusieurs travaux de recherche menés au cours des dernières années ont conduit au développement de plusieurs jeux de données et modèles d'apprentissage profond pour la détection de défauts en béton. Alors qu'il existe plusieurs jeux de données dédiés à des défauts spécifiques, seuls quelques-uns traitent plusieurs défauts à la fois. Par conséquent, la majorité des modèles proposés dans la littérature

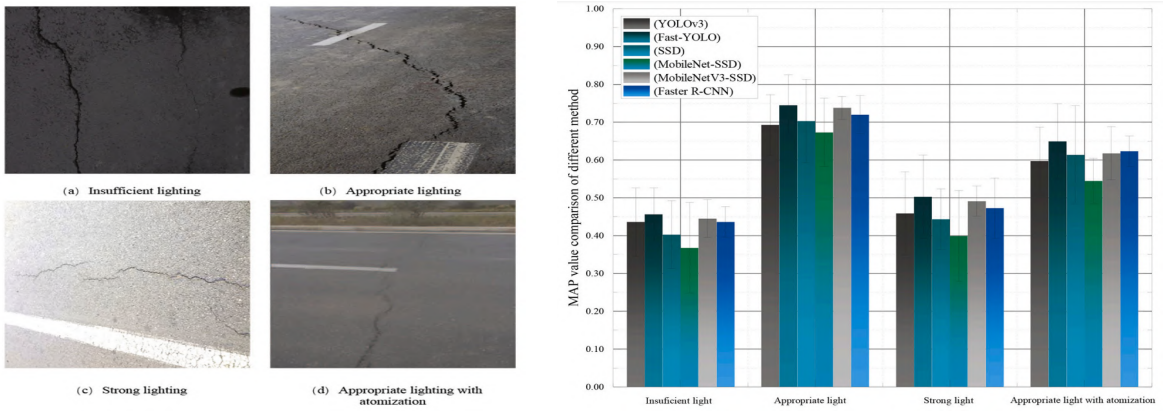


FIGURE 1.3 – L'effet de variation des conditions d'éclairage sur la précision de détection des modèles : les défauts sont mieux détectés dans des conditions d'éclairage appropriées que dans les trois autres cas [40].

ont été conçus pour la détection d'une seule classe de défaut. De plus, l'hétérogénéité entre les jeux de données peut amener un modèle entraîné sur un jeu de données à mal performer sur d'autres jeux de données. Il y a donc beaucoup de recherche à faire pour améliorer la détection/classification de défauts de telle sorte que : a) des modèles uniques peuvent traiter plusieurs défauts à la fois et b) les modèles peuvent se généraliser à différents jeux de données.

A cet égard, le présent travail s'inscrit dans le cadre plus large d'un projet de développement d'un système d'inspection pour les infrastructures de transport basé sur l'intelligence artificielle (AI) et la réalité augmentée (AR). Il est constitué donc de deux modules : *AI module* et *AR module* tel qu'illustré à la figure 1.5. La partie qui nous a été confiée alors dans ce projet est celle de développement du module AI, qui consiste à proposer une approche permettant d'automatiser la classification, la détection et la segmentation de défauts multi-classe multi-étiquette sur des images acquises par drone pour les infrastructures de transport, en utilisant les algorithmes d'apprentissage profond (figure 1.6) . Nous nous concentrons ici sur les structures de ponts en béton, et nous détaillerons notre méthodologie dans les chapitres à suivre.

FIGURE 1.4 – Exemple d'un défaut discontinu (barres exposées) qui pose un problème d'annotation où on peut considérer chaque barre comme une instance d'objet, ou bien toutes les barres constituent une seule et même instance d'objet [58].

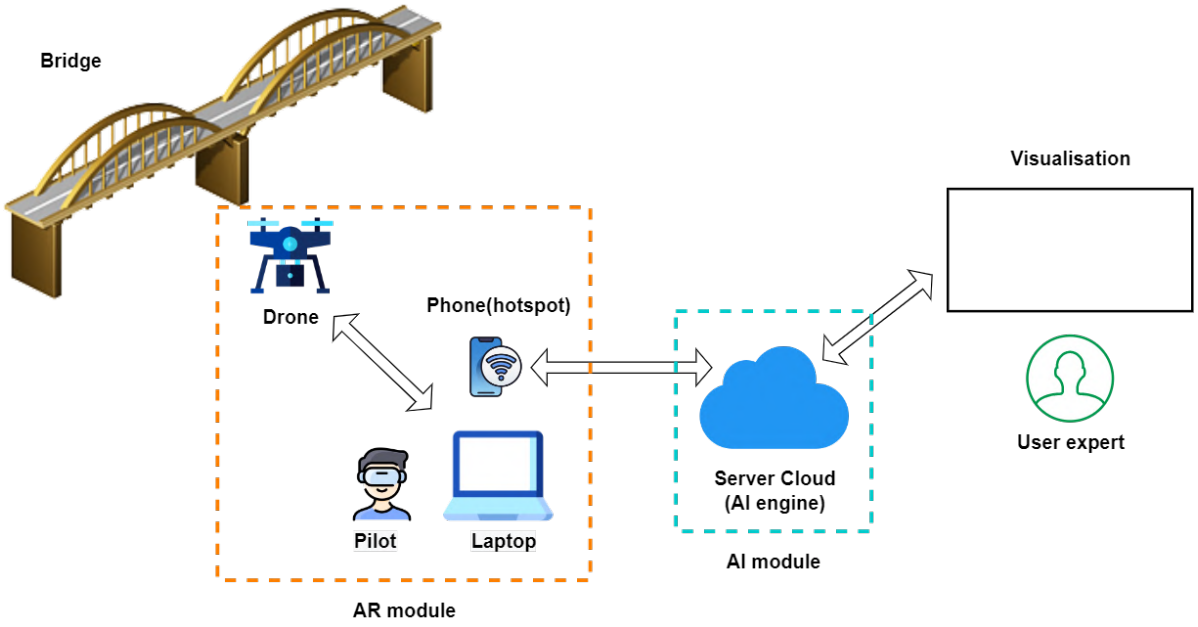


FIGURE 1.5 – Vue globale du projet dans lequel s'inscrit ce mémoire [49].

## 1.2 Objectifs

Le module *AI* à développer dans ce travail va aider dans la partie d'inspection (et pas dans la navigation), et pour cela nous proposons un pipeline dont le but principal est la localisation de défauts au sens de la détection et de la segmentation, en se basant sur la classification tel qu'il est montré dans la figure 1.6. Les étapes qui nous permettent d'arriver à l'objectif de localiser les défauts sur les images couleurs captées par le drone sont :

- **La classification :** étudier l'état de l'art des classificateurs existants (ResNet, Inception, EfficientNet...etc) pour sélectionner les plus performants qui permettent de classifier les défauts en béton selon leurs types, que ce soit en multi-classe (un objet est étiqueté avec une seule classe) ou en multi-étiquette (un objet peut être étiqueté avec plusieurs classes) ;
- **La détection :** localiser grossièrement les défauts (au niveau *bounding boxes*) en développant une méthode basée sur les modèles de détection d'objet existants (YOLO, RetinaNet, Faster R-CNN...etc) et les techniques de saillance visuelle (Grad-CAM, Full-Grad, SmoothGrad...etc), tout dans un seul modèle *End-to-End*, permettant ainsi de rehausser les caractéristiques des défauts afin que ces modèles donnent de meilleurs résultats possibles car les défauts ne sont pas nécessairement détectables à cause des challenges expliqués auparavant ;
- **La segmentation :** localiser localement les défauts (au niveau pixels) en utilisant les modèles de segmentation sémantique existants (FCN, U-Net, DeepLab...etc), afin d'étudier plus tard les caractéristiques des défauts (largeur, profondeur) pour leur affecter un niveau de gravité (objectif à long terme).

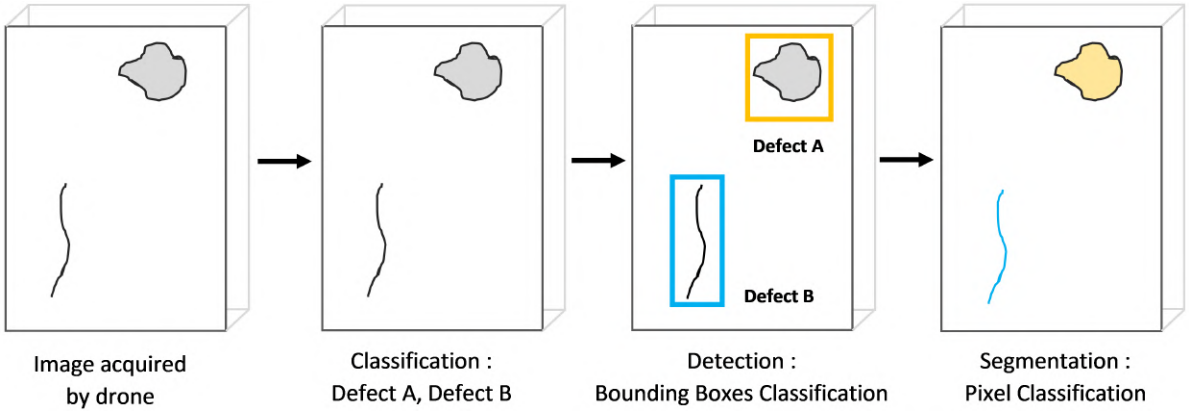


FIGURE 1.6 – Illustration détaillée du module AI qui nous a été confié.

## 1.3 Organisation du mémoire

Ce mémoire est une synthèse globale de notre travail de recherche effectué en vue de réaliser les objectifs cités ainsi que répondre aux questions liées à la problématique. Il est organisé ainsi : le deuxième chapitre présente l'état de l'art où nous présenterons en premier lieu quelques généralités sur les types de défauts des infrastructures de transport (ponts) ; ensuite on décrira le vocabulaire technique nécessaire pour appréhender notre approche proposée pour résoudre la problématique ; on finira le chapitre en passant en revue la littérature des méthodes d'inspection des ponts. Dans le chapitre 3, nous allons détailler les trois éléments qui constituent l'approche proposée, à savoir : la classification, la détection et la segmentation de défaut. Le Chapitre 4 décrit les essais effectués et les résultats obtenus sous forme d'une synthèse comparative. Enfin, le chapitre 5 conclut le mémoire en soulignant les limites de notre approche et en proposant des perspectives pour une future continuité du projet.

# Chapitre 2

# État de l'art

L'inspection manuelle des infrastructures de transport est une tâche complexe qui nécessite beaucoup d'effort, de temps et d'argent. Pour remédier à ce problème, plusieurs recherches ont été menées ces derniers temps sur l'utilisation de drones équipés d'une vision artificielle pour effectuer de nombreuses tâches telle que l'inspection des ponts en béton afin de détecter d'éventuels défauts qui peuvent mener à une catastrophe. Dans le présent chapitre, nous allons d'abord aborder quelques généralités sur les défauts qui peuvent exister dans les ponts et la procédure utilisée pour les inspecter en se basant sur le Manuel d'Inspection des Ponts (MIP)[20] et le Manuel d'Inspection des Structures[83]. Toutes les informations que nous donnerons alors dans la première section de ce chapitre sont issues de ces deux manuels. Ensuite, nous donnerons les concepts techniques liés à la problématique afin d'appréhender l'approche proposée. Enfin, nous allons passer en revue la littérature des méthodes utilisées pour l'inspection des ponts, en particulier celles basées sur l'imagerie par drones et l'apprentissage profond.

## 2.1 Manuel d'Inspection des Ponts (MIP)

Le Manuel d'inspection des ponts (MIP) [20] est un document contenant une politique globale sur l'inspection et l'évaluation des ponts, mise en place afin d'assurer une inspection standard et cohérente de tous les ponts appartenant à TPSGC (Travaux Publics et Services Gouvernementaux Canada), notamment certains ponts inter-provinciaux. Il définit un programme d'inspection que les inspecteurs suivent afin de réaliser des inspections rigoureuses à intervalles réguliers. Ces inspections peuvent être alors de type :

- **Inspection générale :** qui a pour objectifs de détecter tous les défauts qui affectent les éléments de la structure (pont) et d'évaluer leurs niveaux de gravité en utilisant les système d'évaluation des structures définit par l'état, afin d'estimer la durée de vie de la structure.
- **Inspection annuelle :** consiste à effectuer une fois par an (sauf l'année où l'inspection générale est réalisée), un examen visuel des éléments de la structure afin de repérer les défauts qui peuvent causer des accidents de circulation ou affecter la stabilité de la structure.
- **Inspection spéciale :** consiste à examiner profondément les éléments complexes d'une structure afin de trouver les défauts et de préciser leur incidence sur la stabilité de ces éléments. Elle est effectuée généralement lorsque des défauts importants ont été observés sur les éléments principaux d'une structure lors de l'inspection générale.
- **Inspection d'évaluation :** elle consiste à passer un examen profond des éléments principaux d'une structure pour détecter les défauts et de spécifier leur incidence sur la capacité portante de ces éléments par rapport à l'ensemble de la structure.
- **Inspection d'observation :** dans ce type d'inspection, les inspecteurs examinent des éléments spécifiques de la structure qui ont été déjà prédéfinis après avoir dressé un portrait de la structure pour détecter les éléments fragiles.

Les inspections générales et annuelles appartiennent à la catégorie des inspections courantes qui doivent être effectuées à une fréquence préétablie. Quant aux inspections spéciales, d'évaluation, et d'observation, elles appartiennent à la catégorie des inspections particulières qui sont effectuées au fur et à mesure des besoins.

Pour procéder à l'inspection, les inspecteurs suivent les étapes suivantes :

1. **Préparation de l'inspection :** consiste à consulter les plans de la structure, préparer les fiches d'inspection nécessaires, prévoir l'équipement courant d'inspection, etc.
2. **La validation des données d'inventaire :** vérifier que les données d'inventaire contenues dans le système de gestion des structures sont conformes à la réalité. Les inspecteurs doivent alors valider des données comme : le type de la structure, le nom de la route, l'année de sa construction, les dimensions de ses éléments...etc.
3. **Le déroulement de l'inspection :** l'inspection se déroule d'une façon méthodique de telle sorte que tous les éléments de la structure soient correctement ins-

pectés. Pour cela, il est suggéré de procéder en commençant par le dessous de la structure, lorsqu'il s'agit d'un pont, pour finir par le dessus du tablier.

4. **La finalisation de l'inspection :** là où on génère les rapports d'inspection et on procède à la fermeture de l'inspection.

Lors de l'inspection d'un pont, différents défauts peuvent être observés selon la nature du matériau qui constitue le pont qui peut être : le béton, la maçonnerie, l'acier, l'aluminium, le bois, l'enrobé. Dans notre cas, on considére les défauts de nature béton qui est principalement combiné à de l'acier d'armature pour former du béton armé. Les principaux défauts du béton dans ce cas sont décrits ci-dessous. La figure 2.1 illustre chacun d'eux et le tableau 2.1 montre le degré de sévérité de chaque type de défaut, allant de **léger (A)**, **moyen (B)**, **important (C)** à **très important (D)**.

- **La désagrégation :** est la détérioration physique du béton en de petites particules.
- **La corrosion de l'armature :** causé par l'oxydation d'une ou plusieurs barres d'armature en acier. Des taches de rouille à la surface du béton sont les premiers signes de corrosion de l'armature.
- **Le délaminage :** Le délaminage est une fissuration interne entre la couche de surface et la couche en profondeur, sans qu'il n'y ait détachement.
- **L'éclatement :** est le détachement de fragments de béton causé par la pression due à la corrosion de l'armature ou à la formation de glace dans les parties délaminées.
- **La fissuration :** est une fracture linéaire qui traverse partiellement ou complètement le béton. La détection de fissure est l'un des sujets les plus actifs en vision par ordinateur !
- **L'efflorescence :** se forme lorsque de l'eau migre à l'intérieur du béton en formant un dépôt de sel, généralement blanc.

À noter que certains défauts peuvent se chevaucher, par exemple la corrosion avec des barres d'armature exposées, sont liés à l'éclatement (logiquement, lorsque une barre d'armature est exposée, cela veut dire qu'il y a éclatement du béton). Tout comme l'efflorescence qui est liée aussi à la fissuration. Ceci nous amène à introduire le concept de **détection de défauts multi-étiquette : dans une image donnée, un objet**


FIGURE 2.1 – Les principaux défauts des ponts en béton [58].

TABLE 2.1 – Degré de gravité de chaque type de défaut.

| **Les défauts** | **État du matériau** | | | |
|---|---|---|---|---|
| | **Léger** | **Moyen** | **Important** | **Très important** |
| **Désagrégation.** | jusqu'à 25 mm | de 25 à 50 mm | de 50 à 100 mm | plus de 100 mm |
| **Corrosion.** | aucune | armature apparente | jusqu'à 30 % | plus de 30 % |
| **Délaminage.** | – | – | présence | – |
| **Éclatement.** | – | – | présence | – |
| **Fissure.** | aucune | inférieure à 0,8 mm | de 0,8 à 3 mm | supérieure à 3 mm |
| **Efflorescence.** | – | – | présence | – |

**(défaut) peut appartenir à plusieurs classes**. Nous détaillerons ce concept dans la prochaine section.

Pour évaluer l'état général des éléments d'une structure, les inspecteurs (experts) utilisent le système d'évaluation de l'état des éléments qui repose sur deux indicateurs : le premier concerne l'état du matériau constituant l'élément ; le deuxième concerne l'aptitude de l'élément à jouer son rôle dans la structure : l'évaluation de comportement. L'évaluation de l'état du matériau d'un élément consiste à déterminer le niveau de dégradation le caractérisant (A, B, C, D), à partir de quatre degrés de sévérité allant de léger, moyen, important à très important. Cette dernière se base principalement sur une observation visuelle d'un expert, mais elle peut être validée dans certains cas manuellement (p. ex. via un marteau). Il est à noter que lorsqu'un élément est affecté par plusieurs défauts dans la même zone, le pire défaut doit être retenu.

L'évaluation du comportement d'un élément consiste à apprécier, à l'aide d'une cote représentative, l'aptitude de l'élément à jouer son rôle dans la structure. La cote d'éva-

luation du comportement d'un élément est désignée « cote CEC ». Elle varie de 1 à 4 tel qu'il est montré dans la figure 2.2.

| Comportement | Pourcentage de diminution de l'aptitude d'un élément à jouer son rôle | |
|---|---|---|
| Cote | Élément principal | Élément secondaire |
| 4 | 0 à 10 % | 0 à 10 % |
| 3 | 10 à 20 % | 10 à 30 % |
| 2 | 20 à 30 % | 30 à 50 % |
| 1 | > 30 % | > 50 % |

FIGURE 2.2 – Évaluation du comportement des éléments d'une structure [83].

## 2.2 Vision artificielle et apprentissage profond : Généralités

Dans cette section, nous allons donner des généralités sur les concepts techniques relatifs à notre sujet de recherche.

### 2.2.1 Vision artificielle

Dans le domaine de la vision artificielle (appelée aussi vision par ordinateur), il existe plusieurs problèmes fondamentaux de la reconnaissance visuelle. Par la définition donnée en [69], on définit alors :

- **Classification d'objet :** algorithmes qui produisent une liste des catégories d'objets présentes dans l'image.
- **Détection d'objet :** algorithmes qui produisent une liste des catégories d'objets présentes dans l'image ainsi qu'une boîte englobante (en anglais *bounding box*) indiquant la position de chaque instance de chaque catégorie d'objets.
- **Segmentation d'objet :** identifier les parties de l'image et comprendre à quelles catégories d'objets elles appartiennent. Il existe deux types de segmentation :

**Segmentation sémantique :** la segmentation sémantique est une classification mais au niveau pixel. Elle associe chaque pixel d'une image à une étiquette (label) de classe telle qu'une personne, une fleur, une voiture, etc. Elle traite plusieurs objets de la même classe comme une seule entité ;

**Segmentation d'instance :** en revanche, la segmentation d'instance traite plusieurs objets de la même classe comme des instances individuelles distinctes. On décrit souvent la segmentation d'instance comme étant l'intersection de la détection d'objet avec la segmentation sémantique.

La figure 2.3 illustre la différence entre les trois tâches de la vision par ordinateur : classification, détection et segmentation.

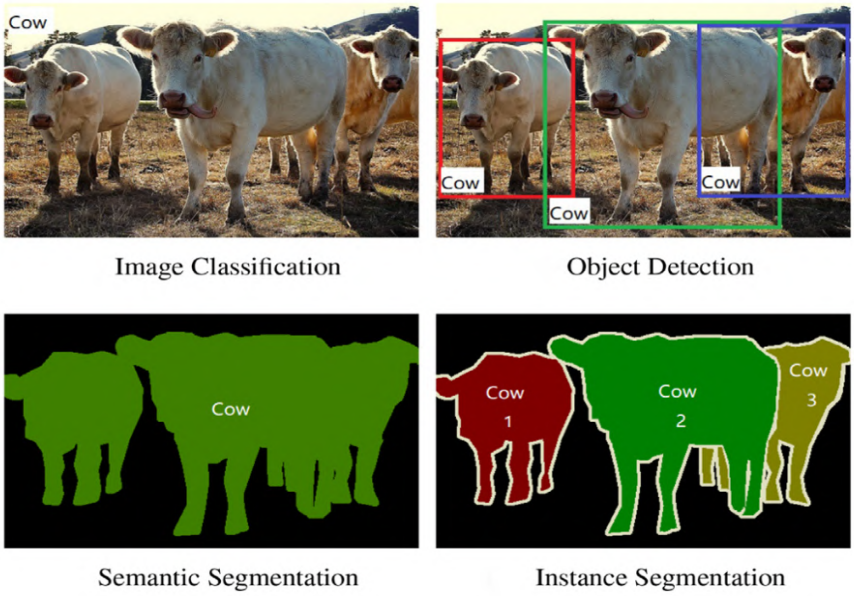


FIGURE 2.3 – Les tâches de la vision par ordinateur [89].

### 2.2.2 Les réseaux de neurones convolutifs

En 2012, lors de la compétition annuelle de vision par ordinateur ILSVRC [1], un nouvel algorithme d'apprentissage profond bat tous les records. Il s'agit d'un réseau de neu-

1. ILSVRC (ImageNet Large Scale Visual Recognition Challenge) qui est une compétition phare du domaine de la vision par ordinateur, qui a lieu annuellement depuis 2010. Le but est de localiser et classifier correctement des objets et scènes dans des images naturelles. Ce concours est organisé par le projet ImageNet, qui produit une large base d'images annotées pour faire avancer la recherche en vision par ordinateur.

rones convolutif appelé AlexNet (en raison du prénom de son auteur, Alex Krizhevsky). Les réseaux de neurones convolutifs (en anglais CNN ou ConvNet pour *Convolutional Neural Networks*) ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé : ils reçoivent des images en entrée, extraient les caractéristiques (en anglais *features*) de chacune d'entre elles, puis entraînent un classificateur dessus. Cependant, les caractéristiques sont apprises automatiquement. Les CNNs réalisent eux-mêmes tout le travail fastidieux d'extraction des caractéristiques : lors de la phase d'entraînement, l'erreur de classification est minimisée afin d'optimiser les paramètres du classificateur et les caractéristiques. De plus, l'architecture spécifique du réseau permet d'extraire des caractéristiques de différentes complexités, des plus simples au plus sophistiquées. L'extraction et la hiérarchisation automatiques des caractéristiques, qui s'adaptent au problème donné, constituent une des forces des réseaux de neurones convolutifs.

Il existe quatre types de couches pour un réseau de neurones convolutif :

– **La couche de convolution :** est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche. Son but est de repérer la présence d'un ensemble de caractéristiques dans les images reçues en entrée. Pour cela, on réalise un filtrage par convolution : le principe est de faire "glisser" une matrice représentant la caractéristique concernée sur l'image (qu'on appelle aussi matrice motif ou *Kernel* en anglais), et de calculer le produit de convolution entre cette matrice et chaque portion de l'image balayée. Une caractéristique est alors vue comme un filtre : les deux termes sont équivalents dans ce contexte. Ce processus s'appelle : extraction de caractéristiques (*feature extraction*) tel qu'il est illustré dans la figure 2.4

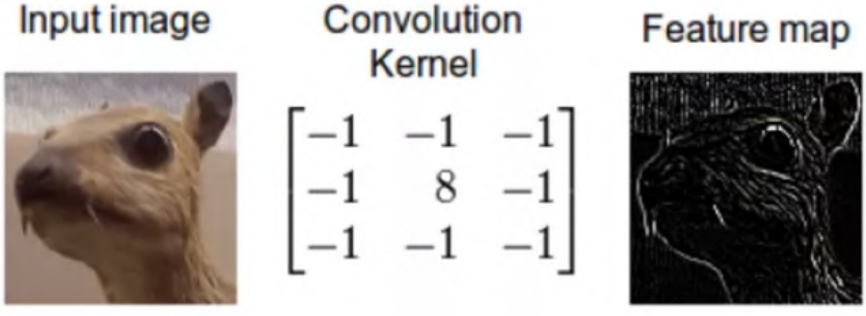


FIGURE 2.4 – Extraction de caractéristiques en utilisant le produit de convolution.

