République du Sénégal
Ministère de l'Enseignement Supérieur,
de le Recherche et de l'Innovation
(MESRI)

UVS
UNIVERSITE VIRTUELLE DU SENEGAL
foo nekk foofu la

PÔLE SCIENCES, TECHNOLOGIES ET NUMÉRIQUE

MATHÉMATIQUES APPLIQUÉES ET INFORMATIQUE

MÉMOIRE DE MASTER

SUJET :

Conception d'un système de détection et de classification des déchets dans l'environnement basé sur l'intelligence artificielle

PRÉSENTÉ PAR : Amy Collé BARRY

SOUS LA DIRECTION DE : Pr. Cherif DIALLO

Remerciements

Avant tout, je tiens à remercier le Bon DIEU le tout puissant et clément de m'avoir illuminé le chemin du savoir et de nous avoir donné le courage, la puissance et la volonté pour accomplir ce modeste projet.

Je tiens particulièrement à exprimer ma profonde gratitude à mon encadreur le Professeur Chérif DIALLO pour sa disponibilité, son engagement et son aide durant toute la période d'encadrement. Je désire lui témoigner ma reconnaissance et mes remerciements les plus sincères.

Je remercie tous les membres du jury pour l'honneur qu'ils nous ont fait en acceptant d'examiner ce travail. Je remercie également Dr. Ismaila SECK pour ses conseils et appuis durant tout mon travail. Je remercie aussi notre père, le Professeur Ousmane SALL, le responsable du Master IA/BDA. Ainsi tout le corps professoral de l'UNCHK.

Un énorme merci à nos familles, amies et camarades étudiants du master IA/BDA.

Merci beaucoup à vous mes chèrs amis Cheikh Abdoul Ahad Mbacké Diop , Hawoly DEME, Mariama Barry .

Dédicaces

A Mes parents sont la joie de ma vie,

Mes frères et sœurs, et toute ma famille,

Mes chers professeurs et amis,

Je dédie ce modeste travail.

Résumé

La gestion des déchets est un enjeu environnemental, sociétal et économique majeur, avec une production de déchets en constante augmentation due à l'urbanisation, la croissance démographique et les habitudes de consommation. Une mauvaise gestion des déchets a des impacts négatifs sur l'environnement, la santé publique et l'économie. Au Sénégal, par exemple, la gestion des déchets pose un défi important, avec des millions de tonnes de déchets produits annuellement**.** Dans ce domaine, la détection d'objet a connu un grand succès suite aux progrès de l'apprentissage profond (Deep Learning) considéré comme l'outils le plus puissant pour répondre au défi de la gestion responsable des déchets. Dans cette optique, nous avons développé un système automatique visant à détecter efficacement les types de déchets (carton, plastique, métal) en exploitant les réseaux du Deep Learning par l'utilisation des techniques d'apprentissage par transfert pour une meilleure performance et un temps de calcul raisonnable. Nous avons proposé deux approches sur la base des réseaux de neurones convolutifs et des Transformers, que sont respectivement les modèles YOLO11 et RT-DETR entrainés sur la base de données COCO. Nous avons mené nos expériences sur la base de données ZeroWaste composé de 4503 images. Les résultats obtenus se sont révélés satisfaisants avec une précision maximale sur les bounding box de 43% et une Précision moyenne (mAP) améliorée de 17,68%.

Mots clés : Environnement, gestion des déchets, Deep Learning, Apprentissage par transfert

Abstract

Waste management is a major environmental, societal and economic issue, with waste production constantly increasing due to urbanization, population growth and consumption habits. Poor waste management has negative impacts on the environment, public health and the economy. In Senegal, for example, waste management poses a significant challenge, with millions of tonnes of waste produced annually. In this area, object detection has enjoyed great success following advances in deep learning, considered to be the most powerful tool to meet the challenge of responsible waste management. In this context, we have developed an automatic system aimed at effectively detecting types of waste (cardboard, plastic, metal) by exploiting Deep Learning networks through the use of transfer learning techniques for better performance and reasonable calculation time. We proposed two approaches based on convolutional neural networks and Transformers, which are respectively the YOLO11 and RT-DETR models trained on the COCO dataset. We conducted our experiments on the ZeroWaste dataset composed of 4503 images. The results obtained were satisfactory with a maximum bounding box précision of 43% and an improved average precision(mAP) of 17,68%.

Sigles et Abréviations

AE : Autoencodeur

AIFI : Adaptive Intra-scale Feature Interaction (Interaction Adaptative des Caractéristiques Intra-échelle)

AP : Average Precision (Précision Moyenne)

CCFM : Cross-scale Feature Fusion Module (Module de Fusion des Caractéristiques Inter-échelles)

CLAHE : Contrast Limited Adaptive Histogram Equalization (Égalisation d'Histogramme Adaptative à Contraste Limité)

CNN : Convolutional Neural Network (Réseau de Neurones Convolutifs)

COCO : Common Objects in Context (Base de données d'objets communs)

CV : Computer Vision (Vision par Ordinateur)

DETR : Detection Transformer (Transformateur de Détection)

DL : Deep Learning (Apprentissage Profond)

FFN : Feedforward Network (Réseau à Propagation Avant)

FLOPs : Floating Point Operations (Opérations en Virgule Flottante)

FN : False Negative (Faux Négatif)

FP : False Positive (Faux Positif)

GAN : Generative Adversarial Network (Réseau Antagoniste Génératif)

GELU : Gaussian Error Linear Unit (Unité Linéaire d'Erreur Gaussienne)

GRU : Gated Recurrent Unit (Unité Récurrente à Portes)

HSI : Hyperspectral Imaging (Imagerie Hyper-spectrale)

IA : Intelligence Artificielle

IoU : Intersection over Union (Intersection sur Union)

LSTM : Long Short-Term Memory (Mémoire à Court et Long Terme)

mAP : Mean Average Precision (Moyenne Précision Moyenne)

ML : Machine Learning (Apprentissage Automatique)

MRF : Material Recovery Facility (Installation de Récupération des Matériaux)

NLP : Natural Language Processing (Traitement Automatique du Langage Naturel)

NMS : Non-Maximum Suppression (Suppression Non Maximale)

OMA : Ordure Ménagère et Assimilée

PCA : Principal Component Analysis (Analyse en Composantes Principales)

ReLU : Rectified Linear Unit (Unité Linéaire Rectifiée)

RepC3 : Reparameterized Convolutional Block (Bloc Convolutionnel Reparamétré)

RepConv : Reparameterized Convolution (Convolution Reparamétrée)

RGB : Red Green Blue (Rouge Vert Bleu)

RNN : Recurrent Neural Network (Réseau de Neurones Récurrents)

RT-DETR : Real-Time Detection Transformer (Transformateur de Détection en Temps Réel)

SGD : Stochastic Gradient Descent (Descente de Gradient Stochastique)

SiLU : Sigmoid Linear Unit (Unité Linéaire Sigmoïde)

SVM : Support Vector Machine (Machine à Vecteurs de Support)

TN : True Negative (Vrai Négatif)

TP : True Positive (Vrai Positif)

YOLO : You Only Look Once

Avant-propos

Ce mémoire s'inscrit dans le cadre de l'obtention du diplôme de Master en Big Data Analyst à l'Université Numérique Cheikh Hamidou Kane. À une époque où les défis environnementaux prennent une place de plus en plus importante dans les débats mondiaux, la nécessité de trouver des solutions innovantes et durables pour la gestion des déchets devient une priorité. Notre intérêt pour l'étude des algorithmes d'intelligence artificielle appliqués à la détection des déchets s'inscrit dans cette dynamique, en explorant comment les technologies de pointe, notamment l'apprentissage profond (Deep Learning), peuvent contribuer à améliorer la détection et la classification des déchets dans des environnements variés.

L'objectif de ce mémoire est de voir comment les technologies de l'IA peuvent contribuer à mettre en place un système de détection des déchets dans le but d'optimiser le processus de gestion des déchets. Au début le but était d'obtenir des données du Sénégal mais après une visite au site de mbeubeuss le constat qui a été fait est qu'il y'a un grand encombrement de la scène et alors il devenait casi impossible avec un temps limité d'annoter ces images correctement avec une grande ingéniosité et très bonne précision. Apres cela je suis entrée en contact avec la direction de la sonaged pour obtenir des données, ce qui malheureusement n'a pas abouti. Raison pour laquelle nous avons alors été contraint de recourir aux bases de données publiques. Ce qui représente un défi supplémentaire, malgré tout cela nous avons réussi à développer une solution qui a donné de bonnes prédictions.

Cette étude que l'on compte poursuivre avec les perspectives mise en place pourrait contribuer au projet de la sonaged qui prévoit de collaborer étroitement avec les universités et les laboratoires de recherche pour développer des solutions innovantes dans la gestion des déchets. D'aprés les propos de son nouveau directeur Khalifa Ababacar Sarr.

Liste des Figures et des tableaux

Listes des figures :

Figure 1 : Relation entre IA,ML et DL [12]. 5
Figure 2 : Système expert en ML [11]. 6
Figure 3 : Résumé des principaux algorithmes et taches d'ingénierie en ML [24]. 8
Figure 4 : Système expert en DL [11]. 9
Figure 5 : Architecture d'un réseau de neurone simple [11]. 9
Figure 6 : Représentation en noir et blanc. 17
Figure 7 : Représentation en niveau de gris. 18
Figure 8 : Représentation en couleur. 18
Figure 9 : Représentation d'une architecture CNN [42]. 19
Figure 10 : Convolution simple sans padding, stride de 1, kernel de taille 3 [43]. 20
Figure 11 : Convolution avec padding de 1, stride de 1, kernel de taille 3 [43]. 21
Figure 12 : Opération de convolution réalisée par les filtres des couches convolutives [44]. 21
Figure 13 : Opération de max-pooling à la sortie d'une couche de convolution (Taille de fenetre de pooling 2x2 et stride de 1 [44]. 22
Figure 14 : Structure dénéral des fonctions d'activation [48]. 23
Figure 15 : Illustration de la couche entièrement connectée [50]. 23
Figure 16 : Illustration de l'utilisation du calcul de la perte [54]. 25
Figure 17 : Opération de rétropropagation [55]. 26
Figure 18 : Architecture traditionnelle des RNN [57]. 27
Figure 19 : Architecture des LSTM [58]. 27
Figure 20 : Architecture des GRU [58]. 28
Figure 21 : Architecture d'un GAN [60]. 29
Figure 22 : Architecture de base d'un autoencodeur [63]. 30
Figure 23 : Architecture du modèle Transformers [66]. 32
Figure 24 : Architecture d'un transformer ViT [65]. 32
Figure 25 : Architecture de base d'un détecteur d'objet [38]. 35
Figure 26 : Processus de détection d'objet. 35
Figure 27 : Extraction avec les méthodes à deux étapes [73]. 36
Figure 28 : Extraction avec les méthodes à une étape [73]. 37
Figure 29 : Processus du NMS [73]. 37
Figure 30 : Illustration des calculs de précision et de rappel [74]. 38
Figure 31 : Représentation de la courbe P-R [75]. 38
Figure 32 : Représentation de nos données à gauche et à droite les vérités terrains correspondantes [81]. 41
Figure 33 : A gauche : Assemblage de chaque appareil d'enregistrement. A droite : Disposition de l'installation d'enregistrement dans l'environnement de recyclage [81]. 42
Figure 35 : Architecture générale des modèles YOLO [82]. 43
Figure 36 : Modules architecturaux clés de YOLO11 [85]. 43
Figure 37 : Performance de yolo11 par rapport aux versions précédentes [84]. 44
Figure 38 : Architecture globale de DETR [87]. 45

Figure 39 : Architecture du modèle RT-DETR [89]. 46
Figure 40 :Le bloc de fusion dans le CCFF [89]. 46
Figure 41 : Effet de l'augmentation des données. 48
Figure 42 : Structure des couches de yolo11 [90]. 51
Figure 43 : Apprentissage de yolo 52
Figure 44 : Structure des couches de RT-DETR [91]. 53
Figure 45 : Apprentissage de RT-DETR. 54
Figure 46 : Courbe de Précision de yolo. 56
Figure 47 : Courbe du Rappel de yolo. 56
Figure 48 : Matrice de confusion normalisée de yolo. 57
Figure 49 : Evolution des pertes de yolo. 58
Figure 50 : Evolution des performances de yolo. 58
Figure 51 : Evolution de perte en entrainement de rt-detr. 59
Figure 52 : Les courbes d'évaluation des MAR en entrainement de rt-detr. 60
Figure 53 : Matrice de Confusion de rt-detr. 60
Figure 54 : Page d'accueil de l'interface streamlit. 61
Figure 55 : Exemples de prédictions sur les données test. 62

Listes des tableaux :

Tableau 1 : Résumé de certains modeles basés sur CNN utilisés dans la classification des dechets solides municipaux [37]..... 13
Tableau 2 : Répartition du nombre total d'images et d'objets dans les ensembles de train, validation et test..... 47
Tableau 3 : Inférence sur les données de validation de yolo..... 55
Tableau 4 : Performances en validation avec rt-detr. 59
Tableau 5 : Comparaison des Performances (map) de notre modele yolo et de la source de ZeroWaste [81]..... 61

Tables des matières

Table des matières

Remerciements iii
Dédicaces iv
Résumé v
Sigles et Abréviations vi
Avant-propos viii
Liste des Figures et des tableaux ix
Introduction Générale 1
CHAPITRE 1 : Cadre Théorique et Méthodologique 3
1.1 Cadre Théorique 3
1.1.1 Les déchets environnementaux 3
1.1.2 La gestion des déchets environnementaux 3
1.2 Cadre Méthodologique 4
1.2.1 L'Intelligence Artificielle 4
1.2.2 L'Apprentissage Automatique 5
1.2.3 L'Apprentissage Profond 8
1.2.4 La vision par ordinateur 10
1.3 Etat de l'art 11
CHAPITRE 2 : Approche de l'apprentissage profond pour la détection d'objet 16
2.1 Introduction à la détection d'objet 16
2.1.1 Définition d'une image numérique 16
2.1.2 Caractéristiques d'une image numérique 16
2.1.3 Les types de représentation d'une image numérique 17
2.2 Les architectures d'apprentissage profond 18
2.2.1 Les Réseaux de Neurones Convolutifs (CNN) 19
2.2.2 Les Réseaux de Neurones Récurrents (RNN) 26
2.2.3 Les Réseaux Antagonistes Génératifs (GAN) 28
2.2.4 Les Autoencodeurs 29
2.2.5 Les Transformers 30

2.3 Transfert Learning 33
2.4 Les Concepts de base de la détection d'objets 33
2.4.1 Architecture d'un détecteur d'objet 34
2.4.2 Extracteurs de caractéristiques 35
2.4.3 Les mesures de performance d'un modèle de détection 37
2.5 Conclusion 38
CHAPITRE 3 : Méthodologie et Implémentation 39
3.1 Introduction 39
3.2 Outils utilisés 39
3.2.1 Langage utilisé 39
3.2.2 Environnement d'implémentation et Bibliothèques utilisés 39
3.3 Dataset utilisé 40
3.4 Méthodologies proposées 42
3.4.1 Modèle yolov11 42
3.4.2 Détection de bout en bout avec les Transformers (Detr) 45
3.5 Prétraitement des données 47
3.5.1 Chargement des données 47
3.5.2 Traitement des données 48
3.6 Apprentissage des modèles 49
3.6.1 YOLO11 49
3.6.2 RT-DETR 52
3.7 Résultats et Discussions 55
3.7.1 YOLO11 55
3.7.2 RT-DETR 59
3.8 Prédiction avec l'interface Streamlit 61
3.9 Conclusion 62
Conclusion Générale 63
Glossaire 64
Bibliographie 67
Annexes A YOLO11 : 72
Annexes B RT-DETR : 73

Introduction Générale

Sans aucun signe de ralentissement, les déchets dans le monde augmentent rapidement en raison de différents facteurs tels que l'urbanisation, la croissance démographique, la croissance économique, ainsi que les habitudes d'achats des consommateurs. Ce problème environnemental concerne tous les pays sans exception. Ces déchets connus comme un problème répandu à l'échelle mondiale nécessitent de sérieux efforts pour la collecte et la bonne gestion. Si rien ne change rapidement, la production mondiale de déchets augmentera de 70 % d'ici 2050, selon un nouveau rapport de la Banque mondiale [1]. Une mauvaise gestion des déchets a des impacts négatifs à diverses échelles locales et mondiales sur l'environnement et les organismes en termes de santé, d'esthétique et d'économie. Au niveau local, au Sénégal, la gestion des déchets pose un défi majeur. Par exemple 2.789.000 tonnes de déchets sont produites annuellement selon un rapport sur la campagne nationale de caractérisation des déchets ménagers [2], ce qui augmente les risques pour la santé publique et contribue à la dégradation des écosystèmes locaux. La propreté n'est pas seulement un défi du quotidien. Elle n'est pas simplement un enjeu environnemental. La propreté est, fondamentalement, un enjeu sanitaire, une exigence socioculturelle, et également économique de haute portée pour l'image et le rayonnement international d'un pays. Par conséquent, la recherche de solutions durables pour la bonne gestion face à l'augmentation rapide des niveaux de déchets, a fait que de sérieux défis technologiques ont été mis en évidence dans les études sur les différents types de déchets et leurs triages. Toutefois, le succès des initiatives de bonne gestion et de recyclage ne repose pas uniquement sur les avancées technologiques. L'éducation des utilisateurs constitue également un élément essentiel dans la quête visant à améliorer les taux de recyclage à l'échelle mondiale. Sensibiliser le public à l'importance du recyclage et au tri correct des matériaux. Une population bien informée et motivée contribue de manière significative à la réussite de ces efforts. Cependant, les méthodes traditionnelles manquent souvent de capacités pour identifier et réagir aux incidents critiques tels que la contamination, les déversements ou le dépôt inadéquat des déchets. C'est dans ce contexte que l'intelligence artificielle (IA) intervient comme une solution innovante. L'un des avantages les plus remarquables de l'IA dans la bonne gestion des déchets est sa capacité à détourner les matières recyclables des décharges. Grâce à des capteurs avancés et à des algorithmes d'apprentissage automatique, l'IA peut détecter, classifier et gérer efficacement les matériaux recyclables dans le flux de déchets.

L'Objectif général de ce mémoire repose la création d'une technologie visant à améliorer la gestion des déchets en utilisant l'intelligence artificielle pour automatiser la détection et l'analyse des déchets dans différents environnements, qu'ils soient urbains ou naturels. Grâce à des technologies de Machine Learning, et plus précisément de Deep Learning. Des drones, caméras ou images satellites capturent des données en temps réel afin de reconnaître et caractériser les types de déchets (plastique, métal, papier, etc.).

Les objectifs spécifiques sont :

1. Collecte d'images de déchets ;
2. Optimisation de quelques algorithmes de Deep Learning pour la détection et la classification des déchets. Et en choisir le plus adapté ;
3. Intégration de notre modèle dans une interface utilisateur pour détecter les déchets en temps réel.

En vue d'atteindre ces objectifs, les travaux seront organisés autour de trois chapitres. Le premier chapitre situe le sujet dans son cadre théorique et méthodologique, en énonçant les notions de base de la gestion des déchets et des concepts de base sur la méthodologie. Ce chapitre fait aussi l'objet d'une revue de la littérature sur la détection et la classification des déchets avec l'intelligence artificielle. Le second chapitre fera l'objet d'une étude appondit des concepts méthodologiques énoncés dans le chapitre1. Et enfin dans le troisième chapitre Nous présenterons en détails les approches proposées, puis nous présenterons les résultats et en discuterons.

CHAPITRE 1 : *Cadre Théorique et Méthodologique*

1.1 Cadre Théorique

1.1.1 Les déchets environnementaux

L'environnement peut être défini comme la somme totale de tous les éléments vivants et non vivants et de leurs effets qui influencent la vie humaine. Cela englobe à la fois le monde naturel, incluant les écosystèmes de la terre, le climat et la biodiversité, ainsi que l'environnement construit, constitué de structures et d'éléments fabriqués par l'homme[3]. L'environnement assure différentes fonctions. Elle nous fournit des ressources renouvelables et non renouvelables, maintient la vie, améliore la qualité de vie et assimile les déchets. Les déchets recouvrent beaucoup de réalités différentes, qui dépendent de leur nature, leur type, leur dangerosité, leur volume, leur production[4]. Sur l'ensemble du cycle de vie du déchet, ces variations déterminent les modes de collecte ainsi que les pratiques de traitement les plus adaptées. Les enjeux environnementaux, sanitaires, économiques, sociétaux, sont aussi des leviers qui influent sur les politiques mises en place[5]. Les déchets environnementaux sont généralement distingués selon leur nature : dangereux, non dangereux et inertes. Une autre distinction se réfère à l'origine de production qui les différencie incluant notamment les déchets ménagers et les assimilés[6]. Les ordures ménagères et assimilées (OMA) comprennent les ordures ménagères résiduelles (poubelles ordinaires des ménages) et les déchets collectés séparément en porte à porte ou en apport volontaire (verre, emballages et papier issus du tri séparé, biodéchets). Elles correspondent aux déchets de routine produits quotidiennement par les habitants.

1.1.2 La gestion des déchets environnementaux

La question de la gestion des déchets est devenue de plus en plus complexe autant pour les pays du Nord que pour ceux du Sud. De plus, la mondialisation des échanges, des techniques et des idées, replace la question de la gestion des déchets dans la liste des préoccupations majeures des sociétés. A ce titre, la problématique de la gestion des déchets s'insère dans un contexte plus global de développement durable et est soumise à des enjeux économiques, sociaux et environnementaux. Une gestion efficiente des déchets est une source de richesse (utilisation des matériaux de valorisation comme matières premières et création d'emplois) et de bien-être social. La population urbaine pauvre des villes du Sénégal souffre de manière disproportionnée des problèmes liés à la faible gestion des déchets solides : pollution de l'air, des eaux environnantes, etc.[7]. L'économie informelle de la récupération, de la collecte et de la valorisation des déchets solides urbains est à la fois une stratégie de survie pour les populations défavorisées dans de nombreux pays en développement, et une partie importante des systèmes de gestion des déchets et de recyclage. Au niveau mondial, on estime que le nombre de travailleurs dans la gestion informelle des déchets impliqués dans la collecte, le tri,

et de recyclage est d'au moins 15 millions[8]. Aujourd'hui, la plupart des économies recourent de plus en plus, à une gestion intégrée et durable des déchets qui implique la mise en place d'un système optimisé de pratiques de gestion de déchets pour un territoire donné[9]. Entre 2016 et 2017 l'UCG indiquent 636 368 et 734 684 tonnes d'ordures ménagères évacuées à la décharge de Mbeubeuss. A cela, s'ajoute 33 816 et 40 402 tonnes de déchets de marchés et 19 418 et 11 930 tonnes de déchets industriels banals (collecte privée) pour 2016 et 2017. Les quantités totales enregistrées à Mbeubeuss seraient supérieures à la production moyenne de 171,82 kg/hab./an (UCG, 2016). La production globale journalière d'ordures ménagères est de 6803,52 t/j au niveau du territoire national selon [africacheck, 2022]. Pour faire face à des quantités aussi massives de déchets, l'État du Sénégal, dans le cadre du Programme National de Gestion des Déchets (PNGDS) a élaboré et mis en œuvre une stratégie pour une gestion intégrée et durable des déchets. Et l'une des objectifs de ce programme est de permettre à toutes les collectivités territoriales du pays, de se doter d'un système de gestion des ordures ménagères formel, durable et moderne. La caractérisation des ordures ménagères et assimilées réalisée par l'UCG en 2014 et 2015 a montré que la poubelle des ménages sénégalais est composée en majorité d'éléments fins (sable, éléments organiques de petite taille) (53%) et de putrescibles (restes alimentaires, tout produit périssable) (12%), suivent ensuite, les plastiques (9%) et les complexes (produit composé de plusieurs matériaux) (5%). Les Déchets Ménagers Spéciaux (DMS), déchets dangereux des ménages restent la catégorie la moins représentée avec 0,84% du flux total des déchets produits dans le pays (UCG 2016)[7]. La collecte, le traitement et l'élimination efficaces des déchets solides municipaux dépendent d'une prévision précise des caractéristiques des déchets. La nécessité d'une bonne gestion des déchets ne se limite pas à la collecte et à l'élimination appropriée des déchets. Elle se poursuit jusqu'au niveau des décharges et à la quantité que nous pouvons éventuellement recycler. Ainsi, les métaux, les cartons ou papiers et les articles en verre peuvent être réutilisés et les plastiques peuvent être détournés des décharges, qui autrement conduisent à l'étouffement de la terre. Toutefois tout cet argumentaire précédent, conduit à un mouvement improductif pour vider les conteneurs, à une consommation inutile de ressources par les camions et les travailleurs de collecte des déchets, et à une quantité notable de déchets laissés autour de poubelles trop remplies produisant un environnement de vie malsain. La gestion des déchets a un impact significatif sur le capital environnemental et sanitaire de toute l'humanité. Elle comprend toutes les tâches et procédures nécessaires pour contrôler les déchets depuis leur création jusqu'à leur élimination. Par conséquent, la conception d'approches modernes et à jour, intelligentes et respectueuses de l'environnement pour les tâches de gestion des déchets devient un besoin crucial.

1.2 Cadre Méthodologique

1.2.1 L'Intelligence Artificielle

Employée pour la première fois en 1955 par le mathématicien John McCarthy, l'expression "intelligence artificielle" est aujourd'hui au centre de tous les débats. Selon lui qui est le père de L'Intelligence Artificielle (IA), "The science and engineering of making intelligent machines,

especially intelligent computer programs" [10]. L'IA consiste à étudier comment le cerveau humain apprend, pense et décide pour résoudre un problème. Puis, les résultats de cette étude seront utilisés comme base du développement de systèmes intelligents pour résoudre d'autre problème[11]. Elle a pour but de rendre la machine indistinguable de l'être humain et englobe l'apprentissage automatique et l'apprentissage profond (voir figure 1).

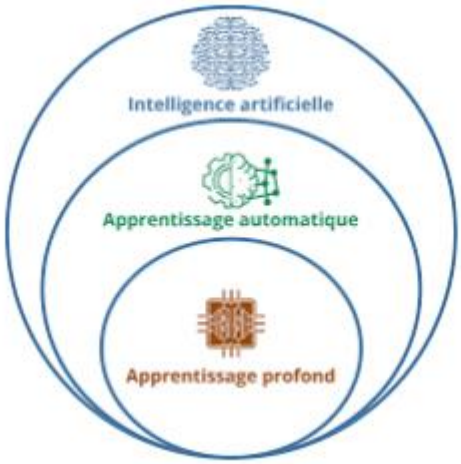


Figure 1 : Relation entre IA,ML et DL [12].

