

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
TLEMCEM



المدرسة العليا للعلوم التطبيقية
Ecole Supérieure en
Sciences Appliquées

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
تلمسان

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master

Filière : Electronique

Spécialité : Informatique Industrielle

Présenté par : Mohammed Youcef BEMRAH

Mohammed Yacine BEMRAH

Thème

Contrôle de la qualité des poteries grâce à
l'intelligence artificielle

Soutenu publiquement, le 20 / 06 / 2026, devant le jury composé de :

Dr. KARAOUZENE Zoheir	MCB	ESSA. Tlemcen	Président
Dr. M'HAMEDDI Mohamed	MCB	ESSA. Tlemcen	Encadrant
Dr. NEDJAR Imane	MCA	ESSA. Tlemcen	Co-Encadrant
Dr. BEREKSI REGUIG Nassima	MCB	ESSA. Tlemcen	Examinatrice
Dr. ABDUL-RAHUMAN Ahmed	Docteur	HEXALOGY Tlemcen	Examineur

Année Universitaire : 2025 / 2026

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Remerciements

Au terme de ce travail, nous tenons à adresser nos sincères remerciements à toutes les personnes qui ont contribué à la réalisation de ce projet de fin d'études et qui nous ont accompagnés tout au long de cette expérience enrichissante.

Nous exprimons tout d'abord notre profonde gratitude à nos encadrants, **M. M'HAMEDI Mohammed** et **Mme. Nedjar Imane** pour leur disponibilité, leurs conseils avisés, leur suivi rigoureux ainsi que leurs orientations pertinentes.

Nous adressons également nos plus vifs remerciements à l'entreprise **Tlemcen Faïence SARL** pour nous avoir accueillis au sein de sa structure et nous avoir offert un cadre propice à la réalisation de ce projet. Cette expérience nous a permis de découvrir le monde industriel, de mettre en pratique nos connaissances académiques et d'acquérir une expérience professionnelle particulièrement enrichissante.

Nous tenons à remercier tout particulièrement le gérant de l'entreprise, **M. MERABET Abdennour**, pour son accueil chaleureux, sa confiance, sa disponibilité ainsi que les moyens mis à notre disposition durant toute la période du stage. Son soutien et son intérêt pour notre travail ont grandement contribué à la réussite de ce projet.

Nous exprimons également notre gratitude aux honorables membres du jury qui ont accepté d'examiner et d'évaluer ce travail. Nous les remercions pour l'intérêt porté à notre mémoire ainsi que pour leurs remarques, observations et recommandations qui contribueront à enrichir notre réflexion et notre expérience.

Pour terminer, nous adressons une pensée reconnaissante à nos familles, à nos proches ainsi qu'à toutes les personnes qui nous ont soutenus, encouragés et accompagnés durant notre parcours universitaire.

À toutes celles et tous ceux qui ont contribué, de près ou de loin, à l'aboutissement de ce travail, nous adressons nos plus sincères remerciements.

Dédicace

Nous dédions ce travail...

À nos parents, À ceux qui ont tout sacrifié pour nous, qui ont veillé tard pour que nous dormions sereinement, qui ont renoncé à leurs rêves pour que les nôtres deviennent réalité. Aucun mot, aucune page, aucun diplôme ne pourra jamais exprimer la profondeur de notre gratitude et de notre amour. Ce travail est avant tout le vôtre.

À notre sœur, Pour sa présence, son soutien discret mais sincère, et pour tous ces petits moments du quotidien qui nous ont donné la force d'avancer. Qu'elle trouve dans ces pages un témoignage de notre affection et de notre fierté de l'avoir à nos côtés.

À toute notre famille et nos proches, À chacun d'entre eux qui, de près ou de loin, nous a encouragés, cru en nous et porté dans ses prières. Leur soutien silencieux a été une lumière dans les moments de doute.

Une dédicace spéciale à notre **cousin Hachemi, notre chère cousine,** et à notre **khalou Chouaib.** Pour leur chaleur, leur humour et leur présence qui a toujours su nous redonner le sourire aux moments où on en avait le plus besoin. Ils savent ce qu'ils représentent pour nous.

À nos amis, nos frères de vie, À ceux qui ont partagé nos fous rires, nos galères, et nos victoires :

Guebrili Aboubaker, Rezig Ahmed, Lakehal Sidahmed, Boukhendeg Aissameddine, Embouazza Habib, Laib Amine, Sebaa Zinou, Razali Abderahman, Hammadi Salah.

Merci d'avoir été là, tout simplement. Certaines amitiés ne s'expliquent pas, elles se vivent, et nous avons eu la chance de vivre la nôtre avec vous.

À toute la promotion Électronique 2026, À cette promotion que nous avons eu l'honneur de partager, avec ses moments de solidarité, ses défis collectifs et ses souvenirs qui resteront gravés. Que chacun trouve sa voie et y brille.

À tous les membres du club scientifique Tex Squad, À cette équipe qui nous a appris que la passion peut devenir un projet, et qu'un projet peut devenir quelque chose de grand. Une dédicace toute particulière à **Tessoumi Mohammed** et **Bettahar Abderrahmane,** pour leur engagement, leur esprit d'équipe et tout ce que nous avons construit ensemble au sein de ce club.

Résumé

Ce mémoire présente la conception et le développement d'un système automatique de contrôle qualité pour les tuiles en terre cuite. Il repose sur la vision par ordinateur et l'apprentissage profond. Le travail se divise en trois approches progressifs, basés sur un ensemble de données de 340 images collectées manuellement dans un atelier réel. La première approche utilise ResNet-50 pour classer les tuiles en trois catégories. Ce cas révèle une confusion visuelle entre les tuiles égorger et les tuiles cassées. La seconde approche résout ce problème en séparant les classes et en améliorant le pipeline d'entraînement. Cela inclut l'augmentation de données, le rééquilibrage des classes, l'utilisation de l'Early Stopping et le Mixed Precision Training. Ce cas atteint une précision de 100% en validation. La troisième approche dépasse une limite importante en adoptant YOLOv8n pour détecter et classer les tuiles simultanément dans des scènes complexes. Il obtient un mAP50 de 99.5%. Le modèle final est déployé sur un système embarqué NVIDIA Jetson Nano avec une webcam USB, et il constitue un prototype fonctionnel de contrôle qualité en temps réel.

Mots-clés : contrôle qualité, vision par ordinateur, apprentissage profond, classification d'images, détection d'objets, ResNet-50, Transfer Learning, YOLOv8, Multi-Task Learning, système embarqué, NVIDIA Jetson Nano.

Abstract

This thesis presents the design and development of an automatic quality control system for terracotta roof tiles that uses computer vision and deep learning techniques. The work consists of three case studies based on a dataset of 340 images collected in a real production workshop. The first case study uses a ResNet-50 classification model with three classes, showing confusion between cut tiles and broken tiles. The second case study tackles this issue by separating the classes and improving the training process through data augmentation, class rebalancing, Early Stopping, and Mixed Precision Training, which leads to 100% validation accuracy. The third case study addresses a more significant limitation by using YOLOv8n for detecting and classifying tiles in complex scenes, achieving a mAP50 of 99.5%. The final model is implemented on an NVIDIA Jetson Nano embedded system with a USB webcam, creating a functional real-time quality control prototype.

Keywords: quality control, computer vision, deep learning, image classification, object detection, ResNet-50, Transfer Learning, YOLOv8, Multi-Task Learning, embedded system, NVIDIA Jetson Nano, terracotta roof tile.

ملخص

تقدم هذه المذكرة تصميم وتطوير نظام تلقائي لمراقبة جودة قرميد الفخار، يعتمد على تقنيات الرؤية الحاسوبية والتعلم العميق. يركز العمل على ثلاث حالات دراسية متتالية، بدءاً من مجموعة بيانات تتضمن 340 صورة جمعت يدوياً في ورشة إنتاج حقيقية. تعتمد الحالة الأولى على نموذج تصنيف ResNet-50 بثلاث فئات، وقد كشفت عن تشابه بصري بين القرميد المقطوع والقرميد المكسور. تعالج الحالة الثانية هذه المشكلة من خلال الفصل بين الفئات وتحسين نظام التدريب. يتم ذلك عبر تعزيز تحويلات البيانات، وإعادة توازن الفئات، والإيقاف المبكر، والتدريب بدقة مختلطة، مما يؤدي إلى دقة تحقق تبلغ 100%. تتجاوز الحالة الثالثة حدوداً أكثر أهمية، حيث تعتمد YOLOv8n للكشف عن القرميد وتصنيفه في نفس الوقت ضمن مشاهد معقدة، محققةً mAP50 بنسبة 99.5%. يُنشر النموذج النهائي على منصة NVIDIA Jetson Nano المدمجة مع كاميرا USB، ليشكل نموذجاً أولياً وظيفياً لمراقبة الجودة في الوقت الحقيقي.

الكلمات المفتاحية: مراقبة الجودة، الرؤية الحاسوبية، التعلم العميق، تصنيف الصور، كشف الأجسام، ResNet-50، التعلم بالنقل، YOLOv8، التعلم متعدد المهام، الأنظمة المدمجة، NVIDIA Jetson Nano.

Sommaire

Remerciements.....	3
Dédicace.....	4
Résumé	5
Sommaire	6
Liste d’Abréviations	9
Liste des Tableaux.....	10
Liste des figures	11
Introduction	13
Chapitre 1 : États de l’art et Revue de Littérature	15
Introduction.....	16
1.1 Le contrôle qualité dans l’industrie de la poterie	16
1.2 La vision par ordinateur pour la détection de défauts.....	17
1.3 Travaux existants.....	18
1.3.1 Approches analogiques et manuelles.....	18
1.3.2 Approches automatiques classiques	18
1.3.3 Approches basées sur l’intelligence artificielle	19
1.4 La détection d’objets, de la classification à la localisation.....	20
1.5 YOLOv8 et le Mutli-Task Learning	21
1.6 Synthèse et positionnement du travail	21
Conclusion	22
Chapitre 2 : Fondements Théoriques	23
Introduction.....	24
2.1 Les réseaux de neurones convolutifs (CNN).....	24
2.1.1 Principe général.....	24
2.1.2 L’architecture d’un CNN	25
2.1.3 Les fonctions d’activation.....	27
2.2 L’architecture ResNet-50 et le Transfer Learning	28
2.2.1 Le problème de la dégradation et les connexions résiduelles	28
2.2.2 Le Transfer Learning et le fine-tuning	28
2.3 L’architecture YOLOv8.....	29
2.3.1 De la classification à la détection	29
2.3.2 Spécificités de YOLOv8	30

2.3.3 La version YOLOv8n	31
2.4 Le Multi-Task Learning	31
2.5 Techniques d'optimisation utilisées	32
Conclusion	33
Chapitre 3 : Méthodologie et Préparation du Dataset.....	34
Introduction.....	35
3.1 Méthodologie de travail	35
3.2 Collecte manuelle des données.....	36
3.2.1 Conditions de prise de photo	37
3.2.2 Composition du dataset	37
3.3 Analyse et distribution des classes.....	38
3.4 Préparation selon les approches	39
3.4.1 Split train / validation / test	39
3.4.2 Redimensionnement et normalisation.....	40
3.4.3 Augmentation des données	40
3.5 L'annotation avec Roboflow (Approche 3).....	41
Conclusion	43
Chapitre 4 : Expérimentations et Résultats.....	44
Introduction.....	45
4.1 Approche 1 (Classification ResNet-50, 3 classes)	45
4.1.1 Formulation du problème et architecture	45
4.1.2 Conditions d'entraînement	46
4.1.3 Résultats et analyse.....	46
4.2 Approche 2 (Classification ResNet-50, 4 classes)	48
4.2.1 Motivation et reformulation du problème.....	48
4.2.2 Architecture et stratégie de fine-tuning.....	49
4.2.3 Améliorations du pipeline d'entraînement.....	49
4.2.4 Résultats et analyse.....	49
4.3 Approche 3 (Classification et Détection YOLOv8n)	52
4.3.1 Motivation et changement de paradigme	52
4.3.2 Préparation du dataset et configuration de l'entraînement.....	52
4.3.3 Résultats et analyse.....	53
4.4 Comparaison et analyse des trois approches.....	55
4.5 Déploiement sur NVIDIA Jetson Nano.....	56
4.5.1 Le déploiement embarqué	56
4.5.2 Présentation de la plateforme	57

4.5.3 Résultats en conditions réelles.....	58
Conclusion	59
Conclusion	60
Bibliographie	62

Liste d'Abbreviations

Abréviation	Signification
IA	Intelligence Artificielle
YOLO	You Only Look Once
SVM	Support Vector Machine
HOG	Histogram of Oriented Gradient
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
SSD	Single Shot MultiBox Detector
GPU	Graphics Processing Unit
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
MTL	Multi-Task Learning
RGB	Red Green Blue
StepLR	Step Learning Rate Scheduler
AMD	Advanced Micro Devices
RAM	Random Access Memory
CPU	Central Processing Unit
ARM	Advanced RISC Machines
USB	Universal Serial Bus
FPS	Frames Per Second
eMMC	embedded MultiMediaCard
ONNX	Open Neural Network Exchange

Liste des Tableaux

Tableau 1.1 – Détection d’objets vs reconnaissance d’objets

Tableau 2.1 – Tableau comparatif des variantes YOLOv8

Tableau 3.1 – Tableau récapitulatif des paramètres de préparation par approche

Tableau 4.1 – Métriques de classification par classe sur le jeu de test, Approche 1

Tableau 4.2 – Métriques de classification par classe sur le jeu de validation, Approche 2

Tableau 4.3 – Métriques de détection par classe, Approche 3

Tableau 4.4 – Tableau de comparaison entre les modèles des 3 approches

Liste des figures

Figure 1.1 - Principe de fonctionnement de la vision par ordinateur

Figure 1.2 - Conditions d'éclairage et de vision courantes

Figure 1.3 - L'évolution de la détection d'objets

Figure 1.4 - NVIDIA Jetson Nano Developer Kit 4Go

Figure 2.1 - Schéma d'un CNN complet

Figure 2.2 - Exemple d'image d'entrée d'un CNN

Figure 2.3 - Exemple de convolution

Figure 2.4 - Exemple de pooling

Figure 2.5 - Exemple de Flattening

Figure 2.6 - Architecture minimal d'un CNN

Figure 2.7 - Architecture ResNet-50 avec blocs résiduels

Figure 2.8 - Stratégie de fine-tuning : couches gelées vs couches libres

Figure 2.9 - Principe de grille et de bounding box dans YOLO

Figure 2.10 - Architecture générale de YOLOv8

Figure 2.11 - Courbe illustrative de l'effet de l'Early Stopping

Figure 3.1 - Schéma général de la méthodologie du travail

Figure 3.2 - Exemple d'images pour chacune des quatre classes

Figure 3.3 - Distribution des classes du dataset

Figure 3.4 - Comparaison visuelle entre tuile égorgée et tuile simple

Figure 3.5 - Interface Roboflow lors de l'annotation des bounding boxes

Figure 3.6 - Exemples d'ensemble d'images de la dataset annotées

Figure 4.1 - Architecture ResNet et stratégie de gel

Figure 4.2 – Résultats positifs pour l'approche 1.

Figure 4.3 - Courbes de loss et d'accuracy sur 30 époques, Approche 1

Figure 4.4 - Matrice de confusion sur le jeu de test, Approche 1

Figure 4.5 - Résultats sur des images inédites, et une photo mal classée

Figure 4.6 – Résultats positif pour l’approche 2.

Figure 4.7 - Courbes de loss et d’accuracy, Approche 2

Figure 4.8 - Matrice de confusion sur le jeu de test, Approche 2

Figure 4.9 - Exemple d’images mixte classée comme pas_de_tuile

Figure 4.10 - Courbes mAP50 et mAP50-95 en fonction des époques, Approche 3

Figure 4.11 - Courbes de loss d’entraînement et de validation, Approche 3

Figure 4.12 - Matrice de confusion normalisée du modèle finale YOLOv8n

Figure 4.13 - Exemples d’images inédites avec bounding boxes et classification

Figure 4.14 - NVIDIA Jetson Nano Developer kit 4GB avec webcam USB

Figure 4.15 - Captures d’écran du système en fonctionnement sur la plateforme embarquée

Introduction

L'industrie de la poterie, bien qu'ancrée dans des savoir-faire artisanaux parfois séculaires, n'échappe pas aux exigences croissantes de la production moderne. La tuile en terre cuite reste un matériau de construction largement utilisé dans de nombreuses régions du monde, notamment en Afrique du Nord et dans le bassin méditerranéen. Sa fabrication génère inévitablement une proportion non négligeable de pièces défectueuses : tuiles fissurées, cassées, ou mal taillées, qui doivent être écartées avant toute mise sur le marché. Aujourd'hui encore, ce tri repose dans la plupart des ateliers sur l'œil humain, avec toute la fatigue, la variabilité et les coûts que cela implique.