- **La couche de *pooling* :** ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs caractéristiques (*feature maps*), et applique à chacune d'entre elles l'opération de *pooling*. Sur une petite région d'image (quelques pixels en carrés), l'opération de pooling consiste à ne conserver que la valeur la plus élevée pour réduire la taille des images, tout en préservant leurs caractéristiques importantes.
- **La couche de correction ReLU :** ReLU (*Rectified Linear Units*) désigne la fonction réelle non-linéaire définie par :

$$ReLU(x) = max(0, x) \tag{2.1}$$

  La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.
- **La couche *fully-connected* :** Constitue toujours la dernière couche d'un réseau de neurones, convolutif ou non – elle n'est donc pas caractéristique d'un CNN. La dernière couche *fully-connected* permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe. Cette probabilité est donnée par une fonction d'activation. On trouve deux types de fonctions d'activation qui sont largement utilisées dans la pratique : la fonction Sigmoïde qui est utilisée généralement pour la classification binaire (mais on peut l'adapter à un problème multi-étiquette, ce qui est notre cas !), et la fonction Softmax qui est utilisée pour la classification multi-classe (figure 2.5). On utilise alors la sortie de la fonction Sigmoïde pour activer une classe qui est supérieur à un seuil donné (qui vaut généralement 0.5), et la sortie de la fonction Softmax pour activer la classe ayant la probabilité la plus grande. **Dans un problème multi-étiquette où plusieurs classes peuvent être activées en même temps, on peut donc utiliser la fonction Sigmoïde qui va calculer la probabilité de chaque classe indépendemment des autres (OVR : *One-vs-Rest*), ce qui nous permettra d'activer plus d'une classe à la fois (toutes les classes ayant une probabilité supérieure à 0.5 seront activées).**

Différents modèles sont conçus à base des CNNs ces derniers temps pour les différentes tâches de la vision par ordinateur : VGG16, ResNet, Inception, EfficientNet...pour la

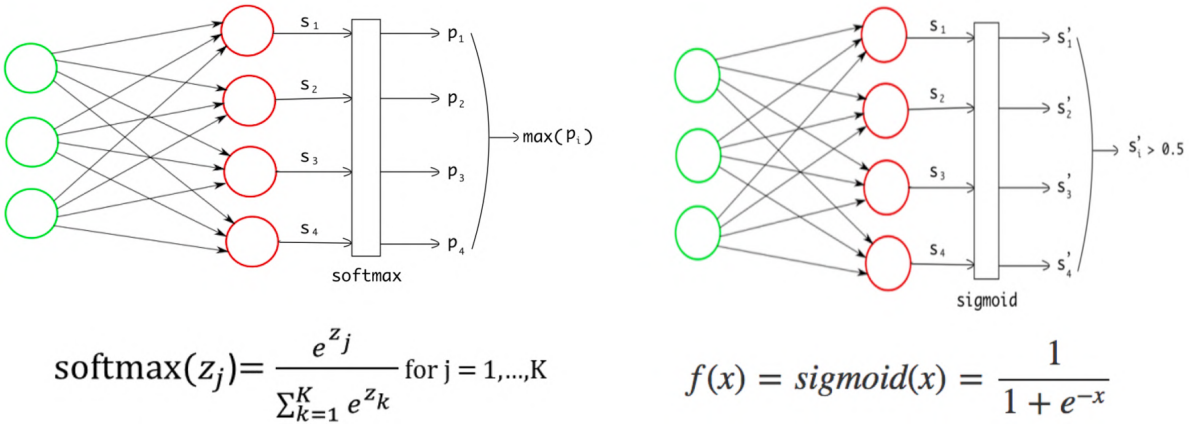


FIGURE 2.5 – Sigmoïde vs Softmax.

classification, Faster R-CNN, SSD, YOLO...pour la détection et U-Net, FCN...pour la segmentation. La figure 2.6 représente l'architecture d'un réseau de neurone convolutif.

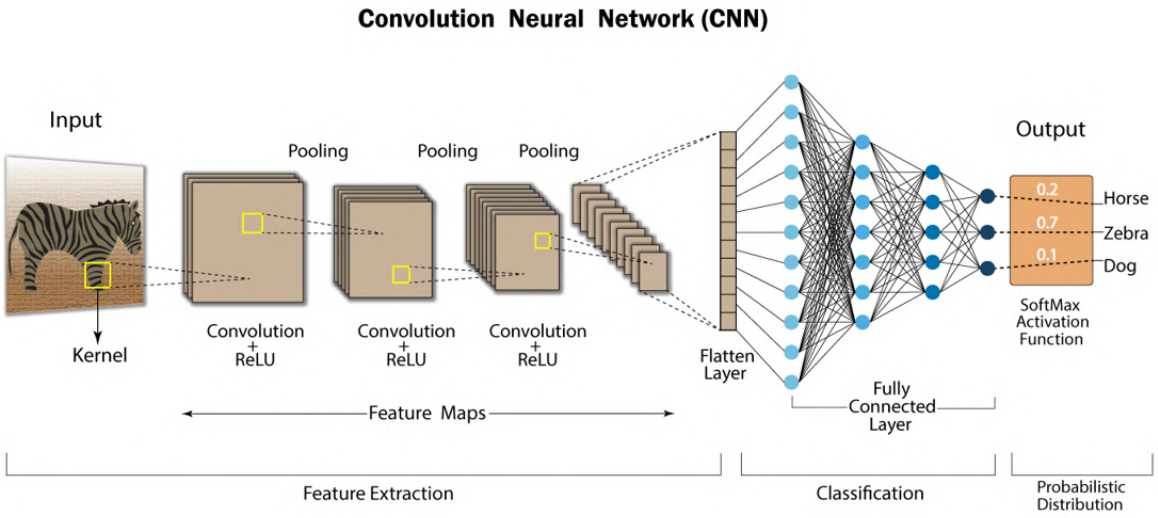


FIGURE 2.6 – Architecture générale d'un CNN.

### 2.2.3 Les *transformers*

Les *transformers* sont des modèles d'apprentissage profond utilisés à la base en traitement automatique des langues (NLP : *Natural language processing*), mais ces derniers temps on voit leurs apparition dans le domaine de la vision par ordinateur après la

performance qu'ils ont montré comparés aux CNNs traditionnels, surtout lorsqu'ils sont utilisés avec des grands jeux de données. Que ce soit en traitement automatique des langues ou en vision par ordinateur, l'architecture d'un *transformer* est principalement composée de deux blocs :

- **Encodeur :** l'encodeur reçoit une entrée et en construit une représentation (ses caractéristiques). Cela signifie que le modèle est optimisé pour acquérir une compréhension à partir de l'entrée.
- **Décodeur :** le décodeur utilise la représentation (caractéristiques) de l'encodeur avec d'autres entrées pour générer une séquence cible. Cela signifie que le modèle est optimisé pour générer des sorties.

Chacune de ces deux parties peut être utilisée indépendemment ou combinée avec l'autre, ceci selon le problème traité :

- **Modèles avec encodeur :** conviennent aux tâches qui nécessitent une compréhension de l'entrée, telles que la classification des phrases.
- **Modèles avec décodeur :** conviennent aux tâches génératives telles que la génération de texte.
- **Modèles avec encodeur-décodeur ou modèles de séquence à séquence :** bons pour les tâches génératives qui nécessitent une entrée, telles que la traduction ou la synthèse.

Les réseaux de neurones *transformers* utilisent aussi le mécanisme d'attention qui a été introduit pour la première fois en 2017 dans le fameux article "Attention Is All You Need" ! par A. Vaswani et al. [85]. En traitement automatique des langues, la couche d'attention permet de tenir compte du contexte d'un mot, ceci en spécifiant au modèle de prêter une attention particulière à certains mots de la phrase qu'on lui a transmis (et d'ignorer plus ou moins les autres).

Pour une tâche de traduction par exemple, durant l'apprentissage, l'encodeur reçoit des entrées (phrases) dans une certaine langue, tandis que le décodeur reçoit les mêmes phrases dans la langue cible désirée. Dans l'encodeur, les couches d'attention peuvent utiliser tous les mots d'une phrase (puisque la traduction d'un mot donné peut dépendre de ce qui est avant ou après la phrase). Le décodeur, cependant, fonctionne de manière séquentielle et ne peut prêter attention qu'aux mots de la phrase qu'il a déjà traduite (donc, uniquement les mots avant le mot en cours de génération). Pour traduire la phrase

"Loucif a réussi son diplôme car il a travaillé dur", le modèle doit estimer si le mot "il" est plus lié au mot "Loucif" ou au mot "diplôme". La couche d'attention des *transformers* implémente alors une méthode permettant d'avoir cette estimation. La figure 2.7 montre l'architecture originale (détaillée) d'un réseau de neurones *transformer* qui illustre une traduction d'une phrase de l'anglais vers l'italien.

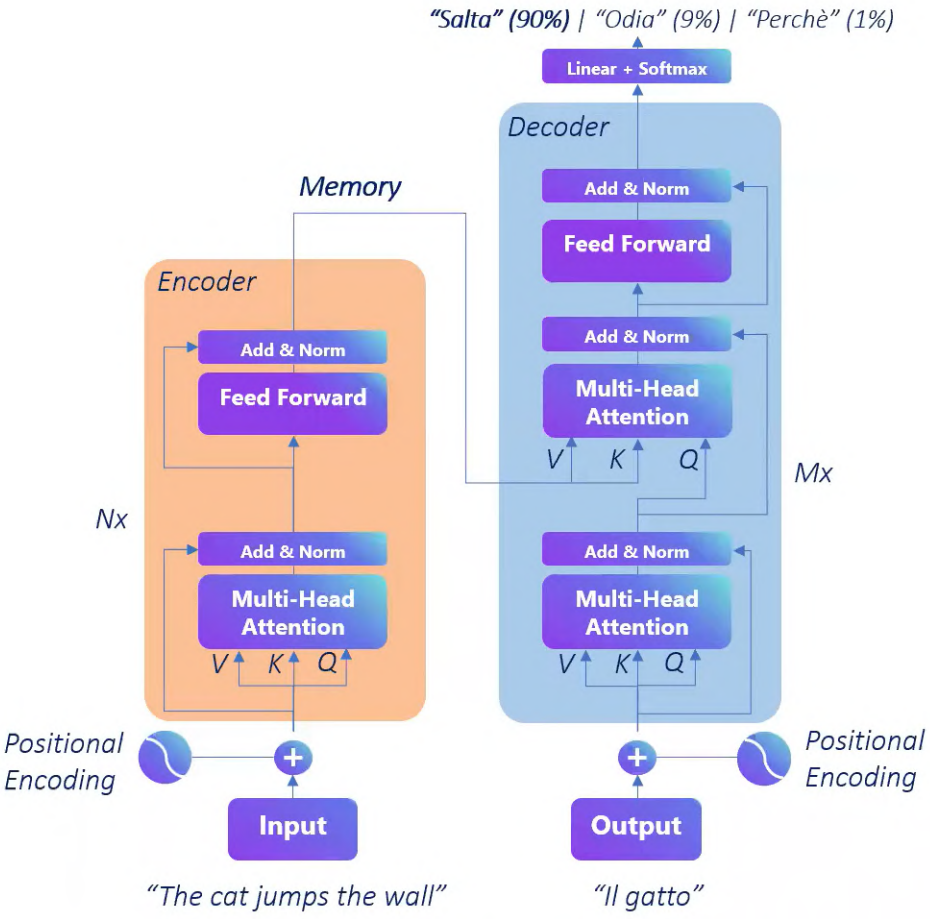


FIGURE 2.7 – Architecture détaillée d'un réseau de neurones *transformer* [54].

Dans le cadre d'une tâche en vision par ordinateur (par ex. la classification), l'équivalent du mot est une petite partie de l'image qu'on appelle *patch*. L'image est donc divisée en patches qui sont intégrés linéairement dans l'encodeur après avoir encoder la position de chaque *patch* pour ne pas perdre leurs ordre. La figure 2.8 décrit l'architecture d'un classificateur d'images à base de *transformer* avec encodeur, appelé ViT (*Vision Transformer*) [17].

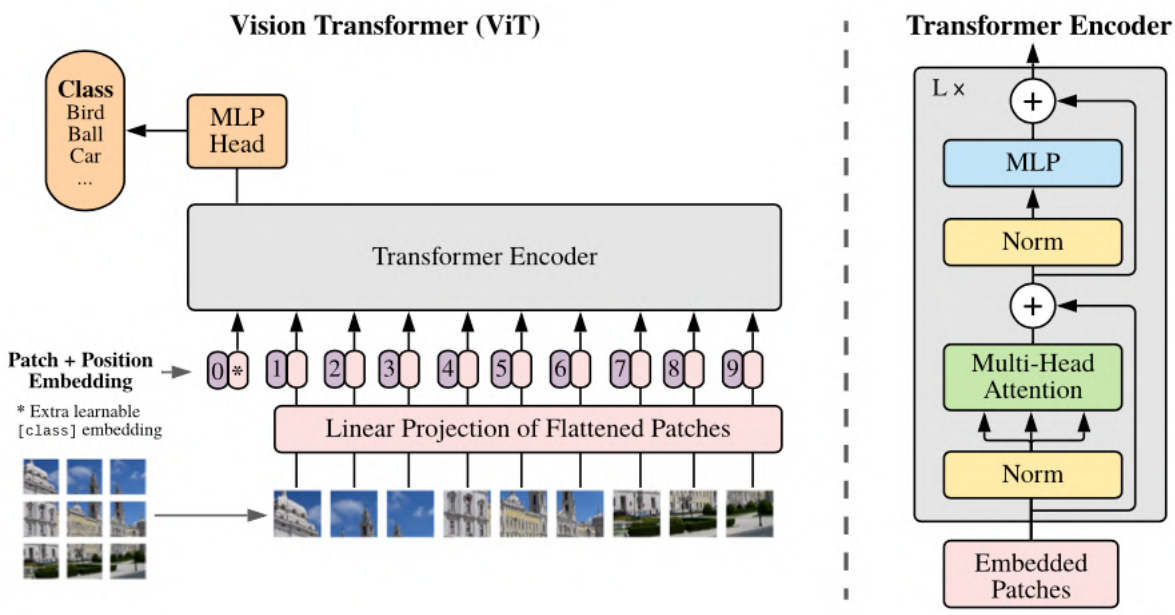


FIGURE 2.8 – Architecture de transformer ViT [17].

### 2.2.4 La saillance visuelle

L. Itti et C. Koch ont introduit pour la première fois le concept de la saillance en 1998 [36]. Lorsqu'une personne regarde une image, son attention est attirée par certaines parties dans cette image plus que d'autres. Donc la saillance d'une chose est le degré d'attention qu'elle attire par rapport aux autres choses présentes dans son environnement. Ces parties en image sont appelées les régions saillantes ou encore les cartes de saillance (*saliency maps*) [29]. Ce concept de la saillance visuelle a de nombreuses applications dans la réalité : compression d'image et vidéo [28], évaluation de la qualité d'image et vidéo [7], détection et reconnaissance d'objet [25], etc. Selon ce qu'on cherche à détecter, un objet ou une région, on distingue deux types d'algorithmes de la saillance visuelle :

- **Algorithmes de détection des objets saillants :** qui cherchent à détecter les parties de l'image qui forment des objets saillants.
- **Algorithmes de détection des régions saillantes :** qui cherchent à détecter les parties de l'image qui forment des régions (points) saillantes (figure 2.9).

Dans les deux cas, on peut catégoriser les algorithmes selon le type de calcul sur lequel ils sont basés, i.e. en deux catégories de méthodes :

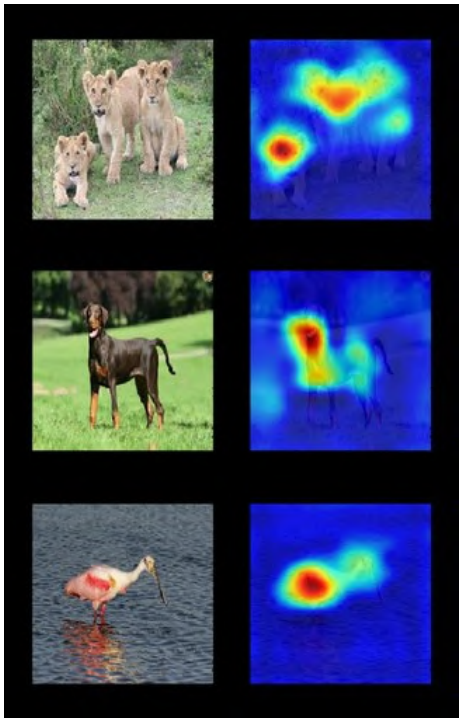

FIGURE 2.9 – Exemple des objets saillants.

- **Méthodes basées sur la perturbation (*perturbation-based*) :** qui manipulent des parties de l'image pour générer des explications en perturbant les valeurs des caractéristiques. Exemples de ce genre de méthodes : *Local Surrogate (LIME), SHapley Additive exPlanations (SHAP ).*
- **Méthodes basées sur le calcul du Gradient (*Gradient-based*) :** qui calculent le gradient de la prédiction du réseau de neurone (le score de la classification) par rapporte aux caractéristiques d'entrée. Elles nous permettent de savoir si un changement au niveau des propriétés d'un pixel donné (intensité, couleur) peut changer (augmente ou diminue) le score de prédiction. Exemples de méthodes de cette gatégorie : *Vanilla Gradient, Grad-CAM, SmoothGrad, FullGrad.*

Ces derniers temps, on utilise aussi même les algorithmes d'apprentissage profond pour la saillance visuelle [39].

### 2.2.5 Le transfert d'apprentissage

L'apprentissage par transfert (*transfer learning*) consiste à transférer les paramètres appris par un CNN pré-entrainé sur un nombre de données important (par ex. ImageNet),

vers un autre jeu de données souvent moins important (le jeu de données que nous allons choisir) pour une autre tâche (classification/détection/segmentation de défaut). Il existe trois stratégies de transfert d'apprentissage tel qu'il est illustré dans la figure 2.10 [11]. Dans la stratégie de transfert d'apprentissage *Cross-domain*, le modèle est entraîné sur un jeu de données très grand différent du domaine de la tâche cible, puis quelques couches supérieures de ce modèle sont ré-entraînées sur le jeu de données cible tout en gardant les autres couches pré-entraînées (les couches inférieures). Cette opération de geler quelques couches et dégeler d'autres s'appelle le *Fine-tuning*. Dans la stratégie *In-domain*, les deux jeux de données d'origine et cible appartiennent au même domaine. La troisième stratégie est similaire à la première stratégie, sauf avec celle-ci on utilise un petit jeu de données qui appartient au même domaine que celui de jeu de donnée cible, pour la validation lors de l'entraînement du modèle.

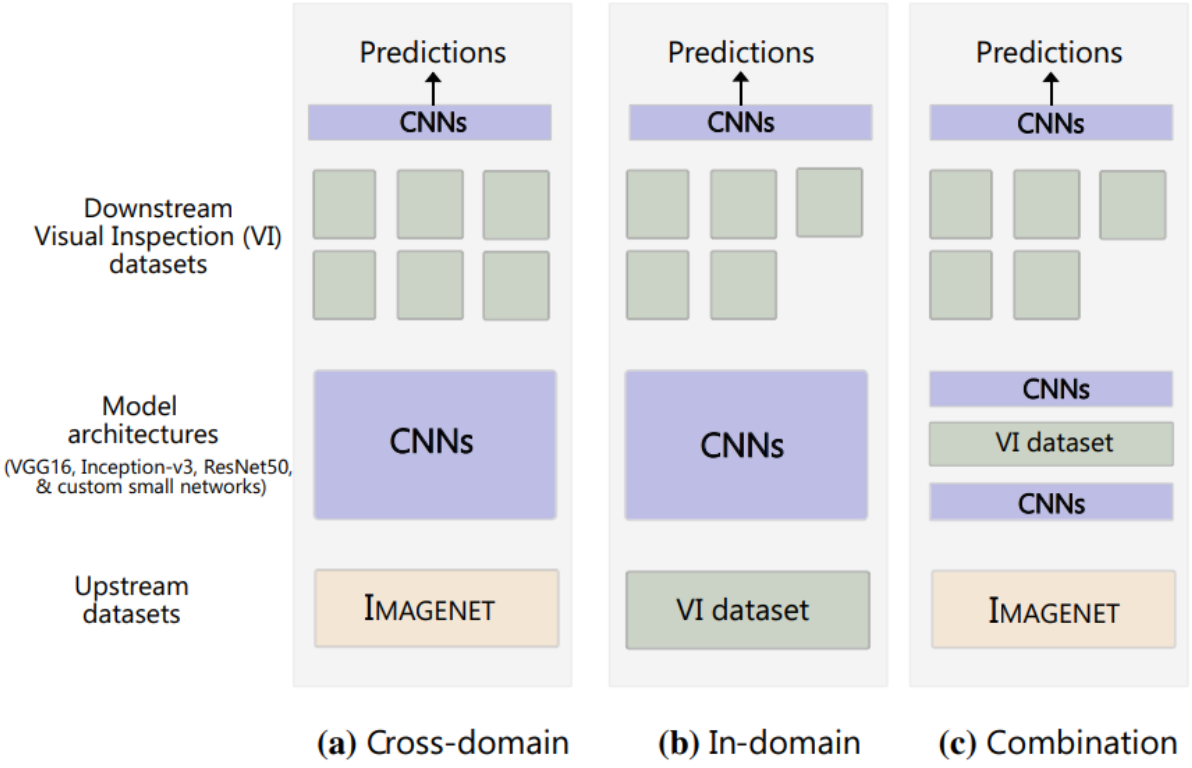


FIGURE 2.10 – Stratégies de transfert d'apprentissage [11].

### 2.2.6 Augmentation de données

L'augmentation de données (*Data augmentation*) est une technique de pré-traitement de données utilisée pour augmenter la généralisabilité du modèle à entraîner en augmentant la diversité des exemples d'apprentissage pour lui. En vision artificielle , il existe deux niveaux d'application de *data augmentation* (figure 2.11) :

- **Image level augmentation :** déformer tous les pixels de l'image avec une opération mathématique qui modifiera soit la couleur de l'image, ex. : contraste, luminosité, ou bien les emplacements des pixels, ex. : rotation, translation.
- **Bounding box level augmentation :** un *bounding box* est le cadre qui délimite un objet dans l'image. Cette technique alors va déformer uniquement le contenu en pixels des *bounding boxes* (le contenu des emplacements des objets à l'intérieur de l'image), ex. : BBox Only Rotate, BBox Only FlipLR. Bounding box level augmentation est une technique très importante qui a été présentée dans un article de Google [97] où les chercheurs ont démontré que cette technique génére des améliorations systémiques, en particulier pour les modèles adaptés à de petits jeux de données (on verra que ça sera le cas pour nous).

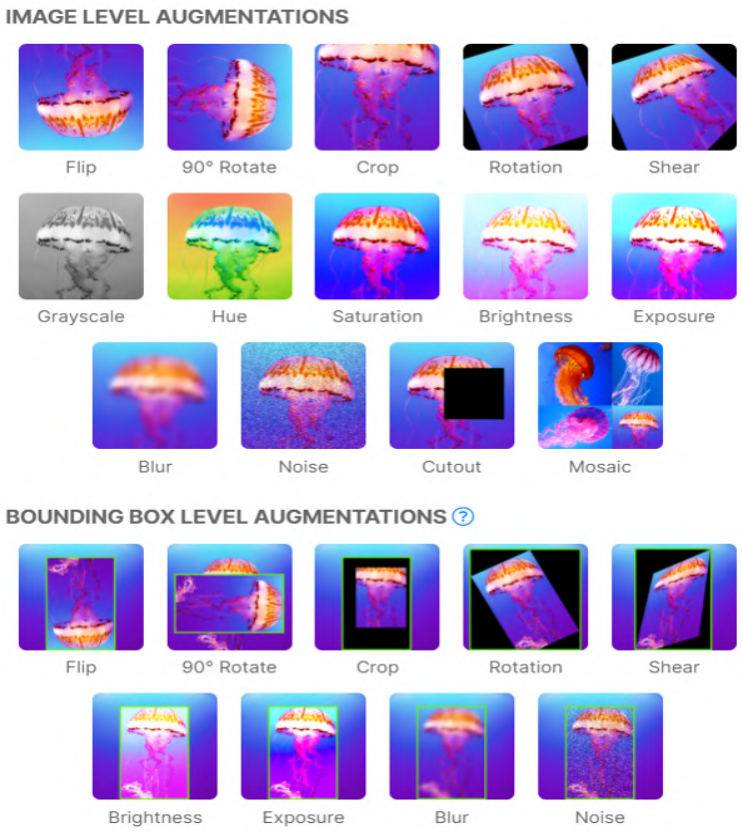


FIGURE 2.11 – Niveaux de l'augmentation de données.

### 2.2.7 Les fonctions de perte

Dans un contexte où on a un problème d'optimisation, la fonction qui sert de critère pour déterminer la meilleure solution est appelée : la fonction objectif. On cherche soit à minimiser ou bien à maximiser cette fonction. Dans le contexte d'apprentissage automatique avec un réseau de neurones, on cherche à minimiser l'erreur, donc cette fonction est appelée dans ce cas précis : la fonction d'erreur ou encore la fonction de perte (en anglais : *Loss Function*). Cette dernière est utilisée durant l'entraînement pour calculer l'erreur de l'apprentissage et propager cette erreur afin d'ajuster les poids de l'algorithme à chaque itération. Toutes les fonctions d'erreur utilisées pour l'apprentissage des réseaux de neurones doivent intégrer une certaine mesure des distances entre les valeurs cibles et les prévisions correspondantes aux entrées. Il existe des fonctions de perte pour chaque type de problème d'apprentissage (figure 2.12). Dans notre cas, dans chaque type de problème que nous allons traiter (classification/détection/segmentation de défaut), on choisira toujours *Binary Cross Entropy* comme fonction de perte et la

sigmoïde comme fonction d'activation vu que notre problème est toujours un problème multi-classe multi-étiquette. La fonction de perte Binary Cross Entropy est définie par la formule 2.2 où $y$ est l'étiquette réel, et $p$ est la probabilité que produit la fonction sigmoïde pour la classe concernée.

$$Loss_{BCE} = -(y \log(p) + (1 - y) \log(1 - p)) \tag{2.2}$$

| Problem Type | Output Type | Final Activation Function | Loss Function |
|---|---|---|---|
| Regression | Numerical value | Linear | Mean Squared Error (MSE) |
| Classification | Binary outcome | Sigmoid | Binary Cross Entropy |
| Classification | Single label, multiple classes | Softmax | Cross Entropy |
| Classification | Multiple labels, multiple classes | Sigmoid | Binary Cross Entropy |

FIGURE 2.12 – Types de fonctions de perte selon le problème d'apprentissage.

