UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR
SEYDINA SY

COMPARAISON D'ALGORITHMES SIAMOIS POUR L'IDENTIFICATION DE VISAGES MASQUÉS

JUILLET 2025

RÉSUMÉ

La reconnaissance faciale est une technologie clé dans les domaines de la sécurité, de la surveillance, et de l'interaction homme-machine. Avec l'avancement des techniques d'apprentissage profond, les modèles de reconnaissance faciale ont considérablement évolué, permettant des performances impressionnantes en termes de précision et de robustesse.

Dans ce mémoire, nous avons développé deux modèles de reconnaissance faciale avec les réseaux siamois qui sont basés sur les Visions Transformers et les réseaux de neurones convolutifs (respectivement Réseau siamois VGG-19 et Siamois Vision Transformer). Ces modèles sont conçus pour comparer deux images faciales en entrée et renvoient un score de similarité pour la reconnaissance. De plus, en raison du manque d'ensemble de données étendu et diversifié pouvant être utilisé pour évaluer la reconnaissance faciale de personne masquée, nous avons construit notre propre ensemble de données. Il s'agit des captures d'images issues de vidéos contenant de l'information sur le visage de différentes célébrités. Pour chaque célébrité, nous avons collecté des images avec visages masqués et non masqués pour permettre à nos modèles de faire l'identification de la personne même quand elle porte un masque facial. Ensuite, les données sont préparées et générées par pairs. Le modèle Réseau siamois VGG-19, grâce aux filtres convolutifs extrait les caractéristiques locales du visage, tandis que le modèle siamois Vision Transformer capture efficacement les dépendances globales grâce à leurs mécanismes d'attention. Les résultats expérimentaux sur notre ensemble de données ont atteint des taux de précisions égales à 94,05 % et 91,27 % (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer).

Mots clés : Apprentissage Profond, CNN, Vision Transformer, Réseaux de Neurones Siamois, Vision par ordinateur, Reconnaissance des visages.

ABSTRACT

Facial recognition is a key technology in the fields of security, surveillance, and human-machine interaction. With the advancement of deep learning techniques, facial recognition models have evolved considerably, enabling impressive performance in terms of accuracy and robustness.

In this report, we have developed two face recognition models with Siamese networks that are based on Vision Transformers and Convolutional Neural Networks (respectively Siamese Network VGG-19 and Siamese Vision Transformer). These models are designed to compare two input facial images and return a similarity score for recognition. In addition, due to the lack of a large and diverse dataset that can be used to evaluate facial recognition, even when the person is wearing a mask, we constructed our own dataset. This consists of video captures containing information about the faces of various celebrities. For each celebrity, we have collected images with masked and unmasked faces to enable our models to identify the person even when they are wearing a face mask. The data is then prepared and generated in pairs. The VGG-19 Siamese Network model extracts local facial features thanks to convolutional filters, while the Vision Transformer Siamese model efficiently captures global dependencies thanks to their attention mechanisms. Experimental results on our dataset achieved accuracies equal to 94,05% and 91,27% (Siamese Network VGG-19 and Siamese Vision Transformer respectively).

Key words: Deep Learning, CNN, Vision Transformer, Siamese Neural Networks, Computer Vision, Face Recognition.

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

Tout d'abord, je remercie chaleureusement mon directeur de recherche le professeur Fadel TOURE et sa codirectrice Mme Oumayma OUEDRHIRI, pour leurs encadrements, leurs conseils précieux et leurs soutiens constants tout au long de cette recherche. Leurs remarques judicieuses et leurs expertises ont été déterminantes pour l'aboutissement de ce travail.

Je souhaite également remercier les membres du jury pour avoir accepté d'évaluer ce mémoire et pour l'intérêt qu'ils ont manifesté à l'égard de mon travail.

Un grand merci à mes professeurs et à mes collègues de l'Université du Québec à Trois-Rivières, qui ont su partager avec moi leur savoir, leurs idées et leur enthousiasme. Leurs soutiens et leurs encouragements ont été essentiels dans la poursuite de mes études et de cette recherche.

Je tiens également à exprimer ma reconnaissance envers ma famille et mes amis, pour leur soutien moral, leur compréhension et leur patience durant cette période exigeante. Leur présence à mes côtés a été une source inestimable de réconfort et de motivation.

Enfin, je souhaite remercier toutes les personnes qui ont contribué, de près ou de loin, à ce projet, que ce soit par leur aide technique, leurs encouragements ou leur amitié. Chacun de vous a joué un rôle dans la réalisation de ce mémoire, et je vous en suis profondément reconnaissant.

TABLE DES MATIÈRES

LISTES DES FIGURES

LISTE DES TABLEAUX

LISTES DES ABRÉVIATIONS

GAN: Generative Adversarial Network

CelebA: CelebFaces Attributes Dataset

DS: Dataset

YOLO: You Only Look Once

Faster-RCNN: Faster Region-based Convolutional Neural Network

ResNet: Residual Network

SVM: Support Vector Machines

RMFD: Real-World Masked Face Dataset

SMFD: Simulated Masked Face Dataset

LFW: Labeled Faces in the Wild

SSDMNV2: Single Shot Multibox Detector MobilenetV2

UPA: Upper Patch Attention

VOC: Visual Object Classes

VGG: Visual Geometry Group

ViT: Vision Transformers

HiMFR: Hybrid Masked Face Recognition

MAFA: Mask Face

DL: Deep Learning

CNN: Convolutional Neural Network

RELU: Rectified Linear Unit

CHAPITRE 1 : INTRODUCTION

1.1 Contexte et motivation

À l'ère numérique, la vérification et la reconnaissance de l'identité recherchent progressivement la commodité et la dissimulation et utilisent de plus en plus les caractéristiques biologiques comme système de vérification. Comment vérifier l'identité des personnes rapidement, facilement et efficacement est devenu un sujet de recherche actif, et les résultats de la recherche peuvent être appliqués à de nombreuses occasions, telles que la vérification de l'identité des gares, le contrôle de la circulation, l'accès, etc.

La reconnaissance faciale est une technologie de plus en plus populaire dans tous les domaines, allant de la sécurité publique aux applications mobiles. Cependant, en décembre 2019, Wuhan (capitale de la province du Hubei en Chine) est devenue le centre d'une épidémie de pneumonie de cause inconnue, qui a suscité une attention intense non seulement en Chine, mais à l'échelle internationale(Wang et al., 2020). L'épidémie est devenue par la suite une pandémie mondiale appelée COVID-19. Des mesures de lutte contre la propagation du virus ont été prises comme le port de masques, la distanciation sociale, etc. L'arrivée des vaccins a initialement apporté un certain soulagement parmi la population du monde entier. Cependant, les cas d'infections au COVID-19 chez les personnes vaccinées ont soulevé une situation alarmante, d'où l'importance de suivre les mesures prises (c'est-à-dire le port de masques faciaux, la distanciation sociale, etc.) par l'OMS (OMS, 2020).

Avant cette pandémie, les masques étaient couramment utilisés par la population pour se protéger de la pollution de l'air (McDonald et al., 2020) ou par le personnel paramédical dans les hôpitaux ou par les criminels pour cacher leurs identités. Cependant, pendant la pandémie, le port des masques faciaux était obligatoire dans les espaces publics pour empêcher la propagation du COVID-19.

Des chercheurs (Al-Ramahi et al., 2021) révèlent que le port d'un masque facial réduit l'expansion du COVID-19 en diminuant la probabilité de transmission de gouttelettes respiratoires

contaminées (chargées de virus). C'est pourquoi de nombreux pays exigent que les gens portent des masques dans les lieux publics pour empêcher la propagation du COVID-19.

Inspecter manuellement les personnes dans les lieux publics et privés pour vérifier leurs identités même quand elles portent des masques faciaux est une tâche difficile. En conséquence, il est nécessaire de créer des systèmes automatisés pour identifier les personnes même si elles portent des masques faciaux.

1.2 Problématique

Le port du masque introduit de nouveaux défis pour les applications traditionnelles de reconnaissance faciale, qui sont généralement conçues pour les visages dévoilés. Ces applications installées à différents points de contrôle (Damer et al., 2021) montrent des performances dégradées dans le cas des visages masqués. Ceci est à cause de la perte d'informations importantes sur des segments du visage tels que le nez, les lèvres, le menton, joues, etc. (Du et al., 2021). L'échec des méthodes de reconnaissance faciale dans le cas des masques faciaux a soulevé des défis importants pour les applications de vérification/authentification telles que les paiements mobiles, les examens de sécurité publique, le déverrouillage des téléphones et la présence, etc. (Ullah et al., 2022). De même, dans les points de contrôle de la sécurité publique tels que les gares ferroviaires et routières, les portes d'entrée sont équipées de caméras installées et s'appuient sur les méthodes conventionnelles de reconnaissance faciale qui ne sont pas en mesure de reconnaître les personnes portant des masques.

La pandémie de COVID a découragé l'utilisation de diverses méthodes biométriques traditionnelles telles que la reconnaissance des empreintes digitales, la reconnaissance faciale, etc., car ces méthodes peuvent propager le virus parmi les utilisateurs. Les systèmes automatisés de vérification des utilisateurs capables de reconnaître les personnes en présence de masques faciaux semblent une solution efficace dans cette pandémie. Cependant, les méthodes de reconnaissance faciale existantes ne sont pas très robustes pour les visages masqués. Par conséquent, cette situation pandémique exige un besoin urgent de développer des systèmes de reconnaissance faciale robustes, capables de reconnaître toute personne portant un masque facial. Il est important d'améliorer les méthodes actuelles qui s'appuient sur tous les points caractéristiques du visage, de

sorte qu'une authentification fiable puisse être effectuée dans le cas des visages masqués avec moins d'expositions du visage.

1.3 Objectifs

L'objectif consiste à évaluer la capacité des réseaux de neurones profonds à identifier une personne à partir de son visage, malgré le port du masque. Pour ce faire, nous allons utiliser les réseaux siamois largement utilisés pour comparer des images. Ainsi, deux réseaux siamois avec des modèles de base de différents seront comparés dans ce mémoire. Ainsi, en raison du manque d'un ensemble de données étendu pour les tâches de la reconnaissance de visage, nous visons à construire notre propre ensemble de données composées des images d'individus avec masque et sans masque, dans le but d'identifier une personne même quand elle porte un masque facial.

Pour répondre à ce défi, nous visons à développer un réseau siamois avec deux architectures : l'une utilisant des Vision Transformers (ViT) et l'autre des réseaux de neurones convolutifs (CNN), spécifiquement VGG-19. Le réseau siamois est une architecture bien adaptée aux tâches de vérification d'identité, car il apprend à comparer des paires d'images et à évaluer leurs différences ou ressemblances. En intégrant les capacités de représentation visuelle des ViT et des CNN, nous visons à améliorer la robustesse et la précision de la reconnaissance faciale en présence de masques.

1.4 Contributions

Dans ce travail, nous avons développé deux architectures de réseaux siamois basés respectivement sur le modèle CNN en utilisant le modèle VGG-19 préentrainé et le modèle ViT pour la reconnaissance faciale. Dans un premier temps, nous avons créé notre propre ensemble de données en collectant des images de visages avec masque et sans masque de différents individus. Les images sont ensuite redimensionnées et organisées par pair (positif ou négatif) pour servir de base d'entrainement aux modèles. Nous avons fourni une évaluation détaillée des performances des architectures proposées sur l'ensemble de données. Les modèles ont été évalués selon des

métriques standards telles que la précision, le rappel, le F1-score, la matrice de confusion, La courbe Precision-Recall (PR) et l'AUC (Area Under the Curve) et La courbe AUC-ROC (Area Under the Curve - Receiver Operating Characteristic). Nous avons ensuite comparé les résultats obtenus par les deux modèles.

CHAPITRE 2 : ÉTAT DE L'ART

2.1 Introduction

La technologie de reconnaissance faciale est un des domaines de recherche qui a suscité le plus d'intérêt au cours de ces dernières années. Elle comprend généralement quatre étapes, qui sont la détection des visages, l'alignement, l'extraction des caractéristiques faciales et la classification des visages. Les caractéristiques utilisées incluent souvent le nez, la bouche et les yeux. Cependant, diverses situations et circonstances, comme la pandémie de COVID-19, ont rendu le port de masques courant, ce qui cache une partie du visage et affecte donc la précision de la reconnaissance faciale. Cela appuie la nécessité de construire des modèles d'apprentissage capables d'identifier avec précision les visages humains masqués. De nombreuses études ont été réalisées pour relever le défi de la reconnaissance faciale dans le domaine de l'apprentissage profond.

2.2 Reconnaissance faciale et réseaux de neurones

Les réseaux neuronaux convolutifs (CNN) ont révolutionné la reconnaissance faciale grâce à leur capacité à apprendre des représentations hiérarchiques et discriminantes à partir d'un ensemble d'images brutes. Les CNN peuvent automatiquement extraire des caractéristiques pertinentes à plusieurs niveaux de complexité, rendant le processus plus efficace et plus précis. Dans cette partie, les méthodes comme YOLO, Faster-RCNN, MobileNet, ResNet et VGGNet sont examinées pour leur application à la reconnaissance des visages masqués.

Din et al. (Din et al., 2020) ont proposé une nouvelle méthode pour la suppression sans interaction d'objets volumineux dans les images faciales, en se concentrant sur l'objet masqué. Pour compléter l'image, ils ont utilisé une peinture d'image basée sur le GAN par le biais d'une approche de traduction d'image à image afin de produire des résultats plausibles. Pour ce faire, ils ont utilisé un réseau basé sur le GAN ayant deux discriminateurs : le premier discriminateur aide à apprendre la structure globale du visage et l'autre intervient pour concentrer l'apprentissage profonde sur la région manquante. Pour entraîner leur modèle de manière supervisée, ils ont créé

un ensemble de données synthétiques appariées en utilisant l'ensemble de données CelebA disponibles publiquement et l'ont testé sur des images réelles collectées sur Internet. Leur modèle a surpassé d'autres approches représentatives de l'état de l'art, tant sur le plan qualitatif que quantitatif.

Militante et Dionisio (Militante & Dionisio, 2020) ont proposé une approche utilisant l'apprentissage profond et les réseaux de neurones convolutifs (CNN) pour les systèmes de reconnaissance faciale masquée en temps réel. L'étude a utilisé des techniques d'apprentissage profond pour faire la reconnaissance faciale et reconnaître si la personne porte un masque ou non. L'ensemble de données recueilli contient 25 000 images d'une résolution de 224x224 pixels. Le système de reconnaissance de masque facial en temps réel développé est basé sur un Raspberry Pi qui alerte et capture l'image faciale si la personne détectée ne porte pas de masque. Cette étude est utile pour lutter contre la propagation du virus. Ce processus donne des résultats précis et rapides pour la détection des masques. Les résultats donnent un taux d'exactitude de 96 % dans la détection des personnes portant un masque facial et de celles qui n'en portent pas.

Venkateswarlu et al.(Venkateswarlu et al., 2020) ont présenté un MobileNet avec un bloc de pooling global pour la détection des masques faciaux. L'observation manuelle du masque facial dans les endroits très fréquentés est une tâche essentielle. C'est ainsi que, les chercheurs ont motivé l'automatisation du système de détection des masques faciaux. MobileNet est pré-entraîné avec un bloc de pooling global pour la détection de ces masques. Le modèle pré-entraîné prend une image couleur et génère une carte de caractéristiques multidimensionnelle. Le bloc de pooling global qui a été utilisé dans le modèle proposé transforme la carte de fonctionnalités en un vecteur de 64 caractéristiques. Enfin, la couche softmax effectue une classification binaire à partir 64 caractéristiques. Le modèle proposé a été évalué sur deux ensembles de données accessibles au public. Il a atteint une précision de 99 % et 100 % sur DS1 et DS2 respectivement. Le bloc de pooling global qui a été utilisé dans le modèle proposé évite le surajustement du modèle.

G. Yang et al. (Yang et al., 2020) ont proposé de remplacer l'inspection manuelle des visages avec masques par une méthode d'apprentissage profond dans le but est de contrôler les personnes avant d'entrer dans un magasin. Ils ont développé une application basée sur YOLOV5 permettant de reconnaître si le visage porte ou non un masque. Pour la reconnaissance, Il suffit de se tenir devant la caméra qui va détecter la présence d'un visage, si la personne porte un masque

le système affiche dans l'interface succès et la porte s'ouvre. Leur expérience a montré un taux de réussite d'environ 97,9 %. Les résultats expérimentaux montrent que l'algorithme proposé dans cet article peut reconnaître efficacement les masques faciaux et réaliser une surveillance efficace du personnel.

Singh et al. (Singh et al., 2021) ont développé deux modèles pour réaliser la tâche de reconnaissance de visage masqué à savoir YOLOv3 et Faster-RCNN. Les deux modèles ont été formés sur un ensemble de données composé d'images de personnes de deux catégories de visages : avec et sans masque facial. Ils ont utilisé une technique qui encadre et délimite (rouges ou verts) des visages des personnes, selon qu'une personne porte ou non un masque. Les résultats sont enregistrés pour chaque catégorie dans une base quotidienne. Les auteurs ont également comparé les performances des deux modèles, c'est-à-dire leur taux de précision et leur temps d'inférence. Faster-RCNN a une meilleure précision, mais pour l'appliquer aux caméras de surveillance du monde réel, il serait préférable d'utiliser le modèle avec l'algorithme YOLO, car il effectue une détection unique et a une fréquence d'images beaucoup plus élevée que Faster-RCNN.

Loey et al. (Loey et al., 2021) ont réalisé une étude portant sur un modèle hybride utilisant l'apprentissage automatique profond et classique pour la détection des visages masqués. Le modèle proposé se compose de deux éléments. Le premier composant est conçu pour l'extraction de fonctionnalités à l'aide de Resnet50, tandis que le deuxième composant est conçu pour le processus de classification des masques faciaux à l'aide d'arbres de décision, d'une machine à vecteurs de support (SVM) et d'un algorithme d'ensemble. Trois ensembles de données avec visages masqués ont été sélectionnés pour cette étude. On trouve l'ensemble de données sur les visages masqués du monde réel (RMFD), l'ensemble de données sur les visages masqués simulés (SMFD) et les visages étiquetés dans la nature (LFW). Le classificateur SVM a atteint une précision de test de 99,64 % en RMFD. En SMFD, il a atteint 99,49 %, tandis qu'en LFW, il a atteint une précision de test de 100 %. L'une des tâches futures possibles consiste à utiliser des modèles d'apprentissage par transfert plus approfondis pour l'extraction de caractéristiques et à utiliser le domaine neutrosophique, car il montre un potentiel prometteur dans les problèmes de classification et de détection.

Nagrath et al. (Nagrath et al., 2021) ont proposé une approche basée sur l'apprentissage profond en utilisant TensorFlow, Keras et OpenCV pour détecter les masques faciaux. L'objectif de leur recherche était de détecter en temps réel les masques faciaux. L'approche SSDMNV2 utilise Single Shot Multibox Detector comme détecteur de visage et l'architecture MobilenetV2 comme cadre pour le classificateur, qui est très léger et peut même être utilisé dans des appareils intégrés (comme NVIDIA Jetson Nano, Raspberry pi) pour effectuer une détection de masque en temps réel. L'ensemble de données de visages masqués utilisé dans cette recherche provient de la plateforme Kaggle. La technique déployée dans cet article donne un score de précision de 93 % et un score F1 de 93 %. Les performances élevées obtenues démontrent l'efficacité de leur approche dans la détection des masques faciaux en temps réel.

Naeem Ullah et al. (Ullah et al., 2022) ont proposé un nouveau framework DeepMasknet capable à la fois de détecter les visages avec masque et de faire de la reconnaissance faciale. De plus, il manque actuellement un ensemble de données étendu et diversifié pouvant être utilisé pour évaluer à la fois la détection et la reconnaissance. À cette fin, ils ont également développé un ensemble de données étendu et diversifié de détection de masque et de reconnaissance faciale masquée (MDMFR) à grande échelle pour mesurer les performances des méthodes. Une précision de 100 % pour la détection des visages avec masques et de 93,33 % pour la reconnaissance faciale a confirmée la supériorité du modèle DeepMasknet sur les techniques contemporaines. De plus, les résultats expérimentaux sur les ensembles de données ont vérifié la robustesse du modèle DeepMasknet proposé dans diverses conditions, c'est-à-dire les variations des angles du visage, les conditions d'éclairage, le sexe, le teint, l'âge, types de masques, occlusions (lunettes), etc. La méthode proposée peut être utilisée à diverses fins, notamment la sécurité et la sûreté des personnes, c'est-à-dire la reconnaissance des voleurs portant un masque facial, etc.

Yuxuan Zhang et al. (Zhang et al., 2022) ont proposé un nouveau modèle CNN profond guidé par l'attention pour résoudre le problème de la reconnaissance des visages masqués. Ils ont développé une stratégie d'entraînement à double branche pour guider le modèle afin qu'il se concentre sur la moitié supérieure du visage afin d'extraire des caractéristiques. Pendant la formation, les fonctionnalités apprises au niveau des couches intermédiaires de la branche globale sont transmises à un module d'attention proposé, nommé Upper Patch Attention (UPA) qui agit comme une branche locale. Les deux branches sont optimisées conjointement pour améliorer l'extraction de fonctionnalités à partir de régions non occultées. Ils proposent également un module

d’auto-attention, qui s’intègre au réseau fédérateur pour améliorer l’interaction entre les canaux et les emplacements spatiaux dans le processus d’apprentissage. Des expériences approfondies sur des ensembles de données de visages synthétiques et masqués réels démontrent l’efficacité de leur méthode.

Emrah Aydemir et al. (Aydemir et al., 2022) ont présenté une étude traitant le problème de l’utilisation inappropriée des masques. Pour résoudre la problématique, 2 075 images d’utilisation de masques faciaux ont été collectées. Les images individuelles ont été étiquetées comme étant un masque, non masqué ou un masque inapproprié. Sur la base de ces étiquettes, ils ont présenté les trois cas suivants : Cas 1 : masque versus pas de masque versus masque inapproprié, Cas 2 : masque versus pas de masque + masque inapproprié et Cas 3 : masque versus pas de masque. Ces données ont été utilisées pour former et tester un modèle hybride de classification approfondie des visages masqués. Ensuite, ils ont proposé un modèle hybride de classification approfondie. Leur modèle comprend un ResNet101 et un DenseNet201 pré-entrainé, qui ont été utilisé comme générateurs de fonctionnalités. Chacun de ces générateurs extrait les caractéristiques les plus discriminantes. Les fonctionnalités choisies ont été utilisées pour former et tester un classificateur de machines à vecteurs de support. Les résultats du modèle ont atteint des taux d’exactitude de classification de 95,95 %, 97,49 % et 100,0 % respectivement dans les cas 1, 2 et 3. Le fait d’avoir atteint ces valeurs de haute précision indique que leur modèle s’est bien adapté à un essai pratique visant à détecter l’utilisation appropriée d’un masque facial en temps réel.

Md Hafizur Rahman et al. (Rahman et al., 2023) ont développé un système capable d’identifier automatiquement la position du masque facial. Dans le but de garantir si une personne porte correctement un masque avant d’entrer dans un espace public pour lutter contre la propagation de la Covid-19. Dans un premier temps, ils ont développé et publié un ensemble de données d’images de visage avec masques, collectées auprès de 391 personnes de différents groupes d’âge et de sexe. Ensuite, ils ont étudié six architectures différentes de modèles d’apprentissage profond pré-entraînés, et ont proposé enfin un modèle qu'ils ont développé, en affinant le modèle MobileNet de pointe. Les performances (précision, score F1 et Kappa de Cohen) de ce modèle ont été évaluées sur l’ensemble de données proposé et MaskedFace-Net, un ensemble de données synthétiques accessible au public créé par édition d’images. Les performances ont également été comparées à d’autres méthodes existantes. Le MobileNet proposé

s’avère être le meilleur modèle offrant une précision, un score F1 et un Kappa de Cohen respectivement de 99,23 %, 99,22 % et 99,19 %.