Elle procède essentiellement par apprentissage et par raisonnement. Plus spécifiquement, l'IA est souvent employée pour réaliser des tâches telles que la reconnaissance de motifs, la perception visuelle, la prise de décision, la compréhension du langage, la traduction des langues, etc. [13]. Etant une discipline en constante évolution, s'appuyant sur un ensemble de facteurs et disciplines qui contribue à sa progression, notamment l'informatique, la linguistique, la psychologie, la biologie, l'ingénierie et les mathématiques. L'intelligence artificielle a été scindée en plusieurs domaines et sous-domaines donnant lieu pour chacun d'entre eux à des recherches spécifiques, des solutions diverses et à des applications différentes. Notamment la branche cognitive de l'intelligence artificielle qui s'appuie sur des domaines et outils tels que l'apprentissage et l'auto-apprentissage, les réseaux de neurones, la langue naturelle, les arbres de décision[14]. Ainsi chacun de ces domaines s'attaquant à des défis spécifiques et exploitant des approches techniques distinctes.

1.2.2 L'Apprentissage Automatique

L'apprentissage automatique est un domaine de l'IA qui présente un potentiel énorme pour la transformations de nombreux domaines de la vie et de la science[15]. Encore appelé Machine Learning c'est un domaine multidisciplinaire qui couvre l'informatique, la théorie des probabilités, les statistiques, la théorie approximative et les algorithmes complexes, et ses théories et méthodes ont été largement utilisées pour résoudre des problèmes complexes dans les applications d'ingénierie[16]. L'apprentissage automatique, permet à des systèmes d'apprendre et de s'améliorer à partir de l'expérience sans être explicitement programmés

pour chaque tâche [3]. Cette technologie repose sur des algorithmes qui reçoivent et analysent des données d'entrée pour prédire des sorties, reconnaître des modèles ou prendre des décisions[17].

L'intelligence artificielle a débuté avec la construction des systèmes experts. Ces systèmes experts sont des systèmes très complexes dont la construction nécessitait des milliers d'ingénieurs, qui apprenaient à l'ordinateur comment accomplir des tâches par la programmation rigoureuse de nombreuses règles. Ils sont généralement constitués d'une base de connaissances (contenant des objets, des attributs et des conditions) et d'un moteur d'inférence capable d'utiliser des faits et des règles pour produire de nouveaux faits, jusqu'à parvenir à la réponse à la question posée (voir Figure 2). En ML, l'extraction des caractéristiques et la classification sont faites de manière séparée.

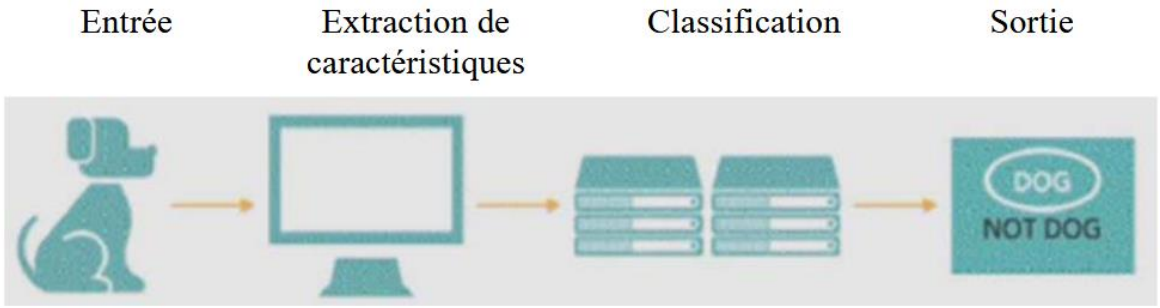


Figure 2 : Système expert en ML [11].

Il existe principalement trois types d'apprentissage en Machine Learning :

- **L'apprentissage supervisé** : il permet le développement de modèles de prédiction qui se basent sur des ensembles d'exemples. L'ensemble d'exemples se compose de données étiquetées, c'est-à-dire d'objets et de leurs classes associées. Cet ensemble d'exemples étiquetés constitue donc l'ensemble d'apprentissage qui inclut à la fois les entrées et les sorties désirées. Lorsque ces algorithmes attribuent une étiquette à chaque entrée et que l'ensemble des valeurs de sortie est un nombre entier, on parle de modèle de classification. Mais si ceux-ci prédisent des valeurs numériques et que la sortie que l'on cherche à estimer est une valeur continue de type réel alors on parle de modèle de régression. Il existe donc plusieurs algorithmes d'apprentissage supervisé qui ont été développés pour la classification et la régression. Les algorithmes typiques de classification incluent les machines à vecteurs de support (SVM), les réseaux de neurones, les Naives bayes[18]. Dans un exemple de classifications d'image ces algorithmes peuvent être utilisés comme classificateurs basés sur des informations extraites des pixels de ces images. Concernant les algorithmes de régression nous pouvons citer la régression linéaire et logistique, les arbres de décisions ou Decision Trees qui peuvent aussi être utilisés pour la classification, Les forets aléatoires ou bien Random Forest. Par exemple, lorsque vous disposez d'un ensemble de données sur les maisons et que vous êtes invité à prédire leurs prix, il s'agit d'une tâche de régression car le prix sera une sortie (valeur) continue.

- **L’apprentissage non supervisé** : selon les auteurs de [19], on parle d’apprentissage non supervisé car, contrairement à l'apprentissage supervisé ci-dessus, il n'y a pas de bonnes réponses et il n'y a pas d'enseignant. Les algorithmes sont laissés à eux-mêmes pour découvrir et présenter la structure intéressante des données. Les algorithmes d'apprentissage non supervisé apprennent peu de caractéristiques à partir des données. Lorsque de nouvelles données sont introduites, elles utilisent les caractéristiques précédemment renseignées pour reconnaître la classe des données. Il est principalement utilisé pour le clustering et la réduction des caractéristiques. Les techniques courantes sont : le clustering, comme l’algorithme K-means, la réduction de dimensionnalité ou de caractéristiques, comme l’analyse en composantes principales (PCA) ou l’analyse factorielle. Exemples d’utilisation : segmentation de la clientèle pour le marketing, détection de fraude dans les données financières en identifiant les transactions qui s'écartent des modèles attendus, traitement de langage naturel (TAL) notamment la modélisation de sujets, le regroupement de documents et l'étiquetage des parties du discours [20].

- **L’apprentissage par renforcement** : l'apprentissage par renforcement est le problème auquel est confronté un agent qui apprend un comportement par le biais d'interactions par essais et erreurs avec un environnement dynamique[21]. L’agent est exposé à un environnement où il doit prendre des décisions, et il reçoit des récompenses ou des pénalités en fonction de ses actions. L'apprentissage par renforcement est l'un des trois paradigmes de base de l'apprentissage automatique, aux côtés de l'apprentissage supervisé et de l'apprentissage non supervisé. On distingue trois types de méthodes d'apprentissage par renforcement : en commençant par l'approche de la programmation dynamique, qui est un ensemble d'algorithmes qui peuvent être utilisés pour calculer la politique optimale à partir d'un modèle complet de l'environnement où il est appliqué, ensuite les méthodes de Monte Carlo font la moyenne des rendements complets à partir de séquences d'actions échantillonnées plutôt que de toutes les séquences d'actions possibles, comme dans la programmation dynamique, et l’apprentissage par différence temporelle est une méthode qui combine des aspects de la programmation dynamique et des méthodes de Monte Carlo, et qui est considérée comme une idée centrale et innovante dans l'apprentissage par renforcement[22]. Applications typiques : Jeux, comme les échecs ou le Robotique[23], optimisation de systèmes de recommandation en ligne pour des tâches nécessitant des ajustements dynamiques.

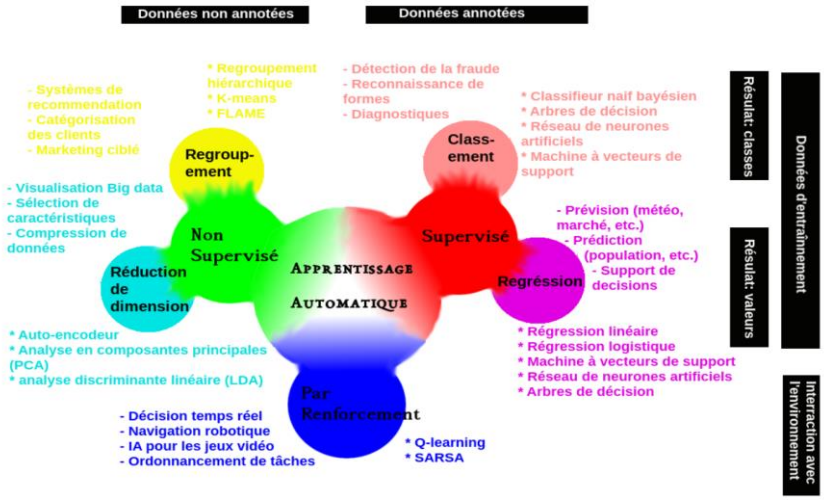


Figure 3 : Résumé des principaux algorithmes et taches d'ingénierie en ML [24].

L'émergence du big data, du cloud computing, des réseaux neuronaux artificiels et de l'apprentissage automatique a permis aux ingénieurs de créer une machine capable de simuler l'intelligence humaine. S'appuyant sur ces technologies, on peut qualifier d'intelligence artificielle (IA) les machines capables de percevoir, de reconnaître, d'apprendre, de réagir et de résoudre des problèmes.

1.2.3 L'Apprentissage Profond

Le Deep Learning, qui fait appel à la fois aux connaissances en neurosciences, aux mathématiques et aux progrès technologiques, est aujourd'hui plébiscité comme une véritable révolution dans le domaine de l'intelligence artificielle. Il a déjà permis d'immenses progrès et de multiples applications dans les domaines de la reconnaissance faciale et vocale, de l'étiquetage d'images, du traitement automatisé du langage ou encore de la vision par ordinateur. D'après les travaux de (Deng L et al.), le « Deep Learning » est une classe de techniques d'apprentissage automatique, il appartient au domaine de « Machine Learning » dans lesquelles de multiples couches de traitement de calcul itératif dans des architectures hiérarchisées supervisées sont exploitées pour les algorithmes d'apprentissage non supervisé pour des tâches d'analyse et de classification[25]. La famille des méthodes d'apprentissage en profondeur s'enrichit de plus en plus, englobant celles des réseaux de neurones, des modèles probabilistes hiérarchiques, ainsi que de nombreux algorithmes d'apprentissage des fonctionnalités supervisés et non supervisés[26]. Elle se distingue principalement de l'apprentissage automatique par ces deux caractéristiques que sont la performance et l'extraction des caractéristiques. L'apprentissage profond combine la puissance de calcul et les réseaux neuronaux, tandis que l'apprentissage automatique tire parti des algorithmes.



Figure 4 : Système expert en DL [11].

Les algorithmes en apprentissage profond sont conçus pour mimer les actions du cerveau humain grâce à des réseaux de neurones. Ces réseaux sont constitués de couches interconnectées dont le nombre renseigne sur la profondeur du réseau. Les réseaux de neurones modélisent les relations complexes et non linéaires entre des données, et ont la capacité d'apprendre et de comprendre ces relations complexes à partir des données en ajustant leurs paramètres durant la phase d'entrainement. Dans la figure 5 on peut voir les composant de base d'un neurone artificiel et leur fonctionnement :

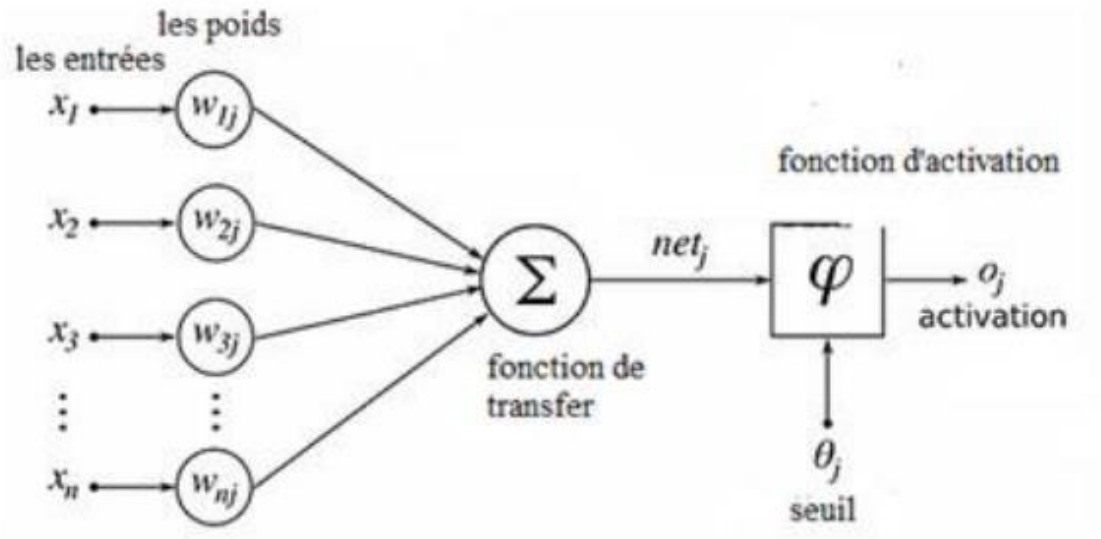


Figure 5 : Architecture d'un réseau de neurone simple [11].

On a les valeurs en entrées (x) qui sont les caractéristiques, la fonction de combinaison calcule la somme pondérée de poids (w). Le résultat n de la somme pondérée s'appelle le niveau d'activation du neurone, lorsque le niveau d'activation atteint ou dépasse le seuil, alors l'argument devient positif (ou nul). Sinon, il est négatif.

Les poids (weight) sont des pondérations des données d'entrée. Les poids négatifs réduisent la valeur d'une sortie. Lorsqu'un réseau neuronal est entrainé sur l'ensemble d'apprentissage, il est initialisé avec un ensemble de poids. Ces poids sont ensuite optimisés pendant la période d'apprentissage et les poids optimaux sont produits. Le biais est simplement une valeur constante (ou un vecteur constant) qui est ajoutée au produit des entrées et des poids. Permet de mieux généraliser à partir des données.

Très performants dans de nombreuses tâches, ils dépassant les méthodes traditionnelles dans des domaines tels que la reconnaissance d'images et vision par ordinateur, la reconnaissance vocale et synthèse vocale, le traitement du langage naturel (NLP), la reconnaissance de caractères et de signatures, les prédictions financières, la traduction automatique (NLP), le diagnostic médical assisté par ordinateur, les jeux vidéo et simulation, les véhicules autonomes, l'automatisation industrielle et la robotique [11].

1.2.4 La vision par ordinateur

La vision par ordinateur (Computer Vision en anglais) est une discipline qui a fait son apparition au début des années 1970 et qui est utilisée pour son potentiel et ses nombreuses applications. La vision par ordinateur comprend des méthodes et des techniques grâce auxquelles des systèmes de vision artificielle peuvent être construits et utilisés de manière raisonnable dans des applications pratiques. Ces méthodes de l'informatique comprennent les logiciels, le matériel et les techniques d'imagerie nécessaires (Davies, 2005). La vision par ordinateur est un domaine d'étude qui cherche à développer des techniques permettant aux ordinateurs de "voir" et de comprendre le contenu des images numériques telles que les images et les vidéos[11]. Lorsque la vision par ordinateur a vu le jour au début des années 1970, elle était perçue comme la composante de perception visuelle d'un programme visant à imiter l'intelligence humaine et à doter les robots d'un comportement intelligent. À l'époque, certains des premiers pionniers de l'intelligence artificielle et de la robotique (dans des lieux tels que le MIT, Stanford et CMU) pensaient que résoudre le problème de « l'entrée visuelle » serait une étape facile. Mais nous savons maintenant que ce problème est légèrement plus difficile que cela.

La vision par ordinateur se distinguent du traitement d'image. [27]Le traitement d'images implique la manipulation de données d'image pour l'affichage, c'est le processus de création d'une nouvelle image à partir d'une image existante[11]. Le contenu est généralement simplifié ou amélioré d'une manière ou d'une autre, mais il ne s'agit pas de comprendre le contenu de l'image. Il est utilisé pour deux objectifs principaux : (a) Améliorer l'apparence visuelle des images pour un observateur humain. (b) Préparer les images pour la mesure des caractéristiques et des structures présentes. Tandis que la vision par ordinateur traite des données d'image utilisées par un ordinateur, telles que l'acquisition, le traitement, la classification, etc.

La vision par ordinateur est utilisée pour créer un modèle du monde réel à partir d'images. Un système de vision par ordinateur récupère des informations utiles sur une scène à partir de ses projections bidimensionnelles.

Vision par ordinateur = Géométrie + Mesure + Interprétation[27].

Il existe de nombreuses applications de vision par ordinateur, nous pouvons citer :

- Classification : Il suffit de classer l'objet principal dans l'image.
- Classification + Localisation : Nous voulons aussi connaître la boîte de délimitation de l'objet principal.
- Détection d'objets : Il y a plusieurs objets dans l'image, et nous voulons connaître la classe et la zone de délimitation de chaque objet pour toutes les classes connues.
- Segmentation sémantique : Classer chaque pixel dans un ensemble fixe de catégories sans différencier les instances d'objet.

- ❖ Segmentation d'instance : Classer les objets individuels et les localiser.

La vision par ordinateur est essentielle pour automatiser le tri des déchets, car elle permet la segmentation des objets de différents types de matériaux tels que le plastique souple, le plastique rigide, le métal, le carton, etc.

1.3 Etat de l'art

Cette section est dédiée à la présentation de quelques travaux dans le domaine de la détection d'objet (plastique, carton, métal, etc.) dans des images de déchets.
Une grande variété de techniques a été explorées pour exploiter les images RVB pour l'identification des types de déchets. Cependant, la quantité limitée de données existantes et exploitables dans le cadre de la détection simultanée à temps réel des différents types de déchets pose un défi pour la formation de modèles basés sur les données. Les méthodes courantes impliquent une formation et des tests pixel par pixel sur un ensemble réduit d'images, ce qui entraîne des fuites d'informations et des capacités de généralisation peu-optimales. La difficulté de segmenter, localiser et de classer automatiquement des déchets présente un grand intérêt pour la communauté de la vision par ordinateur. La majorité des approches proposées sont basées sur des algorithmes d'apprentissage profond. Ici nous présentons certains parmi les travaux dans la littérature qui traitent de la tâche spécifique de classification, de détection et de segmentation des déchets.

Les auteurs de [28] ont proposé une approche de vision par ordinateur pour classer les déchets en catégories. L'objectif est de prendre des images des déchets et de les classer en six catégories : verre, papier, métal, plastique, carton et ordures. L'algorithme Machines à vecteurs de support (SVM) a obtenu de meilleurs résultats que le CNN. Elle a obtenu une précision de test de 63 % en utilisant une répartition des données d'entraînement/test de 70/30. L'erreur d'entraînement était de 30 %. Ils ont démontré que la SVM est un algorithme relativement plus simple que le CNN, ce qui peut expliquer son succès dans cette tâche. Par contre les données utilisées n'ont pas été prise dans un environnement typique de gestion de déchets et ne permettent pas une détection de plusieurs types dans l'image.

Dans l'article [29], les auteurs ont adopté une stratégie d'apprentissage profond pour la détection automatique des déchets. En formant un cadre open source Faster R-CNN avec un réseau de proposition de région et un algorithme de réseau ResNet, ils entrainent leur modèle sur la base d'un modèle COCO pré-entraîné. De plus, pour améliorer la précision de la méthode, une stratégie de fusion et d'augmentation des données est proposée. En conséquence, les expériences montrent que la méthode a une capacité de généralisation favorable et une fonction de détection de haute précision. Les critères d'évaluation AP (Average Precision) est le taux de précision moyen atteint 0,89. Mais la méthode ne détecte qu'un tas de déchets et non une classe spécifique.

Dans le [30], les auteurs ont travaillé sur la classification des déchets biologiques et recyclables à l'aide d'un réseau d'auto-encodeurs avec une méthode de sélection de caractéristiques intégrée dans les modèles de réseaux neuronaux convolutionnels, et ces ensembles de caractéristiques sont ensuite combinés. La méthode de régression de crête (RR) effectuée sur

l'ensemble de caractéristiques combiné a réduit le nombre de caractéristiques et a également révélé les caractéristiques efficaces. Les machines à vecteurs de support (SVM) ont été utilisées comme classificateurs dans toutes les expériences. La précision de classification la plus réussie dans les expériences était de 99,95%. Mais ne peut pas simplement résoudre un problème de détection.

Pour évaluer les performances de détection et de segmentation des déchets à l'aide de TACO, les auteurs de l'article [31], ont évalué Mask R-CNN sur deux tâches distinctes : (i) TACO-1, détection de déchets sans classe où l'objectif est de détecter et de segmenter uniquement les déchets ; et (ii) TACO-10, détection et classification pour distinguer ces 10 classes de détritus (bouteille, bouchon de la bouteille, canette, cigarette, gobelet, couvercle, autre, sac en plastique, languette, pop de paille). L'AP ((Average Precision) est utilisé pour une validation croisée quadruple utilisant différents scores pour classer les prédictions. Accuracy atteint 55%. Cependant les méthodes de détection à deux étapes comme Mask RCNN ne sont pas non plus adaptées au contexte de détection à temps réel.

Avec [32] EfficientDet-D2 est utilisé pour localiser les déchets et EfficientNet-B2 pour classer les déchets détectés en sept catégories. Le classificateur est formé de manière semi-supervisée en utilisant des images non étiquetées. L'approche proposée atteint jusqu'à 70 % de précision moyenne dans la détection des déchets et environ 75 % de précision de classification sur l'ensemble de données de test. Malgré une bonne précision, elles ne s'adaptent pas à un environnement typique de gestion de dechets.

Outre l'augmentation des données, les auteurs de cette étude [33] utilisent un algorithme génétique (GA) pour optimiser la couche entièrement connectée de DenseNet121 afin d'améliorer la précision de la classification des déchets de DenseNet121 sur TrashNet et propose le DenseNet121 optimisé. Les résultats montrent que le DenseNet121 optimisé a atteint la plus grande précision de 99,6 %, par rapport aux CNN d'autres études. Par contre elle se limite uniquement à la classification.

Afin d'obtenir un tri efficace des déchets, sur la base de l'apprentissage profond, le modèle de détection de cible Faster R-CNN et le modèle de classification d'images ResNet50 sont utilisés dans [34] pour identifier et former 3984 images de déchets et prédire 3552 images. Les résultats montrent que la précision de la reconnaissance des déchets est de 89,681 %, la précision moyenne de chaque prédiction des déchets est de 91,68% et la précision de chaque catégorie de prédiction d'images de déchets est supérieure à 93,3%. Grâce à l'identification, à la détection et à la prédiction de classification des images de déchets, il fournit un support de données pour la classification intelligente des déchets domestiques. Reste néanmoins une méthode qui peut poser problème dans des applications en temps réel et très fastidieuse qui prends beaucoup de temps en inférence et rencontre également des problèmes dans la détection des objets hautement déformables.

Dans le « VisDA 2022 Challenge » les auteurs de [35] utilise le SegFormer basé sur un transformateur et un réseau convolutionnel, DeepLabv2 Pour tester les capacités des modèles de vision par ordinateur dans le tri des déchets industriels, afin de répondre à l'un des plus grands défis qu'est l'extrême diversité du flux d'entrée en fonction de facteurs tels que

l'emplacement de l'installation de tri, l'équipement disponible dans l'installation et la période de l'année, qui ont tous un impact significatif sur la composition et l'apparence visuelle du flux de déchets. Une adaptation efficace des modèles à ces domaines est nécessaire pour une segmentation sémantique réussie des déchets industriels.

Les auteurs de [36] ont exploré les avantages de le perception multimodal pour la segmentation d'objets dans des scenarios réels de gestion des déchets. Ils comparent trois architectures différentes au sein de leur pipeline en accordant une attention particulière aux alternatives adaptées à l'inférence en temps réel : MiniNet-v2, SegFormer-B0, CMX un transformateur à fusion hybride basé sur SegFormer. Ils utilisent SpectraleWaste, le premier ensemble de données collectées à partir d'une installation de tri des déchets plastiques opérationnels, qui fournit des images hyper-spectrales et RGB conventionnelles synchronisées. Démontrant ainsi comment le HSI peut améliorer la perception RVB uniquement dans ces environnements industriels réalistes sans trop de frais de calcul.

Tous ces travaux cités et d'autres encore offrent de bonnes performances certes mais sont confrontées à des temps d'inférence très élevé et peuvent être sensible à des arrières plan complexe comme dans le cas d'un tapis roulant de tri de déchets.

La classification étant une partie intégrante dans le processus de détection d'objet, le tableau 1.1 ci-après fait un résumé de certains modèles basés sur CNN utilisés dans la classification des déchets.

Tableau 1 : Résumé de certains modeles basés sur CNN utilisés dans la classification des dechets solides municipaux [37].

Modèle	Sortie de classification	Taille des données	Précision (%)	Avantage	Inconvénient	Référence
CNN (VGGNet, DenseNet et NASNetLarge)	6-classification : papier, verre, métal, plastique, textile et déchets organiques	TrashNet (2527 images) et collecte manuelle (5000 images)	96,5 et 94	Combinez différents classificateurs de candidats pour améliorer la précision	Limiter la quantité de données	Huang et al. (2020a)
CNN (DenseNet amélioré)	6-classification : carton, verre, métal, papier, plastique, poubelle	TrashNet + augmentation des données (10 108 images)	99,6	Optimiser la couche entièrement connectée du CNN	Manque d'expériences comparatives	Mao et al. (2020)

Modèle	Sortie de classification	Taille des données	Précision (%)	Avantage	Inconvénient	Référence
CNN (Réseau amélioré)	6-classification : papier, verre, métal, plastique, textile et déchets organiques 3-classification : déchets organiques, inorganiques, médicaux	TrashNet (2527 images) et ressources Internet (5904 images)	94 et 98	Comparer la précision de classification de plusieurs modèles	Limiter la quantité de données	Vo et al. (2019)
CNN (DenseNet amélioré)	6-classification : carton, verre, métal, papier, plastique, poubelle	TrashNet (2527 images) + augmentation des données	81	Haute efficacité	La précision n'est pas élevée	Bircanoğlu et al. (2018)
AutoEncoder + CNN (AlexNet, GoogLeNet et ResNet) + Régression de crête + SVM	Déchets organiques ou recyclables	Ensemble de données en libre accès (25 077 images de déchets)	99,95	Haute précision	Haute complexité	Togacar et al. (2020)
CNN (AlexNet)+MLP	Recyclable ou les autres	Collecter manuellement (5000 images)	Orientation : 98,2(fixe) 91,6 (aléatoires)	Précision supérieure à celle obtenue avec CNN seul	Manque d'expériences comparatives	Chu et al. (2018)

En somme, la détection des déchets recyclables pose un défi unique en matière de vision par ordinateur, car elle nécessite la détection d'objets hautement déformables et souvent translucides dans des scènes encombrées sans le type d'informations contextuelles généralement présentes dans les ensembles de données centrés sur l'homme. Cette tâche difficile de vision par ordinateur manque actuellement d'ensembles de données ou de

méthodes appropriés dans la littérature disponible. Pour répondre à ce défi, le chapitre suivant se concentrera sur l'approche de l'apprentissage profond pour la détection d'objet. Nous y aborderons les différentes approches architecturales pour la détection des déchets.

