



Université des Comores

Faculté des Sciences et Techniques
Département de Mathématiques,
Physique-Chimie & Informatique



Mémoire de Fin d'études

Présenté par :

SOIRFANE Abdallah Houmadi

Pour l'obtention du diplôme de :

Master de Recherche

Spécialité :

**Probabilités, Analyse,
Statistiques et Applications**

Modèles d'apprentissage profond pour la classification d'images :

Application à la classification des déchets ménagers et assimilés à Moroni

Encadré par :

Dr. HALASSI BACAR Abdoul-Hafar

Soutenu publiquement le 22 Novembre 2025

Devant le jury composé de :

Dr. ABDOULKARIM Mohamed Taki Institut Universitaire de Technologies Président

Dr. YOUSSEUF Ahamada Institut Universitaire de Technologies Examinateur

Dr. HALASSI BACAR Abdoul-Hafar Faculté des Sciences et Techniques Encadrant

Année Universitaire 2024 – 2025

Dédicace

À tout le monde !

Remerciements

Tout comme chaque travail est le fruit collectif de plusieurs esprits, celui-ci ne fait pas exception. De nombreuses personnes ont largement contribué à ce travail, et pour moi, il est plus que nécessaire d'écrire quelques mots pour vous dire, du fond de mon cœur, à quel point je suis reconnaissant.

Tout d'abord, je tiens à exprimer mes sincères remerciements à mon encadrant, **Dr Abdoul-Hafar HALASSI BACAR**. J'ai pris un immense plaisir à travailler avec vous et je vous remercie profondément pour vos précieux conseils, la qualité de votre encadrement, vos investissements, votre écoute et votre disponibilité. Je suis parfaitement conscient que ces mots ne sont pas à la hauteur de tout ce que vous avez fait pour moi tout au long de ce travail. Plus qu'un encadrant académique, vous êtes pour moi un véritable modèle : la curiosité intellectuelle, l'art de poser les bonnes questions et la volonté constante de chercher comment faire avancer la science. J'ai appris toutes ces valeurs à vos côtés, et je vous en suis infiniment reconnaissant.

Je tiens ensuite à remercier chaleureusement les **membres du jury** pour avoir accepté d'évaluer mon travail. Votre présence et vos efforts témoignent de votre engagement pour la rigueur académique et la promotion de la recherche scientifique. Vos observations, vos critiques constructives et vos suggestions ont permis d'enrichir ce mémoire, d'élargir ma réflexion et d'approfondir certains points essentiels. Je mesure l'honneur qui m'a été fait de voir mon travail examiné par des personnalités scientifiques de votre qualité, et je vous exprime toute ma gratitude.

Mes remerciements s'adressent également aux membres de l'équipe chargée de la collecte des données locales : **Yassimou Assulani, El-Yamine Haniou Omar, Nouzdine Riziki Baco, Soilihi Abdou Soilihi, Housseine Ibnou Ali, Zalhata Ahamada et Soirfane Abdallah Houmadi**. C'est grâce à votre bienveillance et à votre amour de la recherche scientifique que vous avez accepté de réaliser ce travail avec dévouement. J'écris ces quelques mots pour témoigner ma profonde reconnaissance et vous remercier du fond du cœur.

Je remercie également le **Laboratoire de Mathématiques, Statistiques, Informatique et Applications (LMSIA)** de la Faculté des Sciences et Techniques, ainsi que tous les enseignants du **Département de Mathématiques, Physique-Chimie & Informatique de Moroni** pour la qualité de leur enseignement et leur engagement dans la formation des étudiants.

Résumé

La gestion des déchets constitue un enjeu majeur, en particulier dans les pays en développement où les moyens de tri restent limités. Ce travail explore l'apport du *Deep Learning* dans la classification automatique des déchets ménagers à Moroni.

Nous avons utilisé plusieurs bases de données publiques *TrashNet*, *Waste Classification Data*, *Recycling and Household Waste Classification* ainsi que des données locales. Deux modèles de classification ont été développés : l'un de 3 806 525 paramètres (93 % de précision) et un second, plus léger, de 562 493 paramètres (92 %). Sur les données locales, les performances ont baissé, révélant un manque de généralisation. Une phase de *transfert learning* a alors été appliquée, permettant d'atteindre 86 % et 85 % respectivement.

Nous avons ensuite étudié la détection d'anomalies afin d'identifier les images ne correspondant à aucune classe connue. Parmi les approches testées seuil de confiance, autoencodeur variationnel et GAN le modèle d'autoencodeur variationnel a obtenu les meilleurs résultats, avec un score F1 de 79 % pour les données normales et 77 % pour les données anormales.

Ce travail met en avant le potentiel du Deep Learning pour le tri automatisé des déchets, tout en soulignant l'importance des données locales dans la performance des modèles.

Mots-clés : Deep Learning, Classification d'images, Détection d'anomalies, Autoencodeur variationnel, GAN, Moroni, Intelligence artificielle.

Abstract

Waste management is a major global challenge, particularly in developing countries where sorting methods remain limited. This work explores the contribution of *Deep Learning* to the automatic classification of household waste in Moroni.

Several public datasets *TrashNet*, *Waste Classification Data*, and *Recycling and Household Waste Classification* were used, along with locally collected data. Two image classification models were developed : one with 3,806,525 parameters achieving an overall accuracy of 93%, and a lighter version with 562,493 parameters reaching 92%. When evaluated on local data, both models showed a noticeable drop in performance, revealing a lack of generalization. A *transfer learning* phase was then applied, improving accuracy to 86% and 85% respectively.

We further investigated anomaly detection to identify images that do not belong to any known class. Among the tested approaches confidence thresholding, variational autoencoder, and GAN the variational autoencoder achieved the best results, with an F1-score of 79% for normal data and 77% for abnormal data.

This study highlights the potential of Deep Learning for automated waste sorting while emphasizing the crucial role of local data in enhancing model performance.

Keywords : Deep Learning, Image Classification, Anomaly Detection, Variational Autoencoder, GAN, Moroni, Artificial Intelligence.

Table des matières

Dédicace	i
Remerciements	ii
Résumé	iii
Introduction Générale	1
1 Gestion des déchets	3
1.1 Introduction	3
1.2 Caractéristiques des déchets : mondial et régional	3
1.2.1 Production mondiale des déchets	4
1.2.2 Production régionale des déchets : Afrique subsaharienne	4
1.3 Caractéristiques de déchets à Moroni	5
1.3.1 La production de déchets à Moroni	5
1.3.2 La composition des déchets à Moroni	5
1.3.3 La collecte et les infrastructures	6
1.4 Impacts environnementaux des déchets à travers le monde	6
1.4.1 Pollution	6
1.4.2 Impacts sur la biodiversité et les écosystèmes	7
1.4.3 Conséquences sanitaires pour les populations	8
1.5 Méthodes actuelles pour la réduction des déchets	8
1.5.1 Réduction à la source	8
1.5.2 Valorisation et recyclage des déchets	9
1.5.3 Valorisation énergétique	9
1.5.4 Freins et limites des méthodes actuelles	10
1.6 Conclusion	10
2 Introduction au Deep learning	11
2.1 Introduction	11
2.2 Historique	11
2.3 Rappel au machine learning	14
2.3.1 Apprentissage supervisé	14
2.3.2 Apprentissage non supervisé	14
2.3.3 Apprentissage par renforcement	15
2.3.4 Régression linéaire	15
2.3.5 Vectorisation des équations	16
2.4 Origine des réseaux de neurones artificiel	17

2.4.1	Fonctionnement des neurones biologiques	17
2.4.2	Fonctionnement des neurones artificielles	17
2.5	Étude du perceptron	18
2.5.1	Le perceptron à une seule couche	18
2.5.2	Perceptron multicouche	22
2.6	Les réseaux de neurones modernes	24
2.6.1	Réseaux de neurones convolutionnels	24
2.6.2	Auto-encodeurs	26
2.6.3	Réseaux antagonistes génératifs(GANs)	29
2.6.4	Réseaux neuronaux résiduels (ResNet)	31
2.7	Conclusion	32
3	Approche Proposée	33
3.1	Introduction	33
3.2	Revue de la littérature	33
3.2.1	Approches par transfert learning	33
3.2.2	Approches sans transfert learning	34
3.2.3	Limites des méthodes existantes	34
3.3	Méthodologie	34
3.3.1	Description de la base de données	34
3.3.2	Architecture du modèle	37
3.3.3	Stratégie d'entraînement	39
3.3.4	Évaluation des performances	40
3.4	Résultats	41
3.4.1	Résultats de l'entraînement	41
3.4.2	Résultats détaillés par classe	42
3.4.3	Matrice de confusion	42
3.4.4	Résumé global des performances	42
3.4.5	Évaluation sur les données locales	43
3.5	Conclusion	46
4	Améliorations du système	47
4.1	Introduction	47
4.2	Amélioration de la classification	47
4.3	Détection d'anomalies	48
4.4	Base de données utilisée	49
4.5	Détection par seuil de confiance	49
4.6	Détection par autoencodeur	50
4.6.1	Prétraitement des données	51
4.6.2	Modèle : <i>Bêta</i> -Autoencodeur Variationnel(β -VAE)	51
4.6.3	Entraînement du modèle	53
4.6.4	Résultats	53
4.6.5	Application	55
4.7	Détection par réseaux antagonistes génératifs(GANs)	56
4.7.1	Prétraitement des données	57
4.7.2	Construction du modèle	57
4.7.3	Stratégie d'entraînement	58
4.7.4	Résultats	59
4.8	Conclusion	60
Conclusion Générale		60

Table des figures

2.1	Neurone biologique	17
3.1	Répartition en pourcentage des images par base de données.	35
3.2	Répartition en pourcentage des images par classe.	35
3.3	Exemples d'images d'entraînement	36
3.4	Nombre d'images utilisées par classe.	37
3.5	Répartition en pourcentage des images par classe.	37
3.6	Courbe d'apprentissage du Modèle 1 (Flatten).	41
3.7	Courbe d'apprentissage du Modèle 2 (AvgPooling).	42
3.8	Comparaison des matrices de confusion des deux modèles sur les données de test.	43
3.9	Courbe d'apprentissage du modèle 1 obtenu par transfert learning.	44
3.10	Courbe d'apprentissage du modèle 2 obtenu par transfert learning.	44
3.11	Comparaison des matrices de confusion des deux modèles sur les 30 % des données locales.	45
3.12	Pourcentage d'images correctement classées et mal classées pour les modèles avec et sans transfert learning.	45
4.1	Évolution du taux de reconnaissance des images connues et inconnues.	50
4.2	Courbe d'apprentissage du modèle β -VAE sur les ensembles d'entraînement et de test.	54
4.3	Exemples d'images originales et reconstruites sur les données d'entraînement.	54
4.4	Exemples d'images originales et reconstruites sur les données de test.	54
4.5	Évolution du nombre de détections (à gauche) et des métriques de performance (à droite) selon le seuil d'anomalie.	55
4.6	Matrice de confusion pour la détection d'anomalies par le modèle β -VAE.	56
4.7	Évolution des pertes du générateur et du discriminateur, ainsi que du ratio G/D au cours de l'entraînement.	59
4.8	Exemples d'images générées après entraînement du modèle GAN.	59

Liste des tableaux

3.1	Répartition des images sélectionnées par classe et par base de données	35
3.2	Nombre d'images réservées pour chaque classe lors de la validation finale.	35
3.3	Paramètres d'augmentation des données et leurs descriptions.	37
3.4	Architecture détaillée du modèle proposé	38
3.5	Résumé des hyperparamètres et des stratégies d'entraînement utilisées	39
3.6	Coefficients de pondération des classes.	40
3.7	Résultats détaillés par classe pour les deux modèles (jeu de test).	42
3.8	Résumé comparatif des deux modèles.	43
3.9	Résultats détaillés par classe pour les deux modèles (sur les 30 % des données locales).	45
4.1	Résultats de l'évaluation par seuil de confiance pour les différents modèles	50
4.2	Bloc résiduel standard utilisé dans l'encodeur et le discriminateur.	52
4.3	Bloc résiduel avec suréchantillonnage utilisé dans le décodeur.	52
4.4	Architecture de l'encodeur du modèle β -VAE.	52
4.5	Architecture du décodeur du modèle β -VAE.	53
4.6	Résultats de la détection d'anomalies par le modèle β -VAE (seuil optimal = 0.01607).	55
4.7	Architecture du générateur du modèle GAN.	57
4.8	Architecture du discriminateur du modèle GAN.	58
4.9	Résumé des paramètres et stratégies d'entraînement du GAN	58

Introduction Générale

Dans un contexte mondial en constante évolution, la gestion des déchets s'impose comme une problématique cruciale, nécessitant des solutions innovantes et rapides. La ville de Moroni, avec une population estimée à environ 82 649 habitants en 2024 et un taux de croissance de 3,33 %¹, fait face à des défis particuliers liés à l'urbanisation rapide et à une architecture urbaine souvent inadaptée. Cette situation rend la gestion des déchets encore plus complexe et difficile à maîtriser.

Il est important de rappeler que tous les éléments constitutifs des déchets ne sont pas forcément destinés à être jetés. En effet, certains peuvent être réutilisés comme matières premières dans d'autres secteurs, contribuant ainsi à réduire le gaspillage des ressources naturelles et à promouvoir une économie circulaire. Cette perspective montre que la gestion des déchets ne se limite pas seulement à l'élimination, mais qu'elle peut aussi devenir une opportunité pour valoriser les ressources disponibles.

Mathématiquement, la gestion efficace des déchets est souvent négligée, en partie parce qu'elle est perçue comme une tâche secondaire ou peu valorisante. Le manque d'infrastructures adaptées et de moyens modernes de tri accentue encore ces difficultés. C'est dans ce contexte que s'inscrit notre projet : l'utilisation des outils d'intelligence artificielle pour automatiser le tri des déchets, offrant ainsi une solution rapide, efficace et mieux adaptée aux besoins réels de la population.

En effet, avec le développement d'outils puissants en intelligence artificielle, notamment dans le domaine de la classification d'images, il nous paraît opportun de développer un projet basé sur l'intelligence artificielle appliquée à la reconnaissance visuelle. L'objectif principal est de concevoir un modèle capable d'identifier automatiquement différentes catégories de déchets, en s'appuyant à la fois sur des données issues de bases ouvertes et sur des collectes locales réalisées à Moroni. Un tel modèle pourrait rendre le tri plus efficace, réduire les erreurs humaines et optimiser les processus de recyclage et de valorisation des déchets. Il s'agit donc d'un travail qui combine une avancée technologique et une réponse à un besoin concret dans le contexte comorien.

Ainsi, ce projet se situe à la croisée de la recherche académique et des enjeux de développement durable. S'il est d'abord pensé pour la gestion des déchets, il ouvre également la voie à d'autres applications dans des domaines où la classification d'images joue un rôle essentiel, tels que la santé, l'agriculture ou encore la sécurité. Ce mémoire témoigne donc d'une volonté de mettre l'intelligence artificielle au service de la société, en explorant des solutions innovantes et adaptées à nos réalités locales.

1. Revue de la population mondiale

La suite de ce travail est composée de quatre chapitres structurés comme suit :

Le premier chapitre, *Gestion des déchets*, présente le contexte général de notre travail. Nous y décrivons les pratiques actuelles de gestion des déchets, les infrastructures existantes, ainsi que leurs impacts environnementaux et sanitaires. Ce chapitre met en évidence les difficultés rencontrées et justifie la nécessité d'intégrer des solutions technologiques innovantes.

Le deuxième chapitre, *Introduction au Deep Learning*, introduit les bases de cette technologie. Nous retracions son évolution historique et ses liens avec le *machine learning*, puis nous présentons l'origine des réseaux de neurones en insistant sur le perceptron et le perceptron multicouche. Ce chapitre fournit ainsi une base solide pour comprendre les méthodes modernes utilisées en classification d'images.

Le troisième chapitre, *Approche proposée*, détaille la première démarche suivie dans ce travail. Nous présentons la base de données utilisée, l'architecture des modèles proposés ainsi que la stratégie d'entraînement mise en œuvre. Ce chapitre inclut également les résultats obtenus à travers différents tests, ainsi qu'une comparaison des performances de notre approche avec celles des modèles existants.

Enfin, le quatrième chapitre, *Améliorations du système*, discute des limites identifiées dans le *chapitre 3* et propose des pistes d'amélioration. Nous y explorons différentes approches pour rendre le système plus adapté aux conditions réelles, ouvrant ainsi la voie à de futures perspectives de recherche et d'application.

Gestion des déchets

1.1 Introduction

La gestion des déchets solides est devenue l'un des défis environnementaux et sanitaires les plus pressants à l'échelle mondiale. L'urbanisation rapide, la croissance démographique et les modes de consommation modernes ont entraîné une augmentation significative de la production de déchets, obligeant les pays à adopter des stratégies plus innovantes, durables et technologiquement avancées pour leur traitement.

Ce chapitre s'intéresse d'abord à la qualité et à la nature des déchets produits à travers le monde, en mettant en évidence les particularités de la région subsaharienne, où la croissance urbaine et la faiblesse des infrastructures accentuent la complexité du problème. Nous examinerons ensuite les impacts environnementaux majeurs liés à la mauvaise gestion des déchets, tels que la pollution des sols, des eaux et de l'air, ainsi que leurs conséquences sanitaires directes pour les populations.

Dans ce chapitre, nous présentons également le cas des **Comores**, et plus particulièrement celui de la ville de **Moroni**, où l'accumulation des déchets dans les zones urbaines constitue une menace croissante pour l'environnement et la qualité de vie. Enfin, nous présenterons les principales méthodes déjà employées pour réduire ces déchets, tout en identifiant les freins structurels, économiques et sociaux qui limitent leur efficacité.

En somme, ce chapitre établit un état des lieux essentiel pour comprendre l'ampleur de la problématique et les défis spécifiques liés à la gestion des déchets aux Comores, préparant ainsi le terrain pour les solutions innovantes que nous proposerons dans les chapitres suivants.

1.2 Caractéristiques des déchets : mondial et régional

La production de déchets solides constitue aujourd'hui un indicateur majeur des dynamiques économiques, sociales et environnementales des sociétés modernes. À l'échelle mondiale, la quantité et la composition des déchets varient fortement selon le niveau de développement et les modes de consommation. L'analyse de ces caractéristiques, tant au niveau global qu'au niveau régional, permet de mieux comprendre les défis spécifiques liés à leur gestion et d'identifier les solutions adaptées aux différents contextes.

1.2.1 Production mondiale des déchets

En 2023, la Banque mondiale a publié un rapport intitulé *What a Waste 2.0*,^[1] selon lequel environ 2,24 milliards de tonnes de déchets municipaux solides ont été générés à travers le monde. En l'absence d'interventions majeures, cette quantité pourrait atteindre 3,4 milliards de tonnes d'ici 2050, soit une augmentation de près de 70 % par rapport aux niveaux actuels. Ces chiffres illustrent l'ampleur croissante du problème et soulignent l'urgence d'adopter des stratégies de gestion plus durables et inclusives.

La production mondiale de déchets n'est toutefois pas uniformément répartie. Les pays à revenu élevé, qui représentent environ 16 % de la population mondiale, génèrent près de 34 % des déchets totaux.^[1] En revanche, les pays à faible revenu ne produisent qu'une faible part des déchets globaux, mais connaissent une croissance rapide de la production en raison de l'urbanisation accélérée et de l'augmentation des niveaux de vie.

Un rapport conjoint du Programme des Nations Unies pour l'Environnement (PNUE) et de l'Association Internationale de Gestion des Déchets (ISWA), intitulé *Global Waste Management Outlook*,^[2] confirme ces tendances. Selon ce rapport, la gestion inadéquate des déchets représente une menace croissante pour la santé publique et l'environnement, particulièrement dans les pays à revenu faible et intermédiaire où les infrastructures de collecte et de traitement restent limitées.

La composition des déchets reflète également les disparités économiques. Dans les pays développés, une grande part des déchets provient des plastiques, du papier, du carton et des produits électroniques, alors que dans les pays en développement, les déchets organiques dominent, représentant parfois plus de 50 % du total.^[3] Ces différences qualitatives soulignent la nécessité de politiques adaptées aux réalités locales, intégrant à la fois la réduction, la valorisation et le recyclage.

1.2.2 Production régionale des déchets : Afrique subsaharienne

L'Afrique subsaharienne est aujourd'hui l'une des régions du monde où la gestion des déchets constitue un défi majeur. Selon le même rapport de la Banque mondiale,^[1] la région produit environ 174 millions de tonnes de déchets municipaux solides par an, un chiffre appelé à tripler pour atteindre près de 516 millions de tonnes d'ici 2050 si aucune mesure structurelle n'est mise en place. Cette augmentation est directement liée à la croissance démographique rapide et à l'urbanisation, la population urbaine devant doubler d'ici le milieu du siècle.

La production par habitant demeure relativement faible comparée aux pays industrialisés, avec une moyenne de 0,46 kg par jour en Afrique subsaharienne, contre 0,74 kg au niveau mondial.^[1] Cependant, la croissance rapide de la population transforme cette faible production individuelle en un volume global considérable, exerçant une pression croissante sur des infrastructures déjà limitées.

Le *Africa Waste Management Outlook*, publié par le PNUE,^[4] met en évidence que plus de 90 % des déchets générés dans la région sont encore déposés dans des décharges non contrôlées ou à ciel ouvert. De plus, près de 19 pays africains ne disposent d'aucune stratégie nationale de gestion des déchets, ce qui accentue les disparités entre zones rurales et urbaines.

La composition des déchets en Afrique subsaharienne est dominée par la fraction organique, qui représente en moyenne 57 % du total, suivie par les plastiques (13 %) et les papiers-cartons (9 %).^[4] Cette forte proportion de matières biodégradables pourrait constituer une opportunité

pour le compostage et la valorisation énergétique, mais le manque d'infrastructures techniques et financières empêche une exploitation optimale de ce potentiel.

1.3 Caractéristiques de déchets à Moroni

La gestion des déchets solides aux Comores, et en particulier dans la capitale Moroni, constitue un défi majeur à la croisée des enjeux environnementaux, sanitaires et socio-économiques. Comme dans de nombreux pays insulaires en développement, l'urbanisation rapide et la croissance démographique s'accompagnent d'une production croissante de déchets, alors que les infrastructures de collecte et de traitement restent très limitées.^[5] Cette situation engendre une accumulation visible des ordures dans les espaces urbains, avec des impacts directs sur la qualité de vie des habitants.

1.3.1 La production de déchets à Moroni

Les études récentes menées à Moroni indiquent une production moyenne pondérée de *0,27 kg de déchets par habitant et par jour*.^[5] Ce niveau est sensiblement inférieur à la moyenne des grandes villes africaines, mais il reste préoccupant dans le contexte insulaire des Comores où les capacités de gestion demeurent limitées.

La production varie fortement selon les quartiers. Les enquêtes de caractérisation montrent que les *biodéchets constituent la fraction la plus importante*, représentant de loin la majorité des déchets ménagers, suivis par les papiers-cartons, les plastiques, puis les verres, textiles, couches et métaux en proportions moindres. Par exemple, dans le quartier de Zilimadjou, la production hebdomadaire a été estimée à près de 489 kg de biodéchets, contre environ 44 kg de papiers-cartons et 93 kg de plastiques.^[5]

Des relevés spécifiques au marché de Volo-Volo, principal centre commercial de la ville, confirment ces tendances : au cours de la première semaine d'étude, *145,2 kg de biodéchets et 205,4 kg de papiers-cartons* ont été collectés, tandis que la fraction plastique atteignait *495,4 kg*.^[5] La semaine suivante, la quantité de biodéchets a augmenté à *243,5 kg*, tandis que les papiers-cartons ont atteint *747,4 kg*. Ces chiffres traduisent non seulement la dominance de la matière organique, mais également la place croissante des emballages plastiques et papiers-cartons dans les déchets urbains.

1.3.2 La composition des déchets à Moroni

La caractérisation des déchets solides ménagers à Moroni met en évidence une forte hétérogénéité selon les quartiers, mais confirme la prédominance des déchets organiques. En moyenne, les biodéchets représentent environ 64 % du total, suivis par les plastiques (près de 11 %), puis les papiers et cartons (environ 3 %). Les autres catégories, telles que les textiles, les métaux, le verre et les couches, apparaissent en proportions plus réduites mais contribuent néanmoins à la complexité de la gestion.^[5]

Un cas particulier est celui du marché de Volo-Volo, principal centre commercial de la capitale. Les enquêtes menées en 2019 indiquent une composition très spécifique des déchets produits : *41 % de biodéchets, 37 % de produits carnés, 18 % de cartons et seulement 4 % de plastiques*. Les autres types de déchets sont quasiment inexistant. Cette structure reflète directement la nature des acti-

vités du marché, centrées sur la vente de denrées alimentaires et de produits emballés.

Ces résultats soulignent deux constats essentiels. D'une part, la proportion élevée de biodéchets dans l'ensemble de la ville révèle un fort potentiel pour des stratégies de valorisation organique, telles que le compostage ou la méthanisation. D'autre part, la place importante des plastiques et des cartons, particulièrement dans les zones commerciales, met en évidence la nécessité de développer des filières locales de tri et de recyclage afin de limiter leur impact environnemental.

1.3.3 La collecte et les infrastructures

Malgré les engagements politiques pris depuis plusieurs décennies en faveur de la protection de l'environnement, la gestion des déchets à Moroni reste marquée par de profondes insuffisances. Dès 1994, une loi-cadre a inscrit la préservation des ressources et la gestion durable des déchets dans la stratégie nationale de développement. Divers financements ont été mobilisés par les partenaires internationaux (Nations Unies, Union européenne, coopération bilatérale), mais les solutions proposées n'ont jamais été mises en œuvre de manière efficace.^[5]

En pratique, le système de collecte demeure rudimentaire et inadapté. Jusqu'en 2007, les déchets étaient acheminés vers le site de Sélea, situé à une dizaine de kilomètres de Moroni, avant que celui-ci ne soit fermé pour cause de saturation et de mauvaise gestion. Depuis, les déchets sont provisoirement stockés dans des conteneurs placés sur l'ancienne piste de l'aéroport de Moroni, puis transportés vers le site de Bahani. Toutefois, cette organisation reste aléatoire et le planning de ramassage n'est pas respecté, entraînant l'apparition régulière de dépôts sauvages dans les rues, les ravines et le littoral.