Wei Zeng et al. (Zeng et al., 2023) ont proposé une méthode de détection de visages masqués sur l’apprentissage profond. Ils ont utilisé le modèle YOLOX basé sur un ConvNeXt amélioré qui intègre de nombreux avantages de Swin-Transformer avec des performances de classification efficaces. Pour résoudre les problèmes de convergence instable de l’intersection sur union (IOU) et de coordonnées de régression inexactes de l’algorithme, ils ont adopté la distance de Manhattan comme nouveau terme de pénalité, et la nouvelle fonction de perte combine les avantages de la distance euclidienne et de la distance de Manhattan. Il n’y a pas eu d’explosion de gradient au début de la période d'entraînement, et il a pu descendre progressivement jusqu’à un point stable. L’algorithme proposé a été robuste aux effets négatifs tels que la poussière, la lumière, les ombres et les occultations, et convient bien à la détection des visages masqués dans des scénarios industriels et publics. Ils ont testé le modèle sur l’ensemble de données en classes d’objets visuels (VOC) et l’ensemble de données open source à visages masqués AIZOO, obtenant une amélioration de 5,58 % sur l’ensemble de données VOC et atteignant une précision de 96,66 % sur l’ensemble de données à visages masqués.

Depuis le début de la pandémie de COVID-19, il est demandé à chacun de porter un masque facial qui cache la partie inférieure de la zone du visage. Seuls quelques modèles de reconnaissance faciale existants sont capables de détecter et de reconnaître les visages masqués, mais la précision moyenne s’est détériorée par rapport à ceux qui détectent le visage entier d’une personne. Poonpinij et al. (Poonpinij et al., 2023) se sont fixés comme objectif de développer de nouveaux modèles capables de détecter et de reconnaître les visages masqués, et de comparer leurs performances avec les modèles existants. Trois réseaux de neurones convolutifs (CNN) sont utilisés, à savoir VGG16, VGGFace et InceptionResNetV2 (IRv2). De nombreux cas expérimentaux sont considérés, notamment les cas où seules des images de visage masqué sont utilisées, ou les deux autres cas où des images de visage complet et des images de visages masqués sont utilisées ensembles. Les modèles donnent une précision maximale à 93,3 % où 50 individus sont reconnus et identifiés.

2.3 Reconnaissance faciale et Vision Transformer (ViT)

Les Vision Transformers (ViTs) sont une approche plus récente dans le domaine de la vision par ordinateur. Contrairement aux réseaux convolutifs traditionnels, les ViT utilisent une architecture basée sur les transformateurs, initialement développée pour le traitement du langage naturel. Les ViT ont montré des performances prometteuses dans diverses tâches de vision par ordinateur, y compris la reconnaissance de visage. Leur capacité à capturer des relations globales dans une image grâce au mécanisme d'attention permet d'extraire des caractéristiques riches et discriminatives des visages.

Inspirés par des récentes méthodes de retouche d'image, Md Imran Hosen et Md Baharul Islam (Hosen & Islam, 2022) ont proposé un système hybride de reconnaissance faciale masquée de bout en bout appelé HiMFR, composé de trois parties importantes : détecteur de visage masqué, retouche de visage et reconnaissance faciale. Le module de détection de visage masqué applique un Vision Transformer pré-entraînée (ViT_b32) pour détecter si les visages sont recouverts de masques ou non. Le module de retouche utilise un modèle de retouche d'image finement ajusté basé sur un réseau antagoniste génératif (GAN) pour restaurer les visages. Enfin, le module de reconnaissance faciale hybride basé sur ViT avec un backbone EfficientNetB3 reconnaît les visages. Ils ont implémenté et évalué leur modèle sur quatre ensembles de données différentes disponibles au public : CelebA, SSDMNV2, MAFA, Pubfig83 avec leur petit ensemble de données collecté localement, à savoir Face5. Les résultats expérimentaux montrent l'efficacité de leur méthode avec des performances compétitives. Bien que le modèle puisse reconnaître les visages masqués avec une grande précision, il peut échouer en cas de complétion inexacte du visage par le module d'incrustation d'image (en raison d'un suivi et d'une segmentation imprécise du masque du visage).

Zhonglin Sun et Georgios Tzimiropoulos (Sun & Tzimiropoulos, 2022) ont proposé un modèle de reconnaissance faciale. Dans un premier temps, ils ont utilisé le Vision Transformer comme architecture pour former une base de référence très solide pour la reconnaissance appelée fViT. Ensuite, ils ont capitalisé sur la propriété inhérente des Transformers à traiter les informations (jetons visuels) extraites de grilles irrégulières pour concevoir un pipeline qui rappelle les méthodes basées sur des parties. Leur pipeline est formé de bout en bout sans

supervision de repère et comprend un réseau léger pour prédire les coordonnées des repères faciaux suivis par le Vision Transformer opérant sur des patchs extraits des repères prédits. Le modèle en apprenant à extraire des patchs discriminants améliore la précision de leur base de référence en atteignant une bonne précision lors de plusieurs tests de reconnaissance faciale.

Inspirés par l'application réussie de l'auto-attention en vision par ordinateur, Yiming Ge et al. (Ge et al., 2023) ont proposé un réseau convolutionnel d'auto-attention visuelle (CVSAN), qui utilise l'auto-attention pour augmenter l'opérateur de convolution. Cela a été réalisé en connectant une carte de caractéristiques convolutionnelles, qui applique les fonctionnalités locales, à une carte de fonctionnalités d'auto-attention capable de modéliser des dépendances à longue portée. Puisqu'il n'existait aucune donnée sur les visages masqués à grande échelle accessible au public, ils ont généré un ensemble de données contenant des visages masqués à partir du VGGFace2 pour entraîner le modèle. Les expériences montrent que l'algorithme CVSAN améliore considérablement les performances de reconnaissance des visages masqués (MFR) par rapport à d'autres algorithmes. Les résultats expérimentaux sur plusieurs ensembles de données de test montrent que le modèle proposé atteint des performances satisfaisantes en matière de reconnaissance des visages masqués.

Minchul Kim et al. (Kim et al., 2024) ont abordé le défi de rendre les modèles de Visons Transformers (ViTs) plus robustes aux transformations affines invisibles. Une telle robustesse devient utile dans diverses tâches de reconnaissance telles que la reconnaissance faciale lorsque des échecs d'alignement d'images se produisent. Ils ont proposé une nouvelle méthode appelée KP-RPE qui exploite les points de repère faciaux pour rendre les ViT plus robustes face à la translation d'échelle et aux variations de pose. Ils ont commencé par observer que le codage de position relative (RPE) est un bon moyen d'apporter une généralisation de transformation affine aux ViT. Le RPE ne peut cependant injecter au modèle qu'une connaissance préalable selon laquelle les pixels proches sont plus importants que les pixels éloignés. Le RPE des points clés (KP-RPE) est une extension de ce principe où la signification des pixels n'est pas uniquement dictée par leur proximité, mais aussi par leurs positions relatives par rapport à des points clés spécifiques dans l'image. En ancrant la signification des pixels autour des points clés, le modèle peut conserver plus efficacement les relations spatiales même lorsque ces relations sont perturbées par des transformations affines. Les résultats expérimentaux démontrent l'efficacité de la

reconnaissance faciale à partir d'images de faible qualité, en particulier lorsque l'alignement est sujet à des échecs.

Keresh et al. (Keresh & Shamoi, 2024) ont abordé la vulnérabilité aux attaques d'usurpation d'identité, où les attaquants utilisent des photos, des vidéos ou des masques pour se faire passer pour des utilisateurs légitimes. Ils ont utilisé l'architecture du ViT avec le Framework DINO sur la tâche anti-usurpation d'identité faciale. Le Framework DINO facilite l'apprentissage auto supervisé, permettant au modèle d'apprendre des caractéristiques distinctives à partir de données non étiquetées. Les auteurs ont comparé les performances du modèle ViT proposé à l'aide du Framework DINO par rapport à un modèle CNN traditionnel EfficientNet b2. De plus, ils ont collecté leur propre ensemble de données à partir d'une application biométrique pour valider davantage leurs résultats. De nombreux tests sur des ensembles de données standard ont montré que le modèle ViT est plus performant que le modèle CNN en termes de précision et de résistance à différentes méthodes d'usurpation d'identité. Cette étude met en évidence les performances supérieures de l'architecture basée sur un transformateur dans l'identification d'indices d'usurpation d'identité complexes, ce qui conduit à des avancées significatives en matière de sécurité biométrique.

2.4 Reconnaissance faciale et Réseaux siamois

Les réseaux siamois sont souvent utilisés pour l'identification de visages, où le modèle apprend à distinguer les paires de visages similaires des paires de visages dissemblables. Des architectures convolutives (CNN) telles que VGG-Face, FaceNet, et DeepFace ont été largement utilisées comme base pour les réseaux siamois en raison de leur capacité à extraire des caractéristiques discriminatives à partir des images de visage.

Ahmed et al. (Ahmed et al., 2020) ont proposé une approche de vérification d'identité faciale qui est basée sur l'architecture hybride des réseaux siamois sous des ressources limitées. Ils ont développé une architecture simple qui apprend la similitude et la dissimilarité des visages à partir des images en entrées. Le modèle a atteint un taux de précision de 98,9 % sur l'ensemble de données Labeled Faces in the Wild (LFW) et 99,1 % sur l'ensemble de données des visages arabes. Ce qui est compétitif par rapport aux autres techniques présentées dans la littérature. De

plus, les caractéristiques apprises par l'architecture proposée sont utilisées dans la construction d'un système de clustering de visages en utilisant une version mise à jour du clustering spatial basé sur la densité des applications avec bruit (DBSCAN). Pour relever le défi de la qualité du cluster, une nouvelle procédure d'optimisation post-clustering est proposée. Elle surpasse les approches de clustering courantes, comme K-Means et spectral de 0,098 et jusqu'à 0,344 selon la mesure F1.

Xian-Feng Xu et al. (Xu et al., 2020) ont proposé une structure simplifiée de réseau neuronal convolutionnel profond conçue pour la reconnaissance faciale. Ayant une grande robustesse dans des conditions illimitées, cette structure peut améliorer la vitesse d'apprentissage et la précision de la reconnaissance faciale, et convient aux ensembles de données à petite échelle. Ils ont développé un modèle de réseau siamois basé sur l'architecture Inception. Ils ont utilisé les bases de données faciales standard CASIA-webface et Extended Yale B pour entraîner et tester leur modèle. Il a atteint une précision de 99,36 % et 99,21 % respectivement dans les bases de données CASIA-webface et Extended Yale B. Le résultat de la simulation démontre qu'il peut atteindre une grande précision et qu'il s'agit d'un excellent algorithme de reconnaissance faciale.

Yazid Aufar et Imas Sukaesih Sitanggang (Aufar & Sitanggang, 2022) ont proposé une méthode pour améliorer les performances de détection des visages à l'aide du CNN siamois. Ils ont utilisé la technique de l'augmentation pour améliorer leur modèle. Les expériences montrent que la méthode proposée pour la détection des visages à l'aide de la technique d'augmentation donne des résultats supérieurs à celle qui ne l'utilise pas. L'ensemble de données d'images LFW a été utilisé pour tester le modèle. Il a été testé sur un total de 9 000 images de visages, avec un taux de précision de 98 %. Le taux de confiance de la reconnaissance est influencé par le nombre de photos utilisées pour l'entraînement.

Cunli Song et Shouyong Ji (Song & Ji, 2022) ont conçu un réseau neuronal siamois basé sur le modèle binaire local (également appelé LBP) et la perception des caractéristiques de fréquence pour réaliser la reconnaissance faciale dans des conditions non restreintes. L'algorithme LBP est utilisé pour éliminer l'effet de l'éclairage sur l'image et fournir une entrée de niveau vectoriel au modèle de réseau. La perception des caractéristiques de fréquence est utilisée pour diviser les caractéristiques de l'image en caractéristiques basse fréquence et caractéristiques haute fréquence. Les caractéristiques basse fréquence sont compressées dans le réseau neuronal siamois pour augmenter l'efficacité de reconnaissance du réseau tout en échangeant des informations avec

les caractéristiques haute fréquence pour conserver leurs données de caractéristiques tout en éliminant les données de bruit cible. Cela maintient le taux de reconnaissance du réseau et améliore la vitesse de calcul du réseau. Des expériences de simulation sont menées sur les jeux de données de visages standard CASIA-WebFace, Yale-B et LFW, et comparées à d'autres modèles de réseau. Les résultats expérimentaux montrent que le modèle proposé dans cet article peut améliorer la précision de reconnaissance faciale.

Solomon et al. (Solomon et al., 2023) ont proposé d'utiliser les réseaux siamois pour la reconnaissance des visages, éliminant ainsi la nécessité de disposer d'images de visages étiquetées. Ils y sont parvenus en exploitant stratégiquement les échantillons négatifs avec leurs équivalents les plus proches, établissant ainsi des paires positives et négatives par le biais d'une méthodologie non supervisée. Les données d'entraînement positives sont sélectionnées au sein d'un ensemble de données sur la base de leurs scores de similarité en cosinus les plus élevés avec une ancre désignée, tandis que les données d'entraînement négatives sont éliminées de manière parallèle, bien qu'elles soient tirées d'un autre ensemble de données. Le cadre architectural adopte un encodeur VGG, entraîné comme un réseau siamois à double branche. Pendant la formation, le réseau siamois proposé a effectué une classification binaire par perte d'entropie croisée. Ensuite, pendant la phase de test, ils ont extrait directement les scores de vérification des visages de la couche de sortie du réseau. Les résultats expérimentaux ont révélé que leur modèle offre des performances comparables à celles d'un système de base similaire qui sont entièrement supervisées.

2.5 Conclusion

La reconnaissance de visage est un domaine clé de la vision par ordinateur et de l'intelligence artificielle. Les CNN, les Visions Transformers et les réseaux siamois représentent des approches puissantes et complémentaires pour la reconnaissance de visage. Les réseaux de neurones convolutifs (Convolutional Neural Networks, CNNs) ont révolutionné la reconnaissance de visage grâce à leur capacité à apprendre des représentations hiérarchiques des images. Les réseaux siamois ont une longue histoire de succès dans ce domaine grâce à leur efficacité et leur robustesse, tandis que les ViT offrent une approche novatrice capable de capturer des relations plus globales dans les images. L'avenir de la reconnaissance de visage pourrait bénéficier de la combinaison de

ces deux approches, exploitant les forces des convolutions pour l'extraction de caractéristiques locales et des transformateurs pour l'intégration contextuelle globale.

CHAPITRE 3: CONCEPTS ET DÉFINITIONS

3.1 Introduction

La reconnaissance de visage est une technologie de plus en plus utilisée dans divers domaines. Il existe plusieurs architectures qui sont très prometteuses pour cette tâche, notamment les Visions Transformers (ViT), les réseaux de neurones convolutifs (CNN) et les réseaux de neurones siamois. Ce chapitre introduit les concepts et les définitions de ces différents modèles, expliquant leurs caractéristiques et leurs applications.

3.2 Neurone biologique

Le neurone est l'unité de base du système nerveux central. Il est constitué d'un corps cellulaire ou soma, qui produit des branches appelées dendrites. Les dendrites transmettent les informations de l'extérieur ou d'autres neurones au soma. Une fois que le corps cellulaire a traité les informations, il les transmet aux autres neurones via un processus de sortie appelé axone. À la jonction d'un axone et d'une dendrite se trouve un minuscule espace appelé synapse. La transmission à travers une synapse est de nature chimique.

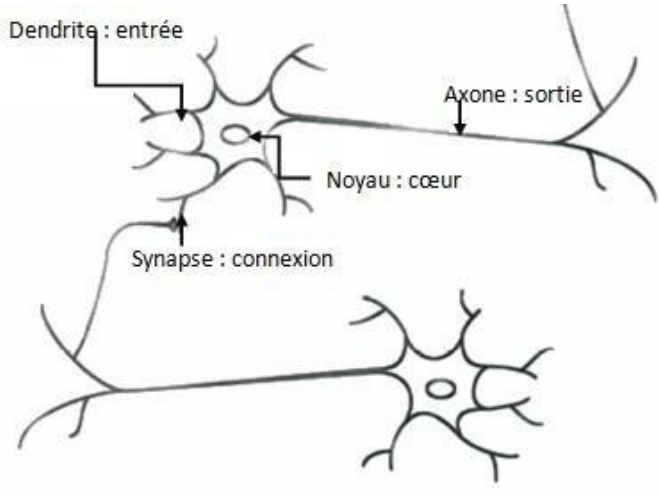

Figure 3.1 (Gruet, 2018) : Neurone biologique

Le cerveau est composé d'environ 86 à 100 milliards de cellules nerveuses, ou neurones (Neuromedia, 2021). Chaque neurone a une sortie qui est connectée à l'entrée de milliers d'autres neurones. Le traitement local est assuré par le neurone lui-même : il collecte les entrées (signaux) au niveau d'extensions spéciales appelées dendrites et les additionne dans le noyau du neurone ; si l'amplitude résultante dépasse un seuil interne, le neurone envoie un signal par sa connexion de sortie, l'axone, à d'autres neurones.

3.3 Neurone artificiel

Un neurone artificiel est une fonction mathématique dont la conception est inspirée du fonctionnement des neurones biologiques. Le neurone artificiel peut être considéré comme un opérateur qui prend un nombre variable d'entrées provenant de l'environnement externe ou d'autres neurones, chacune de ces entrées étant affectée d'un poids, appelé poids synaptique, et une sortie n'est produite que si cette somme dépasse un seuil interne.

L'évaluation de la sortie se fait par la somme pondérée des entrées et passé le résultat à une fonction non-linéaire φ. Cela peut être mathématiquement modélisé par l'équation suivante :

$$S = \sum_{m=1}^{n} w_m . x_m + \mathrm{b}$$

x_m : Composantes du vecteur d'entrée.

w_m : Composantes du vecteur poids synaptique.

b : le biais.

S : Somme pondérée appelée potentiel.

Le biais est la valeur du seuil interne qui doit être dépassée pour l'activation de la sortie du neurone. La sortie finale y du neurone est obtenue en appliquant une fonction d'activation φ(.).

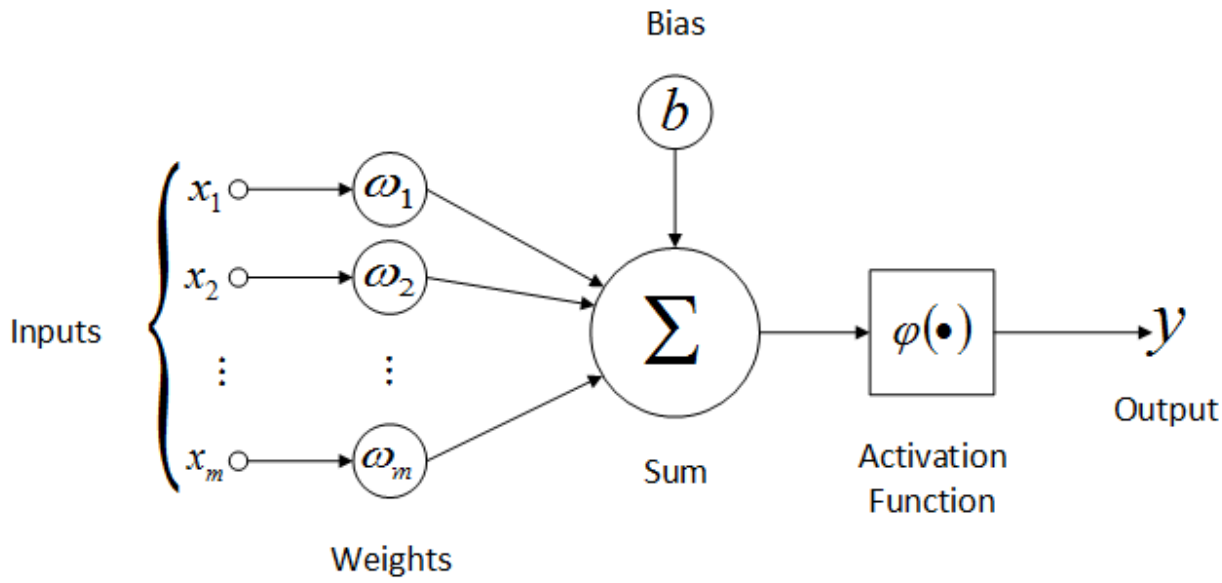

Figure 3.2 (Oliveira et al., 2017) : Modèle d'un neurone artificiel

- Types de fonctions d'activations

Dans le domaine des réseaux de neurones artificiels, une fonction d'activation représente une fonction mathématique qui s'applique à un signal en sortie d'un neurone artificiel. Le terme « fonction d'activation » vient du terme biologique « potentiel d'activation », qui est en fait un seuil de stimulation. C'est-à-dire qu'une fois atteint, la réponse neuronale sera provoquée par le neurone. Il existe plusieurs types de fonctions d'activation, qui limitent l'amplitude du signal de sortie du neurone et imitent l'effet de seuil typique des neurones naturels ; avec tous les types de fonctions d'activation, pour qu'un signal de sortie se propage, la valeur réelle de son entrée doit dépasser un seuil spécifique.

- Fonction d'activation sigmoïde :

La fonction d'activation sigmoïde, également appelée courbe en S est une fonction non linéaire qui associe toute entrée à une valeur de sortie comprise entre 0 et 1. Elle est définie par la fonction suivante :

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Fonction d'activation Tanh (Tangente hyperbolique) :

La fonction d'activation tanh est une fonction non linéaire qui mappe les valeurs d'entrée aux valeurs de sortie entre -1 et 1. Elle est définie par la fonction suivante :

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

- Fonction d'activation ReLU :

La fonction d'activation ReLU (Rectified Linear Unit) est une fonction linéaire par morceaux qui mappe toute entrée à une valeur comprise entre 0 et l'entrée. Elle est définie par la fonction suivante :

$$f(x) = \max(0, \mathrm{x}) = \begin{cases} 0 & si\ \mathrm{x} < 0 \\ x & si\ x \geq 0 \end{cases}$$

- Fonction d'activation Softmax :

La fonction d'activation Softmax est une généralisation de la fonction d'activation sigmoïde utilisée pour les tâches de classification multi-classe. Elle mappe toute entrée à une valeur comprise entre 0 et 1 et est définie par la fonction suivante :

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}\ pour\ i = 1, \ldots, K$$

3.4 Les réseaux de neurones artificiels

Le réseau de neurones artificiels (ANN) est un modèle mathématique inspiré des neurones biologiques, ce qui signifie que plusieurs neurones travaillent ensemble pour recevoir des signaux d'entrée, traiter des informations et déclencher des signaux de sortie.

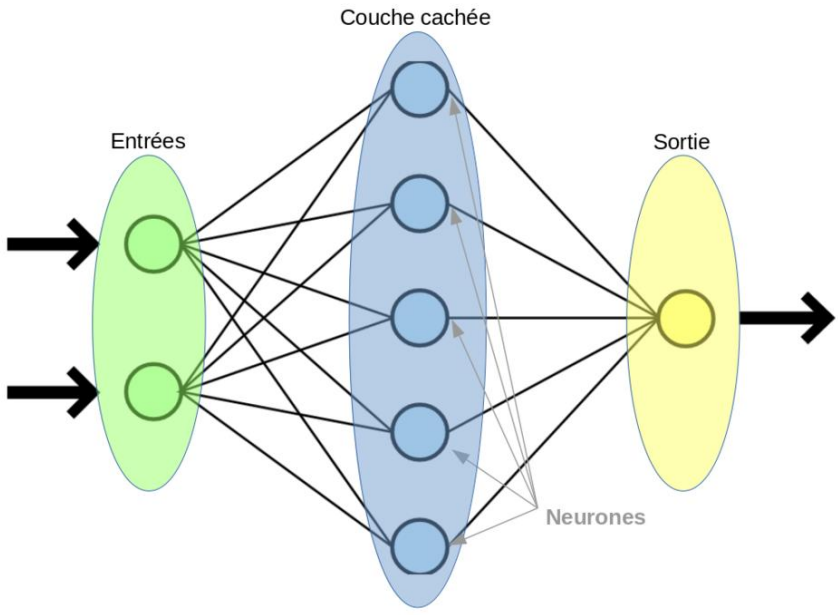

Figure 3.3 (Elkhaloui, 2018) : Architecture d'un réseau de neurones artificiels

Un réseau de neurones est constitué de trois parties fondamentales : la couche d'entrée, qui est un intermédiaire entre le réseau et les données, qui transforme simplement ces dernières sans les modifier, qui est suivie par la couche de traitement (c'est-à-dire en d'autres termes mots, couches cachées ou Hidden Layers).