CHAPITRE 2 : *Approche de l'apprentissage profond pour la détection d'objet*

2.1 Introduction à la détection d'objet

La référence [38] définit la détection d'objet comme une technique de vision par ordinateur qui consiste à identifier la présence et l'emplacement de toute instance d'une classe d'objets présente dans l'image ou la vidéo (via une boîte englobante associée à une classe d'objets. Elle constitue un élément essentiel de la reconnaissance visuelle, aux côtés de la classification et de la récupération d'images. De plus, la détection d'objets peut être considérée comme une combinaison de classification, de localisation et de segmentation. Il s'agit de classer correctement et de localiser efficacement un ou plusieurs objets dans une image, généralement à l'aide d'algorithmes supervisés à partir d'un ensemble d'apprentissage étiqueté suffisamment grand. Les algorithmes d'apprentissage automatique pour la détection d'objets sont largement utilisés dans l'apprentissage supervisé et peuvent résoudre de nombreux problèmes pratiques. Cependant des avancées significatives ont été réalisées grâce à une meilleure représentation des objets et à l'utilisation de modèles de réseaux neuronaux profonds. La détection d'objets a été abordée à l'aide de deux techniques, à savoir la détection en une étape et en deux étapes [38].

2.1.1 Définition d'une image numérique

Une image numérique est une image acquise, traitée et stockée en bits. Une image numérique est un tableau de pixel : chaque pixel est codé par un nombre binaire pour un niveau de gris, ou par trois nombres binaires qui correspond à une nuance de rouge, de vert et de bleu (codage RVB). Les images numériques peuvent être enregistrées dans différents formats de fichiers, tels que JPEG, PNG, GIF, TIFF, TFRECORD, etc. Les formats de l'image numérique sont très divers.

2.1.2 Caractéristiques d'une image numérique

Une image numérique se caractérisent essentiellement par sa résolution, sa profondeur de couleur, son format et son contenu. En tant qu'objet, l'image est décrite comme une suite de valeurs de pixels organisées selon un tableau de données à deux dimensions. Un tableau de valeurs en chaque entité élémentaire de l'image, le pixel (une petite carre, picture element en anglais). En image numérique, c'est le nombre de pixels qui composent l'image en hauteur (axe vertical) et en largeur (axe horizontal)[39]. La résolution d'une image est définie par un nombre de pixels par unité de longueur de la structure à numériser. Ce paramètre est défini lors de la numérisation (passage de l'image sous forme binaire). Plus le nombre de pixels par

unité de longueur de l'image à numériser est élevé, plus la quantité d'information qui la décrit est importante et plus la résolution est élevée. La résolution d'une image numérique définit le degré de détail de l'image. Ainsi, plus la résolution est élevée, meilleure est la restitution. Les pixels déterminent l'espace visuel de l'image, c'est le support formel qui permettrait l'affichage, la restitution et la manipulation subséquente des formes.

2.1.3 Les types de représentation d'une image numérique

Aux pixels affichés sur la grille, il est possible ensuite d'ajouter une coloration, c'est-à-dire des valeurs chromatiques. Actuellement, un pixel est souvent enregistré avec une profondeur de couleur de 8 bits ; ce qui permet d'ajouter trois couches de couleurs par pixel, à savoir rouge, vert, bleu. On parle alors d'une image « true color » ou d'une imade de 24-bits (un pixel correspond à 24 bits). Cette composition de couleurs a été introduite par le modèle de couleur RVB et elle est la plus utilisée. Selon la représentation de ces trois couleurs nous distinguons trois types d'images numériques : Image noir et blanc, Image niveau de gris et Image couleur.

- **Image noir et blanc** : ou image binaire où les pixels sont représentés par deux états logiques, 0 (noir) et 1(blanc). C'est un codage de l'image sur 1 bit**.**

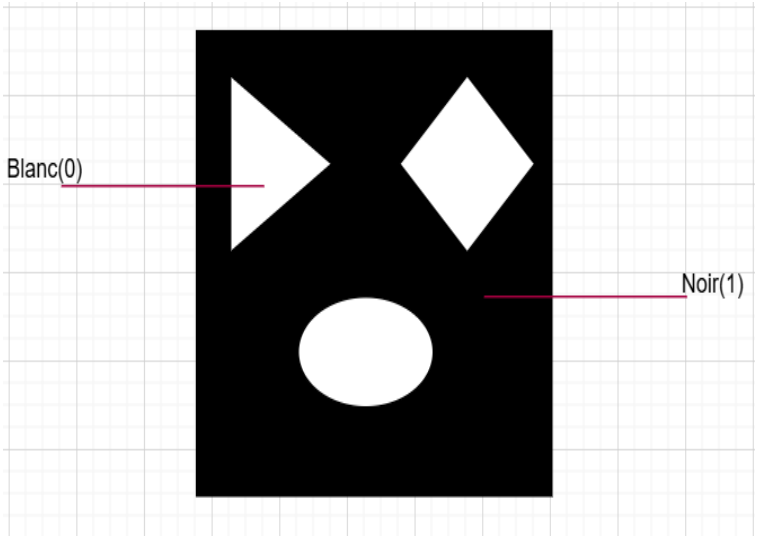


Figure 6 : Représentation en noir et blanc.

- **Image en niveaux de gris** : chaque pixel est codé sur N bits, ce qui lui affecte des valeurs entières comprises entre 0 (noir) et 2 *N-1 (blanc). Un pixel correspond à 8 bits (soit un octet) : 256 niveaux de gris.

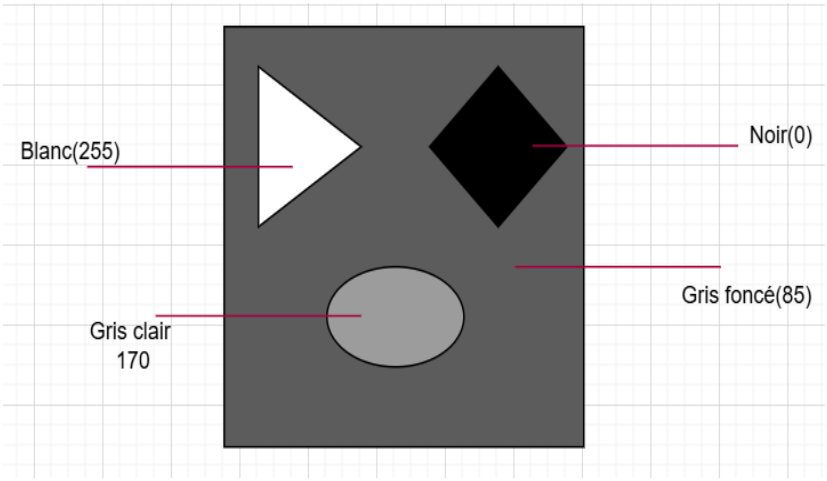


Figure 7 : Représentation en niveau de gris.

- **Image couleur :** chaque pixel correspond à la synthèse additive de 3 couleurs vert, rouge et bleu. Chaque couleur utilise 8 bits, donc chaque pixel nécessite 3 * 8 = 24 bits permettant de représenter plusieurs couleurs (256x256x256 = 16 777 216 couleurs). C'est la plus utilisée parmi les types de représentation d'image.

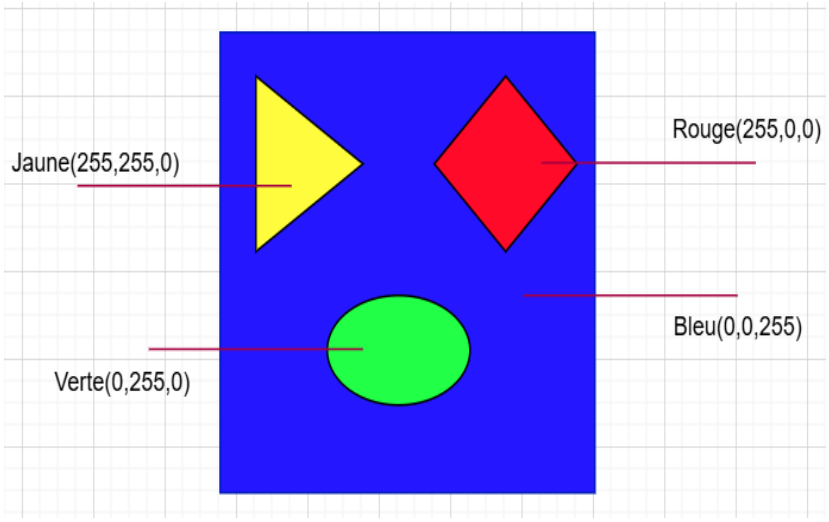


Figure 8 : Représentation en couleur.

2.2 Les architectures d'apprentissage profond

Les méthodes d'apprentissage profond basées sur les réseaux de neurones sont constituées de plusieurs couches et chaque couche contient plusieurs unités [40]. L'activation y de chaque unité représente une combinaison linéaire du vecteur d'entrée x et des paramètres apprenables w ainsi que de la base b, suivie d'une fonction de non-linéarité élément par élément f(y) = f(wx+b) où la fonction f(y) peut être une fonction sigmoïde ou une unité linéaire restreinte. Les réseaux profonds empilent plusieurs couches avec différentes structures de connexion, appelées architectures. Diverses architectures d'apprentissage en profondeur, telles que les réseaux de neurones convolutifs (CNN), les réseaux de neurones récurrents (RNN) et les réseaux contradictoires génératifs (GAN), les auto-encodeurs (AE), les transformers, ont été

proposées et utilisées avec succès dans de nombreux domaines. Il a été démontré qu'ils obtiennent des résultats de pointe sur de nombreuses tâches. Ces architectures sont fondamentales et peuvent être étendues ou combinées pour générer de nouveaux cadres pour des tâches spécifiques. Dans les sections suivantes, ces cinq architectures seront brièvement présentées pour illustrer leurs différences. Cependant nous allons plus mettre l'accent sur les CNN étant donnée les solutions proposées dans ce mémoire.

2.2.1 Les Réseaux de Neurones Convolutifs (CNN)

Les réseaux de neurones convolutionnels sont basés sur les perceptrons multicouches, et inspirés du comportement du cortex visuel des vertébrés (). Les réseaux neuronaux convolutifs (CNN), également appelés ConvNets, sont un type de réseaux neuronaux à anticipation bien adaptés aux tâches liées au domaine de la vision par ordinateur, notamment à la reconnaissance d'objets[41]. De nos jours, les CNN sont considérés comme les algorithmes les plus utilisés parmi les techniques d'intelligence artificielle (IA) basés sur les réseaux de neurones. Ils offrent une méthode efficace et performante pour la reconnaissance et l'analyse d'images ainsi que des performances accrues par rapport aux réseaux de neurones artificiels classiques.

Une architecture CNN typique comprend généralement des couches alternées de convolution, et de pooling suivi d'une ou plusieurs couches entièrement connectées à la fin. Dans certains cas, une couche entièrement connectée est remplacée par une couche de pooling moyenne globale. La disposition des composants CNN joue un rôle fondamental dans la conception de nouvelles architectures et donc dans l'obtention de performances améliorées.

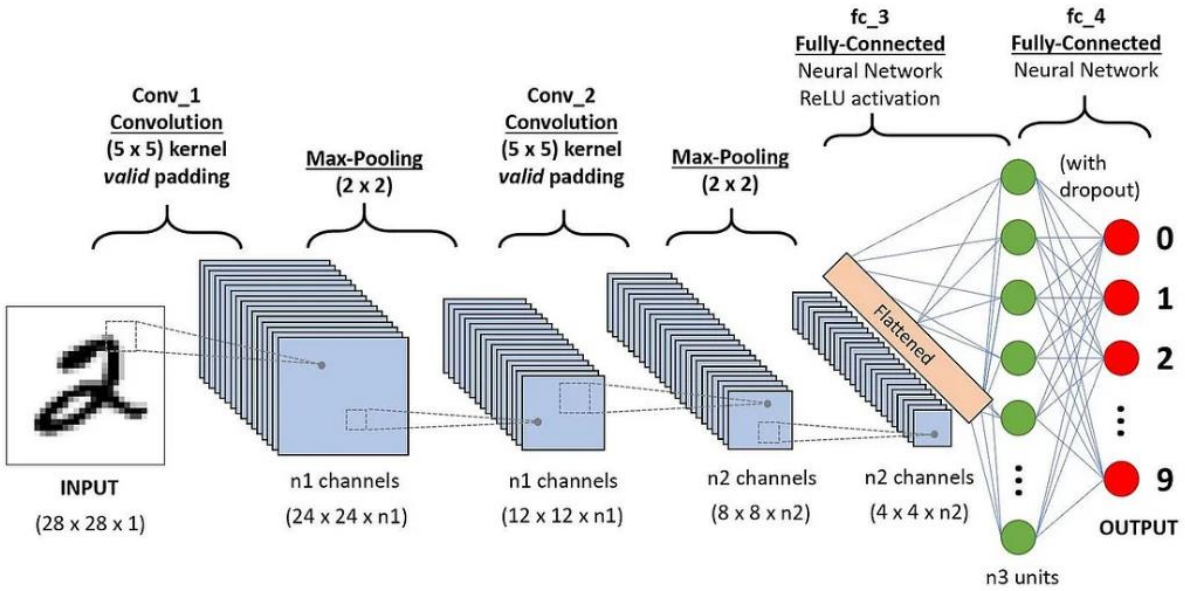


Figure 9 : Représentation d'une architecture CNN [42].

❖ **Couche de Convolution :**

Est l'élément de base d'un CNN. Son but est de mener des opérations de convolutions en appliquant des filtres sur l'images d'entrée afin d'en extraire des caractéristiques locales. Le filtre (ou noyau ou encore kernel) est une matrice qui est convoluée sur toute l'image. Pour cela, on réalise un filtrage par convolution.

Ces couches prennent en entrée un ensemble (D) de feature maps (c'est-à-dire un tenseur soit l'équivalent en 3 dimensions d'une matrice), chaque feature map étant une matrice de taille nx × ny. On a donc une entrée de taille nx × ny × D. On compte généralement plusieurs couches déterminant la profondeur du réseau. La profondeur de la couche s'agit du nombre de noyaux de convolution utilisés, ce qui correspond également au nombre de neurones associés à un même champ récepteur. Une profondeur plus élevée permet au réseau d'apprendre des caractéristiques plus complexes.

Sur cette entrée, on applique les convolutions, chacune avec un filtre de convolution de taille w×h×D,on a généralement w = h = k qu'on appelle taille du kernel. Ce sont ces filtres qui constituent les paramètres que l'on va apprendre. Chaque convolution avec un filtre produit une feature map de sortie, on a donc en sortie un ensemble de feature maps de taille n′x × n′ y × C, où n′x et n′ y dépendent de la façon dont on réalise la convolution.

En plus du nombre de filtres et de la taille du kernel, il y a deux hyperparamètres courants pour la convolution, le padding et le stride. Le padding indique combien de rangées de 0 on ajoute autour de l'entrée. Le stride indique de combien de pas on se déplace entre deux calculs de la convolution. La convolution "classique" a une sortie plus petite que l'entrée à cause de la taille du filtre que l'on doit placer à l'intérieur de la feature map d'entrée, voir figure (10). Ajouter du padding permet par exemple de retrouver la taille initiale, voir figure (11). Ajouter un stride permet de sauter des valeurs, cela correspond à faire du sous-échantillonnage de la sortie.

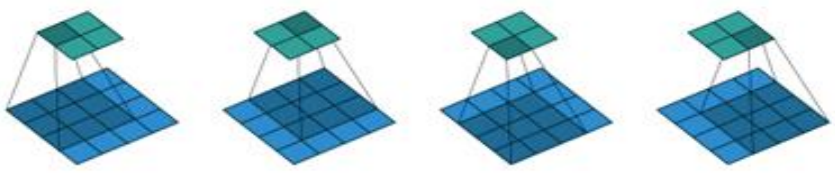


Figure 10 : Convolution simple sans padding, stride de 1, kernel de taille 3 [43].

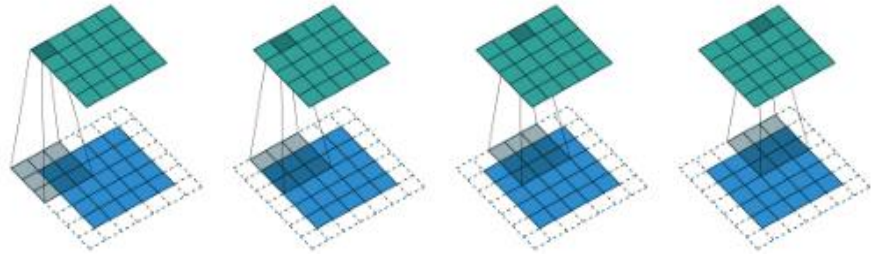


Figure 11 : Convolution avec padding de 1, stride de 1, kernel de taille 3 [43].

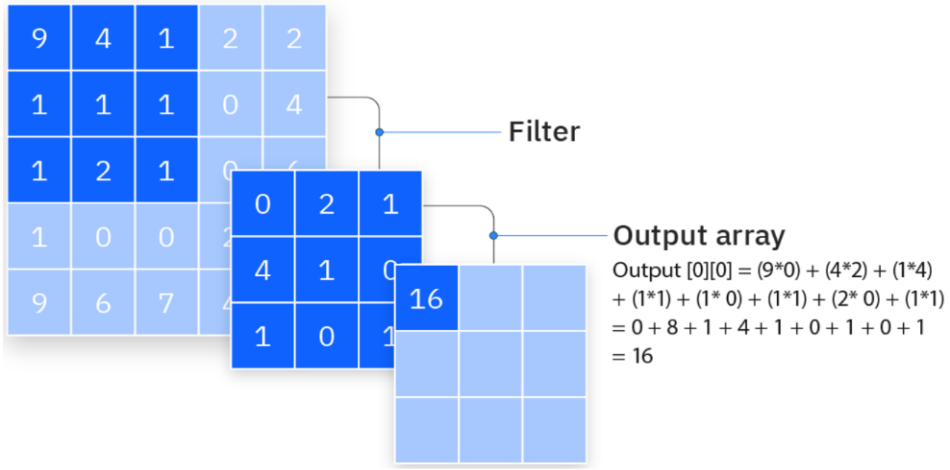


Figure 12 : Opération de convolution réalisée par les filtres des couches convolutives [44].

- **Couche de pooling :**

  Le pooling ou le sous-échantillonnage est une opération locale intéressante. Il résume des informations similaires dans le voisinage du champ récepteur et génère la réponse dominante dans cette région locale [45].Elle est généralement placée entre deux couches de convolution et effectue le sous-échantillonnage non linéaire.

  Le pooling est une fonction de sous-échantillonnage spatial. Elle prend toujours en entrée des feature maps de taille nx×ny×D, et réduit les deux premières dimensions spatiales. Le calcul s'effectue selon le même principe qu'une convolution, mais s'applique sur chaque feature map indépendamment [46]. On parcourt donc chaque feature map avec une fenêtre glissante, mais au lieu de réaliser un produit de convolution entre la fenêtre et un filtre, on réalise une opération de pooling sur la fenêtre d'entrée pour produire une valeur. On obtient donc une sortie de taille n'x × n' y × D avec n'x et n' y significativement plus petits que nx et ny (souvent d'un facteur2). Une couche de pooling a une taille de kernel (définissant la taille de la fenêtre), un stride et du padding qui se comportent comme pour la convolution, et une opération de pooling, les plus courantes étant le max pooling (on garde la valeur maximale dans la fenêtre)[45] et la couche de pooling peut également effectuer d'autres comme l'average pooling (on prend la valeur moyenne de la fenêtre). Un pooling très courant est un max pooling avec kernel de taille 2, stride de taille 2 et sans padding, comme présenté dans la figure 13.

La couche de pooling sert à réduire progressivement la taille spatiale des représentations intermédiaires, le nombre de paramètres et la quantité de calcul dans l'architecture CNN et donc à contrôler le surajustement.

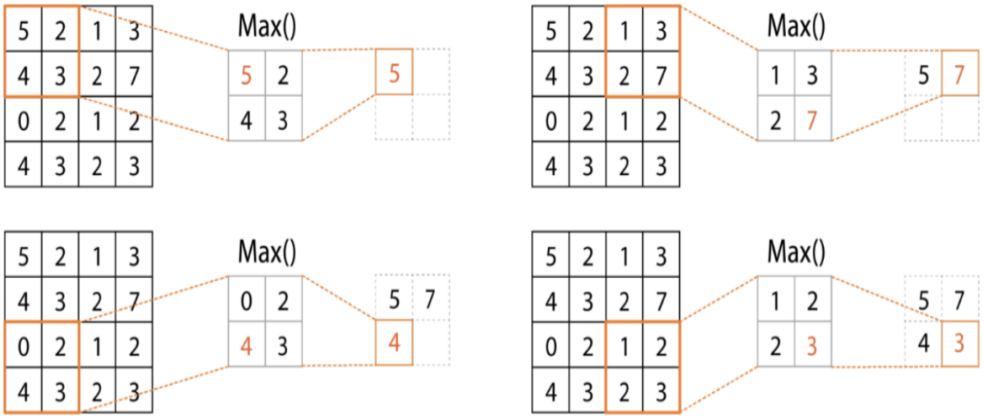


Figure 13 : Opération de max-pooling à la sortie d'une couche de convolution (Taille de fenetre de pooling 2x2 et stride de 1 [44].

- **Fonction d'activation :**

Les fonctions d'activation sont des composants fondamentaux des réseaux neuronaux convolutifs (CNN), Les fonctions d'activation ou fonction de seuillage, servent à introduire dans les couches une relation de non-linéarité. Elles permettent de déterminer le seuil en dessous duquel une couche est non-active[47].

Nous avons les fonctions d'activation qui interviennent juste après la couche de convolution et avant la couche de pooling, comme la Fonction Relu (Unité Linéaire Rectifiée). Et d'autres fonction d'activation utilisées dans les couches de sorties après la couche de fully connected, elle transforme les scores bruts en probabilités, facilitant la compréhension des prédictions. Les lus utilisées sont la fonction sigmoïd pour la classification binaire et le softmax pour la classification multiclasses.

La fonction d'activation sert à améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique sur les signaux de sortie. La fonction ReLU est la plus utilisée dans la pratique, elle est définie comme suit : $(F(x) = \max(0,x))$. La fonction ReLU force les neurones à retourner des valeurs positives.

Sans fonctions d'activation non linéaires, un CNN serait limité à des opérations linéaires, limitant considérablement sa capacité à représenter avec précision les comportements complexes et non linéaires typiques de nombreux phénomènes du monde réel[48].

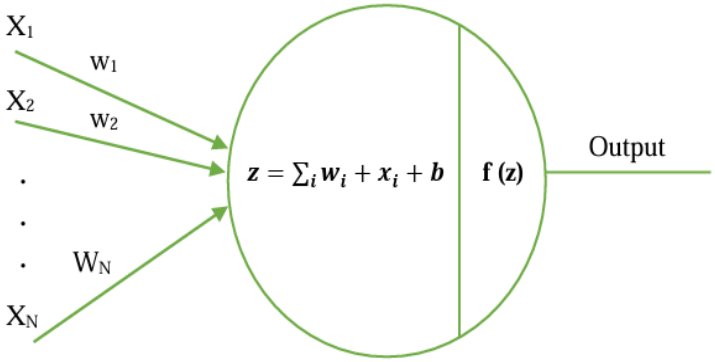


Figure 14 : Structure dénéral des fonctions d'activation [48].

❖ **Couche entièrement connectée (fully connected) :**

La couche entièrement connectée (en anglais fully connected layer) (FC) s'applique sur une entrée préalablement aplatie où chaque entrée est connectée à tous les neurones. Les couches fully connected sont typiquement présentes à la fin des architectures de CNN et peuvent être utilisées pour optimiser des objectifs tels que les scores de classe[49]. Après avoir reçu un vecteur en entrée, la couche fully connected applique successivement une combinaison linéaire, puis une fonction d'activation pour produire la sortie du neurone. Cette sortie devient ensuite l'entrée pour les neurones de la couche suivante.

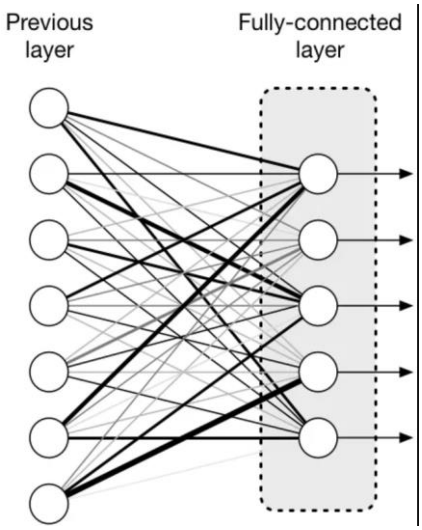


Figure 15 : Illustration de la couche entièrement connectée [50].

❖ **Batch Normalisation :**

Dans les réseaux profonds, un taux d'apprentissage (learning rate en anglais) trop élevé peut entraîner l'explosion ou la disparition des gradients, ainsi que le blocage de minima locaux médiocres. La normalisation par lots permet de résoudre ces problèmes. En normalisant les activations dans tout le réseau, cela empêche de petites

modifications des paramètres de s'amplifier en des modifications plus importantes et sous-optimales des activations dans les gradients[51].

Une couche de normalisation par lots est ajoutée après le pooling pour stabiliser l'entraînement en standardisant les valeurs des caractéristiques extraites. La normalisation par lots est incluse dans la couche convolutif pour aider à stabiliser le réseau pendant l'entraînement. Elle calcule la moyenne et la variance de toutes les caractéristiques extraites pour garantir qu'elles sont toutes à la même échelle. Les données sont ensuite aplaties en un tenseur unidimensionnel pour les préparer aux couches entièrement connectées.

- **Dropout :**

  Le Dropout introduit une régularisation au sein du réseau, ce qui améliore finalement la généralisation en sautant aléatoirement certaines unités ou connexions avec une certaine probabilité. La suppression aléatoire de certaines connexions ou unités produit plusieurs architectures de réseau affinées, et enfin, un réseau représentatif est sélectionné avec de faibles poids. Cette architecture sélectionnée est ensuite considérée comme une approximation de l'ensemble des réseaux proposés[52].

- **Couche de perte ou fonction de perte (LOSS)** :

  La couche de perte est la dernière couche d'un réseau neuronal, responsable de calculer l'erreur entre les prédictions du réseau et les valeurs réelles. Il existe plusieurs fonctions de perte différentes dans les réseaux neuronaux, telles que l'erreur quadratique moyenne, la perte d'entropie croisée, la perte charnière, etc. [53]. Le choix de la fonction de perte dépend du type de tâche pour laquelle le modèle est utilisé et du type de modèle utilisé. Par exemple, l'erreur quadratique moyenne est souvent utilisée pour les tâches de régression, tandis que la perte d'entropie croisée est couramment utilisée pour les tâches de classification. Lorsqu'il s'agit d'une tâche de classification où la variable aléatoire est discrète, avec des valeurs 0 ou 1 représentant l'appartenance ou non à une classe, la fonction de perte la plus courante et appropriée est l'entropie croisée (cross-entropy en anglais). En plus d'indiquer les performances du modèle, la fonction de perte peut fournir des informations sur d'autres aspects du comportement du modèle, tels que le sur-ajustement ou le sous-ajustement des données.