Face à ce constat, plusieurs systèmes de contrôle qualité automatisés ont été proposés dans la littérature pour l'inspection de surfaces céramiques et de matériaux similaires. Boukouvalas et al, ont développé dès 1999 un système d'analyse de texture par descripteurs statistiques pour la détection de défauts sur carreaux céramiques, posant les premières bases d'une inspection automatisée [1]. Par la suite, Xie et Mirmehdi ont proposé un système basé sur les filtres de Gabor et les exemplaires de texture, capable de détecter des anomalies sur des surfaces à structure répétitive [3]. Plus récemment, Shahriari et al, ont combiné des descripteurs HOG avec un classifieur SVM pour inspecter des surfaces texturées industrielles, obtenant des résultats supérieurs aux approches purement basées sur des filtres [4]. Malgré leurs apports respectifs, ces systèmes partagent une limite commune : ils reposent sur des caractéristiques définies manuellement, ce qui les rend peu adaptés à la variabilité naturelle des conditions de production réelles.

Le contrôle qualité manuel présente par ailleurs des limites évidentes : il est lent, subjectif, et difficilement scalable dès lors que les cadences de production augmentent. Dans ce contexte, l'automatisation par vision artificielle et apprentissage profond apparaît comme une réponse naturelle et techniquement mûre. Toutefois, la mise en œuvre concrète de tels systèmes soulève des problématiques techniques importantes. La première concerne la donnée elle-même : aucun dataset public de tuiles en terre cuite ne couvrant les catégories spécifiques de ce travail n'étant disponible, il a fallu constituer un dataset from scratch, avec les contraintes que cela implique en termes de volume, de variabilité et de déséquilibre entre classes. La seconde problématique est d'ordre visuel : la confusion entre une tuile intentionnellement égorgée et une tuile cassée est difficile à résoudre par classification globale d'image, car les deux présentent des bords similaires

dont la distinction requiert une analyse fine de la morphologie locale. Enfin, la troisième problématique concerne la localisation : un modèle de classification est incapable de détecter plusieurs objets dans une même scène, ce qui limite son applicabilité dans un environnement industriel réel.

Le travail présenté ici suit une démarche progressive à travers trois approches. Partant d'un dataset de 340 photographies collectées manuellement, un premier modèle ResNet-50 a été entraîné sur trois classes, révélant une confusion persistante entre tuiles égorger et tuiles cassées. Un second modèle, avec une séparation explicite des classes et des techniques d'optimisation renforcées, a atteint d'excellentes performances en validation, mais s'est heurté à l'impossibilité de localiser les tuiles dans des scènes complexes. C'est ce constat qui a motivé le passage à YOLOv8n, capable de détecter et classifier simultanément les tuiles, et finalement déployé sur un système embarqué NVIDIA Jetson Nano pour une inférence en temps réel.

La mémoire est organisée en quatre chapitres principaux : un état de l'art des travaux existants, les fondements théoriques, la méthodologie et préparation du dataset, les trois approches avec comparaison et déploiement, et enfin la conclusion avec les perspectives d'amélioration.

Chapitre 1 : États de l'art et Revue de Littérature



Introduction

Le contrôle qualité industriel a connu, depuis les années 1990, une transformation progressive mais profonde. Les méthodes traditionnelles, fondées sur l'inspection humaine ou sur des capteurs mécaniques simples, ont peu à peu cédé la place à des systèmes automatisés capables de traiter un grand volume de pièces avec une constance que l'opérateur humain ne peut garantir sur la durée. C'est dans ce mouvement général que s'inscrit le présent travail. Avant d'exposer les choix techniques qui ont guidé notre démarche, il convient de parcourir les principales contributions de la littérature, afin de situer notre approche par rapport à l'existant et d'en justifier les orientations.

1.1 Le contrôle qualité dans l'industrie de la poterie

La poterie et la céramique constituent l'un des secteurs industriels où le contrôle qualité reste encore largement manuel dans de nombreux pays. La nature même du matériau, fragile, hétérogène, sensible aux variations de cuisson, rend la détection de défauts particulièrement délicate. Les défauts rencontrés sont variés : fissures superficielles ou profondes, éclats, déformations géométriques, ou encore irrégularités de surface liées au processus de fabrication.

Dès les années 2000, plusieurs travaux ont cherché à automatiser cette inspection. Boukouvalas et al, ont proposé un système d'analyse de texture pour la détection de défauts sur carreaux céramiques, en exploitant des descripteurs statistiques calculés sur des images en niveaux de gris [1]. Cette approche, bien que pionnière, restait limitée aux défauts de surface réguliers et nécessitait des conditions d'éclairage strictement contrôlées. Par la suite, d'autres travaux ont eu recours aux transformées de Fourier et aux filtres de Gabor pour caractériser les textures anormales, avec des résultats encourageants sur des surfaces homogènes [2].

Toutefois, ces méthodes classiques partagent une faiblesse commune : elles reposent sur des caractéristiques extraites à la main (hand-crafted features), ce qui les rend peu généralisables à des conditions de prise de vue variables ou à des défauts de morphologie irrégulière. C'est précisément ce type de situation que l'on rencontre dans un atelier réel, où l'éclairage n'est pas uniforme et où les tuiles présentent des formes de cassures très diverses.

1.2 La vision par ordinateur pour la détection de défauts

La vision par ordinateur s'est imposée comme le cadre naturel pour aborder les problèmes d'inspection automatique. Dans ce domaine, les premières approches opérationnelles reposaient sur le traitement d'image classique : seuillage, détection de contours, analyse morphologique. Ces techniques ont été appliquées avec un certain succès à des produits industriels aux géométries bien définies, comme les circuits imprimés ou les pièces métalliques usinées.

Pour les matériaux à surface irrégulière comme la céramique ou la terre cuite, les résultats étaient moins satisfaisants. Xie et al, ont montré que les méthodes basées sur les filtres de Gabor permettaient d'atteindre des taux de détection corrects sur des textures régulières, mais se dégradaient rapidement en présence de variations naturelles du matériau [3]. Cette limite a poussé la communauté à explorer des représentations plus riches et plus flexibles.

L'introduction des méthodes d'apprentissage automatique classique, SVM, forêts aléatoires, AdaBoost, a constitué une première avancée notable. Shahriari et al, ont par exemple combiné des descripteurs HOG (Histogram of Oriented Gradients) avec un classifieur SVM pour détecter des défauts sur des surfaces texturées, obtenant des performances supérieures aux approches purement basées sur des filtres [4]. Néanmoins, la qualité de ces systèmes restait fortement dépendante du choix des descripteurs.

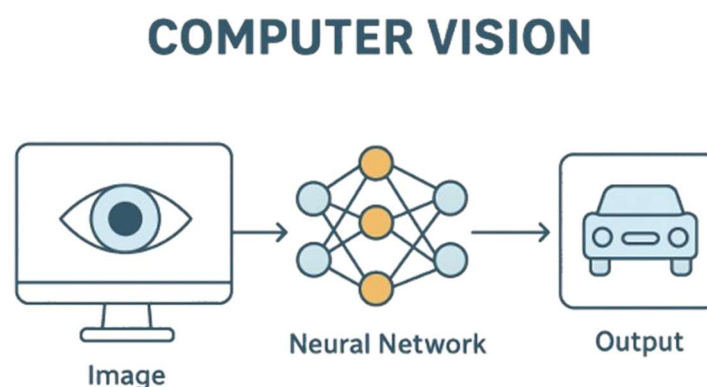


Figure 1.1 – Principe de fonctionnement de la vision par ordinateur [5].

1.3 Travaux existants

Le domaine du contrôle qualité automatique a connu trois grandes générations de travaux, chacune apportant des améliorations concrètes par rapport à la précédente. Nous présentons ici trois travaux représentatifs, sélectionnés pour leur pertinence directe avec la problématique de ce mémoire.

1.3.1 Approches analogiques et manuelles

L'inspection visuelle manuelle a longtemps constitué la seule méthode disponible pour le contrôle qualité dans l'industrie de la poterie et de la céramique. Un opérateur humain examine chaque pièce individuellement en s'appuyant sur son expérience pour détecter les défauts visibles : fissures, éclats, déformations ou irrégularités de surface. Cette approche souffre cependant de limitations évidentes, notamment la fatigue de l'opérateur, la subjectivité du jugement et la difficulté à maintenir une cadence d'inspection élevée sur la durée.

C'est dans ce contexte que Boukouvalas et al, ont proposé l'un des premiers systèmes d'inspection automatisée de carreaux céramiques [1]. Leur dispositif repose sur une caméra CCD placée au-dessus d'un convoyeur industriel, couplée à un éclairage structuré, et utilise des descripteurs statistiques de texture calculés sur des images en niveaux de gris pour détecter les anomalies de surface. Bien que pionnier, ce système nécessitait des conditions d'acquisition strictement contrôlées et restait limité aux défauts réguliers et répétitifs, montrant ses limites face à des défauts de morphologie irrégulière comme les cassures accidentelles.

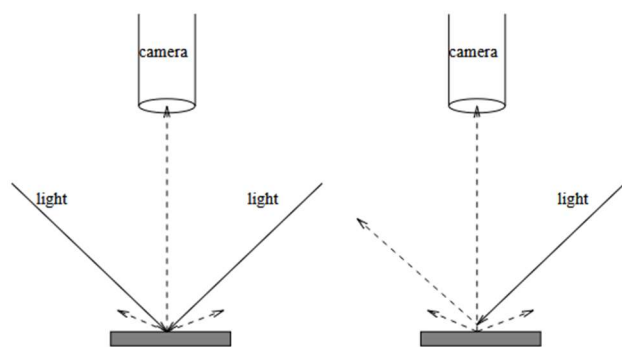


Figure 1.2 – Conditions d'éclairage et de vision courantes [1].

1.3.2 Approches automatiques classiques

Face aux limites des méthodes analogiques, la communauté scientifique s'est tournée vers des approches combinant le traitement d'image automatisé et l'apprentissage

automatique classique. Ces méthodes extraient des caractéristiques pertinentes à partir des images, puis utilisent un classifieur pour distinguer automatiquement les pièces conformes des pièces défectueuses.

Oztemel et Kayabasi ont développé dans ce cadre un système de contrôle qualité automatique pour une ligne de production de carreaux céramiques en conditions industrielles réelles [7]. Leur pipeline repose sur une segmentation par seuillage adaptatif pour isoler les zones défectueuses, suivie d'un réseau de neurones multicouche pour classifier le type de défaut détecté. Testé sur une ligne industrielle réelle en Turquie, le système atteignait un taux de reconnaissance de 89%. Cependant, ses performances se dégradaient en présence de variations d'éclairage ou de changements dans les propriétés visuelles du matériau, révélant la fragilité inhérente des descripteurs extraits manuellement face à la variabilité des conditions réelles.

1.3.3 Approches basées sur l'intelligence artificielle

L'avènement des réseaux de neurones convolutifs profonds a marqué une rupture fondamentale dans le domaine du contrôle qualité industriel. Contrairement aux approches précédentes, les CNN apprennent automatiquement les représentations visuelles les plus discriminantes directement à partir des données brutes, sans qu'il soit nécessaire de définir explicitement les caractéristiques des défauts à détecter.

Feng et Lu, ont illustré concrètement ce potentiel en proposant un système de détection de défauts sur tuiles céramiques basé sur un CNN profond, entraîné sur un dataset de plus de 5000 images collectées en conditions réelles [8]. Leur architecture s'appuie sur un backbone VGG-16 pré-entraîné sur ImageNet et adapté par Transfer Learning, capable d'identifier simultanément plusieurs types de défauts : fissures, taches, déformations géométriques et irrégularités de glaçure. Déployé sur une ligne de production active, le système atteignait une précision de 96.4% avec un temps d'inférence compatible avec les cadences industrielles, surpassant largement les approches classiques évaluées sur le même jeu de test.

1.4 La détection d'objets, de la classification à la localisation

La classification d'images répond à une question simple : quelle est la catégorie dominante dans cette image ? Mais dans de nombreuses situations industrielles, plusieurs objets sont présents simultanément dans le champ de la caméra. Il devient alors nécessaire non seulement de reconnaître les objets, mais aussi de les localiser précisément, c'est le problème de la détection d'objets.

Les premières architectures de détection performantes reposaient sur une approche en deux étapes. R-CNN, proposé par Girshick et al, générait d'abord des régions candidates puis les classifiait indépendamment [11]. Cette méthode, bien qu'efficace, était trop lente pour une utilisation temps réel. Ses successeurs, Fast R-CNN puis Faster R-CNN, ont progressivement réduit ce coût computationnel en partageant les calculs de convolution entre les régions [12] [13].

C'est dans ce contexte que les architectures à passage unique ont pris de l'importance. SSD (Single Shot MultiBox Detector) et la famille YOLO ont montré qu'il était possible d'atteindre des performances comparables en une seule passe du réseau, avec un gain de vitesse considérable [14] [15].

Critère	Détection d'objets (Object Detection)	Reconnaissance d'objets (Object Recognition)
Définition	Identifie et localise les objets dans une image ou vidéo.	Identifie et classe les objets.
Objectif	Déterminer où se trouvent les objets dans une image ou une vidéo	Déterminer ce qu'est l'objet.
Résultat	Fournit des boîtes englobantes (bounding boxes) autour des objets détectés.	Attribue une étiquette ou nom à l'objet détecté.
Exemple	Une caméra de surveillance détecte la présence de visage dans une foule sans identifier les personnes	Un téléphone se déverrouille en reconnaissant le visage du propriétaire à partir de données faciales enregistrées.

Tableau 1.1 – Détection d'objets vs reconnaissance d'objets [16].

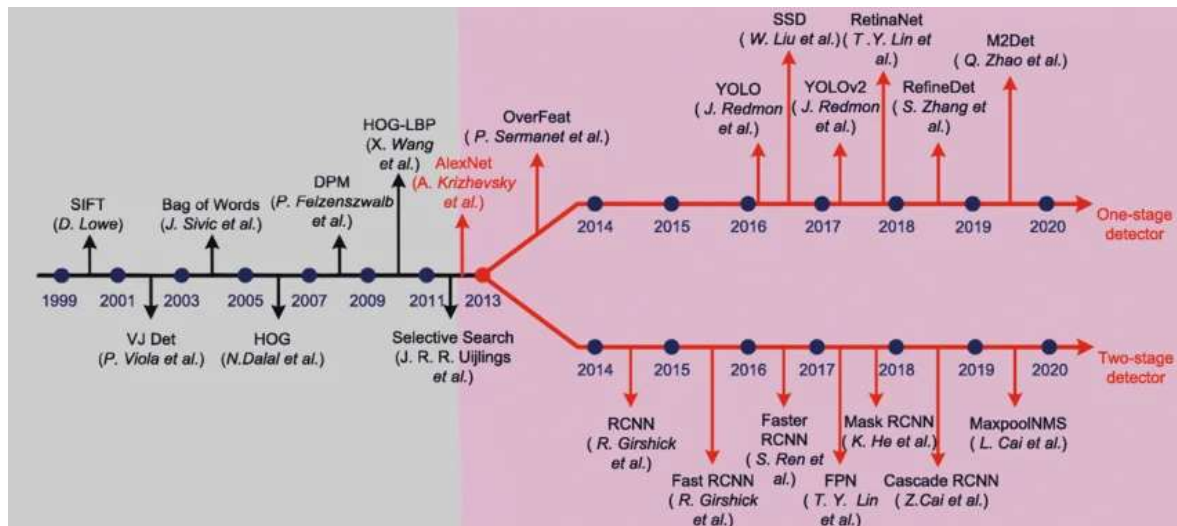


Figure 1.3 – L'évolution de la détection d'objets [17].

1.5 YOLOv8 et le Mutli-Task Learning

La famille YOLO a connu plusieurs révisions majeures depuis sa version originale. YOLOv8, publié par Jocher et al, en 2023, représente l'état de l'art actuel de cette famille [18]. Il introduit une architecture anchor-free, une tête de détection découplée, et un pipeline d'entraînement modernisé qui le rend plus facile à adapter à des tâches variées.

Par ailleurs, le Multi-Task Learning, qui consiste à entraîner un même réseau à accomplir plusieurs tâches simultanément, a démontré des bénéfices réguliers dans la littérature. Caruana a montré que le partage de représentations entre tâches connexes permettait souvent d'améliorer les performances sur chacune d'elles [19]. Dans le cadre de la détection d'objets, cela se traduit naturellement par la combinaison de la localisation et de la classification au sein d'un même réseau, ce qui est précisément ce que réalise YOLOv8 dans notre troisième approche.

1.6 Synthèse et positionnement du travail

La revue de la littérature présentée dans ce chapitre permet de dégager plusieurs constats. D'abord, les méthodes classiques de traitement d'image ont montré leurs limites face à la variabilité des défauts sur matériaux hétérogènes comme la terre cuite. Ensuite, les CNN ont constitué une avancée majeure, mais leur application directe à la classification globale d'image atteint ses propres limites dès lors que la scène est complexe. Enfin, les architectures de détection d'objets de type YOLO, couplées au déploiement embarqué, représentent aujourd'hui la solution la plus aboutie pour un système de contrôle qualité opérationnel en temps réel.

Notre travail ne prétend pas rivaliser avec des systèmes industriels développés sur des années avec des milliers d'images. Il cherche plutôt à montrer, à travers une démarche progressive et rigoureuse, comment ces outils peuvent être mis en œuvre de façon cohérente sur un problème concret, avec des moyens limités.