- Couche d'entrée

La première couche est appelée couche d'entrée. Elle reçoit les données que nous souhaitons utiliser pour l'analyse. Les neurones de cette couche reçoivent les valeurs d'entrée du réseau et les transferts aux neurones de la couche cachée. Chaque neurone reçoit une valeur, il ne fait donc pas de sommation.

- Couche cachée

Les réseaux neuronaux artificiels peuvent avoir un grand nombre de couches cachées. Chaque couche cachée analyse la sortie de la couche précédente, la transforme et la transmet à la couche suivante.

- Couche de sortie

La couche de sortie donne le résultat final de tous les traitements de données effectués par le réseau neuronal artificiel. Il peut avoir un ou plusieurs nœuds. Par exemple, si nous avons un problème de classification binaire (oui/non), la couche de sortie pourra être constituée d'un nœud de sortie qui fournira le résultat sous forme de 1 ou 0. Cependant, si nous avons un problème de classification multi-classes, la couche de sortie peut être constituée de plus d'un nœud de sortie.

3.5 Le réseau de neurones convolutif

Le réseau de neurones convolutif (CNN) est un réseau neuronal multicouche spécifiquement utilisé pour la classification d'images et la reconnaissance des formes (Kertous et al., 2021). Le réseau de neurones convolutif est l'algorithme d'apprentissage profond le plus populaire pour la reconnaissance d'images, la classification d'images, la reconnaissance de formes et d'autres opérations d'extraction de caractéristiques à partir d'une image (Wiskott et al., 2022).

3.5.1 Architecture du CNN

L'architecture CNN de base contient 4 types de couches : la couche convolutive, la couche de mise en commun (pooling), la couche non linéaire et la couche entièrement connectée. Les deux premières couches sont paramétrées et les deux autres ne sont pas paramétrées. Les couches paramétrées sont des couches convolutives et des couches entièrement connectées. Les couches non paramétrées sont des couches non linéaires et des couches de mise en commun. Cependant, cette architecture peut changer en fonction des exigences du problème.

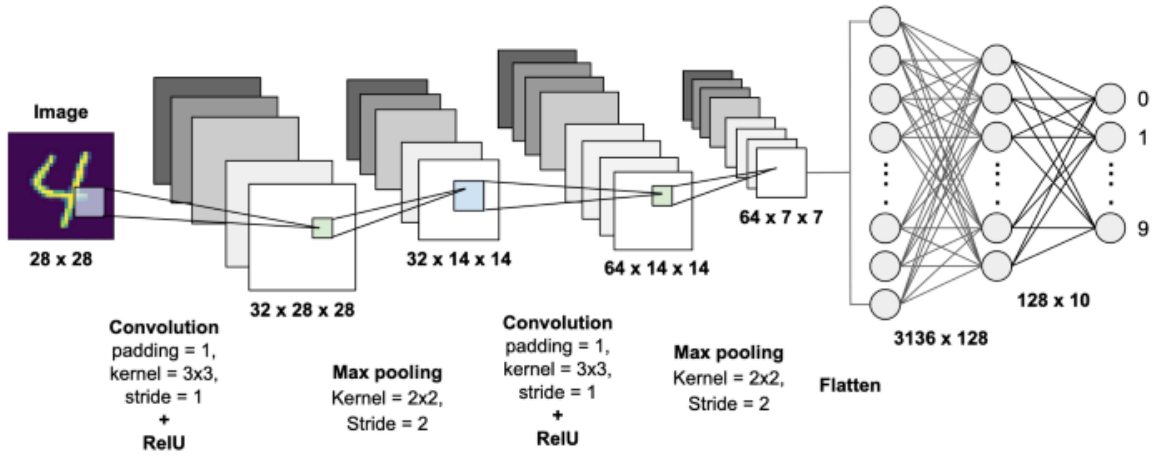

Figure 3.4 (Dasun, 2020) : Architecture du CNN

- Images d'entrée CNN

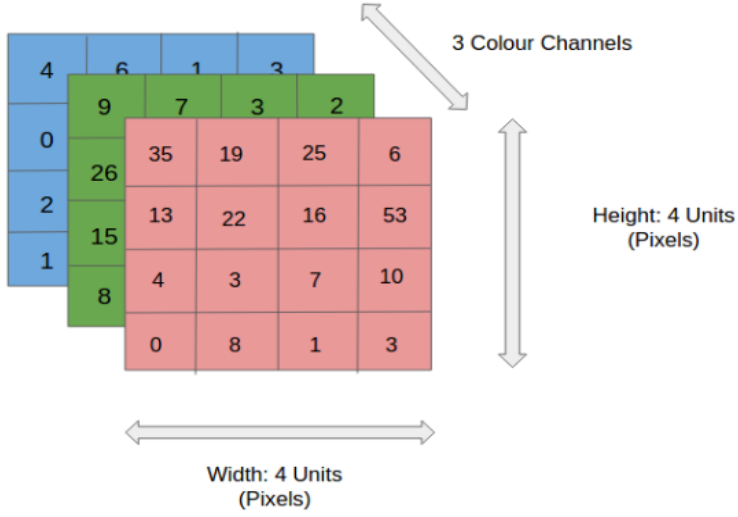

Figure 3.5 (Dasun, 2020) : Image 4 (hauteur) x 4 (largeur) x 3 (canal couleur)

Les images sont généralement divisées en trois canaux bruts : rouge, vert et bleu ; c'est-à-dire qu'elles sont représentées par trois cellules colorées de taille égale. Chaque cellule de la matrice contient une valeur de pixel. Ces pixels constituent les niveaux de couleur. Ces valeurs vont de 0 à 255, où chaque chiffre représente l'intensité d'un pixel. Les lignes représentent la largeur de l'image, les colonnes correspondent à la hauteur de l'image et le nombre de canaux correspond à la profondeur de l'image.

- Couche convolutive

L'architecture d'un réseau neuronal convolutif est basée sur une ou plusieurs couches convolutives connectées sélectivement aux couches précédentes. Une couche convolutive est une combinaison d'opérations de convolution et de fonctions d'activation.

La convolution est une opération de multiplication de somme linéaire par élément entre une partie de l'image d'entrée et une matrice de poids ou un filtre. Pendant l'opération de convolution, une matrice de poids parcourt l'image de telle sorte que tous les pixels soient couverts au moins une fois pour donner la sortie de convolution. Cette sortie de convolution est la carte de caractéristiques.

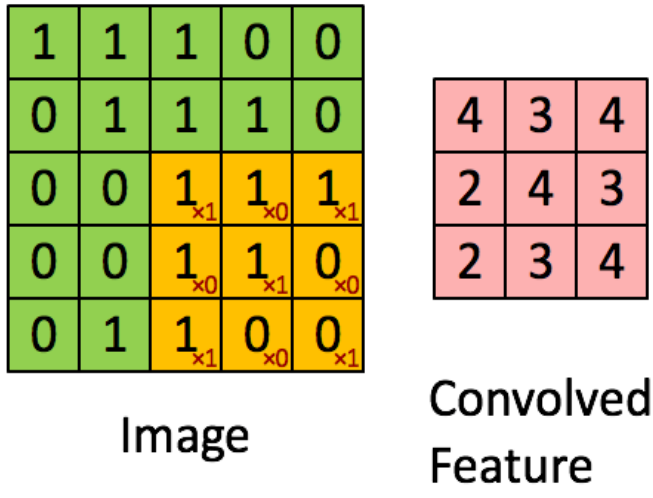

Figure 3.6 (Dasun, 2020) : Opération de convolution

Les matrices de poids ou filtres extraient des informations spécifiques de la matrice d'image d'entrée. Chaque filtre a une fonction spécifique. La taille et le nombre de filtres sont définis par l'utilisateur comme paramètres.

Le filtre se déplace progressivement sur toute l'image d'entrée. Le pas (stride) est la distance entre deux positions de consécutive du filtre. Le pas 1 est un choix courant lors de la convolution.

À mesure que la valeur du pas augmente, la taille de l'image diminue progressivement. Pour surmonter ce problème, une méthode de rembourrage est utilisée sur l'image d'entrée. Le

remplissage est utilisé pour préserver les dimensions spatiales de l'image d'entrée après avoir effectué une opération de convolution sur la carte d'identité.

Dans un CNN, la première couche extrait les caractéristiques les plus générales, tandis qu'à mesure que le réseau s'approfondit, les couches suivantes permettent d'obtenir les caractéristiques plus abstraites. À la sortie de chaque couche convolutive, une fonction d'activation est introduite. Cette dernière est une fonction non linéaire qui permet au réseau neuronal de mieux s'adapter à diverses données.

- Couche de mise en commun (Pooling)

Les couches de mise en commun sont généralement placées entre deux couches convolutives. Ces couches reçoivent les entrées de plusieurs entités pour simplifier les données. Ces données sont sous-échantillonnées. Il s'agit principalement de réduire la puissance de calcul requise pour traiter les données grâce à la réduction de dimensionnalité. En outre, l'autre rôle important de cette couche est d'extraire les caractéristiques dominantes qui sont à la fois invariantes à la rotation et à la position dans l'image d'entrée, contribuant ainsi à maintenir une forte capacité de prédiction.

Il existe deux types courants de regroupement : le regroupement maximal (Max Pooling) et le regroupement moyen (Average Pooling). Max Pooling renvoie la valeur maximale de la partie de l'image couverte par le filtre. Average Pooling renvoie la moyenne de toutes les valeurs de la partie de l'image couverte par le filtre.

Max Pooling fonctionne également comme un suppresseur de bruit. Il supprime complètement les activations bruyantes et effectue également la suppression du bruit avec la réduction de la dimensionnalité. Tandis que, Average Pooling effectue simplement une réduction de dimensionnalité comme un mécanisme de suppression du bruit. Par conséquent, nous pouvons dire que la mise en commun maximale est généralement bien meilleure que la mise en commun moyenne.

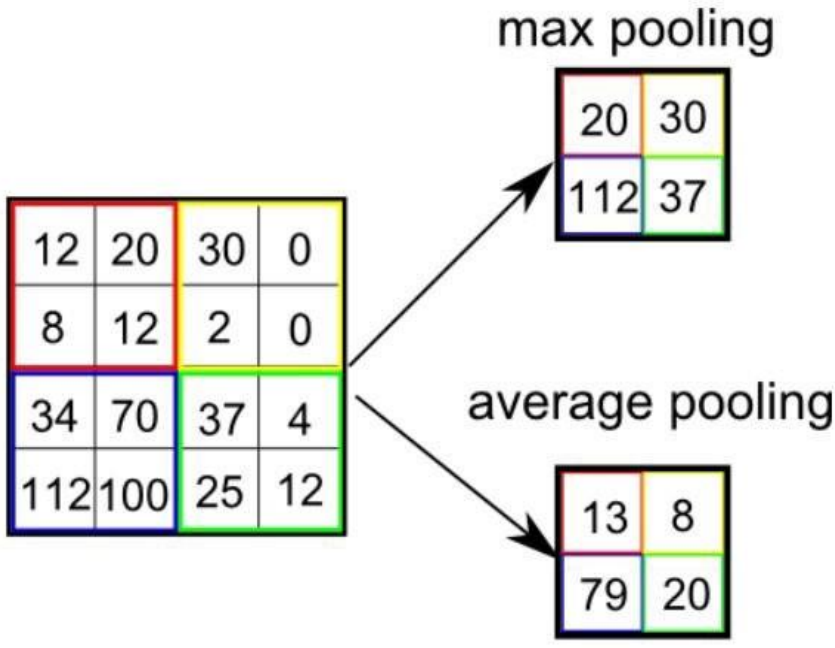

Figure 3.7 (Dasun, 2020) : Différences entre mise en commun maximale et mise en commun moyenne

- Couche entièrement connectée (couche FC)

Après avoir effectué une série d'opérations de regroupement et de convolution, il faut créer des couches entièrement connectées. Chaque neurone est connecté à tous les neurones de la couche précédente.

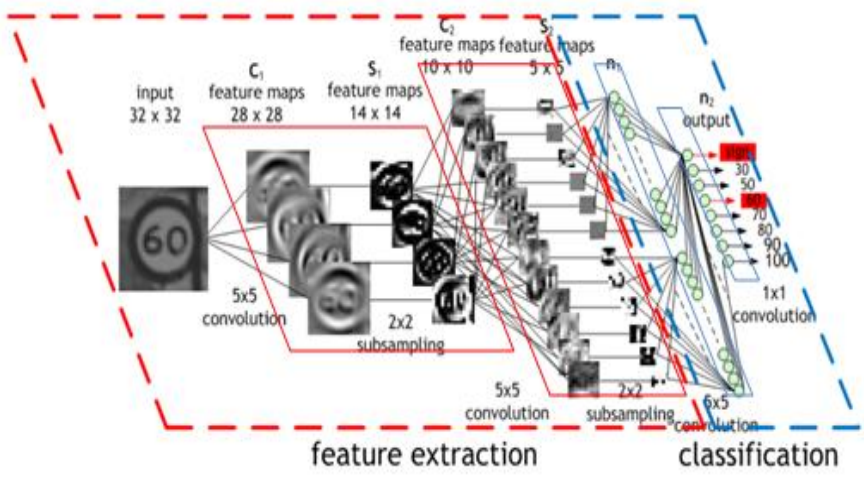

Figure 3.8 (Dasun, 2020) : La répartition entre l'extraction d'entités (couches de convection et de regroupement) et la classification (couches FC)

- Couche de sortie

La couche de sortie d'un réseau neuronal convolutionnel (CNN) joue un rôle essentiel, car c'est la couche finale qui produit la sortie réelle du réseau, généralement sous la forme d'un résultat de classification ou de régression.

Pour les tâches de classification, la couche de sortie utilise généralement une fonction d'activation Softmax, qui convertit l'entrée des couches précédentes en une distribution de probabilité sur les classes prédéfinies. La fonction Softmax garantit que les probabilités de sortie totalisent 1, ce qui les rend directement interprétables comme des probabilités de classe.

Pour les tâches de régression, la couche de sortie peut être constituée d'un ou plusieurs neurones avec une fonction d'activation linéaire, fournissant des valeurs de sortie continues.

3.5.2 Différents types d'architectures de réseaux de neurones convolutifs (CNN)

Il existe plusieurs architectures de réseaux de neurones convolutifs, nous allons en présenter quelques-unes :

- LeNet-5

LeNet-5 est une architecture de réseau neuronal convolutif (CNN) développé en 1998 par Yann LeCun et al. (Lecun et al., 1998). Il s'agit de l'une des architectures CNN les plus anciennes et les plus utilisées. Il a été conçu pour la reconnaissance de chiffres manuscrits, mais a également été appliqué à des tâches de classification d'images. L'architecture est composée de plusieurs couches. Une couche d'entrée prend des images en niveaux de gris 32x32 en entrée. Ensuite, deux couches convolutives extraient les caractéristiques spatiales de l'entrée. Les couches de regroupement moyennes réduisent les dimensions spatiales et améliorent l'invariance de translation. Les couches entièrement connectées interprètent les fonctionnalités extraites et classent l'entrée. En fin La Couche de sortie produit une distribution de probabilité sur 10 classes (chiffres 0 à 9). Bien que LeNet-5 est développé il y a plus de 25 ans, il reste un modèle populaire et influent dans le domaine de l'apprentissage en profondeur, et son architecture est toujours d'actualité et continue d'être utilisée.

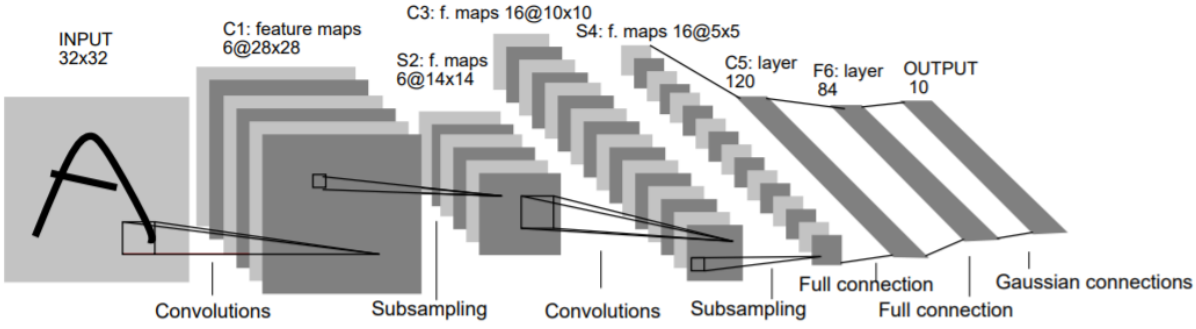

Figure 3.9 (Lecun et al., 1998) : Architecture de LeNet-5

- Couche d'entrée :32×32 images en niveaux de gris.
- C1 : Première couche convolutive avec 6 filtres de taille 5×5. Sorties 6×28×28.
- S2 : La couche de sous-échantillonnage (pooling moyen) réduit les dimensions à 6×14×14.
- C3 : Deuxième couche convolutive avec 16 filtres de taille5×5. Sorties 16×10×10
- S4 : Une autre couche de sous-échantillonnage réduit les dimensions à 16×5×5.
- C5 : La couche convolutionnelle entièrement connectée génère un vecteur de taille 120.
- F6 : La couche entièrement connectée génère un vecteur de taille 84.

- Sortie : Couche entièrement connectée avec 10 neurones de sortie pour la classification.
- Les fonctions d'activations sigmoïdes ou tanh ont été utilisées dans l'implémentation pour la non-linéarité.

- AlexNet

AlexNet est une architecture de réseau neuronal convolutif (CNN) développé par Krizhevsky et al. (Krizhevsky et al., 2012). Le nom du réseau vient du nom de l'auteur principal, Alex. AlexNet a été développé pour classer les images de l'ensemble de données ImageNet, qui contient plus d'un million d'images dans 1000 classes. Il a obtenu des performances nettement supérieures que les architectures CNN précédentes et a remporté le défi de reconnaissance visuelle à grande échelle ImageNet (ILSVRC) en 2012 avec un résultat avec 15,3 % d'erreurs, devançant le second au classement de 10 %. AlexNet présente une séquence de couches convolutives, de mise en commun et entièrement connectées. Il a introduit plusieurs innovations qui sont devenues la norme dans les CNN modernes, notamment l'utilisation d'unités linéaires rectifiées (ReLU) comme fonction d'activation et l'utilisation de l'abandon (dropout) pour éviter le surajustement. À une échelle plus large, AlexNet est une contribution significative au domaine de l'apprentissage en profondeur. Son succès sur l'ensemble de données ImageNet a contribué à faire des CNNs un outil puissant pour les tâches de classification d'images.

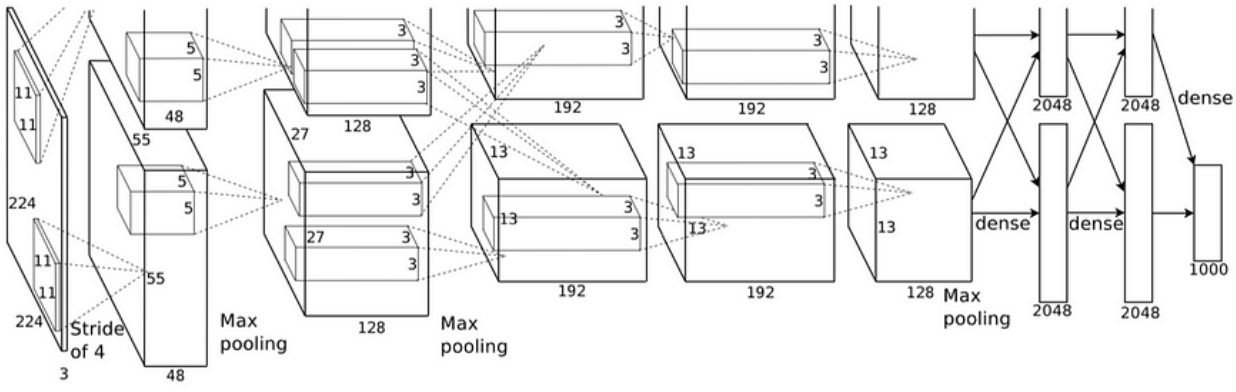

Figure 3.10 (Krizhevsky et al., 2012) : Architecture d'AlexNet

Le modèle prend en entré des images de taille 224×224×3 (images RVB). Il contient cinq couches convolutives pour l'extraction de fonctionnalités. Trois couches entièrement connectées sont utilisées pour la classification et une couche de sortie Softmax.

La fonction d'activation de l'unité linéaire rectifiée (ReLU) est utilisée pour d'atténuer le problème du gradient évanescent et d'accélérer la formation.

La régularisation des pertes a été appliquée aux couches entièrement connectées pour réduire le surajustement.

Des couches de regroupement maximal superposées ont été utilisées au lieu de couches non superposées pour mieux capturer les hiérarchies spatiales.

L'ensemble de données a été augmenté à l'aide de techniques telles que le recadrage aléatoire, le retournement et le changement de couleur pour améliorer la généralisation.

- GoogLeNet (Inception V1)

GoogLeNet (également connu sous le nom d'Inception V1) est une architecture de réseau neuronal convolutif (CNN) développée par Christian Szegedy et al. (Szegedy et al., 2015) chez Google en 2014. Cette architecture a aussi été développé pour classer les images de l'ensemble de données ImageNet, qui rappelons-le, contient plus d'un million d'images dans 1000 classes. Il a obtenu des performances nettement meilleures que les architectures CNN précédentes et a remporté le ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2014. Les résultats obtenus diminuent considérablement la marge d'erreur comparée aux résultats obtenus dans les précédentes versions du concours ainsi qu'avec son second concurrent. Ceci en utilisant des sous-réseaux nommés « module Inception » qui sous-entend le fait d'aller plus profondément dans le réseau. GoogLeNet est connu pour son utilisation de modules de démarrage, qui sont conçus pour réduire le nombre de paramètres et le coût de calcul du réseau. Il a également introduit l'utilisation de classificateurs auxiliaires, qui sont des classificateurs supplémentaires formés avec le classificateur principal afin d'améliorer les performances.

GoogLeNet a relevé les défis des architectures CNN précédentes en introduisant le concept de module Inceptron. Il est un type de bloc de construction conçu pour améliorer l'efficacité et la précision des calculs. Le module combine des couches convolutives de différentes tailles (1×1, 3×3, 5×5) et une couche de mise en commun maximale en parallèle, permettant au réseau de capturer des fonctionnalités multi-échelles.

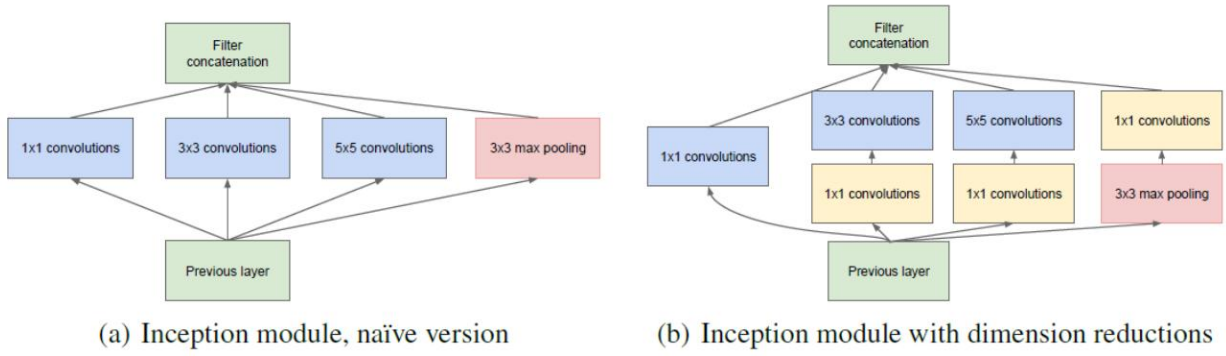

Figure 3.11(Szegedy et al., 2015) : Module Inception

Comme nous pouvons le voir sur l'image ci-dessus la version naïve du module, des convolutions directes 3×3, 5×5 sont utilisées, ce qui est trop coûteux à calculer. Ainsi, dans la deuxième image du module, une convolution 1×1 est utilisé pour réduire la dimension et ce qui conduit à un calcul moins coûteux.