Le taux global de collecte est estimé à seulement 30 à 35 % des déchets produits. La majorité des déchets ménagers échappent donc au système formel et se retrouvent abandonnés dans l'espace public ou incinérés à ciel ouvert, sans tri ni traitement préalable. Cette situation engendre des nuisances sanitaires et environnementales importantes, allant de la pollution de l'air à la contamination des sols et des eaux souterraines.^[5]

1.4 Impacts environnementaux des déchets à travers le monde

La croissance démographique, l'urbanisation rapide et la transformation des modes de consommation ont entraîné une augmentation massive de la production mondiale de déchets solides, dépassant 2,24 milliards de tonnes par an en 2023.^[1] Lorsque ces déchets ne sont pas correctement collectés et traités, ils génèrent des effets néfastes sur les sols, les eaux, l'air et la biodiversité, tout en accentuant les risques sanitaires pour les populations. Cette section s'intéresse aux principaux impacts environnementaux observés à travers le monde, afin de mieux situer les enjeux globaux auxquels font face les pays en développement comme les Comores.

1.4.1 Pollution

Pollution des sols et des eaux

La gestion inadéquate des déchets solides, notamment via les dépotoirs sauvages et les décharges non contrôlées, constitue une source majeure de pollution des sols et des ressources hydriques. Les lixiviats produits par la décomposition des déchets contiennent souvent des métaux lourds, des polluants organiques et des micro-organismes pathogènes qui s'infiltrent dans les sols et contaminent

les nappes phréatiques.^[6] Cette contamination représente un risque direct pour la qualité des terres agricoles et la disponibilité d'eau potable pour les populations riveraines.

Des études en Afrique et en Asie confirment ces risques. À Lagos, au Nigéria, les lixiviats des décharges ouvertes ont entraîné une pollution significative des eaux souterraines par le plomb, le cadmium et le nickel, dépassant les seuils recommandés par l'OMS.^[7] En Inde, des travaux similaires ont montré une forte contamination des sols et des eaux autour des décharges, avec des concentrations de métaux lourds supérieures aux normes environnementales.^[6]

En Amérique latine, la Banque mondiale estime que plus de 40 % des sites d'enfouissement ne disposent d'aucune infrastructure de confinement, accentuant la contamination diffuse des sols et des nappes phréatiques.^[1] Le ruissellement des eaux pluviales transporte en outre des déchets plastiques et organiques vers les rivières et les zones côtières, provoquant une dégradation des écosystèmes aquatiques et une perte de biodiversité.

Pollution atmosphérique

L'incinération à ciel ouvert des déchets solides, une pratique encore très répandue dans de nombreux pays en développement, génère une quantité importante de polluants atmosphériques et contribue de manière significative aux émissions de gaz à effet de serre. Ce processus libère des particules fines (PM_{2.5} et PM₁₀), des oxydes d'azote, du dioxyde de soufre, ainsi que des hydrocarbures aromatiques polycycliques, reconnus pour leurs effets néfastes sur la santé humaine et l'environnement.^[8]

Au-delà des polluants locaux, la combustion incontrôlée des déchets est également une source majeure de dioxyde de carbone (CO₂) et de méthane (CH₄), deux gaz responsables du réchauffement climatique. Une étude menée au Nigéria a montré que la combustion non réglementée des déchets contribue fortement à la dégradation de la qualité de l'air et à l'augmentation des gaz à effet de serre régionaux, aggravant le changement climatique.^[9] De plus, les feux de décharge mal contrôlés, fréquents dans les zones urbaines denses, libèrent également du carbone noir, un polluant à fort potentiel de réchauffement planétaire.^[10]

Selon la Banque mondiale, près de 40 % des déchets mondiaux sont encore éliminés par brûlage à ciel ouvert, une pratique qui accentue non seulement la pollution atmosphérique locale mais aussi les perturbations climatiques globales.^[1] La réduction progressive de cette pratique est donc essentielle pour atténuer les impacts sanitaires et climatiques liés à la mauvaise gestion des déchets.

1.4.2 Impacts sur la biodiversité et les écosystèmes

La mauvaise gestion des déchets exerce une pression considérable sur les écosystèmes terrestres et aquatiques. Les lixiviats issus des dépotoirs non contrôlés s'infiltrent dans les sols et les nappes phréatiques, libérant des substances toxiques qui affectent la fertilité des terres et réduisent la productivité agricole. Des études récentes ont montré que ces infiltrations peuvent altérer la composition des sols et entraîner une perte de biodiversité locale.^[11]

Les écosystèmes marins et d'eau douce sont particulièrement vulnérables à la pollution plastique. Les déchets plastiques se fragmentent en microplastiques et nanoplastiques, qui sont ingérés par de nombreuses espèces aquatiques, provoquant des effets toxiques allant de la réduction de la croissance à des perturbations physiologiques graves.^[12] Cette pollution affecte non seulement la faune

aquatique mais compromet aussi les chaînes alimentaires dont dépendent des millions de personnes.

Une autre étude souligne que la pollution plastique marine a déjà entraîné un déclin significatif de certaines populations d'espèces, en particulier les oiseaux marins et les tortues, souvent piégés ou étouffés par les déchets flottants.^[13] Dans les environnements côtiers, ces accumulations modifient les habitats et perturbent les écosystèmes entiers, compromettant la résilience des zones humides et des récifs coralliens.

1.4.3 Conséquences sanitaires pour les populations

La gestion inadéquate des déchets solides représente une menace majeure pour la santé publique, en particulier dans les zones urbaines des pays en développement. L'accumulation de déchets dans les rues et les dépotoirs sauvages favorise la prolifération de vecteurs de maladies tels que les moustiques, les mouches et les rongeurs, responsables de pathologies comme la dengue, le paludisme et la leptospirose.^[14]

L'incinération à ciel ouvert, largement pratiquée faute d'infrastructures adaptées, expose directement les populations aux fumées contenant des particules fines, des dioxines et des hydrocarbures aromatiques polycycliques. Ces substances sont reconnues pour provoquer des maladies respiratoires chroniques, des troubles cardiovasculaires et, à long terme, un risque accru de cancers.^[15]

Par ailleurs, la contamination des eaux de surface et des nappes phréatiques par les lixiviats des décharges constitue une autre voie d'exposition. Des études ont montré une corrélation entre la proximité des décharges à ciel ouvert et l'augmentation des cas de maladies gastro-intestinales, de typhoïde et de choléra.^[16] L'impact est particulièrement préoccupant pour les populations vulnérables, notamment les enfants, qui sont davantage exposés aux agents pathogènes et aux polluants chimiques.

1.5 Méthodes actuelles pour la réduction des déchets

Face à l'augmentation constante de la production mondiale de déchets, de nombreuses stratégies ont été développées pour limiter leur impact environnemental et sanitaire. Ces méthodes reposent à la fois sur la réduction à la source, le tri et la valorisation des déchets, ainsi que sur des approches plus récentes de transformation énergétique. Cette section présente les principales pratiques mises en œuvre à travers le monde, tout en soulignant leurs limites et leurs perspectives d'adaptation aux contextes locaux.

1.5.1 Réduction à la source

La réduction à la source constitue la première étape dans toute stratégie de gestion durable des déchets. Elle consiste à limiter la production de déchets dès la conception des produits, en favorisant des pratiques telles que l'éco-conception, l'allongement de la durée de vie des biens et la substitution des matériaux difficiles à recycler par des alternatives plus durables.^[17]

Un exemple marquant est la politique de réduction des plastiques à usage unique, mise en œuvre dans de nombreux pays. L'Union européenne a interdit depuis 2021 plusieurs articles plastiques à usage unique, comme les pailles et les couverts, afin de réduire la pollution plastique marine.^[18] En Afrique, des pays comme le Rwanda et le Kenya ont adopté des interdictions strictes des sacs

plastiques, démontrant l'efficacité des mesures réglementaires pour limiter les déchets dès leur production.^[19]

La consommation durable repose également sur un changement de comportement des citoyens et des entreprises. La promotion de la réutilisation, la réduction du gaspillage alimentaire et l'encouragement aux circuits courts permettent de limiter la production de déchets tout en valorisant les ressources locales.^[20] Ces approches contribuent non seulement à diminuer la quantité de déchets générés, mais aussi à renforcer la durabilité économique et sociale des systèmes de production et de consommation.

1.5.2 Valorisation et recyclage des déchets

La valorisation et le recyclage des déchets constituent des piliers essentiels d'une gestion durable, permettant de transformer des matières considérées comme résiduelles en ressources réutilisables. Cette approche contribue à réduire la pression sur les ressources naturelles, à limiter la quantité de déchets envoyés en décharge et à créer de nouvelles opportunités économiques et sociales.^[21]

Dans les pays industrialisés, le recyclage des matières premières secondaires (plastiques, métaux, papiers-cartons, verre) est soutenu par des infrastructures performantes et des marchés organisés. Par exemple, l'Allemagne et la Suède figurent parmi les leaders mondiaux avec des taux de recyclage des déchets municipaux supérieurs à 50 %.^[22] En revanche, dans les pays en développement, le recyclage reste souvent dominé par le secteur informel, où des milliers de récupérateurs jouent un rôle central dans la collecte et la valorisation, malgré des conditions de travail précaires et un manque de reconnaissance institutionnelle.^[23]

Au-delà du recyclage classique, de nouvelles formes de valorisation se développent. Le recyclage chimique des plastiques, le réemploi des matériaux de construction et la valorisation énergétique des résidus non recyclables sont des pistes prometteuses pour renforcer l'économie circulaire.^[24] Toutefois, ces solutions nécessitent des investissements technologiques et un cadre réglementaire adapté pour éviter les effets négatifs, tels que les émissions polluantes lors de procédés mal contrôlés.

1.5.3 Valorisation énergétique

La valorisation énergétique des déchets consiste à transformer la fraction non recyclable en source d'énergie, principalement sous forme de chaleur, d'électricité ou de biogaz. Cette approche présente un double avantage : elle permet de réduire le volume de déchets destinés à l'enfouissement tout en contribuant à la diversification du mix énergétique.^[5]

Aux Comores, où les besoins énergétiques demeurent largement insatisfaits et où la dépendance aux importations de combustibles fossiles est forte, cette stratégie revêt une importance particulière. Les travaux de caractérisation physico-chimique des déchets ménagers de Moroni ont montré un potentiel énergétique non négligeable, notamment grâce à la proportion élevée de biodéchets et de matières combustibles.^[25] Ces résultats confirment que les déchets solides peuvent constituer une ressource alternative pour la production d'énergie, réduisant à la fois la pression environnementale liée aux dépotoirs sauvages et la vulnérabilité énergétique du pays.

L'expérience d'autres pays africains montre également le potentiel de cette approche. Au Nigeria, une étude récente a démontré que la mise en place d'unités de production d'énergie à partir des

déchets municipaux pourrait contribuer de manière significative à l'approvisionnement énergétique, tout en réduisant les impacts environnementaux liés aux dépotoirs non contrôlés.^[26] Ce type de modèle pourrait inspirer des solutions adaptées au contexte comorien, à condition d'intégrer les spécificités locales et les contraintes économiques.

1.5.4 Freins et limites des méthodes actuelles

Malgré les avancées importants dans la mise en place de stratégies de gestion durable, de nombreux obstacles persistent et limitent l'efficacité des méthodes actuelles. Le premier frein majeur concerne le tri des déchets, qui reste souvent insuffisant ou mal appliqué. Dans de nombreux contextes, notamment en Afrique subsaharienne, l'absence de systèmes de collecte sélective et le manque de sensibilisation de la population conduisent à un mélange des fractions organiques, plastiques et métalliques, réduisant fortement les possibilités de recyclage et de valorisation.^[14]

À cela s'ajoutent des contraintes techniques et économiques. La mise en place d'infrastructures modernes de tri, de compostage ou de valorisation énergétique nécessite des investissements considérables, souvent inaccessibles aux pays en développement.^[27] De plus, le manque d'entretien des équipements et l'absence de filières locales viables de recyclage entraînent une dépendance coûteuse aux marchés internationaux.

Sur le plan institutionnel, la faiblesse du cadre réglementaire et l'absence de politiques publiques incitatives limitent l'efficacité des initiatives existantes. Les conflits de compétences entre autorités locales et nationales, combinés à une gouvernance fragile, ralentissent la mise en œuvre des programmes. Enfin, les barrières socioculturelles telles que le manque de sensibilisation, la faible implication citoyenne et la perception des déchets comme une simple nuisance plutôt qu'une ressource constituent un obstacle supplémentaire à l'adoption de pratiques durables.^[28]

1.6 Conclusion

Dans ce chapitre, nous avons étudié la question des déchets solides en partant de la situation mondiale pour arriver au cas particulier des Comores, et plus précisément de la ville de Moroni. Nous avons montré que la production de déchets augmente rapidement, surtout dans les pays en développement où les infrastructures sont limitées et où l'urbanisation rend le problème encore plus complexe.

Nous avons aussi vu que la mauvaise gestion des déchets entraîne de nombreux impacts négatifs, comme la pollution de l'air, de l'eau et des sols, la dégradation des écosystèmes et des risques pour la santé des populations. Le cas de Moroni illustre bien ces difficultés, avec une forte proportion de biodéchets, une collecte incomplète et la présence de dépotoirs sauvages.

Enfin, nous avons présenté les méthodes actuellement utilisées pour réduire ou valoriser les déchets, comme la réduction à la source, le tri, le recyclage, le compostage et la valorisation énergétique. Si ces solutions offrent des perspectives intéressantes, elles restent limitées par le manque de moyens techniques et financiers, ainsi que par une organisation insuffisante.

En résumé, ce chapitre a permis de mettre en évidence les défis mais aussi les opportunités liés à la gestion des déchets. Ces éléments ouvrent la voie à une réflexion sur des solutions adaptées au contexte comorien, qui seront développées dans les chapitres suivants.

Introduction au Deep learning

2.1 Introduction

Le Deep Learning représente aujourd’hui un des piliers les plus prometteurs de l’intelligence artificielle, permettant des avancées notables dans le traitement des données complexes, notamment en reconnaissance d’images, traitement du langage naturel, et bien plus encore. Ce chapitre commence par un aperçu historique du Deep Learning, retracant son évolution depuis ses origines, où il s’est distingué des méthodes traditionnelles grâce à l’amélioration exponentielle des capacités de calcul et l’accessibilité croissante à de vastes ensembles de données.

Le chapitre s’intéresse également à la place du Deep Learning au sein du Machine Learning. Un rappel des concepts essentiels du Machine Learning est proposé pour mieux situer le Deep Learning dans ce cadre plus large, expliquant comment ces modèles avancés se sont construits sur les bases existantes. Nous aborderons aussi l’origine des réseaux de neurones, en examinant leur structure inspirée du fonctionnement du cerveau humain, avec une attention particulière au perceptron et au perceptron multicouches, qui constituent le socle des architectures modernes.

En somme, ce chapitre offre une base solide pour comprendre les mécanismes et les applications du Deep Learning, préparant ainsi le lecteur à plonger dans les techniques plus avancées qui seront explorées dans les chapitres suivants.

2.2 Historique

Tout a commencé en 1943, lorsque Warren McCulloch, un neurophysiologiste, et Walter Pitts, un logicien, ont proposé un modèle mathématique du neurone biologique dans leur article fondateur intitulé *A Logical Calculus of the Ideas Immanent in Nervous Activity*.^[29] Cet article a suscité un vif intérêt, certains allant jusqu’à espérer la création imminente de machines intelligentes. Toutefois, ce modèle ne prévoyait aucun mécanisme d’apprentissage.

En 1958, Frank Rosenblatt, un psychologue américain, introduit le *perceptron*, un modèle de neurone artificiel doté d’un algorithme d’apprentissage supervisé, dans son article *The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain*.^[30] Malgré l’enthousiasme initial, ses limitations notamment son incapacité à résoudre des problèmes non linéaires ont mené à une désaffection pour les recherches en intelligence artificielle, marquant ainsi le premier « hiver de l’IA ».

Un tournant majeur s'opère en 1986 avec la publication de l'article *Learning Representations by Back-Propagating Errors* de Rumelhart, Hinton et Williams.^[31] Ce travail introduit l'algorithme de rétropropagation, permettant l'entraînement efficace des perceptrons multicouches et la résolution de problèmes non linéaires complexes. Il constitue le socle des réseaux de neurones profonds modernes.

En 1989, Yann LeCun et ses collaborateurs posent les bases des *réseaux de neurones convolutionnels (CNN)* pour le traitement d'images. Leur efficacité est largement démontrée dans l'article de 1998, *Gradient-Based Learning Applied to Document Recognition*,^[32] qui montre l'excellent rendement des CNN pour la reconnaissance de chiffres manuscrits.

Durant les années 1990, les *réseaux de neurones récurrents (RNN)* gagnent en importance pour le traitement des séquences. En 1997, Sepp Hochreiter et Jürgen Schmidhuber introduisent les *Long Short-Term Memory (LSTM)*,^[33] une version améliorée des RNN capable de capturer les dépendances à long terme.

Bien que les *autoencodeurs* aient été introduits dès les années 1980 comme technique de réduction de dimension, ce n'est qu'en 2006 que Geoffrey Hinton et Ruslan Salakhutdinov relancent leur utilisation à grande échelle dans un cadre d'apprentissage profond. Dans leur article *Reducing the Dimensionality of Data with Neural Networks*,^[34] ils proposent les **autoencodeurs empilés**, une approche non supervisée capable d'apprendre des représentations compressées et utiles, même sans annotations. Ce travail marque un tournant important vers des architectures de réseaux de neurones plus profondes et performantes.

En 2012, le modèle **AlexNet**, proposé par Krizhevsky, Sutskever et Hinton, remporte haut la main la compétition *ImageNet*.^[35] Ce succès spectaculaire marque le début de la révolution moderne du *deep learning*, en démontrant l'efficacité des architectures CNN profondes pour la reconnaissance visuelle à grande échelle.

En 2013, Diederik P. Kingma et Max Welling introduisent l'**autoencodeur variationnel (VAE)**.^[36] Contrairement aux autoencodeurs classiques, qui apprennent une représentation déterministe des données, le VAE adopte une approche probabiliste en contrignant l'espace latent à suivre une loi de probabilité bien définie, généralement gaussienne. Ce travail marque une étape importante dans l'évolution des modèles génératifs, ouvrant la voie à des applications variées comme la génération d'images, la détection d'anomalies ou encore la synthèse de données.

En 2014, Ian Goodfellow propose les **Generative Adversarial Networks (GANs)**,^[37] une architecture innovante où deux réseaux s'affrontent : un générateur tente de produire des données réalistes, tandis qu'un discriminateur cherche à les distinguer des vraies données. Ce mécanisme d'entraînement en compétition permet de générer des images, vidéos et même du son de qualité impressionnante.

En 2015, l'équipe de Kaiming He introduit **ResNet**,^[38] une architecture à connexions résiduelles qui permet d'entraîner des réseaux extrêmement profonds sans souffrir du problème de dégradation des performances. ResNet devient un standard dans de nombreuses tâches de vision.

En 2017, Google révolutionne le traitement du langage avec le modèle **Transformer**.^[39] Ce modèle, basé uniquement sur des mécanismes d'attention, élimine les contraintes des RNN et s'adapte rapidement à d'autres domaines.

En 2018, Devlin et al. introduisent **BERT** (**Bidirectional Encoder Representations from Transformers**),^[40] qui repose sur l'apprentissage par masquage de tokens et fournit des représentations contextuelles bidirectionnelles. La même année, OpenAI propose le premier **GPT**,^[41] un modèle de langage pré-entraîné de manière autoregressive, marquant le début d'une nouvelle génération de grands modèles de langage (LLM).

En 2019, OpenAI publie **GPT-2**,^[42] doté de 1,5 milliard de paramètres, capable de générer du texte cohérent sur de longues séquences. Cette version attire une attention considérable en raison de ses capacités de génération réaliste et de ses risques potentiels liés à la désinformation.

En 2020, Brown et al. présentent **GPT-3**,^[43] avec 175 milliards de paramètres, démontrant l'efficacité de l'échelle (scale) des modèles pour l'émergence de nouvelles capacités comme le raisonnement en contexte ou le *few-shot learning*. La même année, Dosovitskiy et al. proposent le **Vision Transformer (ViT)**,^[44] qui applique l'architecture Transformer au domaine de la vision en représentant une image comme une séquence de patchs. Toujours en 2020, Ho et al. introduisent les **modèles de diffusion**,^[45] qui génèrent des images réalistes en apprenant un processus inverse de débruitage progressif.

En 2021, OpenAI présente **CLIP** (Contrastive Language–Image Pretraining),^[46] qui apprend à relier texte et image grâce à un apprentissage contrastif sur de larges corpus multimodaux. CLIP devient une brique essentielle pour la génération conditionnelle d'images et la compréhension multimodale.

En 2022, Wei et al. introduisent le paradigme **Chain-of-Thought (CoT)**,^[47] qui consiste à inciter les modèles à expliciter leurs étapes de raisonnement intermédiaires, améliorant ainsi leur capacité de résolution de problèmes complexes. Dans la même année, Kojima et al. montrent que cette capacité peut émerger même en **zéro-shot**, avec des instructions simples comme « Let's think step by step ».^[48]

En 2023, Wang et al. proposent la méthode **Self-Consistency**,^[49] qui renforce CoT en agrémentant plusieurs chemins de raisonnement afin d'accroître la robustesse des réponses. La même année, Yao et al. développent le **Tree-of-Thought (ToT)**,^[50] qui généralise CoT en explorant de manière arborescente différents scénarios de raisonnement. Toujours en 2023, Meta introduit le **Segment Anything Model (SAM)**,^[51] une architecture universelle de segmentation capable de détecter et segmenter n'importe quel objet sans réentraînement.

En 2024, émergent les approches dites **multi-agents** et **augmentées par outils**, illustrées par le paradigme **ReAct** (Reason + Act),^[52] qui combine raisonnement explicite et exécution d'actions dans des environnements externes (moteurs de recherche, bases de données, etc.). Ces approches ouvrent la voie vers des systèmes d'IA plus interactifs et fiables.

Enfin, en 2025, les recherches se concentrent sur les **modèles unifiés**, capables de traiter du texte, des images, du son et même des actions dans une seule architecture. Des projets comme *Gato* et *Gemini*^[53] incarnent cette vision d'une intelligence artificielle plus générale et polyvalente, au-delà des modèles spécialisés.

2.3 Rappel au machine learning

Le *machine learning*, ou apprentissage automatique, est un sous-domaine de l'intelligence artificielle qui vise à développer des algorithmes capables d'apprendre à effectuer des tâches à partir d'exemples concrets. Ces exemples sont fournis sous forme de données, que l'algorithme utilise pour construire un modèle capable de faire des prédictions ou de prendre des décisions. Pour cela, il existe trois méthodes principales : **l'apprentissage supervisé**, **l'apprentissage non supervisé**, et **l'apprentissage par renforcement**. Chaque approche correspond à une manière particulière d'apprendre en fonction de la nature des données disponibles et du problème à résoudre.

2.3.1 Apprentissage supervisé

L'apprentissage supervisé est le cadre le plus courant dans le domaine de l'apprentissage automatique. Le principe est simple : on dispose d'un ensemble de données d'entrée X , et pour chaque entrée, on connaît la sortie attendue y . Le but est alors d'entraîner un modèle à prédire correctement y à partir de x .

Pour mieux comprendre, prenons l'exemple d'une fonction linéaire simple : $f(x) = ax + b$. Dans ce cas, le rôle de l'apprentissage automatique est de déterminer automatiquement les valeurs optimales des paramètres a et b , de façon à ce que la fonction s'ajuste le mieux possible aux données observées. Ainsi, le modèle apprend à généraliser à partir des exemples fournis pour faire de bonnes prédictions sur de nouvelles données.

Pour atteindre cet objectif, on utilise un *algorithme d'optimisation*, qui fonctionne en deux étapes :

- Premièrement, on calcule l'erreur entre la prédiction du modèle $\hat{y} = f(x)$ et la valeur réelle y ;
- Deuxièmement, on modifie les paramètres a et b de manière à minimiser cette erreur.

En d'autres termes, l'algorithme cherche la combinaison des paramètres qui réduit le plus possible l'écart entre les prédictions du modèle et les données réelles.

2.3.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, dans l'apprentissage non supervisé, on ne dispose pas des sorties attendues y . On a uniquement les données d'entrée X , sans indication sur ce que le modèle est censé produire. Le but ici n'est donc pas de prédire une valeur cible, mais plutôt de découvrir des structures ou des motifs cachés dans les données.

Pour mieux illustrer, reprenons notre exemple de fonction affine $f(x) = ax + b$. Supposons cette fois que nous ne connaissons pas les valeurs de y . Dans ce cas, il est impossible de calculer directement une erreur de prédiction, puisque nous ne savons pas à quoi le modèle devrait ressembler.

Cependant, on peut toujours analyser les données disponibles afin de détecter des regroupements naturels ou des similarités. Par exemple, un algorithme non supervisé peut regrouper les points qui semblent proches les uns des autres, formant ainsi des clusters. Cette approche est couramment utilisée pour la segmentation, la compression de données ou encore la réduction de dimensionnalité.

2.3.3 Apprentissage par renforcement

L'apprentissage par renforcement constitue une approche différente des deux précédentes. Dans ce cadre, le modèle que l'on appelle souvent un agent n'a ni les sorties attendues y , ni même un ensemble de données fixes X . Il apprend en interagissant avec un environnement, en recevant des récompenses ou des punitions selon les actions qu'il entreprend.