Conclusion

Ce premier chapitre a permis de parcourir l'évolution des approches utilisées pour l'inspection automatique de surfaces industrielles, depuis les méthodes de traitement d'image classiques jusqu'aux architectures de détection d'objets modernes. Chaque génération de méthodes a apporté des améliorations réelles, mais aussi mis en lumière de nouvelles limites, ce qui explique la trajectoire adoptée dans ce travail. Le chapitre suivant posera les bases théoriques nécessaires à la compréhension des modèles utilisés.

Chapitre 2 : Fondements Théoriques



Introduction

Les choix techniques effectués au cours de ce travail reposent sur un ensemble de concepts et d'outils issus du domaine de l'apprentissage profond. Ce chapitre a pour objectif d'exposer ces fondements de manière suffisamment précise pour que les décisions prises dans les chapitres suivants soient compréhensibles et justifiées. On abordera successivement les réseaux de neurones convolutifs, l'architecture ResNet-50 et le principe du Transfer Learning, l'architecture YOLOv8, le Multi-Task Learning, puis les principales techniques d'optimisation utilisées lors de l'entraînement.

2.1 Les réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs constituent une architecture fondamentale en vision par ordinateur. Leur capacité à extraire automatiquement des caractéristiques pertinentes à partir des images explique leur large adoption dans les systèmes modernes d'intelligence artificielle.

2.1.1 Principe général

Un réseau de neurones convolutif est un type particulier de réseau de neurones artificiels, conçu pour traiter des données organisées sous forme de grilles, les images en étant l'exemple le plus courant. L'idée centrale repose sur l'opération de convolution : plutôt que de connecter chaque neurone à l'ensemble des pixels de l'image d'entrée, on applique un filtre local qui se déplace sur l'image et produit une carte d'activation (feature map). Cette approche réduit considérablement le nombre de paramètres à apprendre, tout en capturant les structures locales comme les contours, les textures ou les formes.

Un CNN typique est composé de plusieurs types de couches qui s'enchaînent. Les couches de convolution extraient des caractéristiques de plus en plus abstraites au fur et à mesure que l'on avance dans le réseau. Les couches de pooling, généralement de type max pooling, réduisent la dimension spatiale des cartes d'activation, ce qui rend le réseau plus robuste aux petites translations et diminue le coût computationnel. Enfin, les couches entièrement connectées (fully connected) en fin de réseau combinent ces caractéristiques pour produire la décision finale de classification.

LeCun et al, ont été parmi les premiers à démontrer l'efficacité de cette architecture sur la reconnaissance de chiffres manuscrits avec le réseau LeNet-5, posant ainsi les bases conceptuelles des CNN modernes [21]. C'est cependant la publication d'AlexNet en 2012 qui a véritablement lancé l'ère des CNN profonds pour la vision par ordinateur [6].

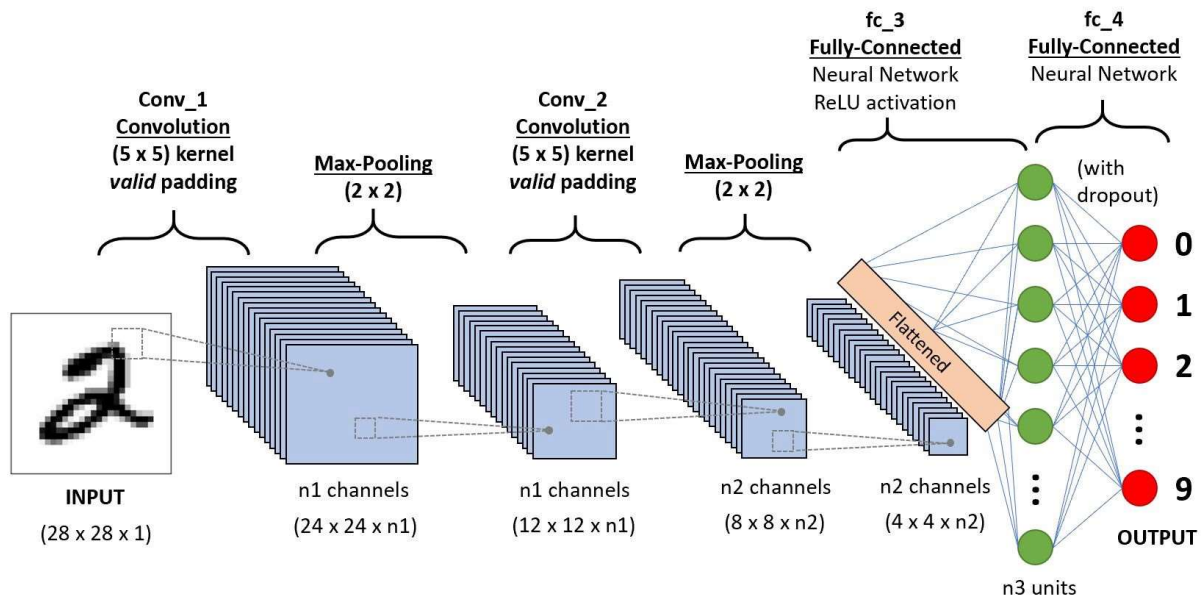


Figure 2.1 – Schéma d'un CNN complet [22].

2.1.2 L'architecture d'un CNN

Les principaux composants de l'architecture d'un réseau de neurones convolutifs (CNN) sont les suivants :

❖ Image d'entrée (Input Image) :

L'image d'entrée constitue la donnée initiale fournie au réseau. Elle est représentée sous forme de matrice numérique composée de pixels, où chaque valeur traduit une intensité lumineuse ou une information de couleur. Cette représentation matricielle permet au réseau de traiter automatiquement les informations visuelles et d'en extraire des caractéristiques pertinentes.

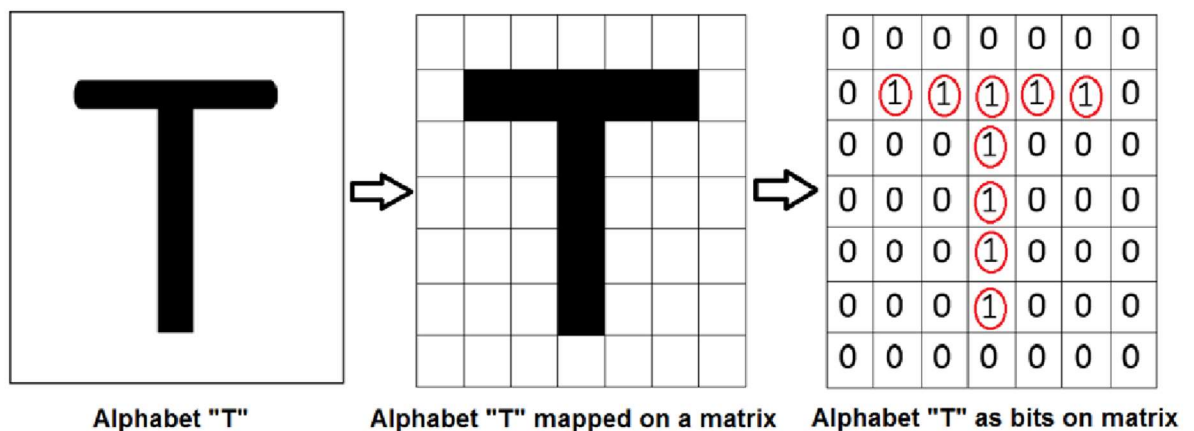


Figure 2.2 – Exemple d'image d'entrée d'un CNN [23].

❖ Couche de convolution (Convolutional Layer) :

La couche de convolution représente l'élément fondamental d'un CNN. Elle applique plusieurs filtres, appelés noyaux de convolution, sur différentes régions de l'image afin d'extraire des caractéristiques locales telles que les contours, les textures ou les formes. Les caractéristiques apprises deviennent progressivement plus complexes au fur et à mesure de la profondeur du réseau.

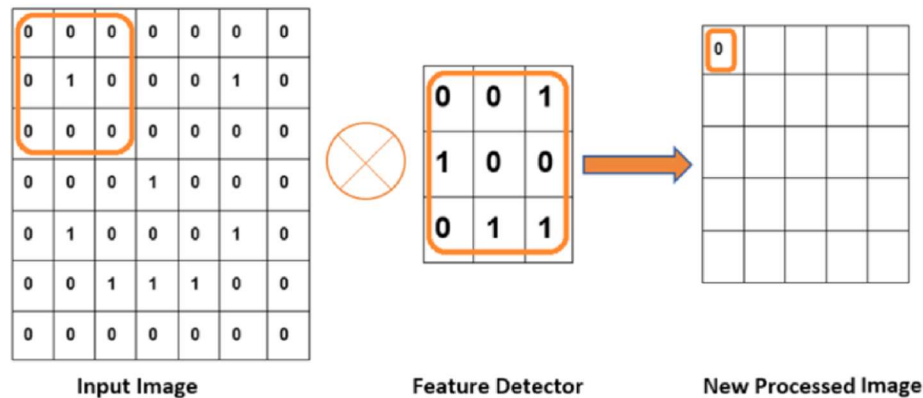


Figure 2.3 – Exemple de convolution [23].

❖ Couche de pooling (Pooling Layer):

La couche de pooling est utilisée pour réduire la dimension spatiale des cartes de caractéristiques produites par la convolution. Cette opération permet de diminuer le nombre de paramètres et le coût computationnel, tout en conservant les informations les plus importantes. Le max pooling est la méthode la plus couramment utilisée, car elle sélectionne les valeurs les plus significatives de chaque région analysée.

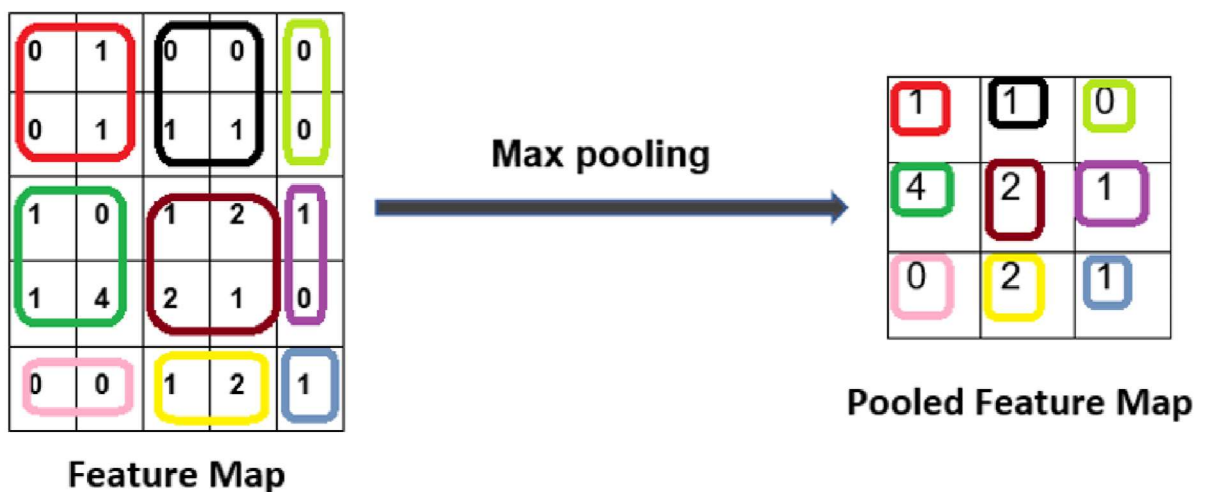


Figure 2.4 – Exemple de pooling [23].

❖ **Aplatissement (Flattening):**

L'étape d'aplatissement consiste à transformer les cartes de caractéristiques multidimensionnelles en un vecteur unidimensionnel. Cette transformation permet de connecter les données extraites aux couches entièrement connectées du réseau, lesquelles sont responsables de la prise de décision finale, notamment dans les tâches de classification.

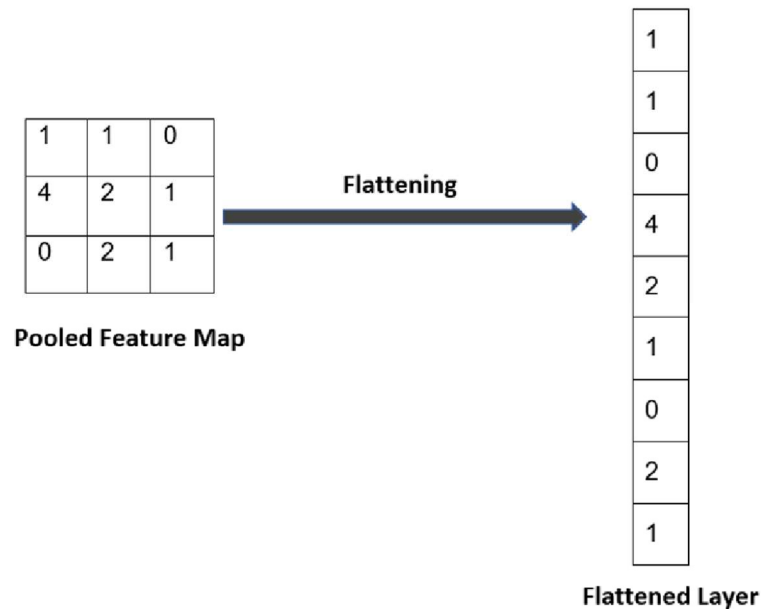


Figure 2.5 – Exemple de Flattening [23].

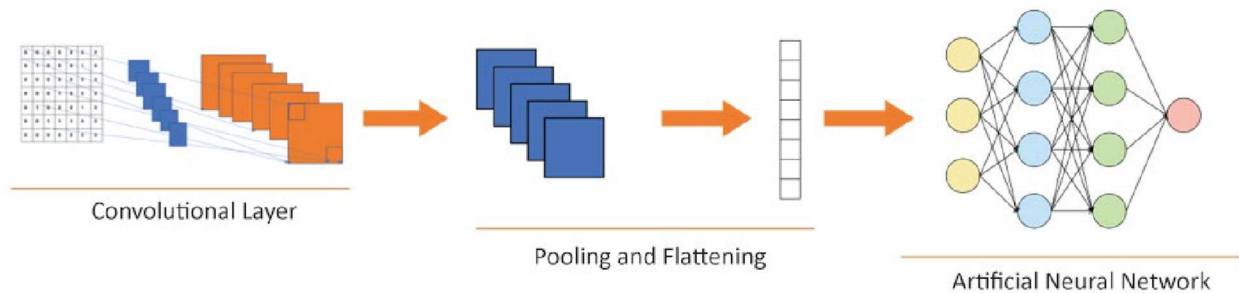


Figure 2.6 – Architecture minamal d'un CNN [23].

2.1.3 Les fonctions d'activation

Entre chaque couche, une fonction d'activation introduit une non-linéarité indispensable à la capacité d'apprentissage du réseau. La fonction ReLU (Rectified Linear Unit), définie par $f(x) = \max(0, x)$, s'est imposée comme le choix standard dans la quasi-totalité des architectures modernes. Elle présente l'avantage d'être peu coûteuse en calcul et de ne pas souffrir du problème de disparition du gradient qui affectait les fonctions sigmoïde et tangente hyperbolique utilisées auparavant [24].

2.2 L'architecture ResNet-50 et le Transfer Learning

ResNet-50 est l'une des architectures de réseaux de neurones profonds les plus utilisées en vision par ordinateur. Son efficacité repose principalement sur l'utilisation des connexions résiduelles, qui facilitent l'apprentissage dans les réseaux de grande profondeur.

2.2.1 Le problème de la dégradation et les connexions résiduelles

À mesure que les chercheurs ont cherché à construire des réseaux plus profonds pour améliorer les performances, un phénomène inattendu est apparu : au-delà d'un certain nombre de couches, les performances se dégradaient, même sur les données d'entraînement. Ce n'était pas un problème de surapprentissage, mais bien une difficulté d'optimisation liée à la profondeur elle-même.

He et al, ont proposé en 2016 une solution élégante à ce problème : les connexions résiduelles (skip connections) [25]. L'idée est simple, au lieu d'apprendre directement une transformation $H(x)$, on demande au réseau d'apprendre le résidu $F(x) = H(x) - x$, et l'on ajoute l'entrée x à la sortie de chaque bloc. Cette modification permet au gradient de se propager plus facilement à travers les couches profondes lors de la rétropropagation. ResNet-50 est la variante à 50 couches de cette famille, qui offre un bon équilibre entre précision et coût computationnel.