Dans l'architecture précédente comme AlexNet, les couches entièrement connectées sont utilisées à la fin du réseau. Ces couches entièrement connectées contiennent la majorité des paramètres de nombreuses architectures, ce qui entraîne une augmentation du coût de calcul.

Dans GoogLeNet, il existe une méthode appelée global average pooling qui est utilisée à la fin du réseau. Cette couche prend une carte de caractéristiques de 7×7 et la moyenne à 1×1. Cela réduit le surapprentissage et diminue le nombre de paramètres.

L'architecture d'Inception utilise des branches de classificateur intermédiaires au milieu de l'architecture, ces branches sont utilisées uniquement pendant la formation. Elles sont composées d'une couche de mise en commun moyenne 5×5 avec un pas de 3, de convolutions 1×1 avec 128 filtres, de deux couches entièrement connectées de 1024 sorties et 1000 sorties et d'une couche de classification Softmax. La perte générée par ces couches s'ajoute à la perte totale avec un poids de 0,3. Ces couches aident à lutter contre le problème de disparition du gradient et assurent également la régularisation.

L'architecture globale comporte 22 couches. L'architecture a été conçue pour garder à l'esprit l'efficacité informatique. L'idée derrière l'architecture est que l'architecture peut être

exécutée sur des appareils individuels même avec de faibles ressources de calcul. L'architecture contient également deux couches de classificateur auxiliaires connectées à la sortie des couches Inception (4a) et Inception (4d).

type	patch size/ stride	output size	depth	#1 x 1	#3 x 3 reduce	#3 x 3	#5 x 5 reduce	#5 x 5	pool proj	params	ops
convolution	7 x 7 / 2	112 x 112 x 64	1							2.7K	34M
max pool	3 x 3 / 2	56 x 56 x 64	0								
convolution	3 x 3 / 1	56 x 56 x 192	2		64	192				112K	360M
max pool	3 x 3 / 2	28 x 28 x 192	0								
inceptron (3a)		28 x 28 x 256	2	64	96	128	16	32	32	159K	128M
inceptron (3b)		28 x 28 x 480	2	128	128	192	32	96	64	380K	304M
max pool	3 x 3 / 2	14 x 14 x 480	0								
inceptron (4a)		14 x 14 x 512	2	192	96	208	16	48	64	364K	73M
inceptron (4b)		14 x 14 x 512	2	160	112	224	24	64	64	437K	88M
inceptron (4c)		14 x 14 x 512	2	128	128	256	24	64	64	463K	100M
inceptron (4d)		14 x 14 x 528	2	112	144	288	32	64	64	580K	119M
inceptron (4e)		14 x 14 x 832	2	256	160	320	32	128	128	840K	170M
max pool	3 x 3 / 2	7 x 7 x 832	0								
inceptron (5a)		7 x 7 x 832	2	256	160	320	32	128	128	1072K	54M
inceptron (5b)		7 x 7 x 1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7 x 7 / 1	1 x 1 x 1024	0								
dropout(40%)		1 x 1 x 1024	0								
linear		1 x 1 x 1000	1							1000K	1M
softmax		1 x 1 x 1000	0								

Figure 3.12 (Szegedy et al., 2015) : Architecture de GoogleNet

Cette architecture prend des images de taille 224 x 224 avec des canaux de couleur RVB. Toutes les convolutions à l'intérieur de cette architecture utilisent des unités linéaires rectifiées (ReLU) comme fonctions d'activation.

- VGGNet (VGG-16)

VGGNet est une architecture de réseau neuronal convolutif (CNN) qui a été développée par Karen Simonyan et Andrew Zisserman(Simonyan & Zisserman, 2014) en 2014. Son nom vient du Visual Geometry Group de l'Université d'Oxford, où l'architecture a été développée. VGGNet a aussi été développé pour classer les images de l'ensemble de données ImageNet. Il a atteint des performances de pointe sur l'ensemble de données ImageNet et a été finaliste du ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2014 en obtenant la seconde place. Connu pour son utilisation d'une architecture profonde et étroite, avec un grand nombre de couches convolutives et un nombre relativement faible de filtres dans ces couches. Il a également introduit l'utilisation de petits filtres convolutifs de 3x3, qui sont devenus un standard dans les CNNs modernes.

L'architecture se caractérise par sa profondeur, avec des configurations allant de 11 à 19 couches. Les versions les plus populaires sont VGG16 (16 couches) et VGG19 (19 couches). Cette profondeur permet au réseau d'apprendre des fonctionnalités complexes à partir d'images.

Au lieu d'utiliser des filtres plus grands comme 7×7 ou 11×11, VGGNet utilise exclusivement des petits filtres de taille 3×3 avec foulée 1. Cela réduit le nombre de paramètres tout en conservant la taille du champ récepteur des filtres plus grands (obtenu en empilant plusieurs 3×3 couches).

Les fonctions d'activation de l'unité linéaire rectifiée (ReLU) sont appliquées après chaque opération de convolution, introduisant ainsi une non-linéarité dans le modèle.

Les Couches de mise en commun maximal avec un 2×2 la fenêtre et la foulée 2 sont utilisées pour le sous-échantillonnage spatial.

Après les couches convolutives et de pooling, le réseau comprend trois couches entièrement connectées, suivies d'une couche softmax pour la classification.

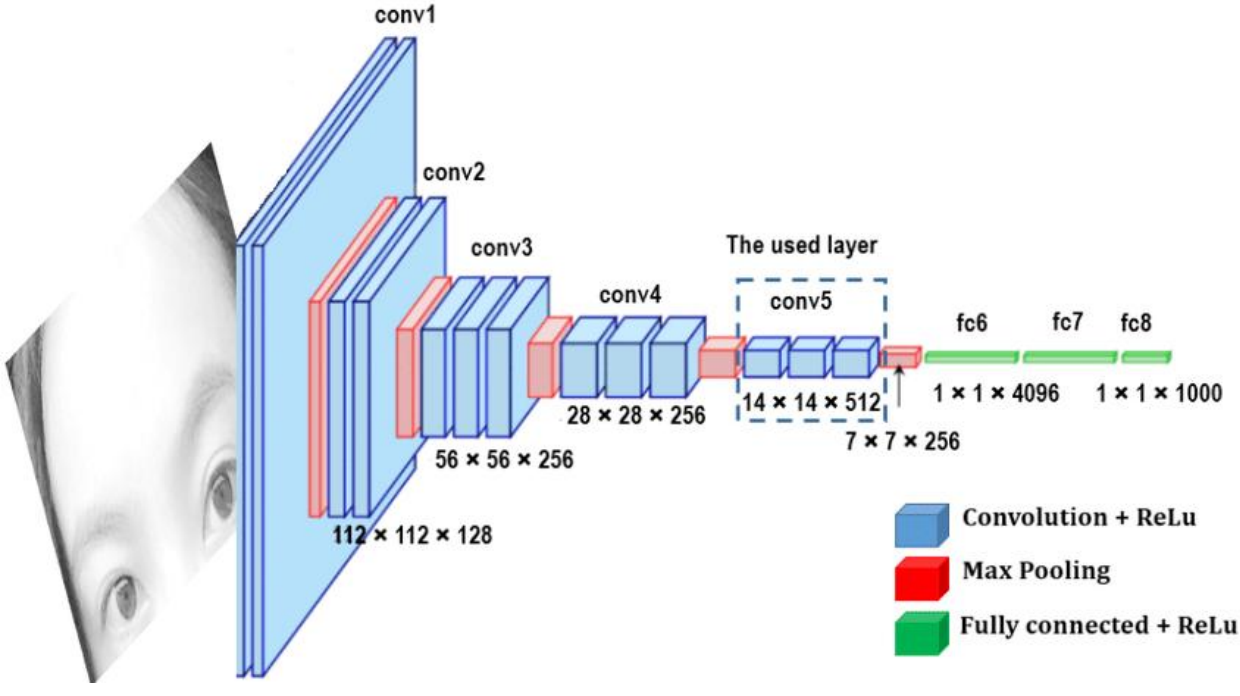

Figure 3.13 (Hariri, 2022) : Architecture de VGGNet

- ResNet

ResNet (Residual Network) est une architecture de réseau neuronal convolutif (CNN) développé par Kaiming He et al.(He et al., 2015), en 2015. Il s'agit aussi à l'instar de LeCun, d'un travail de pionnier dans le domaine de l'apprentissage en profondeur et est largement considéré comme une étape majeure dans le développement des CNN pour la classification des images. Il est développé pour classer les images du jeu de données ImageNet. Il a obtenu des performances nettement meilleures que les architectures CNN précédentes et a remporté le ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2015 avec un taux d'erreurs de 3,57 % sur l'ensemble de test ImageNet.

ResNet est connu pour son utilisation de connexions résiduelles, qui sont des connexions raccourcies qui sautent une ou plusieurs couches et permettent au réseau d'apprendre le mappage résiduel entre l'entrée et la sortie d'une couche. Cette connexion permet d'atténuer le problème de gradient de fuite et rend possible la formation de réseaux profonds.

Les connexions de saut ajoutent l'entrée d'un bloc directement à sa sortie. Cela permet de préserver les informations et les gradients pendant la rétropropagation. En terme mathématique cela signifie :

$$y = \mathrm{F}(x) + x$$

Où F(x) est la transformation appliquée par les couches convolutionnelles, x est l'entrée du bloc et y le résultat final de la couche.

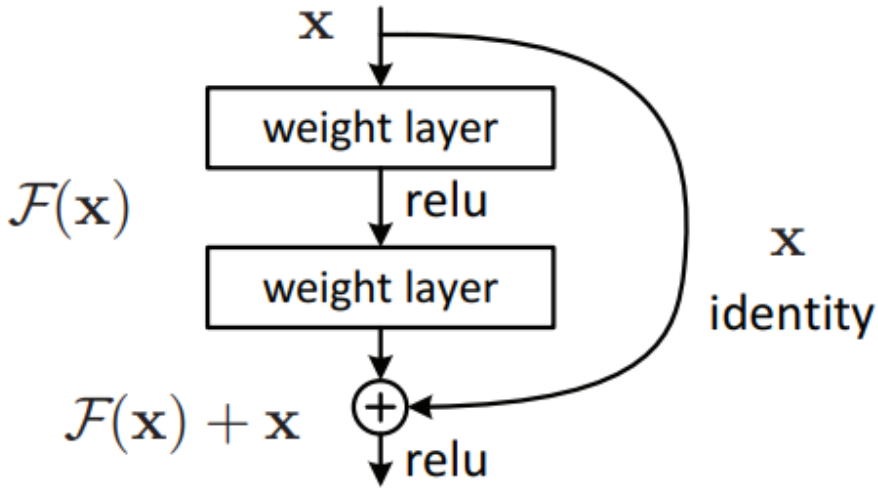

Figure 3.14(Choi et al., 2018) :Un bloc résiduel

L'avantage d'ajouter ce type de connexion de saut est que si une couche nuit aux performances de l'architecture, elle sera ignorée par la régularisation. Ainsi, cela permet de former un réseau neuronal très profond sans les problèmes causés par la disparition ou explosion du gradient.

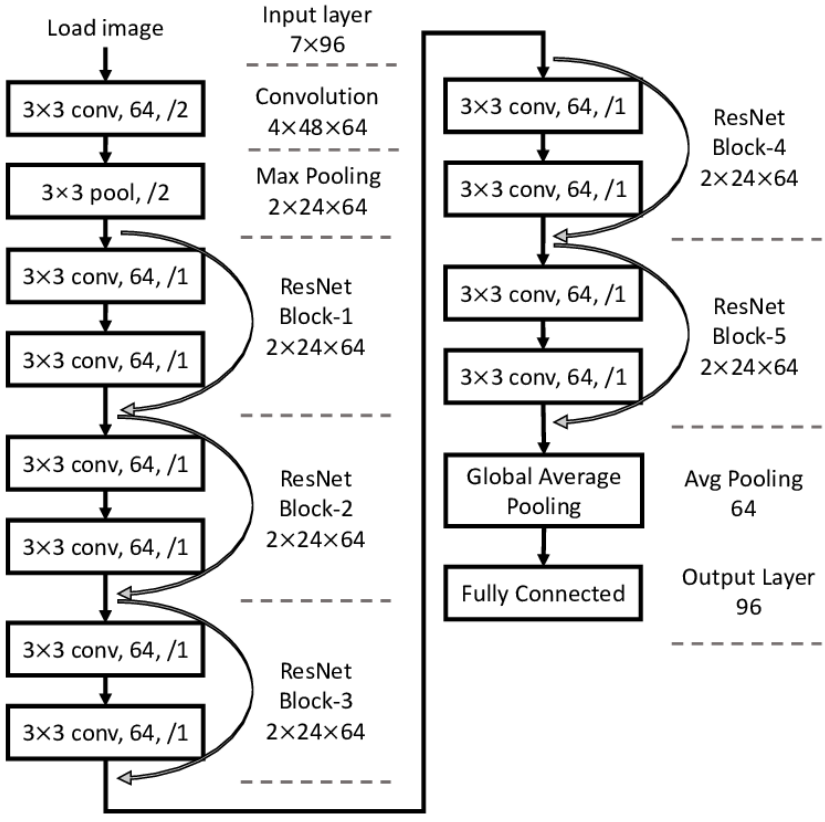

Figure 3.15 (Choi et al., 2018) : Architecture de ResNet

- MobileNet

Les MobileNets sont des CNN qui peuvent être installés sur un appareil mobile pour classer des images ou détecter des objets avec une faible latence. Les MobileNets ont été développés par Andrew G Howard et al. (Howard et al., 2017). Il s'agit généralement de très petites architectures CNN, ce qui les rend faciles à exécuter en temps réel à l'aide d'appareils intégrés comme les smartphones et les drones. L'architecture est également flexible, elle a donc été testée sur des CNN avec 100 à 300 couches et fonctionne toujours mieux que d'autres architectures comme VGGNet. Parmi les exemples concrets d'architecture CNN de MobileNets, on peut citer les CNN intégrés aux téléphones Android pour exécuter l'API mobile Vision de Google, qui peut identifier automatiquement les étiquettes des objets populaires dans les images.

MobileNet utilise la convolution séparable en profondeur qui divise une convolution standard en deux couches, la convolution en profondeur et la convolution ponctuelle.

Fondamentalement, la première couche est utilisée pour filtrer les canaux d'entrée et la deuxième couche est utilisée pour les combiner afin de créer une nouvelle fonctionnalité.

Les convolutions en profondeur sont utilisées pour appliquer un seul filtre à chaque canal d'entrée. Cela diffère d'une convolution standard dans laquelle les filtres sont appliqués à tous les canaux d'entrée.

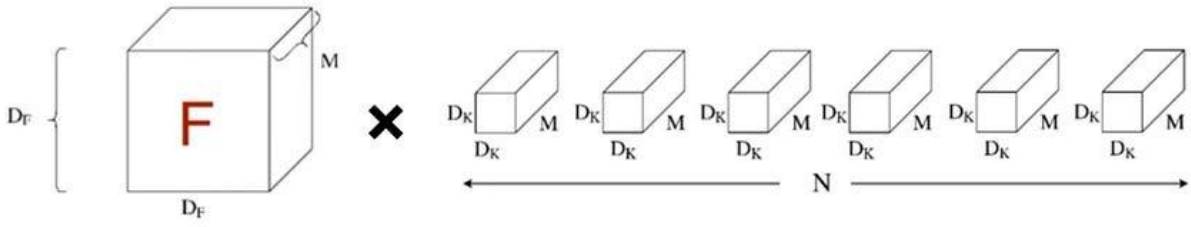

Figure 3.16 (PA, 2020) : Convolution standard

À partir de l'image ci-dessus, le coût de calcul peut être calculé comme suit :

$$D_K . D_K . M . N . D_F . D_F$$

Où DF correspond aux dimensions spéciales de la carte des caractéristiques d'entrée et DK à la taille du noyau de convolution. Ici, M et N correspondent respectivement au nombre de canaux d'entrée et de sortie.

Pour une convolution standard, le coût de calcul dépend de manière multiplicative du nombre de canaux d'entrée et de sortie et des dimensions spatiales de la carte des caractéristiques d'entrée et du noyau de convolution.

En cas de convolution en profondeur, comme le montre l'image ci-dessous, contient une carte de caractéristiques d'entrée de dimension DF*DF et un nombre M de noyaux de taille de canal 1.

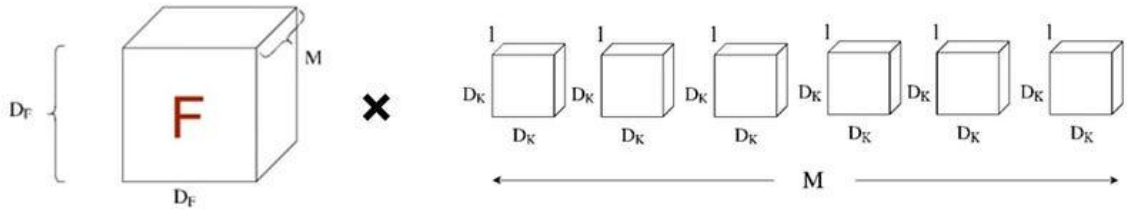

Figure 3.17 (PA, 2020) : Convolution en profondeur

Comme le montre l'image ci-dessus, nous pouvons clairement voir que le coût de calcul total peut être calculé comme suit :

$$D_K . D_K . M . D_F . D_F$$

Cependant, cette méthode n'est utilisée que pour filtrer le canal d'entrée.

Convolution ponctuelle

Étant donné que la convolution en profondeur n'est utilisée que pour filtrer le canal d'entrée, elle ne les combine pas pour produire de nouvelles caractéristiques. Une couche supplémentaire appelée couche de convolution ponctuelle est donc créée, qui calcule une combinaison linéaire de la sortie de la convolution en profondeur à l'aide d'une convolution 1 × 1.

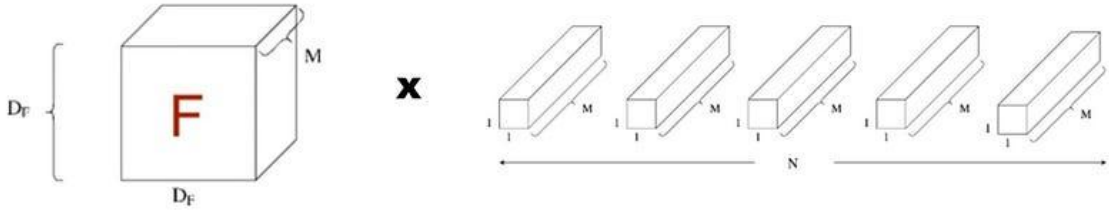

Figure 3.18 (PA, 2020) : Convolution ponctuelle

Comme le montre l'image, calculons à nouveau le coût de calcul :

$$M.N.D_F.D_F$$

Ainsi, le coût de calcul total des convolutions séparables en profondeur peut être calculé comme suit :

$$D_K.D_K.M.D_F.D_F + M.N.D_F.D_F$$

En le comparant au coût de calcul de la convolution standard, nous obtenons une réduction du calcul, qui peut s'exprimer comme suit :

$$\frac{D_K.D_K.M.D_F.D_F + M.N.D_F.D_F}{D_K.D_K.M.N.D_F.D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 3.19 (Howard et al., 2017) : Architecture de MobileNet

3.6 Transformer

Un Transformer est une architecture d'apprentissage profond développé par Google et basé sur le mécanisme d'attention multi têtes, proposé dans un article de 2017 « Attention Is All You Need » (Vaswani et al., 2017). Depuis lors, le Transformer est devenu un choix d'architecture populaire pour une variété de tâches. Il est capable de capturer les dépendances à longue portée des données, ce qui en fait un outil puissant non seulement pour le traitement du langage naturel (NLP), mais également pour la vision par ordinateur, l'audio et le repliement des protéines.

3.6.1 Architecture

L'architecture globale du Transformer est représentée dans la Figure 3.7. Elle utilise des couches d'auto-attention empilées et entièrement connectées par points pour l'encodeur et le décodeur, illustrées respectivement dans la partie gauche et droite de la figure 3.7.

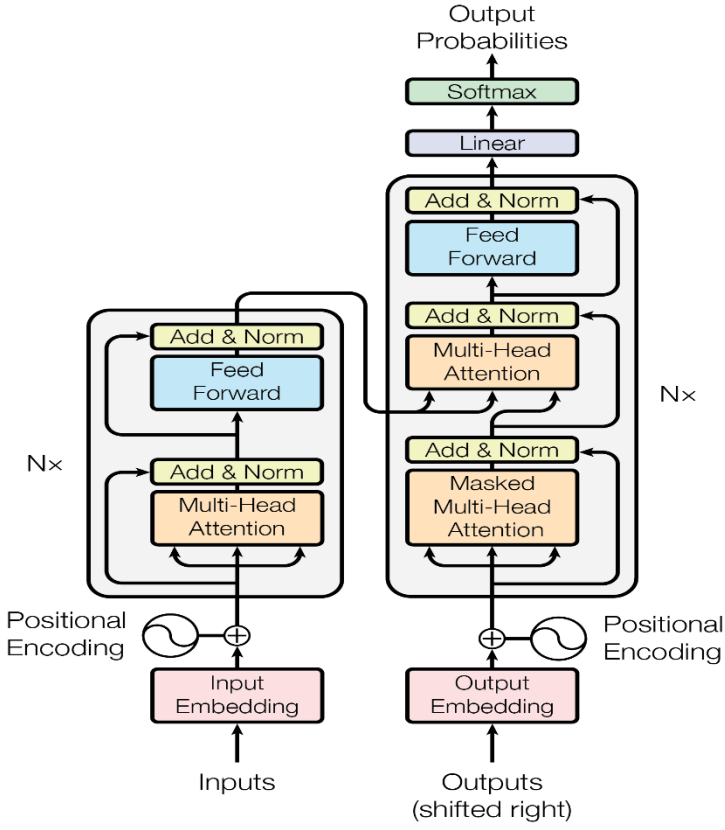

Figure 3.20 (Vaswani et al., 2017) : Architecture de transformer

3.6.1.1 Piles d'encodeurs et de décodeurs

- Encodeur :

L'encodeur est composé d'une pile de N couches identiques. Chaque couche comporte deux sous-couches. Le premier est un mécanisme d'auto-attention multi têtes, et le second est un simple réseau de rétroaction (feed-forward) entièrement connecté en termes de position. Une

connexion résiduelle autour de chacune des deux sous-couches est employée, suivie d'une normalisation des couches. Pour faciliter ces connexions résiduelles, toutes les sous-couches du modèle, ainsi que les couches d'intégration, produisent des sorties de dimension d = 512.

- Décodeur :

Le décodeur est constitué d'une pile de N couches identiques. En plus des deux sous-couches dans chaque couche de l'encodeur, le décodeur insère une troisième sous-couche permettant d'effectuer une attention multi têtes sur la sortie de la pile de codeurs. Semblable à l'encodeur, il utilise des connexions résiduelles autour de chacune des sous-couches, suivies d'une normalisation des couches. On note une modification de la sous-couche d'auto-attention dans la pile de décodeurs pour empêcher les items des positions courante s d'influencer les items des positions suivantes. Ce masquage, combiné au fait que les plongements de sortie sont décalés d'une position, garantit que les prédictions de position i ne peuvent dépendre que des sorties connues à des positions inférieures de i.

3.6.1.2 Attention

L'attention peut être décrite comme une fonction mappant une requête et un ensemble de paires clé-valeur à une sortie, où la requête, les clés, les valeurs et la sortie sont tous des vecteurs. La sortie est obtenue sous la forme d'une somme pondérée des valeurs, où une fonction de compatibilité de la requête avec la clé correspondante est utilisée pour calculer le poids à attribuer à chaque valeur.