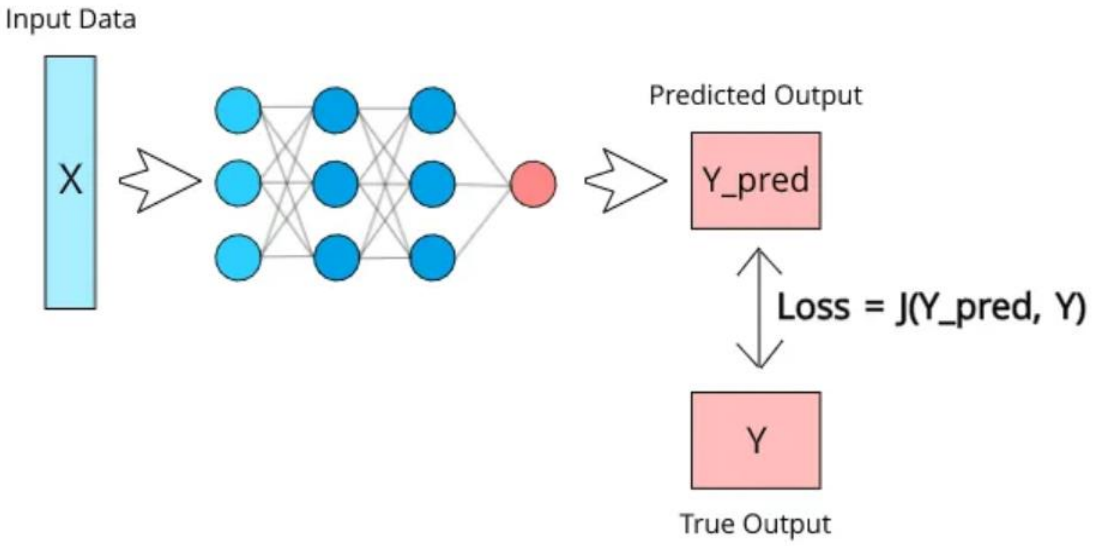


Figure 16 : Illustration de l'utilisation du calcul de la perte [54].

❖ **Algorithme de rétropropagation du gradient :**

Afin de faire un apprentissage performant, la fonction de perte doit être bien minimisée en calculant la différence entre les résultats corrects et les résultats prédits. Une fois la perte calculée, la fonction de perte est dérivée pour en trouver le minimum (si la dérivée première d'une fonction s'annule en changeant de signe en un point donné alors ce point représente potentiellement un minimum de la fonction en question). Le but ici est de trouver le signe de la dérivée afin de savoir si les poids doivent être optimisés afin de minimiser la perte et donc de faire des prédictions plus exactes. La dérivée d'une fonction avec des opérateurs tenseurs s'appelle un gradient d'où le nom descente du gradient (gradient descent) pour l'algorithme d'optimisation des poids[55]. Il comprend les étapes suivantes :

- Tirer un lot de données d'entrainement Xi et les prédictions correctes Yi correspondantes.
- Exécuter l'algorithme du feedforward pour obtenir les Yi' prédits.
- Calculer la différence/perte entre les sorties attendues et obtenues en utilisant la fonction de perte (MSE ou autre).
- Calculer le gradient (la dérivée) de la fonction de perte avec comme valeurs des inconnues, les paramètres du réseau qui ont généré les résultats obtenus.
- Modifier les poids des connexions du réseau dans la direction réduisant le gradient comme illustré dans l'image ci-dessous.
- Faire passer le prochain lot de données et réexécuter les étapes 1–5 jusqu'à obtenir des prédictions satisfaisantes.

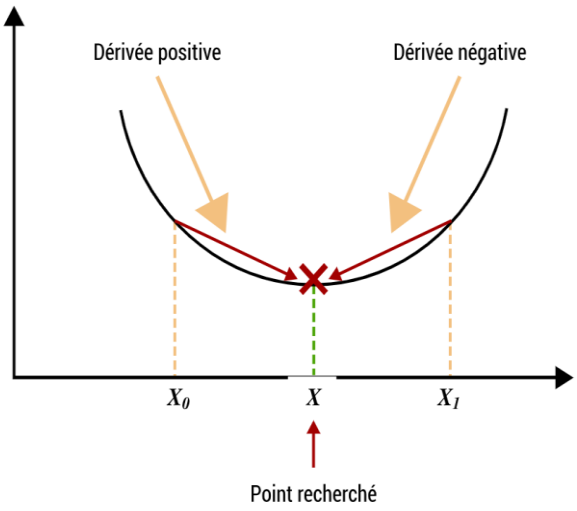


Figure 17 : Opération de rétropropagation [55].

2.2.2 Les Réseaux de Neurones Récurrents (RNN)

Souvent utilisés dans le Traitement Automatique du Langage Naturel (NLP ou TALN) par la Combinaison de modélisation fondée sur des règles du langage humain et des modèles statistiques de l'apprentissage automatique et dans le but de transformer des données de textes brutes dans un format facilement compris et analysé par les algorithmes d'apprentissage automatique.

Les réseaux de neurones récurrents (RNN) ont la capacité, en théorie, de faire face à ces dépendances temporelles grâce à la mémoire à court terme mise en œuvre par leurs connexions récurrentes (feedback). Les réseaux de neurones récurrents sont une classe de modèles connexionnistes qui possèdent un état interne ou une mémoire à court terme en raison de connexions de rétroaction récurrentes, ce qui les rend adaptés pour traiter problèmes séquentiels, tels que la classification, la prédiction et la génération de la parole. Cependant, en pratique, il est difficile de les entraîner avec succès lorsque la mémoire à long terme est requise[56]. Pour traiter ce type de données, le LSTM et le GRU ont été mis en place. La mémoire à long terme (LSTM) et l'unité récurrente fermée (GRU). Elles apportent la solution au problème de disparition du gradient ainsi que pour capturer les dépendances à long terme.

Les RNN simples entraînés avec une descente de gradient stochastique ont des difficultés à apprendre les dépendances à long terme (à cause du phénomène de la disparition du gradient ou plus rarement de l'explosion du gradient) codées dans les séquences d'entrée en raison de la disparition du gradient empêchant les mises à jour de poids négligeables et ralentissant, voire même, l'apprentissage du modèle.

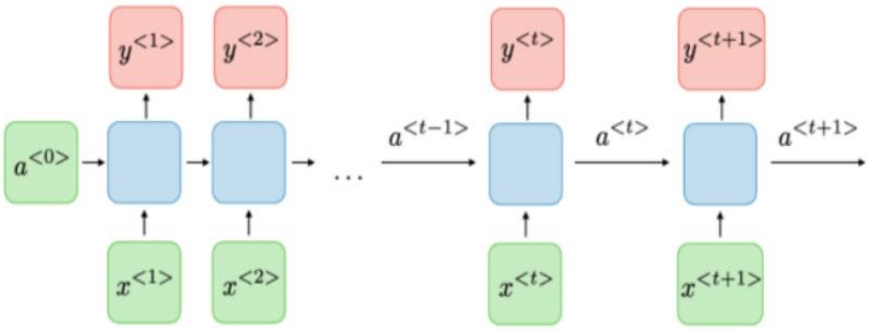


Figure 18 : Architecture traditionnelle des RNN [57].

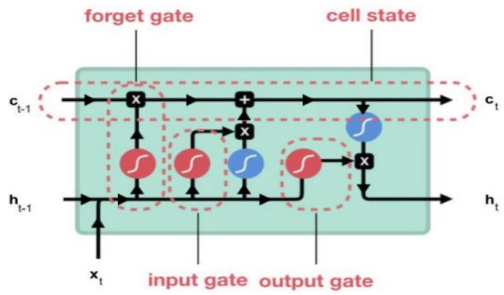
Dans les RNN, les sorties dépendent des entrées et des sorties passées (en mémoire).

Pour chaque pas de tempst *t*, l'activation *un*< *t* > et la sortie <t> s'expriment comme suit :

$$un^{<t>} = g_1(L_{un\,un}un^{<t-1>} + L_{un\,x}x^{<t>} + b_{un}) \quad \text{et} \quad et^{<t>} = g_2(L_{et\,un}un^{<t>} + b_{et})$$

Où *Lun x*, *Lun un*, *Let un*, *bun*, *bet* sont des coefficients qui sont partagés temporellement et *g*1 et *g*2 des fonctions d'activation [57].

- Forget gate (porte d'oubli)
- Input gate (porte d'entrée)
- Output gate (porte de sortie)
- Hidden state (état caché)
- Cell state (état de la cellule)



Cellule LSTM (crédit : image modifiée de Michaël Nguyen)

Figure 19 : Architecture des LSTM [58].

Dans la figure 19, Les LSTM utilisent trois portes : une porte d'entrée, une porte de sortie et une porte d'oubli. Deux états : état de la cellule Ct et état caché ht. La cellule de mémoire Ct conserve des informations sur de longues périodes de temps (mémoire tampon sur une longue durée). La porte d'entrée (Input Gate) it : Détermine quelles nouvelles informations doivent être ajoutées à l'état de cellule. La porte d'oubli (Forget Gate) ft : Détermine quelles informations de l'état de cellule précédent Ct-1 doivent être oubliées. La porte de sortie (Output Gate) Ot : Détermine quelles informations de l'état de cellule doivent être sorties en tant que nouvel état caché ht.

L'architecture des RNN, se compose généralement de plusieurs couches répétitives qui sont empilées les unes sur les autres. Chaque couche récurrente peut être un simple RNN, LSTM ou un GRU. L'architecture d'un RNN est constituée de :

- ❖ Entrée (Input) : Les données séquentielles sont introduites dans le réseau par l'intermédiaire de l'entrée. Chaque séquence de données est représentée par une série d'éléments (par exemple, des mots dans une phrase ou des instants temporels dans une série temporelle).
- ❖ Couche récurrente (Récurrent Layer) : Les informations séquentielles sont traitées dans la couche récurrente. Cette couche possède des connexions de rétroaction qui permettent aux informations de circuler d'une étape à l'autre dans la séquence. Cela permet au réseau de capturer les dépendances temporelles et de modéliser les relations complexes entre les éléments séquentiels.
- ❖ Optionnel : Stacking de couches récurrentes (Stacking Récurrent Layers) : Il est possible d'empiler plusieurs couches récurrentes les unes sur les autres pour former un réseau de neurones récurrents profond. Chaque couche récurrente traite les informations provenant de la couche précédente, permettant ainsi au réseau de capturer des niveaux de représentation plus abstraits et complexes.
- ❖ Couche de sortie (Output Layer) : La couche de sortie génère les prédictions ou les sorties souhaitées en fonction des informations traitées par les couches récurrentes. La nature de la tâche détermine le type de couche de sortie utilisée. Par exemple, pour la classification, une couche de sortie dense avec une fonction d'activation appropriée peut être utilisé[59].

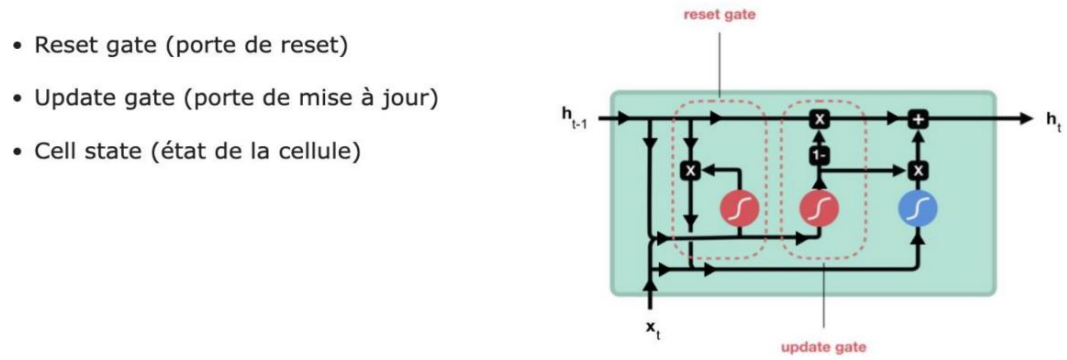


Figure 20 : Architecture des GRU [58].

Dans la figure 20, les GRU utilisent deux portes : une porte de mise à jour et une porte de réinitialisation et un état ht.

2.2.3 Les Réseaux Antagonistes Génératifs (GAN)

Un réseau antagoniste génératif (GAN) est une architecture de deep learning. Il entraîne deux réseaux neuronaux à se faire concurrence afin de générer de nouvelles données plus authentiques à partir d'un jeu de données d'entraînement donné. Par exemple, vous pouvez générer de nouvelles images à partir d'une base de données d'images existante ou de la musique originale à partir d'une base de données de chansons. Un GAN est appelé antagoniste parce qu'il entraîne deux réseaux différents et les oppose l'un à l'autre [60]. Un réseau génère de nouvelles données en prélevant un échantillon de données d'entrée et en

le modifiant autant que possible. L'autre réseau essaie de prédire si les données générées appartiennent au jeu de données d'origine. En d'autres termes, le réseau de prédiction détermine si les données générées sont fausses ou réelles. Le système génère des versions plus récentes et améliorées de fausses valeurs de données jusqu'à ce que le réseau de prédiction ne puisse plus distinguer les fausses de l'original.

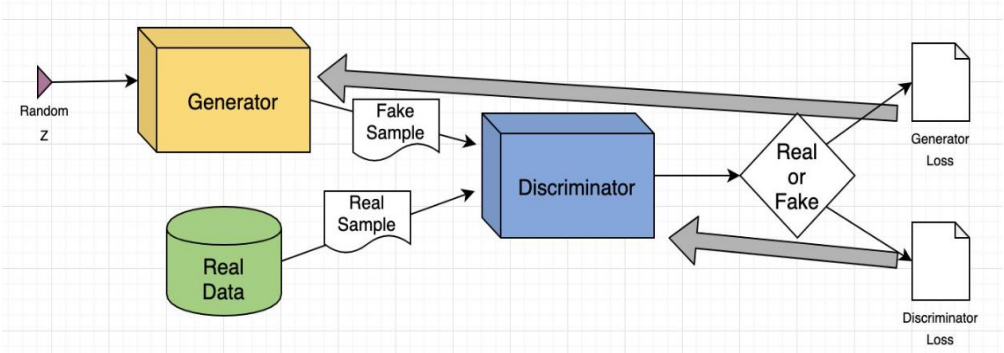


Figure 21 : Architecture d'un GAN [60].

Les GAN (Generative Adversarial Networks) fonctionnent selon des principes dérivés des réseaux neuronaux, les données de l'ensemble d'entrainement sont utilisées comme entrée afin de produire de nouvelles données d'entraînement similaires aux précédentes. Le fonctionnement d'un GAN peut être décrit étape par étape comme suit[61]:

1. Un réseau discriminatif crée un générateur qui produit de nouvelles données sur la base de la distribution des données existantes.
2. La capacité du discriminateur s'améliore grâce aux entrainements pour distinguer les candidats synthétisés des candidats réels. Ainsi le générateur pourra mieux tromper le discriminateur.
3. Les entrainements se font de manière répétitive jusqu'à atteindre le niveau de précision souhaitée.
4. La rétropropagation est utilisée pour mettre à jour à la fois le discriminateur et le générateur : le premier améliore sa capacité à identifier les images réelles, tandis que le second devient plus performant dans la production d'images synthétiques réalistes.
5. Les réseaux de neurones convolutifs (CNN) sont couramment utilisés comme discriminateurs, tandis que les réseaux de neurones déconvultionnels sont employés comme générateurs.

2.2.4 Les Autoencodeurs

Un autoencodeur est un type spécifique de réseau neuronal, principalement conçu pour encoder l'entrée dans une représentation compressée et significative, puis la décoder de manière à ce que l'entrée reconstruite soit aussi similaire que possible à l'entrée d'origine[62]. Son objectif fondamental est d'apprendre une représentation significative des données de manière non supervisée.

Un autoencodeur est un réseau de neurones à propagation avant, dans lequel l'entrée est identique à la sortie [63]. Pour ce faire la donnée(information) est encodée à l'entrée dans la

couche cachée, puis la décoder en vue de reconstruire au plus près l'information initiale. L'intérêt d'un autoencodeur n'est pas dans la restitution exacte de l'information à la sortie, mais dans l'effet induit d'apprendre une représentation condensée et représentative d'un ensemble de données (exemples) d'un espace à grande dimensions, vers un espace à petite dimensions (appelé espace des variables latentes) [63]. L'autoencodeur variationnel a l'avantage supplémentaire d'assurer une bonne correspondance entre une petite variation au niveau de l'espace latent et une petite variation au niveau de l'espace des données.

Il se compose d'une couche interne cachée qui définit un code représentant les données d'entrée. Un auto-codeur comme le montre la gure1.16 est composé de trois parties : l'encodeur, le code (les variables latentes) et le décodeur [64]. L'encodeur compresse l'entrée et génère le code, que le décodeur utilise ensuite pour recréer l'entrée Les modèles d'autoencodeur trouvent une utilité dans diverses tâches d'apprentissage non supervisées, telles que la modélisation générative la réduction de la dimensionnalité, l'extraction de caractéristiques, la détection d'anomalies ou de valeurs aberrantes et la débruitisation.

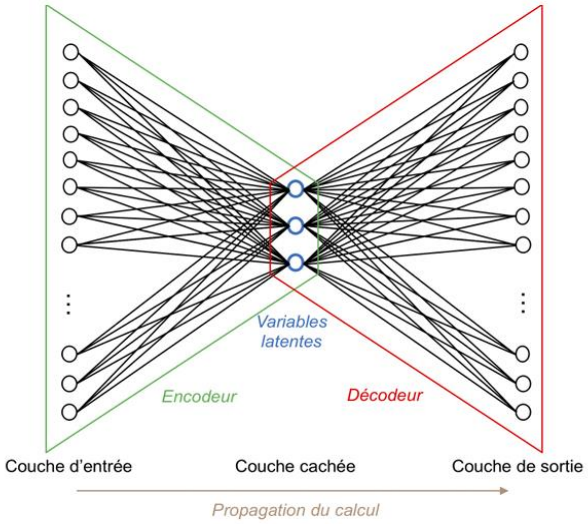


Figure 22 : Architecture de base d'un autoencodeur [63].

2.2.5 Les Transformers

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 : recoit une entrée et construit une représentation de ses caractéristiques. Cela signifie que le modèle est optimisé pour acquérir une compréhension à partir de l'entrée.

- Décodeur : le décodeur combine les caractéristiques extraites 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 indépendamment 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[65].

Les transformateurs 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., 2017 [66] comme un nouveau bloc de construction basé sur l'attention pour la traduction automatique. L'encodeur est composé d'un empilement de canapés identiques, chacun comprenant deux sous-couches : un mécanisme d'auto-attention multi-tête constitue la première sous-couche ; une couche entièrement connectée positionnelle constitue la deuxième sous-couche.

Les couches d'auto-attention qui, de manière similaire aux réseaux neuronaux non locaux [67] analysent chaque élément d'une séquence et la mettent à jour en agrégeant les informations de l'ensemble de la séquence. L'un des principaux avantages des modèles basés sur l'attention est leurs calculs globaux et leur mémoire parfaite, ce qui les rend plus adaptés que les RNN sur de longues séquences. Les transformateurs remplacent désormais les RNN dans de nombreux problèmes de traitement du langage naturel, de traitement de la parole et de vision par ordinateur.

L'apprentissage RNN prend une très longue période car il est séquentiel et itératif. L'apprentissage du transformateur, en revanche, est parallèle et permet d'apprendre toutes les fonctionnalités simultanément, améliorant considérablement l'efficacité du calcul et réduisant le temps nécessaire à l'apprentissage du modèle.

La conception globale du transformateur de l'encodeur et du décodeur est illustrée dans la figure suivante.

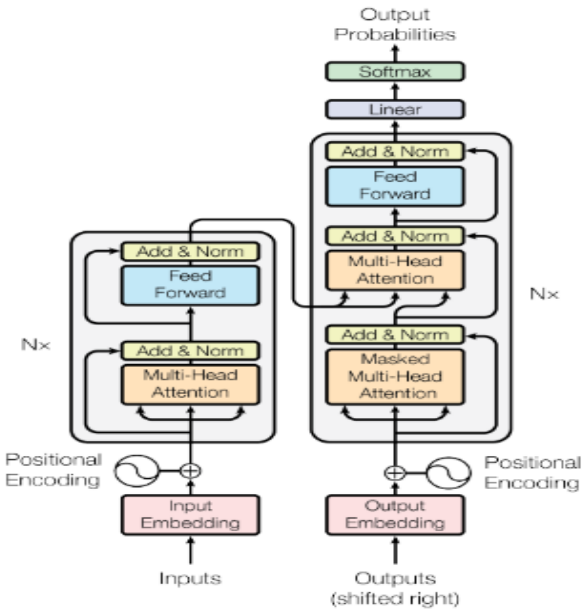


Figure 23 : Architecture du modèle Transformers [66].

En 2020, Dosovitskiy et al. ont introduit une architecture de transformateur, Vision Transformer (ViT), spécialement conçue pour l'analyse d'images, qui a montré des résultats compétitifs[68].

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 ordres. La figure suivante décrit l'architecture d'un classificateur d'images à base de transformer avec encodeur, appelé ViT (Vision Transformer)[65].

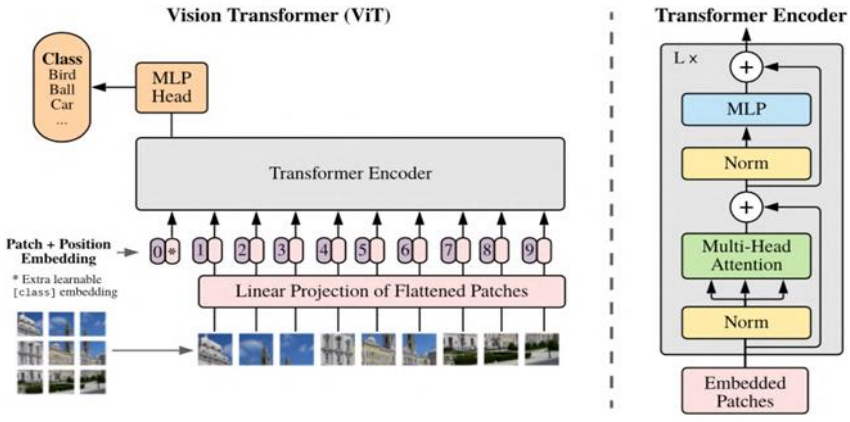


Figure 24 : Architecture d'un transformer ViT [65].

2.3 Transfert Learning

L'apprentissage par transfert désigne l'ensemble des méthodes qui permettent de transférer les connaissances acquises en résolvant des problèmes spécifiques pour résoudre un autre problème connexe. Il repose sur la réutilisation d'un modèle pré-entraîné sur une grande quantité de données pour résoudre une tache différente. Plutôt que de créer un nouveau modèle à partir de zéro et de l'entraîner sur un ensemble de données de taille limitée, on peut tirer parti des connaissances déjà acquises par le modèle pré-entraîné en l'adaptant à notre tâche spécifique. En utilisant des modèles pré-entraînés comme point de départ, l'apprentissage par transfert permet le développement rapide de modèles performants et la résolution efficace de problèmes complexes dans différents domaines en général et en traitement d'images en particulier [69].

Ainsi, la stratégie que vous utiliserez pour faciliter le TL dépendra du domaine du modèle que vous créez, de la tâche à accomplir et de la disponibilité des données d'entraînement. Le feature extraction et le fine-tuning sont les techniques les plus utilisées pour exploiter les connaissances d'un modèle pré-entraîné.

- Feature extraction : utilise les représentations du réseau pré-entraîné pour extraire des caractéristiques significatives pour une nouvelle tâche. Un nouveau classificateur est ajouté et entraîné à partir de zéro, en réutilisant les caractéristiques apprises. Les paramètres du modèle pré-entraîné restent gés pendant l'entraînement.
- Fine-tuning : lors du fine-tuning, les paramètres du modèle pré-entraîné sont affinés sur le jeu de données de la tâche cible, ainsi que les couches supplémentaires. Les couches supérieures (spécifiques à la tâche) sont généralement réentraînées, tandis que les couches inférieures (plus générales) peuvent être fixées ou ajustées légèrement.

Rendre les poids de l'ancien modèle entraînables peut aider à spécialiser le nouveau modèle pour le nouveau jeu de donnés disponible. Cependant, si l'ancien modèle est volumineux, le nombre élevé de poids entraînera souvent un surajustement (Overfitting) si le nouvel ensemble de données est trop petit.

2.4 Les Concepts de base de la détection d'objets

La détection d'objets combine deux processus clés : la classification et la localisation. La classification permet d'identifier les objets, tandis que la localisation précise l'emplacement de ces objets dans l'image, généralement en dessinant une boîte englobante autour d'eux. Pour ce faire, on utilise généralement des algorithmes sophistiqués, souvent basés sur des réseaux neuronaux, qui apprennent à reconnaître les modèles et les traits qui caractérisent les différents objets. Comme mentionné précédemment, les méthodes de détection d'objets sont

communément divisées en deux catégories, les méthodes en deux phases (ou étape) et les méthodes en une phase.

Les méthodes en une étape privilégient la vitesse d'inférence et celles en deux étapes privilégient la précision de la détection. Les méthodes en deux phases utilisent une première phase de propositions d'objets candidats, et une seconde phase de raffinement où les meilleures propositions sont choisies ainsi que mieux positionnées et classifiées [70], ces méthodes sont certes très précises mais aussi fastidieuses et lentes. Les méthodes en une phase quant à elle vont classifier et positionner les objets parmi toutes les possibilités directement sans utiliser de candidats.

Les approches convolutives traditionnelles à deux étapes (two-stage detectors, les exemples de modèles incluent Faster R-CNN, Mask R-CNN et Cascade R-CNN, SPP-net, FPN) extraient les régions les plus susceptibles de contenir un objet (régions d'intérêt lors d'une première étape. Ensuite, la seconde étape analyse ces régions et prédit à la fois la probabilité de présence d'un objet ainsi que sa classe d'appartenance. Des équipes ont alors développé des approches dites "en une étape" (one stage detectors, telles que SSD, YOLO et RetinaNet), qui s'avèrent être plus rapides et plus adaptées pour la détection en temps réel. Ces modèles analysent l'image, localisent et identifient les objets en une seule passe.

2.4.1 Architecture d'un détecteur d'objet

L'architecture typique d'un détecteur d'objet comprend trois parties : l'épine dorsale (backbone), le cou (neck) et la tête de détection (detection head). Le processus de base de la détection d'objet est composé de quatre étapes principales (figure 25) :

- Regional proposal : l'étape qui prend le plus du temps. Il s'agit de déterminer les localisations potentielles de l'objet (ou régions d'intérêts, en anglais appelées region proposals or Regions Of Interest ROI) dans l'image. Elle est effectuée dans l'épine dorsale.
- Feature extraction : les cartes de caractéristiques (appelées parfois cartes d'activations) générées sont ensuite introduites dans le cou du détecteur, responsable de l'extraction des caractéristiques pertinentes et de leurs fusions à multi-échelle.
- Regional Classification : dans la tête de détection on passe 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 régions 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 25 : Architecture de base d'un détecteur d'objet [38].



Figure 26 : Processus de détection d'objet.