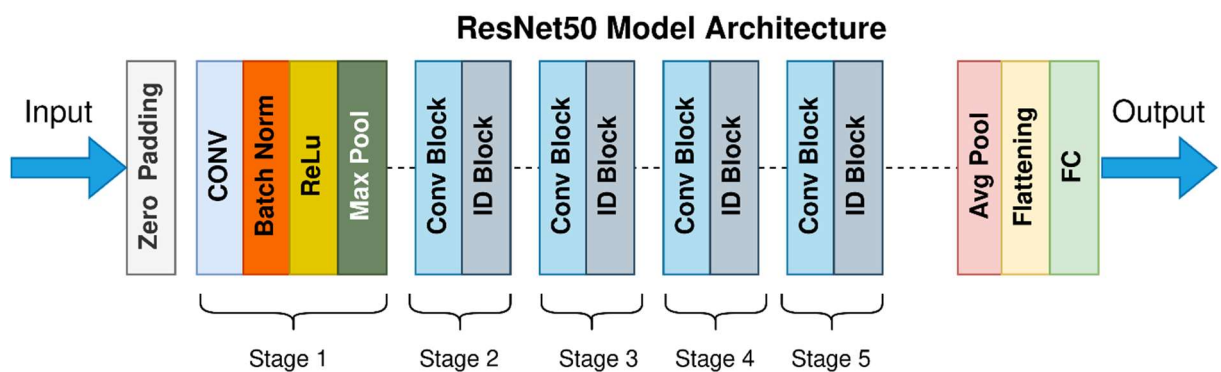


Figure 2.7 – Architecture ResNet-50 avec blocs résiduels [26].

2.2.2 Le Transfer Learning et le fine-tuning

Entraîner un réseau comme ResNet-50 from scratch nécessite des millions d'images et plusieurs jours de calcul sur GPU. Dans la grande majorité des applications industrielles, un tel volume de données n'est pas disponible. Le Transfer Learning répond à ce problème en réutilisant un réseau déjà entraîné sur une tâche source, généralement ImageNet, et en l'adaptant à une tâche cible différente [9].

En pratique, cela se traduit par deux stratégies complémentaires. La première consiste à geler (freeze) les premières couches du réseau, dont les filtres ont appris des représentations génériques comme les contours et les textures, et à ne laisser apprendre que les couches supérieures plus spécialisées. La seconde, le fine-tuning, consiste à dégeler progressivement certaines couches profondes pour les adapter aux spécificités visuelles du nouveau domaine. Dans notre travail, nous avons gelé les couches layer1 et layer2 de ResNet-50, et laissé libres layer3 et layer4.

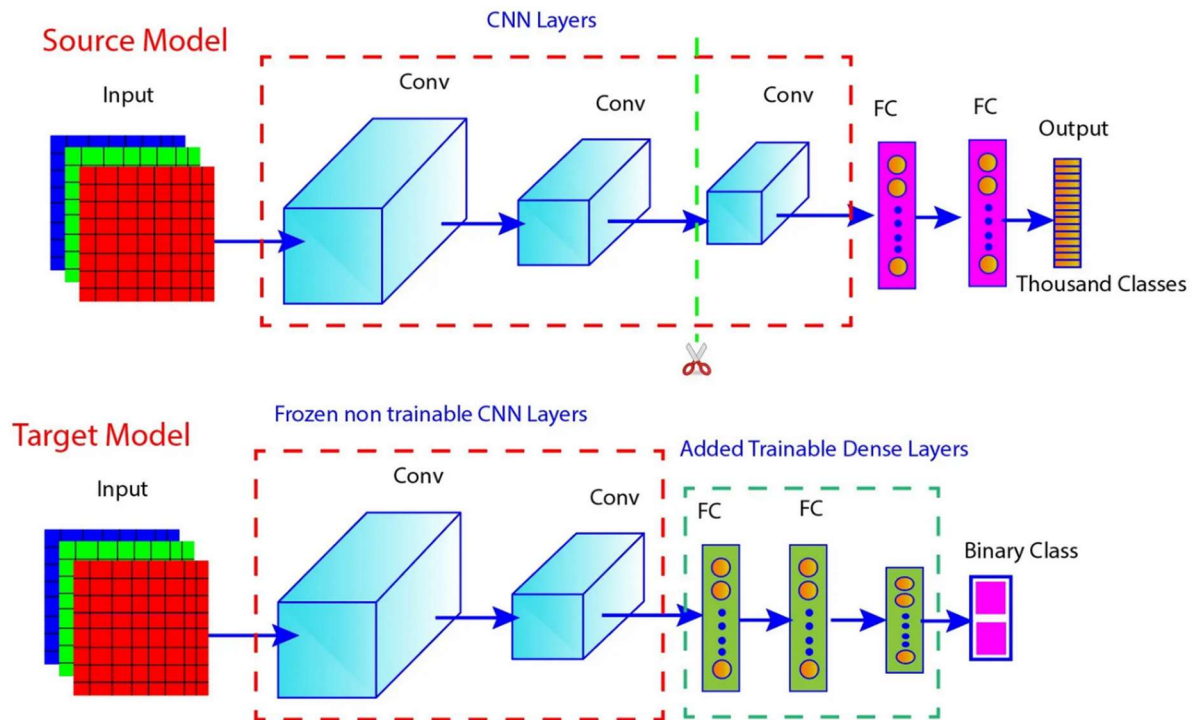


Figure 2.8 – Stratégie de fine-tuning : couches gelées vs couches libres [27].

2.3 L'architecture YOLOv8

YOLOv8 fait partie des architectures modernes de détection d'objets conçues pour offrir un bon compromis entre précision et rapidité d'exécution. Cette architecture introduit plusieurs améliorations visant à optimiser les performances du modèle dans différents contextes d'utilisation. Les principes généraux ainsi que les principales caractéristiques de YOLOv8 sont présentés dans les sections suivantes.

2.3.1 De la classification à la détection

Les architectures de classification comme ResNet produisent, pour chaque image, un vecteur de probabilités sur les classes possibles. Elles ne fournissent aucune information sur la position des objets dans l'image. La détection d'objets ajoute cette dimension spatiale : pour chaque objet détecté, le modèle prédit une boîte englobante

(bounding box) définie par ses coordonnées (x, y, largeur, hauteur), ainsi que la classe de l'objet et un score de confiance.

La famille YOLO aborde ce problème par une approche unifiée : l'image est divisée en une grille de cellules, et chaque cellule est responsable de prédire les objets dont le centre tombe dans sa zone. Cette formulation permet de traiter la détection comme un problème de régression directe, résolu en une seule passe du réseau, d'où le nom You Only Look Once [15].

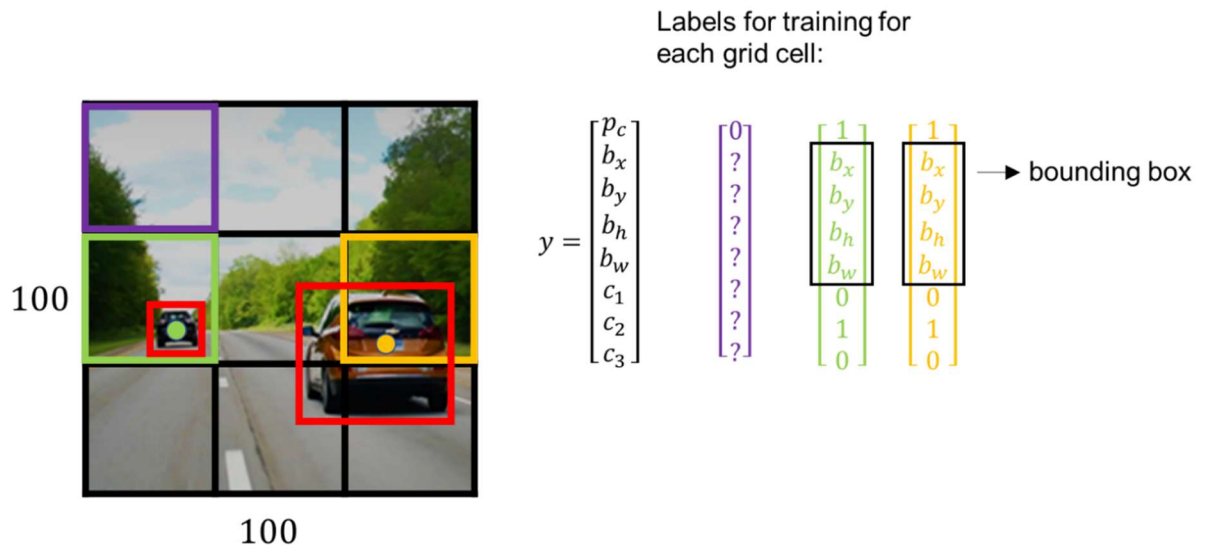


Figure 2.9 – Principe de grille et de bounding box dans YOLO.

2.3.2 Spécificités de YOLOv8

YOLOv8, publié par Jocher et al, en 2023, apporte plusieurs améliorations importantes par rapport à ses prédécesseurs [18]. Il adopte une architecture anchor-free, ce qui signifie qu'il prédit directement les coordonnées des boîtes sans avoir recours à des ancres prédéfinies. Sa tête de détection est découplée (decoupled head) : la prédiction de la classe et la régression de la boîte englobante sont réalisées par deux branches séparées, ce qui améliore la précision globale.

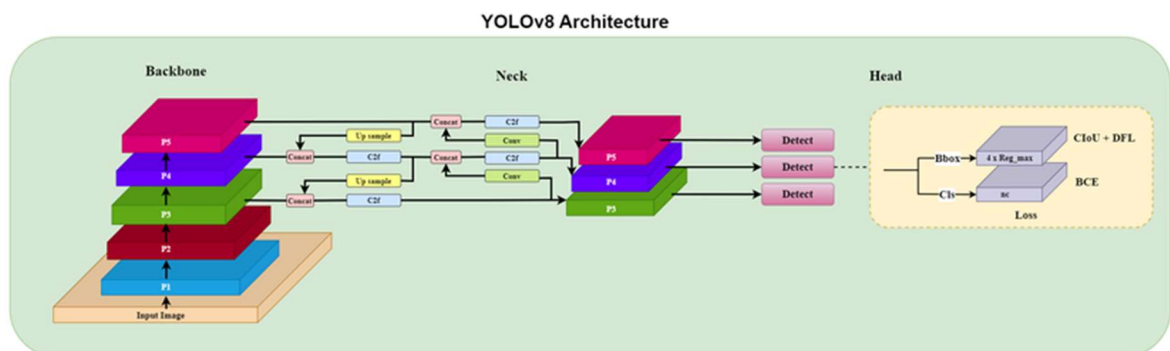


Figure 2.10 – Architecture générale de YOLOv8.

2.3.3 La version YOLOv8n

YOLOv8 est disponible en plusieurs variantes de taille, nano (n), small (s), medium (m), large (l), extra-large (x). La variante nano est la plus légère de la famille : elle compte environ 3,2 millions de paramètres, contre plusieurs dizaines de millions pour les variantes plus grandes. Cette réduction est obtenue en diminuant la largeur et la profondeur du réseau, tout en conservant la même architecture générale.

Ce compromis est particulièrement adapté aux systèmes embarqués comme le NVIDIA Jetson, où la mémoire disponible et la puissance de calcul sont limitées. Sur ce type de plateforme, les variantes plus lourdes de YOLO atteignent difficilement des fréquences d'inférence compatibles avec un usage temps réel, tandis que YOLOv8n permet de traiter plusieurs images par seconde avec une précision qui reste satisfaisante pour des applications industrielles.

Model	size (pixels)	mAP ^{val} 50-95	mAP ^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Tableau 2.1 – Tableau comparatif des variantes YOLOv8 [28].

2.4 Le Multi-Task Learning

Le Multi-Task Learning (MTL) désigne une famille de méthodes d'apprentissage où un même modèle est entraîné à résoudre plusieurs tâches simultanément, en partageant une partie de ses représentations internes. L'intuition derrière cette approche est que des tâches connexes partagent des structures sous-jacentes communes, et que les apprendre conjointement permet à chaque tâche de bénéficier de l'information contenue dans les autres.

Caruana a formalisé ce concept dès 1997 et montré empiriquement que le MTL améliorerait les performances de généralisation sur un ensemble de problèmes de reconnaissance [19]. Dans le domaine de la vision par ordinateur, le MTL se manifeste

naturellement dans les architectures de détection d'objets, où localisation et classification sont deux tâches distinctes mais intimement liées, résolues conjointement par le même réseau.

2.5 Techniques d'optimisation utilisées

L'entraînement des modèles développés dans ce travail repose sur un ensemble de techniques d'optimisation aujourd'hui standard dans la pratique du deep learning. L'optimiseur Adam [29] a été retenu pour sa capacité à adapter automatiquement le taux d'apprentissage pour chaque paramètre, ce qui accélère la convergence par rapport à une descente de gradient classique. Le Dropout [30] a été intégré dans la tête de classification de ResNet-50 comme mécanisme de régularisation, afin de limiter le surapprentissage sur un dataset de taille modeste.

Par ailleurs, un scheduler ReduceLROnPlateau a été utilisé pour réduire automatiquement le taux d'apprentissage lorsque la perte de validation stagne, et un mécanisme d'Early Stopping a permis d'interrompre l'entraînement dès que les performances cessaient de progresser. Enfin, le Mixed Precision Training [31] a été activé pour réduire le temps de calcul sur GPU sans affecter la qualité des résultats. Ces choix constituent aujourd'hui une base commune que l'on retrouve dans la grande majorité des pipelines d'entraînement modernes.

Early Stopping

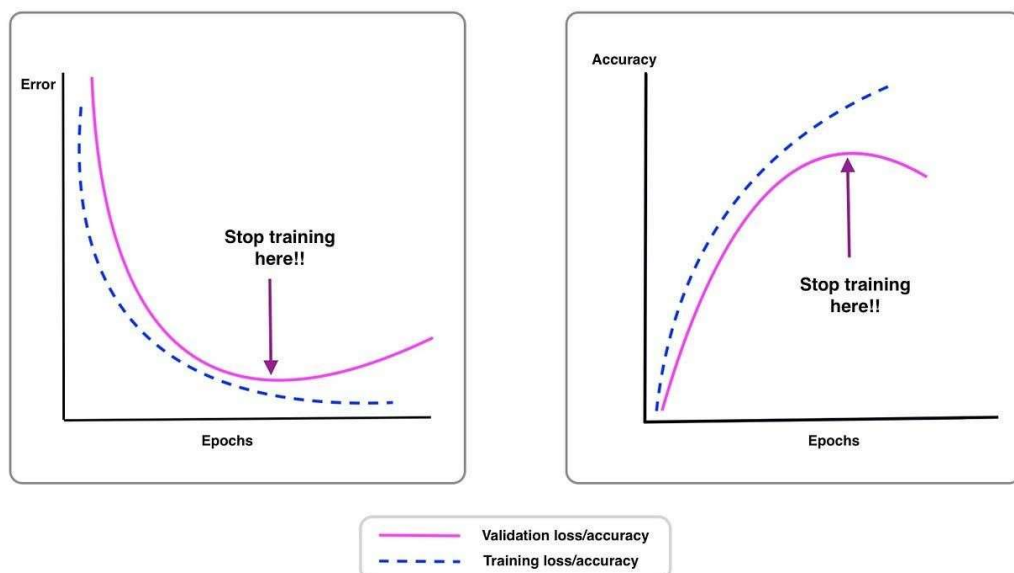


Figure 2.11 – Courbe illustrative de l'effet de l'Early Stopping [32].

Conclusion

Ce chapitre a posé les bases théoriques sur lesquelles repose l'ensemble du travail expérimental. Les CNN constituent le socle commun des deux familles de modèles utilisées, ResNet-50 pour la classification et YOLOv8 pour la détection. Le Transfer Learning justifie la faisabilité de l'approche avec un dataset de taille modeste. Les techniques d'optimisation décrites ne sont pas des choix arbitraires, mais des réponses à des problèmes concrets rencontrés lors de l'entraînement, comme on le verra dans les chapitres suivants.

Chapitre 3 : Méthodologie et Préparation du Dataset



Introduction

Tout système d'apprentissage automatique repose, en dernière instance, sur la qualité des données qui lui sont fournies et sur la cohérence de la démarche méthodologique adoptée. Dans le cadre de ce travail, aucun dataset public de tuiles en terre cuite n'était disponible avec les caractéristiques recherchées, notamment la distinction entre tuile égorgée et tuile cassée, qui constitue précisément le cœur de la problématique. Il a donc fallu constituer ce dataset from scratch, depuis la prise de vue jusqu'à l'annotation, en passant par l'organisation et la préparation des données selon les besoins de chaque approche. Ce chapitre présente d'abord la méthodologie globale du travail, puis décrit en détail la constitution et la préparation du dataset.

3.1 Méthodologie de travail

Ce travail repose sur une démarche itérative organisée autour de deux contributions principales. La première contribution consiste en un système de classification d'images par deep learning, développé en deux versions successives à partir de l'architecture ResNet-50 et du principe du Transfer Learning. La seconde contribution consiste en un système de détection d'objets en temps réel, basé sur YOLOv8n et déployé sur une plateforme embarquée NVIDIA Jetson Nano. Les deux contributions partagent le même dataset de base, collecté et préparé spécifiquement pour ce travail.

La démarche suit une progression en trois approches. La première approche aborde le problème comme une classification à trois classes avec ResNet-50, et met en évidence une limite visuelle fondamentale entre les classes. La seconde approche corrige cette limite en séparant les classes et en enrichissant le pipeline d'entraînement. La troisième approche franchit une limite plus profonde en passant à la détection d'objets avec YOLOv8n, permettant la localisation et la classification simultanées dans des scènes complexes, jusqu'au déploiement sur système embarqué. Chaque étape est motivée par les observations concrètes de l'étape précédente, c'est cette progression qui donne sa cohérence à l'ensemble du travail.