On peut voir cela comme un jeu où l'agent ne connaît pas les règles à l'avance. Il doit les découvrir progressivement en essayant différentes actions. Si l'action choisie conduit à un bon résultat, il recevra une récompense positive ; sinon, une pénalité.

Revenons à notre analogie de fonction affine : ici, l'agent ne connaît même pas la fonction, $f(x) = ax + b$, ni les données X. Il doit expérimenter plusieurs entrées, observer les résultats, et petit à petit ajuster son comportement afin de maximiser les récompenses cumulées. Ce type d'apprentissage est largement utilisé dans les jeux, la robotique et les systèmes de recommandation.

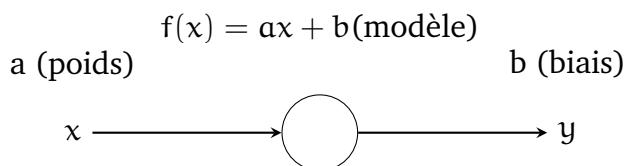
2.3.4 Régression linéaire

En **apprentissage supervisé**, les algorithmes cherchent à établir des relations entre des données d'entrée (ou caractéristiques) et des résultats (ou cibles). La **régression linéaire** est un modèle fondamental qui vise à ajuster une droite pour modéliser la relation entre les variables d'entrée x et les variables de sortie y . Ce modèle permet de prédire y à partir de x en utilisant la fonction suivante :

$$f(x) = ax + b$$

où :

- a représente la pente de la droite, c'est-à-dire la manière dont y varie par rapport à x ,
 - b est l'ordonnée à l'origine, c'est-à-dire la valeur de y lorsque $x = 0$,
 - x désigne les variables d'entrée (ou caractéristiques) de notre modèle.



Afin d'évaluer la qualité de l'ajustement du modèle, nous utilisons une fonction coût, notée $J(a, b)$, qui mesure l'erreur entre les prédictions du modèle et les valeurs réelles. Dans le cadre de la régression linéaire, on peut utiliser par exemple l'erreur quadratique moyenne définie comme suit :

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})^2$$

où :

- m représente le nombre total d'exemples dans notre jeu de données,
 - $x^{(i)}$ et $y^{(i)}$ désignent respectivement les i -èmes valeurs des variables d'entrée et de sortie.

L'objectif est de minimiser cette fonction coût, c'est-à-dire de trouver les valeurs de a et b qui réduisent l'erreur entre les prédictions $f(x)$ et les valeurs réelles y . Pour cela, on utilise la méthode de la descente de gradient.

Les gradients, qui indiquent la direction à suivre pour ajuster les paramètres a et b , sont calculés comme suit :

$$\frac{\partial J(a, b)}{\partial a} = \frac{1}{m} \sum_{i=1}^m x^{(i)}(ax^{(i)} + b - y^{(i)})$$

Ce gradient montre comment ajuster la pente a en fonction des erreurs commises par le modèle. De manière similaire, pour b , le gradient est :

$$\frac{\partial J(a, b)}{\partial b} = \frac{1}{m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})$$

Ce gradient mesure l'impact des erreurs sur l'ordonnée à l'origine b .

Pour minimiser la fonction coût, on utilise l'algorithme de descente de gradient, qui consiste à ajuster progressivement a et b en suivant la direction inverse des gradients. Les règles de mise à jour des paramètres sont les suivantes :

$$a := a - \alpha \frac{\partial J(a, b)}{\partial a}$$

$$b := b - \alpha \frac{\partial J(a, b)}{\partial b}$$

où α est le taux d'apprentissage, un paramètre qui contrôle la vitesse à laquelle les paramètres sont mis à jour. Un choix approprié de α est crucial pour garantir une convergence efficace de l'algorithme.

2.3.5 Vectorisation des équations

Pour optimiser l'algorithme de régression linéaire, nous utilisons la vectorisation, qui permet de traiter plusieurs exemples simultanément. Plutôt que de travailler avec des scalaires, nous représentons les variables sous forme de vecteurs et de matrices.

La fonction de prédiction $f(x)$ peut être réécrite en termes de vecteurs. Si $X \in \mathbb{R}^{m \times n}$ représente la matrice des caractéristiques, où chaque ligne correspond à un exemple et chaque colonne à une caractéristique, et que $\theta \in \mathbb{R}^{1 \times n}$ est le vecteur des poids (ou paramètres), alors la fonction de prédiction vectorisée est :

$$f(X) = X\theta^T$$

La fonction coût $J(\theta)$, qui mesure l'erreur quadratique moyenne, est définie de manière vectorisée comme suit :

$$J(\theta) = \frac{1}{2m} \sum (X\theta^T - Y)^2$$

Pour minimiser la fonction coût, il est nécessaire de calculer les gradients par rapport à θ . Le gradient de la fonction coût $J(\theta)$ par rapport au vecteur des paramètres θ est donné par :

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T (X\theta^T - Y)$$

L'algorithme de descente de gradient permet de mettre à jour les paramètres θ pour minimiser la fonction coût. Les mises à jour des paramètres se font selon la règle suivante :

$$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

2.4 Origine des réseaux de neurones artificiel

2.4.1 Fonctionnement des neurones biologiques

Les neurones biologiques sont les cellules de base du système nerveux, chargées de transmettre les informations sous forme de signaux électriques et chimiques dans tout le corps. Leur fonctionnement est essentiel pour les processus cognitifs tels que la perception, la mémorisation et le mouvement. Un neurone typique se compose de trois parties principales : les dendrites, le corps cellulaire (ou soma) et l'axone.

Les dendrites sont de fines extensions qui entourent le corps cellulaire et agissent comme des antennes, recevant les signaux provenant d'autres neurones. Ces signaux arrivent par des points de contact spécialisés appelés synapses, où ils peuvent être de nature excitatrice (stimuler l'activité du neurone) ou inhibitrice (réduire l'activité du neurone). Chaque neurone peut recevoir des milliers de signaux simultanés à travers ses dendrites.

Le corps cellulaire contient le noyau du neurone et traite les informations reçues. Il calcule la somme des signaux entrants et détermine s'ils sont suffisants pour déclencher une réponse. Si la somme des signaux dépasse un seuil critique, le neurone déclenche un potentiel d'action, qui est un signal électrique.

Une fois que le potentiel d'action est généré, il est envoyé le long de l'axone, une extension allongée qui conduit le signal vers d'autres neurones ou cellules cibles. Ce signal se propage rapidement le long de l'axone jusqu'aux terminaisons synaptiques, où il déclenche la libération de neurotransmetteurs dans la synapse. Ces neurotransmetteurs, à leur tour, transmettent le signal à d'autres neurones en se liant aux récepteurs situés sur leurs dendrites.

Ce processus de communication entre neurones est essentiel à toutes les fonctions du cerveau, qu'il s'agisse de la pensée, de l'apprentissage, de la mémoire ou de la régulation des mouvements corporels. En particulier, la manière dont les signaux sont modulés et traités par les synapses joue un rôle clé dans l'adaptation et la plasticité neuronale, deux mécanismes fondamentaux pour l'apprentissage et la mémorisation.

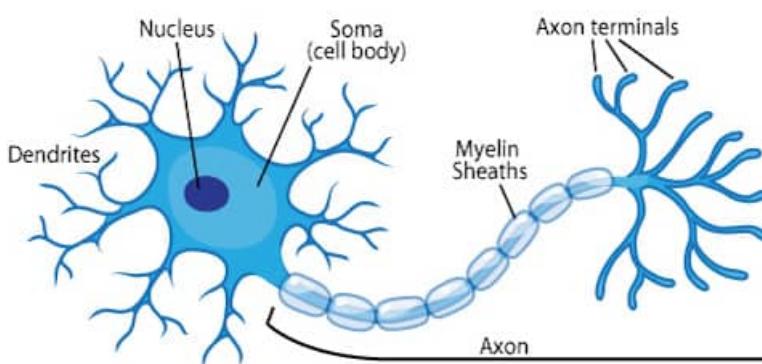


Figure 2.1 – Neurone biologique

Source : <https://images.app.goo.gl/kXQLJorThjaqVwQE7>

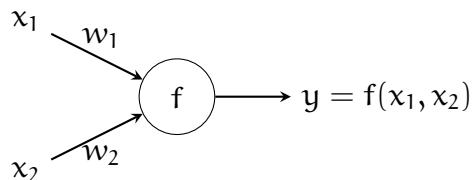
2.4.2 Fonctionnement des neurones artificielles

En **machine learning** classique , le principe fondamental repose sur la capacité des algorithmes à apprendre des relations à partir de données. Ces algorithmes permettent de modéliser des fonctions

capables de prédire des résultats ou de classer des données en fonction des informations fournies. Cependant, dans le cadre des réseaux de neurones artificiels, le modèle va bien au-delà de tous les autres modèles. Il repose sur l'assemblage et l'interconnexion de multiples fonctions, créant ainsi un réseau de neurones artificiels capables d'apprendre des relations complexes à partir de grandes quantités de données.

Pour comprendre le fonctionnement d'un réseau de neurones artificiels, il est d'abord essentiel de bien saisir ce qui se passe au niveau d'une seule unité de ce réseau : le neurone artificiel. Chaque neurone artificiel est une unité de traitement qui reçoit des informations en entrée, les transformer, puis produit une sortie qui sera transmise aux neurones suivants.

Dans cette section, nous introduirons le modèle de McCulloch et Pitts, qui constitue l'une des premières tentatives formelles de modélisation des neurones biologiques en termes mathématiques et logiques. Proposé en 1943 par Warren McCulloch et Walter Pitts, ce modèle repose sur la représentation des neurones comme des unités de traitement binaires capables d'effectuer des calculs logiques simples, tels que *ET*, *OU* et *NON*. Le neurone reçoit plusieurs signaux en entrée, les pondère, puis génère une sortie si la somme pondérée dépasse un certain seuil. Bien que rudimentaire, ce modèle a jeté les bases des réseaux de neurones artificiels modernes en introduisant la notion d'activation neuronale et de traitement parallèle d'informations.



Au cœur de ce modèle, deux étapes essentielles définissent le fonctionnement d'un neurone artificiel. Premièrement, il y a la phase d'agrégation. Durant cette étape, on calcule la somme pondérée des entrées x_1, x_2, \dots, x_n avec leurs poids respectifs w_1, w_2, \dots, w_n . Ces poids représentent l'influence synaptique des connexions. Par analogie avec les neurones biologiques, un poids w_i positif représente un signal excitateur, tandis qu'un poids w_i négatif correspond à un signal inhibiteur. La fonction d'agrégation peut ainsi être exprimée comme suit :

$$f(x_1, x_2, \dots, x_n) = w_1x_1 + w_2x_2 + \dots + w_nx_n$$

Ensuite, nous avons la phase d'activation. Après avoir obtenu le résultat de la fonction d'agrégation, celui-ci est comparé à un seuil prédéfini. Si ce résultat dépasse le seuil, le neurone s'active et produit une sortie $y = 1$. Dans le cas contraire, si la valeur est inférieure au seuil, le neurone reste inactif et la sortie est $y = 0$. Cela peut être résumé par la fonction d'activation suivante :

$$y = \begin{cases} 1 & \text{si } f(x_1, x_2, \dots, x_n) > \text{seuil} \\ 0 & \text{sinon} \end{cases}$$

2.5 Étude du perceptron

2.5.1 Le perceptron à une seule couche

Le perceptron est l'unité de base des réseaux de neurones. Il s'agit d'un modèle de classification binaire, capable de séparer deux classes linéairement séparables. Le fonctionnement du perceptron est similaire à celui des modèles précédents, avec une fonction d'agrégation définie par :

$$z(x_1, x_2) = w_1x_1 + w_2x_2 + b$$

où b est une constante appelée *biais*.

Supposons que nous avons deux classes : la première associée à $y = 1$ et la seconde à $y = 0$. Le modèle du perceptron, dans sa forme classique, utilise cette fonction d'agrégation pour séparer ces deux classes.

L'équation $z(x_1, x_2) = 0$ est appelée *frontière de décision*. Si, pour un point donné, la valeur de z est négative, ce point sera prédit comme appartenant à la classe $y = 0$. En revanche, si la valeur de z est positive, le point sera prédit comme appartenant à la classe $y = 1$.

Cependant, au lieu de simplement décider de la classe en fonction du signe de z , nous pouvons introduire une notion de probabilité. Pour cela, nous utilisons la **fonction sigmoïde**. La fonction sigmoïde est une fonction mathématique qui prend une valeur quelconque et la transforme en une probabilité entre 0 et 1. Elle est définie par l'expression suivante :

$$a(z) = \frac{1}{1 + e^{-z}}$$

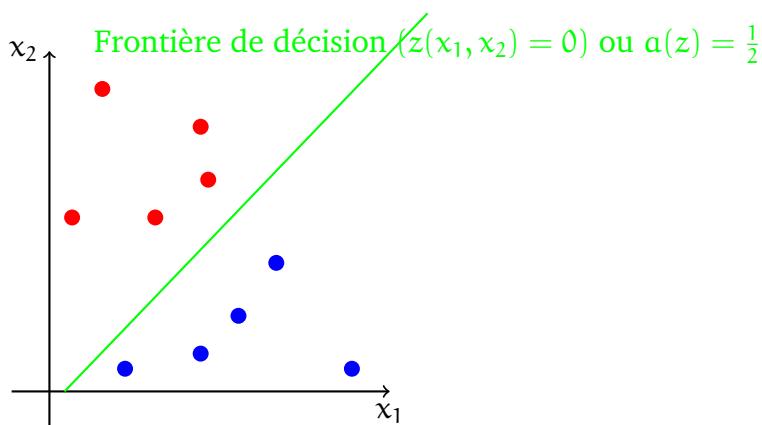
En utilisant la fonction sigmoïde, nous pouvons calculer la probabilité que $y = 1$, étant donné les entrées x_1 et x_2 . Ainsi, la probabilité que le point appartienne à la classe $y = 1$ est donnée par :

$$\mathbb{P}(y = 1|x_1, x_2) = a(z(x_1, x_2))$$

Cette probabilité suit une **loi de Bernoulli**, qui est une distribution de probabilité utilisée pour les variables aléatoires binaires (avec deux résultats possibles : 0 ou 1). La probabilité qu'une variable y suive une loi de Bernoulli s'exprime ainsi :

$$\mathbb{P}(y_i) = p^{y_i} (1 - p)^{1-y_i}$$

Dans ce cas, $p = a(z(x_1, x_2))$ est la probabilité que $y = 1$, et $1 - p$ est la probabilité que $y = 0$. Cela signifie que notre modèle prédit non seulement à quelle classe un point appartient, mais aussi la probabilité associée à cette prédiction.



En apprentissage automatique, il est essentiel de définir une fonction pour quantifier les erreurs commises par un modèle. Cette fonction est généralement appelée *fonction coût* (ou *loss function* en anglais). Elle permet de mesurer l'écart entre les prédictions du modèle et les résultats attendus.

Pour le faire, nous utilisons souvent le concept de *vraisemblance*, qui exprime la probabilité que les paramètres du modèle expliquent correctement les données observées. Plus la vraisemblance est

élevée, plus le modèle est considéré comme étant en adéquation avec les données réelles. L'objectif en apprentissage automatique est de trouver les paramètres du modèle (w_1, w_2 et b) qui maximise cette vraisemblance ou, de manière équivalente, qui minimise la fonction coût.

Dans le cas d'une variable aléatoire qui suit une loi de Bernoulli, où chaque observation y_i peut prendre deux valeurs (0 ou 1), la vraisemblance L est définie par :

$$L = \prod_{i=1}^m P(y_i) \implies L = \prod_{i=1}^m p^{y_i} (1-p)^{1-y_i}$$

où :

- p est la probabilité que $y_i = 1$,
- $1 - p$ est la probabilité que $y_i = 0$,
- m est le nombre d'exemples dans les données.

Cette expression de la vraisemblance permet de calculer à quel point le modèle est cohérent avec les observations réelles.

Comme vous pouvez le constater, dans le cadre de la vraisemblance, nous calculons le produit des probabilités associées à chaque donnée. Cela signifie que plus nous avons de données, plus le résultat de ce produit tend à se rapprocher de zéro, ce qui peut rendre les calculs numériquement instables.

Pour éviter cette situation, nous utilisons le *logarithme de la vraisemblance*. En effet, prendre le logarithme d'un produit revient à effectuer la somme des logarithmes des probabilités.

$$\begin{aligned} \log(L) &= \log \left(\prod_{i=1}^m P(y_i) \right) \\ \log(L) &= \sum_{i=1}^m (y_i \log(p) + (1 - y_i) \log(1 - p)) \end{aligned}$$

avec $p = a(z_i)$ et on pose $a(z_i) = a_i$

Le problème réside dans le fait que en apprentissage automatique les algorithmes d'optimisation sont conçus pour minimiser une fonction, et non pour la maximiser. Par conséquent, nous devons prendre l'opposé du *logarithme de la vraisemblance*, tout en normalisant nos résultats en multipliant par $\frac{1}{m}$. Ainsi, la fonction de coût est définie par :

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^m (y_i \log(a_i) + (1 - y_i) \log(1 - a_i))$$

À présent, nous allons calculer les gradients de la fonction coût afin d'ajuster les paramètres de notre modèle en utilisant l'algorithme de descente de gradient. La fonction coût est notée \mathcal{L} , et nous voulons dériver cette fonction par rapport à chaque paramètre du modèle : w_1, w_2 , et b .

Tout d'abord, pour w_1 , nous avons :

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w_1}$$

Et que :

$$\frac{\partial z(x_1, x_2)}{\partial w_1} = x_1, \quad \frac{\partial a(z)}{\partial z} = a(z)(1 - a(z))$$

et

$$\frac{\partial \mathcal{L}}{\partial a} = -\frac{1}{m} \sum_{i=1}^m \left(\frac{y_i}{a_i} - \frac{1-y_i}{1-a_i} \right)$$

En combinant ces éléments, nous obtenons :

$$\frac{\partial \mathcal{L}}{\partial \omega_1} = \frac{1}{m} \sum_{i=1}^m (a_i - y_i)x_1$$

De manière similaire, pour ω_2 :

$$\frac{\partial \mathcal{L}}{\partial \omega_2} = \frac{1}{m} \sum_{i=1}^m (a_i - y_i)x_2$$

Enfin, pour le biais b , le gradient est :

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum_{i=1}^m (a_i - y_i)$$

Pour manipuler efficacement nos données en grande dimension, nous utilisons la vectorisation. Cette technique est essentielle en *machine learning* et *Deep learning*, car elle permet d'effectuer des opérations sur des matrices et des vecteurs, plutôt que de les exécuter de manière itérative à l'aide de boucles. Cela optimise le calcul, notamment pour la mise à jour des paramètres du modèle. Ainsi, notre fonction d'agrégation z s'écrit :

$$Z = XW + b$$

où

- $X \in \mathbb{R}^{m \times n}$, avec m le nombre d'observations et n le nombre de variables. Dans cet exemple, $n = 2$ car nous avons deux variables entre x_1 et x_2 ,
- $W \in \mathbb{R}^{n \times 1}$ est le vecteur des poids (avec $n = 2$),
- $b \in \mathbb{R}$ est le biais.

Ensuite, le vecteur d'activation A est donné par la fonction sigmoïde :

$$A(Z) = \frac{1}{1 + e^{-Z}}$$

À présent, nous faisons la vectorisation de la fonction coût ainsi que des équations des gradients :

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^m (y_i \log(A) + (1-y_i) \log(1-A))$$

Les mises à jour des paramètres se font ainsi :

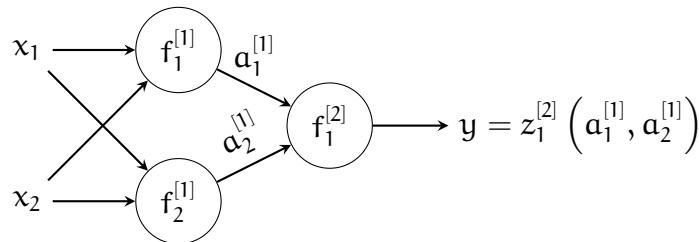
$$W = W - \alpha \frac{\partial \mathcal{L}}{\partial W} \quad \text{et} \quad b = b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$

où :

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{m} X^T (A - y) \quad \text{et} \quad \frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum_{i=1}^m (A - y)$$

2.5.2 Perceptron multicouche

Le perceptron multicouche (ou MLP pour Multilayer Perceptron) est une extension du perceptron classique qui permet de traiter des problèmes de classification plus complexes, notamment ceux qui ne sont pas linéairement séparables. Contrairement au perceptron simple qui ne comporte qu'une seule couche de neurones, le perceptron multicouche est constitué de plusieurs couches : une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie. Chaque neurone d'une couche est connecté à ceux de la couche suivante via des poids. Ces connexions permettent au modèle d'apprendre des représentations plus abstraites des données en passant par des transformations non linéaires grâce aux fonctions d'activation (souvent la fonction sigmoïde ou ReLU). Chaque couche intermédiaire joue ainsi un rôle essentiel dans la capture des interactions complexes entre les variables d'entrée, permettant au réseau de résoudre des tâches comme la reconnaissance d'images ou la classification de données non linéaires. Le perceptron multicouche est entraîné via l'algorithme de rétropropagation de l'erreur, une méthode qui ajuste les poids du réseau en fonction de l'erreur de prédiction, rendant ce modèle puissant et adaptable à une large variété de problèmes.



avec :

$$\begin{cases} z(x_{11}, x_{12})^{[1]} = w_{11}^{[1]}x_{11}^{[1]} + w_{12}^{[1]}x_{12}^{[1]} + b_1^{[1]} \\ z(x_{21}, x_{22})^{[1]} = w_{21}^{[1]}x_{21}^{[1]} + w_{22}^{[1]}x_{22}^{[1]} + b_2^{[1]} \end{cases} \quad z^{[2]}(a_1^{[1]}, a_2^{[1]}) = w_{11}^{[2]}a_1^{[1]} + w_{12}^{[2]}a_2^{[1]} + b_1^{[2]}$$

Dans un réseau de neurones, écrire les équations des fonctions d'agrégation pour chaque neurone individuellement devient rapidement une tâche extrêmement complexe, voire impraticable pour des architectures de grande taille. Pour simplifier cette démarche, nous utilisons une représentation vectorielle des équations, qui permet de modéliser efficacement l'ensemble des calculs réalisés au sein du réseau.

$$Z^{[1]} = \begin{bmatrix} x_{11}^{(1)} & x_{21}^{(1)} \\ x_{12}^{(2)} & x_{22}^{(2)} \end{bmatrix} \times \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix} \quad Z^{[2]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} \times \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \end{bmatrix}$$

Ainsi, toutes les équations de notre modèle, ainsi que les activations associées, peuvent être exprimées comme suit :

$$Z^{[1]} = W^{[1]} \cdot X^T + b^{[1]} \quad Z^{[2]} = W^{[2]} \cdot A^{[1]} + b^{[2]}$$

$$A^{[1]} = \frac{1}{1 + e^{-Z^{[1]}}}$$

$$A^{[2]} = \frac{1}{1 + e^{-Z^{[2]}}}$$

- $X \in \mathbb{R}^{m \times n}$,
- $W^{[1]} \in \mathbb{R}^{n \times n}$, n le nombre de neurone dans la couche et ${}^{[1]}$ le numéro de la couche
- $b^{[1]} \in \mathbb{R}^{1 \times n}$

$$\mathcal{L} = -\frac{1}{m} \sum [y \times \log(A^{[2]}) + (1 - y) \times \log(1 - A^{[2]})]$$

- y représente les labels réels.
- $A^{[2]}$ est la sortie du réseau après activation.
- m est le nombre total d'exemples dans le jeu de données.

Cette technique est indispensable pour permettre l'ajout d'un nombre arbitraire de neurones de manière flexible, tout en garantissant une représentation efficace et généralisable des calculs au sein du réseau.

À ce stade, une question essentielle se pose : **comment entraîner un tel réseau de neurones ?** En d'autres termes, comment faire en sorte que le modèle apprenne à faire de bonnes prédictions ? Pour cela, on ajuste les paramètres du réseau c'est-à-dire les poids et les biais en minimisant une **fonction de coût**, qui mesure l'erreur entre les prédictions du réseau et les valeurs attendues.

L'algorithme utilisé pour ce processus est appelé *rétropropagation des gradients* (ou *backpropagation*). Il a été popularisé par Geoffrey Hinton et ses collaborateurs en 1986. Cet algorithme repose sur une idée simple mais puissante : calculer l'erreur à la sortie du réseau, puis la propager en sens inverse, couche par couche, afin d'ajuster les poids dans la bonne direction.