2.4.2 Extracteurs de caractéristiques

Les images numériques sont de nature complexe et présentent des informations de haut niveau, telles que des objets, des scènes et des modèles. Dans le but de pouvoir analyser interprétés par exemple les objets contenus dans ces images, il est nécessaire de pouvoir extraire les caractéristiques de ces objets-là. Cependant, extraire des informations de haut niveau à partir de données d'image peut s'avérer difficile en raison des variations de luminosité, de pose, de fouillis d'arrière-plan, etc. [68].

L'extraction de caractéristiques est une étape cruciale pour la détection d'objet. En effet, le choix de l'architecture utilisée a un impact sur la performance du modèle (vitesse d'inférence et précision). Les réseaux de neurones profonds apprennent et transmettent des caractéristiques complexes à plusieurs niveaux à partir de données d'entrée. Cependant, plus la profondeur du réseau s'accentue, une saturation puis une dégradation de la précision se produisent[71].

Avec les méthodes en une deux passes comme par exemple Mask R CNN, l'architecture dorsale convolutionnelle transforme l'image brute en une représentation riche de ses caractéristiques visuelles en extrayant les caractéristiques pertinentes de l'image d'entrée. Il est composé généralement de plusieurs couches convolutives, d'opérations de regroupement et d'autres opérations non linéaires qui permettent de capturer des informations du niveau bas au niveau élevé dans l'image [72]. Et cette extraction est réalisée dans la première phase, et

c'est ainsi qu'on obtient les régions susceptibles de contenir un objet. Ce sont les régions d'intérêt ou (en anglais Region of Interest (RoI)). Après, ces RoI sont extraites de la carte des caractéristiques, généralement à l'aide d'une couche de regroupement de RoI. Enfin, toutes les régions extraites sont régressées et classées pour obtenir les objets prédits.

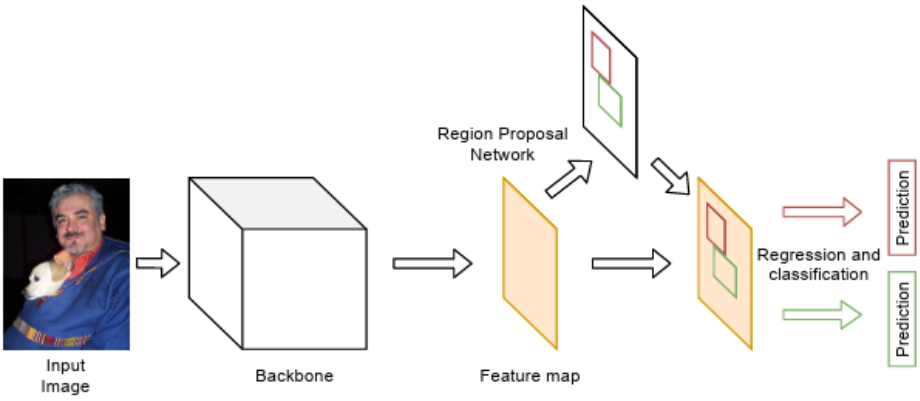


Figure 27 : Extraction avec les méthodes à deux étapes [73].

Dans les méthodes en une passe comme YOLO, à l'opposé des détecteurs à deux étapes, plutôt que d'utiliser les boîtes prédites pour extraire des caractéristiques pour une deuxième étape de traitement, la classe est directement associée. YOLO génère plusieurs boites de prédictions pour un même objet (à cause des grilles et des ancres utilisées pour couvrir l'image), ce qui fait qu'on a plusieurs boites prédites qui se chevauchent. Afin de filtrer les boîtes qui correspondent au même objet, on utilise généralement la suppression non maximale (NMS). Dans un NMS tout d'abord, un réseau de détection génère une liste de boîtes prédites avec une liste de scores de confiance associés, (c'est-à-dire un score de confiance que le réseau a d'une boîte comme étant une détection réelle, elle est mesurée par l'IoU (Intersection over Union)) entre la boîte sélectionnée et une autre boîte.). L'IoU mesure le chevauchement entre la boîte englobante prédite et la boîte de vérité fondamentale. Un seuil IoU (généralement défini entre 0,5 et 0,95) est utilisé pour déterminer si une prédiction est considérée comme un vrai positif. Ensuite, les étapes suivantes sont répétées jusqu'à ce qu'il ne reste qu'une seule boîte dans la liste des boites prédites :

- La boîte avec le score de confiance le plus élevé est sélectionnée parmi les boites prédites,
- Cette boîte est comparée à toutes les boîtes restantes dans la liste des boites prédites,
- Si une boîte restante est trop proche d'une boîte parmi les boites prédites, alors cette boîte est considérée comme une détection de doublon et est donc supprimée des boites prédites.
- La boîte sélectionnée est ajoutée à la liste des boîtes retenues.

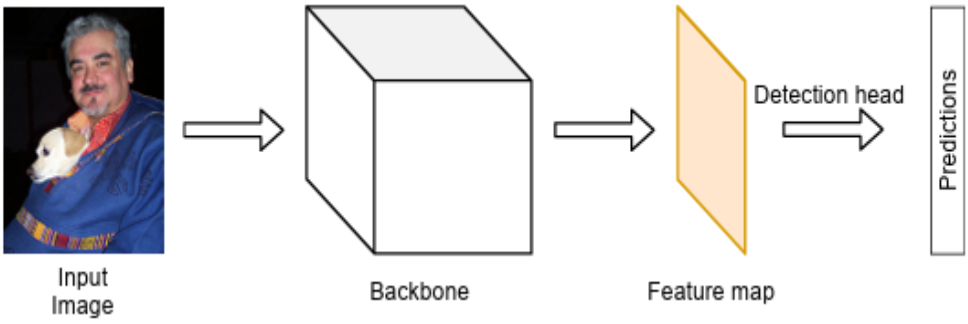


Figure 28 : Extraction avec les méthodes à une étape [73].

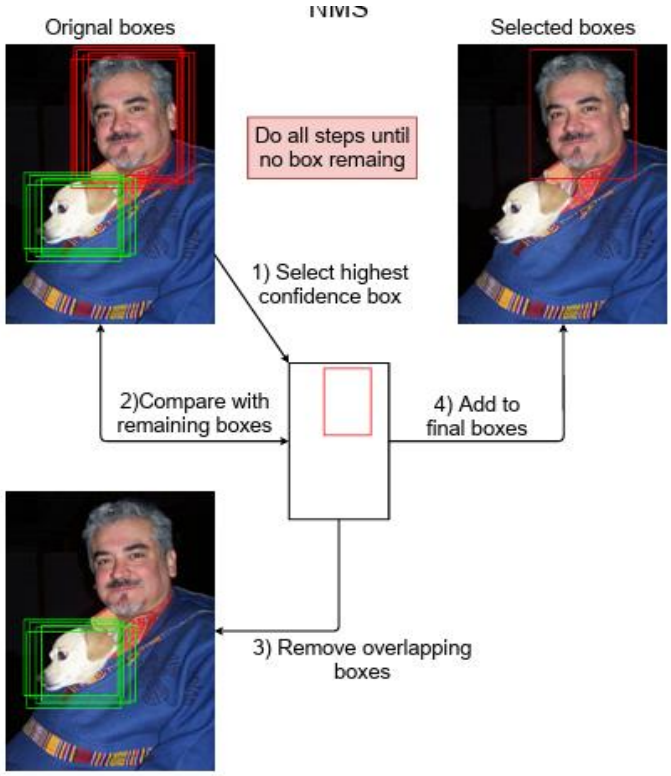


Figure 29 : Processus du NMS [73].

2.4.3 Les mesures de performance d'un modèle de détection

En 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. Elle repose sur le calcul de la précision et du rappel, qui sont respectivement définis comme le nombre d'éléments vrais positifs sur le total des éléments positifs et le nombre d'éléments vrais positifs sur le total des éléments à prédire (figure 30) [73]. Le mAP est alors défini comme la valeur moyenne de la précision moyenne (AP) discrète de chaque classe à détecter, où l'AP discret est défini comme l'aire sous la courbe de Precision-Recall (figure 31). 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.

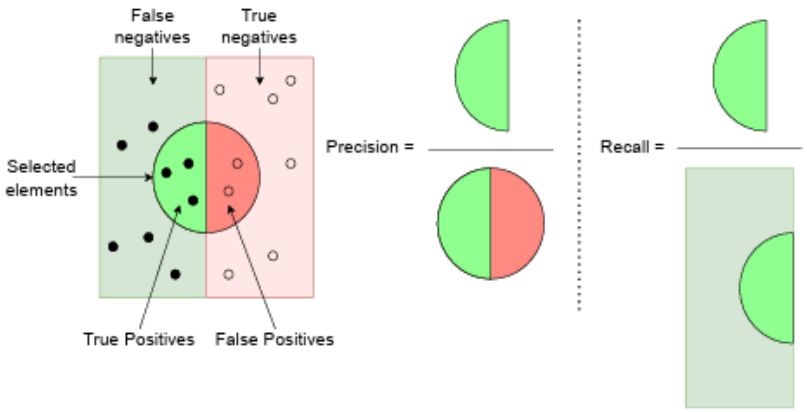


Figure 30 : Illustration des calculs de précision et de rappel [74].

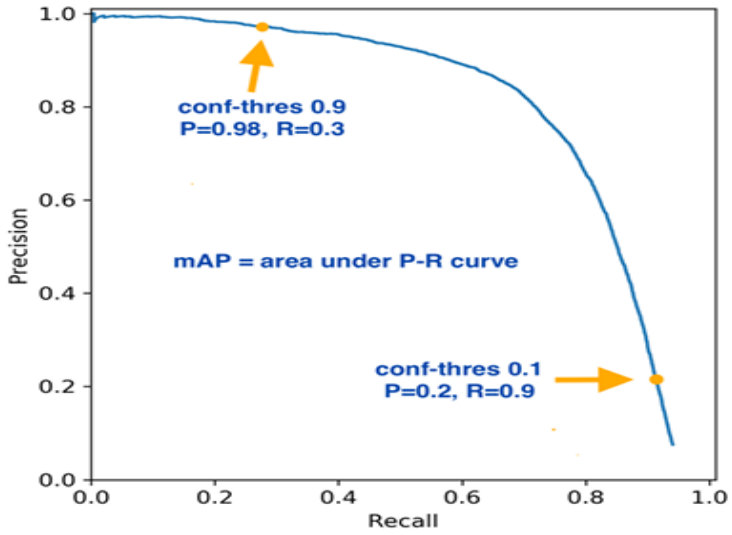


Figure 31 : Représentation de la courbe P-R [75].

2.5 Conclusion

En tant que branche la plus importante du ML, l'apprentissage profond (DL) s'est développé rapidement ces dernières années et est progressivement devenu un domaine très prisé de recherche en ML. Cette progression a fait que la communauté de vision par ordinateur a amélioré les résultats en détection d'objets. La détection d'objet joue un rôle important dans beaucoup de domaines notamment celui de la gestion des déchets. Dans ce chapitre nous avons abordé les différentes approches des réseaux neuronaux, leur architecture, et leur fonctionnement. Ensuite nous avons abordé les concepts de base de la détection des objets et leur principes fondamentaux. Ainsi nous avons posé les bases pour entrer dans le vif de notre sujet qu'est la détection des déchets au chapitre suivant.

CHAPITRE 3 : *Méthodologie et Implémentation*

3.1 Introduction

Ce chapitre représente le corps du notre travail de recherche. Dans les deux chapitres précédents nous avons vu les principes d'apprentissage automatique en général et d'apprentissage profond en particulier, puis mentionné les deux techniques principales de détection d'objets. Sur la base de ces deux techniques nous avons choisi le modèle de détection en une passe YOLO. De plus nous utiliserons aussi une des méthodes de détection de bout en bout avec les Transformers (detr). Ainsi pour répondre à notre problématique de détection et de classification des types de déchets pour optimiser leur gestion, nous commencerons par décrire les étapes de conception, puis nous présenterons le dataset avec tout le prétraitement nécessaire et enfin nous présenterons en détails toutes les méthodologies utilisées, leurs performances et discuterons des résultats.

3.2 Outils utilisés

3.2.1 Langage utilisé

Dans l'implémentation nous avons utilisés le langage python. Python est un langage de programmation généraliste interprété de haut niveau. Il est l'un des langages de programmation les plus populaires pour la science des données, le machine learning, le deep learning et open source. Sa simplicité, sa cohérence, et son indépendance vis-à-vis des systèmes d'exploitation rendent l'apprentissage et le développement dans ces domaines beaucoup plus accessibles. La richesse des bibliothèques et frameworks disponibles accélère la mise en œuvre des projets, permettant aux développeurs d'explorer, d'innover et de déployer des solutions efficaces [76]. Dans notre implémentation nous avons utilisé la version 3.12 de python.

3.2.2 Environnement d'implémentation et Bibliothèques utilisés

L'implémentation a été sur le cloud de google (Google Colab) pour bénéficier des unités de GPU qu'offrent cette plateforme. Colab est un services hébergé de Jupyter Notebooks, facile d'utilisation et ne nécessite aucune configuration. Les étapes d'implémentation sont réalisées à l'aide des Principales bibliothèques telles que : Tensorflow, Keras, PyTorch, Matplotlib, Numpy.

- **Tensorflow**

  TensorFlow est une plateforme open source de bout en bout pour l'apprentissage automatique. Il dispose d'un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires qui permet aux chercheurs de pousser l'état de l'art

en ML et aux développeurs de facilement construire et déployer des applications alimentées par ML. Techniquement il repose sur plusieurs concepts clés qui rendent possible les entrainements des modèles et les rend flexibles. Mais se concentre particulièrement sur la formation et l'inférence des réseaux de neurones profonds[77].

- **Keras**

  Keras est une API d'apprentissage profond écrite en Python. L'API Keras est d'ailleurs packagée avec TensorFlow sous la forme tf.keras. Depuis la version 2.0, il s'agit de la principale API TensorFlow [78]. Keras est l'une des APIs de réseaux de neurones les plus utilisées pour le développement et le test de réseaux de neurones. Elle permet de créer très facilement des couches neuronales ou de mettre en place des architectures complexes.

- **PyTorch**

  PyTorch est une bibliothèque populaire pour l'apprentissage profond qui fournit des graphes de calcul dynamiques [79]. Elle offre flexibilité et facilité d'utilisation dans la création et la formation de réseaux neuronaux. Elle permet l'apprentissage de modèle profond, offre une option pour gérer des datasets, ainsi que des métriques pour mesurer la performance des modèles.

- **Matlplotlib**

  Matplotlib est une bibliothèque de visualisation de données en Python qui permet de créer une grande variété de graphiques, tels que des graphiques linéaires, des graphiques à barres, des graphiques circulaires, des diagrammes de dispersion, des histogrammes, des cartes, des diagrammes en boîte, etc. Elle offre également des fonctionnalités avancées pour la personnalisation des graphiques, la gestion des axes, des légendes, des titres, des couleurs, des annotations, etc [80].

- **Numpy**

  NumPy est une bibliothèque pour le langage de programmation Python, ajoutant la prise en charge de grands tableaux et matrices multidimensionnels, ainsi qu'une grande collection de fonctions mathématiques de haut niveau pour opérer sur ces tableaux.

3.3 Dataset utilisé

D.Bashkirova et al, ont proposé le premier jeu de données ZeroWaste-f entièrement annoté pour la détection d'objets de déchets industriels. Le jeu de données ZeroWaste-f présente un problème complexe de vision par ordinateur en conditions réelles, consistant à détecter des objets hautement déformables dans des scènes extrêmement encombrées [81].

Les images ont été prises dans une installation de récupération des matériaux (MRF) qui sont au cœur du processus de gestion des déchets. Ces installations sont les endroits où les déchets recyclables collectés sont triés en balles séparées de plastique, de papier, de métal et de verre

(et d'autres sous-catégories). ZeroWaste est un ensemble de données fondamentalement différent des repères de détection et de segmentation populaires : niveau élevé d'encombrement, présence d'objets hautement déformables et translucides, ainsi qu'une différence fine entre les classes d'objets. Tous ces aspects posent un défi unique pour la vision automatisée. De plus, en raison de la nature en constante évolution du flux, le contenu et les qualités visuelles du flux sont souvent spécifiques au MRF et dépendent fortement de la saison, par conséquent l'algorithme de détection doit être efficace en termes d'étiquetage et capable d'apprendre et de s'adapter aux changements du flux avec seulement quelques exemples étiquetés. Nous avons annoté les objets amovibles de quatre types de matériaux comme premier plan : carton (y compris les colis, les boîtes de céréales et autres emballages alimentaires en carton), plastique souple (par exemple les sacs en plastique, les emballages), plastique rigide (par exemple les récipients alimentaires, les bouteilles en plastique) et métal (par exemple les canettes en métal). L'arrière-plan comprend le tapis roulant et les objets en papier. Un encombrement et des occlusions importantes, une grande variabilité des formes et des textures des objets du premier plan, ainsi que de graves déformations d'objets habituellement absents des autres jeux de données de segmentation, rendent ce domaine très difficile pour la détection d'objets [81].



Figure 32 : Représentation de nos données à gauche et à droite les vérités terrains correspondantes [81].

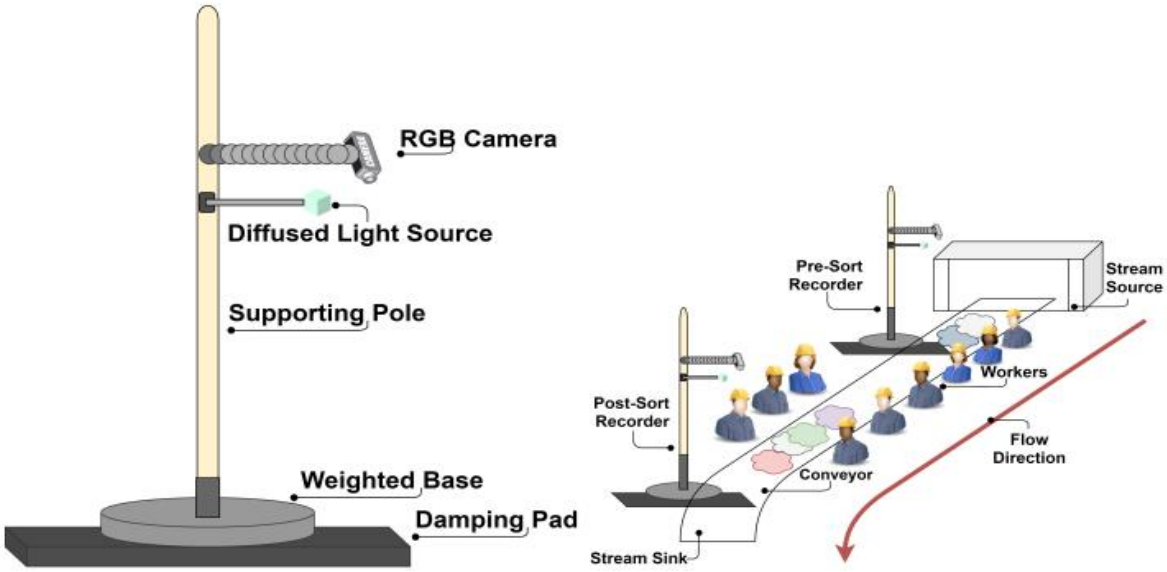


Figure 33 : A gauche : Assemblage de chaque appareil d'enregistrement. A droite : Disposition de l'installation d'enregistrement dans l'environnement de recyclage [81].

3.4 Méthodologies proposées

Dans le cadre de ce mémoire, nous proposons les modèles YOLO11 et RT-DETR, en utilisant la méthode d'apprentissage par transfert, qui est une technique précieuse en apprentissage automatique, permettant de réutiliser des modèles existants, initialement formés pour une tâche spécifique, afin de résoudre un problème différent. Cette approche présente de nombreux avantages en économisant le temps et les ressources nécessaires à la construction de nouveaux modèles neuronaux à partir de zéro.

3.4.1 Modèle yolov11

YOLO (You Only Live Once) en français « Tu ne regardes qu'une seule fois », est l'un des algorithmes populaires de détection d'objets utilisé par les chercheurs du monde entier. Il a été décrit pour la première fois en 2015 dans l'article de Joseph Redmon et al [82].

Le réseau utilise les caractéristiques de l'image entière pour prédire chaque boîte englobante. Il prédit également toutes les boîtes englobantes de toutes les classes d'une image simultanément. Cela signifie que ce réseau raisonne globalement sur l'ensemble de l'image et sur tous les objets qu'elle contient. La conception YOLO permet un apprentissage de bout en bout et des vitesses en temps réel tout en maintenant une précision moyenne élevée[82].

L'architecture YOLO se compose essentiellement de trois composants fondamentaux. Tout d'abord, le **backbone** sert d'extracteur de caractéristiques principal, utilisant des réseaux neuronaux convolutionnels pour transformer les données d'image brutes en cartes de caractéristiques multi-échelles. Ensuite, le composant **du cou** agit comme une étape de traitement intermédiaire, utilisant des couches spécialisées pour agréger et améliorer les représentations de caractéristiques à différentes échelles. Troisièmement, le composant **de la tête** fonctionne comme mécanisme de prédiction, générant les résultats finaux pour la localisation et la classification des objets en fonction des cartes de caractéristiques raffinées [73].

En substance, YOLO divise l'image d'entrée en une grille *S* x *S*. Si l'objet se trouve au centre de la cellule de la grille, cette cellule doit détecter cet objet. Cela se fait en prédisant les boîtes englobantes *B* et les scores de confiance dans cette cellule de la grille. Chaque boîte englobante est définie par un tuple à cinq éléments *(x, y, h, w, confiance)*. Les coordonnées *(x, y)* sont les coordonnées du centre, tandis que *w* et *h* sont la largeur et la hauteur relatives[82]. La suite concernant le choix de la meilleure prédiction par le processus de suppression non maximale (SNM) est expliquée à la section **2.4.2**.

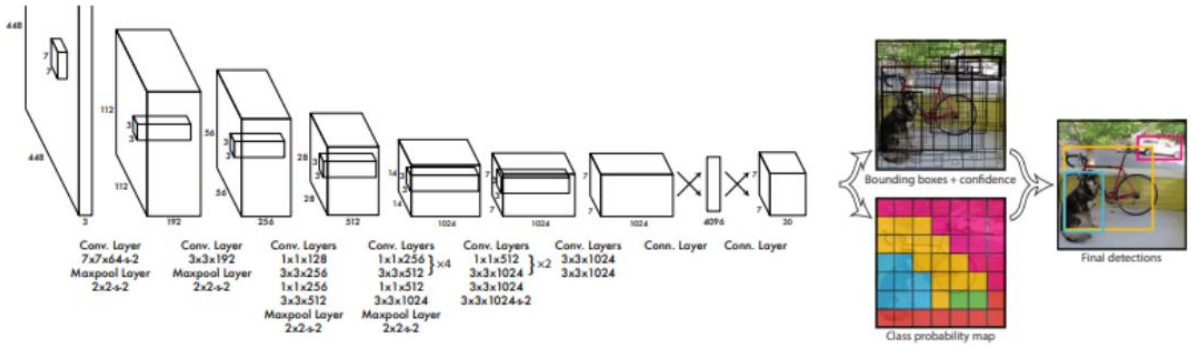


Figure 34 : Architecture générale des modèles YOLO [82].

Yolo11 est la dernière itération de la série des détecteurs d'objets en temps réel. Ultralytics yolo de détecteurs d'objets en temps réel, redéfinit une précision, une vitesse et une efficacité de pointe. S'appuyant sur les avancées impressionnantes des versions précédentes de yolo, yolo11 apporte des améliorations significatives à l'architecture et aux méthodes d'apprentissage, ce qui en fait un choix polyvalent pour un large éventail de tâches de vision par ordinateur [84].

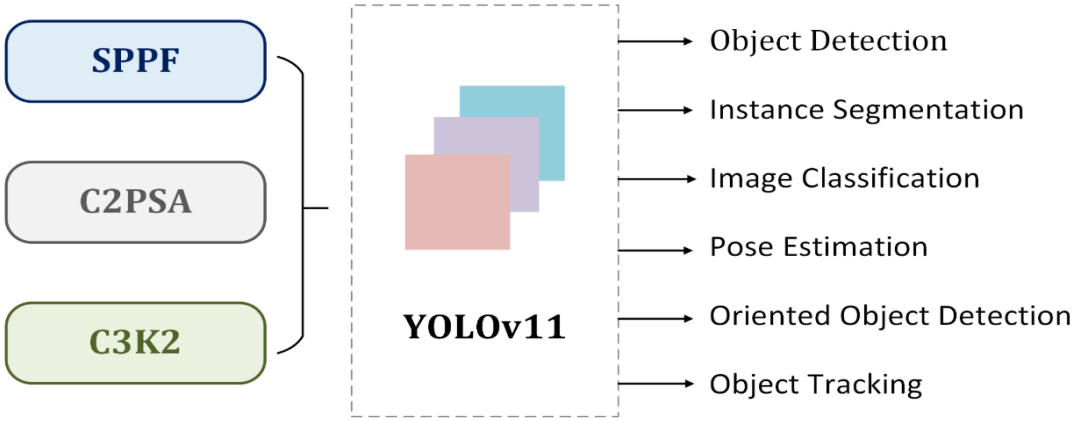


Figure 35 : Modules architecturaux clés de YOLO11 [85].

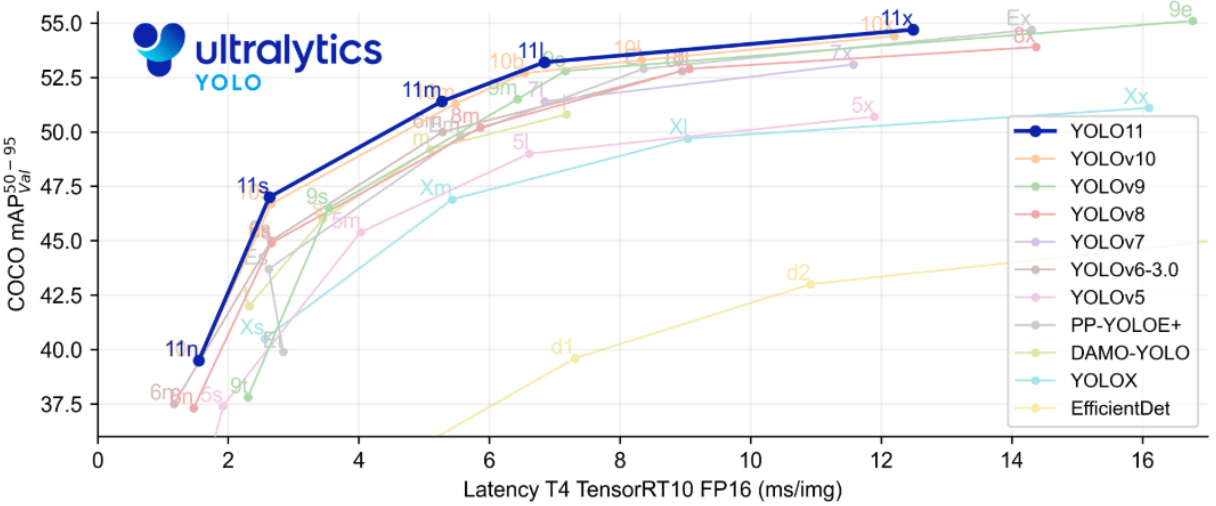


Figure 36 : Performance de yolo11 par rapport aux versions précédentes [84].