La figure ci-dessous présente le schéma général de la méthodologie adoptée, qui couvre l'ensemble des étapes depuis la collecte des données jusqu'au déploiement du système final.

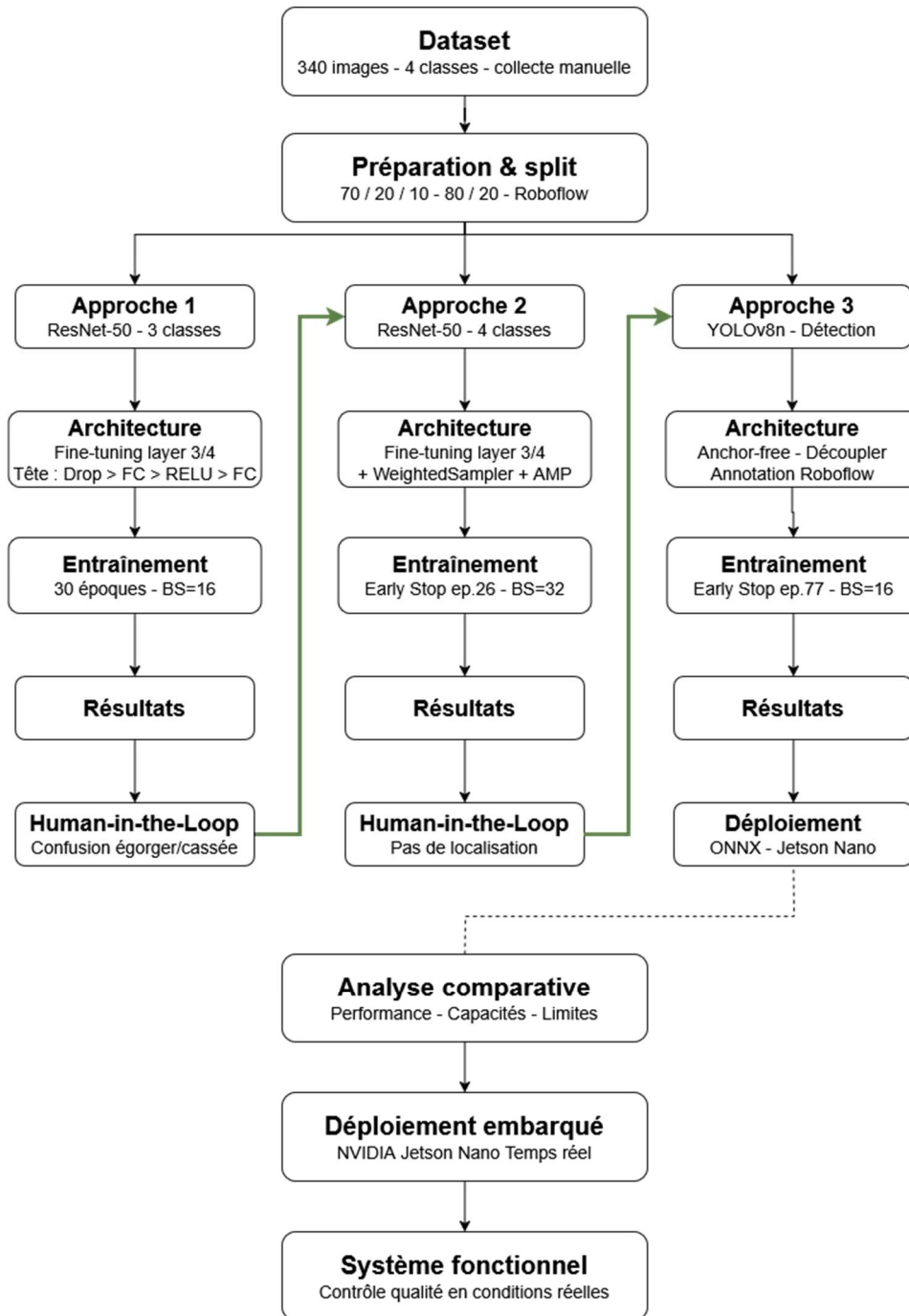


Figure 3.1 – Schéma général de la méthodologie du travail.

3.2 Collecte manuelle des données

La collecte des données constitue la première étape concrète de ce travail, et sans doute la plus déterminante. Avant de décrire la composition finale du dataset, il convient de préciser les conditions dans lesquelles les photos ont été prises, car ces conditions influencent directement la nature des données et le comportement des modèles qui en sont issus.

3.2.1 Conditions de prise de photo

La collecte des images a été réalisée entièrement à la main, au sein d'un atelier de fabrication de tuiles. Nous avons utilisé un smartphone pour prendre les photos, dans des conditions proches de celles que l'on rencontre réellement dans ce type d'environnement : éclairage à la lumière artificielle d'atelier, fond naturel qui varie d'une photo à l'autre, et angles de prise de vue intentionnellement variés. Cette variabilité n'est pas un défaut de protocole, elle est au contraire recherchée, car elle force les modèles à apprendre des représentations robustes plutôt qu'à exploiter des artefacts de contexte.

Toutes les classes ont été photographiées dans ces mêmes conditions, ce qui assure une certaine cohérence entre elles et évite que le modèle ne distingue les classes sur la base de différences d'environnement plutôt que de caractéristiques visuelles réelles des tuiles.

3.2.2 Composition du dataset

Au total, 340 photographies ont été collectées, réparties en quatre classes. La classe `tuile_ok_simple` regroupe 80 images de tuiles entières sans défaut visible. La classe `tuile_ok_egorger` contient 82 images de tuiles ayant subi une découpe fonctionnelle, c'est-à-dire taillées exprès pour s'adapter à une configuration de pose particulière. La classe `tuile_cassee` rassemble 102 images présentant des fractures visibles, résultant d'un choc ou d'un échec de cuisant. Enfin, la classe `pas_de_tuile` comprend 76 images de scènes sans aucune tuile présente, utilisée dans les deux premières approches pour apprendre au modèle à reconnaître l'absence d'objet d'intérêt.

La distribution entre classes est relativement équilibrée, avec un écart maximum d'une vingtaine d'images entre la classe la plus représentée (`tuile_cassee`, 102 images) et la moins représentée (`pas_de_tuile`, 76 images). Ce déséquilibre modéré a néanmoins été pris en compte lors de l'entraînement.



Figure 3.2 – Exemple d'images pour chacune des quatre classes.

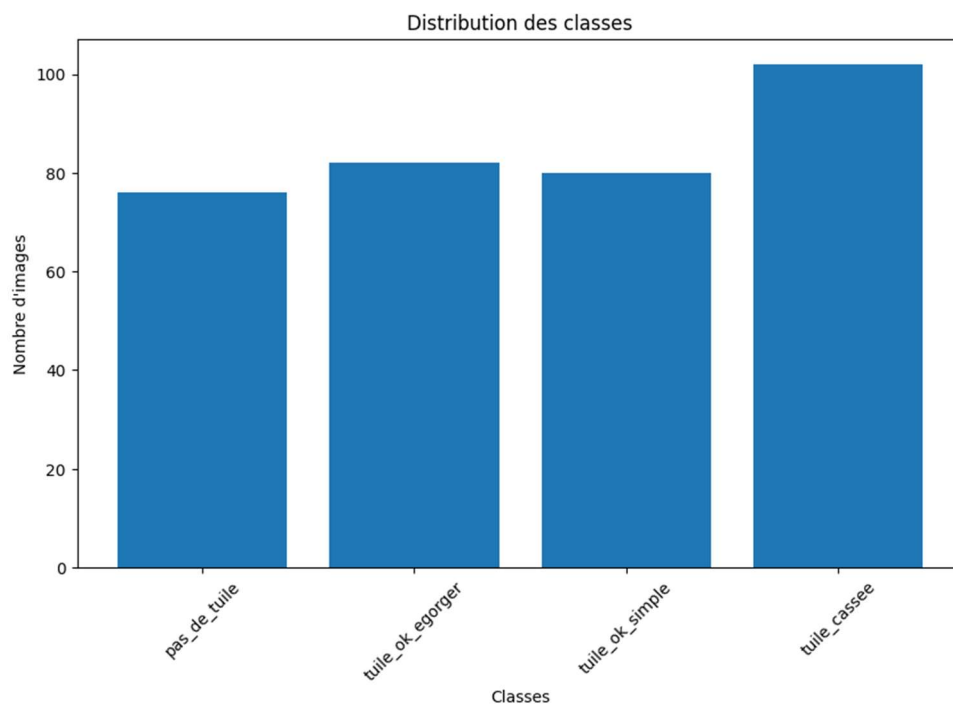


Figure 3.3 – Distribution des classes du dataset.

3.3 Analyse et distribution des classes

Un aspect important à souligner concerne la ressemblance visuelle entre certaines classes. La frontière entre une tuile égorgé (*tuile_ok_egorger*) et une tuile cassée (*tuile_cassee*) n'est pas toujours évidente à l'œil nu, surtout lorsque l'angle de prise de vue ou l'éclairage ne sont pas optimaux. Dans les deux cas, le bord de la tuile présente une rupture de matière, la différence réside dans le caractère net et rectiligne de la découpe intentionnelle et aussi par opposition au bord irrégulier et accidentel d'une cassure. C'est précisément cette ambiguïté visuelle qui a posé les premières difficultés lors de la classification.

Par ailleurs, les images présentent des dimensions hétérogènes, conséquence directe de l'utilisation d'un smartphone sans cadrage fixe. Ce point a nécessité un redimensionnement systématique des images avant tout entraînement, dont les modalités sont décrites dans la section suivante.



Figure 3.4 – Comparaison visuelle entre tuile égorgée et tuile simple.

3.4 Préparation selon les approches

La préparation des données diffère sensiblement d'une approche à l'autre, en fonction des exigences propres à chaque architecture. Nous décrivons ici les principales décisions prises, répartition des données, redimensionnement, normalisation et augmentation, en précisant à chaque fois les raisons qui ont motivé ces choix.

3.4.1 Split train / validation / test

Pour les trois approches, nous avons appliqué une répartition des données entre ensemble d'entraînement, de validation et de test. Pour l'approche 1, nous avons utilisé un ratio 70/20/10. Pour l'approche 2, la répartition adoptée dans le code était de 80/20 entre entraînement et validation, afin de maximiser le volume d'entraînement face à un problème de classification à quatre classes. Pour l'approche 3, Roboflow a géré le split directement lors de l'export du dataset annoté et augmenté (88 / 8 / 4).

Le split a été réalisé de façon aléatoire, avec une stratification par classe pour garantir que chaque sous-ensemble respecte la distribution globale du dataset.

3.4.2 Redimensionnement et normalisation

Les images collectées présentant des dimensions variables, un redimensionnement uniforme a été appliqué en amont de chaque pipeline d'entraînement. Pour les approches 1 et 2, basés sur ResNet-50, les images ont été redimensionnées à 224×224 pixels, dimension d'entrée standard pour cette architecture [26], héritée de son entraînement initial sur ImageNet. Pour l'approche 3, le redimensionnement a été réalisé directement dans Roboflow à une résolution de 640×640 pixels, conformément aux recommandations d'entraînement de YOLOv8.

La normalisation appliquée dans les approches 1 et 2 utilise les valeurs moyennes et écarts-types d'ImageNet, respectivement $[0.485, 0.456, 0.406]$ et $[0.229, 0.224, 0.225]$ pour les trois canaux RGB. Ce choix est cohérent avec l'utilisation du Transfer Learning depuis un réseau pré-entraîné sur ce dataset.

3.4.3 Augmentation des données

Face à un dataset de 340 images, l'augmentation des données joue un rôle crucial pour améliorer la capacité de généralisation des modèles et réduire le risque de surapprentissage. Pour les approches 1 et 2, l'augmentation a été appliquée directement lors de l'entraînement via les transformations de torchvision : retournement, horizontal et vertical aléatoire, rotation jusqu'à $\pm 15^\circ$, variations légères de luminosité, contraste et saturation via ColorJitter. Pour l'approche 2, nous avons ajouté un RandomGrayscale, un RandomPerspective et un GaussianBlur.

Pour l'approche 3, cette augmentation de base a été complétée par un second niveau appliqué directement dans Roboflow : recadrage aléatoire avec un zoom entre 0% et 20%, rotations entre -15° et $+15^\circ$, variations de luminosité de $\pm 25\%$, flou gaussien jusqu'à 2.5 pixels, et bruit aléatoire affectant jusqu'à 0.1% des pixels. Cette combinaison vise à rendre le modèle de détection plus robuste aux variations d'éclairage et aux légères dégradations d'image.

Le tableau ci-dessous synthétise l'ensemble des paramètres de préparation appliqués à chaque approche, permettant de visualiser en un seul coup d'œil les différences de configuration entre les trois approches expérimentées, depuis le choix de la résolution d'entrée jusqu'au format d'export du dataset.

Paramètre	Approche 1	Approche 2	Approche 3
Modèle	ResNet-50	ResNet-50	YOLOv8n
Taille d'entrée	224 x 224 px	224 x 224 px	640 x 640 px
Split données	70 / 20 / 10	80 / 20	88 / 8 / 4
Normalisation	ImageNet $\mu = [0.485, 0.456, 0.406]$ $\sigma = [0.229, 0.224, 0.225]$	ImageNet $\mu = [0.485, 0.456, 0.406]$ $\sigma = [0.229, 0.224, 0.225]$	Intégrée YOLOv8
Augmentation code	Flip H/V, Rotation $\pm 15^\circ$, ColorJitter	Flip H/V, Rotation $\pm 30^\circ$, ColorJitter, RandomGrayscale, RandomPerspective, GaussianBlur	Mosaic 1.0, Flip H/V, Scale 0.4, HSV-V 0.4
Augmentation Roboflow	Non	Non	Flip H, Crop zoom 0–20%, Rotation $\pm 15^\circ$, Brightness $\pm 25\%$, Blur 2.5px, Noise 0.1%
Annotation bounding box	Non	Non	Oui (manuelle via Roboflow)
Outil de préparation	torchvision	torchvision	Roboflow + Ultralytics
Format export	ImageFolder PyTorch	ImageFolder PyTorch	YOLOv8 (images + labels .txt)

Tableau 3.1 – Tableau récapitulatif des paramètres de préparation par approches.

3.5 L'annotation avec Roboflow (Approche 3)

Le passage à la détection d'objets dans la troisième approche a nécessité une étape supplémentaire absente des deux premiers : l'annotation manuelle des boîtes englobantes. Contrairement à la classification, où il suffit d'associer une étiquette à chaque image, la détection requiert de délimiter précisément la position de chaque objet dans l'image par un rectangle (bounding box).

Cette annotation a été réalisée entièrement à la main via la plateforme Roboflow, pour l'ensemble des images des trois classes concernées `tuile_ok_simple`, `tuile_ok_egorger` et `tuile_cassee`. La classe `pas_de_tuile` ayant été supprimée pour cette approche, puisque YOLOv8 gère naturellement l'absence d'objet par l'absence de détection. Chaque image a donc été ouverte dans l'interface d'annotation, et une boîte

englobante a été tracée manuellement autour de chaque tuile visible. Le dataset annoté a ensuite été exporté directement au format YOLOv8.

Cette étape, bien que fastidieuse, est déterminante pour la qualité du modèle de détection : une boîte mal ajustée ou une tuile non annotée constitue une erreur de supervision directe qui se répercute sur les performances finales.

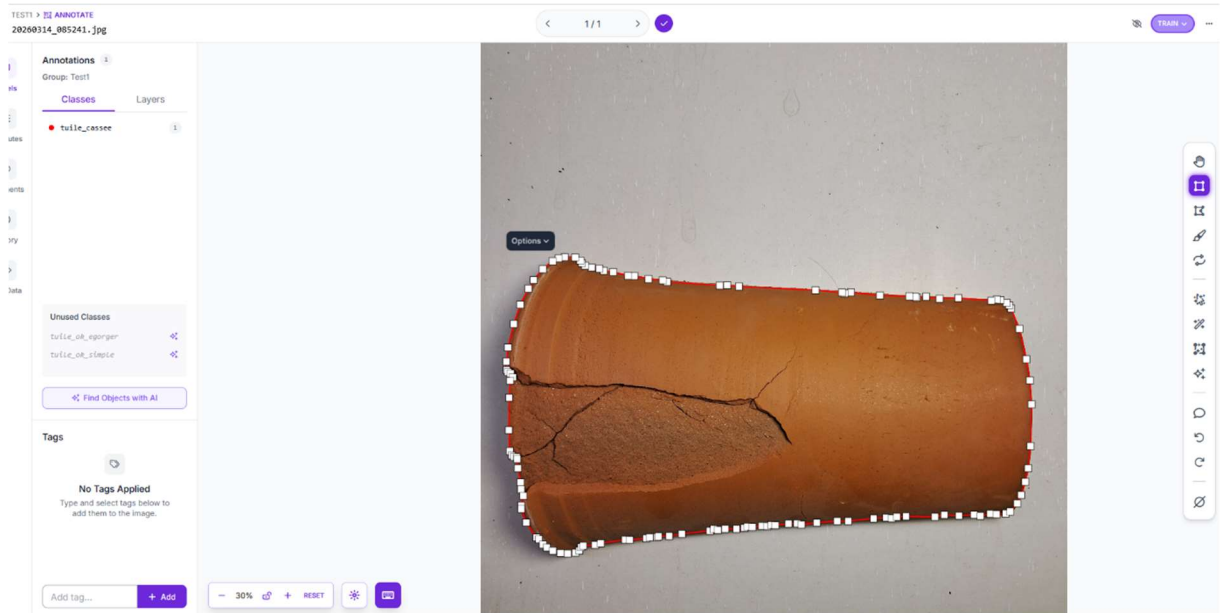


Figure 3.5 – Interface Roboflow lors de l’annotation des bounding boxes.