L'erreur se propage à l'aide de la règle de la dérivation en chaîne. Ainsi, pour une architecture comportant deux couches (une cachée et une de sortie), les gradients des poids et biais peuvent être exprimés comme suit :

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w^{[2]}} &= \frac{\partial \mathcal{L}}{\partial A^{[2]}} \times \frac{\partial A^{[2]}}{\partial Z^{[2]}} \times \frac{\partial Z^{[2]}}{\partial w^{[2]}} & \frac{\partial \mathcal{L}}{\partial w^{[1]}} &= \frac{\partial \mathcal{L}}{\partial A^{[2]}} \times \frac{\partial A^{[2]}}{\partial Z^{[2]}} \times \frac{\partial Z^{[2]}}{\partial A^{[1]}} \times \frac{\partial A^{[1]}}{\partial Z^{[1]}} \times \frac{\partial Z^{[1]}}{\partial w^{[1]}} \\ \frac{\partial \mathcal{L}}{\partial b^{[2]}} &= \frac{\partial \mathcal{L}}{\partial A^{[2]}} \times \frac{\partial A^{[2]}}{\partial Z^{[2]}} \times \frac{\partial Z^{[2]}}{\partial b^{[2]}} & \frac{\partial \mathcal{L}}{\partial b^{[1]}} &= \frac{\partial \mathcal{L}}{\partial A^{[2]}} \times \frac{\partial A^{[2]}}{\partial Z^{[2]}} \times \frac{\partial Z^{[2]}}{\partial A^{[1]}} \times \frac{\partial A^{[1]}}{\partial Z^{[1]}} \times \frac{\partial Z^{[1]}}{\partial b^{[1]}}\end{aligned}$$

Pour des raisons d'efficacité computationnelle, ces dérivées sont réécrites sous forme matricielle, utilisée dans la pratique lors de l'entraînement. Elle découle directement de la chaîne de dérivations présentée ci-dessus.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial A^{[2]}} &= - \left(\frac{y}{A^{[2]}} - \frac{1-y}{1-A^{[2]}} \right) & \frac{\partial \mathcal{L}}{\partial A^{[2]}} \times \frac{\partial A^{[2]}}{\partial Z^{[2]}} &= A^{[2]}(1-A^{[2]})A^{[1]} \\ \frac{\partial A^{[2]}}{\partial Z^{[2]}} &= A^{[2]}(1-A^{[2]}), \quad \frac{\partial Z^{[2]}}{\partial w^{[2]}} = A^{[1]} & \frac{\partial Z^{[2]}}{\partial A^{[1]}} = w^{[2]}, \quad \frac{\partial A^{[1]}}{\partial Z^{[1]}} = A^{[1]}(1-A^{[1]}), \quad \frac{\partial Z^{[1]}}{\partial w^{[1]}} = X^T \\ \frac{\partial \mathcal{L}}{\partial w^{[2]}} &= \frac{1}{m} (A^{[2]} - y)(A^{[1]})^T & \frac{\partial \mathcal{L}}{\partial w^{[1]}} &= \frac{1}{m} [(w^{[2]})^T (A^{[2]} - y)A^{[1]}(1-A^{[1]})] X^T \\ \frac{\partial \mathcal{L}}{\partial b^{[2]}} &= \frac{1}{m} \sum_{i=1}^m (A_i^{[2]} - y_i) & \frac{\partial \mathcal{L}}{\partial b^{[1]}} &= \frac{1}{m} \sum_{i=1}^m [(w^{[2]})^T (A^{[2]} - y)A^{[1]}(1-A^{[1]})]_i\end{aligned}$$

Ces expressions constituent le **cœur de la rétropropagation**, car elles permettent de calculer les dérivées de la fonction de coût par rapport à tous les paramètres du réseau.

Une fois les gradients calculés, on met à jour les paramètres avec la règle de la descente de gradient. Pour chaque couche l , on applique la règle suivante :

$$w^{[l]} := w^{[l]} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial w^{[l]}}, \quad b^{[l]} := b^{[l]} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial b^{[l]}}$$

2.6 Les réseaux de neurones modernes

2.6.1 Réseaux de neurones convolutionnels

Dans la section précédente, nous avons étudié le perceptron multicouche (*Multi-Layer Perceptron*, MLP), un modèle capable d'apprendre à partir d'exemples en ajustant automatiquement ses poids. Bien que performant sur des données tabulaires ou de petite dimension, le MLP classique montre rapidement ses limites lorsqu'il s'agit de traiter des images.

Avant l'essor des réseaux de neurones modernes, l'analyse d'images reposait principalement sur des méthodes dites *traditionnelles* d'extraction de caractéristiques. Ces techniques consistaient à définir manuellement, pour chaque tâche, un ensemble de descripteurs visuels (comme les contours, les textures ou les formes géométriques) à l'aide d'algorithmes spécialisés, tels que :

- **SIFT** (Scale-Invariant Feature Transform) : pour détecter des points-clés invariants aux transformations géométriques.
- **HOG** (Histogram of Oriented Gradients) : pour capturer la structure locale des gradients dans une image.
- **SURF, LBP**, ou encore les filtres de Gabor.

Ces méthodes nécessitaient une expertise humaine approfondie pour bien choisir les paramètres, et leur efficacité dépendait fortement du contexte et du type d'image. De plus, les modèles utilisés ensuite (comme les SVM ou les k-NN) travaillaient uniquement sur les caractéristiques extraites, sans capacité à apprendre directement à partir des pixels bruts.

Le problème fondamental était donc : *comment faire en sorte qu'un modèle apprenne automatiquement les bonnes caractéristiques à partir des images ?* Autrement dit, *comment permettre au modèle de détecter lui-même les contours, textures ou motifs pertinents pour accomplir une tâche donnée, sans avoir à les définir manuellement ?*

C'est dans ce contexte qu'est apparue une solution qui a changé complètement la donne : les **réseaux de neurones convolutionnels** (*Convolutional Neural Networks*, CNN),^[32] introduits par Yann LeCun et ses collaborateurs à la fin des années 1980. Ces réseaux ont été spécialement conçus pour exploiter la structure spatiale des images. Grâce aux opérations de convolution, ils sont capables de détecter automatiquement, à différents niveaux de profondeur, des motifs visuels de plus en plus complexes, tout en maintenant la cohérence spatiale des données.

Dans la suite de cette section, nous allons étudier le fonctionnement des CNN, leur architecture, ainsi que leur impact sur la vision par ordinateur moderne.

L'opération de convolution^[54]

La convolution est une opération mathématique fondamentale utilisée dans de nombreux domaines bien avant son adoption en apprentissage profond. Elle a notamment été développée dans le cadre de l'analyse des signaux, où elle permet de filtrer ou d'extraire certaines caractéristiques d'un signal.

Par exemple, un signal bruité peut être nettoyé en le faisant passer par une fonction de filtrage notée g , appliquée à un signal d'entrée f , dans le but de produire une sortie plus exploitable. Cette opération est définie en continu par :

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(t) \cdot g(x - t) dt$$

Dans le cas discret, la formule devient une somme :

$$(X * w)(i) = \sum_{a=-\infty}^{+\infty} X(i) \cdot w(i - a)$$

Et pour les données bidimensionnelles, comme les images, la convolution est généralisée sous la forme :

$$(X * w)(i, j) = \sum_m \sum_n X(m, n) \cdot w(i + m, j + n)$$

Dans les CNNs, on ajoute généralement un terme supplémentaire appelé **biais**, noté b , ce qui donne la formule d'agrégation suivante :

$$Z = X * w + b$$

Une fois cette opération effectuée, le processus d'apprentissage continue à l'aide des méthodes classiques telles que la **descente de gradient**. Cette étape permet de mettre à jour automatiquement les paramètres du réseau, en particulier les poids du filtre w (également appelé *noyau de convolution*) ainsi que le biais b . Ces paramètres sont ajustés de manière à extraire les caractéristiques les plus pertinentes de l'image, comme les *bords*, les *textures*, ou encore certaines *formes géométriques* utiles à la tâche de classification ou de détection.

L'opération de Pooling^[54]

En *Deep Learning*, les images sont souvent très grandes et complexes à manipuler. Bien que les convolutions permettent d'extraire des caractéristiques importantes, elles ne suffisent pas à elles seules pour résoudre le problème de la taille des données.

En effet, les images comportent des centaines, voire des milliers de pixels, ce qui rend le traitement computationnellement coûteux. Pour pallier ce problème, on introduit une étape supplémentaire appelée **pooling**.

L'objectif du pooling est de réduire la taille des représentations intermédiaires tout en conservant les informations les plus importantes. Pour cela, on applique un filtre (souvent de taille 2×2) sur des sous-régions de l'image, et on remplace cette sous-région par une seule valeur.

Les types de pooling les plus courants sont :

- **Max Pooling** : sélectionne le pixel de plus grande valeur dans la région analysée.
- **Average Pooling** : calcule la moyenne des pixels dans la région.

Ce principe permet non seulement de réduire le nombre de paramètres et le coût de calcul, mais aussi de rendre le modèle plus robuste aux petites variations ou déplacements dans l'image. Il est donc largement utilisé dans les CNN modernes.

Couche de convolution

Comme on peut le deviner, une couche de convolution n'est rien d'autre que la combinaison de deux opérations essentielles : la convolution pour filtrer les images (éliminer le bruit ou extraire des motifs). Cette opération est suivie par l'application d'une fonction d'activation.

On utilise généralement la fonction *ReLU* comme activation, car elle introduit de la non-linéarité de manière simple et efficace.

Après la convolution, on applique une opération de **pooling**, qui a pour rôle de réduire la taille de l'image tout en mettant en évidence les caractéristiques les plus importantes. Par exemple, dans le cas du *max pooling*, on prend la valeur maximale parmi les pixels d'une région donnée, ce qui permet de ne conserver que les zones les plus marquantes.

Dans la pratique, il a été constaté que prendre le maximum (max pooling) fonctionne souvent mieux que de prendre la moyenne (average pooling), car cela conserve mieux les contours et les détails clés.

De plus, dans une même couche de convolution, on peut appliquer plusieurs filtres en parallèle à l'image d'entrée. Cela signifie qu'au lieu de passer une seule fois un filtre w sur l'image X , on utilise plusieurs filtres w_1, w_2, \dots, w_k , ce qui permet de détecter différentes caractéristiques à différents endroits.

Réseau de neurones convolutionnels

Un réseau de neurones convolutionnels (CNN) est une suite organisée de couches de convolution et de pooling. L'objectif est d'extraire, couche après couche, des caractéristiques de plus en plus complexes tout en réduisant progressivement la taille des images.

Après plusieurs couches, on obtient un ensemble de caractéristiques dites « profondes », que l'on a extraites grâce à l'empilement des filtres.

Ces caractéristiques sont ensuite « aplatis » pour être données en entrée à un **perceptron multicouche**, souvent utilisé comme classifieur final.

Ce classifieur prend l'ensemble des caractéristiques extraites et les combine pour prédire la classe de l'image.

En résumé, un CNN applique plusieurs filtres sur l'image, réduit la dimension des données tout en préservant l'information essentielle, puis utilise un réseau de neurones classique pour effectuer la prédiction. C'est un peu comme si le réseau apprenait lui-même à construire ses propres descripteurs, plutôt que de les programmer manuellement.

2.6.2 Auto-encodeurs

Depuis l'apparition des CNNs, ces derniers ont largement dominé le domaine du traitement d'images. Ils ont progressivement remplacé toutes les méthodes dites traditionnelles, et jusqu'à présent, aucune autre approche n'a démontré de meilleures performances pour les tâches classiques comme la classification.

Cependant, la classification supervisée montre quelques limites dès qu'on aborde des cas un peu plus subtils. En réalité, ce n'est pas la classification en elle-même qui pose problème, mais plutôt le fait que certaines tâches sont trop complexes pour être résolues uniquement par des méthodes de classification.

Prenons les cas suivants :

- Déetecter des comportements anormaux dans une vidéo de surveillance ;
- Identifier un produit mal formé sur une chaîne de production automatisée.

Dans ce type de situation, il est difficile, voire impossible, d'obtenir suffisamment d'exemples de comportements dits « anormaux » pour entraîner un modèle de classification fiable. En effet, ces événements sont rares, imprévisibles, et souvent définis selon le contexte.

De plus, un modèle entraîné aujourd’hui sur une forme d’anomalie pourrait devenir obsolète si les comportements évoluent avec le temps. Cela rend l’approche purement supervisée peu efficace dans ces cas.

Le véritable problème réside donc dans le fait que :

Ce n'est pas l'abondance des données normales qui pose un défi, mais plutôt la rareté et la variabilité des données anormales.

Ainsi, la solution la plus efficace serait d’entraîner un modèle uniquement sur les données normales et à lui apprendre que tout ce qui ne leur ressemble pas est une anomalie.

C'est dans cette logique qu'en **2006, Geoffrey Hinton et ses collègues ont proposé une solution spectaculaire** : les *autoencodeurs*. Et c'est justement ce que nous allons développer dans cette section.

Principe de l’auto-encodeur

Ici le principe est très simple. On construit deux sous-réseaux de neurones convolutionnels : un encodeur noté \mathcal{E} et un décodeur noté \mathcal{D} . L’encodeur projette les données d’entrée dans un espace de plus faible dimension, appelé espace latent (noté \mathcal{Z}), tandis que le décodeur tente de reconstruire les données d’origine à partir de cette représentation réduite.

Autrement dit, un auto-encodeur cherche une fonction $\mathcal{H} = \mathcal{D} \circ \mathcal{E}$ telle que :

$$\mathcal{H}(x) = x \quad \forall x \in \mathbb{R}^n$$

où :

- $\mathcal{E} : \mathbb{R}^n \rightarrow \mathbb{R}^d$ est l’encodeur,
- $\mathcal{D} : \mathbb{R}^d \rightarrow \mathbb{R}^n$ est le décodeur.

Cependant, un petit problème se pose à ce niveau. Dans la section précédente, on a vu que le principe des réseaux de neurones convolutionnels est d’extraire des caractéristiques dans les images tout en réduisant progressivement leur taille spatiale. Or, dans un auto-encodeur, on souhaite également reconstruire l’image dans sa taille initiale, après l’avoir projetée dans l’espace latent.

Pour résoudre cela, on utilise dans le décodeur ce qu’on appelle des **couches de convolution transposée**. Ces couches permettent d’agrandir les dimensions spatiales d’une image, c’est-à-dire de faire l’inverse d’une convolution classique.

Mais comment fonctionne une convolution transposée ?

Contrairement à ce que son nom pourrait laisser penser, une convolution transposée n'est pas simplement l'inverse mathématique d'une convolution. Il s'agit plutôt d'une opération qui applique la même logique qu'une convolution classique, mais dans un sens où la sortie est plus grande que l'entrée.

L'idée est la suivante : pour chaque pixel de l'entrée, on applique un petit noyau (le filtre), mais cette fois en *rétrogradant* sa contribution sur une plus grande surface de l'image de sortie. On peut alors considérer que l'on effectue une convolution classique, mais avec un *padding* négatif ou un *stride* fractionnaire inversé.

Cela permet d'augmenter la taille des images de façon contrôlée et permet au modèle de réapprendre les détails nécessaires pour reconstituer l'image d'origine.

Entraînement de l'auto-encodeur

Une fois le réseau défini, l'objectif est d'ajuster les poids w du réseau pour que la sortie reconstruite soit la plus proche possible de l'entrée. On minimise donc une fonction de coût (erreur quadratique, absolue, ou entropie croisée), par exemple :

$$w^* = \arg \min_w \mathcal{L}(x, w) = \|\mathcal{D}(\mathcal{E}(x)) - x\|$$

Cette fonction est ensuite minimisée par descente de gradient, en ajustant les poids du décodeur puis de l'encodeur.

Auto-encodeur variationnel (VAE) [36]

L'auto-encodeur variationnel est une version améliorée de l'auto-encodeur classique, introduite en 2013 par **Diederik P. Kingma et Max Welling**. L'idée est presque la même, mais on considère le décodeur non pas comme un modèle qui va reconstruire simplement l'entrée de façon exacte, mais plutôt comme un modèle génératif. Et l'encodeur n'est plus un modèle qui projette juste une donnée dans un espace de petite dimension, mais un modèle qui, pour chaque donnée x , la projette dans un espace latent tout en produisant plusieurs z (possibilité). Donc, l'encodeur ne fournit pas une seule représentation abstraite de x , mais une distribution de probabilité $p(z|x)$.

Dans la pratique, pour simplifier le problème, on choisit une distribution bien connue : une distribution gaussienne, dont les paramètres (μ_x , σ_x) sont prédits par l'encodeur. Ainsi, on peut approximer une distribution générale postérieure. On approxime la distribution a priori $p(z)$ par la moyenne empirique des distributions postérieures $p_\phi(z|x_i)$ issues de l'encodeur par :

$$p(z) = \frac{1}{n} \sum_{i=1}^n p_\phi(z|x_i) = \frac{1}{n} \sum_{i=1}^n \mathcal{N}(\mu_{x_i}, \sigma_{x_i})$$

Le problème, c'est que si l'on a peu de données et que z est de grande dimension, on peut mal approximer $p(z)$. Et si l'on a beaucoup de données, cela peut devenir très coûteux en termes de calcul.

Pour éviter cela, on constraint $p(z)$ à suivre une loi a priori. Cette loi doit être la plus proche possible de la vraie (celle dont les paramètres sont prédits par l'encodeur). Pour ce faire, on utilise une mesure de dissimilarité : la divergence de **Kullback-Leibler**.

Définition 2.6.1 Kullback-Leibler

Pour deux distributions de probabilité discrètes P et Q sur un ensemble X , la divergence de Kullback-Leibler de P par rapport à Q est définie par :

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

Ainsi, l'objectif est de faire en sorte que $p(z)$ suive approximativement une loi normale dans \mathbb{R}^d .

On a alors :

$$D_{KL}(q_\phi(z|x) \| p(z)) < \varepsilon$$

Avec :

- $q_\phi(z|x) = \mathcal{N}(\mu, \sigma)$: l'approximation postérieure issue de l'encodeur,
- $p(z) = \mathcal{N}(0, I)$: la loi choisie a priori.

En résumé, nous voulons faire en sorte que la distribution générée par l'encodeur ne soit pas très différente d'une loi bien définie, et que les données générées par le décodeur ne soient pas trop différentes des données d'entrée.

Cela peut se résumer par la fonction coût suivante :

$$L = \mathbb{E}_{q_\phi(z|x)} [\|x - \hat{x}\|^2] + D_{KL}(q_\phi(z|x) \| p(z))$$

Le but est de chercher les paramètres θ et ϕ qui maximisent le logarithme de la vraisemblance sur l'ensemble des données sur \mathcal{D} :

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)]$$

Alors on a :

$$\begin{aligned} \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x)] &= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{p_\theta(z|x)} \right] \\ &= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z|x)} \cdot \frac{q_\phi(z|x)}{p_\theta(z|x)} \right] \\ &= \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] + \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)} \right] \end{aligned}$$

$$\text{On pose : } \mathcal{L}(\theta, \phi, x) = \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \quad \text{et} \quad D_{KL}(q_\phi(z|x) \| p(z)) = \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)} \right]$$

Avec $\mathcal{L}(\theta, \phi, x)$ appelé Evidence Lower Bound(ELBO).

Cela nous amène à exprimer la fonction coût finale sous la forme suivante :

$$\mathcal{L}(\theta, \phi, x) = \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) \| p(z))$$

Enfin, on applique les méthodes classiques (descente de gradient) pour la mise à jour des paramètres θ et ϕ .

2.6.3 Réseaux antagonistes génératifs(GANs)

Dans la section précédente, nous avons introduit un modèle génératif : l'auto-encodeur variationnel. Et l'une des principales difficultés de ce dernier, c'est l'approximation de la distribution $p(z|x)$ dans l'espace latent. Donc on peut voir les GANs comme des VAE, mais au lieu de contraindre $p(z|x)$ à suivre une loi $p(z)$ dans \mathbb{R}^d , on définit dès le départ une loi $p(z)$ dans \mathbb{R}^d .

Ainsi, l'encodeur qui prenait une donnée en entrée et la projetait dans un espace latent en cherchant à capturer des propriétés. Et bien ici, il prend en entrée une distribution aléatoire (bruit) bien définie, et essaie de construire quelque chose de similaire aux données réelles. D'où son nom : générateur. Et le décodeur, qui avant servait à générer des données similaires à celles de l'entraînement, à présent il aura comme objectif de faire la différence entre les données réelles et les données générées par le générateur. D'où son appellation : discriminateur. Donc le générateur apprend à générer des données de plus en plus réalistes pour arriver à tromper le discriminateur, et le discriminateur

développe ses performances pour ne pas se faire avoir.

Dans cette section, nous allons introduire les GANs, comprendre leur fonctionnement, leur impact dans les modèles génératifs, et leur évolution depuis leur introduction jusqu'à présent.

En 2014, Ian Goodfellow et ses collaborateurs ont posé les bases des **réseaux antagonistes génératifs (GANs)**. L'idée principale est la suivante : on considère deux réseaux de neurones qui s'opposent dans un cadre de jeu à somme nulle :

- un **générateur** G , qui prend en entrée une variable aléatoire $z \sim p(z)$ (souvent choisie comme $z \sim \mathcal{N}(0, I)$) et produit une donnée synthétique $G(z)$;
- un **discriminateur** D , qui reçoit soit une donnée réelle $x \sim p_{\text{data}}$, soit une donnée générée $G(z)$, et doit décider si l'échantillon est réel ou artificiel.

Formellement, on cherche à construire une fonction composée $\mathcal{H} = D \circ G$, où :

- $G : \mathbb{R}^d \rightarrow \mathbb{R}^n$ est le générateur,
- $D : \mathbb{R}^n \rightarrow [0, 1]$ est le discriminateur.

Le discriminateur agit comme un classifieur binaire. On souhaite donc qu'il attribue une probabilité $D(x)$ élevée pour une donnée réelle $x \sim p_{\text{data}}$, et une probabilité faible pour une donnée générée $G(z)$. L'apprentissage peut être formulé comme un problème de mesure de similarité entre distributions de probabilité.

Une première idée naturelle serait d'utiliser la **divergence de Kullback–Leibler (KL)** pour évaluer la dissimilarité entre la distribution réelle p_{data} et la distribution générée p_g . Cependant, cette divergence présente une limite importante : elle n'est pas symétrique, c'est-à-dire que pour deux distributions P et Q , on a en général :

$$D_{\text{KL}}(P||Q) \neq D_{\text{KL}}(Q||P).$$

Cette asymétrie peut conduire à une mauvaise estimation de la proximité entre les distributions, et donc à une génération de données de qualité insuffisante.

Pour pallier cette limitation,^[37] ont introduit l'usage d'une autre mesure de dissimilarité : la **divergence de Jensen–Shannon (JS)**. Celle-ci est une version symétrisée et bornée de la divergence de KL, définie comme suit :

$$\text{JS}(P||Q) = \frac{1}{2} D_{\text{KL}}(P||M) + \frac{1}{2} D_{\text{KL}}(Q||M),$$

où $M = \frac{1}{2}(P + Q)$ est la distribution moyenne de P et Q .

En résumé, le rôle du discriminateur est double : d'une part, il doit **maximiser l'espérance de la log-vraisemblance** pour les échantillons issus de la distribution réelle p_{data} , et d'autre part, il doit **minimiser la probabilité d'accepter comme réelles les données synthétiques** générées par G . Mathématiquement, son objectif peut s'écrire comme la maximisation des deux termes suivants :

$$\mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] \quad \text{et} \quad \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))].$$

De son côté, le générateur G cherche à produire des échantillons suffisamment réalistes pour tromper le discriminateur. Cela revient à **minimiser** la probabilité que D rejette les données générées, soit :

$$\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))].$$

En combinant ces deux objectifs opposés, on obtient la formulation complète du problème sous forme d'un **jeu min-max** entre G et D :

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))].$$

Ainsi, l'entraînement d'un GAN peut être interprété comme une compétition dynamique : le discriminateur D apprend à distinguer les vraies données des fausses, tandis que le générateur G améliore progressivement sa capacité à produire des échantillons indiscernables de ceux issus de p_{data} . À l'équilibre, les deux réseaux atteignent un point de convergence où $p_g \approx p_{\text{data}}$ et le discriminateur n'est plus capable de faire mieux que deviner aléatoirement, c'est-à-dire $D(x) \approx 0.5$ pour toute donnée.

2.6.4 Réseaux neuronaux résiduels (ResNet)

Au cours des dernières années, les réseaux neuronaux ont occupé une place de plus en plus importante dans le domaine de l'apprentissage automatique. Plus ces réseaux sont profonds, plus ils se montrent performants. Toutefois, cette profondeur accrue n'est pas sans inconvénients. En effet, lorsque le réseau devient très profond, les gradients ont tendance à disparaître ou à exploser au fur et à mesure de la rétropropagation. Ce phénomène complique considérablement l'entraînement et limite les performances des modèles.

Pour remédier à ce problème, He et al.^[38] ont proposé une nouvelle architecture appelée **réseau résiduel (ResNet)**.

Principe du bloc résiduel

Pour bien comprendre l'apport des réseaux résiduels, rappelons brièvement le fonctionnement d'un réseau de convolution classique. Ce dernier a pour rôle d'extraire progressivement les caractéristiques importantes d'une image à travers une succession de couches convolutives et de couches de pooling, en s'appuyant sur l'algorithme de la descente de gradient. Or, lorsque le réseau est très profond, il arrive fréquemment que les gradients disparaissent (*vanishing gradients*) ou, au contraire, explosent (*exploding gradients*). Dans ces cas, les caractéristiques apprises ne reflètent plus fidèlement l'image initiale, ce qui dégrade les performances du modèle.

La question fondamentale est donc la suivante : *comment construire des réseaux neuronaux profonds tout en évitant la disparition ou l'explosion des gradients ?*

La solution proposée par les réseaux résiduels consiste à introduire des **connexions de saut (skip connections)**. Plutôt que d'empiler simplement les couches convolutives les unes après les autres, un **bloc résiduel** est défini par deux couches convolutives (en général), auxquelles on ajoute directement l'entrée initiale du bloc. Dans ResNet, on reformule l'apprentissage en termes de *fonction résiduelle* $F(x) = H(x) - x$. Ainsi, au lieu d'apprendre directement $H(x)$, le réseau apprend $F(x)$ et on définit :

$$H(x) = F(x) + x.$$

Le bloc résiduel peut donc être représenté par :

$$y = \mathcal{F}(x, W) + x,$$

où $\mathcal{F}(x, W)$ correspond aux transformations effectuées par un petit sous-réseau de poids W (typiquement deux ou trois convolutions). Cependant, une difficulté supplémentaire apparaît lorsque la sortie $F(x)$ d'un bloc résiduel n'a pas la même dimension que l'entrée x . Dans ce cas, l'addition $F(x) + x$ ne peut pas être effectuée directement, car les tenseurs ne sont pas alignés en termes de dimensions spatiales ou de profondeur de canaux.

Pour résoudre ce problème, [38] ont utilisées la **projection linéaire (Convolution 1×1)** : une approche plus efficace consiste à appliquer une convolution 1×1 à x , notée W_s , afin de le projeter dans l'espace latent approprié. On obtient alors :

$$H(x) = F(x) + W_s x,$$

où W_s est appris conjointement avec les autres paramètres du réseau. Cette méthode permet d'adapter à la fois la profondeur et, si nécessaire, la résolution spatiale.