- « Attention aux produits scalaires à l'échelle »

Une attention particulière est appelée attention aux produits scalaires à l'échelle (Scaled Dot-Product Attention) (Figure 3.8). L'entrée est constituée de requêtes et de clés de dimension d_k et les valeurs de dimension d_v.

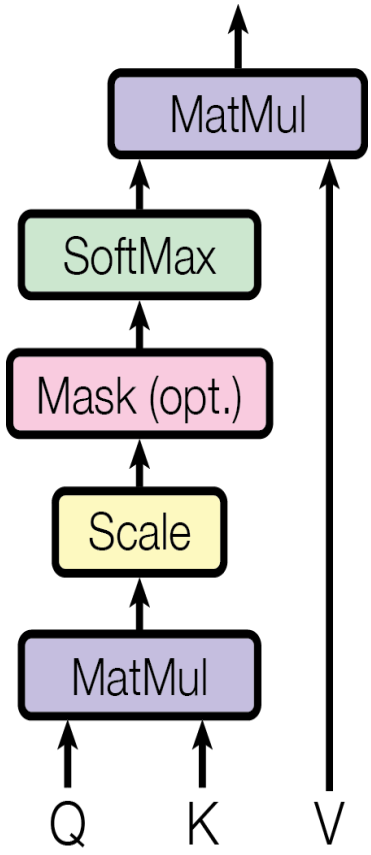

Figure 3.21 (Vaswani et al., 2017) : Attention au produit scalaire mis à l'échelle.

La fonction d'attention sur un ensemble de requêtes simultanément regroupées dans une matrice Q est les produits scalaires de la requête avec toutes les clés. Le résultat est ensuite divisé par $\sqrt{d_k}$, et on y applique une fonction Softmax pour obtenir les poids associés aux valeurs. Les clés et les valeurs sont également regroupées dans des matrices Ket V.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Les deux fonctions d'attention les plus couramment utilisées sont l'attention additive, et l'attention sur le produit scalaire (multiplicative). L'attention sur le produit scalaire est identique à l'algorithme, à l'exception du facteur d'échelle de $1/\sqrt{d_k}$. L'attention additive calcule la fonction de compatibilité à l'aide d'un réseau à action directe avec une seule couche cachée. Bien que les deux soient similaires en termes de complexité théorique, l'attention au produit scalaire est

beaucoup plus rapide et plus efficace en termes d'espace, car elle peut être mise en œuvre à l'aide d'un code de multiplication matricielle hautement optimisé.

Alors que pour de petites valeurs de d_k, les deux mécanismes fonctionnent de manière similaire, l'attention additive surpasse l'attention du produit scalaire sans mise à l'échelle pour des valeurs plus grandes de d_k. Pour de grandes valeurs de d_k, les produits scalaires augmentent en ampleur, poussant la fonction Softmax dans des régions où elle a des gradients extrêmement faibles. Pour contrecarrer cet effet, il faut mettre à l'échelle les produits scalaires en $1/\sqrt{d_k}$.

- Attention multi têtes

C'est la projection linéaire des requêtes, clés et valeurs h fois avec différentes projections linéaires apprises respectivement pour d_q, d_k, et d_v, dimensions. Sur chacune de ces versions projetées des requêtes, clés et valeurs, est ensuite effectuée la fonction d'attention en parallèle, ce qui donne une valeur de sortie dimensionnelle d_v. Celles-ci sont concaténées et projetées à nouveau pour donner les valeurs finales, comme indiquées dans la figure 3.9.

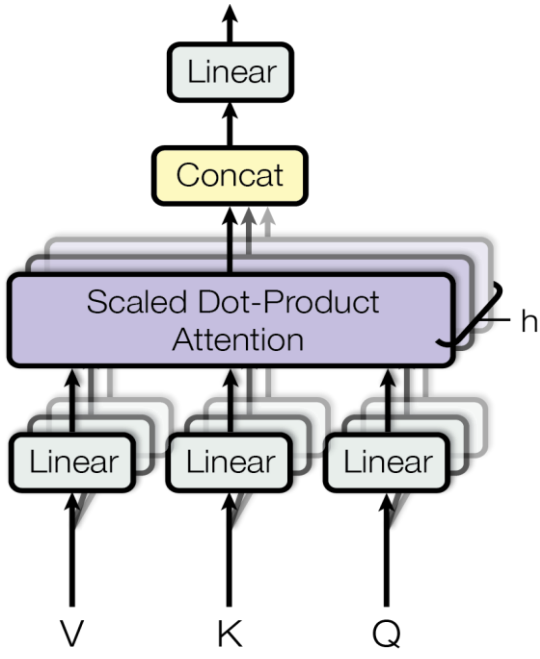

Figure 3.22 (Vaswani et al., 2017) : L'attention multi têtes se compose de plusieurs couches d'attention fonctionnant en parallèle.

L'attention multi têtes permet au modèle de s'occuper conjointement des informations provenant de différents sous-espaces de représentation à différentes positions. Avec une seule tête d'attention, la moyenne inhibe cela.

$$Multitête(Q, K, V) = Concat(\, tête_1, \ldots, tête_h)W^0$$

$$\text{où } tête_i = Attention(\, QW_i^Q, KW_i^K, VW_i^V)$$

Où les projections sont des matrices de paramètres :

$$W_i^Q \in \mathbb{R}^{d \times d_k}, W_i^K \in \mathbb{R}^{d \times d_k}, W_i^V \in \mathbb{R}^{d \times d_v} \text{ et } W^0 \in \mathbb{R}^{hd_v \times d}$$

- Applications de l'attention dans le modèle

Le Transformer utilise l'attention multi têtes de trois manières différentes :

- Dans les couches attention encodeur-décodeur, les requêtes proviennent de la couche de décodeur précédente, et les clés et valeurs mémoires proviennent de la sortie de l'encodeur. Cela permet à chaque position du décodeur de s'appliquer à toutes les positions de la séquence d'entrée. Cela imite les mécanismes d'attention typiques de l'encodeur-décodeur dans les modèles séquence à séquence.
- L'encodeur contient des couches d'auto-attention. Dans une couche d'auto-attention, toutes les clés, valeurs et requêtes proviennent du même endroit, dans ce cas, la sortie de la couche précédente dans l'encodeur. Chaque position de l'encodeur peut s'occuper de toutes les positions de la couche précédente de l'encodeur.
- De même, les couches d'auto-attention dans le décodeur permettent à chaque position dans le décodeur de s'occuper de toutes les positions dans le décodeur jusqu'à cette position incluse.

3.6.1.3 Réseaux Feed-Forward par position

En plus des sous-couches d'attention, chacune des couches de l'encodeur et décodeur contient un réseau de rétroaction entièrement connecté, qui est appliqué à chaque position

séparément et de manière identique. Il s'agit de deux transformations linéaires avec une activation ReLU entre les deux.

$$FFN(x) = max(\,0, xW_1 + b_1)W_2 + b_2$$

Bien que les transformations linéaires soient les mêmes dans différentes positions, elles utilisent des paramètres différents d'une couche à l'autre. Une autre façon de décrire cela est de considérer deux convolutions avec une taille de noyau de 1. La dimensionnalité de l'entrée et de la sortie est $d = 512$, et la couche interne a une dimensionnalité $d_{ff} = 2048$.

3.6.1.4 Intégrations

L'intégration dans les Transformers se réfère principalement à deux processus essentiels : l'encodage des entrées (souvent appelé « embedding ») et l'encodage positionnel.

- Encodage des entrées

Dans cette section, l'encodeur commence par convertir les jetons d'entrée (mots ou sous-mots) en vecteurs à l'aide de couches d'intégration. Ces intégrations capturent la signification sémantique des jetons et les convertissent en vecteurs numériques de taille 512 (taille fixe). Ces vecteurs d'encodage sont appris lors de l'entrainement du modèle et permettent au modèle de comprendre le contexte d'un mot dans une phrase.

- Encodage positionnel

Puisque l'architecture du transformer ne contient ni récurrence ni convolution, il faut injecter des informations sur la position relative ou absolue des jetons dans la séquence pour que le modèle utilise l'ordre de la séquence. À cette fin, il est nécessaire d'ajouter des encodages positionnels aux intégrations d'entrée au bas des piles d'encodeurs et de décodeurs. Les encodages positionnels ont la même dimension d comme les plongements, afin que les deux puissent être additionnés.

Il existe de nombreux choix d'encodages positionnels :

$$PE_{(pos,\ 2i)} = \sin(\text{pos}/10000^{2i\ /\ d_{modèle}})$$

$$PE_{(pos,\ 2i+1)} = \cos(\text{pos}/10000^{2i\ /\ d_{modèle}})$$

Où pos est la position et i est la dimension. Autrement dit, chaque dimension du codage de position correspond à une sinusoïde. Les longueurs d'onde forment une progression géométrique de 2π à $10000 \cdot 2\pi$.

3.6.1.5 Sortie Softmax

Le produit scalaire entre les vecteurs de requête (Query) et de clé (Key) est calculé pour chaque paire de jetons. Cela produit des scores non normalisés, qui sont ensuite normalisés à l'aide de la fonction Softmax pour produire des poids d'attention. La fonction Softmax est essentielle dans le mécanisme d'attention, car elle permet de convertit les produits scalaires calculées en score de probabilité. Ces probabilités déterminent à quel point chaque mot de la séquence d'entrée devrait influencer la représentation finale d'un mot donné. La fonction Softmax transforme ces scores en une distribution de probabilités (somme à 1), assurant que chaque score soit compris entre 0 et 1.

3.6.2 Transformer et Vision par ordinateur

L'architecture de transformer telle que présentée ci-dessus a eu un impact énorme dans le domaine du NLP. Mais ses applications dans le domaine de la vision par ordinateur étaient limitées. En 2021, une équipe de recherche de Google a présenté l'article « une image vaut 16x16 mots : transformateurs pour la reconnaissance d'images à l'échelle (2021) » (Dosovitskiy et al., 2020), qui appliquait l'architecture de l'encodeur Transformer à la tâche de reconnaissance (classification) d'images. L'idée de l'article est de créer un Vision Transformer en utilisant l'architecture d'encodeur Transformer, avec le moins de modifications possible, et de l'appliquer aux tâches de classification d'images. Un Vision Transformer (ViT) est un transformer conçu pour la vision par ordinateur. Un ViT décompose une image d'entrée en une série de patchs (plutôt que de diviser le texte en jetons), sérialise chaque patch en un vecteur et le mappe à une dimension

plus petite avec une seule multiplication matricielle. Ces intégrations vectorielles sont ensuite traitées par un codeur de transformateur comme s'il s'agissait d'intégrations de jetons.

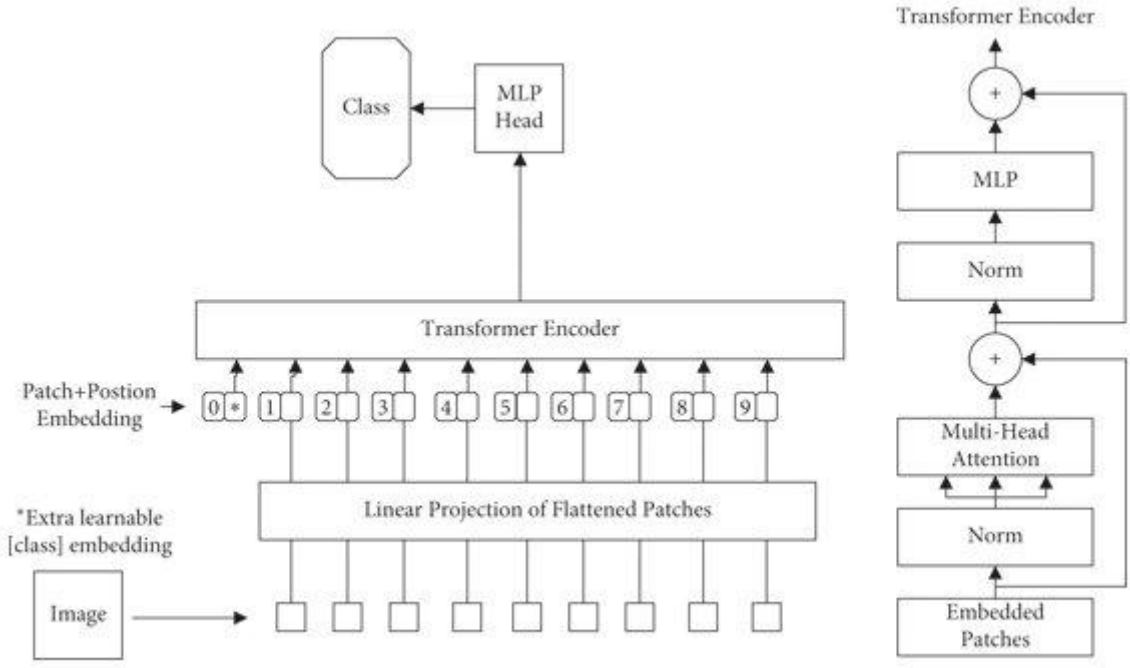

Figure 3.23 (Liang et al., 2021) : Architecture du réseau Vision Transformer

3.7 Réseau Siamois

En informatique, la similitude est un aspect important pour l'identification d'un groupe d'objets, de valeurs, de données, etc. Chaque fois que deux vecteurs d'éléments sont comparés, de nombreuses approches de similarité différentes peuvent être utilisées en fonction du but de la comparaison (distance euclidienne, coefficient de corrélation Pearson, coefficient de corrélation de rang de Spearman, et autres). Cependant, si la comparaison doit se faire avec un grand nombre de données dans ce cas, il serait nécessaire d'utiliser les réseaux siamois pour de meilleurs résultats. Un réseau neuronal siamois est une classe d'architectures de réseaux neuronaux qui contiennent deux ou plusieurs sous-réseaux identiques. L'idée du réseau siamois est d'exécuter deux réseaux identiques et comparer leurs sorties. Le réseau prend en entrée deux images (x1 et x2) qui passent à travers deux réseaux identiques. Le réseau de neurones produit deux vecteurs qui

sont les sorties extraites des deux images d'entrée. Puis nous calculons la distance (d) entre les deux vecteurs de sorties pour comparer les deux images et voir si elles sont similaires ou dissimilaires.

3.7.1 Architecture du modèle :

Le réseau siamois est un réseau de neurones d'apprentissage profond supervisé. Son architecture conçue pour apprendre à différencier ou comparer des paires de données en entrées, ce qui est utile pour des tâches de vérification comme la reconnaissance faciale. Contrairement aux réseaux de neurones traditionnels, un réseau siamois prend deux entrées simultanément et les traite à l'aide de deux sous-réseaux (ou branches) identiques c'est-à-dire qu'ils ont les mêmes paramètres et les mêmes poids.

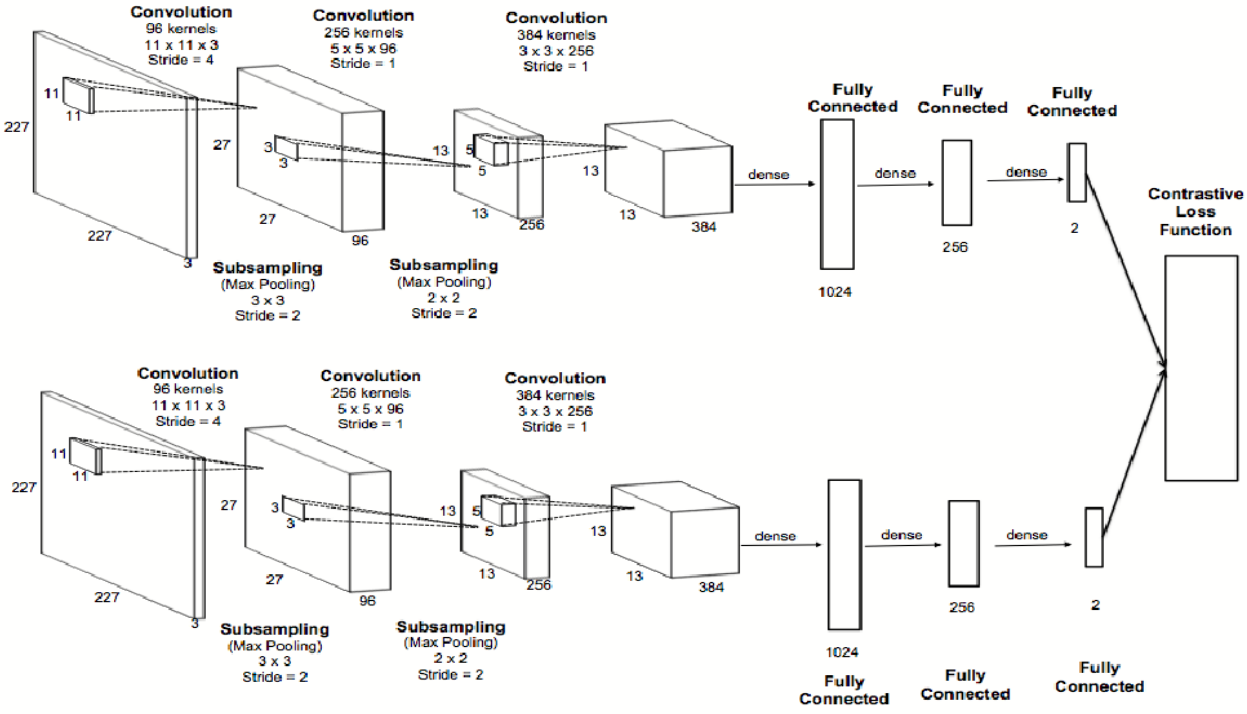

Figure 3.24 (Rihabziani, 2022) : Architecture du réseau siamois

Dans l'architecture de ce modèle, nous avons comme entrée deux images de tailles (227, 227, 3) qui passent à travers deux réseaux convolutifs de mêmes paramètres. Il est composé d'une série de couches convolutives suivie de couches denses pour chaque réseau. La différence avec un réseau CNN classique est que dans le réseau siamois nous nous arrêtons aux couches denses il n'y a pas de couches Softmax. Les deux couches denses sont donc liées pour former un seul neurone avec la fonction d'activation sigmoïde (0 ou 1). Ensuite nous calculons la distance entre les deux vecteurs de sorties. Plus la distance est grande, plus les deux images sont dissimilaires et inversement.

$$\text{Avec } d = || f(x_1) - f(x_2) ||$$

3.8 Conclusion

Les CNN et les ViT représentent deux approches utilisables en treaitement d'image, particulièrement dans la reconnaissance faciale, chacune avec ses propres forces. Les CNN sont excellents dans l'extraction de caractéristiques locales grâce à leurs filtres convolutifs, tandis que les ViT capturent efficacement les dépendances globales grâce à leurs mécanismes d'attention. Dans la suite de ce mémoire, nous fournirons les solutions proposées au problème de l'identification des personnes, en faisant usage du modèle de réseaux siamois basés sur le CNN et ViT. Ces solutions seront présentées dans le cadre du chapitre suivant de ce mémoire.

CHAPITRE 4 : MÉTHODOLOGIE

4.1 Introduction

Dans ce chapitre, nous présentons la méthodologie que nous avons élaborée pour résoudre la problématique abordée dans ce mémoire. Nous commençons par expliquer les démarches de collecte et d'organisation des données. Ensuite, nous explorons deux approches principales pour les réseaux siamois dans la reconnaissance de visage : les réseaux siamois basés sur les Visions Transformers (ViT) et ceux basés sur les réseaux convolutifs (CNN). Les solutions proposées pour résoudre ce problème consistent à apprendre des représentations de paires d'images et à déterminer si ces paires appartiennent à la même classe.

4.2 Ensemble de données :

La reconnaissance faciale est une tâche complexe dans des scénarios du monde réel. Pour cela, un ensemble de données d'entrainement correctement étiqueté est requis. Recueillir des images faciales et les étiqueter correctement est une tâche qui prend du temps. Il existe de nombreux ensembles de données accessibles au public à cet effet. Au fil du temps, de plus en plus de chercheurs et d'entreprises se lancent dans ce domaine. Ils investissent du temps et de l'argent, de sorte que la taille de l'ensemble de données devient de plus en plus grande. Certains ensembles de données accessibles au public ont traité des millions d'images de visages.

De nos jours, la plupart des images utilisées pour créer de nouveaux ensembles de données sont collectées sur différents réseaux sociaux ou sites Web. Le principal problème de la reconnaissance faciale par image est que la plupart des traits du visage changent avec la pose ou l'âge. Pour mettre en évidence ce problème, certains chercheurs ont ajouté des images dans différentes poses et restrictions d'âge. Certains ensembles de données contiennent également des images de visages synthétiques pour augmenter le nombre d'images dans leur collection, comme GANFaces500k. La reconnaissance des visages portant des masques a attiré l'attention des chercheurs après l'épidémie de covid-19. Certains ensembles de données (par exemple, RMFRD,

SMFRD) sur les personnes portant des masques et ne portant pas de masques sont déjà accessibles au public. Ces données permettent de faire la détection des visages avec masques ou sans masques. Cependant, nous voulons faire l'identification des personnes même s'ils portent un masque. Pour ce faire, nous allons créer notre propre ensemble de données en collectant des images de visages avec masque et sans masque de différents individus. Pour chaque individu, nous allons stocker les images dans des dossiers annotés afin de pouvoir faire l'identification des personnes.

4.3 Préparation des données

La première étape consiste à collecter les données de visages masqués et non masqués de plusieurs personnes. Ensuite, nous procédons au filtrage des données afin de s'assurer que nos images soient bien claires, éliminer les doublons et supprimer les images floues. Cette étape est cruciale avant la création de notre base de données, car ceci facilitera l'apprentissage du modèle lors de l'entrainement. Par la suite, nous séparons nos données en plusieurs classes : dans chaque classe, nous avons des images de visages avec masques et de visages sans masques, dans le but d'identifier la même personne avec ou sans masque. Cette étape est la partie la plus longue et la plus importante, car elle nous permettra une analyse beaucoup plus efficace et limitera les erreurs et imprécisions qui peuvent subvenir lors du traitement. Enfin, nous allons créer des couples d'images de notre base de données, 1 indique que les deux images sont de la même personnes et 0 indique qu'elles appartiennent à des personnes différentes.

4.4 Création des modèles

Nous proposons deux modèles de réseaux de neurones siamois qui sont basés sur la convolution (Réseaux Siamois VGG-19) et l'auto-attention (Siamois Vision Transformer). Chaque modèle est composé de deux sous-réseaux identiques (respectivement VGG-19 et ViT). Les sous-réseaux extraient les caractéristiques des paires d'images prises en entrées, et la distance entre ces caractéristiques est ensuite calculée pour déterminer la similarité entre les deux images. Les modèles (respectivement Réseaux Siamois VGG-19 et Siamois Vision Transformer) tirent parti de la capacité respectivement de VGG-19 à extraire des caractéristiques locales d'une image

grâce à des couches de convolution et des Visions Transformers à traiter une image en la divisant en patchs, puis en appliquant le mécanisme d'attention pour extraire des caractéristiques globales.

4.4.1 Réseaux Siamois VGG19

Dans ce modèle se concentre principalement sur l'extraction des caractéristiques locales des images. Il utilise des couches convolutives et des filtres qui opèrent sur de petites régions de l'image (appelées champs réceptifs). Cela permet au modèle de détecter des motifs locaux tels que les bords, les textures ou les formes simples, qui sont ensuite combinés dans des couches plus profondes pour identifier des structures complexes. Cette approche hiérarchique est efficace pour capturer des détails fins dans les images, mais elle peut avoir des limitations lorsqu'il s'agit de modéliser des relations globales ou à longue distance entre les pixels.

4.4.1.1 Architecture de VGG-19

Dans le cas d'un modèle CNN, nous disposons d'une série de couches convolutionnelles et de regroupement suivi de couches denses et d'une couche de sortie avec une fonction Softmax.