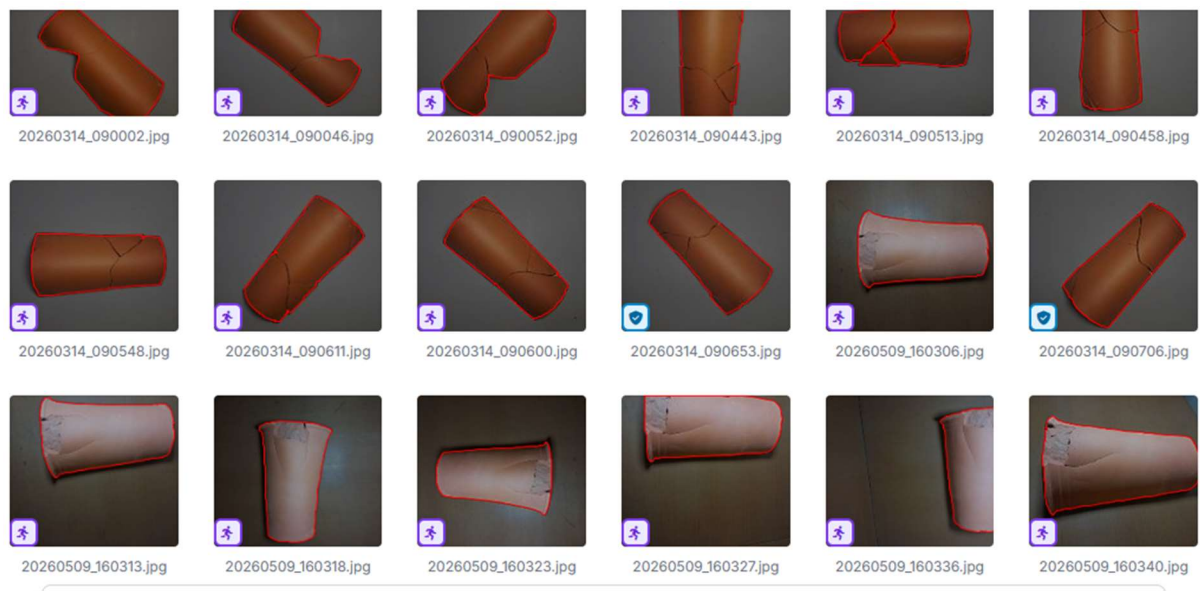


Figure 3.6 – Exemples d’ensemble d’images de la dataset annotées.

Conclusion

La constitution de ce dataset représente un travail concret et non négligeable, souvent sous-estimé dans les travaux académiques qui privilégient la description des modèles. Les 340 images collectées manuellement, dans des conditions variables et proches du réel, constituent la matière première sur laquelle repose l'ensemble des expérimentations. Les choix de préparation, redimensionnement adapté à chaque architecture, double niveau d'augmentation pour l'approche 3, annotation manuelle complète, ont été faits de façon réfléchie et cohérente avec les contraintes de chaque approche. Le chapitre suivant décrit les trois approches expérimentales.

Chapitre 4 : Expérimentations et Résultats



Introduction

Ce chapitre constitue le cœur expérimental de ce travail. Nous y décrivons successivement les approches menées, en détaillant pour chacun les choix d'architecture, les conditions d'entraînement, et les résultats obtenus. Chaque approche a été motivée par les limites observées dans le précédent, c'est cette progression itérative qui donne sa cohérence à l'ensemble de la démarche. Une analyse comparative des trois approches clôture le chapitre, suivie du déploiement sur NVIDIA Jetson Nano.

4.1 Approche 1 (Classification ResNet-50, 3 classes)

Cette première approche pose les fondations de l'ensemble de la démarche. Nous décrivons ici successivement l'architecture retenue et les choix de conception, les conditions d'entraînement, puis les résultats obtenus et les limites observées qui ont motivé la suite du travail.

4.1.1 Formulation du problème et architecture

Dans cette première approche, nous avons abordé le problème comme une classification d'image à trois classes : `pas_de_tuile`, `tuile_cassee` et `tuile_ok`, cette dernière regroupant les tuiles intactes et les tuiles égorgées en une seule catégorie. Ce choix initial semblait raisonnable, dans la mesure où les deux types de tuiles fonctionnelles partagent le même statut du point de vue du contrôle qualité.

Le modèle retenu est ResNet-50, pré-entraîné sur ImageNet et adapté par Transfer Learning à notre tâche. Nous avons remplacé la tête de classification originale par une tête personnalisée, construite dans une fonction `build_model()` et composée comme suit : une couche Dropout(0.4), suivie d'une couche linéaire `Linear(2048, 256)`, d'une activation ReLU, d'un second Dropout(0.3), et enfin d'une couche de sortie `Linear(256, 3)`. Ce choix architectural visait à réduire progressivement la dimensionnalité des représentations tout en limitant le surapprentissage.

Concernant la stratégie de gel, nous avons maintenu les paramètres de `layer1` et `layer2` fixes, et laissé libres `layer3`, `layer4` ainsi que la nouvelle tête de classification. Les couches basses, qui ont appris des représentations génériques comme les contours et les textures de base, sont conservées telles quelles, tandis que les couches plus profondes sont adaptées aux spécificités visuelles des tuiles.

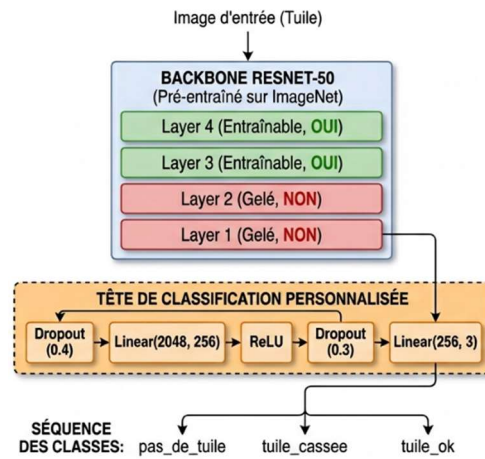


Figure 4.1 – Architecture ResNet et stratégie de gel.

4.1.2 Conditions d'entraînement

L'entraînement a été conduit sur la plateforme Kaggle, en exploitant un GPU Tesla T4. Nous avons utilisé l'optimiseur Adam avec un taux d'apprentissage initial de $1e-4$, et un scheduler de type StepLR qui réduit ce taux d'un facteur 0.5 tous les 10 époques. La taille de batch retenue est de 16, et le modèle a été entraîné sur 30 époques au total. La fonction de perte utilisée est la CrossEntropyLoss standard, sans pondération particulière des classes.

4.1.3 Résultats et analyse

La convergence du modèle a été rapide et stable. Dès la deuxième époque, la précision de validation atteignait 98%, et à partir de l'époque 7, elle se maintenait à 100% de façon quasi continue jusqu'à la fin de l'entraînement. La loss d'entraînement, qui débutait à 0.93 lors de la première époque, a chuté rapidement pour se stabiliser en dessous de 0.01 à partir de l'époque 14.



Figure 4.2 – Résultats positif pour l'approche 1.

Sur le jeu de test final, les résultats sont les suivants : accuracy globale de 100%, avec une précision, un rappel et un F1-score de 1.00 pour chacune des trois classes. La matrice de confusion ne présente aucune erreur de classification sur les 50 exemples du jeu de test. Le tableau ci-dessous détaille ces métriques par classe.

Classe	Précision	Rappel	F1-score	Support
pas_de_tuile	1.00	1.00	1.00	14
tuile_cassee	1.00	1.00	1.00	20
tuile_ok	1.00	1.00	1.00	16
Moyenne	1.00	1.00	1.00	50

Tableau 4.1 – Métriques de classification par classe sur le jeu de test, Approche 1.

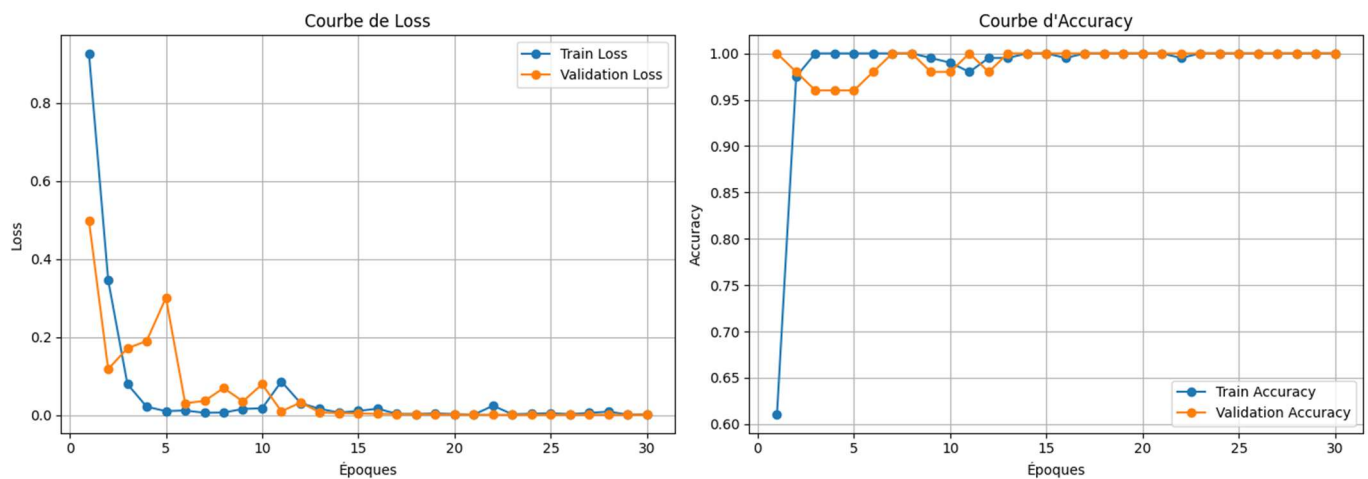


Figure 4.3 – Courbes de loss et d'accuracy sur 30 époques, Approche 1.

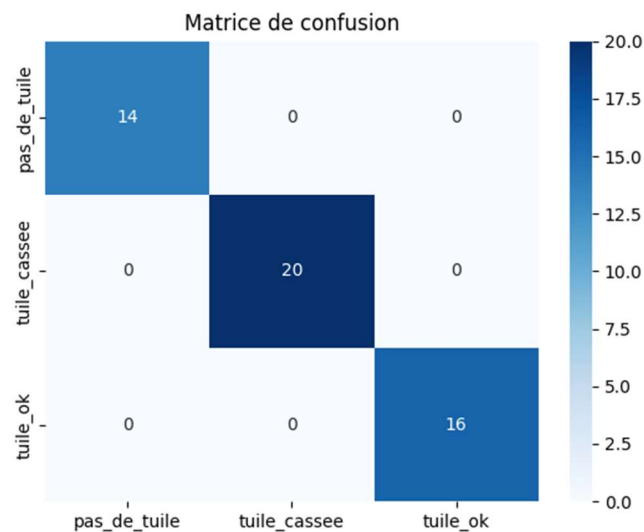


Figure 4.4 – Matrice de confusion sur le jeu de test, Approche 1.

Ces résultats, bien que remarquables en apparence, doivent être interprétés avec prudence. La véritable limite de ce modèle est apparue lors de tests sur des photographies inédites : plusieurs tuiles égorgées ont été systématiquement classées comme tuile_cassee.

Ce comportement s'explique par la ressemblance visuelle entre les deux types de bords, que le modèle n'avait pas appris à distinguer, puisque les deux étaient fondus dans une seule classe tuile_ok. Ce constat a constitué le point de départ de la deuxième approche.



Figure 4.5 – Résultats sur des images inédits, et une photo mal classée.

4.2 Approche 2 (Classification ResNet-50, 4 classes)

Cette seconde approche est directement né des observations du précédent. Plutôt que de repartir d'une architecture différente, nous avons conservé la même base, ResNet-50 avec Transfer Learning, en apportant des modifications ciblées au niveau de la formulation du problème et du pipeline d'entraînement. Les sections suivantes détaillent ces ajustements et leurs effets sur les résultats.

4.2.1 Motivation et reformulation du problème

La première approche avait montré que le modèle, bien qu'efficace sur le jeu de validation, échouait à distinguer les tuiles égorgées des tuiles cassées lorsqu'il était confronté à des images inédites. En fusionnant tuile_ok_simple et tuile_ok_egorger dans une seule classe tuile_ok, nous avons privé le modèle de toute supervision permettant d'apprendre cette distinction. La solution naturelle consistait donc à séparer ces deux catégories en classes distinctes, passant ainsi d'une classification à trois classes à une classification à quatre classes.

4.2.2 Architecture et stratégie de fine-tuning

La tête de classification personnalisée est identique à celle de l'approche précédente : $\text{Dropout}(0.4) \rightarrow \text{Linear}(2048, 256) \rightarrow \text{ReLU} \rightarrow \text{Dropout}(0.3) \rightarrow \text{Linear}(256, 4)$ seule la dimension de sortie passant de 3 à 4. La stratégie de gel des couches reste également inchangée, layer1 et layer2 sont maintenues gelées, tandis que layer3, layer4 et la tête de classification sont libres.

4.2.3 Améliorations du pipeline d'entraînement

Plusieurs modifications ont été apportées par rapport à la première approche. La première concerne l'augmentation des données, sensiblement enrichie : aux transformations de base déjà utilisées, nous avons ajouté un RandomGrayscale avec une probabilité de 10%, un RandomPerspective appliqué dans 30% des cas, et un GaussianBlur avec un sigma variable entre 0.1 et 2.0.

La deuxième modification concerne le rééquilibrage des classes. Un WeightedRandomSampler a été mis en place pour que chaque batch d'entraînement contienne une représentation équilibrée des quatre classes. Ce mécanisme est complété par une CrossEntropyLoss pondérée, où le poids attribué à chaque classe est inversement proportionnel à son nombre d'exemples.

La troisième modification porte sur le scheduler. Contrairement au StepLR à paliers fixes utilisé dans l'approche 1, nous avons opté pour un ReduceLROnPlateau qui surveille la loss de validation et réduit le taux d'apprentissage d'un facteur 0.5 dès que celle-ci stagne pendant 4 époques consécutives. Un mécanisme d'Early Stopping avec une patience de 10 époques a également été ajouté. La batch size a été augmentée à 32, et la répartition des données adoptée était de 80/20 entre entraînement et validation.

4.2.4 Résultats et analyse

La convergence du modèle a été remarquablement rapide. Dès la troisième époque, la précision de validation atteignait 100%, et la val_loss continuait de décroître régulièrement jusqu'à l'époque 16, où elle atteignit sa valeur minimale de 0.0139, c'est à cette époque que le meilleur modèle a été sauvegardé. L'Early Stopping s'est déclenché à l'époque 26, après 10 époques sans amélioration. On observe également les réductions successives du taux d'apprentissage par le scheduler : de $1e-4$ à $5e-05$ à l'époque 10, puis à $2.5e-05$ à l'époque 21, et enfin à $1.25e-05$ à l'époque 26.



Figure 4.6 – Résultats positif pour l'approche 2.

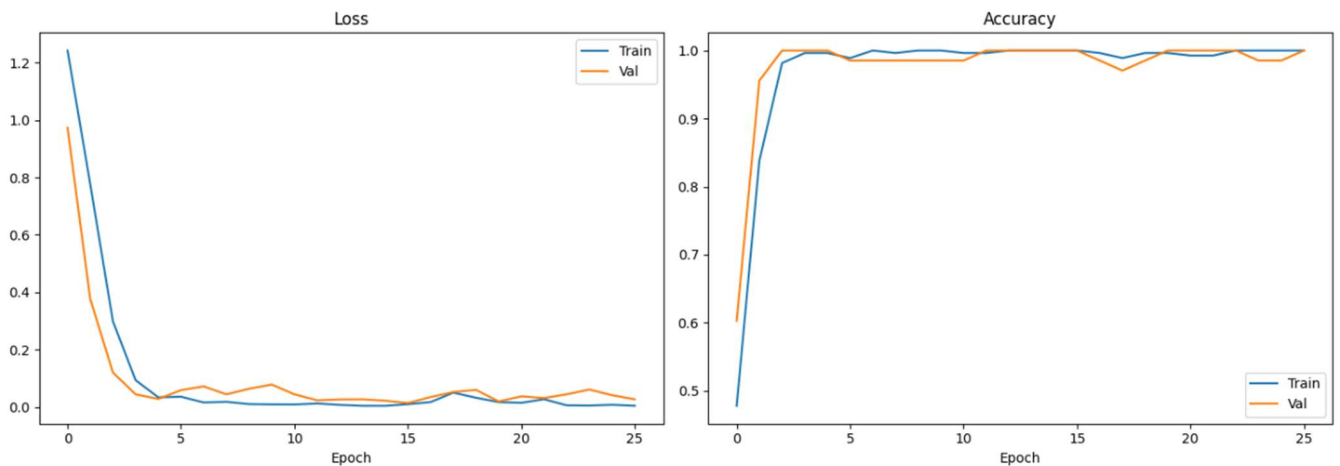


Figure 4.7 – Courbes de loss et d'accuracy, Approche 2.