2.7 Conclusion

Dans ce chapitre, nous avons vu les bases du deep learning en commençant par un rappel au machine learning. Nous avons ensuite retracé l'évolution historique des réseaux de neurones, depuis les premiers modèles inspirés du fonctionnement biologique jusqu'aux architectures modernes qui dominent aujourd'hui la recherche et les applications industrielles. Cette progression nous a permis de comprendre comment les notions de perceptron, de perceptron multicouche et d'algorithmes de rétropropagation ont jeté les fondations des réseaux neuronaux profonds.

Nous avons également présenté les principaux modèles actuels, tels que les réseaux convolutionnels, les autoencodeurs, les GANs et les réseaux résiduels, en insistant sur leurs mécanismes de fonctionnement et leurs domaines d'application. L'analyse de ces différentes architectures met en évidence la capacité du deep learning à traiter des données complexes, à extraire automatiquement des caractéristiques pertinentes et à fournir des performances remarquables dans des tâches variées.

En somme, ce chapitre nous a permis d'acquérir une compréhension claire des concepts fondamentaux du deep learning et de ses principales méthodes. Cette base théorique constitue un point d'appui indispensable pour la suite de ce mémoire, où nous proposerons une approche concrète de classification d'images appliquée au tri des déchets, en mobilisant les techniques étudiées ici.

Approche Proposée

3.1 Introduction

Dans ce chapitre, nous présentons la démarche suivie pour mener notre étude, depuis la revue de la littérature jusqu'aux résultats obtenus. Nous commençons par examiner les travaux existants dans le domaine, afin de mettre en évidence les approches déjà proposées et leurs principales limites. Cette étape permet de situer notre recherche et de justifier les choix méthodologiques adoptés.

Ensuite, nous décrivons la méthodologie utilisée. Nous détaillons les bases de données mobilisées, les étapes de préparation et de traitement des images, ainsi que l'architecture du modèle développé. Nous expliquons également les stratégies d'entraînement et les critères d'évaluation retenus pour mesurer la performance de notre approche.

Enfin, la dernière partie est consacrée aux résultats expérimentaux. Nous y analysons les performances du modèle, en soulignant à la fois ses points forts et ses limites. Cette analyse permet de mieux comprendre les apports de notre travail et ouvre des perspectives pour de futures améliorations.

3.2 Revue de la littérature

Depuis leur succès retentissant lors de la compétition ImageNet en 2012, les réseaux de neurones convolutionnels (CNN) sont devenus le pilier central de la vision par ordinateur. De nombreux chercheurs considèrent aujourd'hui ces approches comme une solution efficace et innovante pour la gestion et le tri automatisé des déchets. Pour cela, deux grandes approches sont largement utilisées : les approches basées sur le transfert learning et celles développées sans recours au transfert learning.

3.2.1 Approches par transfert learning

Le *transfert learning* consiste à adapter des modèles pré-entraînés (ImageNet) aux tâches de tri des déchets. Plusieurs études confirment son efficacité :^[55] ont exploité ResNet50V2 (95,1 % de précision), tandis que^[56] ont montré la supériorité de YOLOv8 pour les déchets électroniques. De même,^[57] proposent YOLOX-DW (mAP@0.5 de 99,16 %), et^[58] démontrent l'intérêt de Faster R-CNN avec Inception pour des scénarios pratiques.

D'autres travaux confirment la performance des architectures EfficientNet et ResNet :^[59] privilient EfficientNetV2S pour son compromis entre précision (96,07 %) et impact environnemental,

tandis que^[60] atteignent 81,2 % avec EfficientNet-B0 affiné. Dans la même logique, ^[61] montrent la robustesse de DenseNet121 (96,43 %). Enfin, des optimisations récentes confirment l'intérêt de YOLO : ^[62] améliorent YOLOv8 par CBAM et SE (92,73 %), et ^[63] comparent YOLOv4 et sa version « tiny » (89,59 % vs 81,84 %).

3.2.2 Approches sans transfert learning

Parallèlement, plusieurs chercheurs développent des architectures sur mesure, souvent plus légères et adaptées à des contextes locaux.^[64] combinent CNN et GLSTM pour atteindre 97,55 %. De leur côté, ^[65] (5 couches convolutionnelles) atteignent 80,88 % pour une classification binaire, et ^[66] rapportent 91 % sur le biowaste.^[67] intègrent un CNN dans un système robotisé avec 99 % de précision, tandis que^[68] proposent un modèle simple atteignant 85 %.

Ces résultats montrent que, bien que moins généralisables, les architectures personnalisées offrent des solutions rapides, légères et adaptées aux besoins spécifiques.

3.2.3 Limites des méthodes existantes

Tout d'abord, la majorité des approches repose sur des ensembles de données restreints, tels que *TrashNet* ou *CompostNet*, qui ne couvrent pas toute la diversité des déchets rencontrés en conditions réelles. Cette dépendance limite fortement la capacité de généralisation des modèles proposés.

De plus, de nombreux travaux exploitent des architectures lourdes et exigeantes en ressources de calcul, comme YOLO, ^{[62], [63]} EfficientNet et ResNet. ^{[55]-[61]} Ces approches, bien que performantes, sont difficilement adaptées à des environnements contraints en ressources (systèmes embarqués, collectivités locales, etc.).

Enfin, même les travaux qui s'appuient sur des architectures plus légères ou sur mesure montrent une forte dépendance à des jeux de données limités, qui ne reflètent pas toujours la réalité locale. C'est le cas par exemple des études de.^{[64]-[68]} Ainsi, les systèmes développés peuvent perdre en efficacité lorsqu'ils sont appliqués dans des contextes différents de ceux des bases utilisées pour l'entraînement.

3.3 Méthodologie

Dans cette section, nous présentons l'ensemble des méthodes utilisées pour répondre aux objectifs de cette étude. Nous commençons par décrire la base de données employée, puis nous détaillons l'architecture du modèle proposé. Nous exposons également les différentes stratégies d'entraînement mises en œuvre, ainsi que les techniques d'évaluation adoptées. Enfin, nous concluons par une comparaison de notre approche avec celles de l'état de l'art, tout en justifiant notre démarche.

3.3.1 Description de la base de données

L'ensemble de données utilisé dans cette étude comprend cinq classes principales : canette, organique, plastique, textile et verre. Ces catégories ont été choisies en fonction de la quantité et de la qualité des déchets produits à Moroni, ^[5] des besoins identifiés dans des initiatives locales de recyclage, ^{[5], [25]} ainsi que de notre expertise acquise sur le terrain.

Les données proviennent principalement de quatre grandes bases de données open source (*TrashNet*, ^[69] *Waste Classification Data*, ^[70] *Garbage Classification*^[71] et *Recyclable and Household Waste Classification*^[72]), auxquelles s'ajoute un enrichissement issu d'images collectées localement dans différents dépôts de déchets de la ville de Moroni.

Cependant, une partie des images que nous avons collectées n'ont pas été utilisées pour l'entraînement ni pour les tests. Elles sont réservées à une tâche bien spécifique : l'évaluation finale des modèles.

Dans le Tableau 3.1, nous présentons le nombre total d'images sélectionnées par base de données et par classe.

Table 3.1 – Répartition des images sélectionnées par classe et par base de données

Base de données	Canette	Organique	Plastique	Textile	Verre	Total
TrashNet	0	0	482	0	501	983
Waste Classification Data	533	1622	533	800	250	3738
Garbage Classification	0	720	865	1765	1925	5275
Recyclable and Household Waste Classification	750	0	600	0	0	1350
Données locales (Moroni)	509	0	103	100	80	792
Total	1792	2342	2583	2665	2756	12138

Les figures 3.1 et 3.2 présentent respectivement la répartition en pourcentage des images sélectionnées selon les bases de données et selon les classes. Cette visualisation permet de mettre en évidence la contribution de chaque source ainsi que la représentation relative de chaque catégorie de déchets dans l'ensemble global.

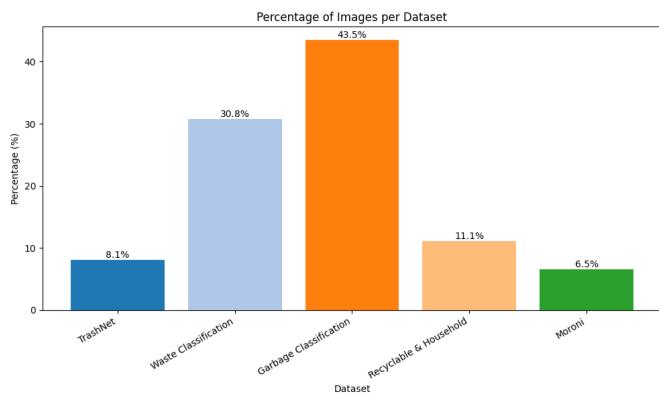


Figure 3.1 – Répartition en pourcentage des images par base de données.

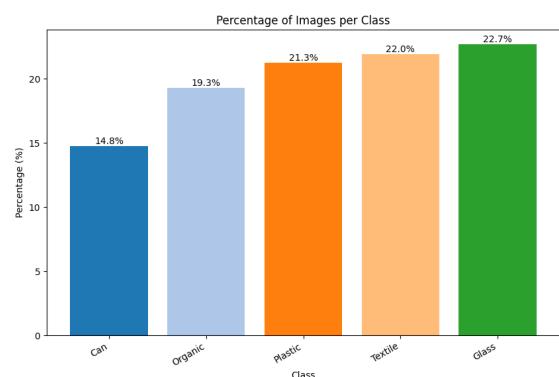


Figure 3.2 – Répartition en pourcentage des images par classe.

Le tableau 3.2 présente le nombre d'images réservé pour chaque classe lors de la validation finale.

Table 3.2 – Nombre d'images réservées pour chaque classe lors de la validation finale.

ID	Classe	Nombre d'images
0	Canette	100
1	Organique	108
2	Plastique	170
3	Textile	77
4	Verre	82

Les données ont été rechargées à l'aide d'une classe que nous avons développée spécifiquement afin de prétraiter et de visualiser les images de manière plus efficace et flexible. Cette classe prend en

entrée la liste des chemins des images, les extensions de fichiers à conserver (pour éviter d'importer d'autres types de fichiers indésirables), ainsi que les dimensions souhaitées pour les images. Nous avons implémenté toutes les méthodes nécessaires pour le traitement des données, notamment le redimensionnement ($240 \times 240 \times 3$), l'encodage des labels, et l'attribution automatique de l'identifiant de classe en fonction de l'ordre des chemins fournis. Des fonctions de réduction de dimensionnalité ont également été intégrées, si nécessaire, afin d'optimiser l'efficacité et la performance du modèle.



Figure 3.3 – Exemples d'images d'entraînement

Par ailleurs, une étape essentielle de notre préparation des données a consisté à effectuer une augmentation des données (data augmentation). Cette technique vise à introduire davantage de diversité dans l'ensemble d'entraînement, en appliquant des transformations aléatoires telles que des rotations, des traductions, des zooms, des inversions, des modifications de luminosité, etc. Ces opérations permettent d'augmenter la robustesse du modèle et d'améliorer sa capacité de généralisation face à des images variées en conditions réelles.

Le tableau 3.3 ci-dessous résume de manière détaillée les transformations appliquées ainsi que leurs paramètres respectifs.

Table 3.3 – Paramètres d’augmentation des données et leurs descriptions.

Paramètre	Valeur	Description
Rescale	1./255	Normalise les valeurs des pixels pour l’entraînement.
Rotation range	35°	Effectue des rotations aléatoires pour simuler différentes orientations.
Width shift range	0.2	Décale horizontalement les images pour plus de variabilité spatiale.
Height shift range	0.2	Décale verticalement les images pour simuler des translations.
Shear range	0.2	Applique une transformation en cisaillement pour simuler des déformations.
Zoom range	0.3	Effectue des zooms aléatoires pour diversifier les échelles.
Horizontal flip	Oui	Retourne les images horizontalement, utile pour la symétrie.
Vertical flip	Oui	Retourne les images verticalement pour augmenter la variabilité.
Brightness range	[0.6, 1.4]	Ajuste la luminosité pour simuler différents éclairages.
Channel shift range	30.0	Modifie aléatoirement les canaux de couleur pour plus de robustesse.
Fill mode	nearest	Méthode de remplissage des pixels vides après transformation.

Dans les figures 3.4 et 3.5, nous présentons respectivement le nombre d’images utilisées par classe et leur répartition en pourcentage. La séparation en 80 % pour l’entraînement et 20 % pour le test a été appliquée de manière uniforme à l’ensemble des données. Cette analyse permet d’évaluer et de visualiser d’éventuels déséquilibres entre les classes, aussi bien dans les jeux d’entraînement que de test, et ainsi de mieux comprendre la distribution globale des données utilisées.

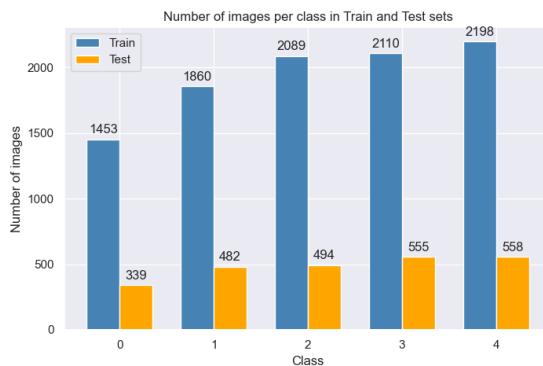


Figure 3.4 – Nombre d’images utilisées par classe.

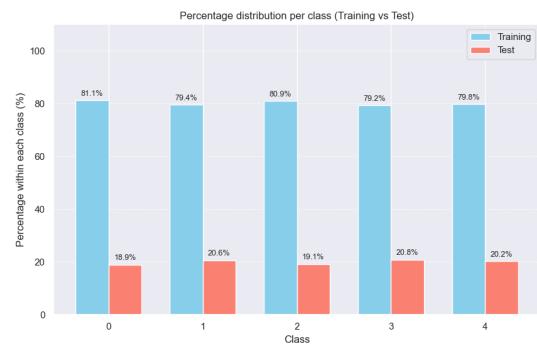


Figure 3.5 – Répartition en pourcentage des images par classe.

Ces figures mettent en évidence certains déséquilibres entre les différentes classes, tant dans l’ensemble d’entraînement que dans l’ensemble de test. Il est donc important de tenir compte de ces écarts lors de l’évaluation des performances, en s’appuyant sur des métriques adaptées qui ne se limitent pas uniquement à la précision globale. Par exemple, l’utilisation du score **F1**, de la **moyenne pondérée** ou encore de l’aire sous la courbe **ROC** permet d’obtenir une évaluation plus équilibrée et représentative des performances du modèle sur chaque classe.

3.3.2 Architecture du modèle

Dans cette section, nous détaillons l’architecture du modèle proposé, en tenant compte des contraintes matérielles lors d’un éventuel déploiement.

Le modèle prend en entrée des images de taille $240 \times 240 \times 3$, normalisées avant traitement. L’architecture se compose de cinq blocs convolutifs, chacun incluant une couche de convolution, une normalisation par lot (*Batch Normalization*), une couche de pooling et une couche de dropout

pour réduire le surapprentissage. Toutes ces couches utilisent la fonction d'activation ReLU, définie par :

$$\text{ReLU}(x) = \max(0, x).$$

Après ces blocs, le modèle comprend quatre couches entièrement connectées (*dense*) qui transforment les caractéristiques extraites en un vecteur final de classification. La dernière couche dense est équipée d'une fonction d'activation Softmax, exprimée comme suit :

$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)},$$

où z_i est la sortie correspondant à la classe i et K est le nombre total de classes.

Afin de limiter la complexité et d'éviter le surapprentissage, nous appliquons une régularisation L2 aux couches entièrement connectées.

Dans un souci d'efficacité et pour proposer une version encore plus légère, nous avons également conçu une variante du modèle où la couche *Flatten* est remplacée par une couche *Global Average Pooling* (notée *AverageFlatten*). Cette modification permet de réduire significativement le nombre de paramètres tout en conservant des performances comparables, facilitant ainsi le déploiement sur des dispositifs à ressources limitées.

Dans le tableau 3.4, nous présentons en détail l'architecture complète du modèle.

Table 3.4 – Architecture détaillée du modèle proposé

Couche	Type	Paramètres principaux	Shape de sortie
Entrée	Image	240×240×3	(None, 240, 240, 3)
Bloc 1	Conv2D	32 filtres, kernel 3×3	(None, 240, 240, 32)
	BatchNorm	-	(None, 240, 240, 32)
	MaxPooling	2×2	(None, 120, 120, 32)
Bloc 2	Conv2D	64 filtres, kernel 3×3	(None, 120, 120, 64)
	BatchNorm	-	(None, 120, 120, 64)
	MaxPooling	2×2	(None, 60, 60, 64)
Bloc 3	Conv2D	128 filtres, kernel 3×3	(None, 60, 60, 128)
	BatchNorm	-	(None, 60, 60, 128)
	MaxPooling	2×2	(None, 30, 30, 128)
Bloc 4	Conv2D	128 filtres, kernel 3×3	(None, 30, 30, 128)
	BatchNorm	-	(None, 30, 30, 128)
	MaxPooling	2×2	(None, 15, 15, 128)
Bloc 5	Conv2D	132 filtres, kernel 3×3	(None, 15, 15, 132)
	BatchNorm	-	(None, 15, 15, 132)
	MaxPooling	2×2	(None, 7, 7, 132)
Flatten	-	-	(None, 6468)
Dense 1	Dense	512 neurones, ReLU	(None, 512)
Dense 2	Dense	256 neurones, ReLU	(None, 256)
Dense 3	Dense	128 neurones, ReLU	(None, 128)
Dense 4	Dense	100 neurones, ReLU	(None, 100)
Dense 5	Dense	64 neurones, ReLU	(None, 64)
Sortie	Dense	5 classes, Softmax	(None, 5)

3.3.3 Stratégie d'entraînement

La stratégie d'entraînement constitue une étape cruciale dans le développement d'un modèle de classification performant.^[73] Elle comprend notamment la configuration fine des hyperparamètres, tels que le *learning rate* initial, ainsi que l'application de stratégies dynamiques permettant d'ajuster ce taux au cours de l'apprentissage.

Le *learning rate* (η) joue un rôle fondamental dans la vitesse et la stabilité de convergence du modèle. Celui-ci peut être ajusté de manière progressive, selon un schéma d'ordonnancement, souvent formulé par :

$$\eta_t = \eta_0 \times \gamma^{\lfloor \frac{t}{s} \rfloor},$$

où η_0 est le taux d'apprentissage initial, γ est le facteur de réduction (par exemple, 0.95), t est l'époque actuelle, et s l'intervalle de réduction.

Le modèle est entraîné en minimisant une fonction coût, notée $L(\theta)$, qui mesure l'erreur entre les prédictions du modèle et les vraies étiquettes. Pour un problème de classification multiconcours, on utilise couramment l'entropie croisée (cross-entropy loss) :

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_{i,k} \log(\hat{y}_{i,k}),$$

où N est le nombre d'exemples, K le nombre de classes, $y_{i,k}$ la vérité terrain (1 si l'exemple i appartient à la classe k , sinon 0), et $\hat{y}_{i,k}$ la probabilité prédictive.

Concernant l'optimiseur, nous avons choisi **RMSprop**, qui améliore la descente de gradient classique (SGD) en adaptant dynamiquement le taux d'apprentissage pour chaque paramètre. Les mises à jour des poids s'effectuent comme suit :

$$v_t = \rho v_{t-1} + (1 - \rho) \nabla L(\theta_t)^2,$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \nabla L(\theta_t),$$

où θ_t désigne les poids au pas t , $L(\theta)$ la fonction coût, ρ un coefficient de décroissance, et ϵ un petit terme pour éviter la division par zéro.

Enfin, pour prévenir le surapprentissage, nous avons employé la stratégie d'*early stopping*, qui interrompt l'entraînement lorsque la précision de validation cesse de s'améliorer, ainsi qu'un mécanisme de sauvegarde automatique des meilleurs poids (*model checkpoint*).

Le tableau 3.5 résume en détail l'ensemble de ces configurations utilisées pour l'entraînement.

Table 3.5 – Résumé des hyperparamètres et des stratégies d'entraînement utilisées

Paramètre	Valeur / Description
Optimiseur	RMSprop
Taux d'apprentissage initial	10^{-4}
Fonction de perte	Sparse categorical cross-entropy
Métrique	Accuracy
Early stopping	Patience = 30, monitor = val_accuracy
Reduce LR on plateau	Facteur = 0.2, patience = 15, monitor = val_loss
Model checkpoint	Sauvegarde si val_accuracy s'améliore
Nombre d'époques max	200
Batch size	32
Callback verbose	Activé (verbose=1)

Lors de l'analyse exploratoire, nous avons constaté un déséquilibre entre les classes. Pour limiter son impact sur l'apprentissage, nous avons appliqué un **pondérage des classes** dans la fonction de perte, afin de donner plus de poids aux catégories moins représentées.

Le poids de chaque classe i est défini par :

$$w_i = \frac{\text{moyenne}(n_c)}{n_i},$$

où n_i représente le nombre d'échantillons dans la classe i et $\text{moyenne}(n_c)$ correspond au nombre moyen d'échantillons par classe.

La fonction de perte pondérée devient alors :

$$\mathcal{L}_{\text{pondérée}} = -\frac{1}{N} \sum_{i=1}^N w_{y_i} \sum_{k=1}^K y_{i,k} \log(\hat{y}_{i,k}),$$

où w_{y_i} désigne le poids associé à la classe réelle de l'exemple i . Cette approche renforce la contribution des classes peu représentées pendant l'entraînement.

Le tableau 3.6 présente les coefficients obtenus pour chaque classe.

Table 3.6 – Coefficients de pondération des classes.

ID	Classe	Coefficient w_i
0	Canette	1.3365
1	Organique	1.0441
2	Plastique	0.9296
3	Textile	0.9204
4	Verre	0.8835

3.3.4 Évaluation des performances

Afin d'évaluer de manière rigoureuse la performance du modèle et d'obtenir une vision fidèle de son comportement en conditions réelles, nous avons préparé l'ensemble des données initiales en suivant une répartition standard : 80 % pour l'entraînement et 20 % pour le test. En complément, une partie des données collectées localement a été réservée spécifiquement pour la validation finale, afin de simuler un scénario d'utilisation réelle.

Pour mesurer les performances, plusieurs métriques complémentaires ont été utilisées : l'accuracy (exactitude globale), la précision, le rappel, le F1-score, ainsi que la moyenne des précisions moyennes (mAP). Ces métriques tiennent compte des déséquilibres entre classes observés lors de l'exploration initiale des données, garantissant une évaluation plus équilibrée et représentative.

Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

où TP, TN, FP et FN désignent respectivement les vrais positifs, vrais négatifs, faux positifs et faux négatifs.

Précision

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

Rappel

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

F1-score

$$\text{F1} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

mAP (mean Average Precision)

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i$$

où AP_i est l'aire sous la courbe précision-rappel pour la i -ème classe, et N le nombre total de classes.

3.4 Résultats

Dans cette section, nous présentons les résultats issus de l'entraînement et de l'évaluation des deux modèles proposés. Le premier correspond à l'architecture initiale décrite précédemment, intégrant une couche *Flatten*. Le second est une variante plus légère, où cette couche est remplacée par une *Global Average Pooling*. L'objectif de cette comparaison est d'évaluer non seulement les performances de classification, mais aussi l'efficacité computationnelle et la légèreté des modèles.

3.4.1 Résultats de l'entraînement

Les deux modèles ont été entraînés sur l'ensemble d'apprentissage avec une limite maximale de 200 époques, en appliquant une stratégie d'*early stopping* et une réduction automatique du taux d'apprentissage. Le Modèle 1 a nécessité un temps d'entraînement de **1 jour 14 heures 59 minutes et 25 secondes**, avec un taux d'apprentissage décroissant de 10^{-4} jusqu'à 10^{-6} . Le Modèle 2, plus léger, a convergé en **1 jour 12 heures 17 minutes et 4 secondes**, avec un taux d'apprentissage réduit progressivement de 10^{-4} à 10^{-7} , et a terminé son entraînement au bout de 190 époques.

Les figures 3.6 et 3.7 illustrent respectivement l'évolution de la fonction de perte et de la précision pour chaque modèle. Ces courbes permettent d'analyser la rapidité de convergence et la stabilité de l'entraînement.

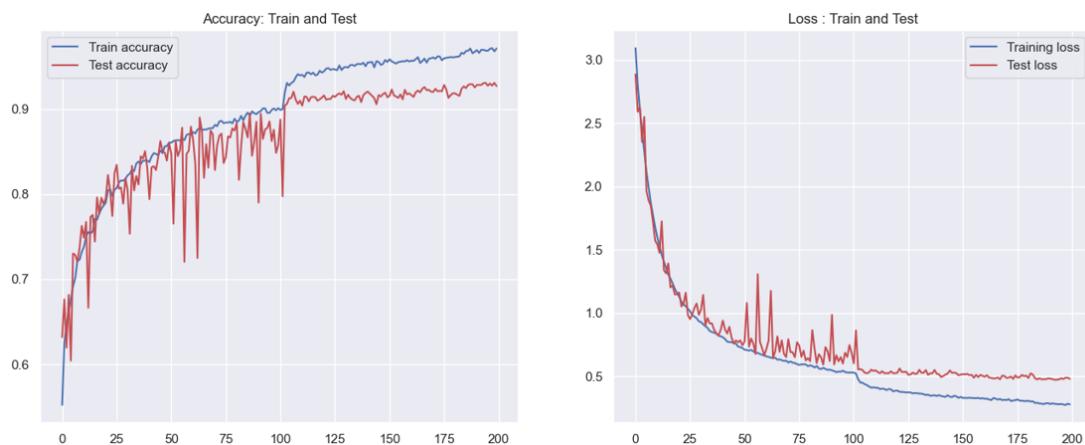


Figure 3.6 – Courbe d'apprentissage du Modèle 1 (Flatten).