Les innovations architecturales des modèles, notamment l'introduction du bloc C3k2 (Cross Stage Partial avec taille de noyau 2), les composants SPPF (Spatial Pyramid Pooling - Fast) et C2PSA (Convolutional block with Parallel Spatial Attention), contribuent à améliorer les performances des modèles de plusieurs manières, comme l'extraction améliorée de fonctionnalités [85].

- Yolo11 conserve une structure similaire à ses prédécesseurs, utilisant des couches convolutionnelles initiales pour sous-échantillonner l'image. Ces couches constituent la base du processus d'extraction des caractéristiques, réduisant progressivement les dimensions spatiales tout en augmentant le nombre de canaux. Une amélioration significative de Yolo11 est l'introduction du bloc C3k2, qui remplace le bloc C2f utilisé dans les versions précédentes. Le bloc C3k2 est une implémentation plus efficace du point de vue du calcul du goulot d'étranglement partiel à étapes croisées (CSP). Il utilise deux convolutions plus petites au lieu d'une seule grande convolution. Le « k2 » dans C3k2 indique une taille de noyau plus petite, ce qui contribue à un traitement plus rapide tout en maintenant les performances. Il est introduit après le suréchantillonnage et la concaténation, ce qui se traduit par une vitesse et des performances améliorées [85]. le bloc C3K2 améliore la représentation des caractéristiques avec moins de paramètres par rapport aux blocs C2f de YOLOv8 [86].

- Yolo11 introduit un nouveau bloc Cross Stage Partial with Spatial Attention (C2PSA). Le bloc C2PSA est un ajout notable qui améliore l'attention spatiale dans les cartes de caractéristiques. Ce mécanisme d'attention spatiale permet au modèle de se concentrer plus efficacement sur les régions importantes de l'image. En regroupant les caractéristiques spatialement, le bloc C2PSA permet à YOLO11 de se concentrer sur des zones d'intérêt spécifiques, améliorant potentiellement la précision de détection pour des objets de plus petites tailles et de positions variées [85].

- La tête de détection de Yolo11 comprend plusieurs CBS (Convolution-BatchNorm-Silu) couches après les blocs C3k2. Ces couches affinent encore les cartes de

caractéristiques en : extrayant des fonctionnalités pertinentes pour une détection précise des objets ; stabilisant et normalisant le flux de données grâce à la normalisation par lots ; utilisant la fonction d'activation de l'unité linéaire sigmoïde (SiLU) pour la non-linéarité, ce qui améliore les performances du modèle [85].

3.4.2 Détection de bout en bout avec les Transformers (Detr)

L'architecture globale de DETR est étonnamment simple et illustrée dans la Figure 3.7. Elle contient trois composants principaux, que nous décrivons ci-dessous : une dorsale CNN pour extraire une représentation compacte des caractéristiques, un transformateur encodeur-décodeur et un simple réseau d'alimentation directe (FFN) qui fait la prédiction de détection finale. Contrairement à de nombreux détecteurs modernes, DETR peut être implémenté dans n'importe quel cadre d'apprentissage profond qui fournit une dorsale CNN commune et une implémentation d'architecture de transformateur avec seulement quelques centaines de lignes. **Encodeur du Transformer** : chaque couche d'encodeur a une architecture standard et se compose d'un module d'auto-attention multi-têtes et d'un réseau à propagation avant (FFN). **Décodeur du Transformer** : le décodeur suit l'architecture standard du transformateur, en utilisant des mécanismes d'auto-attention multi-têtes et d'encodeur-décodeur [87].

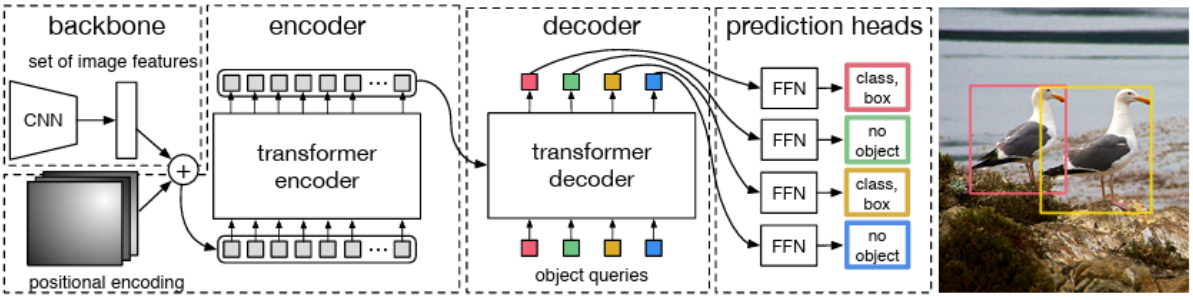


Figure 37 : Architecture globale de DETR [87].

Ici nous utilisons le modèle RT-DETR (Real-Time Détection Transformer) un détecteur d'objets en temps réel basé sur un transformateur de vision.

Real-Time Détection Transformer (RT-DETR), est un détecteur d'objets de bout en bout à la pointe de la technologie qui offre des performances en temps réel tout en conservant une grande précision. Il est basé sur l'idée de DETR, tout en introduisant une colonne vertébrale basée sur la convolution et un codeur hybride efficace pour gagner en vitesse en temps réel. RT-DETR traite efficacement les caractéristiques multi-échelles en découplant l'interaction intra-échelle et la fusion inter-échelle. Le modèle est hautement adaptable, permettant un ajustement flexible de la vitesse d'inférence en utilisant différentes couches de décodeur sans réentraînement [88]. RT-DETR traite les images pour identifier et localiser plusieurs objets à l'intérieur.

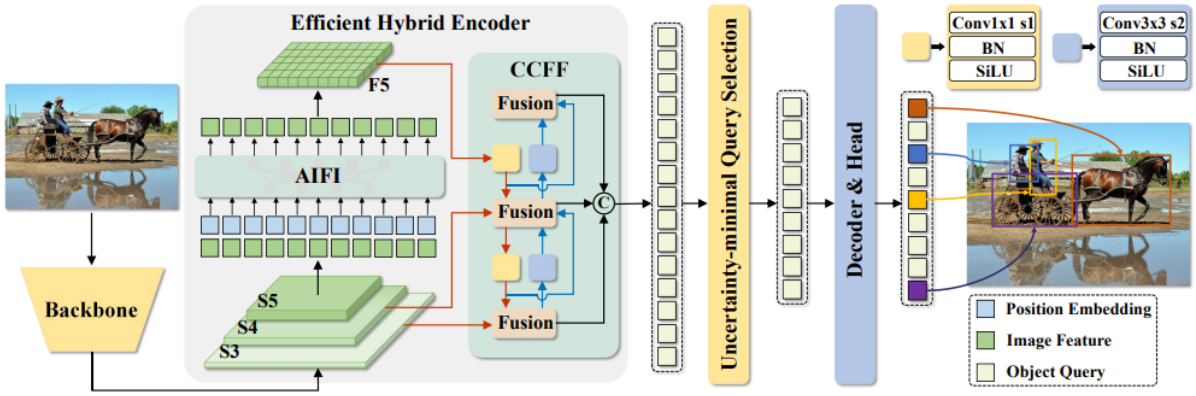


Figure 38 : Architecture du modèle RT-DETR [89].

Le diagramme d'architecture du modèle RT-DETR montre les trois dernières étapes de l'épine dorsale {S3, S4, S5} en tant qu'entrée du codeur. Le codeur hybride efficace transforme les caractéristiques multi-échelles en une séquence de caractéristiques d'image grâce à l'interaction des caractéristiques intrascales (AIFI) et au module de fusion des caractéristiques inter-échelles (CCFM). La sélection de requête sur la base du seuil de l'IoU (Intersection over Union) est employée pour sélectionner un nombre fixe de caractéristiques d'image qui serviront de requêtes d'objet initiales pour le décodeur. Enfin, le décodeur avec les têtes de prédiction auxiliaires optimise itérativement les requêtes d'objets pour générer des boîtes et des scores de confiance [89].

Les dernières étapes de la dorsale réduisent non seulement considérablement la latence (35% plus rapide), mais améliore également la précision (0,4% AP plus élevé). CCFF est optimisé sur la base du module de fusion inter-échelles, qui insère plusieurs blocs de fusion constitués de couches convolutionnelles dans le chemin de fusion. Le rôle du bloc de fusion est de fusionner deux caractéristiques d'échelle adjacentes en une nouvelle caractéristique, et sa structure est illustrée dans la figure suivante. Le bloc de fusion contient deux convolutions 1×1 pour ajuster le nombre de canaux, N RepBlocks composés de RepConv [8] sont utilisés pour la fusion des caractéristiques, et les sorties à deux chemins sont fusionnées par addition élément par élément [89].

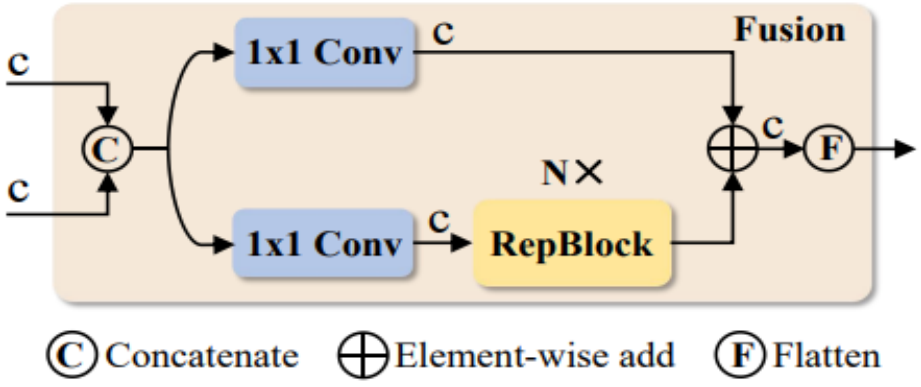


Figure 39 :Le bloc de fusion dans le CCFF [89].

Comment fonctionne le RT-DETR :

- Dans un premier temps, une image est traitée à l'aide d'un réseau neuronal convolutionnel pré-entraîné, en particulier une variante Resnet. Ce réseau extrait les caractéristiques des trois dernières couches de l'architecture.

- On utilise un encodeur hybride efficace qui traite les caractéristiques multi-échelles en découplant l'interaction intra-échelle et la fusion inter-échelle. Cette conception unique basée sur les transformateurs de vision réduit les coûts de calcul et permet la détection d'objets en temps réel. Les caractéristiques multi-échelles sont converties en un tableau séquentiel de caractéristiques d'image.

- Ensuite, un décodeur, équipé de têtes de prédiction auxiliaires, est utilisé pour affiner les requêtes d'objet. Ce processus facilite la génération directe de boîtes englobantes, éliminant ainsi le besoin de post-traitement supplémentaire (comme les NMS avec yolo) pour acquérir les scores et les coordonnées des boîtes englobantes.

- Puis une amélioration de l'initialisation des requêtes d'objets en utilisant une sélection de requêtes consciente de l'unité d'exploitation. Cela permet au modèle de se concentrer sur les objets les plus pertinents de la scène, améliorant ainsi la précision de la détection.

- Une vitesse d'inférence adaptable, permet des ajustements flexibles de la vitesse d'inférence en utilisant différentes couches de décodeur sans qu'il soit nécessaire de procéder à une nouvelle formation. Cette adaptabilité facilite l'application pratique dans divers scénarios de détection d'objets en temps réel.

3.5 Prétraitement des données

Le prétraitement des données est une étape cruciale pour garantir que les modèles puissent apprendre efficacement à détecter les déchets dans les images. Plusieurs étapes de prétraitement ont été effectuées pour préparer les données avant l'entraînement des modèles.

3.5.1 Chargement des données

Apres chargement des données à partir de Roboflow on compte 4503 images au total. Nous remarquons un déséquilibre significatif (cf. Tableau) dans la distribution des classes de l'ensemble de données ZeroWaste, ce qui peut dégrader les performances des méthodes de détection, par exemple les objets métalliques sont plus de 40 fois plus rares que le carton. Nous avons pu déterminer qu'il n'y'a que 166 images dans lesquelles la classe « cardboard » est absente parmi 3002 dans l'ensemble d'entrainement, et 29 parmi 572 dans l'ensemble de validation.

Tableau 2 : Répartition du nombre total d'images et d'objets dans les ensembles de train, validation et test.

Split	Images	Cardboard	Soft plastic	Rigid plastic	Métal	Objects
Train	3002	12940	4862	116	263	19225
Validation	572	2167	855	305	60	3387

Test	929	3428	1236	315	63	5042
Total	4503	18535	6953	1780	386	27744

3.5.2 Traitement des données

Pour pallier au déséquilibre entre les classes nous avons effectué un upsamling (suréchantillonnage) sur les images ne contenant pas la classe « cardboard », afin d'augmenter les valeurs en pixel des autres classes au détriment de celle-ci, permettant aux modèles de mieux capturer leurs caractéristiques et aussi de pénaliser la classe « cardboard » par rapports aux autres.

Le upsampling est effectué de la maniére suivante : chaque image sans la classe « carboard » est chargée avec la bibliothèque PIL, puis elle est convertie en tensor pytorch pour appliquer l'upsampling avec la fonction torch.nn.fonctional.interpolate pour un scale_factor=2 (ce qui signifie que l'image est multipliée par 2 sur la longueur et sur la largeur, elle devient donc 4x plus grande en pixel) en mode bucibique. Puis dans le fichier d'annotation les cordonnées des boites englobantes (bounding box) sont mises à l'échelle en fonction du scale_factor pour qu'elles ne soient pas corrompues.

De plus nous effectuons d'autres transformations spécifiques à chaque modèle, pour que les données traitées correspondent précisément aux attentes du modèle pré-entraîné.

1. RT-DETR

Nous convertissons l'image et les bounding boxes en tenseurs PyTorch. Une augmentation des données est effectuée avec la bibliothèque Albumentation qui garantit que les transformations affectent l'image et mettent à jour les cadres de délimitation des boites englobantes en conséquence. Pour aider le modèle à rester robuste face aux variations potentielles dans les images réelles, on applique :

- Une rotation aléatoire de 90 degrés
- Un retournement horizontal et vertical de 50%
- Un ajustement aléatoire sur la luminosité, le contraste, la saturation et la teinte de l'image. Rendant le modèle moins sensible aux variations de couleur et d'éclairage.



Figure 40 : Effet de l'augmentation des données.

AutoImageProcessor s'occupe du traitement des données d'image en créant les pixels_values, pixel_mask et labels avec lesquels le modèle DETR va s'entraîner. Puis effectue un reformatage des annotations pour répondre aux attentes du modèle. Avec AutoImagerocessor la normalisation avec (image_mean=[0.485, 0.456, 0.406]et image_std=[0.229, 0.224, 0.225]) est incluse. Un redimensionnement des images et des bounding boxes à une taille de 640x640 pixels, est aussi effectué pour uniformiser la taille des images en entrée.

Le RT-DETR applique aussi des couches d'usampling durant l'entrainement

2. YOLO11

Aprés téléchargement nous avons remarqué que le format des annotations (bounding box) ne correspondait avec le format des annotations qu'attend YOLO. Ainsi nous avons effectué des conversions pour que chaque bounding box soit définie par les coordonnées du centre (x, y), la largeur et la hauteur de la boîte, normalisées par rapport aux dimensions de l'image. Ensuite le fichier « data.yaml » a été modifié pour inclure les nouveaux chemins des annotations et des images après le prétraitement. Ce fichier est essentiel pour YOLOv11, car il contient les informations sur les classes d'objets à détecter et les chemins vers les données d'entraînement et de validation.

Notons qu'avec YOLO, Pendant l'entraînement, des techniques d'augmentation de données ont été appliquées en temps réel pour diversifier les données et améliorer la généralisation du modèle. Ces techniques incluent le flou gaussien, le flou médian, la conversion en niveaux de gris, et l'application de l'égalisation d'histogramme (CLAHE). Yolo applique un upsampling de manière générale par défaut.

Apart cela nous avons effectué les mêmes traitements que pour le RT-DETR.

3.6 Apprentissage des modèles

3.6.1 YOLO11

L'apprentissage du modèle YOLOv11 a été réalisé en utilisant le Framework Ultralytics, qui fournit l'implémentation des modèles YOLO. Dans notre cas nous avons utilisé le modèle yolo11s.pt qui est l'un des plus performant parmi ceux dédiés à la détection d'objet. Voici les étapes clés de l'apprentissage :

3.6.1.1 Initialisation du modèle :

Le modèle yolo11s a été initialisé avec des poids pré-entraînés sur le dataset COCO. Cette initialisation permet de bénéficier des caractéristiques déjà apprises par le modèle sur un large ensemble de données de 80 classes, ce qui permet d'avoir de meilleures performances qu'un modèle entrainé à partir de zéro.

3.6.1.2 Choix des hyperparamètres et entrainement :

Le modèle a été entrainé sur :

- Nombre d'époque : 50 ;
- Taille d'image fixée à 640x640 pixels ;
- Un batch size défini à 16 pour optimiser l'utilisation de la mémoire GPU tout en maintenant une bonne vitesse d'entraînement.
- Le modèle a été entraîné en utilisant l'algorithme d'optimisation AdamW, pour minimiser les fonctions de perte ;
- Learning rate (ou taux d'apprentissage) initial de 0.0001 ;
- Un verbose à True
- Fonction d'activation Silu

Le modèle est formé par des successions de plusieurs couches de convolution, d'extraction de caractéristiques, de pooling, de concaténation de caractéristiques (ou couche fully connected), de suréchantillonnage et une dernière couche de détection. Ci-dessous la structure des couches et leurs successions :

Input : L'image d'entrée est traitée par une série de couches de convolution (Conv) et de modules C3k2 pour extraire des caractéristiques.

Backbone : Les couches Conv et C3k2 forment le "backbone" du modèle, qui extrait des caractéristiques à différents niveaux de granularité. Le module SPPF (Spatial Pyramid Pooling Fast) est utilisé pour agréger les caractéristiques à différentes échelles.

Neck : Les modules C2PSA, Upsample, et Concat forment le "neck" du modèle, qui fusionne les caractéristiques de différentes résolutions pour améliorer la détection.

Head : La couche Detect du modèle, qui prédit les bounding boxes, les classes et les scores de confiance.

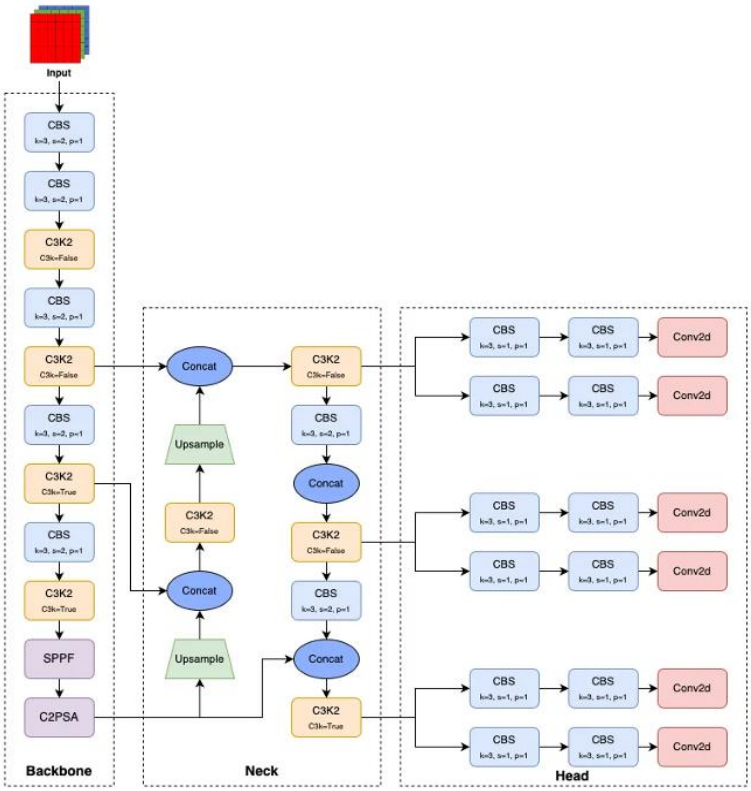


Figure 41 : Structure des couches de yolo11 [90].

3.6.1.3 Métriques d'évaluation :

Après chaque époque, le modèle a été évalué sur l'ensemble de validation pour surveiller les performances et éviter le surapprentissage :

- La précision (precision) ;
- Le rappel (recall) ;
- La mAP (mean Average Precision) pour un seuil de 50%, et un autre en 50 et 95%.

Nous avons aussi défini des métriques pour surveiller les pertes en entrainement et en validation :

- IoU (Intersection over Union) fixé à un seuil de 0.7 pour la correspondance entre la boite réelle et la boite prédite ;
- Box_loss (erreur sur les bounding boxes), fixé à un seuil de 0.75 ;
- Cls_loss (erreur sur les classes prédites), fixé à un seuil de 0.5 ;
- Dfl_loss (Perte de distribution focalisé), fixé à un seuil de 1.5 ;
- Pose : coefficient de la perte de pose (utilisé pour les modèles de détection de pose) fixé à un seuil de 12.0 ;
- Kobj (objectness loss), coefficient de la perte d'objet, fixé à un seuil de 1.0.

Avec YOLO la perte globe se calcule comme suit :

(Total Loss=box*box_loss+cls*cls_loss+dfl*dfl_loss+pose*pose_loss+kobj*obj_loss

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/50      4.52G      1.206      2.033      1.326        112        640: 100% 188/188 [01:26<00:00,  2.18it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:09<00:00,  1.87it/s]
             all        572       3687      0.224      0.278      0.203      0.114

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/50      4.36G      1.232      1.729      1.318        117        640: 100% 188/188 [01:24<00:00,  2.23it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:07<00:00,  2.25it/s]
             all        572       3687      0.299      0.312      0.226      0.134

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/50      4.43G      1.248      1.728      1.334        153        640: 100% 188/188 [01:23<00:00,  2.26it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:07<00:00,  2.55it/s]
             all        572       3687      0.546      0.288      0.242      0.146

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 4/50      4.42G      1.237      1.684      1.322         98        640: 100% 188/188 [01:22<00:00,  2.27it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:07<00:00,  2.55it/s]
             all        572       3687      0.305      0.325      0.269      0.166