### 2.2.8 Les métriques d'évaluation

Une métrique d'évaluation est utilisée pour évaluer la performance du modèle. Comme dans les fonctions de perte, il existe aussi des métriques d'évaluation pour chaque type d'apprentissage comme la figure 2.13 le montre.

Pour un problème de classification (tel que notre cas), les métriques d'évaluation des classificateurs peuvent être divisées en trois groupes [24] :

– **Métriques de seuil (*Threshold Metrics*) :** métriques basées sur un seuil et une compréhension qualitative de l'erreur. On trouve alors :

$$Exactitude(Accuracy) = \frac{CorrectPredictions}{TotalPredictions} \tag{2.3}$$

$$Error = 1 - Accuracy \tag{2.4}$$

$$Sensitivity = Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.5}$$

$$Specificity = \frac{TrueNegative}{FalsePositive + TrueNegative} \tag{2.6}$$

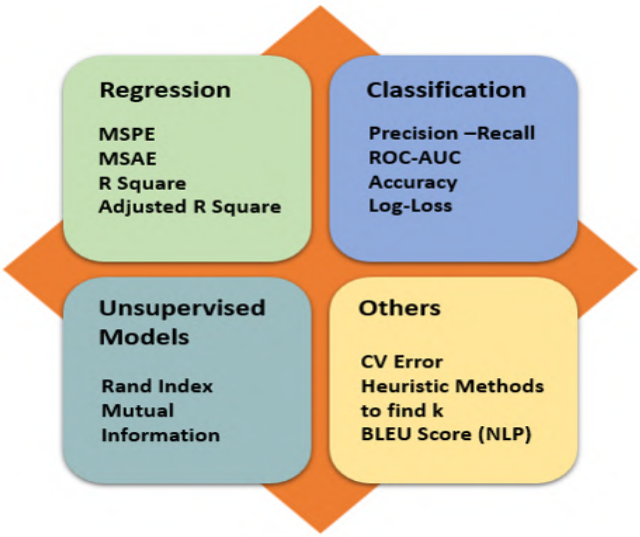


FIGURE 2.13 – Taxonomie des métriques d'évaluation [62].

$$G - Mean = \sqrt{Sensitivity * Specificity} \tag{2.7}$$

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2.8}$$

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.9}$$

tel que :

$TruePositive$ : nombre de prédictions positives correctement classées ;

$FalsePositive$ : nombre de prédictions négatives mal classées comme positives ;

$TrueNegative$ : nombre de prédictions négatives correctement classées ;

$FalseNegative$ : nombre de prédictions positives mal classées comme négatives ;

L'une des limitations des métriques de seuil est qu'elles supposent que la distribution de la classe observée dans l'ensemble de données d'apprentissage correspondra à celle de l'ensemble de test, ce qui n'est pas toujours le cas !

– **Métriques probabilistes (*Probability Metrics*) :** métriques basées sur une compréhension probabiliste de l'erreur, c'est-à-dire la mesure de l'écart par rapport à la vraie probabilité. Un exemple d'une métrique probabiliste est la Binary Cross Entropy que nous avons vu précédemment.

– **Métriques de classement (*Ranking Metrics*) :** métriques qui évaluent le degré ou la mesure de séparabilité entre les classes. On peut trouver deux exemples de ce type de métriques : ROC–AUC et PR-AUC. La ROC (*Receiver Operating Characteristic*) est une courbe de probabilité. Elle trace le taux des vrais positifs en fonction du taux des faux positifs pour différents seuils de classification (figure 2.14). Et la AUC (*Area under the ROC Curve*) est la surface sous la courbe ROC. Elle représente le degré ou la mesure de séparabilité entre les classes. Pour la PR-AUC (*Precision-Recall Area Under Curve*), elle représente la surface sous la courbe *Precision-Recall*, obtenue en traçant des points (*recall, precision*) pour différentes valeurs du seuil de classification.

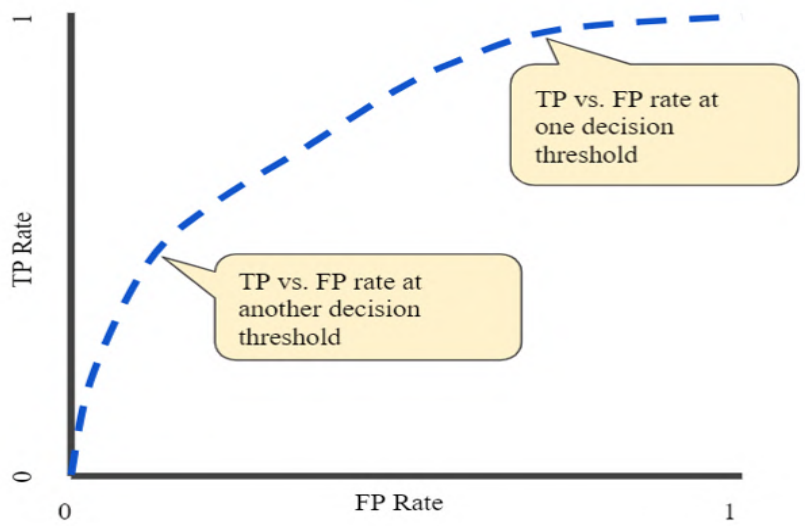


FIGURE 2.14 – La courbe ROC [1].

Pour un problème de détection d'objet, la métrique d'évaluation la plus fréquemment utilisée est le mAP (*Mean Average Precision*) qui représente la valeur moyenne de l'AP de toutes les classes. Le AP est l'aire sous la courbe de *Precision-Recall* (figure 2.15). Pour classer les détections en TP (*TruePositive* ou Vrai positif) ou FP (*FalsePositive* ou Faux positif), un seuil pour l'IOU (*Intersection Over Union*) est fixé à t = 0,5, t = 0,75 et t = 0,95. L'IOU mesure l'intersection sur l'union de la boite englobante réelle et celle prédite par le modèle. En détection d'objet, le concept de TN (*TrueNegative*) n'est pas utilisé, et le FN (*FalseNegative*) est une boite englobante qui n'est pas détectée par le modèle.

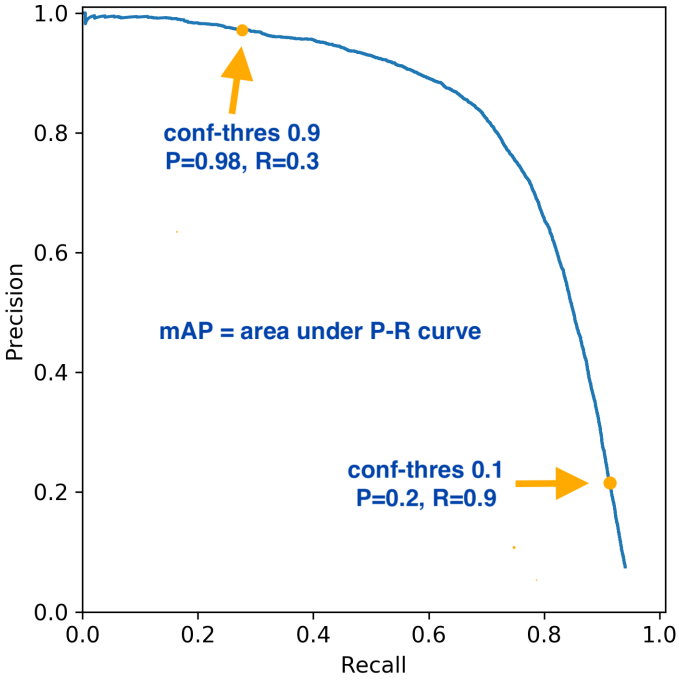


FIGURE 2.15 – La courbe PR.

Dans le cas de la segmentation sémantique, les deux métriques les plus utilisées sont *Pixel Accuracy* et *Dice Coefficient*. Le *Pixel Accuracy* est le pourcentage de pixels qui sont correctement classés (formule 2.10). Le *Dice Coefficient* est défini par la formule 2.11.

$$PixelAccuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.10}$$

$$DiceCoefficient = \frac{2TP}{2TP + FP + FN} \tag{2.11}$$

## 2.3 Méthodes d'inspection des ponts : Revue de la littérature

Plusieurs taxonomies des méthodes d'inspection des ponts ont été proposées dans la littérature [91, 5, 38] sous différents contextes. Dans notre cas, tel que nous l'avons décrit dans notre article en [49], on classe les méthodes d'inspection des ponts en méthodes d'inspection non visuelles et visuelles. Dans la première catégorie, des recherches ont récemment été menées sur l'utilisation de drones et de la réalité augmentée (AR), guidées par des algorithmes d'apprentissage profond pour l'inspection des ponts. Dans ce qui suit, nous détaillons ces méthodes et les résumons dans la figure 2.16

### 2.3.1 Méthodes visuelles pour l'inspection des ponts

Pour l'inspection visuelle, des méthodes ont été proposées utilisant différentes modalités de données visuelles. La plupart des méthodes existantes d'inspection visuelle sont basées sur spectre visible, bien que l'imagerie infrarouge / thermique puisse être utilisée pour détecter des défauts spéciaux tels que le délaminage [84]. Jusqu'à récemment, les méthodes statistiques étaient les plus utilisées pour détecter les défauts dans les images. Des caractéristiques de l'image telles que les bords, la couleur et la texture sont d'abord extraites. Ensuite, des techniques de reconnaissance des formes telles que le regroupement et la classification sont appliquées pour la détection de défauts [37]. Abdel-Qader et al. ont comparé quatre techniques de traitement d'image bien connues : Transformée rapide de Haar (FHT), transformée rapide de Fourier, détecteur de Sobel et détecteur de Canny pour la détection des fissures dans les ponts [4]. Ils ont montré que la performance du FHT dans la détection des fissures est meilleure que celle des autres méthodes. Salman et al. ont proposé une méthode pour détecter automatiquement les fissures, basée sur le filtre de Gabor [70]. Talab et al. ont utilisé le filtre de Sobel et la méthode de Otsu [10] pour détecter les fissures [81].

### 2.3.2 Méthodes non visuelles pour l'inspection des ponts

Différentes techniques ont été utilisées pour inspecter les ponts comme alternative à l'inspection visuelle. Par exemple, des techniques acoustiques ont été utilisées pour détecter les changements dans la hauteur du son résultant de la frappe de la surface en béton avec un marteau [32]. L'un des avantages de ces méthodes est leur faible coût, mais

elles sont moins précises dans la détection de défauts ayant des couches profondes. Ils sont également moins applicables aux ponts qui ne sont pas en béton [52]. Un autre groupe de techniques d'inspection des ponts est celle de radar à pénétration de sol, appelée aussi géoradar (*Ground-Penetrating Radar* : GPR). Dans cette méthode, les inspecteurs utilisent un rayonnement électromagnétique pour imager les couches sous-jacentes de l'élément concerné dans le pont, et détecter les éventuels défauts [14]. L'inconvénient de cette approche est sa forte consommation d'énergie et le besoin d'expertise pour interpréter les données. Une autre méthode est celle de potentiel de demi-cellule (*Half-Cell Potential*) qui évalue la différence du potentiel électrique entre le béton armé et une électrode standard pour la détection de défauts [67]. Cette méthode peut détecter la corrosion dans ses premiers temps, mais à un coût élevé.

### 2.3.3 Réalité augmentée (AR) et l'imagerie par drone pour l'inspection des ponts

La réalité augmentée (AR) et/ou l'imagerie par drones, sont des méthodes émergentes d'inspection des ponts. Les drones permettent d'accéder aux zones difficilement accessibles, éliminant ainsi le besoin de présence physique d'une présence sur les lieux. Wells et al. [88] ont utilisé différents types de drones sur huit ponts de différentes tailles, emplacements et conditions pour évaluer l'efficacité des drones à inspecter les ponts. Kilic et al. [45] ont décrit les avantages de combiner la réalité augmentée avec les autres méthodes d'inspection telles que : l'inspection visuelle, radar à pénétration de sol (GPR), capteur de distance laser (LDS), thermographie infrarouge (IRT) et caméra télescopique (TC). Hu et al. [33] ont proposé aussi une méthode combinant AR et GPR pour inspecter les ponts en béton.

### 2.3.4 L'apprentissage profond pour l'inspection des ponts

Les développements récents de l'apprentissage profond (DL), en particulier pour la détection et la classification d'objet, ont libéré d'énormes possibilités d'applications, comme la surveillance et l'inspection visuelle en utilisant les drones équipés d'une vision artificielle pour la détection d'anomalies [50], [58]. Cela a également conduit à la publication de plusieurs jeux de données de référence avec annotations, en facilitant ainsi l'entraînement et l'évaluation des modèles. L'un des jeux de données les plus connus est *CONcrete DEfect BRidge IMage dataset* (CODEBRIM) proposé par Mundt et al.

[58], qui présente de multiples défauts, y compris : fissure (*crack*), éclatement (*spallation*, barre d'armature exposée (*exposed bar*), efflorescence et corrosion. Cela a donné lieu à plusieurs travaux qui utilisaient des techniques d'apprentissage automatique pour détecter les anomalies dans les surfaces des ponts en béton [23]. Par exemple, les auteurs de [47] ont utilisé des CNNs pour détecter les fissures en utilisant la classification d'images. Cette méthode a montré une bonne performance, mais une ombre et un faible contraste peuvent affecter son exactitude (*accuracy*). Dans [92], les auteurs ont proposé d'utiliser un détecteur à un seul étage (*single-stage detector*) pour la détection et la classification des défauts basé sur le fameux modèle *You Only Look Once* V3 (YOLO-v3) [65]. L'approche a réussi à détecter de multiples défauts dans les ponts en béton, en atteignant 80% d'exactitude sur le jeu de données CODEBRIM en surpassant ainsi le détecteur à deux étages (*two-stage detector*) Faster R-CNN [66]. Cependant, l'approche a montré certaines limites lorsqu'il s'agit de défauts petits et peu contrastés. Dans le même contexte, Tabernik et al. [80] ont utilisé une architecture d'apprentissage profond basée sur la segmentation d'objet pour traiter spécifiquement le problème de détection de fissures. Enfin, on a utilisé les algorithmes de détection d'objet YOLO-v3 et MobileNetV3-SSD pour détecter et classifier les défauts en béton, qui ont montré des résultats prometteurs [41]. La table 2.2 résume les algorithmes d'apprentissage profond utilisés dans la littérature pour l'inspection des ponts via le transfert d'apprentissage depuis ImageNet/COCO.

Toutes ces méthodes ont donné des succès remarquables lorsqu'il s'agit d'un problème de détection avec une seule classe de défauts telle que les fissures, mais elles perdent leurs efficacité lorsqu'on a un problème avec d'autres types de défauts . En effet, développer un modèle unique capable de traiter avec toutes les différentes classes de défauts est difficile à achever. La détection et la classification de défaut, et en particulier dans les ponts en béton, posent plusieurs défis. Premièrement, même si les défauts peuvent être détectés et classés en différentes classes, il y a une grande variabilité intra-classe en raison des conditions variables de l'éclairage, les changements d'angle de vue et la taille et le degré de gravité des défauts [23]. Par ailleurs, certaines classes de défauts peuvent avoir un grand chevauchement (*Overlapping*), et c'est le cas souvent qu'on peut avoir dans la réalité, ce qui rend le problème de détection et de classification un problème multi-étiquette. Enfin, les jeux de données existants, contiennent souvent des étiquettes (*labels*)

TABLE 2.2 – État de l'art des méthodes d'inspection des ponts en béton basées sur l'apprentissage profond via le transfert d'apprentissage depuis ImageNet/COCO (Cls : Classification ; Dét : Détection ; Seg : Segmentation) [11].

| **Méthode proposée** | **Année** | **Tâche** | **Modèle pré-entraîné** |
|---|---|---|---|
| CovNet Model [75] | 2017 | Cls de crack | VGG16 |
| Custom CNN [8] | 2017 | Dét de corrosion | VGG16 |
| CNN + edge detection [15] | 2018 | Cls de crack | AlexNet |
| Custom CNN [16] | 2018 | Cls de crack | AlexNet |
| Custom CNN [57] | 2018 | Dét de crack | MobiNet |
| R-CNN [47] | 2018 | Dét de crack | R-CNN |
| TERNAUSNET [9] | 2019 | Seg de crack | VGG16 |
| Custom CNN [22] | 2019 | Cls de défauts | Inception-v3 |
| Custom CNN [96] | 2019 | Cls de défauts | Inception-v3 |
| Multi-classifier [35] | 2019 | Cls de défauts | Inception-v3 |
| Custom CNN [18] | 2019 | Seg de crack | VGG16 |
| Semantic seg. network [95] | 2019 | Seg de crack | VGG16 |
| Extension of YOLOv3 [92] | 2020 | Dét de défauts | YOLOv3 |
| Improved EfficientNetB0 [76] | 2020 | Cls de crack | MobileNetV2 |
| Modifed YOLO [82] | 2021 | Dét de crack | YOLO |
| Shallow CNN [46] | 2021 | Cls de crack | LeNet-5 |
| Custom CNN [63] | 2021 | Seg de défauts | ResNet34 |
| CNN for pixel level [13] | 2021 | Cls + Seg de crack | Multiple modèles |
| YOLO-v3 [41] | 2021 | Dét de défauts | EfficientNet |
| MobileNetV3-SSD [41] | 2021 | Dét de défauts | MobileNetV3 |
| CNN+Transformer [87] | 2022 | Cls de défauts | Custom CNN |

bruités qui posent une difficulté supplémentaire à entraîner et évaluer correctement les algorithmes d'apprentissage [59].

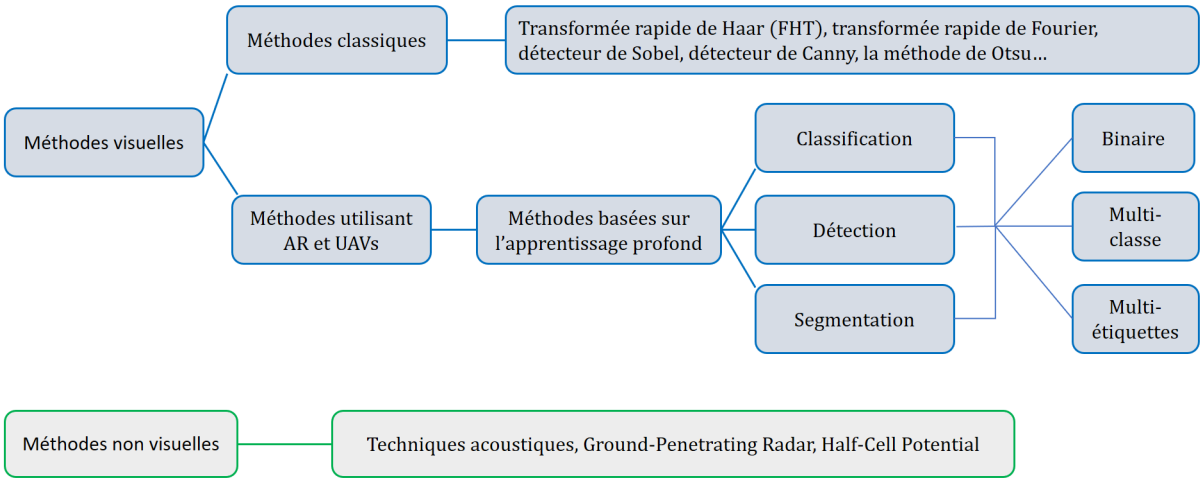


FIGURE 2.16 – Taxonomie des méthodes d'inspection des ponts.

## 2.4 Conclusion

Avec ce chapitre, nous avons présenté en premier lieu quelques généralités sur l'inspection de défauts dans les ponts en béton en se basant sur des manuels d'inspection des ponts et des structures (MIP et MIS). En deuxième lieu, nous avons donné les concepts techniques liés à notre problématique. Et enfin nous avons passé en revue la littérature des techniques utilisées pour inspecter les défauts ; nous avons vu qu'elles sont divisées principalement en trois catégories de méthodes : méthodes visuelles, non visuelles et celles basées sur AR et l'imagerie par drones équipé d'une vision artificielle grâce à l'apprentissage profond. La tendance actuelle est vers l'utilisation de cette dernière méthode avec les développements récents des modèles d'apprentissage profond. Toutefois, ces méthodes ne sont pas très efficaces lorsque il s'agit d'un problème de classification, de détection ou de segmentation de défauts multi-étiquette, ce qui est souvent le cas dans la réalité. Dans le prochain chapitre, on détaillera les éléments qui constituent notre approche proposée pour résoudre la problématique.

# Chapitre 3

# Méthodologie

Ce chapitre représente le corps du notre travail de recherche. On a vu à la fin du chapitre précèdent que les méthodes de détection et de classification de défauts existantes dans la littérature ne résolvent pas les problèmes d'inspection qu'on peut avoir souvent dans la réalité. De ce fait, nous commençons d'abord par décrire notre approche générale. Ensuite, nous détaillons les trois axes de recherche qui la composent : la classification, la détection et la segmentation de défauts multi-classe multi-étiquette. En particulier, nous présentons notre nouvelle méthode appelée Saliency-based Multi-label Defect Detector (SMDD-Net), proposée dans le cadre de la détection de défauts et qui a donnée des résultats prometteurs.

## 3.1 Description générale de l'approche

Notre approche combine deux disciplines : l'intelligence artificielle (AI) et la robotique. Le drone est un agent robotique équipé d'une vision artificielle. Cette dernière consiste à utiliser les trois techniques fondamentales de la vision par ordinateur : la classification, la détection et la segmentation d'objet pour inspecter les défauts qui peuvent être observés dans les ponts, puis évaluer chaque défaut en lui affectant un degré de gravité. Cela se passe principalement par six étapes tel qu'il est illustré par Hüthwohl et al. [35] dans la figure 3.1.

Nous nous sommes alors concentrés sur chacun des trois problèmes liés à l'inspection des ponts, à savoir : 1) la classification de défaut, 2) la détection de défauts et 3) la segmentation de défaut. Chaque problème est résolu à l'aide d'une combinaison d'algorithmes d'apprentissage automatique et de traitement d'images que nous détaillerons

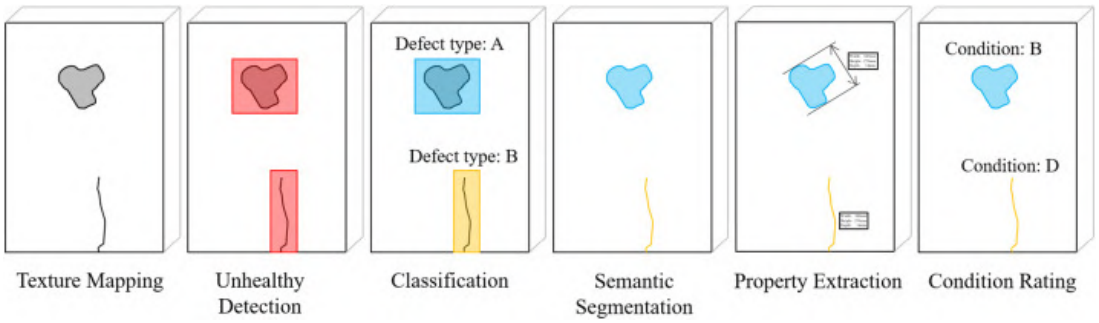


FIGURE 3.1 – Étapes de détection de défauts [35].

dans les prochains sections de ce chapitre. En particulier, les réseaux de neurones profonds ont été utilisés en raison des performances exceptionnelles qu'ils ont atteintes ces dernières années pour les applications de vision par ordinateur tel que nous l'avons vu dans le précèdent chapitre.

Une étude comparative des algorithmes d'apprentissage profond est réalisée alors pour sélectionner les architectures les plus appropriées pour la classification, la détection et la segmentation de défauts sur les images. En particulier, les architectures ResNet, Inception, EfficientNet et leurs variantes sont explorées pour la classification, YOLO, RetinaNet, Faster R-CNN et leurs variantes pour la détection, et FCN, U-Net et leurs variantes pour la segmentation. Pour avoir de meilleurs performances, le *Transfer Learning* avec le *Fine-tuning* sont appliqués aux modèles testés [11]. Les résultats des tests effectués sont discutés dans le prochaine chapitre.

## 3.2 Classification d'objet pour l'inspection de défaut

En raison de la complexité des défauts qui peuvent se manifester dans les ponts en béton, certains chercheurs ont proposé d'utiliser un classificateur à plusieurs étages : multi-classificateur. Hüthwohl et al. ont proposé un jeu de données MCDS (*Multi-Classifier DataSet for reinforced concrete bridge defects*) défini sous forme d'une hiérarchie de classificateurs à trois étages (détaillée dans le prochain chapitre) comme illustré dans la figure 3.2 [35]. Cela consiste à entraîner le classificateur sur le premier étage qui est composé de cinq classes de défauts mutuellement exclusives (plus la classe background : no defect). Si l'une des classes génériques (des classes qui contiennent d'autres classes) est activée, alors on appelle le classificateur de deuxième étage qui est entraîné sur les sous-classes des classes génériques du premier étage, et ainsi de suite. C'est une solu-

tion qui peut résoudre le problème de la classification multi-classe multi-multi-étiquette lorsque les classes sont liées entre elles par une relation de causalité ou de corrélation.