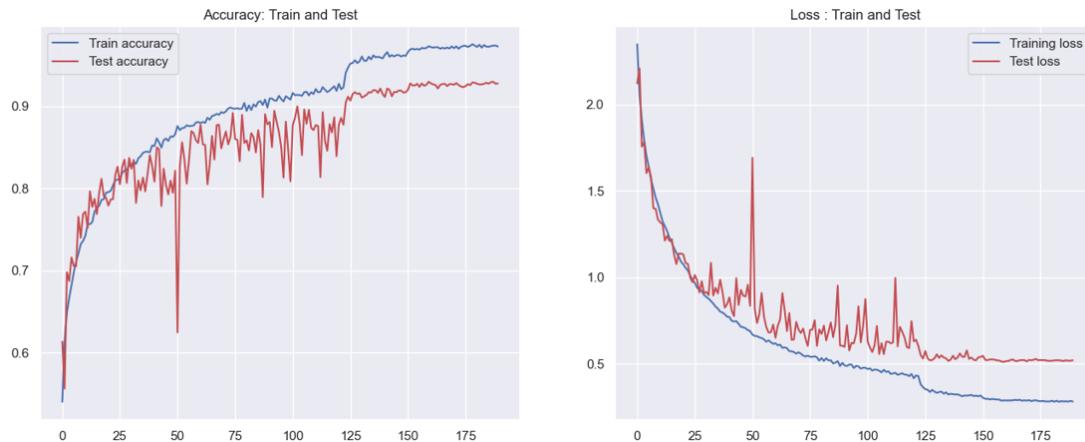


Figure 3.7 – Courbe d'apprentissage du Modèle 2 (AvgPooling).

3.4.2 Résultats détaillés par classe

Le tableau 3.7 présente les scores de précision, rappel et F1 pour chaque classe de déchets sur le jeu de test. Les résultats montrent que les deux modèles atteignent des performances élevées, supérieures à 88 % pour toutes les classes.

Table 3.7 – Résultats détaillés par classe pour les deux modèles (jeu de test).

Classe	Modèle 1 (Flatten)			Modèle 2 (AvgPooling)		
	Précision	Rappel	F1	Précision	Rappel	F1
Canette	0.8870	0.9027	0.8947	0.9115	0.9115	0.9115
Organique	0.9481	0.9481	0.9481	0.9333	0.9585	0.9458
Plastique	0.8912	0.8785	0.8848	0.9097	0.8765	0.8928
Textile	0.9659	0.9712	0.9686	0.9454	0.9676	0.9564
Verre	0.9406	0.9373	0.9390	0.9382	0.9247	0.9314

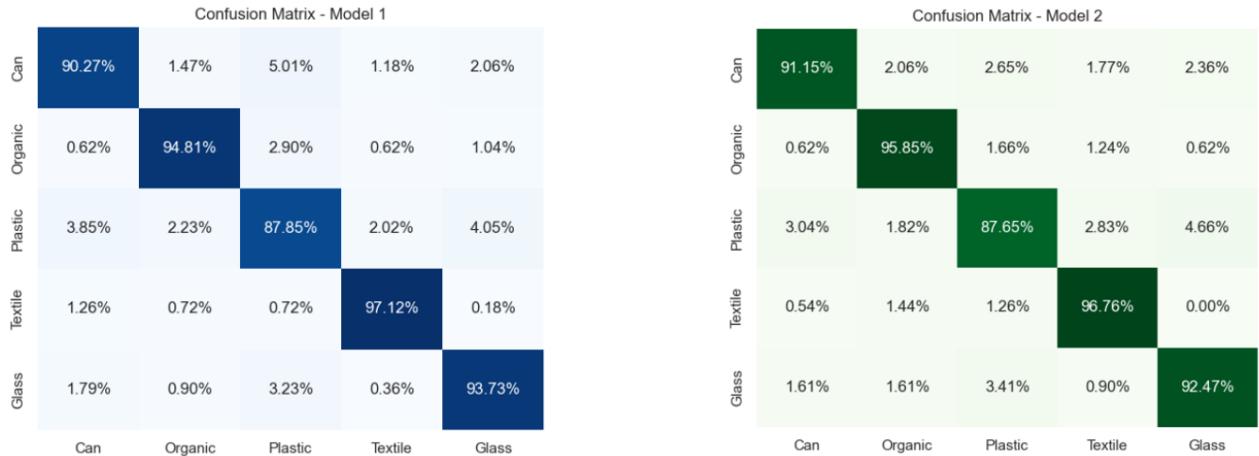
3.4.3 Matrice de confusion

Afin de mieux visualiser la répartition des prédictions correctes et erronées, nous présentons les matrices de confusion pour chacun des deux modèles. La Figure 3.8a illustre la matrice de confusion du Modèle 1, et la Figure 3.8b celle du Modèle 2.

Ces matrices permettent d'identifier les principales confusions entre classes, par exemple entre plastique et canette, et confirment la robustesse générale des deux modèles.

3.4.4 Résumé global des performances

Le tableau 3.8 synthétise les résultats globaux en termes d'*accuracy*, de F1-macro, de complexité et d'efficacité d'inférence. Les deux modèles présentent des performances très proches, avec une précision globale supérieure à 92 %. Le Modèle 1 se distingue par une *accuracy* légèrement plus élevée (93,04 %), alors que le Modèle 2 atteint un meilleur score F1-macro (92,76 %). En revanche, le Modèle 2 est nettement plus léger (près de 7 fois moins de paramètres) et plus rapide en prédiction.



(a) Matrice de confusion du Modèle 1 (Flatten).

(b) Matrice de confusion du Modèle 2 (AvgPooling).

Figure 3.8 – Comparaison des matrices de confusion des deux modèles sur les données de test.

Table 3.8 – Résumé comparatif des deux modèles.

Indicateur	Modèle 1 (Flatten)	Modèle 2 (AvgPooling)
Accuracy	0.9304	0.9296
Macro F1-score	0.9270	0.9276
Nombre de paramètres	3 806 525	562 493
Temps d'apprentissage	1j 14h 59m 25s	1j 12h 17m 04s

3.4.5 Évaluation sur les données locales

L'évaluation du modèle sur les données locales a révélé des résultats relativement décevants. En effet, le score F1 global est resté inférieur à 68 % pour la majorité des classes, à l'exception de la classe *canette*, pour laquelle le modèle a atteint environ 80 %. Cette faible capacité de généralisation peut s'expliquer par la nature des données d'entraînement, issues principalement de bases publiques disponibles en ligne. Ces bases présentent certaines limites : les images sont souvent bien centrées, de résolution relativement élevée, et contiennent généralement un seul objet par image. Ces conditions idéalisées ne reflètent pas fidèlement la complexité des situations rencontrées dans un contexte réel.

Afin d'améliorer la capacité de généralisation du modèle, nous avons choisi de réorienter cette étape vers une approche de **transfert learning**, en utilisant le modèle initial comme point de départ. Pour cela, nous avons constitué un sous-échantillon de 200 images par classe, tirées aléatoirement à partir des données, afin d'entraîner uniquement les trois dernières couches de nos modèles. Parallèlement, 70 % des données locales ont été utilisées pour affiner le modèle, tandis que les 30 % restants ont été réservés à l'évaluation finale. De plus, un lot complémentaire de 20 images par classe, issu des données de test initiales, a été intégré afin de renforcer la robustesse de l'évaluation.

Concernant le prétraitement, l'augmentation des données ainsi que les différentes stratégies d'entraînement, nous avons conservé les mêmes méthodes que celles appliquées lors de l'entraînement initial.

Dans ce qui suit, nous présentons les résultats obtenus après cette phase de transfert learning, ainsi qu'une analyse comparative par rapport à l'évaluation initiale.

Les figures 3.9 et 3.10 présentent respectivement les courbes d'apprentissage des modèles 1 et 2, issus du processus de transfert learning.

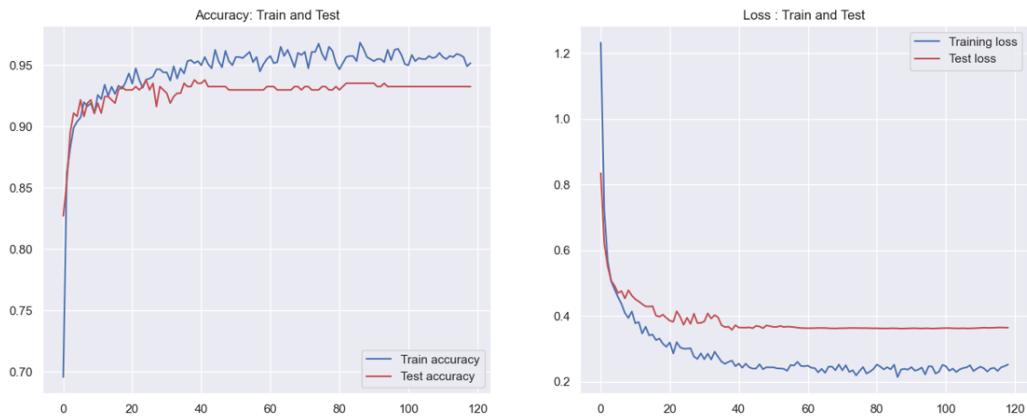


Figure 3.9 – Courbe d'apprentissage du modèle 1 obtenu par transfert learning.

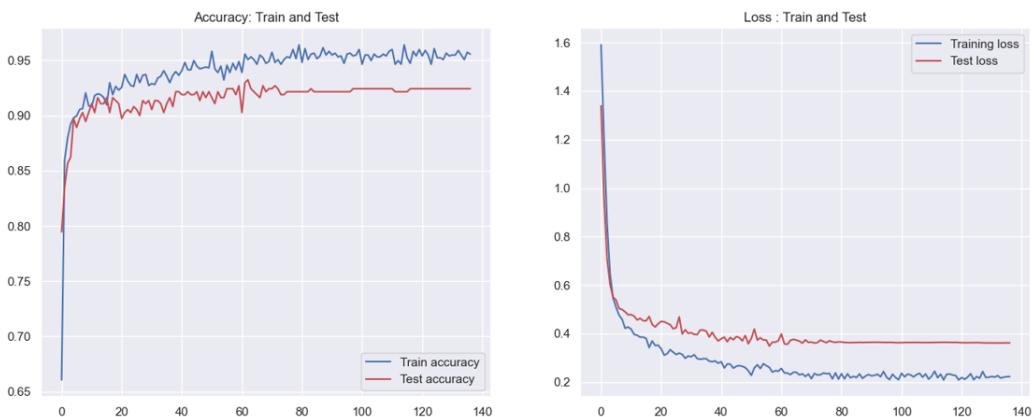


Figure 3.10 – Courbe d'apprentissage du modèle 2 obtenu par transfert learning.

Le tableau 3.9 présente les scores détaillés de précision, rappel et F1 pour chaque classe de déchets, comparant les deux modèles testés sur 30 % des données locales.

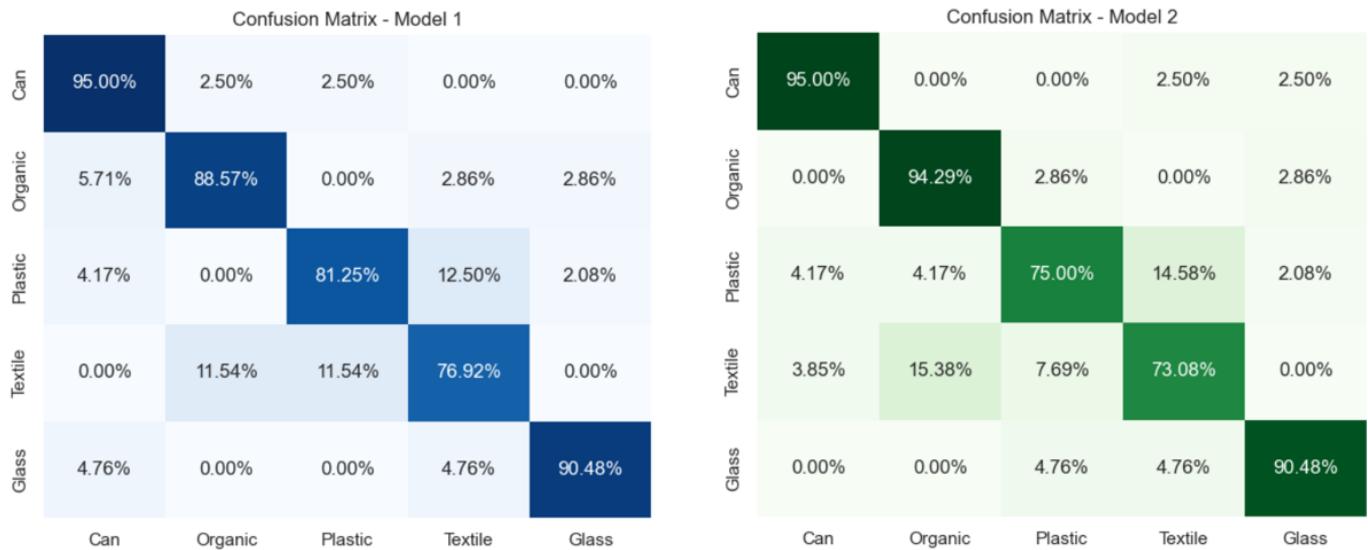
On observe que les deux modèles atteignent de bonnes performances, avec des scores supérieurs à 70 % pour toutes les classes. Le Modèle 1 (Flatten) présente des résultats légèrement supérieurs en F1-score pour certaines classes, tandis que le Modèle 2 (AvgPooling) offre des résultats comparables avec un nombre de paramètres réduit et un temps de prédiction plus rapide, ce qui peut être avantageux pour un déploiement en temps réel.

Afin de mieux visualiser la répartition des prédictions correctes et erronées sur les 30 % des données locales, nous présentons les matrices de confusion. La Figure 3.11 illustre les matrices de confusion pour les deux modèles.

Pour comparer de manière plus approfondie les performances des modèles avec et sans transfert learning, nous réalisons une dernière analyse basée sur le pourcentage d'images correctement classées et d'images mal classées figure 3.12.

Table 3.9 – Résultats détaillés par classe pour les deux modèles (sur les 30 % des données locales).

Classe	Modèle 1 (Flatten)			Modèle 2 (AvgPooling)		
	Précision	Rappel	F1	Précision	Rappel	F1
Canette	0.8837	0.9500	0.9157	0.9268	0.9500	0.9383
Organique	0.8857	0.8857	0.8857	0.8462	0.9429	0.8919
Plastique	0.9070	0.8125	0.8571	0.9000	0.7500	0.8182
Textile	0.7143	0.7692	0.7407	0.6786	0.7308	0.7037
Verre	0.9048	0.9048	0.9048	0.8636	0.9048	0.8837
Accuracy globale	0.8647			0.8529		
Macro F1-score	0.8608			0.8472		



(a) Matrice de confusion du Modèle 1 (Flatten).

(b) Matrice de confusion du Modèle 2 (AvgPooling).

Figure 3.11 – Comparaison des matrices de confusion des deux modèles sur les 30 % des données locales.

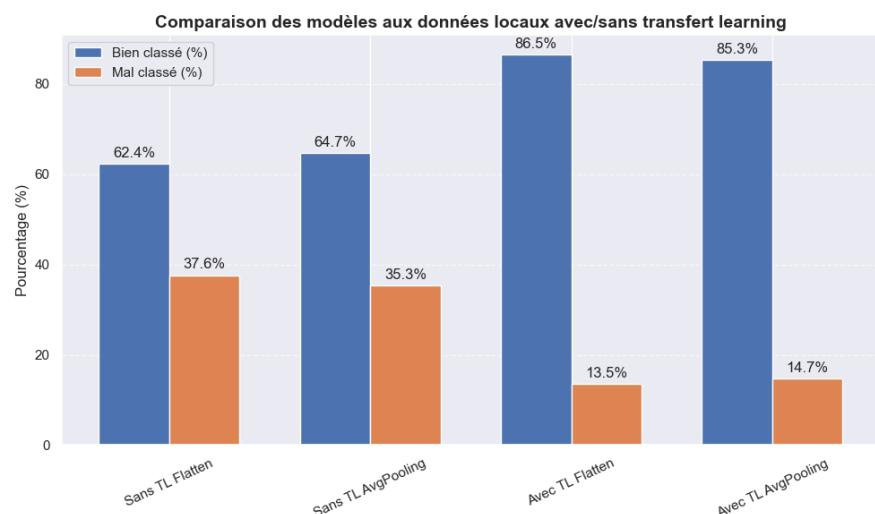


Figure 3.12 – Pourcentage d’images correctement classées et mal classées pour les modèles avec et sans transfert learning.

3.5 Conclusion

Dans ce chapitre, nous avons présenté l'approche proposée pour aborder la problématique du tri des déchets à l'aide de modèles d'apprentissage profond. Notre démarche a couvert l'ensemble du processus, depuis la collecte et le prétraitement des données jusqu'à l'entraînement et l'évaluation des modèles.

L'évaluation sur les données locales a toutefois révélé des limites, avec des performances relativement faibles pour la majorité des classes. Cette situation, liée aux différences entre les bases de données publiques et les conditions réelles, a mis en évidence les défis de la généralisation.

Afin de surmonter ces difficultés, nous avons opté pour une approche de *transfert learning*, qui s'est révélée plus adaptée aux spécificités des données locales. Cette étape nous a permis non seulement d'améliorer les performances, mais aussi d'identifier des points critiques qui méritent une analyse plus approfondie.

Le chapitre suivant sera consacré à l'examen de ces limites et à la proposition de pistes de résolution, afin de renforcer l'efficacité et la pertinence de notre approche.

Améliorations du système

4.1 Introduction

Les méthodes de classification d'images connaissent aujourd'hui un essor considérable et sont largement exploitées pour le tri automatisé des déchets. Ces approches, fondées sur l'apprentissage profond, produisent généralement des résultats satisfaisants et ouvrent de nouvelles perspectives pour la gestion intelligente et durable des déchets.

Cependant, leur mise en œuvre dans des contextes réels révèle plusieurs limites importantes. Tout d'abord, la capacité de généralisation des modèles reste insuffisante face à des données issues d'environnements variés, ce qui entraîne une baisse de la performance. Ensuite, des erreurs de pré-diction inévitables dans des situations complexes, notamment lorsque le modèle est confronté à des images qui ne correspondent à aucune des classes présentes dans l'ensemble d'entraînement initial.

Ce chapitre est consacré à l'analyse de ces limites et à la présentation des différentes stratégies envisagées pour améliorer l'efficacité et la pertinence du système. L'objectif est de proposer des pistes permettant non seulement d'accroître la précision du modèle, mais aussi de renforcer son adaptabilité aux conditions réelles d'utilisation.

4.2 Amélioration de la classification

Dans le chapitre précédent, nous avons décrit le processus suivi pour entraîner et évaluer nos modèles de classification. Les résultats obtenus sur les données publiques utilisées lors de l'apprentissage initial ont été satisfaisants, mais ces performances se sont nettement dégradées lorsqu'il s'est agi d'évaluer les modèles sur les données locales. Cette différence met en évidence un problème classique : un modèle peut être performant sur les données pour lesquelles il a été entraîné, mais se montrer moins efficace lorsqu'il est confronté à des données provenant d'un environnement différent.

Pour réduire cet écart, nous avons mis en place une approche de *transfert learning*. Le principe est le suivant : nous avons repris les modèles que nous avions déjà entraînés lors de la phase initiale, puis nous les avons réentraînés partiellement afin de les adapter aux spécificités des données locales. Cette étape, appelée *fine-tuning*, consiste à conserver une partie des connaissances déjà acquises par le modèle tout en réajustant ses paramètres sur un sous-ensemble représentatif des données locales.

Les résultats obtenus avec cette méthode montrent une amélioration par rapport aux modèles

initiaux non adaptés. Certaines classes de déchets, par exemple les plastiques ou les organiques, ont été mieux reconnues. Cependant, cette amélioration reste limitée. Deux facteurs principaux expliquent cette situation :

- le faible volume d'images locales disponibles, ce qui empêche le modèle d'apprendre correctement les variations propres à notre contexte ;
- le déséquilibre persistant entre les données publiques, beaucoup plus nombreuses, et les données locales, qui restent minoritaires dans l'entraînement.

Ces contraintes réduisent la capacité du modèle à représenter fidèlement les conditions réelles de tri. En pratique, cela signifie que le modèle continue d'être influencé par les caractéristiques dominantes des données publiques, parfois très éloignées des déchets rencontrés dans le contexte local. Par exemple, certaines images en ligne présentent des déchets isolés, bien centrés et éclairés, alors que dans nos données locales, les déchets apparaissent souvent dans un environnement plus complexe, avec des variations d'éclairage ou de positionnement. Le modèle, n'ayant pas vu suffisamment d'exemples de ce second cas, a tendance à mal généraliser.

Ainsi, pour obtenir une classification plus fiable dans un environnement réel, il devient indispensable d'accroître la proportion de données locales dans le jeu d'apprentissage. Ces données, même en nombre réduit, reflètent plus fidèlement les conditions pratiques et permettent au modèle de se rapprocher des situations qu'il devra traiter en déploiement. Toutefois, il n'est pas souhaitable de se passer complètement des données publiques : celles-ci jouent encore un rôle important pour diversifier les exemples et éviter que le modèle se spécialise trop sur des cas très particuliers.

L'objectif n'est donc pas de choisir entre données locales et données publiques, mais de trouver un **équilibre approprié entre les deux**. D'un côté, les données locales assurent l'adaptation du modèle au contexte réel, de l'autre, les données publiques apportent une variété qui contribue à la stabilité de l'apprentissage. C'est dans cette complémentarité que réside la clé pour améliorer durablement la performance de la classification appliquée au tri des déchets.

4.3 Détection d'anomalies

Les méthodes de classification d'images sont aujourd'hui de plus en plus utilisées pour le tri automatisé des déchets et elles offrent des résultats globalement satisfaisants. Cependant, certaines limites apparaissent lorsque l'on souhaite exploiter ces résultats dans un cadre précis, comme le recyclage. Par exemple, un modèle entraîné sur un nombre restreint de classes, comme cinq dans notre cas, ne saura pas traiter correctement des catégories absentes de l'apprentissage, telles que les métaux ou les couches. Dans ce cas, il produira forcément une fausse prédiction. Le problème devient critique si un déchet inconnu est classé comme connu, car cela peut avoir un impact très négatif sur des projets de valorisation, notamment dans la production de biogaz^[25] ou la production d'électricité.^[5]

Pour répondre à ces limites, nous proposons dans ce chapitre une amélioration du système en introduisant des mécanismes de détection d'anomalies avant la classification. Pour cela, trois approches complémentaires sont explorées :

- une méthode basée sur la confiance du modèle, où un seuil est fixé pour accepter ou rejeter une prédiction ;
- une méthode utilisant un autoencodeur, qui détecte les anomalies en fonction de l'erreur de reconstruction ;
- une approche fondée sur le discriminateur d'un GAN, capable de distinguer les données connues de celles qui sont étrangères au modèle.

Ces améliorations visent à rendre le système plus fiable et mieux adapté à des conditions réelles, où la variété des déchets dépasse largement celle prévue dans l'ensemble d'entraînement.

4.4 Base de données utilisée

Les modèles de détection d'anomalies ont été entraînés à partir de la base de données initiale utilisée pour la classification. Cependant, afin d'évaluer plus précisément leurs performances, nous avons construit une base de test spécifique composée de 4 000 images, réparties équitablement entre deux ensembles : 2 000 images connues et 2 000 images inconnues.

Pour les données connues, nous avons sélectionné aléatoirement 400 images par classe parmi les cinq catégories de l'ensemble d'origine. Les données inconnues, quant à elles, proviennent d'images représentant des objets absents de l'entraînement, tels que des batteries, des couches, des cartons ou encore d'autres types de déchets non inclus dans les classes de base.

Cette organisation permet de simuler un contexte réaliste où le modèle est confronté à des données nouvelles, afin d'évaluer sa capacité à distinguer les éléments familiers de ceux totalement inconnus.

4.5 Détection par seuil de confiance

La première amélioration proposée repose sur l'utilisation d'un **seuil de confiance** appliqué aux prédictions du modèle. Le principe est le suivant : lorsqu'un modèle de classification réalise une prédiction, il associe à chaque classe une probabilité. Si la probabilité maximale est inférieure à un seuil fixé, la prédiction est rejetée et considérée comme une anomalie ; dans le cas contraire, elle est acceptée comme valide.

Cette approche permet de limiter les erreurs critiques, notamment celles où des objets inconnus pourraient être attribués à tort à une classe connue. En pratique, le choix du seuil est un paramètre essentiel : un seuil trop bas risque de laisser passer des anomalies, tandis qu'un seuil trop élevé peut entraîner le rejet inutile de données valides. Pour déterminer ce seuil de manière efficace, nous avons testé vingt valeurs distinctes, choisies aléatoirement entre 50 % et 100 %.

L'avantage principal de cette méthode réside dans sa **simplicité de mise en œuvre**, puisqu'elle ne nécessite ni modification de l'architecture du modèle, ni réentraînement supplémentaire. En revanche, elle demeure sensible aux cas ambigus, où une forte confiance peut être attribuée à une mauvaise prédiction. C'est pourquoi cette approche est envisagée comme un premier niveau de filtrage, appelé à être complété par des mécanismes plus robustes, présentés dans les sections suivantes.

La figure 4.1 illustre la répartition des images connues et inconnues correctement identifiées en fonction du seuil de confiance, ainsi que l'évolution de l'exactitude globale du modèle.

Toutefois, il est important de souligner que ces seuils optimaux sont déterminés dans un cadre expérimental et ne reflètent pas nécessairement les conditions réelles d'utilisation, où la variabilité des données et la présence d'objets inconnus peuvent modifier la pertinence du seuil choisi.