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 5/50      4.39G      1.209      1.641       1.31        143        640: 100% 188/188 [01:22<00:00,  2.27it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:06<00:00,  2.57it/s]
             all        572       3687      0.303      0.316      0.244      0.153
```

Figure 42 : Apprentissage de yolo

3.6.2 RT-DETR

Le modèle a été formé à partir d'expériences sur les jeux de données COCO et Objects365, où RT-DETR est formé sur COCO train2017 et validé sur le jeu de données COCO val2017.

3.6.2.1 Initialisation du modèle :

Ici nous chargeons le modèle avec les poids preentrainés de PekingU/rtdetr_r50vd_coco_o365, une variante des modèles RT DETR, qui après une préformation avec Objects365, le RT-DETR-R50 atteint 55,3% AP.

3.6.2.2 Choix des hyperparamètres et entrainement

Nous faisons un résumé des hyperparamètres avec le tableau suivant :

- Nombre d'époque : 25 ;
- Taille d'image fixée : 640x640 pixels ;
- Norme maximale du gradient : 2 ;
- Taux d'apprentissage initial : 0.00005 ;
- Nombre d'étapes d'échauffement avant d'appliquer le taux d'apprentissage : 300 ;
- Nombre d'échantillons par batch en entrainement : 16 ;
- Nombre d'échantillons par batch en validation : 8 ;
- Gradiant cliping (Clippage du gradient) permet d'éviter les explosions de gradient : 0.1 (max_grad_norm)
- Pas de gradient accumulation : 1
- Algorithme d'optimisation : AdamW fixée à 0.00005 ;
- Collate_fn : fonction pour regrouper les données en batch.

❖ Fonction d'activation : gelu (encodeur), relu (décodeur)

Le modèle est formé par des blocs d'encodeur et de décodeur, formé chacun de couches de convolution, de pooling, d'extractions des caractéristiques.

RT-DETR est une architecture combinant des convolutions optimisées (RepC3), un module d'optimisation de caractéristiques (CCFM) et un décodeur basé sur des BasicBlocks, visant à améliorer l'efficacité et la rapidité de la détection d'objets.

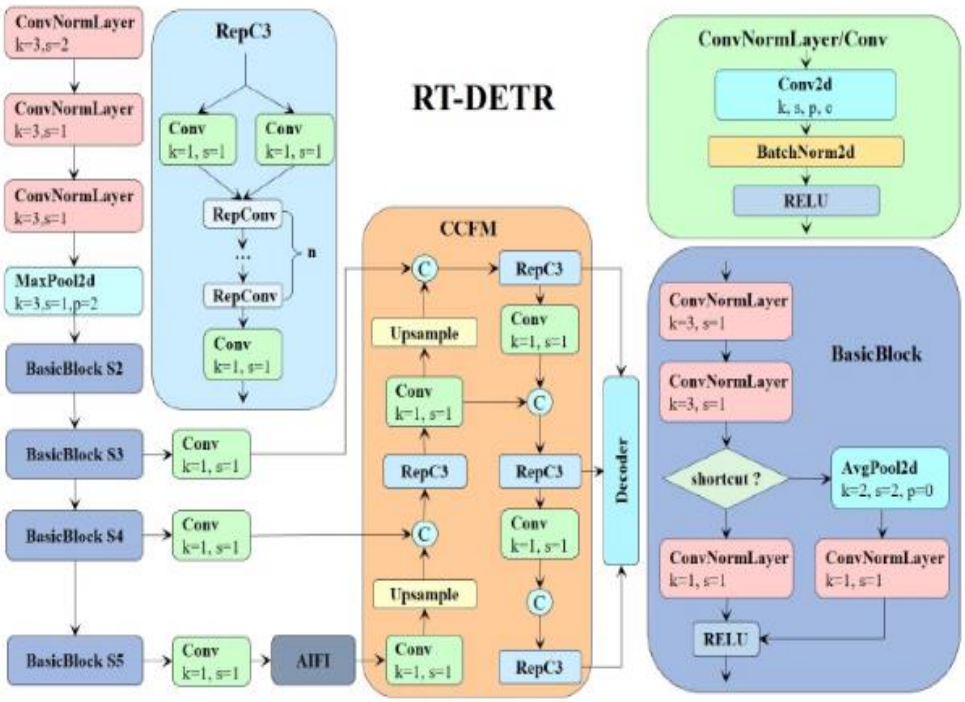


Figure 43 : Structure des couches de RT-DETR [91].

Epoch	Training Loss	Validation Loss	Map	Map 50	Map 75	Map Small	Map Medium	Map Large	Mar 1	Mar 10	Mar 100	Mar Small	Mar Medium	Mar Large
1	No log	18.099192	0.062600	0.096500	0.064500	0.001600	0.024900	0.080800	0.058700	0.274700	0.405500	0.026300	0.298100	0.472200
2	No log	14.844689	0.146600	0.207100	0.152700	0.009500	0.098600	0.189500	0.185000	0.478000	0.575900	0.160100	0.488800	0.653600
3	29.803600	14.291814	0.186300	0.267900	0.189900	0.020000	0.117500	0.242300	0.204200	0.562700	0.659400	0.203600	0.563600	0.745900
4	29.803600	14.637585	0.208100	0.305500	0.216600	0.019900	0.127100	0.266400	0.225400	0.579600	0.676900	0.193700	0.602700	0.742900
5	29.803600	14.757751	0.212800	0.305100	0.217500	0.025800	0.130400	0.271100	0.239100	0.598600	0.682300	0.208900	0.585400	0.764300
6	14.187100	15.323964	0.226900	0.329100	0.234500	0.025300	0.135600	0.288800	0.261500	0.591700	0.684500	0.133400	0.595700	0.764600
7	14.187100	15.598909	0.213200	0.312100	0.223100	0.024100	0.135200	0.272400	0.251800	0.575900	0.668100	0.183400	0.569200	0.752200
8	12.437300	15.799745	0.231900	0.341700	0.236100	0.027700	0.134100	0.301400	0.285400	0.580600	0.657300	0.160900	0.548200	0.754100
9	12.437300	15.394566	0.230800	0.340200	0.233800	0.022900	0.136000	0.309000	0.275400	0.598100	0.693800	0.226200	0.627000	0.748100
10	12.437300	15.920168	0.225400	0.340900	0.230000	0.026600	0.129500	0.293800	0.278100	0.570300	0.653500	0.300700	0.577500	0.712600
11	11.530500	16.514593	0.221600	0.334900	0.221100	0.032100	0.134700	0.296800	0.264200	0.545800	0.614200	0.137500	0.545300	0.673500
12	11.530500	16.827610	0.183600	0.288500	0.185000	0.030500	0.118900	0.253200	0.233900	0.516400	0.584700	0.229300	0.517000	0.640600
13	11.530500	16.947924	0.198900	0.308300	0.203500	0.028900	0.113600	0.262100	0.268700	0.524200	0.593700	0.204300	0.497000	0.673100
14	11.004300	16.500376	0.220000	0.331200	0.222700	0.027100	0.117800	0.294400	0.272400	0.561500	0.629500	0.207000	0.542800	0.697400

Figure 44 : Apprentissage de RT-DETR.

3.6.2.3 Métriques d'évaluation :

Nous rapportons les mesures COCO standard, y compris AP (moyenne sur des seuils IoU uniformément échantillonnés allant de 0,50 à 0,95 avec un pas de 0,05), AP50, AP75, ainsi que AP à différentes échelles : APSmall, APMedian, APLarge. Après chaque époque, le modèle a été évalué sur l'ensemble de validation pour surveiller les performances et éviter le surapprentissage. Les métriques clés telles que :

- La MAP (Mean Average Precision) : Il s'agit de la précision moyenne, utilisé pour évaluer la qualité de la classification. Elle mesure la précision en prenant en compte les résultats classés par ordre décroissant de confiance.
- La MAR (Mean Average Recall) : C'est la moyenne de la précision de rappel (recall), qui mesure la capacité du modèle à identifier correctement les objets dans les données. Le rappel est la proportion des éléments pertinents trouvés par rapport au nombre total d'éléments pertinents.

Nous avons aussi défini des métriques pour surveiller les pertes en entrainement et en validation telles que :

- Utilisation de la perte focale (use_focal_loss = true) avec des hyperparamètres alpha = 0.75 et gamma = 2.
- Pondération des pertes : bbox = 5, giou = 2, vfl = 1.
- Global Step : Le nombre total d'itérations d'entraînement réalisées.
- Gradient Norm : Il mesure la norme du gradient, qui peut être un indicateur de la stabilité du processus d'entraînement. Des gradients trop grands peuvent conduire à un entraînement instable.

- ❖ FLOPs (Floating Point Operations) : fait référence au nombre total d'opérations en virgule flottante effectuées lors de l'entraînement ou de l'inférence.
- ❖ Samples Per Second : Le nombre d'échantillons traités par seconde pendant l'évaluation et l'entraînement du modèle.
- ❖ Steps Per Second : Le nombre de pas ou d'itérations par seconde pendant l'évaluation et l'entraînement du modèle.

3.7 Résultats et Discussions

Les deux modèles ont été évalués sur le jeu de données d'images évoqué dans la section 3.3. Les expériences ont été réalisées sur des sous-ensembles distincts pour l'apprentissage, la validation et le test. Le nombre de paramètres entraînables pour chaque modèle utilisé est de 9.414.348 et environ 42 millions respectivement pour les modèles yolo et rt-detr.

Nous visualiserons les courbes d'évaluation des performances et des pertes afin d'avoir une vision plus claire du comportement des modèles. Nous afficherons également les matrices de confusion pour chaque modèle afin d'analyser et visualiser les erreurs de détection.

3.7.1 YOLO11

L'apprentissage du modèle yolo a pris 1h 25min pour 50 époques avec un temps d'inférence de prétraitement (Preprocess) de 0.2 ms/image, une Inférence de 2.5 ms (temps nécessaire pour faire la prédiction) /image et une inférence de post-traitement (Postprocess) de 2.8 ms (application du seuil de confiance et le NMS) par image. Dans ce qui suit nous allons présenter les résultats de l'inférence sur l'ensemble de validation. Les résultats en entrainement sont presque identiques à ceux en validation.

Tableau 3 : Inférence sur les données de validation de yolo.

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	572	3687	0.43	0.388	0.35	0.235
cardboard	543	2361	0.51	0.598	0.523	0.366
metal	48	60	0.289	0.133	0.11	0.0495
rigid_plastic	236	324	0.438	0.284	0.316	0.224
soft_plastic	398	942	0.482	0.536	0.453	0.299

Le tableau 3 présente les résultats de l'évaluation sur l'ensemble de validation de 572 images contenant 3687 instances. Les performances globales montrent une précision (Box(P)) de 0.433, un rappel (R) de 0.387, une moyenne de la précision moyenne (mAP50) de 0.35 et une mAP50-95 de 0.234. Parmi les classes spécifiques, le carton (la plus représentée) obtient les meilleurs résultats avec une précision de 0.508, un rappel de 0.596, une mAP50 de 0.522. Le métal (sous-représenté) a les performances les plus faibles avec une précision de 0.306, un rappel de 0.133, une mAP50 de 0.11. Le plastique rigide et le plastique souple montrent des performances intermédiaires, avec des précisions respectives de 0.436 et 0.483, et des rappels de 0.284 et 0.535. Ces résultats indiquent des variations significatives dans la détection selon les matériaux

et confirme que la sous-représentation des autres classes (métal, plastique) a eu un grand impact dans les performances.

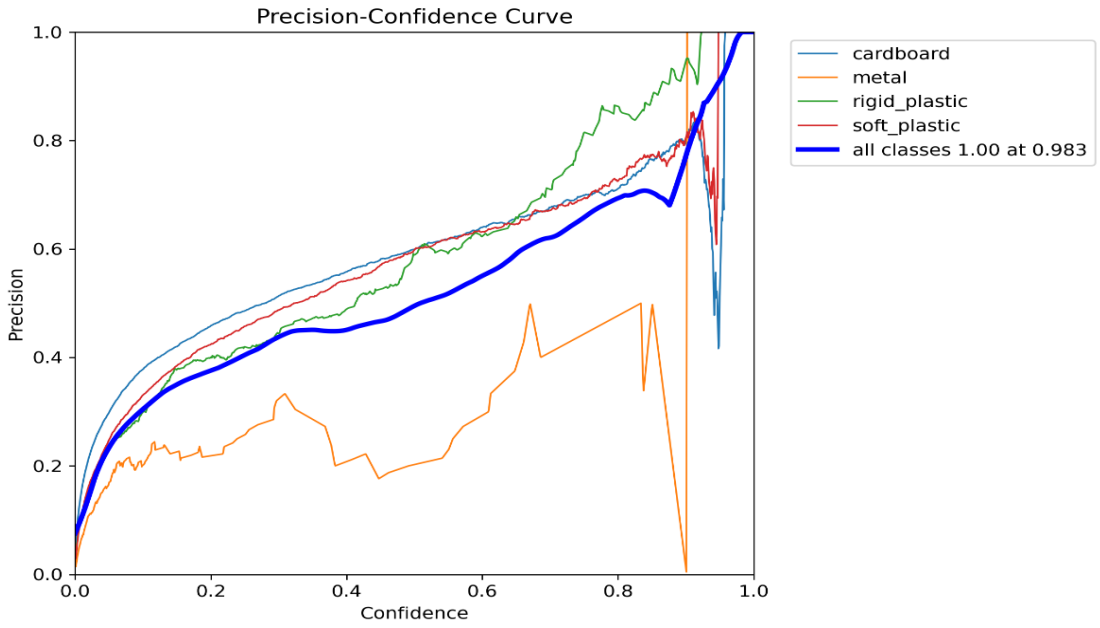


Figure 45 : Courbe de Précision de yolo.

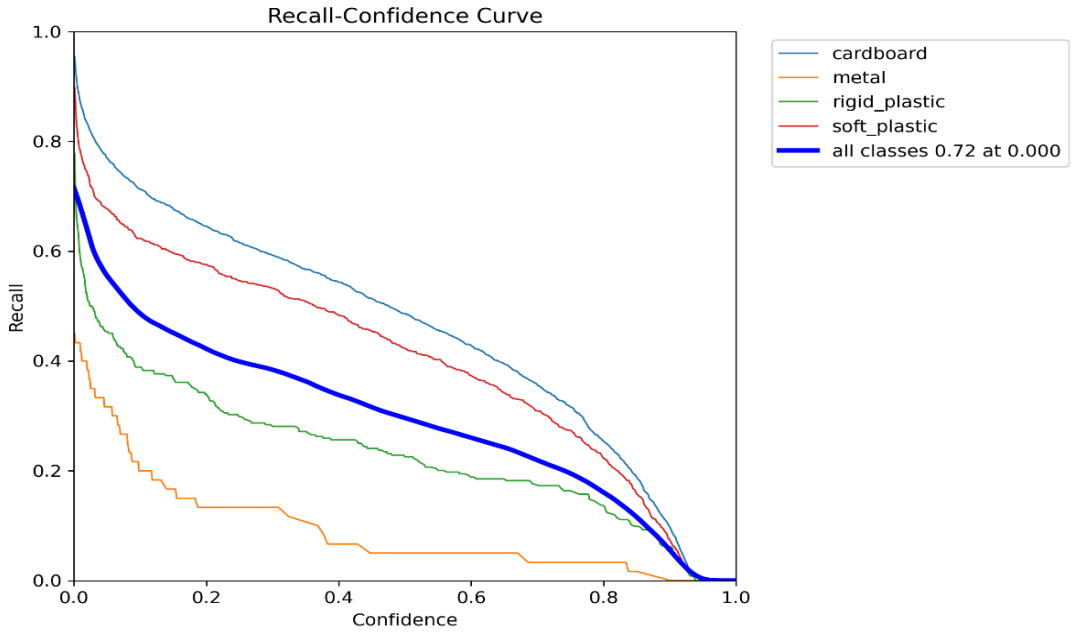


Figure 46 : Courbe du Rappel de yolo.

Les courbes de précision (Figure 46) représente la proportion de prédictions correctes parmi toutes celles effectuées avec un niveau de confiance donné et le rappel (Figure 47) mesure la proportion d'objets détectés parmi tous les objets véritables existants. On voit que, contrairement à la précision, le rappel diminue lorsque la confiance augmente. Car lorsque le seuil de confiance est petit, le modèle devient plus strict en détection et implique que le rappel devient alors plus grand.

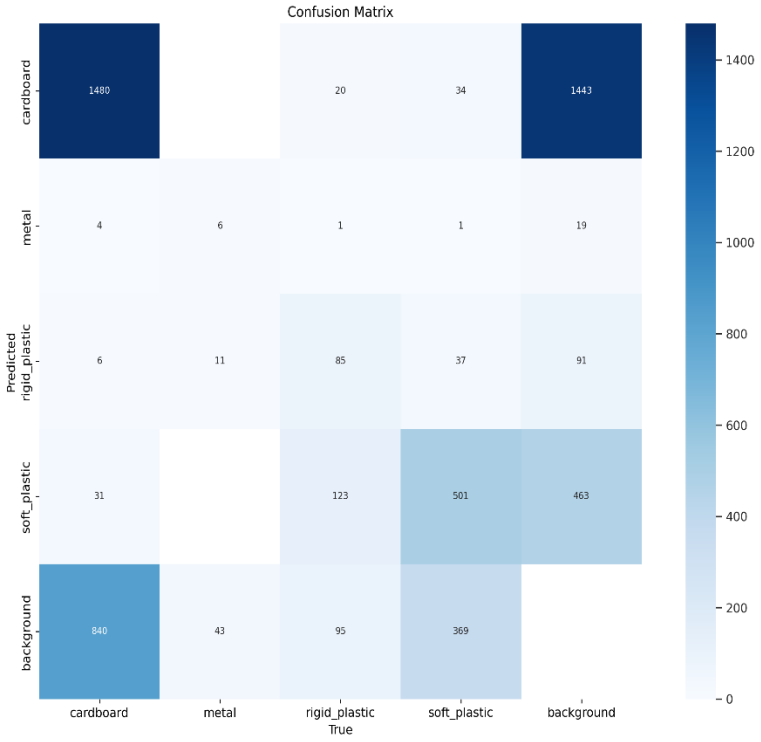


Figure 47 : Matrice de confusion de yolo.

La matrice de confusion permet de comparer les prédictions du modèle avec les véritables classes (valeurs réelles). Chaque ligne de la matrice représente les instances de la classe prédite, tandis que chaque colonne représente les instances de la classe réelle (et vice versa).

La matrice de confusion (Figure 48) révèle que le modèle détecte bien le carton (1 480 bonnes prédictions), mais il le confond souvent avec l'arrière-plan (1 443 erreurs), ce qui indique des difficultés à distinguer certaines textures. La classe métal est très mal détecté avec seulement 6 prédictions correctes, confirmant un faible rappel. Le plastique rigide (85 bonnes prédictions) est souvent confondu avec le plastique souple (37 erreurs) et l'arrière-plan (91 erreurs). Ces résultats montrent que le modèle pourrait être optimisé en améliorant les données pour le métal, ainsi que la détection pour réduire les erreurs avec l'arrière-plan, et en utilisant des techniques d'augmentation plus avancées pour mieux différencier le plastique rigide et souple. Tout ceci confirme que le déséquilibre entre les classes dégrade fortement les performances.

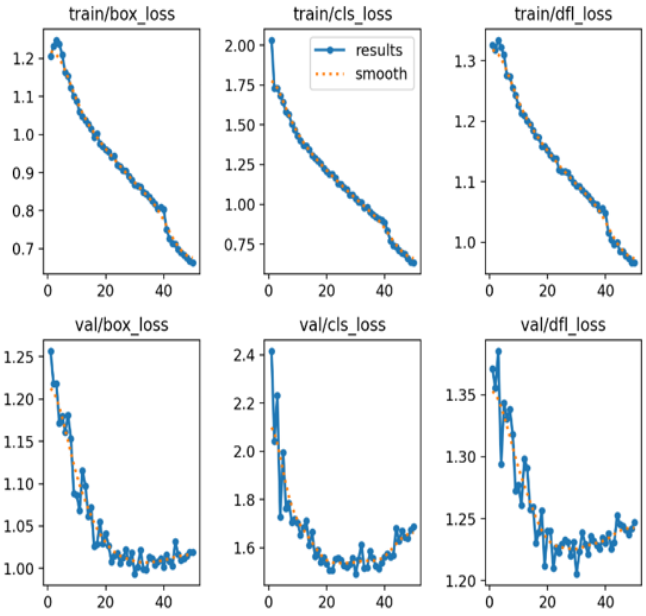


Figure 48 : Evolution des pertes de yolo.

Ces représentations (Figure 49) montrent l'évolution des pertes d'entraînement et de validation, ainsi que des métriques de performance du modèle au fil des itérations. Les pertes box_loss, cls_loss et dfl_loss diminuent régulièrement, indiquant une bonne convergence du modèle.

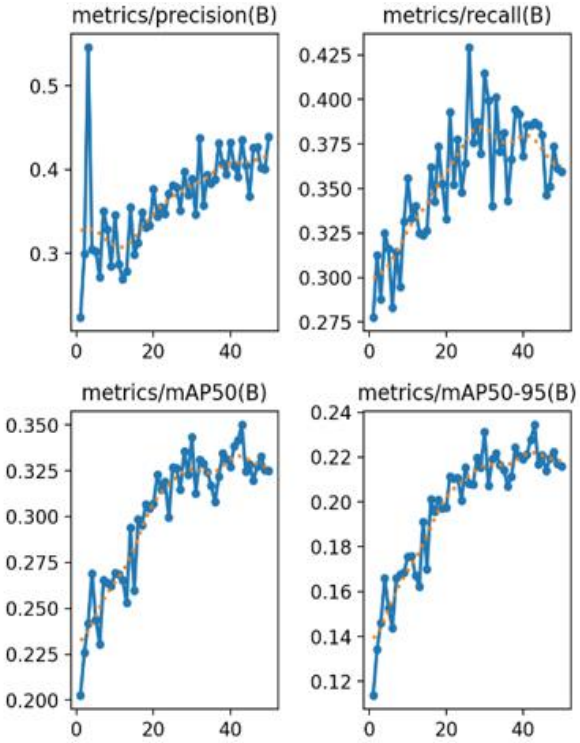


Figure 49 : Evolution des performances de yolo.

Les métriques de performance (Figure 50), comme la précision, le rappel et le mAP (Mean Average Précision), montrent une progressive, avec le mAP50 et le mAP50-95 qui augmentent de manière stable. Cela suggère que le modèle devient plus précis et efficace au fil du temps.

3.7.2 RT-DETR

L'apprentissage du modèle rt-detr a pris 3h 54m 8s pour 25 époques, avec un temps d'inférence d'environ 65,13 secondes par image avec une vitesse de 8,782 échantillons par seconde.

Nous présentons les résultats en entrainement, validation et test.

En entrainement notre modèle obtient globalement pour toutes les classes confondues, un MAP (50_95) de 33%, un MAP50 de 41% et un MAP75 de 35%.

Tableau 4 : Performances en validation avec rt-detr.

MAP Globale	0.1904	MAR_1	0.2403
MAP (50)	0.2955	MAR_10	0.5004
MAP (75)	0.1902	MAR_100	0.5687
MAP (Carboard)	0.2566	MAR_100 cardboard	0.5262
MAP (métal)	0.1053	MAR_1OO métal	0.57
MAP (soft-plastic)	0.2455	MAR_1OO soft-plastic	0.5393
MAP (rigid-plastic)	0.1541	MAR_1OO rigid-plastic	0.6393

Dans le tableau 4, l'évaluation des performances du modèle montre un mAP global de 0.1904, avec une précision plus élevée pour le mAP50 (0.2955) et une performance similaire pour le mAP75 (0.1902). L'analyse par catégorie indique que le modèle détecte mieux le carton (0,2566) et le plastique souple (0,2455), tandis que la performance sur le métal (0,1053) et le plastique rigide (0,1541) reste plus faible. En termes de rappel moyen (MAR), le modèle atteint 0.2403 pour 1 détection, 0.5004 pour 10 et 0.5687 pour 100. Les scores MAR spécifiques par classe montrent une meilleure récupération pour le plastique rigide (0.6393), suivi du métal (0.57), du plastique souple (0.5393) et du carton (0.5262) . Ces résultats font que le modèle est capable de détecter un nombre élevé d'objets, mais nécessitant des ajustements dans les données pour améliorer la qualité des prédictions, notamment sur les catégories les moins performantes.

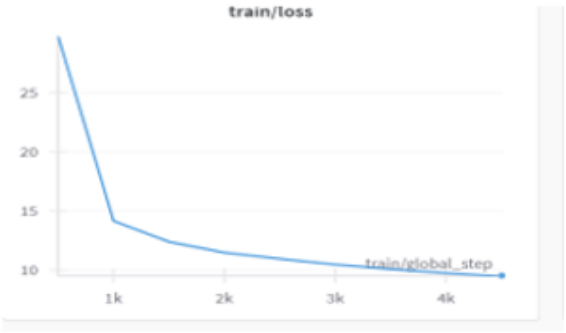


Figure 50 : Evolution de perte en entrainement de rt-detr.

On constante que la courbe de perte se comporte convenablement car la perte diminue progressivement jusqu'à atteindre une certaine constance.

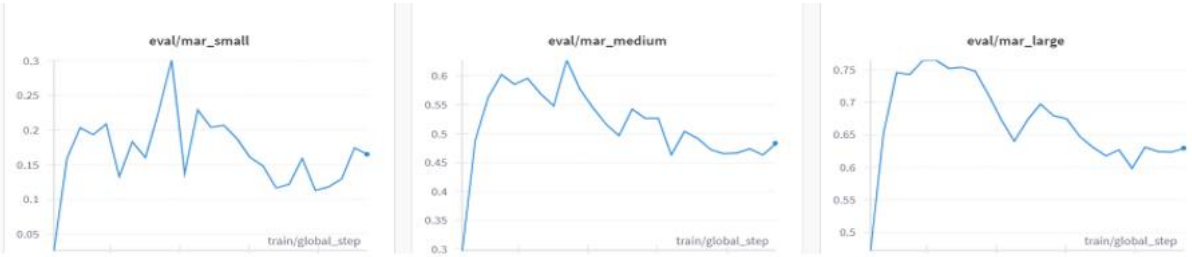


Figure 51 : Les courbes d'évaluation des MAR en entrainement de rt-detr.

On remarque que l'évolution très variable et décroissante, ce qui suggère globalement de meilleures précisions.

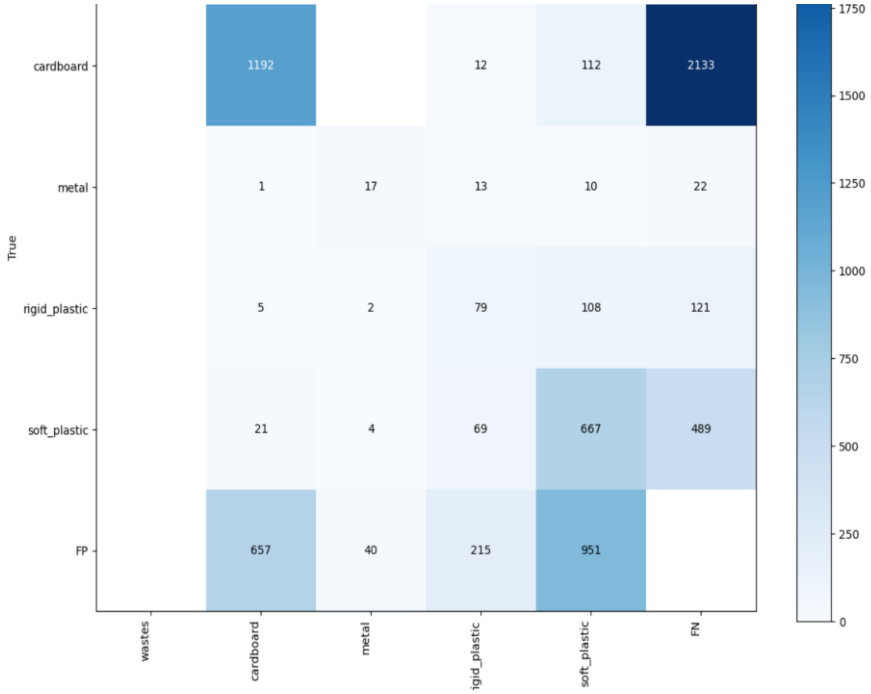


Figure 52 : Matrice de Confusion de rt-detr.

La matrice (Figure 53) montre que, le carton est bien détecté avec 1 192 classifications correctes, mais il y a encore 112 erreurs vers le plastique rigide et 2 133 faux négatifs, ce qui montre un problème de détection manquée. Le métal est faiblement détecté avec seulement 17 classifications correctes et de nombreuses confusions avec le plastique rigide et les faux négatifs. Le plastique rigide est mieux détecté (79 corrects) mais présente des erreurs. Le plastique souple a 667 bonnes classifications, mais un grand nombre (951) de fausses détections, montrant que le modèle confond souvent cette classe avec d'autres. Enfin, la quantité élevée de faux négatifs (FN) et de fausses prédictions (FP) indique que le modèle peine encore à généraliser correctement et nécessite un meilleur équilibrage des classes.

En résumé quand on se réfère à la source des données de ZeroWaste, aux vues de leur performance en détection d'objet avec le modèle Mask RCNN (cf. Tableau 3 .4)

Tableau 5 : Comparaison des Performances (map) de notre modele yolo et de la source de ZeroWaste [81].

YOLO		Mask R CNN	
Carboard	52.3%	Carboard	29.85%
Soft Plastic	45.3%	Soft Plastic	22.09%
Rigid Plastic	31.6%	Rigid Plastic	15.70%
Metal	11%	Metal	1.64%
Toutes classes	35%	Toutes classes	17.32%

Nous pouvons dire que nos performances sont bonnes comparé à celles-ci. Les résultats expérimentaux de YOLO et RT-DETR indiquent que l'ensemble de données ZeroWaste proposé est considérablement plus difficile à utiliser même pour les méthodes de détection les plus avancées, YOLO étant légèrement plus performant que les autres méthodes. Toutes les méthodes ont particulièrement du mal avec les petits objets, qui sont généralement plus difficiles à étiqueter correctement.

3.8 Prédiction avec l'interface Streamlit

Streamlit est un Framework Python open source destiné aux data scientists et aux ingénieurs IA/ML pour fournir des interfaces et des applications dynamiques avec seulement quelques lignes de code. Une interface graphique a été mise en place pour visualiser les redictions en temps réel. Nous avons utilisé notre modèle YOLO pour la détection avec les données test.



Figure 53 : Page d'accueil de l'interface streamlit.

Avec streamlit on crée un environnement virtuel ou on installe toutes les dépendances de l'application puis on l'exécute à partir de notre terminal.

Pour chaque l'image on clique sur le bouton 'parcourir les fichiers', ensuite on charge le fichier et enfin la détection est faite automatiquement.



Figure 54 : Exemples de prédictions sur les données test.

3.9 Conclusion

Lors de la conception de notre système de détection des déchets nous avons constaté l'importance cruciale d'un ensemble d'avoir un ensemble de données diversifié, équilibré et qui offre une bonne représentation des objets. ZeroWaste est l'un des plus grands ensembles de données en matière de détection de déchets abordant l'identification des objets critiques qui ont un impact sur l'efficacité du processus de tri des déchets. Nos résultats montrent que les méthodes de détection de pointe actuelles ne peuvent pas gérer efficacement ce domaine complexe dans la nature. Nous prévoyons que ceci motivera la communauté de la vision par ordinateur à développer des méthodes plus efficaces en matière de données applicables à une plus large gamme de problèmes de vision par ordinateur dans le contexte de la gestion des déchets.

Conclusion Générale

La gestion des déchets est un secteur à fort potentiel économique et de développement durable. Une gestion efficiente des déchets est source de richesse (utilisation des matériaux de valorisation comme matières premières et création d'emplois) et de bien-être social. L'économie informelle de la récupération, de la collecte et de la valorisation des déchets solides urbains est à la fois une stratégie de survie pour les populations défavorisées dans de nombreux pays en développement, et une partie importante des systèmes de gestion des déchets et de recyclage. Par conséquent Le tri automatisé du recyclage, alimenté par les méthodes de vision par ordinateur en générale et d'apprentissage profond en particulier, offre une approche innovante pour une gestion automatisé des déchets. Cette technologie rationalise le processus de recyclage, stimule l'efficacité globale des processus de tri et améliore l'économie circulaire, ouvrant la voie à des avantages économiques et environnementaux encore plus importants.

L'objectif de ce projet était de réaliser un système de détection automatique des déchets, efficace et avec une grande capacité de généralisation des différentes classes à l'aide du Deep Learning. Pour cette fin, nous avons proposé deux méthodes (modèles) de détections basées sur l'apprentissage par transfert : YOLOs11 et RT-DETR pré-entrainés sur la base de données COCO.

On trouve dans ce manuscrit les différentes étapes de notre approche. La première étape consiste au chargement des données suivie des étapes de pré-traitements qui a été une étape très importante. L'autre étape importante a été le choix des réseaux de base et des hyperparamètres adéquats afin d'obtenir des modèles robustes et précis. Et enfin, on termine par la détection et la classification des 4 types de déchets. L'expérience a été réalisée avec la base de données Zerowaste de *[81]* et les résultats obtenus comparé à ceux de *[81]* sont assez bons, avec une précision globale de 43% avec YOLO et 41% avec RT-DETR. Ces performances révèlent l'efficacité de nos modèles, d'autant plus qu'il s'agit de détecter la présence de plusieurs types de déchets dans des scenarios à niveau d'encombrement élevé.

Dans le but de l'amélioration de notre système, plusieurs points nécessitent un approfondissement, ouvrant ainsi la voie à de nouvelles perspectives et futurs travaux de recherche dans la gestion automatique des déchets. D'abord le plus important serait d'avoir une base de données plus diversifié, assez équilibré offrant des représentations différentes pour chaque classe, ce qui permet aux modèles de bien généraliser sur de nouvelles données. Ensuite essayer de faire cette collecte dans le contexte sénégalais, car il est important de pouvoir adapter la solution au contexte pour lequel elle a été développé. Enfin nous envisageons d'élargir l'étude en y incluant plus de classes de déchets. Ce système permettra d'optimiser les processus de collecte des déchets en envoyant des alertes géolocalisées aux services concernés, facilitant ainsi des interventions plus ciblées et efficaces. L'objectif est de rendre la gestion des déchets plus écologique, en réduisant la pollution, les trajets inutiles et le gaspillage des ressources, tout en protégeant les espaces naturels.

Glossaire

Analyse en Composantes Principales (PCA) : Technique de réduction de dimensionnalité qui transforme les données en un ensemble de composantes orthogonales.

Apprentissage Automatique (Machine Learning) : Domaine de l'intelligence artificielle qui permet aux systèmes d'apprendre à partir de données sans être explicitement programmés.

Apprentissage Profond (Deep Learning) : Sous-domaine du machine learning qui utilise des réseaux de neurones profonds pour modéliser des données complexes.

Autoencodeur (AE) : Réseau de neurones utilisé pour apprendre des représentations compressées des données, souvent utilisé pour la réduction de dimension ou la génération de données.

Common Objects in Context (COCO) : Base de données largement utilisée pour la détection d'objets, contenant des images annotées avec des boîtes englobantes et des étiquettes de classe.

Convolutional Neural Network (CNN) : Type de réseau de neurones spécialisé dans le traitement des images, utilisant des couches de convolution pour extraire des caractéristiques.

Descente de Gradient Stochastique (SGD) : Algorithme d'optimisation utilisé pour entraîner les modèles de machine learning.

Égalisation d'Histogramme Adaptative à Contraste Limité (CLAHE) : Technique de traitement d'image utilisée pour améliorer le contraste local.

False Negative (FN) : Cas où le modèle prédit incorrectement l'absence d'une classe.

False Positive (FP) : Cas où le modèle prédit incorrectement la présence d'une classe.

Floating Point Operations (FLOPs) : Mesure du nombre d'opérations en virgule flottante effectuées par un modèle, souvent utilisée pour évaluer la complexité computationnelle.

Gated Recurrent Unit (GRU) : Variante simplifiée des LSTM, utilisée pour modéliser des séquences.

Generative Adversarial Network (GAN) : Architecture de réseau de neurones composée de deux réseaux (générateur et discriminateur) qui s'affrontent pour générer des données réalistes.

Imagerie Hyperspectrale (HSI) : Technique d'imagerie qui capture des informations spectrales détaillées pour chaque pixel d'une image.

Intelligence Artificielle (IA) : Domaine de l'informatique visant à créer des systèmes capables de réaliser des tâches nécessitant normalement l'intelligence humaine.

Intersection over Union (IoU) : Mesure utilisée pour évaluer la qualité de la détection d'objets, calculée comme le rapport entre l'intersection et l'union des boîtes englobantes prédites et réelles.

Long Short-Term Memory (LSTM) : Variante des RNN conçue pour capturer des dépendances à long terme dans les données séquentielles.

Material Recovery Facility (MRF) : Installation où les déchets recyclables sont triés et préparés pour le recyclage.

Mean Average Precision (mAP) : Mesure de la précision moyenne sur plusieurs classes ou seuils de détection.

Natural Language Processing (NLP) : Domaine de l'intelligence artificielle qui traite de la compréhension et de la génération du langage humain.

Non-Maximum Suppression (NMS) : Technique utilisée pour éliminer les boîtes englobantes redondantes en ne conservant que celles avec le score de confiance le plus élevé.

Précision (Precision) : Mesure de la proportion de prédictions correctes parmi toutes les prédictions positives.

Rappel (Recall) : Mesure de la proportion de prédictions correctes parmi toutes les instances positives réelles.

Real-Time Detection Transformer (RT-DETR) : Variante de DETR optimisée pour la détection en temps réel.

Rectified Linear Unit (ReLU) : Fonction d'activation couramment utilisée dans les réseaux de neurones, définie comme max (0, x).

Réseau Antagoniste Génératif (GAN) : Architecture de réseau de neurones composée de deux réseaux (générateur et discriminateur) qui s'affrontent pour générer des données réalistes.

Réseau de Neurones Convolutifs (CNN) : Type de réseau de neurones spécialisé dans le traitement des images, utilisant des couches de convolution pour extraire des caractéristiques.

Réseau de Neurones Récurrents (RNN) : Type de réseau de neurones conçu pour traiter des données séquentielles, comme le texte ou les séries temporelles.

Support Vector Machine (SVM) : Algorithme de classification supervisée utilisé pour la classification et la régression.

True Negative (TN) : Cas où le modèle prédit correctement l'absence d'une classe.

True Positive (TP) : Cas où le modèle prédit correctement la présence d'une classe.

Unité Linéaire d'Erreur Gaussienne (GELU) : Fonction d'activation qui combine les propriétés de la ReLU et de la distribution gaussienne.

Unité Linéaire Rectifiée (ReLU) : Fonction d'activation couramment utilisée dans les réseaux de neurones, définie comme max (0, x).

Unité Linéaire Sigmoïde (SiLU) : Fonction d'activation utilisée dans les réseaux de neurones, combinant les propriétés de la sigmoïde et de la ReLU.

Vision par Ordinateur (Computer Vision) : Domaine de l'intelligence artificielle qui traite de la compréhension et de l'analyse des images et des vidéos.

You Only Look Once (YOLO) : Algorithme de détection d'objets en temps réel qui prédit les boîtes englobantes et les classes en une seule passe.

Bibliographie

[1] « Nouveau rapport, la production mondiale de déchets mondial », World Bank. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://www.banquemondiale.org/fr/news/press-release/2018/09/20/global-waste-to-grow-by-70-percent-by-2050-unless-urgent-action-is-taken-world-bank-report

[2] APA, « Sénégal : vers la valorisation de 25% des déchets solides (experte) », APAnews - Agence de Presse Africaine. Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://fr.apanews.net/environnement/environment/senegal-vers-la-valorisation-de-25-des-dechets-solides-experte/

[3] « Evolutive HyperNetwork for Convolutional Neural Networks (CNN); application to Environmental remote sensing monitoring ».

[4] « Environment- Meaning, Definition, Function, Example », BYJUS. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://byjus.com/commerce/meaning-and-functions-of-environment/

[5] « De la collecte au traitement des déchets | FNADE, Fédération Nationale des Activités de la Dépollution et de l'Environnement ». Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://www.fnade.org/fr/acteur-environnement/collecte-traitement

[6] « De la collecte au traitement des déchets | FNADE, Fédération Nationale des Activités de la Dépollution et de l'Environnement ». Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://www.fnade.org/fr/acteur-environnement/collecte-traitement

[7] « ÉVALUATION DU MARCHÉ DES DÉCHETS EN VUE DE L'UTILISATION DE L'AVANTAGE COOPÉRATIF DANS LE SECTEUR ».

[8] « Cooperation among workers in the informal economy A focus on home-based workers and waste pickers. A Joint ILO and WIEGO Initiative.pdf ».

[9] L. Haouaoui et F. Loukil, « Evaluation du système de gestion des déchets ménagers en Tunisie ».

[10] J. McCarthy, « WHAT IS ARTIFICIAL INTELLIGENCE? ».

[11] G. Yacine, E. B. Djamel, et C.-E. G. Abdelkader, « Apprentissage automatique pour une classification des images basée sur les descripteurs », déc. 2020, Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://dspace.univ-sba.dz/handle/123456789/1614

[12] « RPubs - SCI1035-CH1 ». Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://rpubs.com/alexvezeau/SCI1035-CH1

[13] « L'INTELLIGENCE ARTIFICIELLE DEFINITION - GENERALITES - HISTORIQUE - DOMAINES ».

[14] « Jean-Michel RODRIGUEZ ». Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://www.editions-eni.fr/jean-michel-rodriguez

[15] « An Automated Machine Learning Approach for Smart Waste Management Systems ». Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://ieeexplore.ieee.org/abstract/document/8709695

[16] W. Xia, Y. Jiang, X. Chen, et R. Zhao, « Application of machine learning algorithms in municipal solid waste management: A mini review », *Waste Manag. Res. J. Sustain. Circ. Econ.*, vol. 40, nº 6, p. 609-624, juin 2022, doi: 10.1177/0734242X211033716.

[17] *An Introduction to Machine Learning*. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://books.google.com/books/about/An_Introduction_to_Machine_Learning.html?hl=fr&id=u8OWDwAAQBAJ

[18] « Handbook of Research on Heterogeneous Next Generation Networking - Google Books ». Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://www.google.sn/books/edition/Handbook_of_Research_on_Heterogeneous_

Ne/feeRCPDxXOsC?hl=fr&gbpv=1&dq=Hearst,+1998+Supervised+learning&pg=PA233&printsec=frontcover
[19] B. Mahesh, « Machine Learning Algorithms - A Review », *Int. J. Sci. Res. IJSR*, vol. 9, nº 1, p. 381-386, janv. 2020, doi: 10.21275/ART20203995.
[20] « Machine Learning Tutorial », GeeksforGeeks. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://www.geeksforgeeks.org/machine-learning/
[21] L. P. Kaelbling, M. L. Littman, et A. W. Moore, « Reinforcement Learning: A Survey », *J. Artif. Intell. Res.*, vol. 4, p. 237-285, mai 1996, doi: 10.1613/jair.301.
[22] R. S. Sutton et A. Barto, *Reinforcement learning: an introduction*, Nachdruck. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2014.
[23] W. D. Smart, « Practical Reinforcement Learning in Continuous Spaces ».
[24] « Chapitre I: Introduction | Introduction à l'apprentissage automatique ». Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://projeduc.github.io/intro_apprentissage_automatique/introduction.html
[25] L. Deng, « A tutorial survey of architectures, algorithms, and applications for deep learning », *APSIPA Trans. Signal Inf. Process.*, vol. 3, nº 1, 2014, doi: 10.1017/atsip.2013.9.
[26] H. Iness, « Un système intelligent d'aide a la décision pour la planification urbain. MASTER Filière : (Mathématiques/Informatique) Option : Système d'infomation ».
[27] *Computer Vision and Image Processing*. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://books.google.com/books/about/Computer_Vision_and_Image_Processing.html?hl=fr&id=eSu5I9pU3rUC
[28] M. Yang et G. Thung, « Classification of Trash for Recyclability Status ».
[29] Y. Wang et X. Zhang, « Autonomous garbage detection for intelligent urban management », *MATEC Web Conf.*, vol. 232, p. 01056, 2018, doi: 10.1051/matecconf/201823201056.
[30] « Waste classification using AutoEncoder network with integrated feature selection method in convolutional neural network models », *Measurement*, vol. 153, p. 107459, mars 2020, doi: 10.1016/j.measurement.2019.107459.
[31] P. F. Proença et P. Simões, « TACO: Trash Annotations in Context for Litter Detection », arXiv.org. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2003.06975v2
[32] « Deep learning-based waste detection in natural and urban environments », *Waste Manag.*, vol. 138, p. 274-284, févr. 2022, doi: 10.1016/j.wasman.2021.12.001.
[33] « Recycling waste classification using optimized convolutional neural network », *Resour. Conserv. Recycl.*, vol. 164, p. 105132, janv. 2021, doi: 10.1016/j.resconrec.2020.105132.
[34] Z. Nie, W. Duan, et X. Li, « Domestic garbage recognition and detection based on Faster R-CNN », *J. Phys. Conf. Ser.*, vol. 1738, nº 1, p. 012089, janv. 2021, doi: 10.1088/1742-6596/1738/1/012089.
[35] D. Bashkirova *et al.*, « VisDA 2022 Challenge: Domain Adaptation for Industrial Waste Sorting », in *NeurIPS 2022 Competition Track*, PMLR, août 2023, p. 104-118. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://proceedings.mlr.press/v220/bashkirova23a.html
[36] S. Casao *et al.*, « SpectralWaste Dataset: Multimodal Data for Waste Sorting Automation », arXiv.org. Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2403.18033v1
[37] W. Xia, Y. Jiang, X. Chen, et R. Zhao, « Application of machine learning algorithms in municipal solid waste management: A mini review », *Waste Manag. Res.*, vol. 40, nº 6, p. 609-624, juin 2022, doi: 10.1177/0734242X211033716.

[38] « Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review ». Consulté le: 2 février 2025. [En ligne]. Disponible sur: https://ieeexplore.ieee.org/abstract/document/10098596
[39] « 2006-aubert-vuibert-encyclopedia ».
[40] X. Jiang, A. Hadid, Y. Pang, E. Granger, et X. Feng, Éd., *Deep Learning in Object Detection and Recognition*. Singapore: Springer Singapore, 2019. doi: 10.1007/978-981-10-5152-4.
[41] H. Hakim et A. Fadhil, « Survey: Convolution Neural networks in Object Detection », *J. Phys. Conf. Ser.*, vol. 1804, nº 1, p. 012095, févr. 2021, doi: 10.1088/1742-6596/1804/1/012095.
[42] Dharmaraj, « Convolutional Neural Networks (CNN) — Architectures Explained », Medium. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://medium.com/@draj0718/convolutional-neural-networks-cnn-architectures-explained-716fb197b243
[43] M. Bosch, « Deep Learning for Robust Motion Segmentation with Non-Static Cameras », 22 février 2021, *arXiv*: arXiv:2102.10929. doi: 10.48550/arXiv.2102.10929.
[44] « Réseaux de neurones convolutif Cours RN Master IA-BDA 2O23 ».
[45] R. Delassus, « Apprentissage automatique pour la détection d'anomalies dans les données ouvertes: application à la cartographie ».
[46] O. Schwander, « Apprentissage statistique ».
[47] M. C. N. Nzatsi, « Optimisation des procédures d'imagerie à visée dosimétrique en radiothérapie interne vectorisée », phdthesis, Nantes Université, 2024. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://theses.hal.science/tel-04819093
[48] F. M. Shiri, T. Perumal, N. Mustapha, et R. Mohamed, « A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU », arXiv.org. Consulté le: 31 janvier 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2305.17473v3
[49] M. E. M. Chouini, « Détection d'objets basé Faster R-CNN », Working Paper, 2020. Consulté le: 3 février 2025. [En ligne]. Disponible sur: http://dspace.univ-guelma.dz/jspui/handle/123456789/10135
[50] Y. S. Obam, « Comprendre les Réseaux de Neurones Convolutifs (CNN) », Medium. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://yannicksergeobam.medium.com/comprendre-les-r%C3%A9seaux-de-neurones-convolutifs-cnn-d5f14d963714
[51] S. Ioffe et C. Szegedy, « Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift », arXiv.org. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/1502.03167v3
[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, « Dropout: A Simple Way to Prevent Neural Networks from Overfitting », *J. Mach. Learn. Res.*, vol. 15, nº 56, p. 1929-1958, 2014.
[53] A. Das, « Loss Functions in Deep Learning: An Overview », Analytics India Magazine. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://analyticsindiamag.com/deep-tech/loss-functions-in-deep-learning-an-overview/
[54] H. Bommana, « Loss Functions Explained », Medium. Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://medium.com/deep-learning-demystified/loss-functions-explained-3098e8ff2b27
[55] « RPubs - SCI1035-CH1 ». Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://rpubs.com/alexvezeau/SCI1035-CH1
[56] J. Koutnik, K. Greff, F. Gomez, et J. Schmidhuber, « A Clockwork RNN », in *International Conference on Machine Learning*, PMLR, juin 2014, p. 1863-1871. Consulté le: 31 janvier 2025. [En ligne]. Disponible sur: https://proceedings.mlr.press/v32/koutnik14.html

[57] « CS 230 - Aide-mémoire sur les réseaux neuronaux récurrents ». Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
[58] « Réseaux de neurones recurrents Cours RN Master IA-BDA 2O23 ».
[59] I. C. Kaadoud, « apprentissage de séquences et extraction de règles de réseaux récurrents: application au traçage de schémas techniques. ».
[60] « Qu'est-ce qu'un GAN ? – Définition des réseaux antagonistes génératifs – AWS », Amazon Web Services, Inc. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://aws.amazon.com/fr/what-is/gan/
[61] « Generative adversarial network: An overview of theory and applications », *Int. J. Inf. Manag. Data Insights*, vol. 1, nº 1, p. 100004, avr. 2021, doi: 10.1016/j.jjimei.2020.100004.
[62] D. Bank, N. Koenigstein, et R. Giryes, « Autoencoders », arXiv.org. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2003.05991v2
[63] J.-P. Briot, « Apprentissage profond et génération de musique », 2019.
[64] « Génération d'image faciale synthétique à base de points caractéristiques. »
[65] L. Hebbache, « Détection et segmentation de défauts pour les infrastructures de transport en utilisant l'imagerie par drones et l'apprentissage profond », masters, Université du Québec en Outaouais, Gatineau, 2023. Consulté le: 4 février 2025. [En ligne]. Disponible sur: https://di.uqo.ca/id/eprint/1494/
[66] A. Vaswani *et al.*, « Attention Is All You Need », arXiv.org. Consulté le: 1 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/1706.03762v7
[67] X. Wang, R. Girshick, A. Gupta, et K. He, « Non-local Neural Networks », in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA: IEEE, juin 2018, p. 7794-7803. doi: 10.1109/CVPR.2018.00813.
[68] A. Khan *et al.*, « A survey of the Vision Transformers and their CNN-Transformer based Variants », arXiv.org. Consulté le: 4 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2305.09880v4
[69] « Deep Learning pour l'analyse et la classification d'images. Sadio Alimata - Sadio Guindo 2024 ».
[70] H. Perreault, « Détection et segmentation des usagers de la route dans des images et vidéos », phd, Polytechnique Montréal, 2021. Consulté le: 3 février 2025. [En ligne]. Disponible sur: https://publications.polymtl.ca/9471/
[71] « Détection d'objets en temps réel : Entraînement de réseaux de neurones convolutifs sur images réelles et synthétiques ».
[72] « Mask R-CNN - Expliqué — Picsellia ». Consulté le: 4 février 2025. [En ligne]. Disponible sur: https://www.picsellia.fr/post/mask-r-cnn-explique
[73] B. Deguerre, « Object detection and traffic prediction using Deep Learning on compressed road images and videos ».
[74] « Précision et rappel — Wikipédia ». Consulté le: 4 février 2025. [En ligne]. Disponible sur: https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel
[75] « Precision-Recall curve », CloudFactory Computer Vision Wiki. Consulté le: 4 février 2025. [En ligne]. Disponible sur: https://wiki.cloudfactory.com/docs/mp-wiki/metrics/precision-recall-curve-and-auc-pr
[76] I. Talbi, « Pourquoi utiliser Python pour le deep learning ? », La revue IA. Consulté le: 5 février 2025. [En ligne]. Disponible sur: https://larevueia.fr/pourquoi-utiliser-python-pour-le-deep-learning/
[77] H. Iness, « Un système intelligent d'aide a la décision pour la planification urbain ».
[78] R. Kassel, « Keras : tout savoir sur l'API de Deep Learning », Formation Data Science | DataScientest.com. Consulté le: 5 février 2025. [En ligne]. Disponible sur: https://datascientest.com/keras

[79] Pankaj, « 10 Best Python Libraries for Machine Learning & AI: A Comprehensive Guide », Medium. Consulté le: 5 février 2025. [En ligne]. Disponible sur: https://medium.com/@pankaj_pandey/10-best-python-libraries-for-machine-learning-ai-a-comprehensive-guide-596719add91c
[80] « Application de Deep Learning pour un système de santé Intelligent dans un environnement cloud ».
[81] D. Bashkirova *et al.*, « ZeroWaste Dataset: Towards Deformable Object Segmentation in Cluttered Scenes », 16 mai 2022, *arXiv*: arXiv:2106.02740. doi: 10.48550/arXiv.2106.02740.
[82] F. Mesbah, « Détection d'objets par Deep Neural Network à l'aide du modèle YOLO en temps réel. », Working Paper, 2021. Consulté le: 4 février 2025. [En ligne]. Disponible sur: http://dspace.univ-guelma.dz/jspui/handle/123456789/11668
[83] « How to Code Non-Maximum Suppression (NMS) in Plain NumPy », Roboflow Blog. Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://blog.roboflow.com/how-to-code-non-maximum-suppression-nms-in-plain-numpy/
[84] Ultralytics, « YOLO11 🚀 NOUVEAU ». Consulté le: 5 février 2025. [En ligne]. Disponible sur: https://docs.ultralytics.com/fr/models/yolo11
[85] « YOLOv11 : aperçu des principales améliorations architecturales ». Consulté le: 13 février 2025. [En ligne]. Disponible sur: https://arxiv.org/html/2410.17725v1
[86] S. N. Rao, « YOLOv11 Explained: Next-Level Object Detection with Enhanced Speed and Accuracy », Medium. Consulté le: 27 février 2025. [En ligne]. Disponible sur: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71
[87] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, et S. Zagoruyko, « End-to-End Object Detection with Transformers », 28 mai 2020, *arXiv*: arXiv:2005.12872. doi: 10.48550/arXiv.2005.12872.
[88] Ultralytics, « RT-DETR (Transformateur de détection en temps réel) ». Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://docs.ultralytics.com/fr/models/rtdetr
[89] Y. Zhao *et al.*, « DETRs Beat YOLOs on Real-time Object Detection », arXiv.org. Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2304.08069v3
[90] N. Jegham, C. Y. Koh, M. Abdelatti, et A. Hendawi, « Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors », arXiv.org. Consulté le: 13 février 2025. [En ligne]. Disponible sur: https://arxiv.org/abs/2411.00201v1
[91] « (PDF) Development of a RT-DETR-based model for detecting two wheelers helmet », ResearchGate. Consulté le: 14 février 2025. [En ligne]. Disponible sur: https://www.researchgate.net/publication/380106738_Development_of_a_RT-DETR-based_model_for_detecting_two_wheelers_helmet

Annexes A YOLO11 :

- Installation des dépendances et importation des bibliothèque.