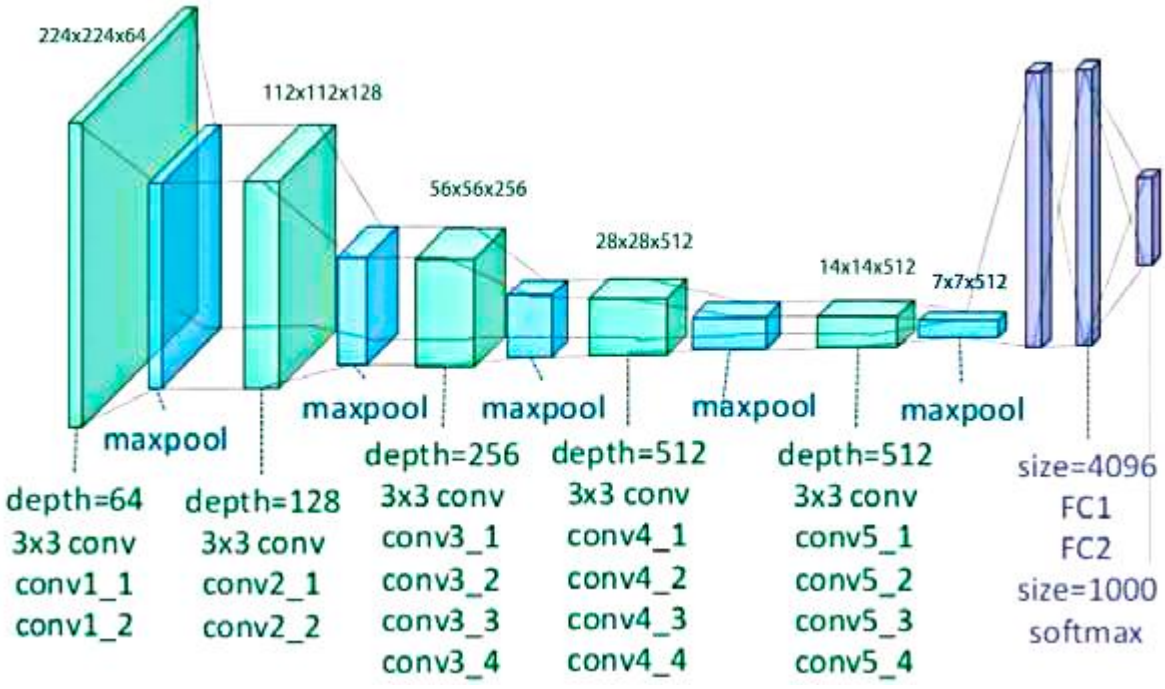

Figure 4.1 (Zheng et al., 2018) : Architecture VGG-19

- Nous utilisons le modèle pré-entraîné VGG-19 qui prend en entrée une image RVB (Rouge, Vert, Bleu) de taille fixe (224 * 224), ce qui signifie que la matrice a la forme (224, 224, 3).
- Le seul prétraitement effectué consiste à soustraire la valeur RVB moyenne de chaque pixel (calculée sur l'ensemble de l'ensemble d'entraînement).
- Utilisant des noyaux de taille (3*3) avec un pas de 1 pixel, cela leur permet de couvrir tous les pixels de l'image.
- Un remplissage spatial est utilisé pour préserver la résolution spatiale de l'image.
- Le pooling maximum est effectué sur une fenêtre de 2 * 2 pixels avec sride 2.
- Vient ensuite l'unité linéaire rectifiée (ReLu), qui introduit la non-linéarité pour améliorer la classification des modèles et réduire le temps de calcul, puisque les modèles précédents utilisaient des fonctions tanh ou sigmoïdes, qui se sont avérées bien meilleures que ces fonctions.

- Trois couches entièrement connectées sont implémentées, pour lesquelles la taille des deux premières couches est de 4096, puis une couche à 1000 cellules, et la dernière couche est une fonction Softmax.

4.4.1.2 Architecture du Réseau Siamois VGG19

Dans l'architecture de ce modèle, nous avons comme entrée deux images de tailles (224, 224, 3) qui passent à travers deux réseaux VGG19 de mêmes paramètres. Il est composé d'une série de couches convolutives et de Pooling suivies de quelques couches denses pour chaque réseau. Avec les réseaux siamois, nous avons une construction similaire de couches convolutives et de pooling, sauf que nous n'avons pas de sortie Softmax. Comme expliqué précédemment, puisque le réseau a deux images en entrées, on se retrouvera avec deux couches denses. Les deux couches denses sont donc liées pour former une seule couche avec l'utilisation d'une couches de deux neurones avec un softmax était préférable.

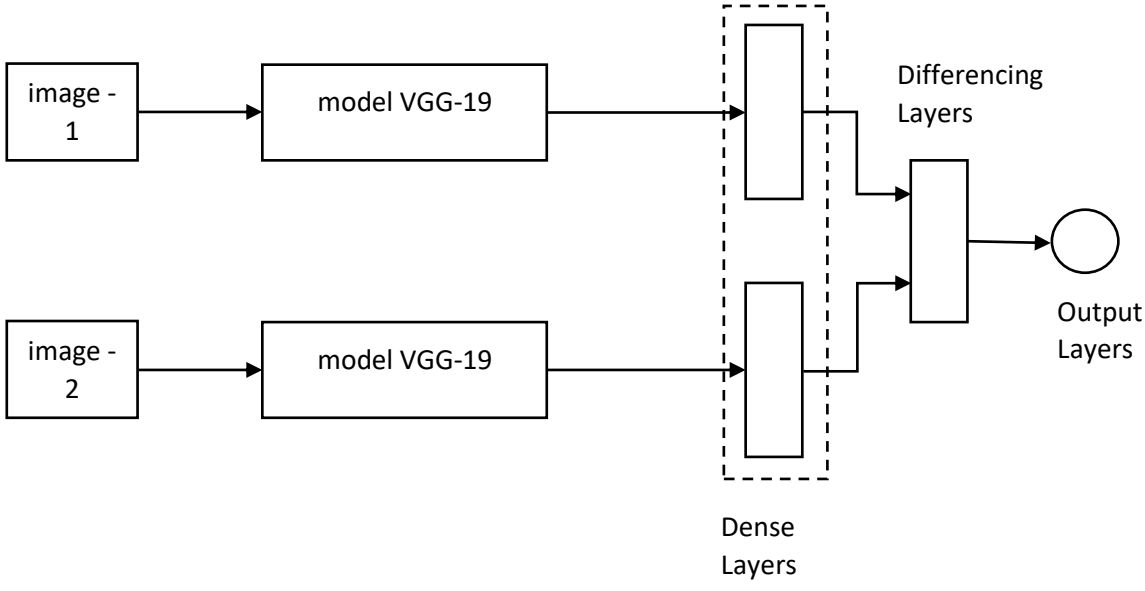

Figure 4.2 : Architecture du Réseau Siamois VGG19

4.4.2 Siamois Vision Transformer

Ce modèle adopte une approche différente en utilisant des mécanismes d'attention pour capturer les dépendances globales dans les images. Dans un ViT, l'image est divisée en patchs (petites régions), qui sont ensuite traités comme une séquence, similaire au traitement de texte dans les Transformers. Grâce à l'attention multi-têtes, le modèle est capable de pondérer l'importance relative de chaque patch par rapport aux autres, permettant ainsi de modéliser les relations entre des régions éloignées de l'image. Cette capacité à comprendre le contexte global rend les ViTs particulièrement adaptés pour des tâches où les interactions à longue portée jouent un rôle crucial.

4.4.2.1 Architecture de Vision Transformer

Le diagramme ci-dessous montre l'architecture Vision Transformer. L'architecture est divisée en 3 composants : l'intégration, l'encodeur de transformateur et le MLP.

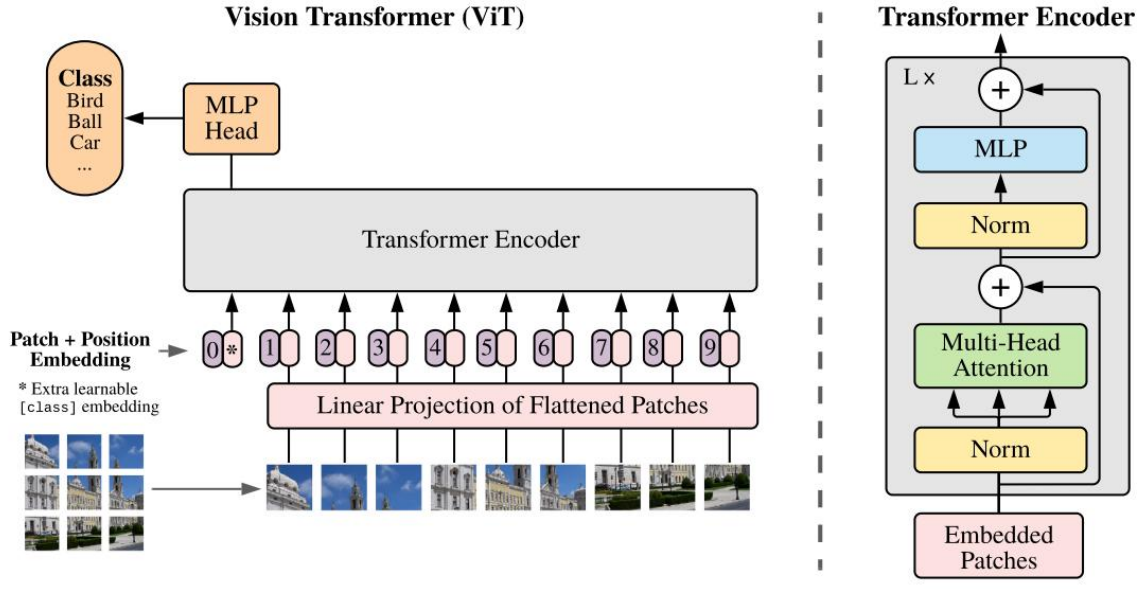

Figure 4.3 (Dosovitskiy et al., 2020) : architecture d'un Vision Transformer

- Étape 1 : intégration

Dans cette étape, nous divisons l'image d'entrée en patchs de taille fixe de dimension [P, P] et les aplatissons linéairement, en concaténant les canaux. Par exemple, un patch de taille [P, P, C] est converti en [P×P×C, 1]. Ce patch linéairement aplati passe ensuite à travers une couche Feed-Forward avec une fonction d'activation linéaire pour obtenir une projection de patch linéaire de la dimension [D, 1]. D est l'hyperparamètre appelé dimension d'intégration utilisé dans tout le transformateur.

L'image peut être corrigée à l'aide d'une couche convolutive en gardant le pas égal à la taille du patch. Cela convertira l'image d'entrée en patchs de la taille requise, qui seront ensuite aplatis et transmis à la couche suivante.

Pour comprendre un peu mieux l'étape d'intégration, analysons les dimensions. Supposons que nous ayons une image d'entrée de taille 224×224×1, elle est ensuite divisée en patchs de taille fixe 16×16. Notons la taille du patch par P et les canaux de l'image par C. Le nombre total de patchs N que nous obtenons est 196.

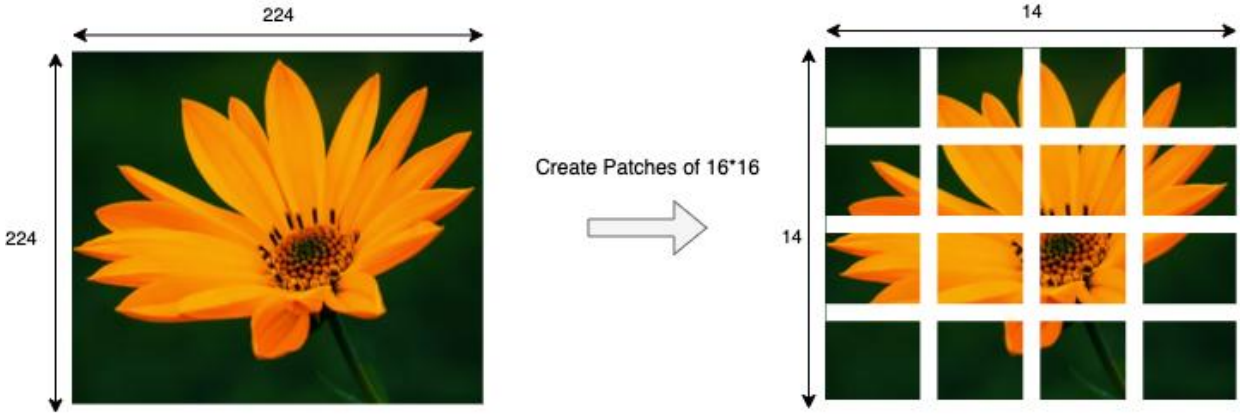

Figure 4.4 (Dzlab, 2021) : configuration d'un patch

Après avoir aplati linéairement tous les patchs pour obtenir un vecteur X de dimension [N, P^2C], Il passe à travers une couche dense pour le convertir en un vecteur de dimension D appelé intégration E [N, D]. Nous ajoutons ensuite une classe apprenable intégrant [1, D] pour convertir le vecteur E en dimension [N+1, D]. La dernière étape consiste à ajouter un codage positionnel

pour obtenir le vecteur final Z. Les intégrations de classe et de position sont des vecteurs initialisés aléatoirement, appris lors de la formation du réseau.

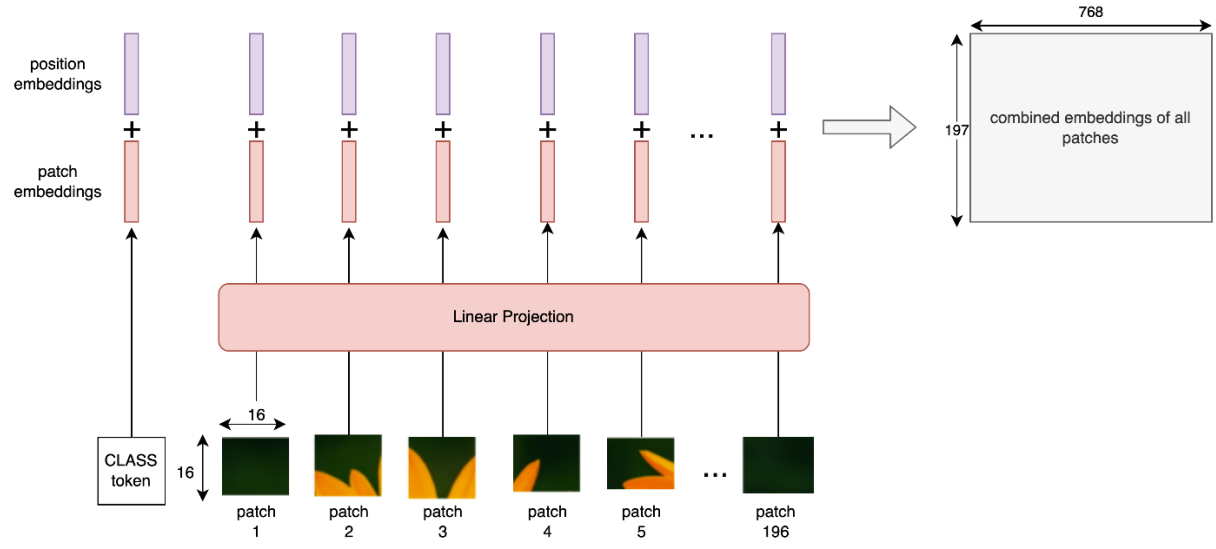

Figure 4.5 (Dzlab, 2021) : Encodage des patchs

Une fois que nous avons notre vecteur Z, nous le faisons passer à travers une couche Transformer-encodeur.

- Étape 2 : Encodeur de Transformer

L'architecture de l'encodeur est un Transformer Encoder. L'encoder est composé de plusieurs piles de blocs identiques. Chaque bloc possède une couche d'attention multi têtes suivie d'une couche Feed-Forward. Il existe une connexion résiduelle autour de chacune des deux sous-couches, suivie d'une normalisation des couches. Toutes les sous-couches ainsi que les couches d'intégration dans le modèle produisent une sortie de dimension intégrée D. Le vecteur Z de l'étape précédente passe par l'architecture de l'encodeur du transformateur pour obtenir le vecteur de contexte C.

L'architecture de l'encodeur du Transformer se compose de plusieurs blocs d'encodeurs, chaque bloc étant doté d'une unité d'attention multi têtes et d'un réseau Feed-Forward. Chaque couche est également suivie d'une couche de normalisation.

➢ Attention multi têtes :

Le composant principal d'une unité d'attention multi têtes est l'attention de produit scalaire. Dans un premier temps, le vecteur d'entrée Z est dupliqué 3 fois et multiplié par les poids Wq, Wk et Wv, pour obtenir respectivement les requêtes, les clés et les valeurs. Les requêtes sont ensuite multipliées par les clés et le résultat est divisé par la racine carrée de la dimension, pour éviter le problème de disparition de gradient. Cette matrice passe par une couche Softmax et est multipliée par les valeurs pour nous donner le résultat final appelé Head H.

L'attention du produit scalaire mise à l'échelle, comme expliquée ci-dessus, est appliquée h fois (h = 8) pour attirer h têtes d'attention. Ces têtes d'attention sont concaténées et passées à travers une couche dense pour obtenir le vecteur final de dimension D intégrée.

Pour en revenir à notre architecture de Transformer-encoder, le vecteur Z traverse plusieurs blocs d'encodeur pour donner le vecteur de contexte final C.

- Étape 3 : Tête MLP

Une fois que nous avons notre vecteur de contexte C, nous ne nous intéressons qu'au jeton de contexte C_0 à des fins de classification. Ce jeton de contexte C_0 est transmis via une tête MLP pour nous donner le vecteur de probabilité final pour aider à prédire la classe. La tête MLP est implémentée avec une couche cachée et tanh comme non-linéarité au stade de pré-entraînement et par une seule couche linéaire au stade de réglage fin.

L'architecture finale est présentée dans la figure 4.5. Les patchs d'image linéaires sont préfixés par un jeton et passés à travers une couche dense pour obtenir le vecteur de codage final Z ainsi que les intégrations positionnelles. Celui-ci est ensuite transmis à travers une architecture de transformer-encodeur pour obtenir le vecteur de contexte C. La valeur du jeton de contexte C_0 est transmise à travers une tête MLP pour obtenir la prédiction finale.

4.4.2.2 Architecture Siamois Vision Transformer

Dans l'architecture de ce modèle, nous avons comme entrer deux images de tailles (224, 224, 3) qui passe à travers deux réseaux ViT de mêmes paramètres. Les images d'entrées sont divisées en patchs de taille fixe. Le jeton est ajouté au début du vecteur de patch et passés à travers une couche dense pour obtenir le vecteur de codage final Z ainsi que les intégrations positionnelles.

Celui-ci est ensuite transmis à travers une architecture d'encodeur Transformer pour obtenir le vecteur de contexte C. La valeur du jeton de contexte C_0 est transmise à travers une tête MLP qui a une couche dense comme dernière couche. Nous nous retrouvons alors avec deux couches denses qui sont liées pour former une seule couche avec la fonction d'activation softmax.

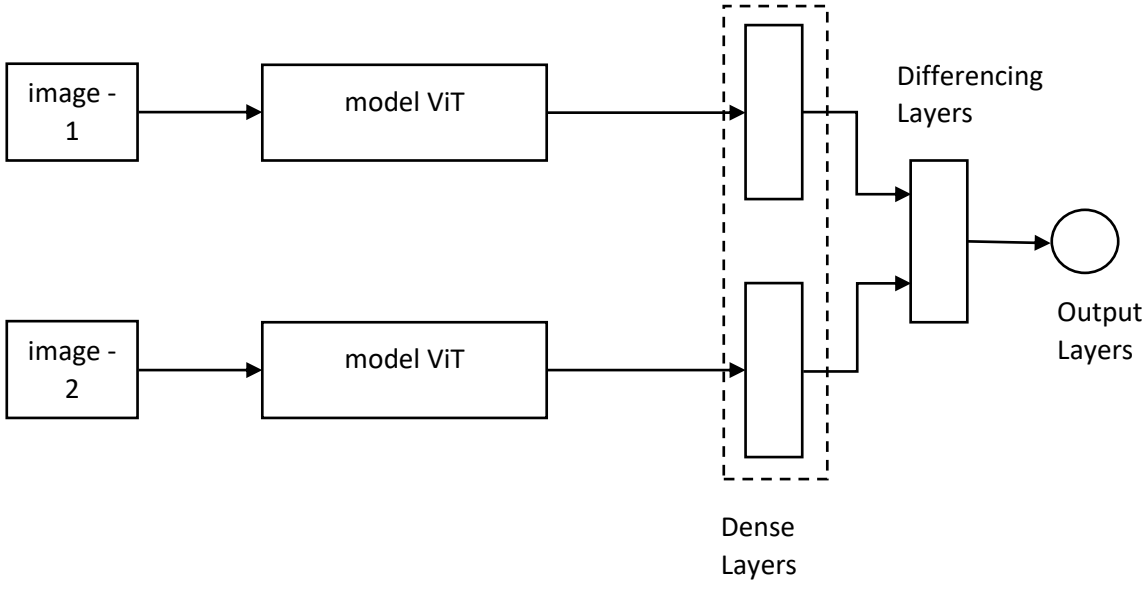

Figure 4.6 : Architecture du Siamois Vision Transformer

4.5 La fonction de perte contrastive

La fonction de perte contrastive permet de mesurer la distance entre les représentations de caractéristiques de deux images en entrée. Elle a pour but d'évaluer à quel point le réseau siamois parvient à différencier deux éléments. Elle prend la sortie du réseau pour un exemple positif et calcule sa distance par rapport à un exemple de la même classe et la compare avec la distance par rapport aux exemples négatifs. Autrement dit, la perte est faible si les échantillons positifs sont codés dans des représentations similaires (plus proches) et les exemples négatifs sont codés dans des représentations différentes (plus éloignées).

Mathématiquement, la fonction de perte est définie comme suit :

$$Loss(A, B, Y) = (Y) * || F(A)-F(B)||^2 + (1-Y) * (max [0, m - || F(A)-F(B)||^2])$$

Avec *Y* qui représente la classe (soit 0 pour les images de différentes et 1 avec les images de mêmes classes) ;

$||F(A) – F(B)||$ qui est la distance euclidienne d entre nos deux sorties ;

Et *m* qui est la marge.

4.6 La validation croisée (ou cross-validation)

C'est une méthode d'évaluation de modèles utilisée en apprentissage automatique pour estimer la performance d'un modèle de manière plus fiable, surtout lorsque les données sont limitées. Au lieu de diviser les données en un simple ensemble d'entraînement et de test, la validation croisée divise les données en plusieurs sous-ensembles (folds), puis elle entraîne et teste le modèle plusieurs fois, à tour de rôle.

On divise le jeu de données en k sous-ensembles de taille égale. Le modèle est entraîné k fois, chaque fois en utilisant :

- k−1 folds pour l'entraînement
- 1 fold pour le test (différent à chaque fois)
- À la fin, on donne la moyenne des performances obtenues sur les k tests.

Il existe plusieurs variantes de validation croisée :

- Stratified k-fold : conserve la proportion des classes (utile pour les données déséquilibrées).
- Leave-One-Out (LOO) : chaque fold ne contient qu'un seul échantillon (k = nombre total d'observations).
- Repeated k-fold : répète la k-fold plusieurs fois avec des répartitions différentes.

4.7 Les métriques de performances

Pour l'évaluation de nos différents modèles, nous avons utilisé les mesures de performance suivantes : Précision, rappel, la mesure F1 et la matrice de confusion.

- La précision représente le nombre de prédictions correctement classées sur le nombre total de prédictions :

$$précision = \frac{\text{vraies positives}}{\text{vraies positives} + \text{faux positives}}$$

- Le rappel détermine l'exactitude de la classe positive d'une prédiction. En d'autres termes c'est le nombre de positifs bien prédits (Vrais Positifs) divisé par l'ensemble des positifs (Vrais Positifs + Faux Négatifs) :

$$rappel = \frac{\text{vraies positives}}{\text{vraies positives} + \text{faux négative}}$$

- La mesure F1 est l'équilibre de la précision et du rappel. Elle est petite si l'une des métriques est petite. Elle est donnée par la formule suivante :

$$F1 = 2\text{x}\frac{\text{précision} * \text{rappel}}{\text{pécision} + \text{rappel}}$$

- La matrice de confusion est un tableau utilisé pour évaluer les performances d'un modèle de classification. Elle compare les prédictions du modèle aux véritables étiquettes de classe, permettant de visualiser où le modèle commet des erreurs.