Sur le jeu de validation, les résultats sont parfaits pour les quatre classes : précision, rappel et F1-score de 1.00 sur l'ensemble des 68 images. La matrice de confusion ne présente aucune erreur, y compris entre `tuile_ok_egorger` et `tuile_cassee`, ce qui constitue l'amélioration essentielle par rapport à l'approche 1.

Classe	Précision	Rappel	F1-score	Support
pas_de_tuile	1.00	1.00	1.00	16
tuile_cassee	1.00	1.00	1.00	16
tuile_ok_egorger	1.00	1.00	1.00	16
tuile_ok_simple	1.00	1.00	1.00	20
Moyenne	1.00	1.00	1.00	68

Tableau 4.2 – Métriques de classification par classe sur le jeu de validation, Approche 2.

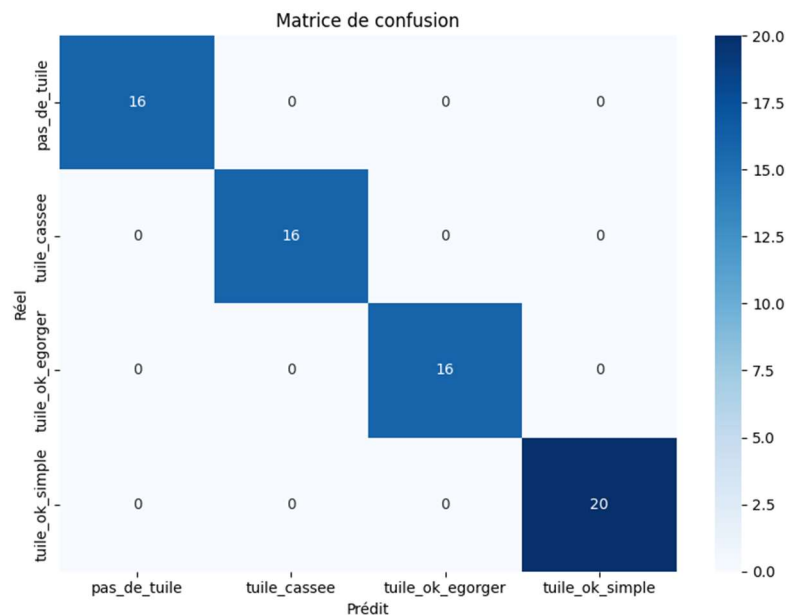


Figure 4.8 – Matrice de confusion sur le jeu de test, Approche 2.

Toutefois, les tests sur images inédites ont révélé une nouvelle limite : lorsqu’une scène contient des tuiles mélangées à d’autres objets, le modèle retourne systématiquement la prédiction `pas_de_tuile`. Ce comportement s’explique par la nature même de la classification globale d’image, qui produit une unique étiquette sans localiser les objets. C’est cette limite fondamentale qui a motivé le passage à une approche de détection d’objets.



Figure 4.9 – Exemple d’images mixte classée comme pas_de_tuile.

4.3 Approche 3 (Classification et Détection YOLOv8n)

La troisième approche marque un changement de paradigme par rapport aux deux précédents. Nous passons ici de la classification globale d’image à la détection d’objets, ce qui implique non seulement un nouveau modèle, mais aussi une reformulation complète du problème, une nouvelle préparation du dataset, et une évaluation basée sur des métriques différentes.

4.3.1 Motivation et changement de paradigme

Les deux approches précédentes ont démontré qu’un modèle de classification, même bien entraîné, atteint une limite structurelle dès lors que la scène à analyser contient plusieurs objets distincts. Dans un contexte industriel réel, où les tuiles peuvent apparaître mélangées à d’autres éléments dans le champ de la caméra, cette limite est rédhibitoire. Le passage à la détection d’objets s’est donc imposé comme une nécessité. Nous avons retenu YOLOv8n pour deux raisons complémentaires : sa capacité à détecter et classifier simultanément plusieurs objets dans une même image, et sa légèreté, indispensable pour un déploiement sur système embarqué.

4.3.2 Préparation du dataset et configuration de l’entraînement

La classe pas_de_tuile a été supprimée du dataset pour cette approche. En détection d’objets, l’absence d’objet se traduit naturellement par l’absence de détection, il n’est pas nécessaire de l’apprendre explicitement. Le dataset se compose donc de trois classes :

tuile_ok_simple, tuile_ok_egorger et tuile_cassee, avec leurs bounding boxes annotées manuellement via Roboflow et exportées au format YOLOv8.

L'entraînement a été conduit sur machine locale, équipée d'un processeur AMD Ryzen 5600G et de 16 Go de RAM, sans GPU dédié. Le script d'entraînement utilise l'interface Python d'Ultralytics, avec une résolution d'entrée de 640×640 pixels et un maximum de 100 époques. L'augmentation intégrée à YOLOv8 a été configurée avec une mosaïque à 1.0, des retournements horizontaux et verticaux, des rotations jusqu'à $\pm 15^\circ$, une variation d'échelle de 0.4, et des ajustements de luminosité HSV. Un label smoothing de 0.1 a également été appliqué pour améliorer la généralisation. L'Early Stopping a été fixée à 20 époques.

4.3.3 Résultats et analyse

L'entraînement s'est arrêté à l'époque 77 sur les 100 prévues, l'Early Stopping ayant détecté l'absence d'amélioration du mAP50 pendant 20 époques consécutives. La progression du modèle révèle plusieurs phases distinctes. Durant les dix premières époques, les métriques sont encore instables, le mAP50 oscille entre 0.53 et 0.96, ce qui est caractéristique de la phase de chauffe du scheduler. À partir de l'époque 15, la convergence s'accélère nettement : le mAP50 dépasse 0.98 et se maintient dans cette plage jusqu'à la fin de l'entraînement.

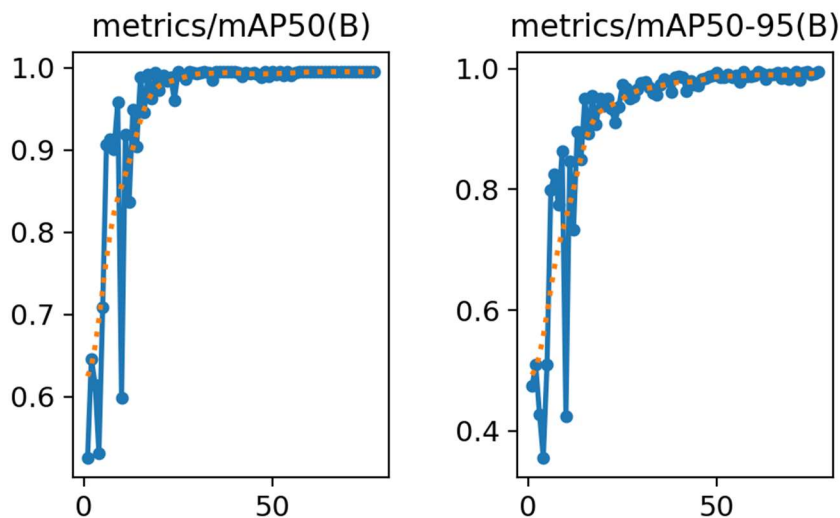


Figure 4.10 – Courbes mAP50 et mAP50-95 en fonction des époques, Approche 3.

Les performances atteintes au meilleur modèle sont les suivantes : un mAP50 de 0.995, un mAP50-95 de 0.995, une précision de 0.995 et un rappel de 1.00. Le tableau ci-dessous détaille les métriques par classe telles que rapportées par YOLOv8 sur le jeu de validation.

Classe	Précision	Rappel	F1-score	mAP50
tuile_cassee	0.995	1.00	0.997	0.995
tuile_ok_egorger	0.995	1.00	0.997	0.995
tuile_ok_simple	0.995	1.00	0.997	0.995
Moyenne	0.995	1.00	0.997	0.995

Tableau 4.3 – Métriques de détection par classe, Approche 3

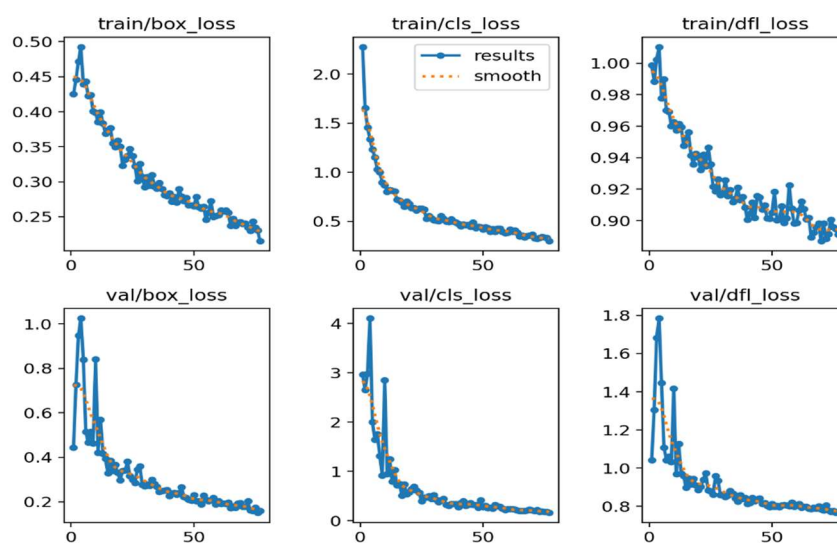


Figure 4.11 – Courbes de loss d'entraînement et de validation, Approche 3.

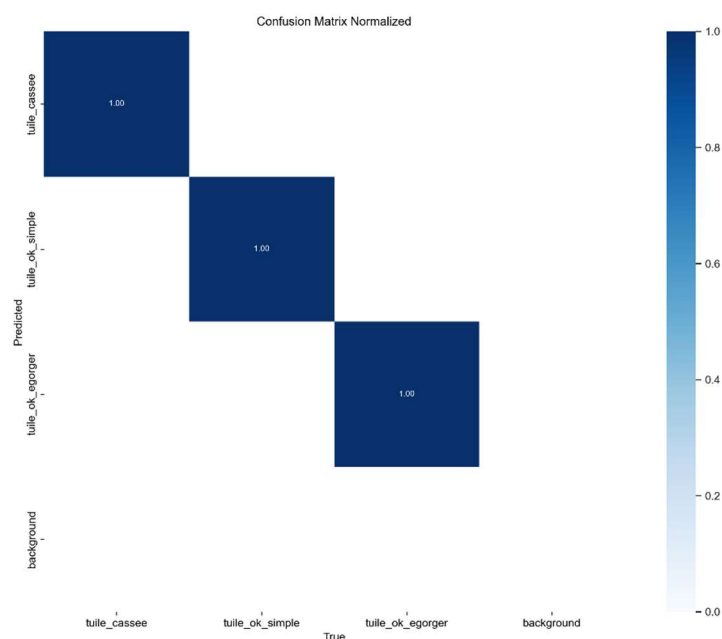


Figure 4.12 – Matrice de confusion normalisée du modèle finale YOLOv8n.

Sur des images inédites, le modèle détecte correctement les tuiles même en présence d'autres objets dans la scène, en plaçant des bounding boxes précises autour de chaque tuile identifiée. La confusion entre *tuile_ok_egorger* et *tuile_cassee*, qui constituait le problème central des deux premières approches, n'est plus observée. Ce résultat valide l'approche Multi-Task Learning adoptée.

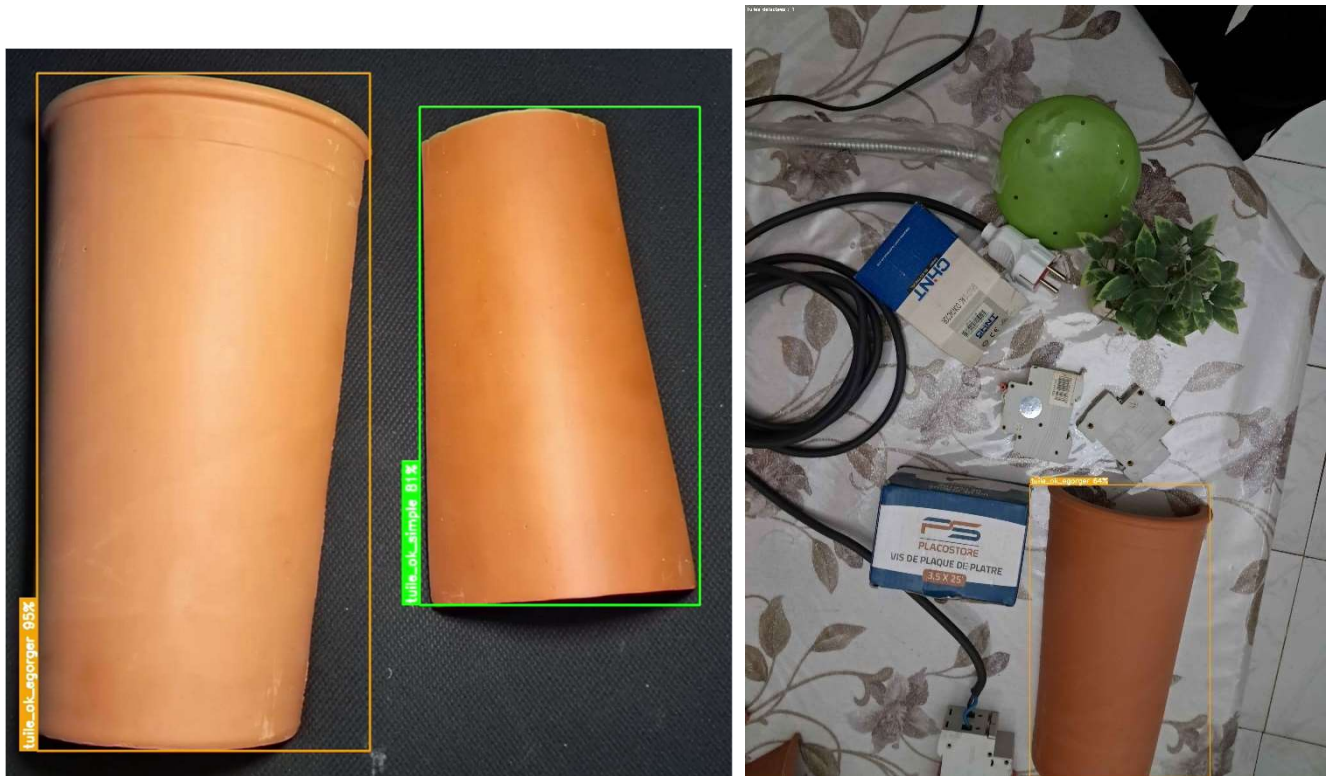


Figure 4.13 – Exemples d'images inédites avec bounding boxes et classification.

4.4 Comparaison et analyse des trois approches

Les trois approches menées dans ce travail ne constituent pas trois tentatives indépendantes, ils forment une progression cohérente où chaque étape répond aux limites de la précédente. La première approche a établi que ResNet-50 avec Transfer Learning converge rapidement et efficacement sur un dataset de petite taille. Mais cette performance masquait une faiblesse conceptuelle : la fusion de deux classes visuellement distinctes en une seule catégorie. La seconde approche a corrigé ce problème, mais une limite plus fondamentale est apparue, celle inhérente à toute architecture de classification globale face à des scènes complexes. La troisième approche a franchi ce cap en changeant de paradigme : de la classification à la détection.

Paramètre	Approche 1	Approche 2	Approche 3
Modèle	ResNet-50	ResNet-50	YOLOv8n
Tâche	Classification	Classification	Détection d'objets
Nombre de classes	3	4	3
Paramètres du modèle	~ 23 M (partiel)	~ 23 M (partiel)	~ 3.2 M
Taille d'entrée	224 x 224 px	224 x 224 px	640 x 640 px
Transfer Learning	ImageNet	ImageNet	COCO (yolov8n.pt)
Optimiseur	Adam	Adam	SGD
Scheduler	StepLR	ReduceLROnPlateau	Cosine Annealing
Learning rate initiale	1e-4	1e-4	1e-2
Batch size	16	32	16
Early Stopping	Non	Oui (patience=10)	Oui (patience=20)
Époques prévues / réelles	30 / 30	40 / 26	100 / 77
Plateforme	Kaggle Tesla T4	Kaggle Tesla T4	AMD Ryzen 5600G
Scènes complexes	Non	Non	Oui
Format exporté du model	.pth	.pth	.onnx

Tableau 4.4 – Tableau de comparaison entre les modèles des 3 approches.