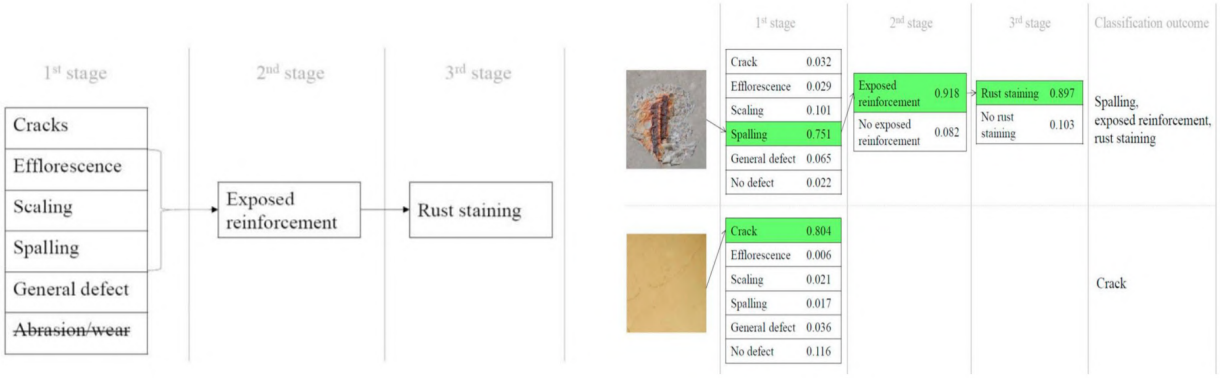


FIGURE 3.2 – MCDS : Multi-Classifier DataSet for reinforced concrete bridge defects[35].

Dans notre cas, on utilise un seul classificateur multi-classe multi-étiquette à base des CNNs, avec la fonction sigmoïde en sortie qui peut activer plusieurs classes en même temps. La fonction de perte *Binary Cross Entropy* sera utilisée comme expliqué avant. nous aurons deux scénarios à examiner : le premier sera d'entraîner juste la dernière couche des classificateurs choisis pour les tests. Le deuxième scénario consistera à dégeler quelques couches inférieures et créer d'autres couches supérieures aussi (le Fine-tuning).

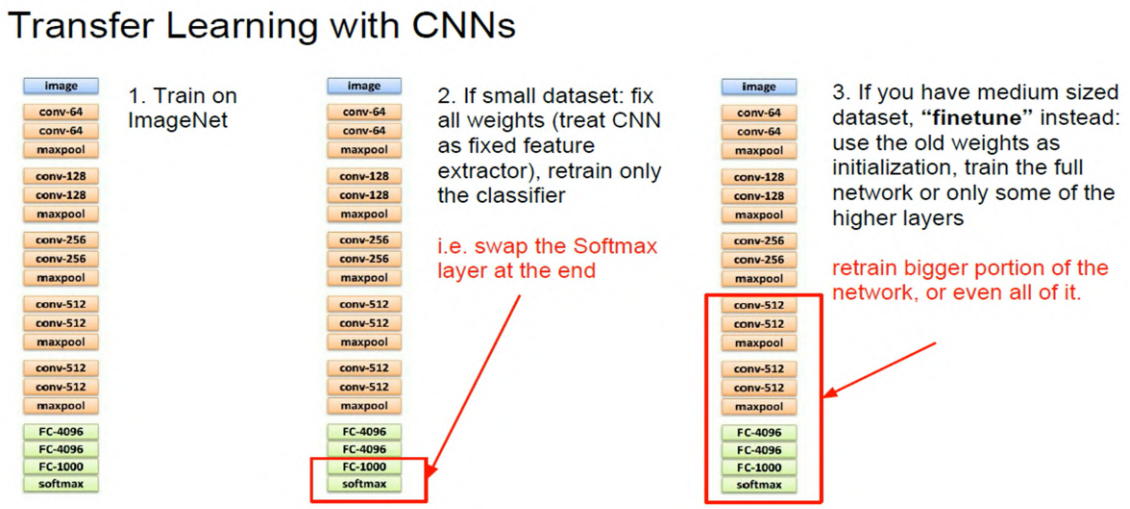


FIGURE 3.3 – Notre approche pour le problème de classification de défauts [51].

Nous choisissons d'entraîner les classificateurs les plus populaires implémentés par la bibliothèque d'apprentissage profond keras (figure 3.4), ainsi que d'autres qui sont basés sur le mécanisme d'attention tel que le transformer ViT. Des techniques comme : *data augmentation*, *shuffling* ,*early stopping* et *dropout* seront utilisées pour éviter le sur-apprentissage et améliorer la performance. Pour évaluer les classificateurs à tester, nous utilisons AUC-ROC comme métrique de base (justifié dans le prochain chapitre). Les résultats seront comparés à ceux de l'état de l'art (figure 3.5 [11])

| Model | Size | Top-1 Accuracy | Top-5 Accuracy | Parameters | Depth |
|---|---|---|---|---|---|
| Xception | 88 MB | 0.790 | 0.945 | 22,910,480 | 126 |
| VGG16 | 528 MB | 0.713 | 0.901 | 138,357,544 | 23 |
| VGG19 | 549 MB | 0.713 | 0.900 | 143,667,240 | 26 |
| ResNet50 | 98 MB | 0.749 | 0.921 | 25,636,712 | - |
| ResNet101 | 171 MB | 0.764 | 0.928 | 44,707,176 | - |
| ResNet152 | 232 MB | 0.766 | 0.931 | 60,419,944 | - |
| ResNet50V2 | 98 MB | 0.760 | 0.930 | 25,613,800 | - |
| ResNet101V2 | 171 MB | 0.772 | 0.938 | 44,675,560 | - |
| ResNet152V2 | 232 MB | 0.780 | 0.942 | 60,380,648 | - |
| InceptionV3 | 92 MB | 0.779 | 0.937 | 23,851,784 | 159 |
| InceptionResNetV2 | 215 MB | 0.803 | 0.953 | 55,873,736 | 572 |
| MobileNet | 16 MB | 0.704 | 0.895 | 4,253,864 | 88 |
| MobileNetV2 | 14 MB | 0.713 | 0.901 | 3,538,984 | 88 |
| DenseNet121 | 33 MB | 0.750 | 0.923 | 8,062,504 | 121 |
| DenseNet169 | 57 MB | 0.762 | 0.932 | 14,307,880 | 169 |
| DenseNet201 | 80 MB | 0.773 | 0.936 | 20,242,984 | 201 |
| NASNetMobile | 23 MB | 0.744 | 0.919 | 5,326,716 | - |
| NASNetLarge | 343 MB | 0.825 | 0.960 | 88,949,818 | - |
| EfficientNetB0 | 29 MB | - | - | 5,330,571 | - |
| EfficientNetB1 | 31 MB | - | - | 7,856,239 | - |
| EfficientNetB2 | 36 MB | - | - | 9,177,569 | - |
| EfficientNetB3 | 48 MB | - | - | 12,320,535 | - |
| EfficientNetB4 | 75 MB | - | - | 19,466,823 | - |
| EfficientNetB5 | 118 MB | - | - | 30,562,527 | - |
| EfficientNetB6 | 166 MB | - | - | 43,265,143 | - |
| EfficientNetB7 | 256 MB | - | - | 66,658,687 | - |

FIGURE 3.4 – Les classificateurs à base des CNNs choisis pour les tests [2].

| | CDS | SDNETv1 | BCD | ICCD | MCDS | CODEBRIM |
|---|---|---|---|---|---|---|
| | *Random Initialization* | | | | | |
| **CBR Tiny** | 0.79±0.02 | 0.74±0.01 | 0.97±0.00 | 0.93±0.13 | 0.67±0.02 | 0.80±0.04 |
| **CBR Small** | 0.78±0.02 | 0.72±0.01 | 0.96±0.0 | 0.98±0.0 | 0.67±0.02 | 0.80±0.03 |
| **CBR LargeW** | 0.75±0.03 | 0.74±0.01 | 0.97±0.01 | 0.96±0.02 | 0.67±0.01 | 0.80±0.04 |
| **CBR LargeT** | 0.74±0.02 | 0.67±0.0 | 0.96±0.00 | 0.96±0.02 | 0.64±0.01 | 0.79±0.01 |
| **VGG16** | 0.78±0.04 | 0.50±0.00 | 0.98±0.01 | 0.78±0.26 | 0.59±0.01 | 0.82±0.01 |
| **Inception-v3** | 0.69±0.03 | 0.60±0.06 | 0.90±0.03 | 0.96±0.01 | 0.61±0.01 | 0.77±0.02 |
| **ResNet50** | 0.69±0.02 | 0.61±0.08 | 0.60±0.20 | 0.93±0.03 | 0.57±0.01 | 0.73±0.02 |
| | *Fine-tuning of pre-trained models.* | | | | | |
| **VGG16** | **0.82±0.02** | **0.84±0.0** | **0.99±0.0** | **0.98±0.0** | **0.77±0.03** | 0.88±0.01 |
| **Inception-v3** | 0.73±0.02 | 0.78±0.01 | 0.98±0.00 | 0.98±0.0 | 0.72±0.01 | 0.89±0.0 |
| **ResNet50** | 0.62±0.02 | 0.82±0.01 | 0.98±0.01 | 0.98±0.0 | 0.54±0.01 | **0.90±0.01** |
| | *Feature extractions from pre-trained models.* | | | | | |
| **VGG16** | 0.77±0.01 | 0.72±0.02 | 0.98±0.01 | 0.93±0.00 | 0.64±0.01 | 0.78±0.00 |
| **Inception-v3** | 0.55±0.01 | 0.50±0.01 | 0.52±0.02 | 0.67±0.04 | 0.54±0.0 | 0.55±0.01 |
| **ResNet50** | 0.50±0.00 | 0.50±0.00 | 0.50±0.00 | 0.50±0.01 | 0.50±0.00 | 0.50±0.00 |

FIGURE 3.5 – Résultats de l'état de l'art de la classification de défaut[11].

## 3.3 Détection d'objet pour l'inspection de défaut

Le problème de la détection d'objet ne consiste pas seulement de classer les objets à l'intérieur de l'image, mais aussi de les localiser à l'intérieur de petites boîtes appelées boites englobantes. La même boite englobante peut avoir plusieurs étiquettes dans la détection d'objet multi-étiquette, ce qui est notre cas d'étude. Le processus de base de la détection d'objet tel qu'il est décrit par W. Hechun et al. [31] est composé de quatre étapes principales (figure 3.6) :

- ***Regional proposal*** **:** l'étape qui prend le plus du temps. Il s'agit de déterminer les localisations potentielles de l'objet (appelées aussi régions d'intérêts, en anglais appelées *region proposals or Regions Of Interest ROI*) dans l'image.
- ***Feature extraction*** **:** extraire des caractéristiques dans l'image permettant d'identifier la classe d'objet visée.
- ***Regional Classification*** **:** consiste à prendre le vecteur caractéristique d'une région d'intérêt candidate comme entrée du classificateur pour prédire la catégorie d'objet de celle-ci.

La dernière étape (*Bounding-Box Regression*) est une étape post-traitement de la détection d'objet. Elle consiste à entraîner un modèle de régression linéaire sur les région d'intérêts ou les petites boites englobantes pré-définis déjà (appelés *anchor boxes*), pour prédire les boites englobantes de la vérité de terrain (*ground truth*).


FIGURE 3.6 – Processus de base de détection d'objet [31].

L'une des problématiques les plus importantes aussi de la détection d'objet, est la mise en place de méthodes simples et robustes dont le choix dépend de nos besoins, à savoir : la vitesse (appelée aussi temps d'inférence) et l'exactitude (accuracy). Il faut savoir qu'il n'existe pas, à l'heure actuelle, des algorithmes aboutis s'adaptant à n'importe quelle situation ! Au cours des deux dernières décennies, il est largement admis que les algorithmes de détection d'objet sont généralement passés par deux périodes historiques : **période de détection d'objet traditionnelle (avant 2014)**, et **période de détection d'objet basée sur l'apprentissage profond (après 2014)**, comme indiqué dans la figure 3.7 [99].

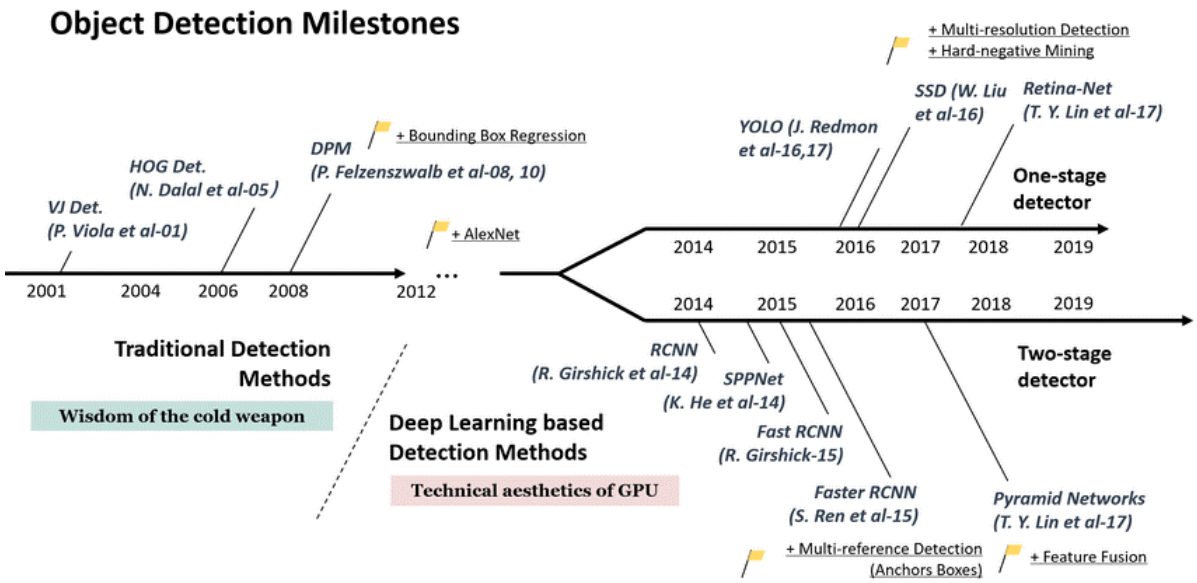


FIGURE 3.7 – Chronologie des méthodes de détection d'objets [99].

La première période des méthodes n'a pas connu beaucoup du succès et elle fut lente en terme d'évolution en raison du manque de représentation d'image efficace à cette époque ainsi que les ressources informatiques limitées. Les méthodes de la deuxième période (après 2014) sont divisées en deux catégories d'approches, celles de *two-stage detector*, détecteurs à deux étages, le plus représentatif est Faster R-CNN (Region-Based

Convolutional Neural Network). L'autre est *one-stage detector*, détecteurs à un seul étage tel que YOLO (*You Only Look Once*) et SSD (*Single Shot MultiBox Detector*) [42].

Dans l'approche de la détection d'objet à deux étages , le premier étage génère l'ensemble des régions d'intérêt (*Region Proposals*) où l'objet est susceptible de se trouver et dans le deuxième étage, ces régions sont passées pour l'extraction des caractéristiques par un extracteur de caractéristiques (feature extractor) spécifique avant qu'elles soient classifiées par un classificateur (SVM par exemple) et détectées par des boites englobantes en utilisant la technique de *bounding box regression* (décrite précédemment). La figure 3.8 montre un exemple de modèle de l'approche à deux étages R-CNN qui est le premier détecteur d'objet basé sur l'apprentissage profond.

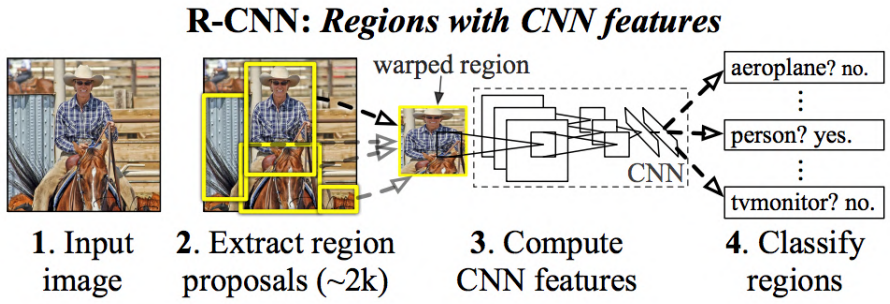


FIGURE 3.8 – L'architecture de détecteur d'objet R-CNN [31].

Avec l'approche à un seul étage, l'image est découpée en cellules et le modèle utilise un scan par cellule pour prédire la probabilité de la classe qui peut contenir chaque cellule. Pour prédire la boite englobante de la classe, le modèle utilise la technique de *bounding box regression* aussi avec des *anchor boxes* déjà prédéfinis. Toutefois, on peut obtenir trop de boites englobantes redondantes, pour cela le modèle utilise une technique appelée NMS (*Non-Maximal Suppression*) qui consiste justement à ne garder que la boite englobante qui couvre le plus l'objet. La figure 3.9 montre un exemple de modèle de l'approche à un seul étage qui est YOLO.

Les détecteurs à deux étages ont une grande précision de localisation et de reconnaissance d'objet, tandis que les détecteurs à un seul étage atteignent une grande vitesse d'inférence. La tendance actuelle est la conception de détecteurs d'objet à un seul étage basés sur la technique *anchor-free boxes* qui n'utilise pas des *anchor boxes* prédefinis pour prédire les boites englobantes des objets. Le modèle connu le plus récent qui utilise cette technique est YOLOX [26].

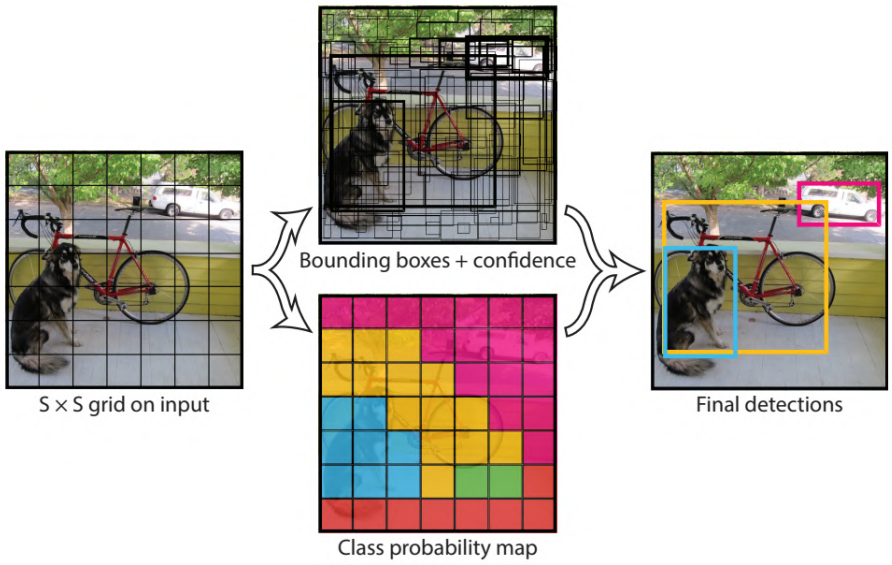


FIGURE 3.9 – Exemple d'une détection avec le modèle YOLO de l'approche à un seul étage [64].

Dans notre cas d'étude qui est la détection de défauts multi-classe multi-étiquette, l'approche que nous présenterons dans ce qui suit a été décrite dans notre article soumis. Cette approche peut être utilisée dans un cadre plus large qui est le problème de détection et de classification des objets qui constituent des anomalies dans les images. En effet, un défaut est une anomalie, et dans ce types de problèmes, le but est généralement d'améliorer la métrique *recall*. En testant les modèles de détection d'objet existants dans la littérature pour les défauts, nous avons remarqué qu'ils donnent de bons résultats en termes de précision mais pas en termes de rappel. Nous avons donc besoin d'un outil capable de générer des régions d'intérêt candidates pour augmenter la détection des défauts. Pour cela, il existe un outil qui convient parfaitement, qui est la saillance visuelle.

Notre idée alors est d'améliorer la détection de défauts en combinant la détection d'objet avec la saillance visuelle. Ceci est motivé par le fait que quel que soit son type, un défaut dans un pont en béton, est généralement caractérisé par une discontinuité de surface locale qui peut être mieux capturé en analysant la saillance locale que d'utiliser simplement la détection d'objet ordinaire. Ainsi, combiner la saillance et la détection d'objet est intuitivement attirant pour booster la détection de défauts tel qu'il est illustré

en bas de Fig. 3.10. Pour cela, nous concevons un pipeline comme indiqué en haut de Fig. 3.10.

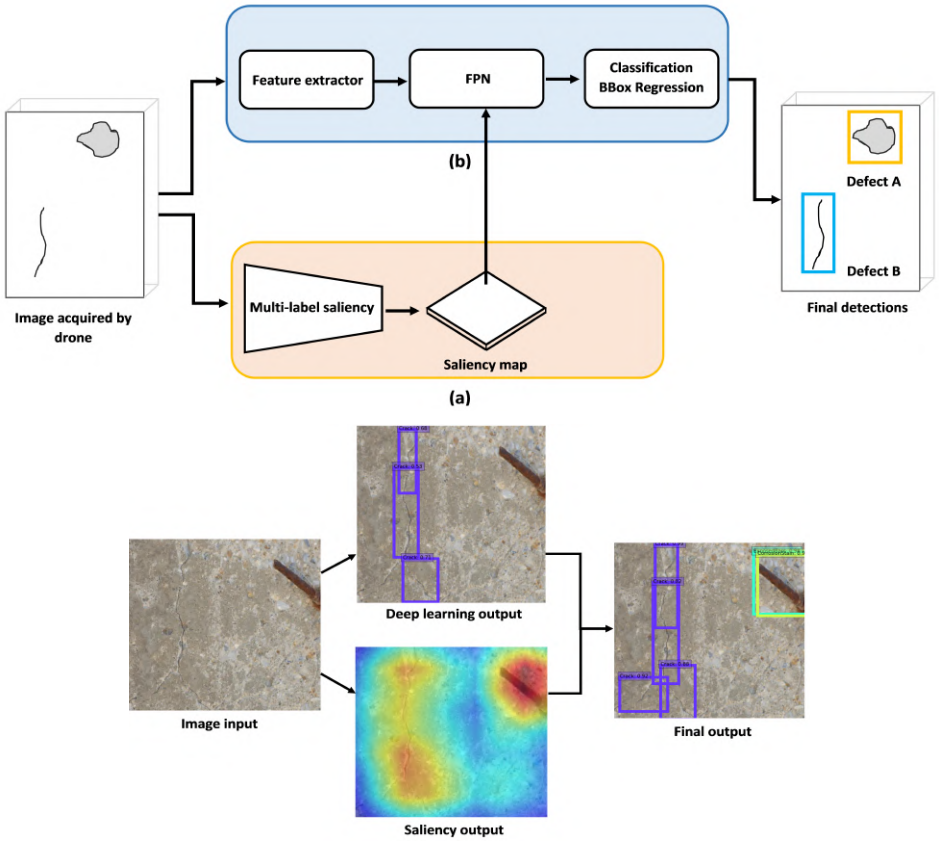


FIGURE 3.10 – Le pipeline du notre méthode proposée SMDD-Net pour la détection de défauts dans les ponts en béton. En haut : *(a) Saliency computation for the attention module*, et *(b) multi-label one-stage concrete defect detection module.* En bas : Un exemple illustrant les avantages du notre méthode pour la détection de défaut.

Notre méthode appelée *Saliency-based Multi-label Defect Detector (SMDD-Net)* combine deux modules : a) Le module d'attention (**the attention module**), et b) le module de détection (**the multi-label one-stage concrete defect detection module**), en utilisant une pyramide de caractéristiques appelée Feature Pyramid Network (FPN) inspiré par le détecteur à seul étage RetinaNet[53]. Le premier module a été conçu pour renforcer la représentation des caractéristiques en attirant l'attention sur les parties de l'image contenant des discontinuités locales. En d'autre termes, la carte de saillance finale générée par ce module en analysant la saillance locale et globale, permettra d'identifier

des régions d'intérêts candidates dans l'image qui peuvent éventuellement contenir des objets de classe (défauts). Le deuxième module consiste à utiliser un détecteur d'objet à un seul étage pour scanner les régions d'intérêts extraites par le premier module afin de détecter et classer les défauts en classes et boites englobantes. La combinaison des deux modules est réalisée à travers un ensemble d'opérations comprenant *max-pooling*, *Upsampling*, *multiplication* et *skip connections*. En utilisant le transfert d'apprentissage avec le *fine-tuning*, le détecteur d'objet choisi dans le deuxième module est entraîné sur le jeu de données CODEBRIM (*COncrete DEfect BRidge IMage Dataset* [58]) qui sera détaillé dans le prochain chapitre. Dans ce qui suit, nous décrivons chaque module séparément en donnant un exemple d'implantation de chaque module comme indiqué dans la figure 3.11 où la méthode Grad-CAM [71] est utilisée pour le module de saillance, et le modèle RetinaNet [53] basé sur ResNet50 comme extracteur de caractéristiques est utilisé pour le module de détection.