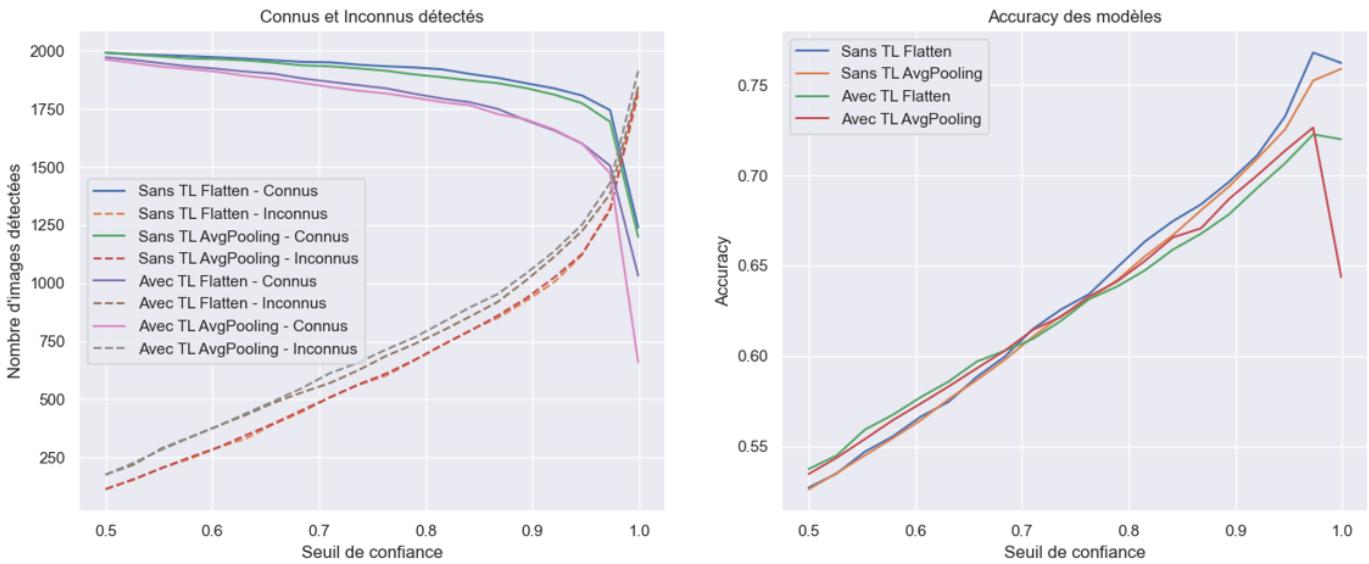


Figure 4.1 – Évolution du taux de reconnaissance des images connues et inconnues.

Table 4.1 – Résultats de l'évaluation par seuil de confiance pour les différents modèles

Modèle	Meilleur seuil	Exactitude (%)
Sans Transfert Learning (Flatten)	0.973	76.79
Sans Transfert Learning (AvgPooling)	0.999	75.89
Avec Transfert Learning (Flatten)	0.973	72.27
Avec Transfert Learning (AvgPooling)	0.973	72.64

4.6 Détection par autoencodeur

Dans la section précédente, nous avons présenté la détection par seuil de confiance, une méthode simple mais efficace permettant d'écartier les prédictions incertaines du modèle. Bien que cette approche offre une première amélioration en réduisant certaines erreurs de classification, elle reste dépendante des probabilités de sortie du réseau et ne permet pas toujours de distinguer clairement les données réellement inconnues de celles simplement mal classées.

Pour aller plus loin, nous proposons ici une méthode différente, fondée sur les *autoencodeurs*. Le principe est le suivant : au lieu de se baser sur les probabilités de prédiction, l'idée consiste à entraîner un modèle capable de reconstruire uniquement les images appartenant aux classes connues. Lorsqu'une image issue d'une distribution différente est présentée, la reconstruction devient moins fidèle, ce qui traduit son caractère anormal. Cette approche repose donc sur l'idée que ce qui ne peut pas être correctement reconstruit par le modèle est probablement inconnu.

Contrairement à la méthode par seuil de confiance, la détection par autoencodeur ne dépend pas directement du classifieur, mais de la capacité du modèle à apprendre la structure interne des données normales. Cette indépendance en fait une solution complémentaire, particulièrement adaptée pour renforcer la fiabilité globale du système de classification.

Dans la suite, nous présenterons successivement les différentes étapes de cette approche : le *pré-traitement des données*, la *construction du modèle*, la *phase d'entraînement*, puis enfin l'analyse des *résultats obtenus*.

4.6.1 Prétraitement des données

Pour cette expérience, nous avons choisi de rester sur un cadre simple et facile à reproduire. Nous avons appliqué uniquement les prétraitements nécessaires avant l'entraînement du modèle. Les données ont été divisées en deux ensembles : un jeu d'entraînement représentant 90 % du total, et un jeu de test correspondant aux 10 % restants. Par la suite, toutes les images ont été normalisées afin de stabiliser l'apprentissage et de faciliter la reconstruction par le modèle.

L'objectif ici n'est pas de concevoir un modèle directement exploitable dans des conditions réelles, mais plutôt de comprendre le comportement général de l'autoencodeur lorsqu'il est confronté à des données connues et inconnues. Pour cette raison, nous avons volontairement conservé un traitement minimal.

Cependant, plusieurs améliorations pourraient être envisagées pour renforcer la qualité du modèle. Par exemple, l'ajout d'un bruit léger aux images d'entraînement permettrait de rendre l'autoencodeur plus tolérant aux petites perturbations, tout en améliorant sa capacité de généralisation. De même, une étape de redimensionnement et de centrage plus rigoureuse pourrait aider à uniformiser les entrées, surtout si les images proviennent de sources différentes. Il serait aussi possible d'introduire des techniques d'augmentation de données (rotations, changements de luminosité ou déformations légères) afin d'enrichir la variété des exemples présentés au modèle.

Néanmoins, ces améliorations demandent des ressources de calcul supplémentaires et une phase d'expérimentation plus longue. Dans notre cas, nous avons préféré conserver une approche simple afin d'analyser le principe fondamental de la détection d'anomalies par reconstruction, sans ajouter de complexité inutile à cette première étape.

4.6.2 Modèle : *Bêta-Autoencodeur Variationnel*(β -VAE)

Le modèle utilisé dans cette expérience est un β -Autoencodeur Variationnel (β -VAE) basé sur une architecture résiduelle. Ce choix permet d'améliorer la stabilité de l'apprentissage tout en conservant une bonne capacité de reconstruction. Contrairement à la version classique du VAE, le β -VAE introduit un paramètre β qui contrôle le compromis entre la fidélité de reconstruction et la régularisation de l'espace latent. Pour cela, on utilise la fonction de coût du VAE sous la forme suivante :

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta D_{KL}(q(z|x)\|p(z))$$

où le coefficient β contrôle l'équilibre entre la fidélité de reconstruction et la régularisation de l'espace latent.

L'architecture se compose de deux parties principales : un **encodeur**, chargé de compresser les images dans un espace latent de dimension réduite, et un **décodeur**, qui reconstruit les images à partir de cette représentation. L'intégration de blocs résiduels dans ces deux composantes facilite la propagation du gradient et améliore la convergence, même lorsque le réseau devient plus profond.

Avant de décrire les deux sous-modèles, il est utile de présenter les deux types de blocs utilisés dans la conception du réseau : le *bloc résiduel standard*, utilisé pour les couches de convolution classiques, et le *bloc résiduel avec suréchantillonnage*, utilisé dans les étapes de décodage pour la reconstruction des images.

Table 4.2 – Bloc résiduel standard utilisé dans l'encodeur et le discriminateur.

Étape	Opération	Détails
1	Convolution 2D	3×3 , stride = 1, filtres = F , padding = same
2	Batch Normalization	(optionnelle selon le bloc)
3	Activation	ReLU
4	Convolution 2D	3×3 , stride = 1, filtres = F
5	Batch Normalization	(optionnelle)
6	Raccourci (skip)	Conv 1×1 si dimensions différentes
7	Somme + Activation	$y = \text{ReLU}(x + F(x))$

Table 4.3 – Bloc résiduel avec suréchantillonnage utilisé dans le décodeur.

Étape	Opération	Détails
1	UpSampling2D	Facteur 2, interpolation bilinéaire
2	Convolution 2D	3×3 , filtres = F , padding = same
3	Batch Normalization	Oui
4	Activation	ReLU
5	Convolution 2D	3×3 , filtres = F
6	Raccourci	UpSampling2D + Conv 1×1
7	Somme + Activation	$y = \text{ReLU}(x + F(x))$

Encodeur

L'encodeur a pour rôle de projeter les images d'entrée dans un espace latent de plus faible dimension. Il est construit à partir d'une succession de blocs résiduels qui permettent d'extraire progressivement les caractéristiques visuelles tout en réduisant la taille spatiale des images. À la fin, une couche de projection dense produit les deux paramètres nécessaires à la distribution latente : la moyenne et la variance logarithmique.

Table 4.4 – Architecture de l'encodeur du modèle β -VAE.

Bloc	Opération	Taille de sortie
Entrée	Image RGB	(240, 240, 3)
Bloc résiduel 1	$F = 8$, stride=1	(240, 240, 8)
Bloc résiduel 2	$F = 16$, stride=2	(120, 120, 16)
Bloc résiduel 3	$F = 32$, stride=2	(60, 60, 32)
Bloc résiduel 4	$F = 64$, stride=2	(30, 30, 64)
Pooling global	Moyenne spatiale	(64,)
Dense	Paramètres latents ($\mu, \log \sigma^2$)	(128,)

Décodeur

Le décodeur effectue l'opération inverse de l'encodeur. Il prend en entrée un vecteur latent et reconstruit l'image d'origine à l'aide de blocs résiduels avec suréchantillonnage. Cette conception permet de générer des reconstructions plus stables et mieux structurées, en limitant les artefacts souvent observés dans les décodeurs simples à convolution transposée.

Table 4.5 – Architecture du décodeur du modèle β -VAE.

Bloc	Opération	Taille de sortie
Entrée	Vecteur latent	(128,)
Dense + Reshape	Transformation en carte de caractéristiques	(30, 30, 64)
Bloc résiduel	$F = 64$	(30, 30, 64)
Bloc résiduel up	$F = 32$	(60, 60, 32)
Bloc résiduel	$F = 32$	(60, 60, 32)
Bloc résiduel up	$F = 16$	(120, 120, 16)
Bloc résiduel	$F = 16$	(120, 120, 16)
Bloc résiduel up	$F = 8$	(240, 240, 8)
Bloc résiduel	$F = 8$	(240, 240, 8)
Sortie	Conv2D (3×3 , sigmoid)	(240, 240, 3)

4.6.3 Entraînement du modèle

Pour cette expérience, nous avions initialement prévu d'entraîner le modèle sur 400 époques afin de lui permettre d'apprendre plus en profondeur les caractéristiques des données normales. L'entraînement s'est effectué en plusieurs tranches successives, chaque session reprenant à partir du modèle sauvegardé précédemment. Cette méthode permet de poursuivre facilement l'apprentissage sans avoir à recommencer depuis le début.

Pour des raisons de stockage et afin de ne pas ralentir le processus d'apprentissage, nous n'avons pas conservé l'historique complet de toutes les époques. En effet, enregistrer en continu l'ensemble des métriques d'entraînement aurait considérablement augmenté le temps de calcul et la taille des fichiers générés. Ainsi, seules les 95 dernières époques ont été enregistrées, correspondant au dernier entraînement prévu pour 100 époques.

Le modèle a été entraîné avec un coefficient $\beta = 0.001$ et une taille de lot de 128. Les données ont été séparées en deux sous-ensembles : 90% pour l'entraînement et 10% pour le test. Aucune augmentation ni ajout de bruit n'a été appliquée aux données, le but étant de conserver une approche simple et maîtrisée, puisque ce modèle n'était pas destiné à une utilisation directe dans des conditions réelles.

4.6.4 Résultats

L'analyse des résultats du modèle β -VAE met en évidence ses limites dans la reconstruction et la généralisation des données. Comme illustré à la Figure 4.2, la perte d'entraînement diminue régulièrement, tandis que celle du test stagne, voire augmente légèrement après plusieurs époques. Ce comportement traduit un phénomène de **surapprentissage (overfitting)** : le modèle s'adapte trop aux exemples d'entraînement sans parvenir à généraliser sur de nouvelles données.

Cette tendance suggère que la représentation latente apprise reste peu efficace et ne capture pas correctement la structure globale des images. Malgré une architecture résiduelle et une régularisation par le coefficient β , le modèle peine à extraire des caractéristiques suffisamment discriminantes pour reconstruire efficacement les données.

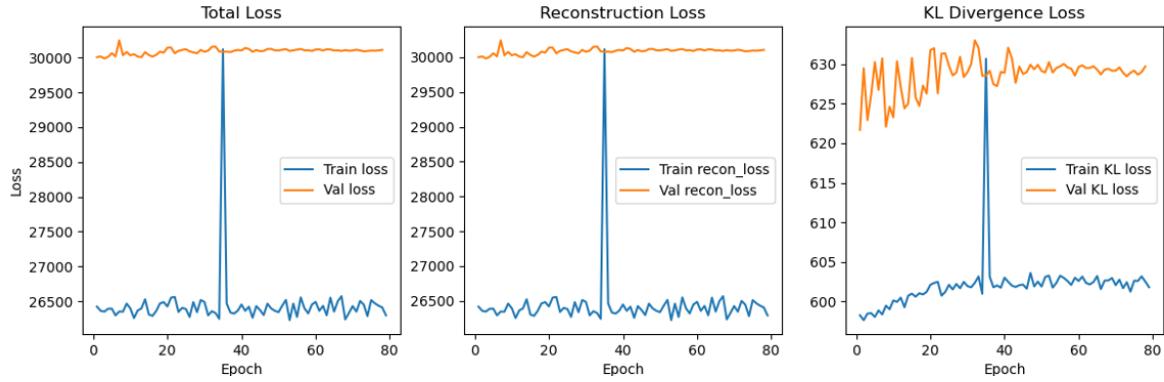


Figure 4.2 – Courbe d'apprentissage du modèle β -VAE sur les ensembles d'entraînement et de test.

Les Figures 4.3 et 4.4 illustrent des exemples d'images reconstruites. On observe des reconstructions globalement imprécises, avec une perte notable de détails et des formes parfois déformées. Ces résultats confirment la faible capacité du modèle à restituer fidèlement les données, en particulier sur le jeu de test.

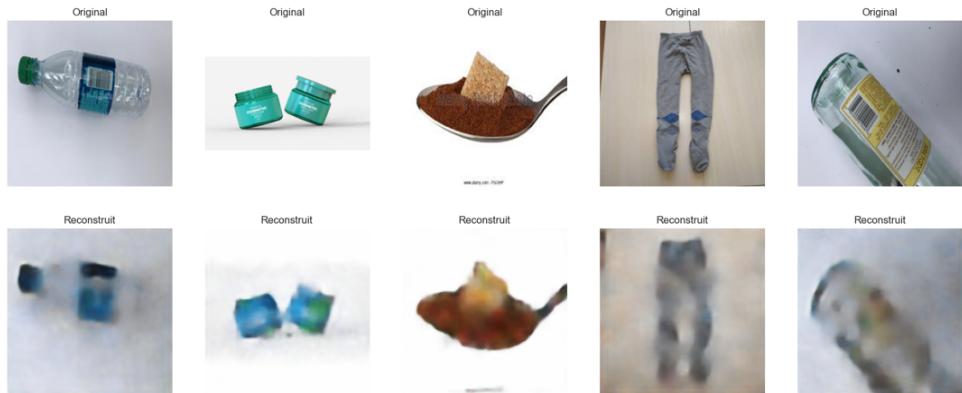


Figure 4.3 – Exemples d'images originales et reconstruites sur les données d'entraînement.



Figure 4.4 – Exemples d'images originales et reconstruites sur les données de test.

En résumé, le β -VAE présente des difficultés à modéliser efficacement la distribution des données normales. Des améliorations comme une régularisation plus forte, une meilleure gestion du bruit ou l'augmentation des données pourraient aider à renforcer la robustesse du modèle et la qualité des reconstructions.

4.6.5 Application

Après l'entraînement du modèle β -VAE, nous avons utilisé l'erreur de reconstruction pour détecter les images anormales, c'est-à-dire celles qui ne ressemblent pas aux données vues pendant l'apprentissage. Plus cette erreur est élevée, plus l'image est considérée comme inconnue.

Pour cette expérience, nous avons utilisé les mêmes données que pour la détection par seuil de confiance. Comme le seuil de reconstruction joue un rôle essentiel dans la séparation entre les données normales et anormales, il ne peut pas être fixé au hasard. Nous avons donc testé trente valeurs comprises entre 0.001 et 0.02 afin de déterminer le seuil offrant le meilleur équilibre entre les principales métriques de performance.

La Figure 4.5 illustre l'évolution du nombre de détections ainsi que des métriques (exactitude, précision, rappel et F1-score) en fonction du seuil. On observe qu'en augmentant le seuil, le nombre d'images considérées comme normales augmente, tandis que celui des anomalies détectées diminue. En parallèle, la précision tend à s'améliorer alors que le rappel baisse progressivement. Le F1-score, qui représente un compromis entre ces deux mesures, atteint sa valeur maximale autour de 0.01607. Ce seuil a donc été retenu comme **seuil optimal**.

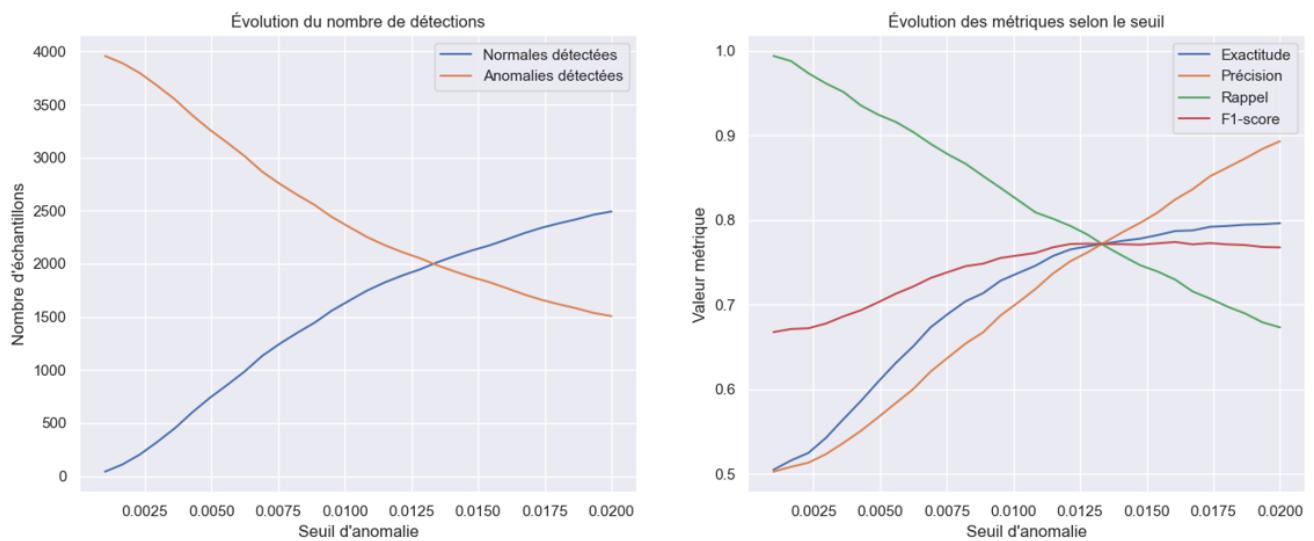


Figure 4.5 – Évolution du nombre de détections (à gauche) et des métriques de performance (à droite) selon le seuil d'anomalie.

Avec ce seuil, le modèle obtient les performances présentées dans le tableau 4.6. Ces résultats montrent que l'autoencodeur parvient globalement à distinguer les données normales des données anormales, bien que certaines erreurs persistent, notamment entre des classes visuellement proches.

Table 4.6 – Résultats de la détection d'anomalies par le modèle β -VAE (seuil optimal = 0.01607).

Classe	Précision	Rappel	F1-score
Normal	0.757	0.844	0.798
Anormal	0.824	0.730	0.774

La Figure 4.6 montre la matrice de confusion correspondant à cette classification. Le modèle identifie une bonne partie des anomalies, mais certaines images normales sont encore mal classées. Ce résultat reflète les limites de l'autoencodeur dans la reconstruction précise des données et dans

la séparation nette entre classes connues et inconnues.

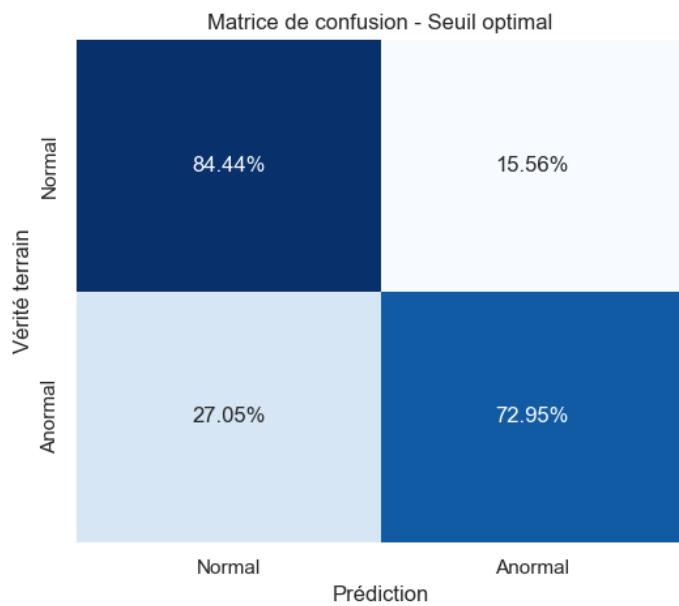


Figure 4.6 – Matrice de confusion pour la détection d'anomalies par le modèle β -VAE.

En résumé, cette approche montre que le modèle β -VAE peut être utilisé pour repérer les images inconnues, mais ses performances restent limitées. Les résultats obtenus constituent une base encourageante pour des améliorations futures.

4.7 Détection par réseaux antagonistes génératifs(GANs)

Dans la section précédente, nous avons présenté notre approche de détection basée sur un autoencodeur variationnel. Malgré sa faible capacité de généralisation, le modèle a tout de même fourni des résultats intéressants pour la détection d'anomalies. Comme précisé au début de ce chapitre, notre objectif n'est pas de produire un modèle directement exploitable dans des conditions réelles, mais d'examiner plusieurs approches afin d'identifier celles qui pourraient, à terme, mieux s'adapter à ce type de problème.

Nous allons donc expérimenter une autre méthode : les *réseaux antagonistes génératifs (GANs)*. Cette technique repose sur l'apprentissage simultané de deux réseaux : un *générateur*, qui tente de créer des images proches de celles issues du jeu d'entraînement, et un *discriminateur*, dont le rôle est de distinguer les images réelles de celles produites artificiellement. Cette confrontation progressive entre les deux réseaux permet d'obtenir un modèle qui apprend à représenter fidèlement les données d'origine.

En résumé, ici l'objectif n'est pas de générer des images réalistes, mais d'exploiter ce que le discriminateur a appris pour identifier les données qui n'appartiennent pas au même domaine que celles vues pendant l'entraînement.

Dans la suite, nous présenterons successivement les différentes étapes de cette approche : le **prétraitement des données**, la **construction du modèle**, la **phase d'entraînement**, puis l'analyse des **résultats obtenus**, avant de conclure par une **application pratique** de cette méthode.

4.7.1 Prétraitement des données

Tout comme dans le modèle d'autoencodeur, nous avons choisi de rester dans un cadre simple et facile à reproduire. Aucune opération complexe de prétraitement n'a été appliquée, uniquement les étapes essentielles nécessaires à la préparation des données avant l'entraînement du réseau antagoniste génératif (GAN).

Contrairement aux modèles de classification ou d'autoencodeur, un GAN ne nécessite pas de jeu de test distinct pour évaluer ses performances. Cette particularité permet, en théorie, d'utiliser l'ensemble des données disponibles pour l'entraînement. Toutefois, un tel choix demanderait une puissance de calcul importante, ce qui dépasse les ressources matérielles dont nous disposons.

Pour rendre l'entraînement plus abordable, nous avons donc redimensionné toutes les images de leur taille initiale de $240 \times 240 \times 3$ à $64 \times 64 \times 3$. Ce compromis permet d'accélérer les calculs tout en conservant les informations visuelles nécessaires à la génération d'images réalistes. Enfin, une normalisation a été appliquée afin de ramener les valeurs des pixels dans l'intervalle $[-1, 1]$, ce qui favorise la stabilité de l'apprentissage et améliore la convergence du modèle.

4.7.2 Construction du modèle

Ici, nous restons dans un cadre bien simple. L'idée n'est pas de concevoir une architecture très complexe, mais plutôt de mettre en place un modèle clair, efficace et facile à entraîner. Le but est surtout de comprendre le comportement du réseau et d'observer sa capacité à apprendre la distribution des données. Pour cela, nous avons adopté une architecture classique de GAN avec des choix de couches et de paramètres adaptés à nos contraintes matérielles et à la taille des images. Ce cadre simple nous permet de nous concentrer sur l'essentiel : la logique d'apprentissage et la qualité des résultats obtenus après l'entraînement. Les *Tableaux 4.7 et 4.8* présentent respectivement leurs architectures détaillées.

Table 4.7 – Architecture du générateur du modèle GAN.