  Vrais positifs (VP) : Nombre de cas correctement prédits comme positifs.

  Faux négatif (FN) : Nombre de cas positifs réels prédits comme négatifs (erreur de sous-estimation).

  Faux positifs (FP) : Nombre de cas négatifs réels prédits comme positifs (erreur de surestimation).

  Vrais négatifs (VN) : Nombre de cas correctement prédits comme positifs.

- La courbe AUC-ROC (Area Under the Curve - Receiver Operating Characteristic) est un outil utilisé pour évaluer les performances d'un modèle de classification binaire. Elle trace le taux de vrais positifs (True Positive Rate, ou Recall) en fonction du taux de faux positifs (False Positive Rate) pour différents seuils de classification. La performance du modèle est déterminée en fonction de la valeur de l'AUC. Elle est meilleur quand la valeur de l'AUC se rapproche plus de 1.
- La courbe Precision-Recall (PR) et l'AUC (Area Under the Curve) associée sont des outils d'évaluation pour les modèles de classification, en particulier lorsque les classes sont déséquilibrées. Elle montre le compromis entre la précision et le rappel à différents seuils de classification. L'aire sous la courbe PR (AUC-PR) quantifie la performance globale du modèle en fonction de sa valeur, plus l'AUC est proche de 1, meilleur est le modèle.

  Plus robuste que l'AUC-ROC en cas de classes déséquilibrées. Une AUC-PR élevée indique que le modèle maintient une bonne précision même avec un rappel élevé.

4.8 Conception et Implémentation

4.8.1 Environnement de travail

Fabricant	HP
Processeur	Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz 2.59 GHz
Mémoire physique (RAM) installée	16.00 Go
Système d'exploitation	Windows 11
Architecture	64 bits
Connexion	À distance en utilisant le protocole RDP (Remote Desktop Protocol)

Tableau 4.1 : Configuration Matérielle

4.8.2 Outils

4.8.2.1 Langage de programmation

Le langage Python (Python, 2024) est très utilisé dans le domaine de vision par ordinateur et de la science des données. Depuis sa création en 1989, le langage Python open source extensible est dynamique, compact, gratuit et modulaire. Il encourage une approche de programmation orientée objet plutôt que de la forcer. Il est très avancé pour la construction des systèmes de reconnaissance faciale.

Son plus grand avantage réside dans le large éventail de bibliothèques qu'il contient. Un autre avantage dont elle dispose est sa forte communauté qui continue de stimuler son développement. Compte tenu de toutes ces qualités, nous avons choisi Python.

4.8.2.2 Bibliothèques

- Numpy (Numerical Python)(Numpy, 2024)

C’est une bibliothèque qui gère les opérations scientifiques permettant de manipuler des tableaux unidimensionnels et multidimensionnels. Il implémente de nombreuses routines en utilisant des tableaux multidimensionnels d’objets. Numpy est une bibliothèque polyvalente et rapide.

Tout au long de ce projet, nous avons largement exploité cette bibliothèque pour charger et traiter nos données, les nettoyer et les visualiser.

- TensorFlow(TensorFlow, 2024) et Keras(Keras, 2024)

Ce sont des bibliothèques utilisées en apprentissage automatique et en apprentissage profond en Python.

TensorFlow est une bibliothèque mathématique symbolique. Pour effectuer des tâches liées à l’inférence et à la formation approfondies des réseaux neuronaux, elle utilise le flux de données et la programmation différenciable. Il s’agit d’une plate-forme open source qui permet la création d’applications d’apprentissage automatique grâce à une variété d’outils, de bibliothèques et de ressources communautaires.

Keras est une interface spécialisée dans la construction de réseaux de neurones qui a été créée indépendamment de Tensorflow. Mais plus tard, dans la version 2.0 de Tensorflow, elle a été iintégrée àTensorFlow en tant qu'API. Elle est intuitive et facile à utiliser pour manipuler les couches dans les réseaux neuronaux.

Dans le cadre de ce travail, nous les avons utilisées pour la conception de nos modèles de reconnaissances faciales.

- Matplotlib(Matplotlib, 2024)

C’est une bibliothèque qui a été introduite en 2002 par John Hunter. Il s’agit d’une bibliothèque de visualisation Python permettant de tracer des graphiques de tableaux de données. C’est l’une des bibliothèques de visualisation de données les plus utilisées au monde.

Nous l'avons utilisée dans ce travail pour la visualisation des données et les résultats des entrainements.

- Scikit Learn (Scikit-learn, 2024)

C'est l'une des meilleures bibliothèques pour l'apprentissage automatique en Python. Elle est incontournable pour tout projet de science des données. C'est un outil simple à utiliser et très rapide pour l'analyse prédictive des données et la modélisation statistique. Cette bibliothèque est construite au-dessus de NumPy, SciPy et Matplotlib.

Nous l'avons utilisé pour générer le rapport de classification des modèles après leurs entrainements.

4.8.3 Environnement de développement

4.8.3.1 Jupyter Notebook

Jupyter Notebook (Jupyter, 2024) est une application Web open source pour le codage et la visualisation de données. Les utilisateurs peuvent créer des blocs de code sous forme de notebook et possède une interface très interactive. L'application est largement utilisée dans le domaine de la recherche, notamment dans les domaines de la science des données et de l'apprentissage automatique, en raison de la facilité avec laquelle ses notebooks peuvent travailler avec du code.

Dans ce projet, nous avons utilisé Jupyter Notebook pour le prétraitement des données et leurs visualisations. Nous l'avons utilisé aussi pour entrainer les modèles et visualiser les résultats d'entrainement.

4.8.3.2 Google Colab

Google Colab (Colab, 2024) est une plate-forme de collaboration hébergée dans le cloud créé par Google. Il s'agit d'un service gratuit qui vous permet de collaborer sur des projets en

exécutant et en partageant du code Python. Cet outil est hébergé dans le cloud, nous libère sur des problématiques de capacité mémoire ou de performance machine pour mieux nous concentrer sur l'essentiel : le code de notre application. Cela signifie qu'il offre un accès aux ressources informatiques destinées à être utilisées dans le domaine de l'apprentissage automatique.

L'un des avantages de cet outil est que la plupart des bibliothèques que nous utilisons sont préinstallées. Nous pouvons donc simplement les importer et en profiter. Il s'intègre également facilement à d'autres services Google, comme Google Drive(Google Drive, 2024), ce qui nous permet de stocker des projets directement dans notre compte Drive(Google Drive, 2024).

Dans ce travail, nous avons utilisé Google Colab (Colab, 2024) pour la conception de nos modèles de reconnaissance faciale.

4.9 Implémentation.

4.9.1 Collecte des données.

La première étape consiste à collecter l'ensemble de données afin de préparer les images de visage pour entrainer nos modèles. Pour cela, nous développons un ensemble de données de visage avec masque et de visage sans masque de 14 célébrités. L'indisponibilité d'un ensemble de données standard unifié pour la reconnaissance de visage masqué nous a motivés à développer notre propre base de données. La reconnaissance faciale masquée nécessite plusieurs images de visages masqués de la même personne. Pour construire l'ensemble de données, nous avons capturé plusieurs images de la même personne dans deux configurations (masque et sans masque) à partir de vidéos extraits de YouTube (YouTube, 2024). Notre base de données contient un total de 700 images de 14 personnes, soit 50 images par personne. Pour chaque personne, nous avons collecté 25 images avec maques et 25 images sans masques. Les images de notre ensemble de données sont diverses en termes de types de masques, de conditions d'éclairage, d'angle de vue, d'occlusions, d'environnement, de dimensions et de taille, etc. Certaines images représentatives de notre ensemble de données sont présentées dans la figure suivante.

Figure 4.7 : Ensemble de données

4.9.2 Prétraitement

Le prétraitement est la deuxième étape avant d'entamer toutes tâches de reconnaissance et de classification. Le prétraitement des données est une technique d'exploration de données qui consiste à transformer des données brutes dans un format compréhensible pour un réseau de neurones.

Dans un premier temps, nous allons charger notre répertoire de base de données. Notre jeu de données est composé de 700 images, qui ont été organisées en 14 dossiers qui correspondent à nos 14 classes. Nous avons un ensemble de données de visages avec masque et de visage sans masque de 14 personnes (classes) avec 25 images avec masques et 25 images sans masque de la même personne dans chaque classe. Donc pour chaque personne (classe), il y aura 25 images avec masques et 25 images sans masque. 80 % des images seront dédiés à l'entrainement et 20% au test.

Ensuite, nous nous allons procéder à l'alignement des visages (ou face alignment) qui est une étape cruciale dans les systèmes de reconnaissance faciale. Il s'agit de repositionner ou transformer un visage détecté dans une image afin qu'il soit bien orienté et standardisé avant d'être analysé par un algorithme de reconnaissance.

Pour aligner le visage, plusieurs méthodes sont possibles. Dans ce travail, nous avons opté pour une méthode simple qui se concentre uniquement sur le contour des yeux. Nous utiliserons les configurations en Haar cascade (modules de détection frontale des visages et des yeux) avec OpenCV pour la détection des visages et des yeux.

Nous avons créé des objets face_cascade et eye_cascadedes objets de classe cv2.CascadeClassifier(). Dans ces deux objets, nous stockons les visages et les yeux détectés.

Figure 4.8 : Visage à aligner

Nous avons converti nos images en niveaux de gris, car les cascades ne fonctionnent que sur les images en niveaux de gris. Par suite, Nous avons détecté les visages et les yeux dans les images en niveaux de gris, mais dessiner des rectangles sur les images couleur correspondantes. Pour extraire les coordonnées d'un rectangle autour des visages, nous devons créer des faces variables. À l'aide de la méthode detectMultiScale(), nous obtiendrons un tuple de quatre éléments, où x et y représentent les coordonnées du coin supérieur gauche, et w et h sont la largeur et la hauteur du rectangle. Cette méthode nécessite plusieurs arguments. Le premier est l'image en niveaux de gris ; le deuxième est le facteur d'échelle qui indique de combien la taille de l'image va être réduite. Le troisième et dernier argument est le nombre minimal de voisins et y représenter les coordonnées d'un coin supérieur gauche, et w et h sont la largeur et la hauteur du rectangle.

Figure 4.9 : Détection du visage

Maintenant que nous avons le visage, nous pouvons passer à la détection des yeux. Pour cela, nous devons d'abord créer deux régions d'intérêt, situées à l'intérieur du rectangle. La première région est nécessaire pour l'image en niveaux de gris, où nous allons détecter les yeux. La seconde région est nécessaire pour l'image couleur, où nous allons dessiner les rectangles. Ensuite, nous allons détecter les yeux avec une méthode similaire à celle décrite précédemment. Nous avons donc créé une boucle for pour segmenter un œil d'un autre. Nous avons stocké les coordonnées du premier et du deuxième œil dans des variables, respectivement eye_1 et eye_2.

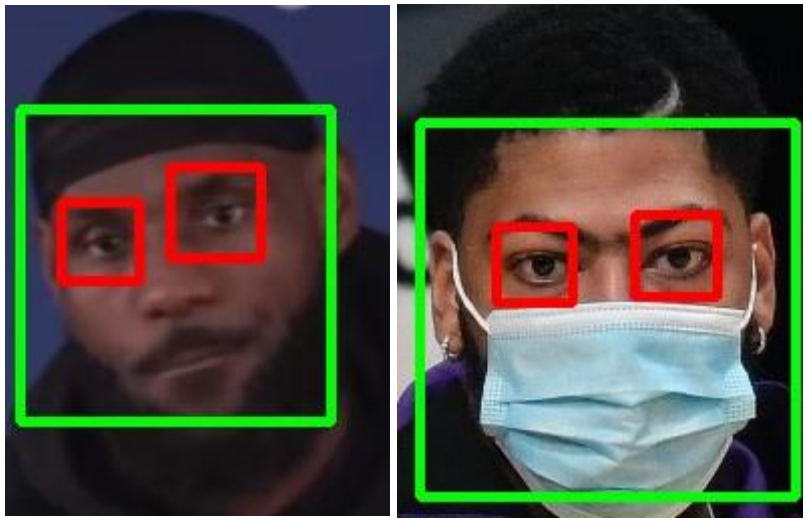

Figure 4.10 : Détection des yeux

Après avoir détecter les yeux, nous traçons une ligne entre les centres de deux yeux. Mais avant cela, nous devons calculer les coordonnées des points centraux des rectangles.

Figure 4.11 : Une ligne entre les centres de deux yeux

L'étape suivante consiste à tracer une ligne horizontale et à calculer l'angle entre cette ligne et celle reliant les deux points centraux des yeux. Notre objectif est de faire pivoter l'image en fonction de cet angle.

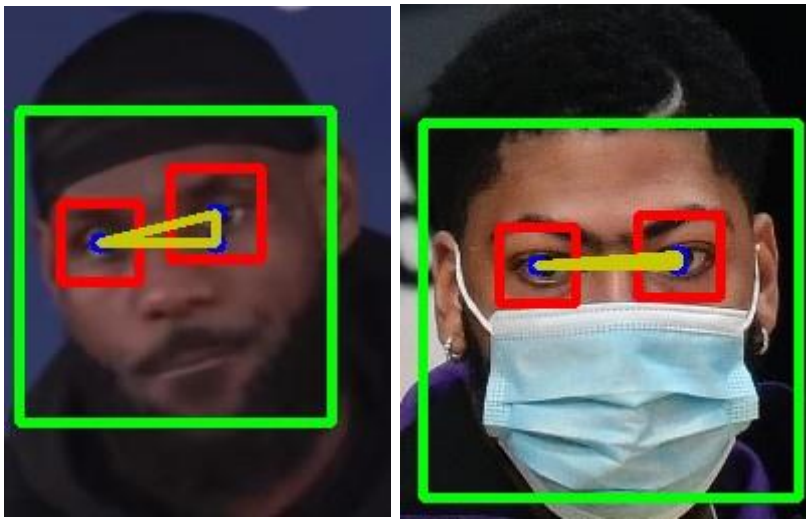

Figure 4.12 : Une ligne horizontale et à calculer l'angle entre cette

Il est important de noter qu'ici, nous avons spécifié le sens de rotation de l'image. Si la coordonnée y de l'œil gauche est supérieure à la coordonnée y de l'œil droit, nous devons faire pivoter l'image dans le sens des aiguilles d'une montre. Sinon, nous la ferons pivoter dans le sens inverse. Pour calculer l'angle, il faut d'abord trouver la longueur des deux côtés d'un triangle rectangle. On peut ensuite trouver l'angle souhaité. Pour cela, nous avons utilisé la fonction np.arctan qui renvoie l'angle en radians. Pour convertir le résultat en degrés, nous multiplions l'angle θ par 180 puis divisez-le par π. Maintenant, nous pouvons enfin faire pivoter notre image d'un angle θ.

Figure 4.13 : Visage aligné

Figure 4.14 : Résultat de l'alignement

Ensuite, nous créons nos paires d'images. Pour cela, une itération sur l'ensemble de données est appliquée. Chaque image est associée avec une image aléatoire de la même classe en tant que paire positive, et une image aléatoire d'un autre individu en tant que paire négative. Les paires positives sont étiquetées à 1 et les paires négatives 0. Ainsi, nous obtenons 1344 paires d'images, dont 1092 paires d'images destinées à l'ensemble d'entrainement et 252 paires d'images de l'ensemble de validation et des tests. Pour garder l'équilibre entre les paires positive et négative, nous nous sommes assuré que le nombre de paires dans chaque classe soit égale.

Puisque les architectures de nos modèles prennent en entrée une paire d'images de dimension 224 x 224 par image. Nous avons effectué un redimensionnement des images à la taille appropriée (224 x 224 pixels) de préférence avec 3 canaux requis, c'est-à-dire les images au format RVB (Rouge, Vert, Bleu).

Une fois redimensionnées, nous avons effectué la normalisation des valeurs des pixels qui est une étape importante dans le traitement des images pour la reconnaissance faciale. Elle permet d'améliorer la performance des modèles d'apprentissage automatique en standardisant les données. Les pixels d'une image ont généralement des valeurs comprises entre 0 et 255 (pour des images en 8 bits). Ces valeurs brutes peuvent entraîner des écarts importants dans les calculs, ralentissant ou rendant inefficace l'entraînement des modèles. En normalisant, on réduit la variance, ce qui facilite la convergence des algorithmes d'optimisation. Ce qui permet au modèle de se concentrer sur des motifs pertinents plutôt que sur des différences arbitraires de plage de valeurs. Dans le mettre en échelle la valeur des images entre 0 et 1, nous divisons chaque valeur de pixel par 255.

Après avoir normalisé, nous appliquons la fonction ImageDataGenerator qui permet d'augmenter artificiellement le nombre d'images d'entrainement que notre modèle voit en appliquant des transformations aléatoires aux images. Par exemple, nous pouvons faire pivoter ou recadrer les images de manière aléatoire ou les retourner horizontalement. Le but de cette étape est d'augmenter la taille des données d'image copiées et augmente la taille globale de toutes les données. Cette opération a été réalisée grâce à au générateur Keras, qui n'utilisera que ces copies au lieu de celles originales fournies. Cela sera utile à chaque époque lors de la phase d'entrainement des modèles.

Figure 4.15 : Exemple de pairs d'images généré

4.9.3 Construction des modèles de reconnaissance faciale

Dans cette partie, nous construisons un modèle de réseau de neurones siamois qui est basé sur l'architecture VGG-19. Nous importons le modèle préentraîné VGG-19, le modèle sera donc construit en utilisant l'apprentissage par transfert. Ensuite, nous aplatissons la sortie de la couche inférieure VGG-19 suivi d'une couche dense.

Après avoir construit le modèle VGG-19, nous allons développer le modelé ViT. Dans un premier temps, nous divisons l'image en patchs pour cela nous allons utiliser dans TensorFlow la fonction tensorflow.image.extract_patches pour extraire les patchs. Ensuite, nous mettons les patchs en séquence l'un après l'autre.

Ensuite, nous mettons en place un PatchEncoder qui prend en entrée les patchs et génère leurs plongements qui sont ensuite transmis au TransformerEncodeur. Pour chaque chemin, il

créera également un vecteur de plongements positionnels. Nous ajoutons ce vecteur aux Patch Embeddings ou plongements de patch précédents. Nous ajoutons également au début les plongements du jeton spécial.

L'implémentation de PatchEncoder est simple, nous avons besoin d'une couche Dense pour projeter une couche dans un vecteur de taille projection_dim, ainsi que d'une couche Embedding pour apprendre les plongements positionnels.

Après avoir implémenté le PatchEncoder, nous allons implémenter un perceptron multicouche (MLP) qui est composé essentiellement de deux couches denses et d'une fonction d'activation GELU. Il est utilisé dans Transformer-encodeur ainsi que dans la couche de sortie finale du modèle ViT. Nous pouvons simplement l'implémenter en tant que couche personnalisée. Ensuite, les plongements de patch sont passés par Transformer Encoder pour obtenir les représentations apprises de chaque jeton.

Après avoir défini tous les principaux composants de l'architecture ViT, nous pouvons les assembler pour construire le modèle. Il suffit de connecter un PatchEncodeur, à un TransformaterEncodeur, à un perceptron multicouche (MPL) qui a une sortie Dense pour former le modèle.

Nous précisons que pour chaque modèle, nous avons utilisé Dropout et la Regularisation L2 qui sont des techniques de régularisations utilisées pendant l'entraînement des modèles pour réduire le surapprentissage (overfitting).

Nous allons enfin créer le modèle Réseau Siamois VGG-19 qui est composé deux réseaux VGG-19 de mêmes paramètres et le modèle Siamois Vision Transformer composé de deux ViTs de mêmes paramètres. Pour chaque modèle, on lie les deux couches denses des deux réseaux pour former un seul neurone avec la fonction d'activation sigmoïde. La distance euclidienne est calculée en trouvant la racine carrée de la somme des carrés de la différence des deux plongements. La couche Lambda est utilisée à partir des couches TensorFlow à cette fin. La valeur de distance est ajustée sur une plage de 0 à 1 à l'aide de Sigmoïde.

4.9.4 Entrainement

Après avoir construit nos modèles, nous avons défini pour chaque modèle la perte de contrastive pour évaluer la qualité du travail du réseau siamois pour distinguer les paires d'images.

Pour éviter le problème surapprentissage et améliorer la généralisation, nous avons introduit le dropout qui est une technique de régularisation couramment utilisée dans les CNNs. Elle consiste à supprimer de manière aléatoire une fraction des neurones pendant l'entraînement. Cela empêche le réseau de devenir trop dépendant de neurones spécifiques, l'encourageant à développer davantage.

Nous avons personnalisé les métriques Précision, Recall et Accuracy en sous-classant la classe tensorflow.keras.metrics, dans lequel nous créons des variables d'état pour chaque métrique et update_state(self, y_true, y_pred, sample_weight=None), qui utilise les cibles y_true et les prédictions du modèle y_pred pour mettre à jour les variables d'état.

Un problème dans les réseaux de neurones est le choix du nombre d'époques lors de la formation. Trop d'époques entraineront un surajustement du modèle, tandis qu'un nombre trop faible d'époques peut entrainer un sous-ajustement.

EarlyStopping est une méthode utilisée lors de la formation des réseaux de neurones qui nous offre l'avantage d'utiliser un grand nombre d'époques de formation et d'arrêter la formation une fois que les performances du modèle cessent de s'améliorer sur l'ensemble de données de validation.

On utilise la validation croisée stratifiée (Stratified K-Fold), ce qui garantit que chaque fold contient approximativement la même proportion d'échantillons de chaque classe. Les modèles vont s'entraîner sur 5 Fold, les variables shuffle=True permet de mélanger les données avant le split et le random_state=42 assure la reproductibilité. Nous compilons ensuite les modèles avec la perte constative, les métriques personnalisées et l'optimisateur Adam pour un taux d'apprentissage de 0,001. Le résultat de chaque fold est stocké dans une variable accuracies = []. Pour chaque fold, on récupère les index d'entraînement et de validation.

Enfin, La moyenne sur tous les folds est affichée, ce qui donne une estimation robuste de la performance du modèle.

4.10 Conclusion

Nous avons exposé dans ce chapitre la méthodologie que nous avons employée pour aborder notre problématique. Nous avons exploité divers modèles et techniques pour traiter des données complexes et hétérogènes, en nous concentrant sur des aspects spécifiques des visages, qu'il s'agisse de leur identification. Le prochain chapitre dévoilera les résultats que nous avons obtenus, ce qui nous permettra d'évaluer l'efficacité de nos approches et de nos modèles.

CHAPITRE 5 : RÉSULTATS ET DISCUSSIONS

5.1 Introduction

Dans cette partie, il sera question pour nous de présenter les résultats des expérimentations menées au cours de notre travail. Ensuite, nous allons comparer les différents résultats obtenus avec nos modèles. Pour évaluer l'efficacité de nos modèles pour la reconnaissance faciale masquée, nous avons utilisé les métriques de précision, de rappel et de score F1.

5.2 Résultats et Discussions

Afin d'évaluer les performances de nos modèles, nous avons conduit un entraînement sur 50 époques en utilisant une stratégie de validation croisée. Cette méthode permet d'obtenir une estimation plus robuste et généralisable des performances du modèle, en le testant sur différentes sous-parties du jeu de données. Chaque pli de la validation croisée offre une opportunité de mesurer la capacité du modèle à généraliser sur des données non vues, tout en limitant le risque de surapprentissage. Les résultats présentés ci-dessous synthétisent les performances obtenues sur l'ensemble des plis, à travers des métriques telles que la précision, le rappel, la F1-score, la matrice de confusion et les courbes d'évaluation (AUC-ROC, PR, etc.).