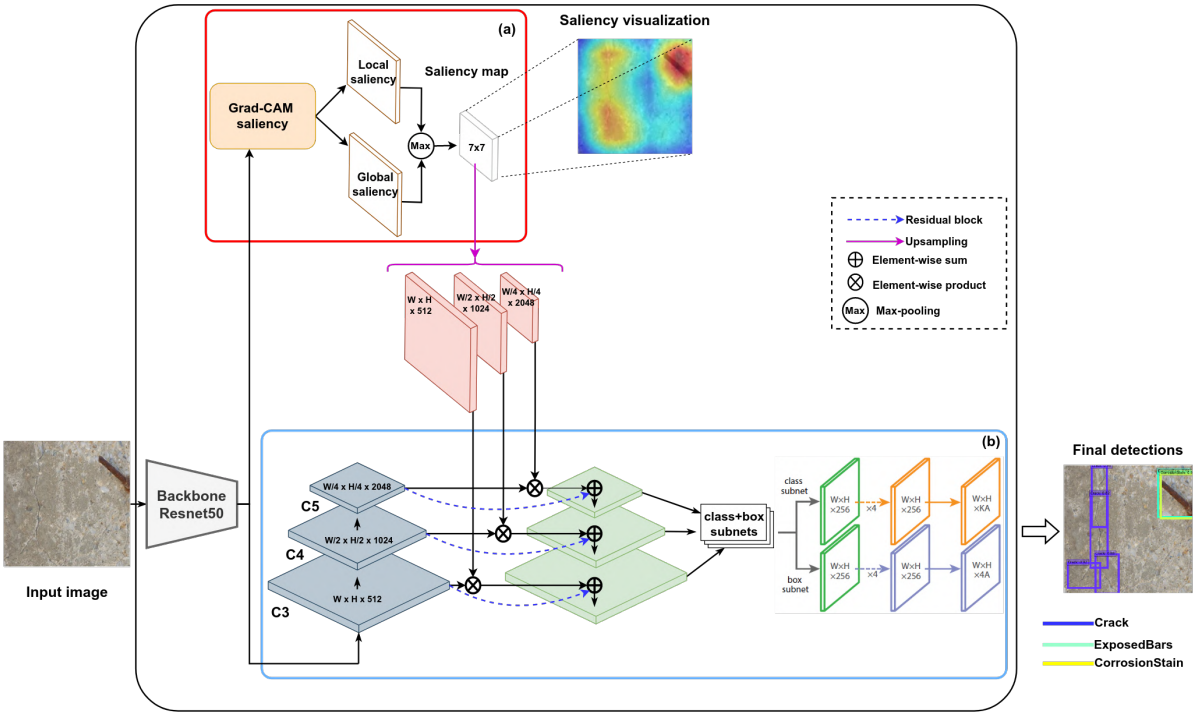


FIGURE 3.11 – Exemple d'implémentation du notre méthode SMDD-Net proposée pour la détection de défaut. (a) : *attention module* qui utilise la méthode Grad-CAM pour le calcul de la carte de saillance ; (b) : *detection module* qui utilise le modèle RetinaNet basé sur ResNet50 comme extracteur de caractéristiques.

– **Saliency for defect region proposals :** la saillance permet d'identifier les régions (les pixels) qui influencent le plus sur la fonction d'activation utilisée dans la sortie de la classification par un réseau de neurones. Bien qu'une région saillante ne constitue pas toujours un défaut, la plupart des défauts présentent un certain degré de saillance. Ainsi, utiliser la saillance pour guider la détection peut être très utile pour réduire les faux négatifs qui est un problème courant dans la détection de défauts multi-classe [38]. Ce module est entraînable, et peut être implémenté par n'importe quelle méthode de la saillance visuelle. Dans notre cas, la méthode SMDD-Net intègre une version locale de Grad-CAM pour mettre en évidence les régions de l'image qui sont responsables d'activation de la classe de défaut. À cet effet, l'image d'entrée est subdivisée en plusieurs parties chevauchantes, où une carte de saillance est calculée séparément pour chaque partie (voir la Figure 3.11.b pour une illustration). Afin de maximiser le taux de détection de défaut, la saillance maximale est conservée dans chaque partie de la division pour produire la carte de saillance finale de l'image d'entrée.

  De manière plus formelle, supposons que nous ayons $K$ classes de défauts et une image d'entrée $I$, qui peut contenir une ou plusieurs instances de ces classes de défaut. Nous avons d'abord subdivisé l'image en $n$ parties qui se chevauchent $P_1, ..., P_n$, tel qu'il est montré dans Figure 3.12.b. La salience induite sur la partie $P_i$ de la $k$-ème carte de caractéristiques après l'activation de la classe de défauts $y_c$ est donnée par $\frac{\partial y_i^c}{\partial A_i^k}$, et la combinaison pondérée des cartes d'activation avant est donnée par :

$$L_i^c = ReLu(\sum_k \alpha_{ik}^c A_i^k) \quad \text{où} \quad \alpha_{ik} = \frac{1}{Z} \sum_p \sum_q \frac{\partial y_i^c}{\partial A_i^k(p,q)} \tag{3.1}$$

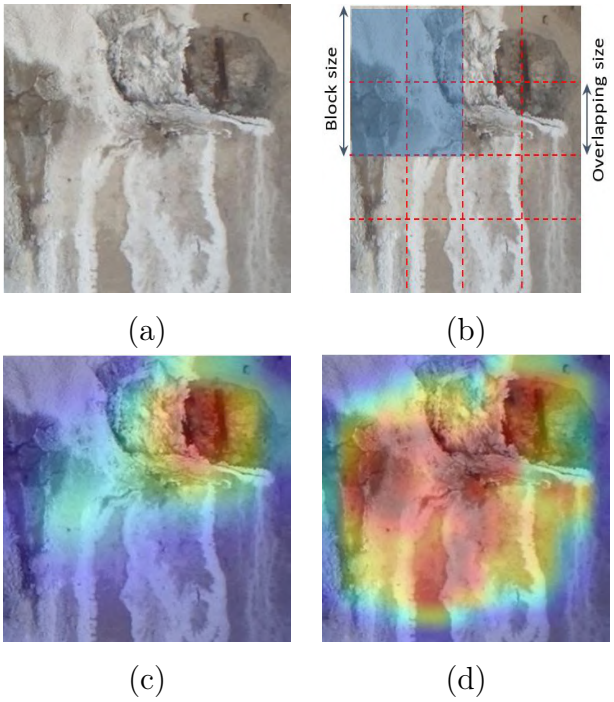


FIGURE 3.12 – Illustration du module d'extraction de la saillance pour la détection de défaut : (a) image d'entrée, (b) subdivision de l'image en blocs chevauchants, (c) extraction globale de la saillance et (d) extraction locale de la saillance.

où $\alpha_{ik}$ représentent les poids d'importance des neurones et $Z$ est la taille spatiale de la carte de caractéristiques (p et q représentent les coordonnées spatiales des cartes de caractéristiques). De plus, pour tenir compte de la relation de corrélation entre les classes de défauts (par exemple, l'oxydation causée par une barre exposée, elle-même causée par l'écaillage), nous avons créé pour l'étiquette de classe $c$ un sous-ensemble d'étiquettes $Q_c$ qui pourraient survenir en conséquence de $y^c$ (par exemple, si $c = \textit{spalling}$, alors $Q_c = \{\textit{exposed bar}, \textit{oxidation}\}$). La carte d'activation finale produite pour la partie $P_i$ est alors donnée par la formule suivante :

$$S_i = ReLu\left(\max\left(\sum_k \alpha_{ik}^c A_i^k, Avg_{c' \in Q_c} \sum_k \alpha_{ik}^{c'} A_i^k\right)\right) \tag{3.2}$$

Enfin, la carte de saillance locale complète $S_{local}$ pour toute l'image $I$ est obtenue en assemblant les cartes de saillance locale $S_i$ générées pour les parties $P_i$ dans leurs positions correspondantes dans l'image $I$. Pour les parties qui se chevauchent entre deux blocs, disons $Pi$ et $Pj$, nous avons pris la saillance maximale entre

$Si$ et $Sj$ pour la zone de chevauchement. De plus, pour améliorer globalement la saillance de la représentation des caractéristiques, une autre carte de saillance $S_{global}$ a été générée en envoyant toute l'image à Grad-CAM. La figure 3.12 illustre le module d'extraction de saillance sur une image contenant plusieurs défauts se chevauchent sur l'ensemble de l'image. Alors que la saillance globale a identifié le défaut le plus saillant sur la droite (barre exposée + écaillage), la saillance locale a identifié l'efflorescence occupant une grande partie de l'image et l'écaillage sur la gauche.

– **Multi-Label One-Stage Defect Detection :**

Comme on l'a vu précédemment, les modèles de détection d'objet de l'approche à un seul étage sont efficaces en terme de temps d'inférence au prix de l'exactitude contrairement à ceux de l'approche à deux étages. Ici, nous essayons de combiner les avantages des deux approches en réduisant simplement la zone de recherche du détecteur à un seul étage, permettant ainsi une détection de défauts rapide et précise à la fois. Pour la plupart des problèmes de détection d'objet, une boite englobante est étiquetée par un seul label. Dans notre problème de détection de défaut, ceci n'est pas toujours vérifié puisque plusieurs défauts peuvent se chevaucher. Par exemple, la barre exposée est souvent liée à la corrosion et la fissure peut être aussi liée à l'efflorescence. De ce fait, nous avons choisi d'entraîner le modèle à un seul étage RetinaNet en activant son option multi-étiquette, permettant ainsi d'attribuer plus d'une étiquette à une boite englobante. RetinaNet est conçu pour résoudre le problème de déséquilibre des données entre les classes d'avant-plan et d'arrière-plan (foreground-background) qui peut empêcher les détecteurs d'objets à un étage d'atteindre de meilleures performances. Ceci est fait en utilisant *Focall Loss (FL)* comme fonction de perte pour la classification durent l'entraînement, au lieu de la fonction standard *Cross Entropy loss (CE)*, afin de focaliser l'apprentissage sur les exemples négatifs qui sont difficiles à détecter [53]. La fonction de perte FL est définie à partir de la fonction CE en introduisant les paramètres de pondération $\alpha$ et $\gamma$ ainsi : supposant $p_t$ est la probabilité estimée par le modèle pour la classe positive, alors FL est définie comme suit :

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t). \tag{3.3}$$

Pour tenir compte de l'aspect multi-étiquette pour la détection de défaut, nous avons augmenté les annotations en créant des copies des boîtes englobantes contenant plusieurs étiquettes de défaut. Cela permettra de régresser différents candidats dans les régions d'intérêt pour cibler la même boîte englobante. L'effet final de ceci est de produire des boîtes englobantes étroitement superposées qui peuvent être étiquetées différemment les unes des autres. Plus formellement, la fonction de perte $FL$ pour une boîte englobante donnée ayant un ensemble $\mathcal{L} = \{c_1, ...c_n\}$ d'étiquettes est donnée par :

$$FL(p_{c_1}, .., p_{c_n}) = -\sum_{c_i \in \mathcal{L}} \alpha_{c_i}(1 - p_{c_i})^{\gamma} log(p_{c_i}) \tag{3.4}$$

Dans toutes les expériences de cette étude, nous avons fixé $\alpha_{c_i} = 0.25$ et $\gamma = 2$. Pour la régression de boîte englobante (*bounding box regression*), nous avons utilisé la fonction de perte $L1$ définie dans [27], qui est moins sensible aux valeurs aberrantes que $L2$. La fonction de perte d'entraînement finale est la somme de $FL$ et $L1$.

Comme illustré dans l'exemple d'implémentation de la figure 3.11, en utilisant le modèle RetinaNet pour ce module basé sur le modèle pré-entraîné ResNet50 sur CODEBRIM comme extracteur de caractéristiques, des caractéristiques multi-échelles $F3, F4, F5$ des trois dernières couches de convolution de ResNet50, sont extraites, puis combinées avec la carte de saillance en utilisant la formule suivante :

$$F_i' = F_i \oplus (F_i \otimes MAX(\uparrow S_{local}, \uparrow S_{global})), i = 3, 4, 5 \tag{3.5}$$

où $\uparrow$ représente une opération de *Upsampling*, $\oplus$ représente une opération de *skip connection*, et $\otimes$ représente une multiplication. Les nouvelles caractéristiques $F_i$ sont ensuite transmises au *Feature Pyramid Network (FPN)* avant d'être classées en classes dans des boîtes englobantes via les sous-réseaux de neurones en tête du module : sous-réseau de neurones de classification (*classification subnet*) et sous-réseau de neurones de régression de boîte (*box regression subnet*).

## 3.4 Segmentation d'objet pour l'inspection de défaut

Parmi les trois problèmes de recherche liés aux défauts que nous traitons (classification, détection et segmentation), celui de la segmentation de défauts est le plus compliqué ! Le problème vient du fait qu'il y a des classes qui se chevauchent (problème multi-classe multi-étiquette). La question est : comment peut-on amener notre modèle choisi pour la segmentation sémantique, à étiqueter un pixel donné par plusieurs étiquettes (classes) ? Les options qui s'offrent à nous dans ce cas sont :

- **Proposition 1 : utiliser la segmentation d'instance**

  La segmentation d'instance traite plusieurs objets de la même classe comme des instances individuelles distinctes. Ce qui veut dire qu'on peut créer plusieurs instances du même catégorie d'objet, avec la même étiquette pour chaque instance. Toutefois, si on prend par exemple le défaut "barres exposées" qui doit être traité en principe comme une seule entité sémantique, la segmentation d'instance va créer une instance pour chaque petite barre à l'intérieur des barres exposées, ce qui va trop compliquer la représentation des objets dans l'image.

- **Proposition 2 : combiner la segmentation sémantique avec la détection d'objet**

  Cela consiste à détecter en premier lieu la boîte englobante de l'objet avec sa classe en utilisant un modèle de détection d'objet comme YOLO, ensuite créer le masque de segmentation qui trace les contours pixels de l'objet à l'intérieur de la boîte englobante tel qu'il est illustré par C. Zhang et al. [93] dans la figure 3.13.

- **Proposition 3 : utiliser la segmentation sémantique binaire pour chaque défaut**

  On aura autant de modèles de segmentation que de nombre de classes de défauts et chaque modèle sera un classificateur de pixel binaire.

- **Proposition 4 : adapter la segmentation sémantique au problème multi-étiquette d'une façon découplée**

  Supposons qu'on a un défaut qui doit être étiqueté par deux labels : ExposedBars (barres exposées) et CorrosionStain (une corrosion) comme illustré dans la figure 3.14. Quel que soit le modèle de segmentation sémantique qu'on choisira, il va prendre une seule image avec une seule étiquette à la fois, car il s'agit d'un

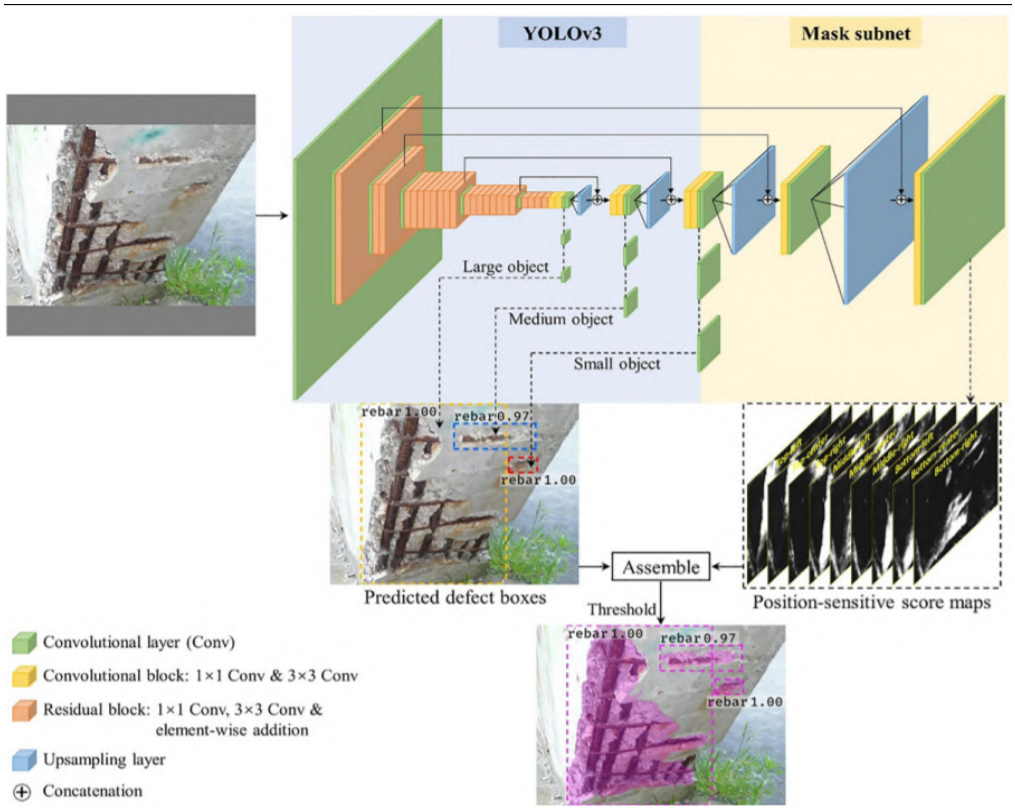


FIGURE 3.13 – Combinaison de YOLOv3 avec Mask subnet pour la segmentation sémantique de défauts [93].

apprentissage supervisé. Pour cela, on va créer un masque de segmentation binaire pour chacune des classes ExposedBars et CorrosionStain séparément, et lors de l'entraînement, on va donner au modèle la même image deux fois : une étiquetée avec ExposedBars et l'autre avec CorrosionStain. Pour la sortie, on va utiliser la fonction d'activation sigmoid comme dans la classification pour pouvoir activer plusieurs masques de classe en même temps. Donc on va donner au modèle la même image autant de fois que le nombre de défauts qui se chevauchent dans l'image. Le processus est illustré dans la figure 3.14.

Nous avons opté alors pour les deux dernières propositions (proposition 3 et proposition 4). Nous avons réussi à implémenter la proposition 3 pour le moment. À ce stade, nous étions obligés de créer des masques de segmentation avec lesquels entraîner nos modèles, nous avons alors créé 250 masques par nous même ! (c'était un projet réalisé en binôme dans le cours *INF6243 - Techniques d'apprentissage*). Le modèle de base que nous avons choisi à tester est le modèle encodeur-décodeur U-Net (Figure 3.15) avec quatre

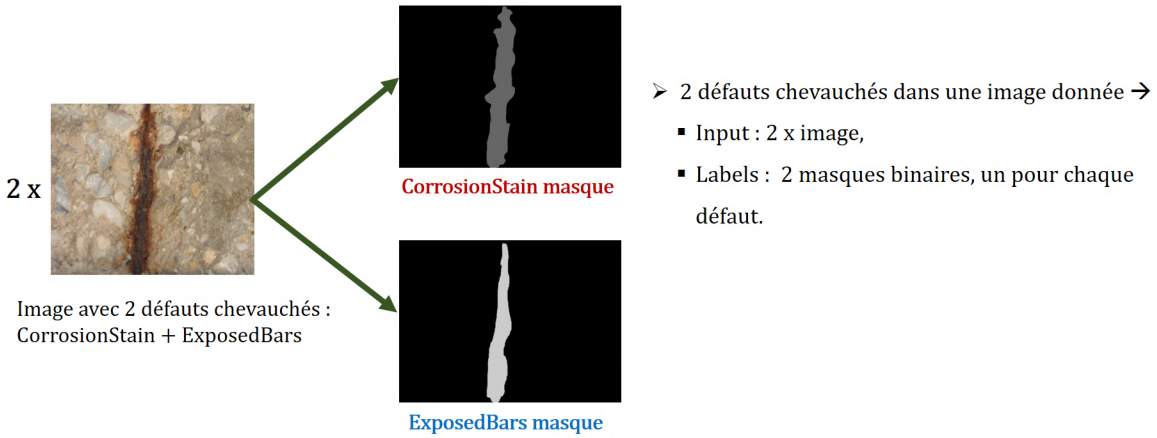


FIGURE 3.14 – Adaptation de la segmentation sémantique au problème multi-étiquette pour l'inspection de défaut.

configurations différentes : ResNet50 comme encodeur, InceptionV3 comme encodeur, VGG16 comme encodeur et MobilNetV2 comme encodeur. L'apprentissage des poids de l'encodeur de chaque configuration est transféré à partir de dataset ImageNet. À noter que nous avons testé d'autres modèles aussi (FPN et LinkNet), mais nous présentons que les résultats de U-Net dans ce rapport. Pour évaluer la performance des classificateurs testés, nous utilisons la métrique Pixel Accuracy définie auparavant.

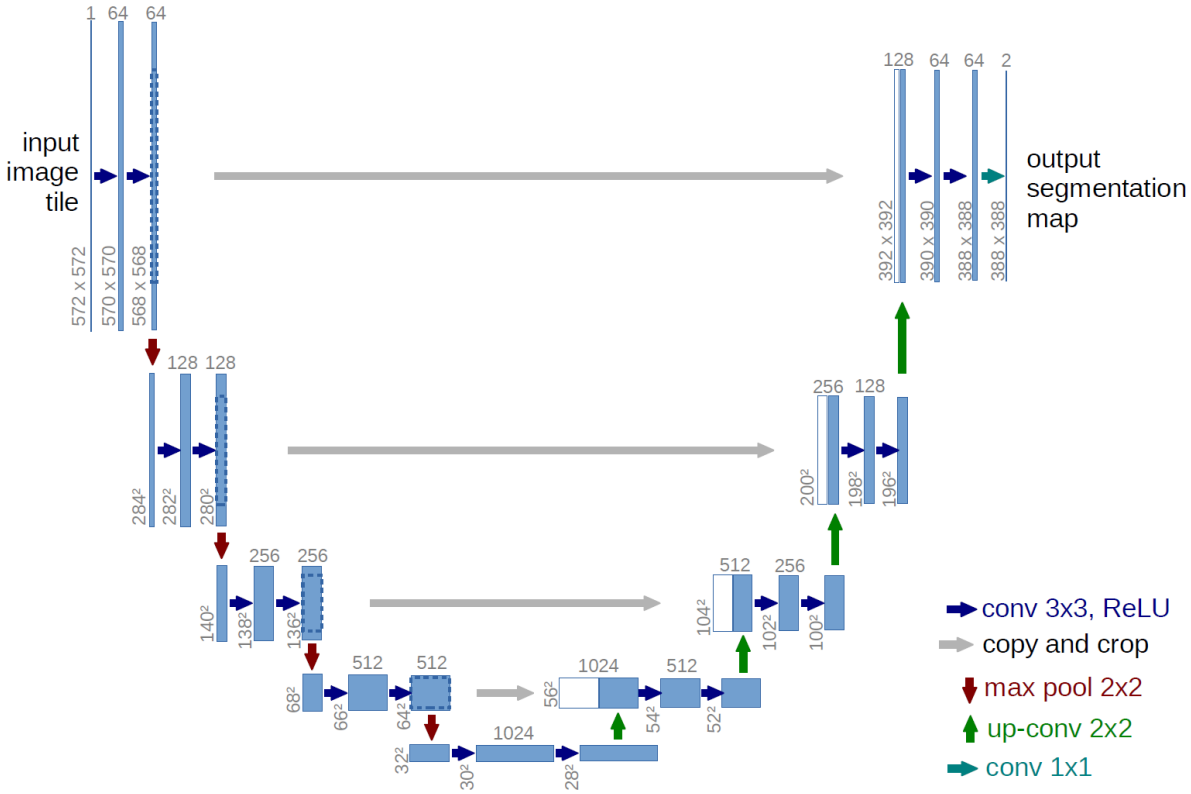


FIGURE 3.15 – Architecture du modèle encodeur-décodeur U-Net [68].

## 3.5 Conclusion

Nous avons, à travers ce chapitre, franchi l'une des étapes les plus importantes du notre projet. En effet, nous avons décrit notre approche proposée pour résoudre la problématique, qui traite trois problèmes de recherche lié aux défauts : la classification, la détection et segmentation de défauts multi-classe multi-étiquette dans les ponts en béton. En particulier, nous avons détaillé la nouvelle méthode proposée SMDD-Net pour résoudre le problème de détection de défaut. Dans le prochain chapitre, nous présenterons les tests que nous avons effectués et les résultats obtenus dans chaque cas.

# Chapitre 4

# Collecte de données, tests et résultats

Après avoir présenté notre approche proposée dans le chapitre précèdent, nous présentons dans ce chapitre les tests menés et les résultats obtenus. Pour cela, nous commençons par décrire les jeux de données que nous avons explorés pour les trois problèmes de la classification, la détection et la segmentation de défaut. Ensuite, nous discuterons les résultats obtenus avec les modèles testés dans chaque cas.

## 4.1 Les jeux de données explorés

Les jeux de données publiques exploités pour la classification, la détection et la segmentation de défauts multi-classe multi-étiquette sont : CODEBRIM (COncrete DEfect BRidge IMage Dataset [58]) et MCDS (Multi-Classifier DataSet for reinforced concrete bridge defects [35]). CODEBRIM est un jeu de données multi-étiquette qui est composé de six classes : Background (2490), et cinq classes de défauts : *Crack* (2507), *Spallation* (1898), *ExposedBars* (1507), *Efflorescence* (833) et *CorrosionStain* (1559). Les images ont été acquises à haute résolution à l'aide d'un drone, puis redimensionnées pour s'adapter à la résolution d'entrée requise par chaque modèle à tester dans notre cas. le jeu de données peut être utilisé pour la classification ainsi que la détection.