Couche	Type	Paramètres principaux	Shape de sortie
Entrée	Dense	$8 \times 8 \times 128$, <i>use_bias=False</i>	(None, 8192)
	BatchNorm	-	(None, 8192)
	LeakyReLU	$\alpha = 0.2$	(None, 8192)
	Reshape	-	(None, 8, 8, 128)
Bloc 1	Conv2DTranspose	128 filtres, kernel 5×5 , stride=2	(None, 16, 16, 128)
	BatchNorm	-	(None, 16, 16, 128)
	LeakyReLU	$\alpha = 0.2$	(None, 16, 16, 128)
Bloc 2	Conv2DTranspose	64 filtres, kernel 5×5 , stride=2,	(None, 32, 32, 64)
	BatchNorm	-	(None, 32, 32, 64)
	LeakyReLU	$\alpha = 0.2$	(None, 32, 32, 64)
Bloc 3	Conv2DTranspose	32 filtres, kernel 5×5 , stride=2	(None, 64, 64, 32)
	BatchNorm	-	(None, 64, 64, 32)
	LeakyReLU	$\alpha = 0.2$	(None, 64, 64, 32)
Sortie	Conv2DTranspose	3 filtres, kernel 5×5 , stride=1, <i>activation=tanh</i>	(None, 64, 64, 3)

Table 4.8 – Architecture du discriminateur du modèle GAN.

Couche	Type	Paramètres principaux	Shape de sortie
Entrée	Conv2D	64 filtres, kernel 5×5 , stride=2	(None, 32, 32, 64)
	LeakyReLU	$\alpha = 0.2$	(None, 32, 32, 64)
	Dropout	0.3	(None, 32, 32, 64)
Bloc 1	Conv2D	128 filtres, kernel 5×5 , stride=2	(None, 16, 16, 128)
	LeakyReLU	$\alpha = 0.2$	(None, 16, 16, 128)
	Dropout	0.3	(None, 16, 16, 128)
Bloc 2	Conv2D	256 filtres, kernel 5×5 , stride=2	(None, 8, 8, 256)
	LeakyReLU	$\alpha = 0.2$	(None, 8, 8, 256)
	Dropout	0.3	(None, 8, 8, 256)
Flatten	GlobalAveragePooling2D	-	(None, 256)
Dense 1	Dense	64 neurones, <i>ReLU</i> , régularisation L2(0.02)	(None, 64)
Sortie	Dense	1 neurone, <i>activation = sigmoid</i>	(None, 1)

4.7.3 Stratégie d'entraînement

Le modèle GAN a été entraîné suivant une approche simple, sans chercher à atteindre des performances optimales. L'objectif principal était de comprendre le comportement du modèle au cours de l'apprentissage et d'observer sa capacité à générer des images réalistes.

Le générateur et le discriminateur ont chacun leur propre optimiseur *Adam*, avec des taux d'apprentissage initiaux différents : 5×10^{-5} pour le générateur et 2×10^{-5} pour le discriminateur. Les coefficients $\beta_1 = 0.5$ et $\beta_2 = 0.999$ sont conservés pour les deux réseaux, selon les pratiques courantes dans ce type de modèle.

L'apprentissage s'est déroulé sur un total de 200 époques. Trois mécanismes standards ont été utilisés pour mieux suivre la progression du modèle :

- **ModelCheckpoint** : sauvegarde automatique du modèle lorsque la perte du générateur (*g_loss*) diminue.
- **EarlyStopping** : arrêt de l'entraînement après 50 époques sans amélioration, avec restauration des meilleurs poids.
- **ReduceLROnPlateau** : réduction du taux d'apprentissage d'un facteur de 0.2 après 15 époques sans progrès.

Ces réglages permettent simplement de contrôler l'évolution du modèle sans intervention manuelle pendant l'entraînement.

Table 4.9 – Résumé des paramètres et stratégies d'entraînement du GAN

Paramètre	Valeur / Description
Nombre d'époques	200
Optimiseur (Générateur)	Adam($\eta=5 \times 10^{-5}$, $\beta_1=0.5$, $\beta_2=0.999$)
Optimiseur (Discriminateur)	Adam($\eta=2 \times 10^{-5}$, $\beta_1=0.5$, $\beta_2=0.999$)
Fonction de perte	Binary Cross-Entropy
Critère de suivi	<i>g_loss</i> (perte du générateur)
EarlyStopping	Patience = 50, mode = min
ReduceLROnPlateau	Facteur = 0.2, patience = 15, min_lr = 10^{-6}
ModelCheckpoint	Sauvegarde automatique du meilleur modèle

4.7.4 Résultats

L'analyse des résultats d'entraînement montre que le modèle GAN n'a pas donné les performances attendues. La Figure 4.7 présente l'évolution des pertes du générateur et du discriminateur au fil des époques, ainsi que le rapport entre les deux. On remarque que les pertes restent globalement instables, sans réelle convergence. Le générateur n'arrive pas à produire des échantillons convaincants, tandis que le discriminateur semble dominer la plupart du temps, ce qui empêche l'équilibre d'apprentissage nécessaire à une bonne génération d'images.

L'entraînement s'est arrêté au bout de 58 époques, signe que le modèle avait cessé de s'améliorer bien avant dès la huitième époque, les courbes montrent un ralentissement net de la progression, indiquant que le processus d'apprentissage était déjà bloqué.

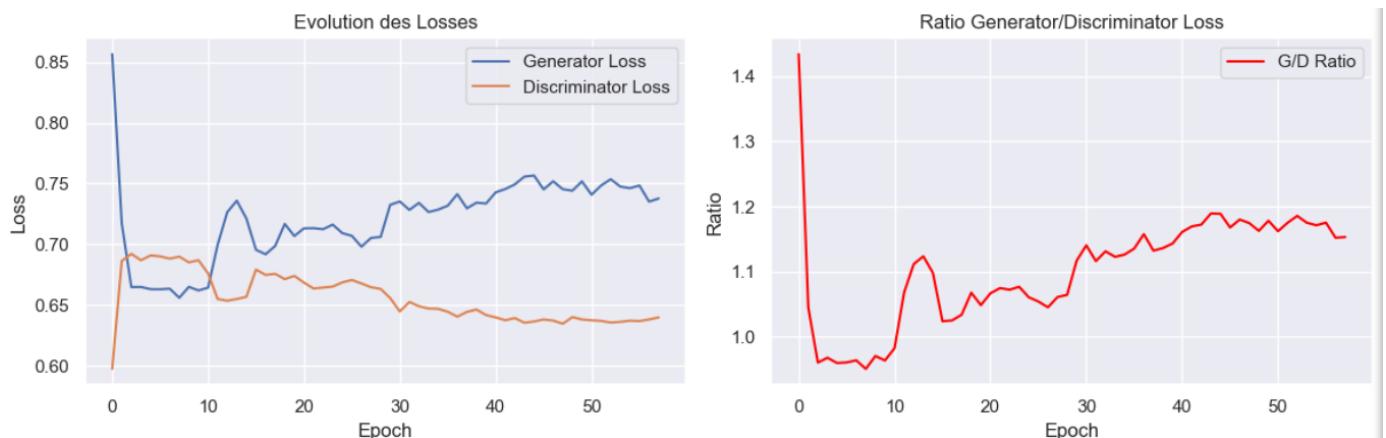


Figure 4.7 – Évolution des pertes du générateur et du discriminateur, ainsi que du ratio G/D au cours de l'entraînement.

La Figure 4.8 illustre quelques exemples d'images générées par le modèle après l'entraînement. Comme on peut le constater, les résultats sont loin d'être exploitables : les formes sont floues, les couleurs peu cohérentes et aucune structure identifiable ne se dégage. Ces images montrent clairement que le modèle n'a pas appris à capturer la distribution des données réelles.

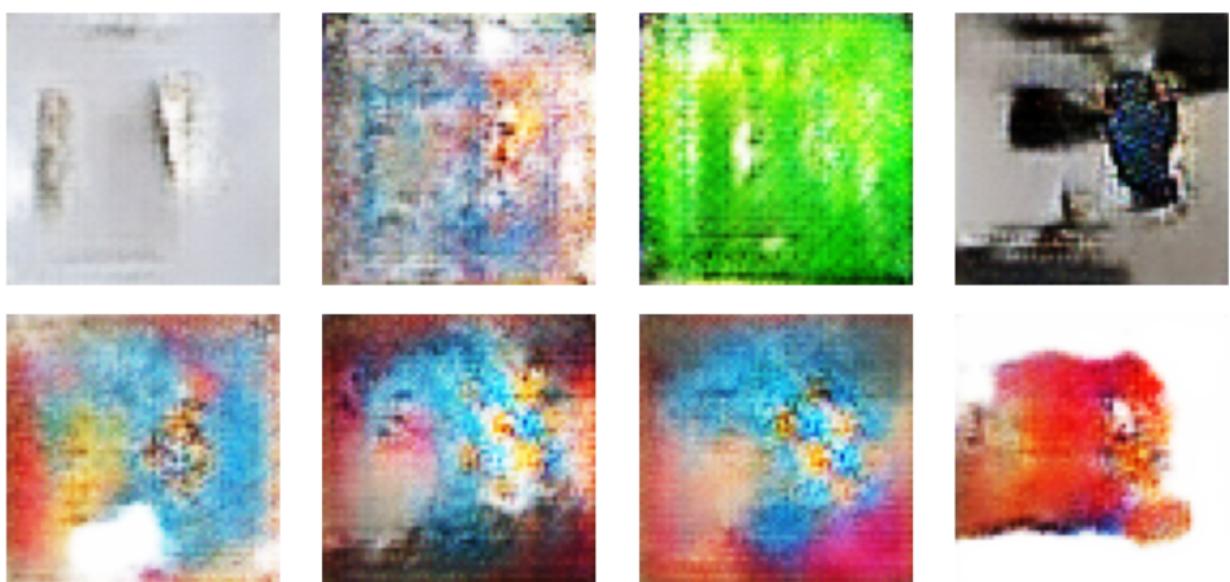


Figure 4.8 – Exemples d'images générées après entraînement du modèle GAN.

Compte tenu de ces résultats, il n'a pas été possible de poursuivre les tests sur la détection d'anomalies, le modèle n'étant pas suffisamment entraîné pour générer ou reconnaître des échantillons valides. Cependant, cette étape reste importante, car elle permet de comprendre les défis associés à l'entraînement des GANs, notamment la difficulté à stabiliser le processus d'apprentissage. Avec un modèle plus performant et des ressources de calcul plus adaptées, cette approche demeure l'une des pistes les plus prometteuses pour la détection d'anomalies visuelles.

4.8 Conclusion

Au terme de ce chapitre, plusieurs approches ont été explorées afin d'améliorer les performances de classification et de détection d'anomalies. Dans un premier temps, nous avons évoqué la nécessité de collecter davantage de données locales et d'appliquer le transfert learning à partir des modèles initiaux. Cette étape permettrait d'adapter le modèle aux spécificités des données réelles rencontrées sur le terrain.

Ensuite, nous avons étudié trois approches de détection d'anomalies. La première, fondée sur le seuil de confiance, bien que simple à mettre en place, reste peu efficace. La seconde, basée sur un autoencodeur variationnel, a montré des résultats plus intéressants, confirmant la capacité de ce type de modèle à distinguer les données connues des inconnues. Enfin, nous avons expérimenté un modèle de type GAN. Les résultats obtenus n'étaient pas satisfaisants, le modèle ayant cessé de s'améliorer après 58 époques, mais cette expérience demeure utile pour comprendre le potentiel de cette approche.

En effet, plusieurs travaux récents ont montré que les GANs bien entraînés peuvent offrir d'excellentes performances pour la détection d'anomalies. C'est notamment le cas des modèles^[74] et des études de^[75] qui soulignent la pertinence de cette approche dans différents contextes.

Ce chapitre pose donc un cadre clair pour la suite : on sait ce qui ne fonctionne pas bien, ce qui fonctionne mieux, et où concentrer nos efforts pour améliorer les résultats dans des études à venir.

Conclusion générale

Nous arrivons au terme de ce travail, un travail qui nous a permis d'explorer deux domaines essentiels à la fois pour notre société et pour l'avenir de la technologie : *la gestion des déchets*, qui représente une préoccupation publique majeure, et le *Deep Learning*, qui constitue aujourd'hui le cœur de nombreuses avancées technologiques, notamment dans le domaine de la vision par ordinateur.

L'objectif principal de ce mémoire était d'étudier comment les techniques d'apprentissage profond peuvent être mises au service de la gestion des déchets, afin de proposer une solution plus intelligente, plus rapide et moins dépendante des interventions humaines. Pour cela, nous avons d'abord dressé un état des lieux détaillé du contexte mondial et local, en soulignant les défis auxquels la ville de Moroni fait face : croissance démographique, urbanisation non planifiée et manque d'infrastructures adaptées. Ces constats ont permis de mieux comprendre l'urgence d'agir et la nécessité d'introduire des approches technologiques modernes dans ce domaine.

Avant de présenter notre contribution, nous avons pris le temps d'introduire les fondements théoriques du Deep Learning. Cette partie n'avait pas pour but de faire un cours exhaustif, mais plutôt de rappeler les notions clés nécessaires pour comprendre les modèles que nous avons mis en œuvre, notamment les réseaux de neurones, les perceptrons multicouches et les architectures de classification d'images.

La suite du travail s'est concentrée sur la conception et l'évaluation de nos modèles. Nous avons utilisé différentes bases de données, combinant des images issues de sources ouvertes et des images locales collectées à Moroni. Cette double approche nous a permis d'évaluer la capacité de généralisation des modèles dans un contexte réel. Les résultats ont montré que les modèles entraînés uniquement sur des données publiques manquent souvent de performance sur les données locales, en raison de la différence de distribution et de conditions d'acquisition.

Pour pallier ce problème, nous avons exploré plusieurs stratégies : d'abord le *transfert learning*, qui consiste à adapter un modèle préentraîné à notre propre jeu de données, puis des méthodes de détection d'anomalies, en commençant par la détection par seuil de confiance une approche simple à mettre en place avant d'expérimenter des modèles plus complexes comme les autoencodeurs variationnels et les réseaux antagonistes génératifs (GANs).

Même si tous les modèles testés n'ont pas donné des résultats directement exploitables, chaque expérience a apporté une meilleure compréhension du comportement des modèles dans un contexte réel et difficile. En particulier, les résultats obtenus avec l'autoencodeur variationnel ont été assez encourageants et ont ouvert la voie à de nouvelles pistes pour les recherches futures.

Ce travail a donc permis de poser les bases d'un système de tri automatisé des déchets adapté au contexte comorien. Bien qu'il reste encore beaucoup à faire, il démontre que l'intelligence artificielle peut réellement contribuer à améliorer la qualité de vie et à répondre à des besoins concrets, même avec des ressources limitées.

En somme, ce projet marque le début d'une aventure scientifique et humaine. Il m'a permis de comprendre que derrière chaque algorithme se cache la possibilité d'apporter une solution utile, et que la technologie n'a de sens que lorsqu'elle sert des causes réelles et humaines. Les perspectives ouvertes par ce travail sont nombreuses, et elles encouragent à poursuivre dans cette direction, avec plus de données, plus d'expérimentation et surtout plus de détermination.

Bibliographie

- [1] Silpa Kaza, Lisa Yao, Perinaz Bhada-Tata, and Frank Van Woerden. *What a Waste 2.0 : A Global Snapshot of Solid Waste Management to 2050*. World Bank, 2018.
- [2] United Nations Environment Programme and International Solid Waste Association. *Global Waste Management Outlook*. UNEP, 2015.
- [3] United Nations Environment Programme. From pollution to solution : A global assessment of marine litter and plastic pollution, 2021.
- [4] United Nations Environment Programme. *Africa Waste Management Outlook*. UNEP, 2018.
- [5] Kamil Abdallah Amarillis. *Valorisation énergétique des déchets solides ménagers et assimilés : application à la gazéification pour la production d'électricité aux Comores*. Thèse de doctorat en physique énergétique, Université de La Réunion et Université d'Antananarivo, 2023. Soutenue le 15 décembre 2023.
- [6] Deepak Chandra Jhariya and Tiwari Kumar. Impact of open dumping of municipal solid waste on soil and groundwater quality : a case study. *Environmental Nanotechnology, Monitoring & Management*, 16 :100467, 2021.
- [7] E. E. Awokunmi, S. S. Asaolu, and K. O. Ipinnmoroti. Effect of leachate on groundwater quality. *African Journal of Environmental Science and Technology*, 4(8) :445–450, 2010.
- [8] R. Khan and M. M. Haque. A critical review on solid waste management in dhaka city and its impact on environment. In *7th International Conference on Environmental Engineering*, 2024.
- [9] D. O. Okoye. Comparative assessment of atmospheric pollutants across geopolitical zones in southern nigeria using sentinel-5p (2019 and 2024). *Research Square*, 2025.
- [10] B. Mayilsankar and S. T. Ramesh. Quantification of thermogenic potential and consequent temperature rise induced by municipal solid waste dumpsites. *Environmental Monitoring and Assessment*, 2025.
- [11] A. Hossain and M. Hasan. Navigating the hidden crisis : Microplastics and heavy metals in soil—transport, impact, and remediation. *Soil and Sediment Contamination*, 2025.
- [12] M. T. Islam and U. B. Antu. Microplastic toxicity in fish : A potential review on sources, impacts, and solutions. *Aquatic Toxicology*, 2025.
- [13] W. Ali and F. Khan. Marine plastic pollution : Impacts on aquatic fauna and ecosystem health. *ResearchGate Preprint*, 2024.
- [14] Chisom Nzediegwu and Oladele Akinbile. Solid waste management in africa : A review. *Journal of Environmental Management*, 261 :110216, 2020.
- [15] Ngoc Kimani. Environmental pollution and impacts on public health : Implications of the dandora municipal dumping site in nairobi, kenya. *United Nations Environment Programme (UNEP)*, 2007.

- [16] R. E. Marshall and K. Farahbakhsh. Systems approaches to integrated solid waste management in developing countries. *Waste Management*, 33(4) :988–1003, 2020.
- [17] S. K. Ghosh and A. Yadav. Sustainable solid waste management : An integrated approach. *Environmental Science and Pollution Research*, 27(36) :44655–44667, 2020.
- [18] European Commission. Single-use plastics : Directive (eu) 2019/904, 2021.
- [19] J. Njeru. The urban political ecology of plastic bag waste problem in nairobi, kenya. *Geoforum*, 37(6) :1046–1058, 2006.
- [20] F. Fuso Nerini, A. Slob, and D. Yumashev. Sustainable consumption and production : A review of progress. *Sustainability*, 13(19) :10892, 2021.
- [21] D. Q. Zhang, S. R. Williams, and G. He. Solid waste management in developing countries : Status, perspectives and capacity building. *Waste Management*, 30(6) :1436–1444, 2010.
- [22] OECD. Municipal waste recycling rates, 2022.
- [23] Martin Medina. *The World's Scavengers : Salvaging for Sustainable Consumption and Production*. AltaMira Press, 2008.
- [24] K. Ragaert, L. Delva, and K. Van Geem. Mechanical and chemical recycling of solid plastic waste. *Waste Management*, 69 :24–58, 2017.
- [25] Malik El'Houyoun Ahamadi, Ali Mmadi, Ahamada Mohamed, Soulé Hamidou, Ali Badria Soulé, and Soulaïmana Abdou Samioun. Physico-chemical characterization of comorian household waste for energy recovery. *International Journal of Energy Engineering*, 14(1) :7–13, 2024.
- [26] T. C. Nzeadibe, R. N. C. Anyadike, and C. A. Okeke. Waste-to-energy potential in nigeria : Prospects and challenges. *Renewable and Sustainable Energy Reviews*, 135 :110263, 2021.
- [27] C. Zunguze, I. Nhapi, and S. Famba. Barriers to sustainable solid waste management in developing countries : A case study of mozambique. *Waste Management & Research*, 40(7) :845–853, 2022.
- [28] Christian Zurbrugg, M. Gfrerer, H. Ashadi, W. Brenner, R. Kuehr, and A. A. Magalang. Urban waste management in low income countries : Challenges and opportunities. *Habitat International*, 37 :159–166, 2012.
- [29] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4) :115–133, 1943.
- [30] Frank Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386, 1958.
- [31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088) :533–536, 1986.
- [32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.
- [33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.
- [34] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786) :504–507, 2006.
- [35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105, 2012.
- [36] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. pages 15–32. Boston – Delft, 2019.

- [37] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. 27, 2014.
- [38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2015.
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.
- [40] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [41] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI technical report*, 2018.
- [42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.
- [43] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33 :1877–1901, 2020.
- [44] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words : Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*, 2021.
- [45] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33 :6840–6851, 2020.
- [46] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [48] Takeshi Kojima, Shixiang Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.
- [49] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *International Conference on Learning Representations (ICLR)*, 2023.
- [50] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts : Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.
- [51] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Yuxin Clegg, Achal Reva, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.
- [52] Shunyu Yao, Dian Yang, Yu Cui, Karthik Narasimhan, Tom Griffiths, and Percy Liang. React : Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.
- [53] Scott Reed, Nando de Freitas, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

- [54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Convolutional networks. In *Deep Learning*, chapter 9, pages 330–371. MIT Press, 2016. Consulté pour l'étude des réseaux de neurones convolutionnels.
- [55] Tammineni Sivakumar, Uttaravalli Priya, Sheik Fathimabebi, and Shaik Bashir Ahamad. A cnn-based image classification model for efficient waste management. *International Journal of Research Publication and Reviews*, 6(4) :15676–15687, 2025. ISSN 2582-7421.
- [56] P. Akhil Rajeev, Vivek Dharewa, D. Lakshmi, G. Vishnuvarthan, Jayant Giri, T. Sathish, and Mubarak Alrashoud. Advancing e-waste classification with customizable yolo based deep learning models. *Scientific Reports*, 15 :18151, 2025.
- [57] Zuohua Li, Quanxue Deng, Peicheng Liu, Jing Bai, Yunxuan Gong, Qitao Yang, and Jiafei Ning. An intelligent identification and classification system of decoration waste based on deep learning model. *Waste Management*, 174 :462–475, 2024.
- [58] Poojesh Shetty, Pranay Shetty, and Sanjeed Shriyan. Classification of waste for efficient disposal & recycling using deep learning. *International Research Journal of Engineering and Technology (IRJET)*, 07(07) :4615–4621, 2020. e-ISSN : 2395-0056, p-ISSN : 2395-0072.
- [59] Suman Kunwar. Managing household waste through transfer learning. *arXiv preprint*, arXiv :2402.09437, 2024.
- [60] Meena Malik, Sachin Sharma, Mueen Uddin, Chin-Ling Chen, Chih-Ming Wu, Punit Soni, and Shikha Chaudhary. Waste classification for sustainable development using image recognition with deep learning neural network models. *Sustainability*, 14(12) :7222, 2022.
- [61] K. Srivatsan, Surender Dhiman, and Anuj Jain. Waste classification using transfer learning with convolutional neural networks. In *IOP Conference Series : Earth and Environmental Science*, volume 775, page 012010. IOP Publishing, 2021.
- [62] Ali Arishi. Real-time household waste detection and classification for sustainable recycling : A deep learning approach. *Sustainability*, 17(5) :1902, 2025.
- [63] Andhy Panca Saputra and Kusrini. Waste object detection and classification using deep learning algorithm : Yolov4 and yolov4-tiny. *Turkish Journal of Computer and Mathematics Education*, 12(6) :5583–5595, 2021.
- [64] Ninghui Li and Yuan Chen. Municipal solid waste classification and real-time detection using deep learning methods. *Urban Climate*, 49 :101462, 2023.
- [65] Nonso Nnamoko, Joseph Barrowclough, and Jack Procter. Solid waste image classification using deep convolutional neural network. *Infrastructures*, 7(4) :47, 2022.
- [66] Ayush Pandey, Ayushma Pandey, Kenish Maharjan, Kiran Shrestha, and Jalauddin Mansur. Enhancing waste management : Automated classification of biodegradable and non-biodegradable waste using cnn. In *Proceedings of the International Conference on Technologies for Computer, Electrical, Electronics & Communication (ICT-CEEL 2023)*. Kathford International College of Engineering and Management, Tribhuvan University, 2023.
- [67] Ali Usman Gondal, Muhammad Imran Sadiq, Tariq Ali, Muhammad Irfan, Ahmad Shaf, Muhammad Aamir, Muhammad Shoaiib, Adam Glowacz, Ryszard Tadeusiewicz, and Eliasz Kantoch. Real-time multipurpose smart waste classification model for efficient recycling in smart cities using multilayer convolutional neural network and perceptron. *Sensors*, 21(14) :4916, 2021.
- [68] Yuhui Chen, Yan He, Jinghan Lin, and Siyu Sun. Garbage image recognition and classification based on cnn. In *Proceedings of the 3rd International Conference on Signal Processing and Machine Learning*, 2023.
- [69] Gregory Hendricks et al. Trashnet dataset. GitHub, 2016. Available at : <https://github.com/garythung/trashnet>.

- [70] Kaggle Contributors. Waste classification data. Kaggle, 2020. Available at : <https://www.kaggle.com/datasets/asdasdasdas/garbage-classification>.
- [71] Kaggle Contributors. Garbage classification dataset. Kaggle, 2020. Available at : <https://www.kaggle.com/datasets/mostafaabla/garbage-classification>.
- [72] Various Contributors. Recyclable and household waste classification. <https://www.kaggle.com/datasets/mostafaabla/recyclable-and-household-waste-classification>, 2022. Consulté en 2024.
- [73] Julia Gusak, Daria Cherniuk, Alena Shilova, Alexander Katrutsa, Daniel Bershatsky, Xunyi Zhao, Lionel Eyraud-Dubois, Oleg Shlyazhko, Denis Dimitrov, Ivan Oseledets, and Olivier Beaumont. Survey on large scale neural network training. *arXiv preprint*, arXiv :2202.10435, 2022.
- [74] Federico Di Mattia, Paolo Galeone, Michele De Simoni, and Emanuele Ghelfi. A survey on gans for anomaly detection. *arXiv preprint*, arXiv :1906.11632v2, 2021.
- [75] Rushikesh Zawar, Krupa Bhayani, Neelanjan Bhowmik, Kamlesh Tiwari, and Dhiraj. Detecting anomalies using generative adversarial networks on images. *arXiv preprint*, arXiv :2211.13808v1, 2022.