4.5 Déploiement sur NVIDIA Jetson Nano

Le déploiement sur système embarqué constitue l'aboutissement concret de ce travail. Cette section décrit la plateforme utilisée, le pipeline mis en place pour passer du modèle entraîné à un système fonctionnel en temps réel, puis les résultats observés en conditions réelles.

4.5.1 Le déploiement embarqué

Les cartes de développement NVIDIA Jetson ont été conçues précisément pour répondre aux besoins de l'inférence embarquée : elles intègrent un GPU capable d'accélérer les calculs de réseaux de neurones dans un format compact et avec une consommation énergétique modérée. Mittal a réalisé une étude comparative des performances de différentes architectures légères sur Jetson Nano, montrant que des modèles comme MobileNet ou les variantes nano de YOLO atteignaient des fréquences

d'inférence compatibles avec un usage temps réel [20]. Le choix de YOLOv8n dans notre travail s'inscrit directement dans cette logique : privilégier un modèle suffisamment léger pour tourner en temps réel sur Jetson, sans sacrifier trop de précision.

4.5.2 Présentation de la plateforme

La NVIDIA Jetson Nano Developer Kit 4GB est une carte de développement embarquée conçue spécifiquement pour les applications d'intelligence artificielle à la périphérie du réseau (edge AI). Elle intègre un GPU NVIDIA Maxwell à 128 cœurs CUDA, un processeur ARM Cortex-A57 quadricœur, et 4 Go de mémoire LPDDR4 partagée entre le CPU et le GPU. Cette architecture unifiée la rend particulièrement adaptée à l'inférence de réseaux de neurones dans des conditions embarquées.

Le système a été déployé sous Ubuntu 22. Il convient de souligner une contrainte importante de cette plateforme : la mémoire de stockage interne est de type eMMC et limitée à 16 Go. Dans la pratique, le système d'exploitation seul occupe une part significative de cet espace, et l'installation des bibliothèques nécessaires, Ultralytics, OpenCV, ONNX Runtime et leurs dépendances, représente un volume non négligeable. Cette contrainte de stockage a constitué la principale difficulté rencontrée lors du déploiement, nécessitant une gestion minutieuse de l'espace disponible.



Figure 4.14 – NVIDIA Jetson Nano Developer kit 4GB avec webcam USB.

4.5.3 Résultats en conditions réelles

Le système déployé sur le Jetson Nano s'est montré fonctionnel dans des conditions proches de celles d'un atelier réel. La webcam USB capture le flux vidéo en continu, et le modèle YOLOv8n exporté en ONNX détecte et classe correctement les tuiles présentées devant la caméra, qu'il s'agisse de tuiles intactes, taillées ou cassées. La distinction entre `tuile_ok_egorger` et `tuile_cassee`, qui avait posé problème dans les deux premières approches, est correctement résolue par le modèle de détection même en conditions réelles.

Le choix de YOLOv8n, la variante la plus légère de la famille YOLOv8, se justifie pleinement à ce stade : un modèle plus lourd aurait probablement saturé les ressources mémoire de la plateforme. La vitesse d'inférence observée se situe approximativement entre 5 et 10 FPS, une cadence modeste et inhérente aux contraintes matérielles du Jetson Nano sans accélération TensorRT, mais suffisante pour un contrôle qualité visuel où chaque pièce est présentée individuellement devant la caméra. Ce déploiement valide la cohérence de l'ensemble de la démarche : un modèle entraîné avec des ressources limitées peut être rendu opérationnel sur une plateforme embarquée accessible, et produire des résultats utiles dans un contexte industriel réel.

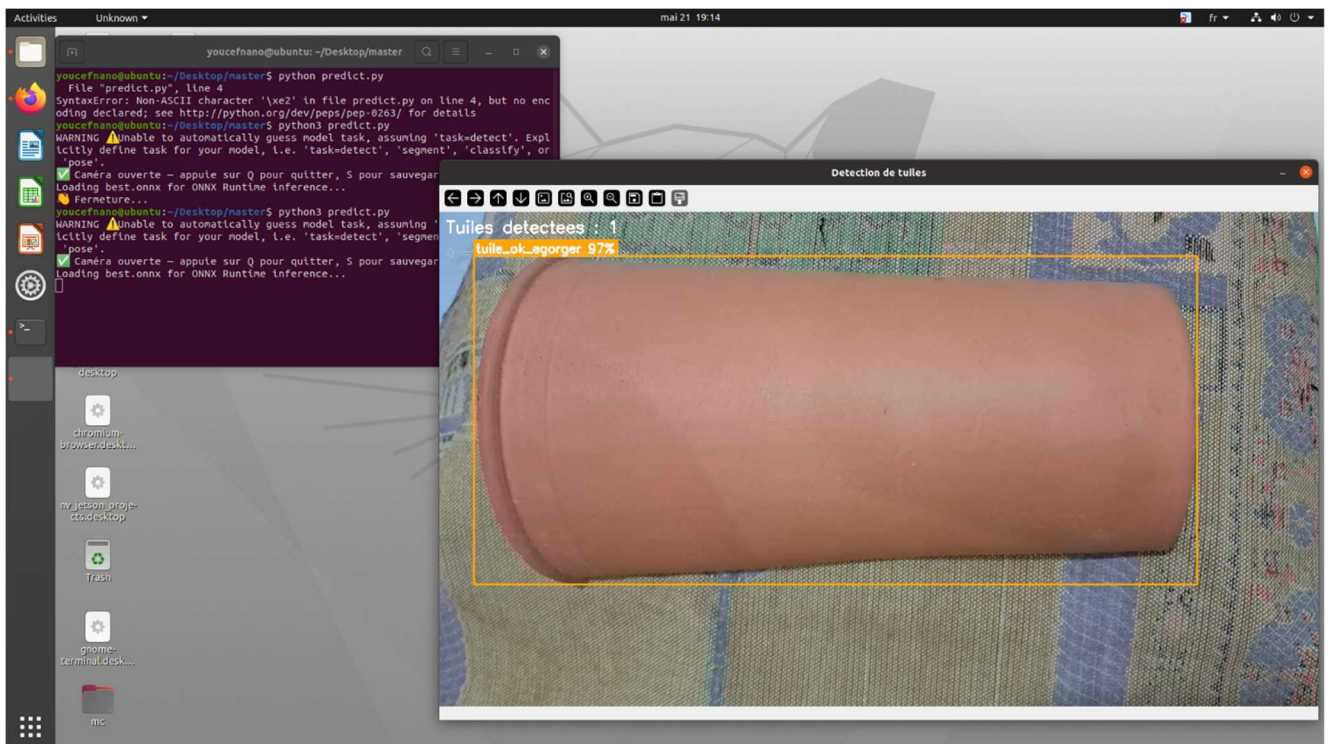


Figure 4.15 – Captures d'écran du système en fonctionnement sur la plateforme embarquée.

Conclusion

Ce chapitre a présenté l'ensemble des expérimentations menées dans ce travail, depuis la première tentative de classification par ResNet-50 jusqu'au déploiement du modèle YOLOv8n sur NVIDIA Jetson Nano. La progression en trois approches, motivée à chaque étape par des limites concrètes observées, constitue la colonne vertébrale de ce mémoire. Les résultats obtenus, 100% d'accuracy pour les deux premiers cas, 99.5% de mAP50 pour le troisième, témoignent de la pertinence des choix techniques effectués.

Conclusion

Le travail présenté dans ce mémoire avait pour objectif de concevoir et de mettre en œuvre un système automatique de contrôle qualité de tuiles en terre cuite, en s'appuyant sur les techniques modernes de vision par ordinateur et d'apprentissage profond. La démarche adoptée a permis de construire ce système en trois étapes successives, chacune répondant aux limites concrètes observées dans la précédente.

La première approche a établi la faisabilité de l'approche par classification avec ResNet-50 et Transfer Learning, en atteignant des performances parfaites en validation sur trois classes. Il a également mis en évidence, dès les premiers tests sur images inédites, la difficulté centrale de ce problème : la confusion visuelle entre une tuile égorgée et une tuile cassée. Cette observation, loin d'être un échec, a orienté de façon décisive la suite du travail. La seconde approche a répondu à cette limite en séparant explicitement les deux catégories de tuiles fonctionnelles et en enrichissant le pipeline d'entraînement, augmentation de données renforcée, rééquilibrage des classes, Early Stopping et scheduler adaptatif. Le modèle obtenu distingue correctement les quatre classes, mais s'est heurté à une limite plus fondamentale : celle de la classification globale d'image, incapable de localiser les objets dans des scènes complexes. La troisième approche a franchi ce cap en adoptant YOLOv8n, qui apprend simultanément à localiser et à classer les tuiles dans une même image. Son déploiement sur NVIDIA Jetson Nano via un export ONNX et un pipeline d'inférence temps réel avec webcam USB a produit un système embarqué fonctionnel, validant ainsi la cohérence de l'ensemble de la démarche.

Au-delà des résultats techniques, ce travail illustre une réalité souvent sous-estimée dans les projets de deep learning appliqué : la constitution du dataset est une étape à part entière, qui conditionne tout le reste. Les 340 images collectées manuellement dans un atelier réel, avec leurs imperfections et leur variabilité naturelle, ont été la matière première sans laquelle aucun des modèles développés n'aurait pu exister.

Plusieurs limites méritent néanmoins d'être mentionnées honnêtement. La taille du dataset reste modeste, 340 images constituent une base acceptable pour une étude exploratoire, mais insuffisante pour garantir une robustesse industrielle à grande échelle. Les performances en validation, bien que parfaites sur les trois approches, ont été évaluées sur des images issues du même environnement de collecte, et une évaluation plus rigoureuse nécessiterait des images provenant d'autres ateliers, avec d'autres éclairages

et d'autres types de tuiles. La vitesse d'inférence sur le Jetson Nano, estimée entre 5 et 10 FPS sans optimisation TensorRT, reste par ailleurs en dessous de ce que pourrait exiger une ligne de production à cadence élevée.

Ces limites ouvrent naturellement plusieurs pistes d'amélioration et d'extension. La première, et sans doute la plus impactante, serait d'enrichir le dataset en collectant davantage d'images dans des conditions plus variées, ou en intégrant des techniques de génération de données synthétiques par réseau génératif adversarial pour simuler des défauts rares. Sur le plan de l'architecture, explorer des approches de segmentation d'instance permettrait non seulement de détecter les tuiles mais aussi de délimiter précisément les zones de défaut à l'intérieur de chaque pièce, une information bien plus riche pour un opérateur industriel. L'optimisation du déploiement embarqué via une conversion au format TensorRT représente également une perspective concrète, permettant d'accélérer significativement l'inférence et d'atteindre des cadences compatibles avec une production industrielle réelle. À plus long terme, l'intégration de ce système dans une chaîne de production automatisée constituerait le passage naturel d'un prototype académique à un outil industriel opérationnel.

Bibliographie

- [1] C. Boukouvalas, J. Kittler, R. Marik, M. Mirmehdi et M. Petrou, "Automatic Grading of Textured Ceramic Tiles", *Proceedings of SPIE, Automated Optical Inspection for Industry*, 1999.
- [2] A. Kumar, "Computer-Vision-Based Fabric Defect Detection: A Survey," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 1, pp. 348-363, 2008. DOI: 10.1109/TIE.2007.907565
- [3] X. Xie et M. Mirmehdi, "TEXEMS: Texture Exemplars for Defect Detection on Random Textured Surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1343-1356, 2007. DOI:10.1109/TPAMI.2007.1153
- [4] M. Shahriari et al., "Fault Detection in Textured Surfaces Using HOG Descriptors and SVM," *International Journal of Advanced Manufacturing Technology*, 2014.
- [5] Sahin, "Computer vision (CV)," *IoThook*, 30 août 2025. [En ligne]. Disponible : <https://iothook.com/en/computer-vision-cv/> . [Consulté le : 17/05/2026]
- [6] A. Krizhevsky, I. Sutskever et G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.
- [7] E. Oztemel et O. Kayabasi, "A Neural Network Based Quality Control System for Ceramic Tile Production", *International Journal of Production Research*, vol. 45, no. 4, pp. 839–851, 2007. DOI: 10.1080/00207540600792373.
- [8] J. Feng, J. Ji, H. Liu et Q. Lu, "Ceramic Tile Surface Defect Detection Based on Deep Learning", *Ceramics International*, vol. 47, no. 5, pp. 10062–10076, 2021. DOI: 10.1016/j.ceramint.2020.12.189.
- [9] S. J. Pan et Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, 2010. DOI : 10.1109/TKDE.2009.191
- [10] A. Ayari, *Architecture CNN — cours apprendre le deep learning avec TensorFlow*, GitHub, 2020. [Image en ligne]. Disponible sur : <https://github.com/anisayari/Youtube-apprendre-le-deeplearning-avec-tensorflow/blob/master/%234%20%20CNN/architecture.png>
- [11] R. Girshick, J. Donahue, T. Darrell et J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *IEEE CVPR*, 2014. DOI: 10.1109/CVPR.2014.81
- [12] R. Girshick, "Fast R-CNN," *IEEE ICCV*, 2015. DOI: 10.1109/ICCV.2015.169
- [13] S. Ren, K. He, R. Girshick et J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *NeurIPS*, 2015.
- [14] W. Liu et al., "SSD: Single Shot MultiBox Detector," *ECCV*, 2016. DOI: 10.1007/978-3-319-46448-0_2

- [15] J. Redmon, S. Divvala, R. Girshick et A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," IEEE CVPR, 2016. DOI : 10.1109/CVPR.2016.91
- [16] A. Vina, " Un guide pour plonger au cœur de la détection d'objets en 2025 ", Ultralytics Blog, 2025. [En ligne]. Disponible sur : <https://www.ultralytics.com/fr/blog/a-guide-to-deep-dive-into-object-detection-in-2025>
- [17] Y. Xiao, Z. Tian, J. Yu, Y. Zhang, S. Liu, S. Du et X. Lan, " A review of object detection based on deep learning ", Multimedia Tools and Applications, vol. 79, n° 33-34, pp. 23729–23791, 2020. [Figure en ligne]. Disponible sur : <https://www.researchgate.net/publication/342140262>
- [18] G. Jocher, A. Chaurasia et J. Qiu, "Ultralytics YOLOv8," GitHub, 2023. [En ligne]. Disponible: <https://github.com/ultralytics/ultralytics>
- [19] R. Caruana, "Multitask Learning," Machine Learning, vol. 28, pp. 41-75, 1997. DOI: 10.1023/A:1007379606734
- [20] A. Mittal, "A Survey of NVIDIA Jetson Technology Based Deep Learning Applications," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2019.
- [21] Y. LeCun, L. Bottou, Y. Bengio et P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998. DOI: 10.1109/5.726791
- [22] P. Ratan, " What is the Convolutional Neural Network Architecture? ", Analytics Vidhya, publié le 28 oct. 2020, mis à jour le 1er mai 2025. [En ligne]. Disponible sur : <https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/> [Consulté le 17 mai 2026]
- [23] V. Ayyadevara et Y. Reddy, The Deep Learning with Keras Workshop: Learn How to Define and Train Neural Network Models with Just a Few Lines of Code. Birmingham : Packt Publishing, 2020. [Catalogue en ligne, Stanford Libraries]. Disponible sur : <https://searchworks.stanford.edu/view/13686145>
- [24] V. Nair et G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," ICML, 2010.
- [25] K. He, X. Zhang, S. Ren et J. Sun, "Deep Residual Learning for Image Recognition," IEEE CVPR, 2016. DOI: 10.1109/CVPR.2016.90
- [26] S. Mukherjee, " The Annotated ResNet-50 ", TDS Archive — Medium, 18 août 2022. [En ligne]. Disponible sur : <https://medium.com/data-science/the-annotated-resnet-50-a6c536034758>
- [27] P. Majumder (Paravision Lab), " Supercharge Your AI: ResNet50 Transfer Learning Unleashed ", Medium, 12 sept. 2024. [En ligne]. Disponible sur : <https://medium.com/@paravisionlab/supercharge-your-ai-resnet50-transfer-learning-unleashed-b7c0e40976c4>

- [28] S. Singh, " Evolution of YOLO Object Detection Model From V5 to V8 ", Labellerr AI Blog, publié le 6 juin 2024, mis à jour le 8 août 2025. [En ligne]. Disponible sur : <https://www.labellerr.com/blog/evolution-of-yolo-object-detection-model-from-v5-to-v8/>
- [29] D. P. Kingma et J. Ba, "Adam: A Method for Stochastic Optimization," ICLR, 2015. arXiv:1412.6980
- [30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.
- [31] P. Micikevicius et al., "Mixed Precision Training," ICLR, 2018. arXiv:1710.03740
- [32] J. de Dieu Nyandwi, " Early Stopping Explained! ", Medium, 20 nov. 2021. [En ligne]. Disponible sur : <https://jeande.medium.com/early-stopping-explained-62eebce1127e>