MCDS est un jeu de données multi-étiquette défini sous forme d'une hiérarchie de classificateurs qui contient en tout dix classes : *crack* (789), *efflorescence* (311), *scaling* (168), *spalling* (427), *general defects* (264), *no defects* (452), *exposed reinforcement* (223), *no exposed reinforcement* (203), *rust straining* (355), et *no rust straining* (415). Il est dédié seulement à la classification. La figure 4.1 montre les jeux de données publiques

existants qui sont dédiés à l'inspection des ponts. On voit qu'il y a que CODEBRIM et MCDS qui sont des jeux de données multi-étiquette, les autres sont binaires [11].

**Overview of (bridge) Visual Inspection Datasets.**

| Dataset | Instances | Classes | Problem |
|---|---|---|---|
| CDS [18] | 1,027 | 2 | Binary |
| SDNETv1 [29] | 13,620 | 2 | Binary |
| BCD [49] | 5,390 | 2 | Binary |
| ICCD [26] | 60,010 | 2 | Binary |
| MCDS [20] | 2,411 | 10 | Multi-label |
| CODEBRIM [32] | 8,304 | 6 | Multi-label |

FIGURE 4.1 – Ensemble des jeux de données publiques dédiés à l'inspection des ponts [11].

On choisit alors CODEBRIM comme notre jeu de données de base pour explorer les problèmes de classification, détection et segmentation. On utilise aussi le jeu de données MCDS pour enrichir CODEBRIM dans le problème de la classification. La figure 4.2 illustre des exemples de classes de jeu de données CODEBRIM, et la figure 4.3 montre la matrice de corrélation entre ces classes. On observe une corrélation de 0.7 entre Spallation et ExposedBars. On peut donc conclure que Spallation est une classe générique de ExposedBars. Il peut y avoir d'autres relations entre ces deux classes comme la relation de causalité. Une question intéressante alors qu'on peut poser comme une future direction de recherche pour ce travail, est : comment implémenter ces deux relations de corrélation et de causalité dans un seul modèle de classification, de détection ou de segmentation ?

Pour la segmentation, il n'existe pas un jeu de données multi-étiquette publique. Toutefois, on trouve quelques jeux de données binaires qui traitent une seule classe comme les fissures. Pour commencer avec le problème de segmentation sémantique, on teste donc ces jeux de données binaires. Crack Segmentation Dataset [55] est un benchmark qui combine plusieurs jeux de données de défauts crack (fissure)[19, 94, 73, 6, 98], construit en utilisant différentes structures (Ponts, murs, routes...etc.)

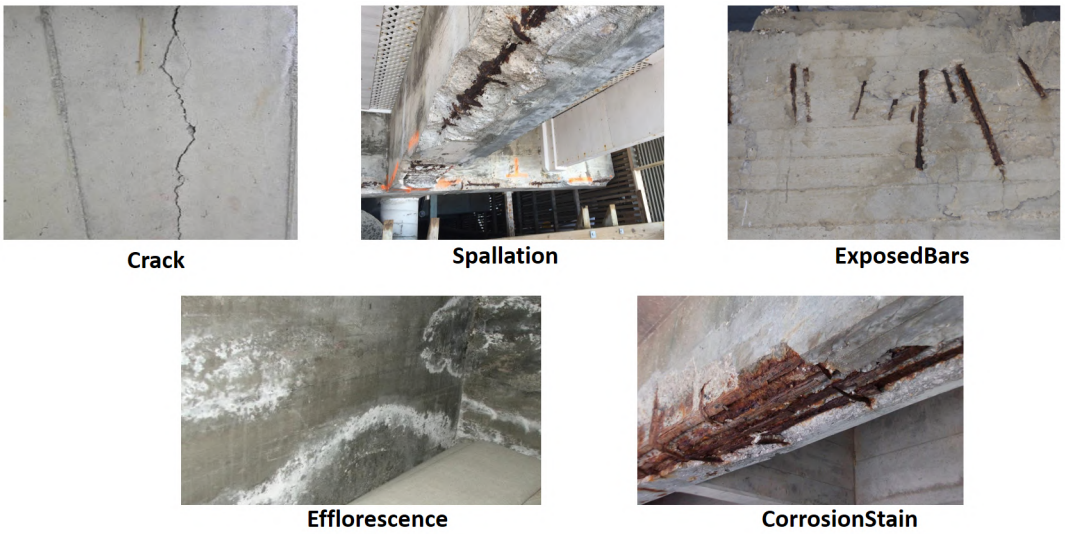


FIGURE 4.2 – Les classes du jeu de données CODEBRIM[58].

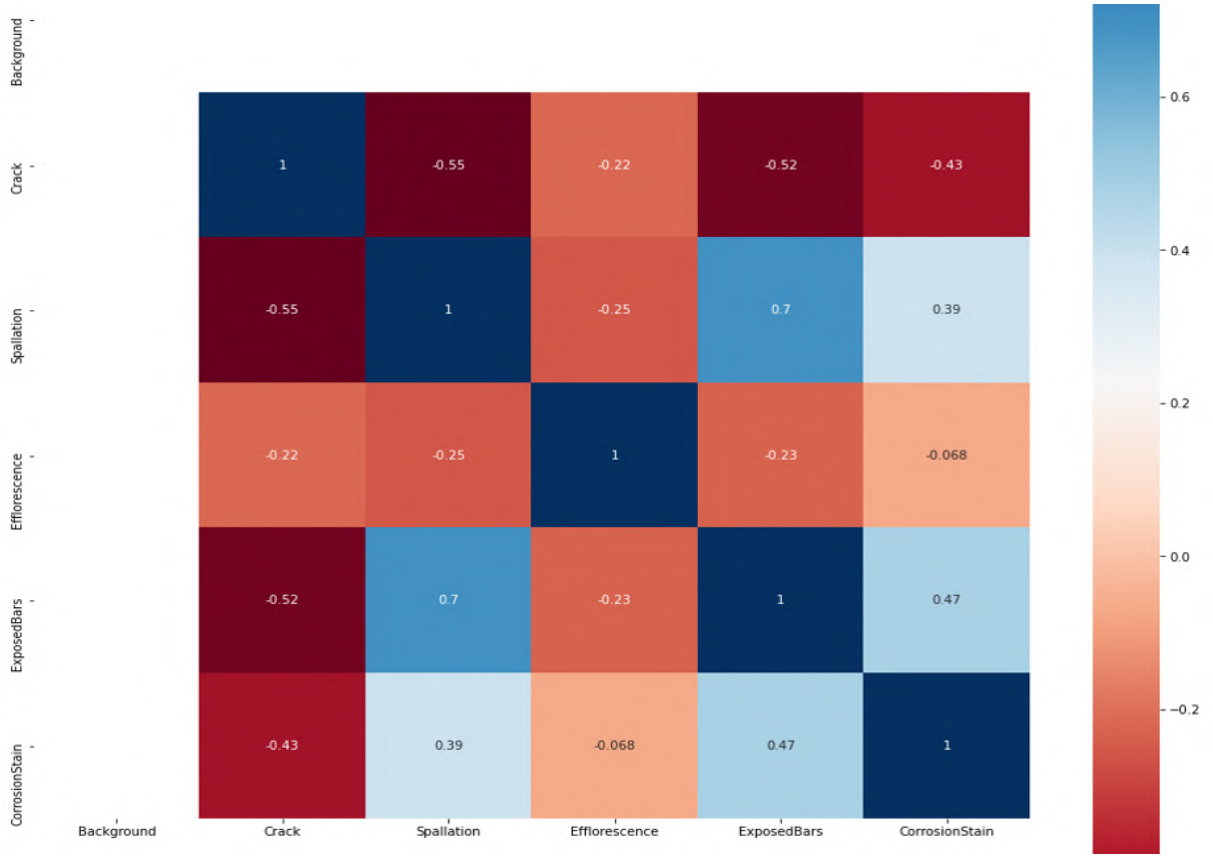


FIGURE 4.3 – Matrice de corrélation entre les classes de CODEBRIM.

## 4.2 Résultats et discussion

Dans cette section nous présentons les tests menés et nous discutons des résultats obtenus. Pour nos expériences, nous avons utilisé la plateforme Google Colab Pro+, qui fournit 52 Go de RAM, 8 cœurs de CPU et un accès prioritaire au GPU P100. Le jeu de données CODEBRIM est divisé en trois parties d'ensembles d'entraînement, de validation et de test avec un ratio approximatif de 70%, 20% et 10%.

### 4.2.1 Classification

#### Métriques d'évaluation

Pour évaluer la performance des modèles de classification testés, nous avons utilisé la métrique AUC-ROC décrite précédemment. Cette métrique est l'une des métriques d'évaluation les plus robustes pour les tâches de classification présentant un problème de déséquilibre des données, ce qui est le cas du jeu de données CODEBRIM. La métrique AUC-ROC mesure la capacité d'un classifieur binaire à distinguer les classes positives et négatives à travers tous les seuils possibles contrairement à d'autres métriques telles que l'exactitude, la précision et le rappel qui dépendent d'un seuil de décision fixe. Cela signifie qu'elle peut évaluer la capacité du classifieur à identifier la classe minoritaire même lorsque le seuil est défini pour favoriser la classe majoritaire.

#### Discussion des résultats

Les tables 4.1 et 4.2 montrent les résultats des tests de classification menés avec les jeux de données CODEBRIM et MCDS respectivement, dans les deux scénarios : entraîner uniquement la dernière couche des modèles, et entraîner plus de couches inférieures. On remarque que le meilleur résultat obtenu est celui de ResNet50 entraîné sur CODEBRIM avec AUC-ROC=0.96, surpassant ainsi les résultats de l'état de l'art. On observe aussi que les modèles entraînés sur le jeu de données CODEBRIM performent mieux que ceux entraînés sur MCDS. Cela est probablement dû à la meilleure qualité d'image de CODEBRIM, qui dispose d'une résolution maximale de 6000x4000 pixels, par rapport à MCDS.

Nous avons calculé aussi le Rappel (Recall), la Précision et le F-score de chaque classe de ResNet50 (figure 4.4 ). Nous avons ensuite augmenté les classes qui ont un F-score faible (encadrées en rouge) avec les images de jeu de données MCDS. Les résultats des

TABLE 4.1 – Comparaison des résultats de la classification des défauts avec le jeu de données CODEBRIM.

**Scénario 1 : entraîner uniquement la dernière couche de classification.**

| Modèles | Nos résultats | Résultats de l'état de l'art [11] |
|---|---|---|
| VGG16 | 0.88 | 0.78 |
| ResNet50 | 0.91 | 0.50 |
| Inception-v3 | 0.90 | 0.55 |
| EfficientNet-B0 | 0.93 | - |
| **DenseNet-201** | **0.94** | - |
| InceptionResNet-v2 | 0.92 | - |
| ViT | 0.87 | - |

**Scénario 2 : entraîner plus de couches inférieures (*Fine-tuning*).**

| Modèles | Nos résultats | Résultats de l'état de l'art [11] |
|---|---|---|
| VGG16 | 0.88 | 0.88 |
| **ResNet50** | **0.96** | 0.90 |
| Inception-v3 | 0.93 | 0.89 |
| EfficientNet-B0 | 0.95 | - |
| DenseNet-201 | 0.95 | - |
| InceptionResNet-v2 | 0.94 | - |

tests de ResNet50 entraîné sur le jeu de données résultant sont montrés dans la figure 4.5. On voit que la combinaison des deux jeux de données n'a pas amélioré la performance, et en analysant les annotations de chaque jeu de données, nous avons trouvé que certaines classes ont une logique des annotations différentes, ce qui constitue un bruit pour le modèle ResNet50. Par exemple sur la figure 4.6, on voit que pour MCDS, l'étiquette prédite par ResNet50 est : "Crack", "Spallation" et "ExposedBars". Alors que pour CODEBRIM, l'étiquette prédit par le modèle est : "Crack" et "ExposedBars". Cela veut dire que le modèle a été entraîné sur des images annotées d'une logique différente. En effet, comme nous l'avons vu précédemment, MCDS est définie d'une façon hiérarchique : la classe "ExposedBars" par exemple est une classe descendante de "Spallation", ce qui veut dire qu'à chaque fois "ExposedBars" est activée, "Spallation' le sera aussi, ce qui n'est pas toujours le cas pour le jeu de données CODEBRIM !

TABLE 4.2 – Comparaison des résultats de la classification des défauts avec le jeu de données MCDS.

**Scénario 1 : entraîner uniquement la dernière couche de classification.**

| Modèles | Nos résultats | Résultats de l'état de l'art [11] |
|---|---|---|
| VGG16 | 0.68 | 0.64 |
| **ResNet50** | **0.90** | 0.50 |
| Inception-v3 | 0.87 | 0.54 |
| **EfficientNet-B0** | **0.90** | - |
| **DenseNet-201** | **0.90** | - |
| InceptionResNet-v2 | 0.88 | - |
| ViT | 0.85 | - |

**Scénario 2 : entraîner plus de couches inférieures (*Fine-tuning*).**

| Modèles | Nos résultats | Résultats de l'état de l'art [11] |
|---|---|---|
| VGG16 | 0.69 | 0.77 |
| **ResNet50** | **0.93** | 0.54 |
| Inception-v3 | 0.90 | 0.72 |
| EfficientNet-B0 | 0.92 | - |
| **DenseNet-201** | **0.93** | - |
| InceptionResNet-v2 | 0.91 | - |

| | precision | recall | f1-score |
|---|---|---|---|
| Crack | 0.83 | 0.89 | 0.86 |
| Spallation | 0.76 | 0.89 | 0.82 |
| Efflorescence | 0.89 | 0.78 | 0.83 |
| ExposedBars | 0.93 | 0.89 | 0.91 |
| CorrosionStain | 0.81 | 0.85 | 0.83 |
| micro avg | 0.84 | 0.86 | 0.85 |
| macro avg | 0.84 | 0.86 | 0.85 |
| weighted avg | 0.84 | 0.86 | 0.85 |
| samples avg | 0.83 | 0.86 | 0.83 |

FIGURE 4.4 – Résultats des tests de classification de ResNet50 sur CODEBRIM.

| | precision | recall | f1-score |
|---|---|---|---|
| Crack | 0.74 | 0.91 | 0.82 |
| Spallation | 0.77 | 0.85 | 0.80 |
| Efflorescence | 0.92 | 0.77 | 0.84 |
| ExposedBars | 0.94 | 0.80 | 0.87 |
| CorrosionStain | 0.80 | 0.74 | 0.77 |
| micro avg | 0.82 | 0.81 | 0.82 |
| macro avg | 0.83 | 0.81 | 0.82 |
| weighted avg | 0.83 | 0.81 | 0.82 |
| samples avg | 0.83 | 0.82 | 0.81 |

FIGURE 4.5 – Résultats des tests de classification de ResNet50 sur CODEBRIM augmenté avec MCDS.

Label prédit par ResNet50 avec CODEBRIM: [1. 0. 0. 1. 0.]
Label prédit par ResNet50 avec MCDS : [1. 1. 0. 1. 0.]

**NB:** Labels= ["Crack","Spallation","Efflorescence","ExposedBars","CorrosionStain"]

FIGURE 4.6 – Exemple de prédiction par ResNet50 entraîné sur CODEBRIM et MCDS.

### 4.2.2 Détection

#### Métriques d'évaluation

Dans le cas de la détection de défaut, pour évaluer la performance du notre méthode proposée et la comparer à d'autres études, nous avons utilisé la métrique mAP (mean Average Precision) qui est décrite auparavant, avec un seuil de décision égale à 0.5 pour décider si un *bounding box* détecté est un TP ou FP. Nous avons implémenté notre méthode proposée SMDD-Net en utilisant Grad-CAM pour le module d'attention et RetinaNet pour le module de détection. En utilisant l'apprentissage par transfert, ResNet50 est d'abord pré-entraîné sur le jeu de données CODEBRIM, puis utilisé comme extracteur de caractéristiques pour les deux modules. Pour aider à réduire les faux positifs (FP), nous ajoutons 10 % d'images d'arrière-plan (background) sans étiquettes à l'ensemble de données d'entraînement . Lors de l'utilisation de l'inférence avec RetinaNet, pour sélectionner la meilleure boîte englobante parmi les multiples boîtes prédites , nous utilisons la technique *Non-Maximal Suppression* (NMS) avec un seuil égal à 0.4.

#### Discussion des résultats

Pour démontrer l'importance du module d'attention, nous avons effectué une étude d'ablation en utilisant SMDD-Net pour la détection de défauts dans quatre scénarios différents : 1) SMDD-Net sans le module d'attention, 2) SMDD-Net sans la saillance globale, 3) SMDD-Net sans la saillance locale, et 4) SMDD-Net sans les connexions résiduelles. Cela nous a permis de voir les avantages de l'utilisation de la saillance. Les valeurs d'AP obtenues pour chaque classe de défaut, ainsi que la valeur mAP dans le premier scénario, sont présentées dans la table 4.3. Les résultats des autres scénarios sont présentés dans la table 4.4. Selon les résultats des deuxième et troisième scénarios, Les avantages de la saliency locale et globale étaient complémentaires. En d'autres termes, les deux modules contribuent également à la détection. Dans le dernier scénario, où les blocs résiduels ont été supprimés, la précision a considérablement diminué car la combinaison a été effectuée uniquement par multiplication de caractéristiques. La carte de saillance extraite du module d'attention peut contenir des valeurs très petites, qui peuvent détruire la pyramide des caractéristiques via l'opération de multiplication. Cela se reflète dans la diminution de la précision pour ce scénario. En conclusion, tous les composants de la méthode SMDD-Net étaient importants pour atteindre une précision de détection prouvée. La Figure 4.7 montre les résultats de détection sur certaines images

par SMDD-Net. Il convient de noter que, dans certains cas, la détection avait un score très élevé (score = 100%). Notez également que, même si la saillance pointe des fois vers des régions fausses (autres que les défauts), le module de détection n'a produit aucune fausse détection FP dans ces régions la (voir image 1, 3 et 4).

TABLE 4.3 – Résultats d'évaluation de SMDD-Net sur l'ensemble de données de test sans utiliser le module d'attention.

| **Classes** | **AP@0.5** | **mAP@0.5** |
|---|---|---|
| Crack | 0.98 | |
| Spallation | 0.96 | |
| Efflorescence | 0.80 | 0.88 |
| Exposed bars | 0.90 | |
| Corrosion stain | 0.74 | |

TABLE 4.4 – Résultats des tests d'ablation pour SMDD-Net.

| **Scenarios** | **#Param.** | **mAP@0.5** |
|---|---|---|
| SMDD-Net without attention module | 36.5 M | 0.88 |
| SMDD-Net without global saliency | 36.5 M | 0.95 |
| SMDD-Net without local saliency | 36.5 M | 0.93 |
| SMDD-Net without residual block | 36.5 M | 0.46 |
| SMDD-Net | 36.5M | 0.99 |

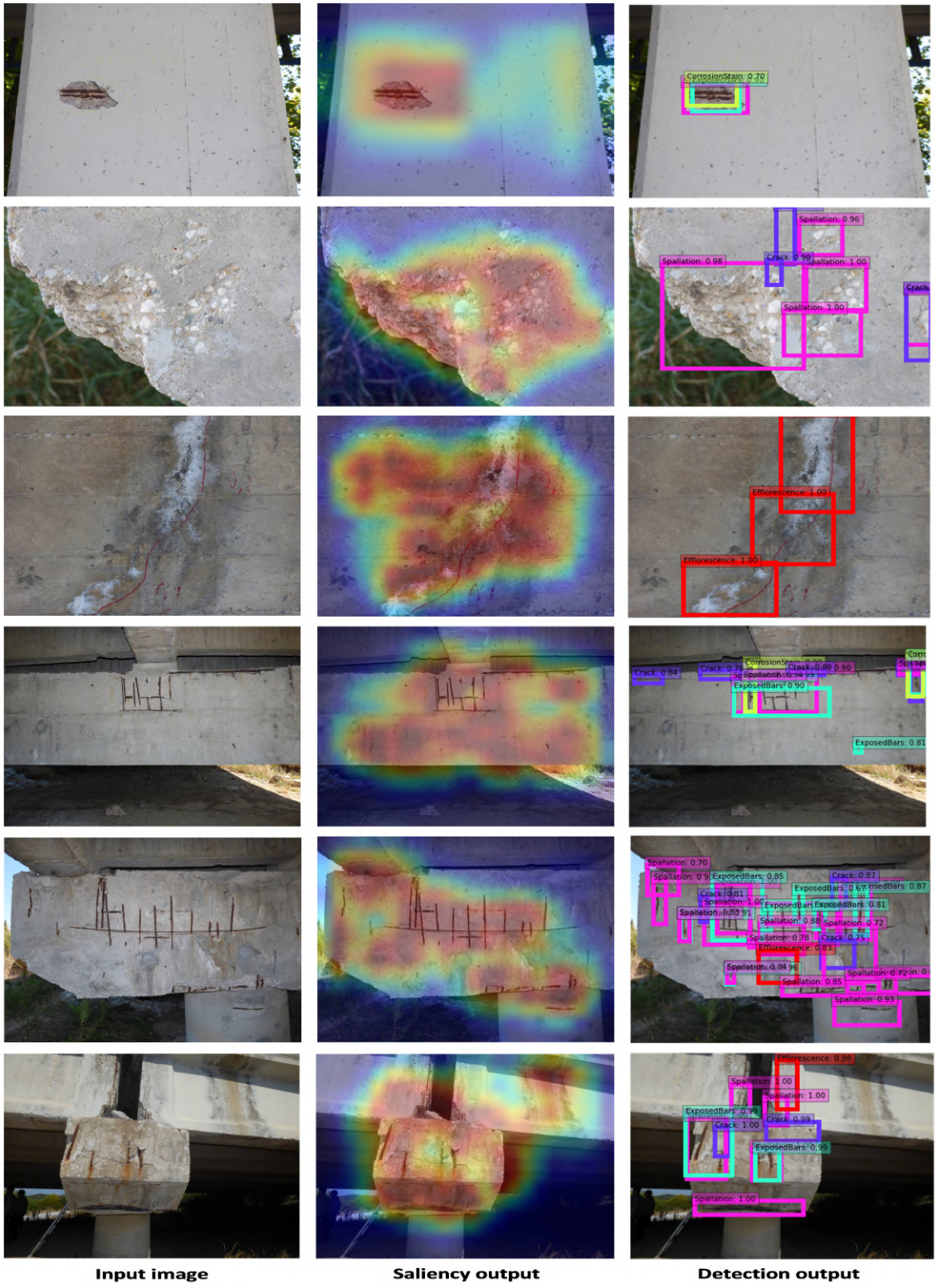


FIGURE 4.7 – Résultats de détection par SMDD-Net sur certaines images.

Afin d'évaluer les mérites de la méthode SMDD-Net, nous avons comparé ses performances avec sept autres méthodes [44, 61, 90, 53, 26, 86, 43]. Patel et al. [61] ont utilisé une version améliorée de Faster-RCNN en ajoutant une fonction de perte multi-étiquette pour la détection des défauts dans les surfaces en béton. Ils ont souligné différents éléments qui affectent la précision du modèle, tels que l'incapacité des boîtes englobantes à décrire avec précision la forme complexe des patchs de défaut, ainsi que des imprécisions dans les annotations du jeu de données CODEBRIM. Xiong et al. [66] ont utilisé le modèle pré-entraîné YOLO-v4 sur le jeu de données MS COCO, pour le réentrainer sur l'ensemble de données d'entraînement CODEBRIM, pour la détection de défauts dans les surfaces en béton, afin de comparer les inspections visuelles avec les inspections automatisées. Les autres méthodes ont étaient implémentées à base de YOLOv5-l [43], YOLOv8-l [44], RetinaNet [53], YOLOX [26] et YOLOR [86]. La table 4.5 montre une comparaison de la méthode SMDD-Net avec ces méthodes. Clairement, notre méthode a surpassé les autres méthodes en termes de précision de détection. Notamment, les performances des modèles de base YOLOR et YOLOX sont arrivées en deuxième et troisième place respectivement, derrière la méthode SMDD-Net, mais avec un écart significatif par rapport à cette dernière (l'écart était de 7,3% par rapport à YOLOX et 9,9% par rapport à YOLOR). Ces résultats ont démontré, entre autres, les avantages de l'utilisation du module d'attention pour améliorer la représentation des caractéristiques locales, ce qui a considérablement amélioré la précision de détection. La figure 4.8 montre une comparaison des courbes de validation pour les méthodes implémentées, démontrant que SMDD-Net, RetinaNet et YOLOX ont convergé plus rapidement que les autres méthodes.

TABLE 4.5 – Comparaison de SMDD-Net avec d'autres méthodes.

| Method | #Param. | One-Stage | Two-Stage | Classification Head | Bounding Box Head | mAP@0.5 (%) | Speed (s) |
|---|---|---|---|---|---|---|---|
| Patel et al. [61] | 74.4 M | - | ✓ | BCE [1] Loss | Smooth L1 Loss | 91.2 | 0.14 |
| Xiong et al. [90] | 52.9 M | ✓ | - | BCE Loss | Smooth L1 Loss | 22.7 | 0.03 |
| YOLOv5-l [43] | 46.1 M | ✓ | - | BCE Loss | Smooth L1 Loss | 41.7 | 0.02 |
| YOLOv8-l [44] | 43.7 M | ✓ | - | BCE Loss | Smooth L1 Loss | 59.6 | 0.02 |
| RetinaNet [53] | 36.5 M | ✓ | - | Focal Loss | Smooth L1 Loss | 88.4 | 0.07 |
| YOLOX-l [26] | 54.2 M | ✓ | - | BCE Loss | Smooth L1 Loss | 91.8 | 0.04 |
| YOLOR-P6 [86] | 36.9 M | ✓ | - | BCE Loss | L2 Loss | 89.2 | 0.04 |
| SMDD-Net | 36.5 M | ✓ | - | Focal Loss | Smooth L1 Loss | 99.1 | 0.11 |

[1] Binary Cross-Entropy.