```
%%capture
!pip install "ultralytics<=8.3.40" supervision roboflow
!pip install roboflow
import ultralytics
from roboflow import Roboflow
```

- Téléchargement de l'ensemble de données spécifique à partir de Roboflow. Les données sont ensuite converties au format YOLOv11.

```
rf = Roboflow(api_key="UkyBT90fTdRbvWBFBE47")
project = rf.workspace("boston-university").project("zerowaste-tundo")
version = project.version(3)
dataset = version.download("yolov11")
```

- Les annotations polygonales sont converties en bounding boxes rectangulaires, ce qui est nécessaire pour l'entraînement du modèle YOLOv11.

```
x_coords = polygon[::2]  # Tous les éléments pairs (x)
y_coords = polygon[1::2]  # Tous les éléments impairs (y)
x_min = min(x_coords)
x_max = max(x_coords)
y_min = min(y_coords)
y_max = max(y_coords)
# Calcul des valeurs YOLO
x_center = (x_min + x_max) / 2
y_center = (y_min + y_max) / 2
width = x_max - x_min
height = y_max - y_min
return [x_center, y_center, width, height]
```

- Entrainement YOLOv11 est entraîné sur l'ensemble de données personnalisé. Les hyperparamètres tels que le nombre d'époques et la taille des images, etc. sont configurés.

```
!yolo task=detect mode=train model=yolo11s.pt
data=/content/zerowaste-3/data.yaml epochs=50 imgsz=640 plots=True
```

- Valider le modèle affiné

```
!yolo task=detect mode=val
model=/content/runs/detect/train/weights/best.pt
data={dataset.location}/data.yaml
```

Annexes B RT-DETR :

- Installation des dépendances

```
%%capture
!pip install -q
git+https://github.com/huggingface/transformers.git
!pip install -q git+https://github.com/roboflow/supervision.git
!pip install -q accelerate
!pip install -q roboflow
!pip install -q torchmetrics
!pip install -q "albumentations>=1.4.5"
```

- Nous chargeons le modèle avec les poids pre-entrainés de PekingU/rtdetr_r50vd_coco_o365

```
CHECKPOINT = "PekingU/rtdetr_r50vd_coco_o365"
DEVICE = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
Model=AutoModelForObjectDetection.from_pretrained(CHECKPOINT).to(D
EVICE)
processor = AutoImageProcessor.from_pretrained(CHECKPOINT)
```

- Chargement des données sous format COCO

```
rf = Roboflow(api_key="UkyBT90fTdRbvWBFBE47")
project = rf.workspace("boston-university").project("zerowaste-
tundo")
version = project.version(3)
dataset = version.download("coco")
```

- On applique les transformations avec AutoImageProcesseur

```
IMAGE_SIZE = 640
processor = AutoImageProcessor.from_pretrained(CHECKPOINT,
do_resize=True,
do_normalize=True,
size={"width": IMAGE_SIZE, "height": IMAGE_SIZE},)
```

- On rassemble toutes les hyperparamètres et on appelle la méthode train

```
trainer = Trainer(
model=model,
args=training_args,
train_dataset=pytorch_dataset_train,
eval_dataset=pytorch_dataset_valid,
tokenizer=processor,
data_collator=collate_fn,
compute_metrics=eval_compute_metrics_fn,)
trainer.train()
```

- Sauvegarde le modèle affiné

```
model.save_pretrained("/content/rt-detr/")
processor.save_pretrained("/content/rt-detr/")
```