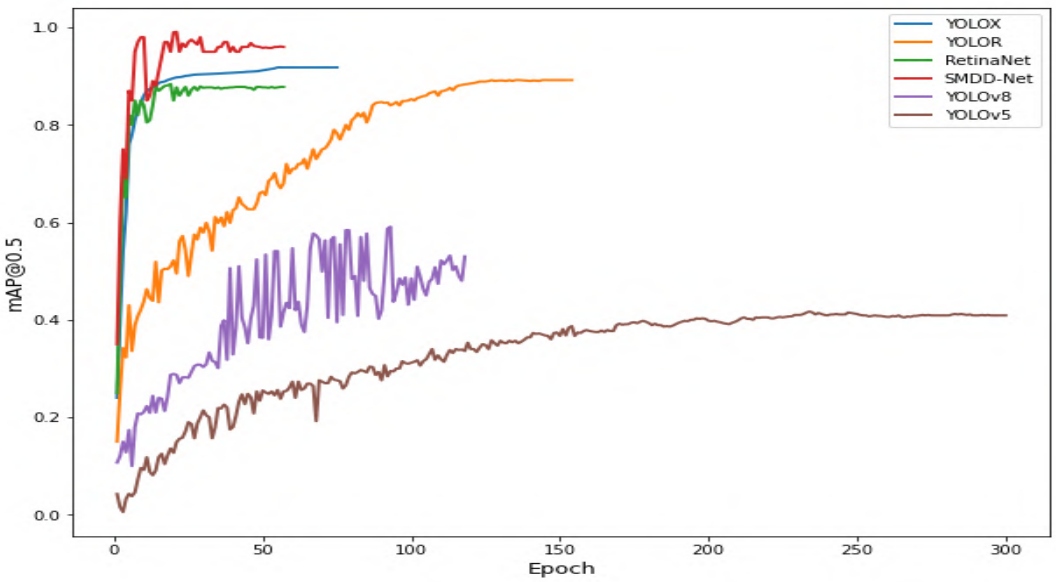


FIGURE 4.8 – Comparaison des courbes de validation.

Pour comparer qualitativement les méthodes implémentées rapportées dans la table 4.5, la figure 4.9 présente des exemples de détection de défauts sur des surfaces en béton. Nous avons sélectionné ces exemples car ils étaient particulièrement difficiles. Dans le premier exemple, l'image a été acquise à une distance courte et contenait deux défauts (une longue fissure à gauche et de la corrosion à droite). Seuls SMDD-Net et YOLOR ont détecté la corrosion. Notez également que, en raison de la détection multi-étiquette, le défaut à droite a été attribué à une autre boîte englobante représentant une barre exposée, ce qui était une détection valide. Les exemples du deuxième au cinquième, ont été acquis à une distance moyenne. Dans le deuxième exemple, l'image contenait plusieurs défauts (deux taches d'oxydation et une fissure). SMDD-Net a détecté tous ces défauts, tandis que les autres méthodes ont manqué un ou plusieurs défauts. Dans le troisième exemple, l'image contenait diverses petites fissures détectées par SMDD-Net, mais la plupart d'entre elles ont été manquées par les autres méthodes. Dans le quatrième exemple, l'image contenait une fissure au milieu d'une grande partie de fond. Seul SMDD-Net a été capable de la détecter, ainsi que certaines efflorescences. Dans le cinquième exemple, l'image contenait une barre exposée et plusieurs petites fissures, qui sont difficiles à détecter même avec l'œil humain. SMDD-Net a détecté tous les défauts, mais les autres méthodes ont manqué la plupart des fissures. Enfin, dans le sixième exemple, l'image contenait de la corrosion dans une zone d'écaillage, ainsi qu'une partie

significative avec des efflorescences. Seuls SMDD-Net et YOLOv8 ont réussi à détecter l'écaillage et la corrosion. Ces résultats ont démontré l'efficacité de la méthode SMDD-Net pour la détection de défauts dans des scénarios difficiles.

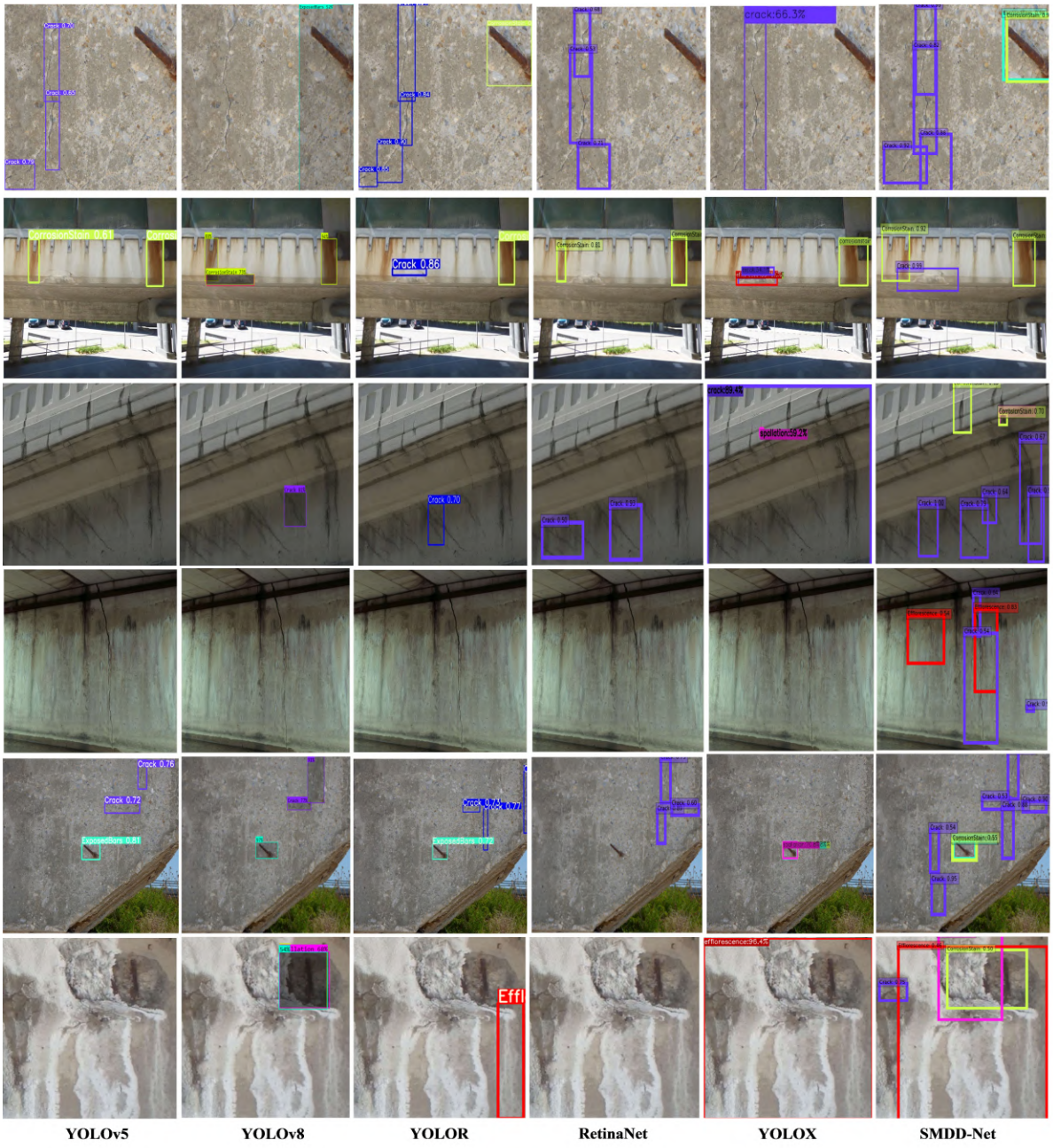


FIGURE 4.9 – Visualisation des résultats quantitatifs sur des images de surface en béton.

### Limites de la méthode proposée SMDD-Net

Notre méthode proposée, SMDD-Net, présente certaines limites qu'il est important de prendre en compte. Tout d'abord, la présence de graffitis sur certains ponts peut altérer les résultats de la détection de défauts (voir la figure 4.10). Un graffiti peut posséder la forme d'un défaut (une fissure par exemple) qui peut mener à un FP, ou bien cacher le défaut lui même et qui peut mener à un FN. En outre, il convient de noter que la méthode SMDD-Net a été entraînée sur des images de portée proche et moyenne, ce qui signifie que si elle est testée sur des images à longue portée, elle peut ne pas être efficace. Enfin, notre méthode est un peu lente en termes de temps d'inférence, en raison du calcul de la saillance globale et locale qui prend du temps. Ces limites doivent être prises en compte pour garantir une utilisation efficace et précise de la méthode SMDD-Net dans la détection de défauts sur les ponts.

FIGURE 4.10 – Exemples d'images contenant des graffitis, prises du jeu de données CODEBRIM [58].

### Déploiement de la solution proposée SMDD-Net

La méthode proposée sera intégrée en premier lieu en mode non-embarqué dans le système développé pour l'inspection des ponts en béton. Le premier essai sur le terrain du système a eu lieu en juin 2022 en utilisant le modèle YOLOX (à cette époque SMDD-Net n'était pas encore développée) pour le module d'intelligence artificielle. Les objectifs de ce vol d'essai étaient d'évaluer la maturité technique du système, mettre en pratique le mode d'emploi et rassembler les leçons apprises afin d'améliorer le système pour le prochain vol. Parmi ces leçons, on cite les deux suivantes qui sont les plus importantes :

– **La lenteur du système d'inspection :** l'envoi d'images vers le serveur pour être analysées par un modèle de détection de défauts (YOLOX ou SMDD-Net), peut être assez lent, avec un débit d'environ 4 images par seconde, et gourmand en bande passante, requérant environ 200 Mbps. Ceci peut ralentir le système d'inspection. Une solution à ce problème est l'utilisation des modèles légers dits "embarqués", qui sont spécialement conçus pour être exécutés sur des appareils tels que les drones. En utilisant un modèle embarqué, le drone pourrait effectuer les inférences directement sur les images, sans avoir besoin de les envoyer vers un serveur distant. Cela réduirait considérablement le temps de traitement et la bande passante nécessaires, ce qui permettrait un système d'inspection de ponts plus rapide et plus efficace. Une autre solution à ce problème consisterait à augmenter la bande passante et la vitesse de traitement en utilisant, par exemple, du matériel plus rapide.

– **Limitation de la solution SMDD-Net pour les objets éloignés :** nous avons observé aussi que lorsque SMDD-Net est exécutée sur des images à longue portée, elle donne des fausses détections tel qu'il est illustré à la figure 4.11, ce qui confirme ce que nous avons mentionné précédemment sur les limites de SMDD-Net.


FIGURE 4.11 – Exemple d'inférence par SMDD-Net sur une vidéo prise par drone. En haut à droite, on peut voir que SMDD-Net a détecté un arbre comme étant un objet de classe, ce qui est incorrect. Cette erreur est probablement due à l'éloignement de l'objet.

### 4.2.3 Segmentation

Pour les tests de segmentation sémantique, nous avons mené deux expériences : la première en utilisant le jeu de données Crack Segmentation Dataset, et la deuxième en utilisant CODEBRIM. Dans ce qui suit, nous détaillerons chaque expérience.

#### Résultats de la première expérience

Les résultats de notre première expérience (avec Crack Segmentation Dataset) sont illustrés dans les figures ci-dessous. On observe que le meilleur résultat sur l'ensemble de données de validation est 97%, obtenu par la configuration U-Net avec InceptionV3 comme encodeur. Pour visualiser ces résultats sur des images comportant des défauts, nous avons exécuté des inférences de chacune des quatre configurations sur des images qui appartiennent au jeu de données CODEBRIM. Un exemple de visualisation sur une image contenant le défaut Crack est donné en figure 4.16.

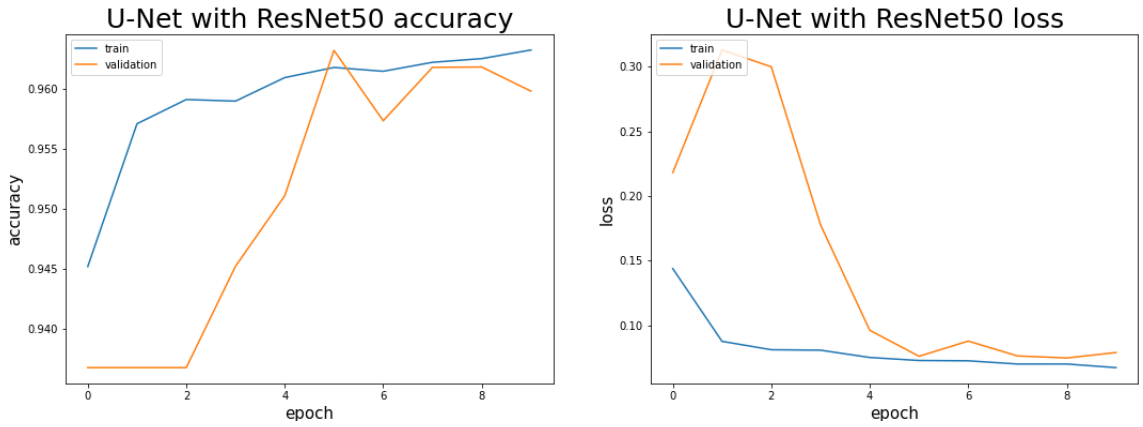


FIGURE 4.12 – Résultats avec Crack Segmentation Dataset : U-Net avec ResNet50 comme encodeur.

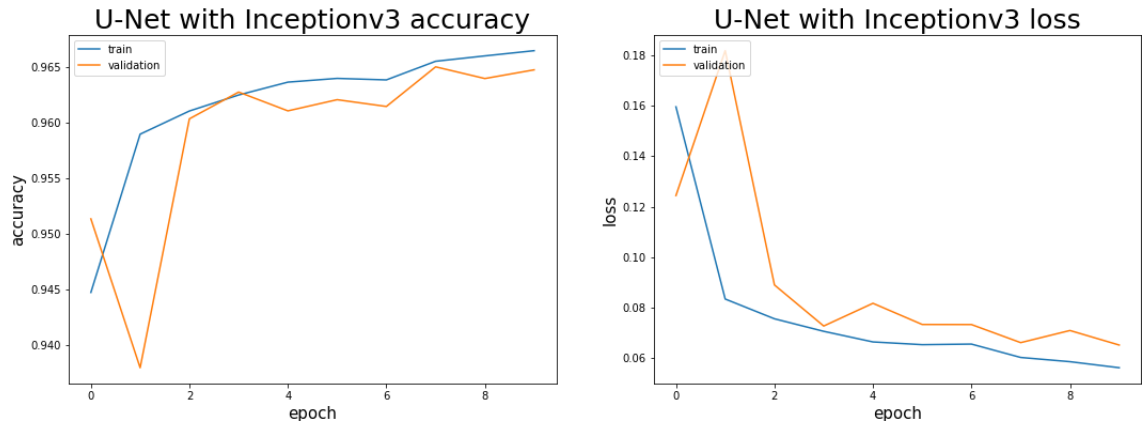


FIGURE 4.13 – Résultats avec Crack Segmentation Dataset : U-Net avec InceptionV3 comme encodeur.

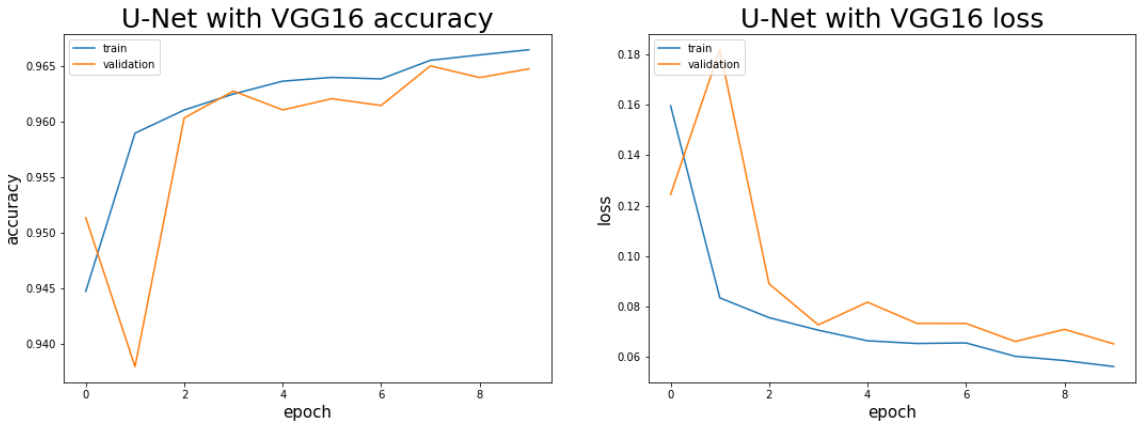


FIGURE 4.14 – Résultats avec Crack Segmentation Dataset : U-Net avec VGG16 comme encodeur.

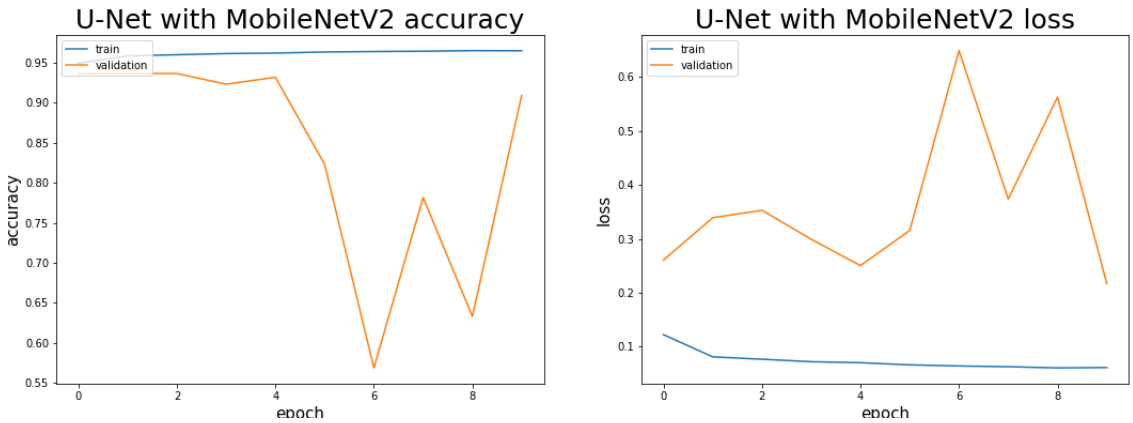


FIGURE 4.15 – Résultats avec Crack Segmentation Dataset : U-Net avec MobilNetV2 comme encodeur.

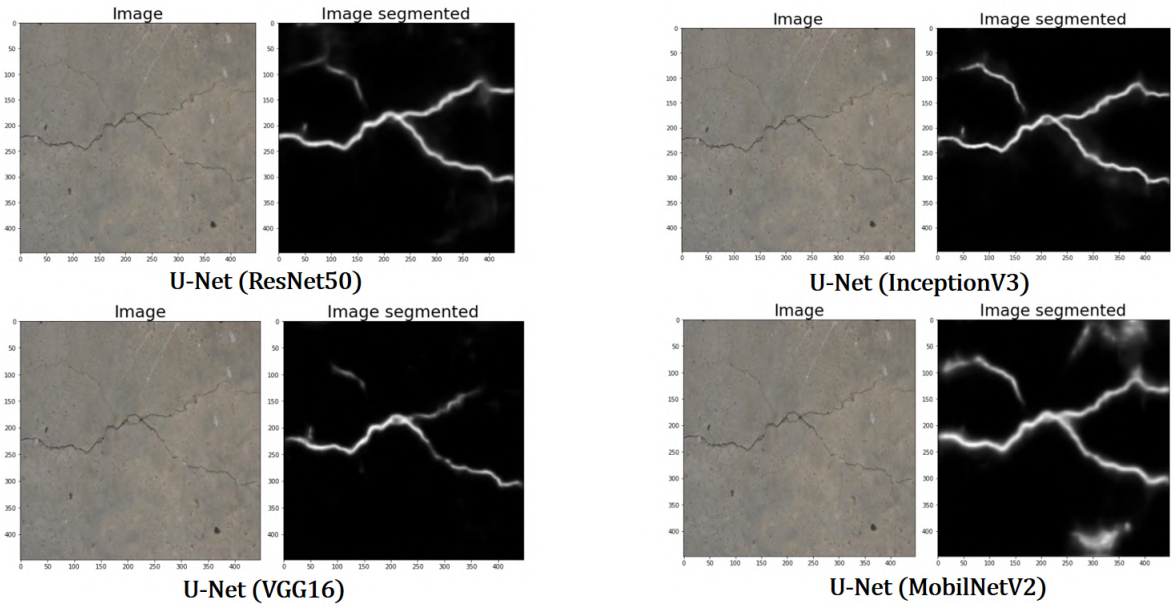


FIGURE 4.16 – Résultats avec Crack Segmentation Dataset : Visualisation sur le jeu de données CODEBRIM.

**Résultats de la deuxième expérience**

Les résultats de notre deuxième expérience (avec CODEBRIM Dataset) sont illustrés dans la table 4.6. On rappelle qu'avec le jeu de données CODEBRIM , nous avons créé cinq classificateurs de pixels binaires, chacun est responsable de segmenter une classe vu que CODEBRIM est un jeu de données multi-étiquette (Proposition 3.4).

TABLE 4.6 – Les résultats des cinq modèles de segmentation binaire testés sur le dataset CODEBRIM.

| **Modèle** | **Accuracy** (%) |
|---|---|
| Modèle 1 : Crack | 99 |
| Modèle 2 : CorrosionStain | 98 |
| Modèle 3 : Efflorescence | 81 |
| Modèle 4 : ExposedBars | 97 |
| Modèle 5 : Spallation | 86 |

La figure 4.17 montre les résultats de visualisation des cinq classificateurs. On observe que l'exactitude (Accuracy) obtenue avec le classificateur entraîné sur la classe Crack n'est pas représentative, car le background domine les masques que nous avons créé pour les cracks. Pour résoudre ce problème, nous avons appliqué les trois techniques suivantes :

- **Dilatation :** utiliser un filtre de dilatation sur les masques de crack crées : Karnel (5,5) avec 1,2,3.... itérations (Figure 4.18).
- **Combinaison de masques :** combiner les masques dilatés avec ceux de Crack Segmentation Dataset.
- **Technique ensembliste :** utiliser une méthode ensembliste pour améliorer l'exactitude d'une nouvelle prédiction :

  Entraîner les modèles $M_1$ :U-Net(ResNet50), $M_2$ :U-Net(InceptionV3) et $M_3$ :U-Net(VGG16) sur le jeu de données résultant de la combinaison ;

  Effectuer une recherche de valeurs pour trouver une meilleure combinaison de [w1,w2,w3] qui maximisera l'exactitude du modèle combiné : $w_1M_1+w_2M_2+w_3M_3$ sur l'ensemble de données de test ; Les valeurs possibles de $w_1$, $w_2$ et $w_3$ son prises dans l'ensemble $\{0.1, 0.2, 0.3\}$. Cette combinaison est utilisée lors de nouvelles prédictions. Avec le script 4.19, on trouvera les valeurs optimale : $(w_1, w_2, w_3) = (0.1, 0.1, 0.3)$.

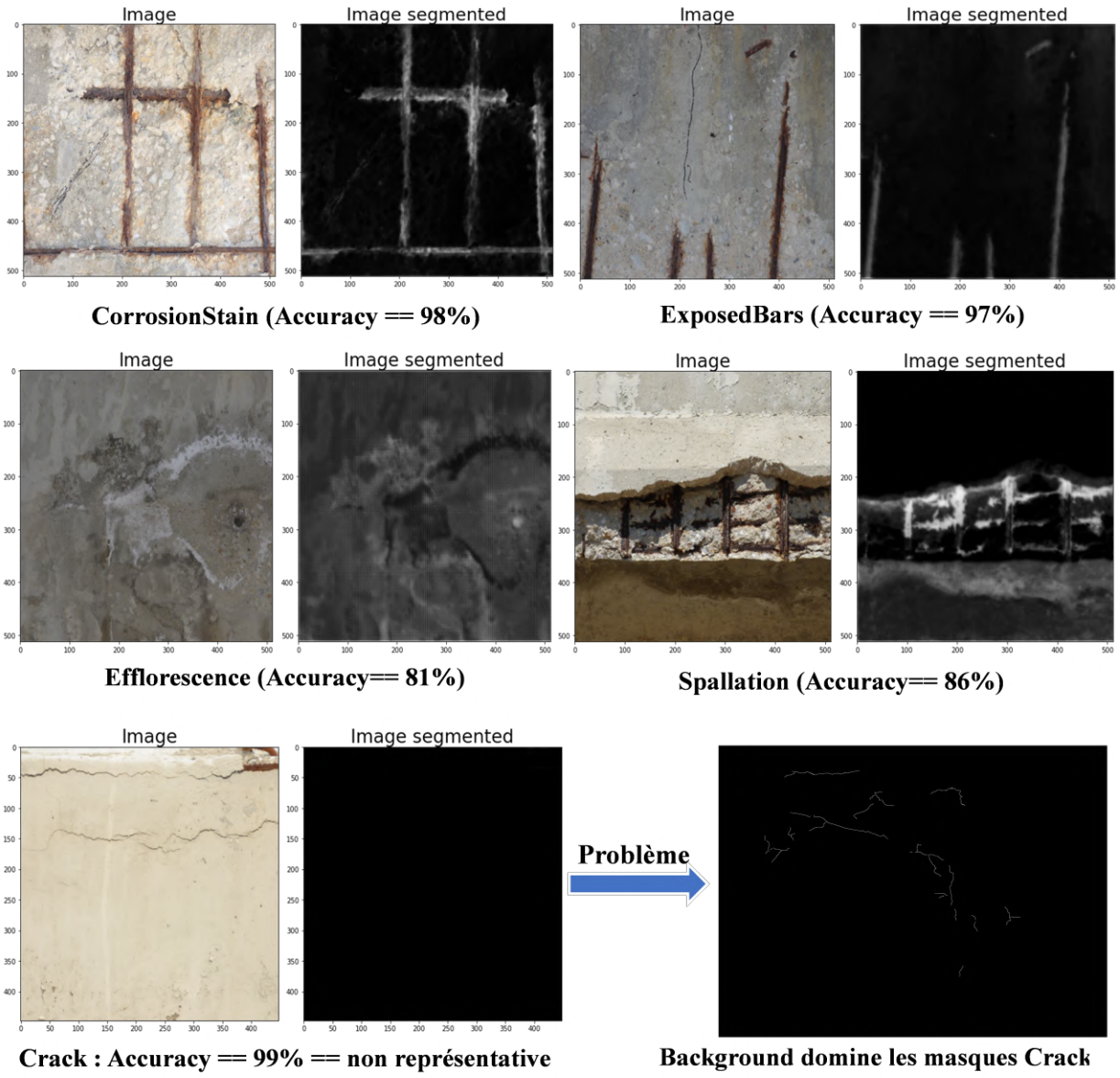


FIGURE 4.17 – Résultats avec le jeu de données CODEBRIM : Visualisation.

Après la dilatation des masques, la combinaison des masques et la méthode ensembliste, le masque prédit de l'image crack précédente est montré dans la figure 4.20

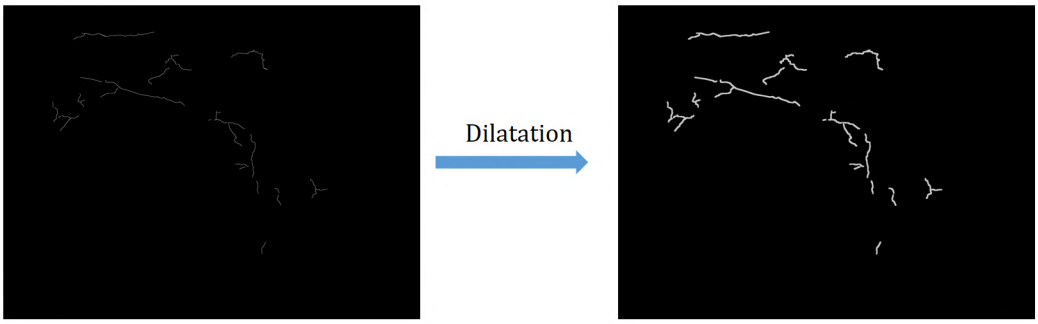


FIGURE 4.18 – Amélioration des résultats de Crack du jeu de données CODEBRIM : Dilatation.

```
#Grid search for the best combination of w1, w2, w3 that gives maximum acuracy

import pandas as pd
df = pd.DataFrame([])

for w1 in range(0, 4):
    for w2 in range(0,4):
        for w3 in range(0,4):
            wts = [w1/10.,w2/10.,w3/10.]

            IOU_wted = MeanIoU(num_classes=n_classes)
            wted_preds = np.tensordot(preds, wts, axes=((0),(0)))
            IOU_wted.update_state(y_test[:,:,:,0], wted_preds)
            print("Now predciting for weights :", w1/10., w2/10., w3/10., " : IOU = ", IOU_wted.result().numpy())
            df = df.append(pd.DataFrame({'wt1':wts[0],'wt2':wts[1],
                                         'wt3':wts[2], 'IOU': IOU_wted.result().numpy()}, index=[0]), ignore_index=True)

max_iou_row = df.iloc[df['IOU'].idxmax()]
print("Max IOU of ", max_iou_row[3], " obained with w1=", max_iou_row[0],
      " w2=", max_iou_row[1], " and w3=", max_iou_row[2])
```

```
pred1 = model1.predict(img)
pred2 = model2.predict(img)
pred3 = model3.predict(img)

preds=np.array([pred1, pred2, pred3])

#Use tensordot to sum the products of all elements over specified ax
weighted_preds = np.tensordot(preds, weights, axes=((0),(0)))

result_img =  weighted_preds.reshape(mask.shape)

mask_class0 = result_img[:,:,0]
```

FIGURE 4.19 – Amélioration des résultats de la classe Crack du jeu de données CODEBRIM : Technique ensembliste.

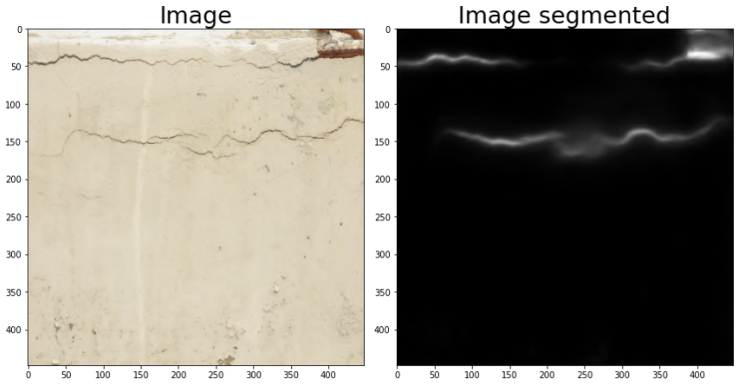


FIGURE 4.20 – Résultats de la segmentation sémantique de la classe Crack du jeu de données CODEBRIM après améliorations.

## 4.3 Conclusion

Ce chapitre résume les résultats des tests menés dans chacun des problèmes de recherche traités. Nous avons surpassé les résultats de l'état de l'art pour la classification de défauts avec AUC-ROC = 96% obtenu par ResNet50. Pour la détection de défaut, la nouvelle méthode proposée qui combine la détection d'objet (RetinaNet) avec la saillance visuelle (Grad-CAM) était très puissante en terme de précision, et elle a permis d'améliorer substantiellement la performance de RetinaNet en passant de mAP = 88.4% à mAP = 99.1%. Toutefois, elle est un peu lente en terme de temps d'inférence par rapport aux autres modèles d'un seul étage (YOLOX, YOLOR, RetinaNet). Dans le cas de la segmentation de défaut, nous avons implémenté la segmentation binaire pour l'instant (proposition 3), et la configuration U-Net avec Inceptionv3 comme encodeur a permis d,obtenir le meilleur résultat parmi les autres configurations. Nous proposons de traiter la segmentation multi-classe (proposition 4) pour la suite de ce travail de recherche.

# Chapitre 5

# Conclusion générale et perspectives

L'inspection périodique des infrastructures de transport est indispensable afin d'assurer leur qualité à long terme. Toutefois, les méthodes traditionnelles d'inspection sont très coûteuses en terme du temps, d'efforts et d'argent. Ceci a motivé l'utilisation croissante des inspections automatisées basées sur des techniques de vision par ordinateur, surtout avec l'apparition des véhicules aériens sans pilote (UAV) qui permettent un accès rapide, sûr et facile aux éléments d'infrastructure éloignés et importants. Les algorithmes d'apprentissage profond, en particulier les CNNs, sont devenus les méthodes les plus utilisées pour la classification, la détection et la segmentation de défauts dans ces infrastructures. Plusieurs recherches alors ont été menées afin de proposer des modèles simples et robustes pour résoudre le problème de classification/détection/segmentation de défauts dans les différentes types de structures, en particulier les ponts en béton. Cependant, la plupart de ces modèles se concentrent principalement sur la classification de défaut, et seuls quelques-uns traitent plusieurs défauts à la fois.

Le travail proposé dans ce mémoire est motivé par cette problématique, et décrit une approche automatisée qui utilise l'apprentissage profond, pour la classification, la détection et la segmentation de défauts multi-classe multi-étiquette sur des images acquises par drone pour des structures en béton telles que les ponts. Comparées à l'état de l'art, les techniques décrites dans notre approche pour les trois problèmes liés aux défauts (classification, détection, segmentation) ont démontré des meilleurs résultats. Néanmoins, certaines limites doivent être tracées pour une future continuité et amélioration de l'approche proposée. C'est pourquoi, on propose comme futures perspectives pour ce travail de recherche d'améliorer notre approche ainsi :

- **Classification de défauts :** dans un problème multi-classe multi-étiquette lié aux défauts, on observe deux relations importantes entre les classes : la corrélation et la causalité. Les classificateurs que nous avons implémenté ne considèrent pas ces relations, et par conséquent, on propose d'investiguer la possibilité d'implémenter ces deux relations dans un classificateur (ResNet, Inception, DenseNet...etc).
- **Détection de défauts :** notre méthode proposée SMDD-Net pour la détection de défauts est très puissante en terme d'exactitude mais pas en terme de vitesse de calcul (temps d'inférence = 0.15 s). On propose alors de l'améliorer en simplifiant l'architecture ou en proposant des outils qui permettent d'accélérer l'inférence. Pour combiner le module de saillance avec celui d'apprentissage profond dans notre méthode, nous avons utilisé la multiplication suivie d'une addition (les connexions résiduelles). D'autres opérations (la concaténation) peuvent donner de mêmes ou de meilleurs résultats. Pour le déploiement de la méthode, on propose de la déployer en mode embarqué afin d'avoir un système d'inspection de ponts plus rapide et plus efficace.
- **Segmentation de défauts :** pour ce problème, nous avons implémenté que la segmentation binaire (proposition 3) pour chaque classe, et nous proposons donc de traiter la segmentation multi-classe (proposition 4) pour la suite.

Par ailleurs, nous proposons aussi comme futures directions de ce travail les deux objectifs à long terme suivants : a) évaluer chaque défaut en lui affectant un degré de gravité, et b) étendre les solutions proposées dans les trois problèmes pour la vidéo.

# Bibliographie


[1] Classification : courbe ROC et AUC. [Online :`https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc` ; accessed 21-February-2023].

[2] Keras documentation : Keras Applications. [Online :`https://keras.io/api/applications/` ; accessed 21-February-2023].

[3] TACM - Concrete Profiles. [Online : `http://technicalconcrete.com/concrete-profiles` ; accessed 01-March-2023].

[4] Abdel-Qader, I., Abudayyeh, O., and Kelly, M. E. Analysis of edge-detection techniques for crack identification in bridges. *Journal of Computing in Civil Engineering 17*, 4 (2003), 255–263.

[5] Abdelkhalek, S. State-of-the-art review of defect detection in concrete bridge deck. In *Proceedings of the CIB World Building Congress* (2019).

[6] Amhaz, R., Chambon, S., Idier, J., and Baltazart, V. Automatic crack detection on two-dimensional pavement images : An algorithm based on minimal path selection.

[7] Amirkhani, D., and Bastanfard, A. An objective method to evaluate exemplar-based inpainted images quality using jaccard index. *Multimedia Tools and Applications* (2021), 1–14.

[8] Atha, D. J., and Jahanshahi, M. R. Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. *Structural Health Monitoring 17*, 5 (2018), 1110–1128.

[9] Benz, C., Debus, P., Ha, H. K., and Rodehorst, V. Crack segmentation on uas-based imagery using transfer learning. In *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)* (2019), IEEE, pp. 1–6.

[10] Boulmerka, A., Allili, M. S., and Ait-Aoudia, S. A generalized multiclass histogram thresholding approach based on mixture modelling. *Pattern Recognition 47*, 3 (2014), 1330–1348.

[11] Bukhsh, Z. A., Jansen, N., and Saeed, A. Damage detection using in-domain and cross-domain transfer learning. *Neural Computing and Applications 33*, 24 (2021), 16921–16936.

[12] Calvi, G. M., Matteo, M., O'Reilly, G. J., Scattarreggia, N., Monteiro, R., Malomo, D., Calvi, M. P., and Pinho, R. Once upon a time in Italy : The tale of the morandi bridge. *Structural Engineering International 29* (2019), 198-217.

[13] Dais, D., Bal, I. E., Smyrou, E., and Sarhosis, V. Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning. *Automation in Construction 125* (2021), 103606.

[14] Diamanti, N., Annan, A. P., and Redman, J. D. Concrete bridge deck deterioration assessment using ground penetrating radar (gpr). *Journal of Environmental and Engineering Geophysics 22*, 2 (2017), 121–132.

[15] Dorafshan, S., Thomas, R. J., and Maguire, M. Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Construction and Building Materials 186* (2018), 1031–1045.

[16] Dorafshan, S., Thomas, R. J., and Maguire, M. Sdnet2018 : An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. *Data in brief 21* (2018), 1664–1668.

[17] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929* (2020).

[18] Dung, C. V., et al. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction 99* (2019), 52–58.

[19] Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M., Ebersbach, D., Stoeckert, U., and Gross, H.-M. How to get pavement distress detection ready for deep learning ? a systematic approach. In *International Joint Conference on Neural Networks (IJCNN)* (2017), pp. 2039–2047.

[20] et Approvisionnement Canada (TPSGC), S. P. *Manuel d'inspection des ponts, Génie structural, naval et génie des transports*. 2010.

[21] Faula, Y., Eglin, V., and Bres, S. One-class detection and classification of defects on concrete surfaces. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (2021), IEEE, pp. 826–831.

[22] Feng, C., Zhang, H., Wang, S., Li, Y., Wang, H., and Yan, F. Structural damage detection using deep convolutional neural network and transfer learning. *KSCE Journal of Civil Engineering 23*, 10 (2019), 4493–4502.

[23] Feroz, S., and Abu Dabous, S. Uav-based remote sensing applications for bridge condition assessment. *Remote Sensing 13*, 9 (2021), 1809.

[24] Ferri, C., Hernández-Orallo, J., and Modroiu, R. An experimental comparison of performance measures for classification. *Pattern recognition letters 30*, 1 (2009), 27–38.

[25] Filali, I., Allili, M. S., and Benblidia, N. Multi-scale salient object detection using graph ranking and global–local saliency refinement. *Signal Processing : Image Communication 47* (2016), 380–401.

[26] Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. Yolox : Exceeding yolo series in 2021. *arXiv preprint arXiv :2107.08430* (2021).

[27] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1440–1448.

[28] Guo, C., and Zhang, L. A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression. *IEEE transactions on image processing 19*, 1 (2009), 185–198.

[29] Harel, J., Koch, C., and Perona, P. Graph-based visual saliency.

[30] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[31] Hechun, W., and Xiaohong, Z. Survey of Deep Learning Based Object Detection. In *Proceedings of the 2nd International Conference on Big Data Technologies - ICBDT2019* (Jinan, China, 2019), ACM Press, pp. 149–153.

[32] Hendricks, L. J., Baxter, J. S., Chou, Y., Thomas, M., Boekweg, E., Guthrie, W. S., and Mazzeo, B. A. High-speed acoustic impact-echo sounding of concrete bridge decks. *Journal of Nondestructive Evaluation 39*, 3 (2020), 1–12.

[33] Hu, D., Hou, F., Blakely, J., and Li, S. Augmented reality based visualization for concrete bridge deck deterioration characterized by ground penetrating radar. In *Construction Research Congress 2020 : Computer Applications* (2020), American Society of Civil Engineers Reston, VA, pp. 1156–1164.

[34] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 4700–4708.

[35] Hüthwohl, P., Lu, R., and Brilakis, I. Multi-classifier for reinforced concrete bridge defects. *Automation in Construction 105* (2019), 102824.

[36] Itti, L., Koch, C., and Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence 20*, 11 (1998), 1254–1259.

[37] Jahanshahi, M. R., Kelly, J. S., Masri, S. F., and Sukhatme, G. S. A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures. *Structure and Infrastructure Engineering 5*, 6 (2009), 455–486.

[38] Jeong, E., Seo, J., and Wacker, J. Literature review and technical survey on bridge inspection using unmanned aerial vehicles. *Journal of Performance of Constructed Facilities 34*, 6 (2020), 04020113.

[39] ji, w. The summary of code and paper for salient object detection with deep learning, Feb. 2022.

[40] Jiang, Y., Pang, D., and Li, C. A deep learning approach for fast detection and classification of concrete damage. *Automation in Construction 128* (2021), 103785.

[41] Jiang, Y., Pang, D., and Li, C. A deep learning approach for fast detection and classification of concrete damage. *Automation in Construction 128* (2021), 103785.

[42] Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. A survey of deep learning-based object detection. *IEEE access 7* (2019), 128837–128868.

[43] Jocher, G. Yolov5. [Online :`https://github.com/ultralytics/yolov5` ; accessed 27-February-2023].

[44] Jocher, G. Yolov8. [Online :`https://github.com/ultralytics/ultralytics` ; accessed 27-February-2023].

[45] Kilic, G., and Caner, A. Augmented reality for bridge condition assessment using advanced non-destructive techniques. *Structure and Infrastructure Engineering 17*, 7 (2021), 977–989.

[46] Kim, B., Yuvaraj, N., Sri Preethaa, K., and Arun Pandian, R. Surface crack detection using deep learning with shallow cnn architecture for enhanced computation. *Neural Computing and Applications 33*, 15 (2021), 9289–9305.

[47] Kim, I.-H., Jeon, H., Baek, S.-C., Hong, W.-H., and Jung, H.-J. Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle. *Sensors 18*, 6 (2018), 1881.

[48] Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., and Fieguth, P. A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics 29*, 2 (2015), 196–210.

[49] Lapointe, J.-F., Allili, M. S., Belliveau, L., Hebbache, L., Amirkhani, D., and Sekkati, H. AI-AR for Bridge Inspection by Drone. In *24TH HCI International Conference on human-computer interaction, HCII 2022* (Accepted).

[50] Li, C.-L., Sohn, K., Yoon, J., and Pfister, T. Cutpaste : Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 9664–9674.

[51] Li, F.-F., Johnson, J., and Yeung, S. Lecture 6 : Training Neural Networks.

[52] Lin, S., Meng, D., Choi, H., Shams, S., and Azari, H. Laboratory assessment of nine methods for nondestructive evaluation of concrete bridge decks with overlays. *Construction and Building Materials 188* (2018), 966–982.

[53] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2980–2988.

[54] Liu, B. Transformers in Vision : From Zero to Hero.

[55] LIU, K. Crack-Detection-and-Segmentation-Dataset-for-UAV-Inspection, Jan. 2022.

[56] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. Deep learning for generic object detection : A survey. *International journal of computer vision 128*, 2 (2020), 261–318.

[57] Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., and Omata, H. Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering 33*, 12 (2018), 1127–1141.

[58] Mundt, M., Majumder, S., Murali, S., Panetsos, P., and Ramesh, V. Meta-learning convolutional neural architectures for multi-target concrete defect classification with the concrete defect bridge image dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 11196–11205.

[59] Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. *Advances in neural information processing systems 26* (2013).

[60] National Transportation Safety Board. Collapse of I-35W highway bridge, Minneapolis, Minnesota, august 1, 2007, 2008.

[61] Patel, R. A., Steinmann, L., Fehrenbach, J., Fehrenbach, D., and Dehn, F. Convolution neural network-based machine learning approach for visual inspection of concrete structures. In *Proceedings of the 1st Conference of the European Association on Quality Control of Bridges and Structures : EUROSTRUCT 2021 1* (2022), Springer, pp. 704–712.

[62] Pedamkar, P. Machine Learning System | Complete Guide to Machine Learning System, Jan. 2020. [Online : `https://www.educba.com/machine-learning-system/` ; accessed 21-February-2023].

[63] Rateke, T., and Von Wangenheim, A. Road surface detection and differentiation considering surface damages. *Autonomous Robots 45*, 2 (2021), 299–312.

[64] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 779–788.

[65] Redmon, J., and Farhadi, A. Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767* (2018).

[66] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn : Towards real-time object detection with region proposal networks. *Advances in neural information processing systems 28* (2015).

[67] RHAZI, J. Half-cell potential test from the upper-side and the lower-side of reinforced concrete slabs : a comparative study. *NDTCE'09, Non-Destructive Testing in Civil Engineering* (2009).

[68] Ronneberger, O., Fischer, P., and Brox, T. U-net : Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[69] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252.

[70] Salman, M., Mathavan, S., Kamal, K., and Rahman, M. Pavement crack detection using the Gabor filter. In *Proceedings of the IEEE Conference on intelligent transportation systems (ITSC 2013)* (2013), pp. 2039–2044.

[71] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-CAM : Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision 128*, 2 (oct 2019), 336–359.

[72] Shao, L., Zhu, F., and Li, X. Transfer learning for visual categorization : A survey. *IEEE transactions on neural networks and learning systems 26*, 5 (2014), 1019–1034.

[73] Shi, Y., Cui, L., Qi, Z., Meng, F., and Chen, Z. Automatic road crack detection using random structured forests. *IEEE Transactions on Intelligent Transportation Systems 17*, 12 (2016), 3434–3445.

[74] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556* (2014).

[75] Słoński, M. A comparison of deep convolutional neural networks for image-based detection of concrete surface cracks. *Computer Assisted Methods in Engineering and Science 26*, 2 (2019), 105–112.

[76] Su, C., and Wang, W. Concrete cracks detection using convolutional neural-network based on transfer learning. *Mathematical Problems in Engineering 2020* (2020).

[77] Sun, Y., Yang, Y., Yao, G., Wei, F., and Wong, M. Autonomous crack and bughole detection for concrete surface image based on deep learning. *IEEE Access 9* (2021), 85709–85720.

[78] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence* (2017).

[79] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[80] Tabernik, D., Šela, S., Skvarč, J., and Skočaj, D. Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing 31*, 3 (2020), 759–776.

[81] Talab, A. M. A., Huang, Z., Xi, F., and HaiMing, L. Detection crack in image using Otsu method and multiple filtering in image processing techniques. *Optik 127* (2016), 1030–1033.

[82] Teng, S., Liu, Z., Chen, G., and Cheng, L. Concrete crack detection based on well-known feature extractor model and the yolo_v2 network. *Applied Sciences 11*, 2 (2021), 813.

[83] Transports Québec, D. d. s., Ed. *Manuel d'inspection des structures*. Québec, 2010.

[84] Vaghefi, K., Melo e Silva, H., Harris, D., and Ahlborn, R. Application of thermal ir imagery for concrete bridge inspection. In *PCI National Bridge Conference, PCI/NBC, Salt Lake City : UT (USA)* (2011), pp. 1–12.

[85] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[86] Wang, C.-Y., Yeh, I.-H., and Liao, H.-Y. M. You only learn one representation : Unified network for multiple tasks. *arXiv preprint arXiv :2105.04206* (2021).

[87] Wang, W., and Su, C. Automatic classification of reinforced concrete bridge defects using the hybrid network. *Arabian Journal for Science and Engineering* (2022), 1–11.

[88] Wells, J. L., Lovelace, B., and Kalar, T. Use of unmanned aircraft systems for bridge inspections. *Transportation Research Record 2612*, 1 (2017), 60–66.

[89] Wu, X., Sahoo, D., and Hoi, S. C. Recent advances in deep learning for object detection. *Neurocomputing 396* (2020), 39–64.

[90] Xiong, R., Liu, P., and Tang, P. Human reliability analysis and prediction for visual inspection in bridge maintenance. In *Computing in Civil Engineering 2021*. 2022, pp. 254–262.

[91] Yaghi, S. R., and Dabous, S. A. State-of-the art practices in bridge inspection. *International Journal of Civil and Environmental Engineering 9*, 10 (2015), 1344–1347.

[92] Zhang, C., Chang, C.-c., and Jamshidi, M. Concrete bridge surface damage detection using a single-stage detector. *Computer-Aided Civil and Infrastructure Engineering 35*, 4 (2020), 389–409.

[93] Zhang, C., Chang, C.-c., and Jamshidi, M. Simultaneous pixel-level concrete defect detection and grouping using a fully convolutional model. *Structural Health Monitoring 20*, 4 (2021), 2199–2215.

[94] Zhang, L., Yang, F., Zhang, Y. D., and Zhu, Y. J. Road crack detection using deep convolutional neural network. In *Image Processing (ICIP), 2016 IEEE International Conference on* (2016), IEEE, pp. 3708–3712.

[95] Zhang, X., Rajan, D., and Story, B. Concrete crack detection using context-aware deep semantic segmentation network. *Computer-Aided Civil and Infrastructure Engineering 34*, 11 (2019), 951–971.

[96] Zhu, J., Zhang, C., Qi, H., and Lu, Z. Vision-based defects detection for bridges using transfer learning and convolutional neural networks. *Structure and Infrastructure Engineering 16*, 7 (2020), 1037–1049.

[97] Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T.-Y., Shlens, J., and Le, Q. V. Learning data augmentation strategies for object detection. In *European conference on computer vision* (2020), Springer, pp. 566–583.

[98] Zou, Q., Cao, Y., Li, Q., Mao, Q., and Wang, S. Cracktree : Automatic crack detection from pavement images. *Pattern Recognition Letters 33*, 3 (2012), 227–238.

[99] Zou, Z., Shi, Z., Guo, Y., and Ye, J. Object detection in 20 years : A survey. *arXiv preprint arXiv :1905.05055* (2019).