	Classe	Précision	Rappel	Score-F1	Support
Réseau Siamois VGG-19	0	0,92	0,96	0,94	126
	1	0,96	0,92	0,94	126
Siamois Vision Transformer	0	0,91	0,91	0,91	126
	1	0,91	0,91	0,91	126

Tableau 5.1 : Résultats obtenus après l'entrainement des modèles.

Le tableau 5.1 montre les résultats obtenus après l'entrainement de nos modèles. L'observations « Support » de notre tableau indique le nombre de pairs d'images générées pour la validation des modèles, ce dernier étant composé de deux catégories : la première contient des pairs d'image négatifs et la seconde inclut des pairs d'images positives.

Nous remarquons que la précision, c'est-à-dire la proportion de bonnes prédictions par rapport à toutes les prédictions est égale à 0,96 (96 %) lors de la formation du modèle Réseau Siamois VGG-19 et 0,91 (91 %) lors de la formation du modèle Siamois Vision Transformer pour la reconnaissance de pairs positifs et respectivement 0.92 (92 %) et 0.91 (91 %) pour la reconnaissance des pairs négatifs. Il y a eu donc des faux négatifs (respectivement 4% et 9%), c'est-à-dire que nous n'avons pas obtenu lors de la phase de test un résultat indiquant qu'une paire d'image était négative alors qu'en réalité il ne l'était pas pour les deux modèles. Nous avons ce pendant 8 % et 9 % de faux positifs c'est-à-dire le pourcentage de pairs négatifs que les modèles (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer) ont classés comme des pairs positifs.

L'indicateur de rappel quant à lui, correspond au nombre d'objets correctement attribués à la classe i par rapport au nombre total d'objets appartenant à la classe i (le total des vrais positifs).

Cela nous permet d'estimer dans une première phase la proportion de paires positives correctement classer sur le nombre total de paires positives, ce pourcentage s'élèvent à 92 % et 91 % (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer). Dans la seconde catégorie qui concerne la reconnaissance des pairs négatifs, nous avons obtenu des pourcentages de 96 % et 91 % (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer).

Le score F1 qui combine la précision et le rappel, il faut savoir que cet indicateur est plus important que la précision, car le nombre de vrai négatif n'est pas pris en compte lorsque nous calculons la précision d'un modèle. Comme nous pouvons le voir dans le tableau 5.1, nous avons obtenu de bons résultats pour nos deux modèles (Réseau Siamois VGG-19 et Siamois Vision Transformer), les pourcentages s'élèves à 94 % et 91 % (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer) pour les deux catégories de pairs d'images. Nous pouvons donc affirmer que le pourcentage de faux négatifs et de faux positifs est faible.

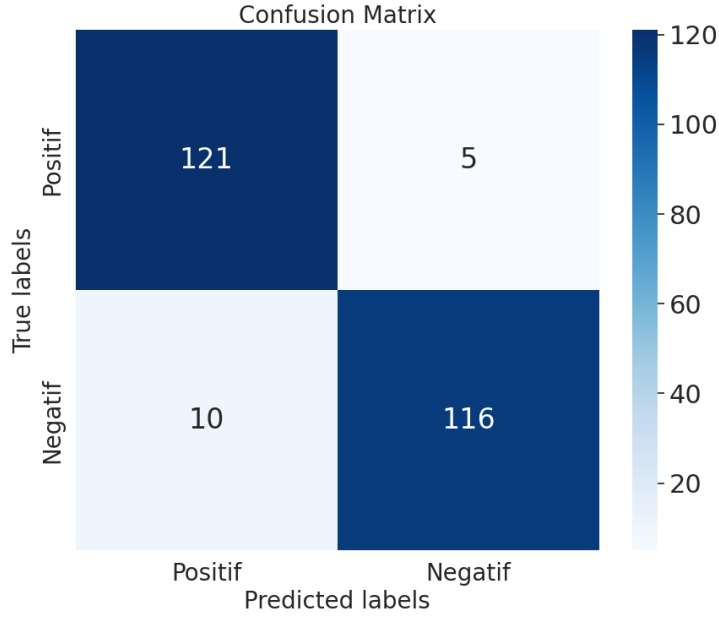

Figure 5.1 : Matrice de confusion pour le modèle Réseau Siamois VGG-19

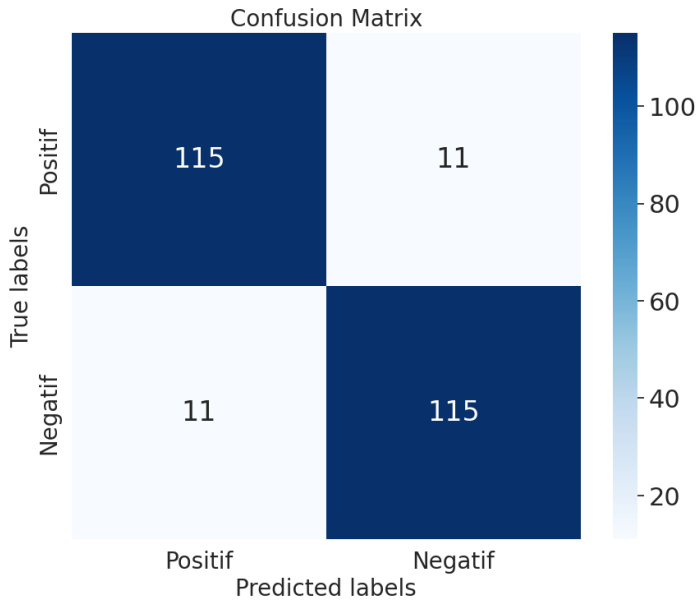

Figure 5.2 : Matrice de confusion pour le modèle Réseau Siamois Vision Transformer

La performance d'un algorithme est directement liée à sa capacité à prédire un résultat. On utilise une matrice de confusion pour comparer les résultats d'un algorithme à la réalité. Les figures ci-dessus montrent les résultats des prédictions sur 252 paires d'images soit 126 pairs positifs et 126 pairs négatifs des deux modèles. On constate que

- Le nombre de vrais positifs est 121 et 115 respectivement pour les modèles respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer.
- Le nombre de vrais négatifs est 116 et 115 respectivement pour les modèles respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer.
- Le nombre de faux positifs est 5 et 11 respectivement pour les modèles respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer.
- Le nombre de faux négatifs est 10 et 11 respectivement pour les modèles respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer.

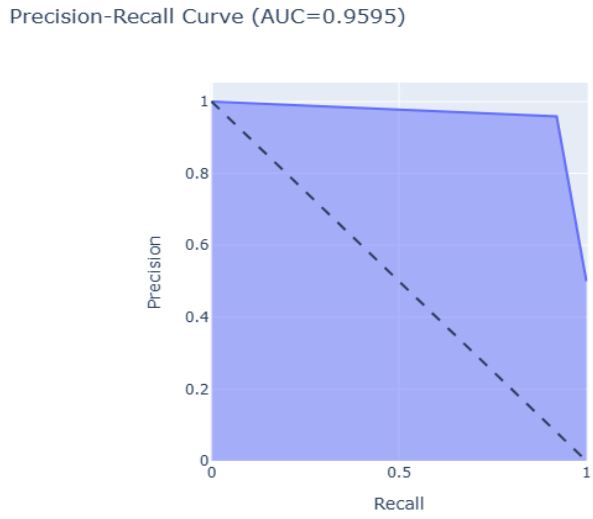

Figure 5.3 Courbe Precision-Recall (PR) et l'AUC (Area Under the Curve) du modèle Réseau Siamois VGG-19

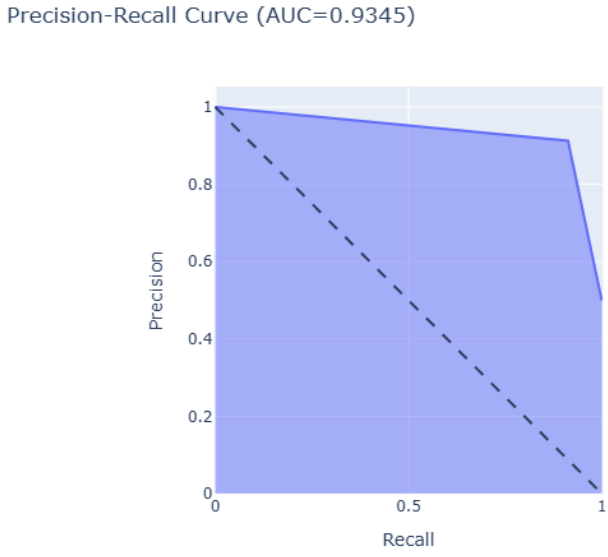

Figure 5.4 : Courbe Precision-Recall (PR) et l'AUC (Area Under the Curve) du modèle Siamois Vision Transformer

Les figures montrent la courbe Precision-Recall (ou PR curve) avec une aire sous la courbe (AUC) pour chaque modèle. On constate que les courbes sont bien au-dessus de la diagonale en pointillés, ce qui montre. On observe aussi une légère chute de la précision quand le rappel approche de 1, ce qui est courant : plus on veut tout détecter (rappel élevé), plus on risque de se tromper (baisse de la précision). L'AUC = 0.9595 et l'AUC = 0.9345 (respectivement pour les modèles respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer) indiquent que les modèles sont très performants dans la gestion du compromis entre précision et rappel. Les modèles ont une bonne capacité à identifier les vrais positifs tout en gardant une bonne précision, ce qui est très utile en cas de classes déséquilibrées.

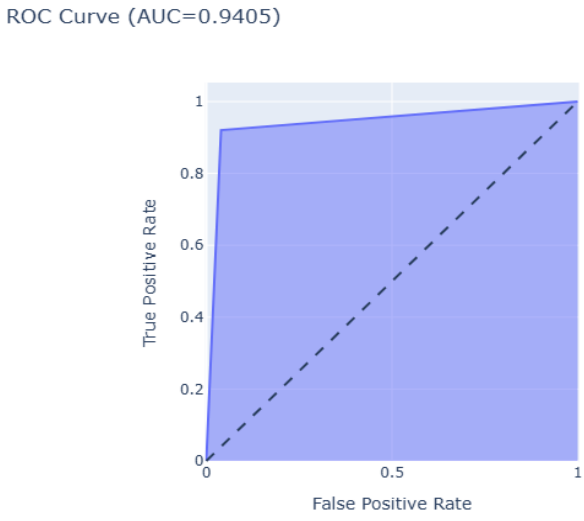

Figure 5.5 : Courbe AUC-ROC (Area Under the Curve - Receiver Operating Characteristic) du modèle Réseau Siamois VGG-19

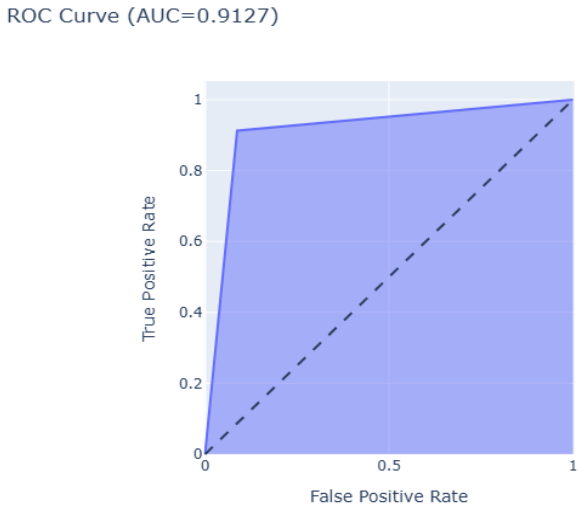

Figure 5.6: Courbe AUC-ROC (Area Under the Curve - Receiver Operating Characteristic) du modèle Siamois Vision Transformer.

Les figures montrent les courbes AUC-ROC (Area Under the Curve - Receiver Operating Characteristic) pour chaque modèle. On constate une montée rapide vers le coin supérieur gauche signifie que les modèles atteignent rapidement un haut rappel avec peu de faux positifs, ce qui est un bon signe. L'AUC = 0.9405 et l'AUC = 0.9127 (respectivement pour les modèles respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer) indiquent que les modèles ont une excellente capacité de discrimination entre les classes positives et négatives et confirme une bonne capacité de classification globale.

D'après les résultats obtenus, la détection des pairs positifs est meilleure avec le modelé Siamois Vision Transformer tandis que la détection des pairs négatifs est meilleure avec le modelé Réseau Siamois VGG-19.

Dans l'ensemble, le modèle Réseau Siamois VGG-19 a obtenu les meilleurs résultats en raison de la capacité d'extraire des caractéristiques profondes. Ces résultats comparatifs illustrent une meilleure performance du modèle Réseau Siamois VGG-19 par rapport au modèle Siamois Vision Transformer pour la reconnaissance faciale avec notre jeu de données.

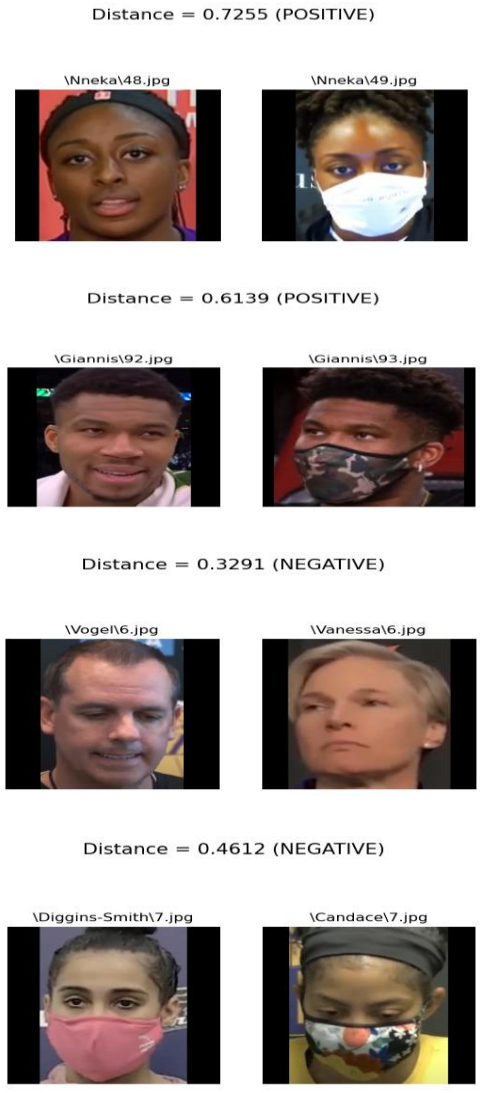

Figure 5.7 : Résultats obtenus

La reconnaissance faciale est une tâche complexe qui a été grandement améliorée par les avancées récentes en apprentissage profond. L'ajout de masques faciaux, surtout depuis la pandémie de COVID-19, a posé de nouveaux défis dans ce domaine. Dans ce travail, nous avons obtenu des résultats satisfaisants avec les deux modèles siamois, avec des performances de 94,05 % et 91,27 % de précisions (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer). Nous avons aussi testé les modèles sur un jeu de données externes disponible sur la plateforme Kaggle (kaggle, 2025). C'est un ensemble de données d'images oculaires humaines. Il est conçu pour les tâches de classification. Il contient des images infrarouges en basse et haute résolution, toutes capturées dans diverses conditions d'éclairage et avec différents appareils. Il est

adapté au test de plusieurs caractéristiques ou classificateurs entrainables. Afin de simplifier la comparaison des algorithmes, les images sont divisées en plusieurs catégories, ce qui les rend également adaptées à l'entraînement et au test des classificateurs. Nous avons obtenu de bons résultats avec les deux modèles qui sont présentés dans la section annexe, avec des précisions de 97,44% et 92,95% (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer). Les résultats obtenus montrent que les modèles parviennent de bien identifier les pairs d'images positives et les pairs d'images négatives.

Le modèle Réseau Siamois VGG-19 est plus optimisé et rapide pour le traitement d'images grâce à son architecture spécifique. Il est très efficace pour les caractéristiques locales, mais peuvent rencontrer des difficultés avec les masques qui couvrent une grande partie du visage, car ils se basent beaucoup sur les détails locaux.

Le modèle Siamois Vision Transformer grâce à leur attention globale, peut mieux gérer les masques, en trouvant des corrélations entre les parties visibles du visage et les caractéristiques globales. Ce pendant, il est plus couteux en termes de calculs et nécessite de plus grandes quantités de données pour être bien entraîné.

5.3 Conclusion

Nous avons présenté dans ce chapitre les résultats obtenus de nos deux modèles de reconnaissance faciale. Ces modèles produisent de très bons résultats avec une précision élevée et une perte moins importante. D'après les résultats obtenus, nous pouvons sans doute dire que nous avons réalisés deux modèles opérationnels et qui performent très bien.

CHAPITRE 6 : CONCLUSION GÉNÉRALE

La présente étude visait à développer un modèle de reconnaissance faciale pour l'identification des personnes malgré le port du masque facial en utilisant des techniques d'apprentissage profond. Nous avons d'abord entrepris une étude approfondie sur l'état des connaissances dans le domaine de la reconnaissance faciale et des différents concepts de l'apprentissage profond. Ensuite, nous avons proposé deux approches avec les réseaux siamois qui sont basés sur les réseaux de neurones convolutifs et les Visions Transformers.

En raison du manque d'ensemble de données étendu, nous avons créé un ensemble de données d'images de visage diversifié pour évaluer la reconnaissance faciale. Pour la classification, nous avons développé deux algorithmes de reconnaissance faciale avec le réseau siamois qui sont basés sur le réseau convolutif VGG-19 (Réseau Siamois VGG-19) et le Vision Transformer (Siamois Vision Transformer). Les modèles siamois sont conçus pour comparer deux images faciales en entrées et déterminer leur similarité. Si les images appartiennent à la même classe, elles sont classées en paires positives ; si les images appartiennent à des classes différentes, elles sont classées en paires négatives.

Les résultats obtenus ont montré que les modèles ont réussi à classer avec précision les paires d'images avec des taux de précisions égales à 94,05 % et 91,27 % (respectivement Réseau Siamois VGG-19 et Siamois Vision Transformer).

Ce projet de reconnaissance faciale présente un potentiel d'extension considérable dans le futur. Cependant, bien que nos modèles aient obtenu de bons résultats de validation, il est important de de noter que ces résultats peuvent ne pas être généralisables à tous les types d'ensembles de données d'entraînement.

Les méthodes proposées peuvent être utilisées à diverses fins, notamment la sécurité et la sûreté des personnes, c'est-à-dire la reconnaissance des voleurs portant un masque facial. Le choix entre ces deux architectures dépend donc des ressources disponibles et des spécificités des données et des applications envisagées. Pour améliorer nos modèles, nous suggérons d'explorer d'autres architectures d'apprentissage profond et d'utiliser des techniques d'augmentation de données pour entraîner nos modèles sur un plus grand nombre d'images faciales. Une approche hybride, qui

pourrait combiner les avantages des différentes architectures, pourrait également être explorée pour tirer parti des points forts de chaque modèle.

ANNEXE

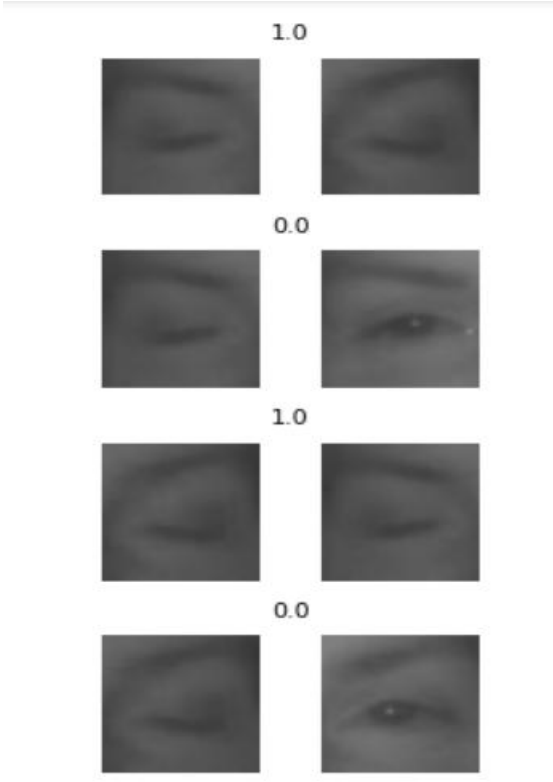

Figure 7.1 : Exemple de pairs d'images généré

(1) (2)

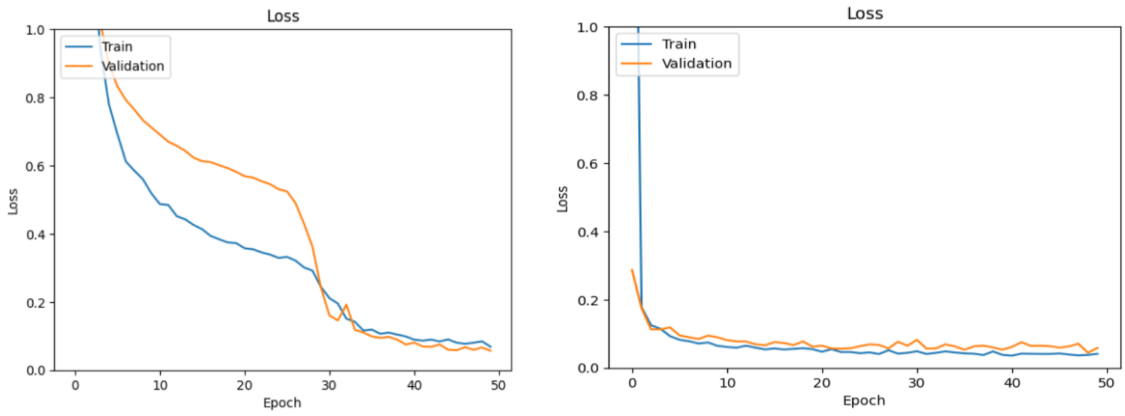

Figure 7.2 : (1) graphique de perte d'entrainement et de validation et (2) graphique de précision d'entrainement et de validation pour le modèle Réseau Siamois VGG-19.

(1) (2)

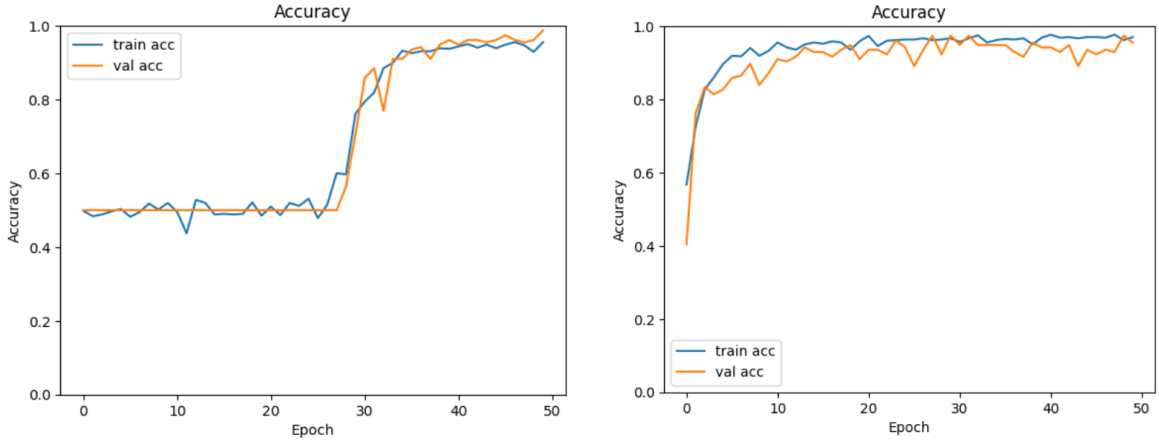

Figure 7.3 : (1) graphique de perte d'entrainement et de validation et (2) graphique de précision d'entrainement et de validation pour le modèle Siamois Vision Transformer.

	Classe	Précision	Rappel	Score-F1	Support
Réseau Siamois VGG-19	0	0,97	0,97	0,97	78
	1	0,97	0,97	0,97	78
Siamois Vision Transformer	0	0,91	0,95	0,93	78
	1	0,95	0,91	0,93	78

Tableau 7.1 : Résultats obtenus après l'entrainement des modèles.

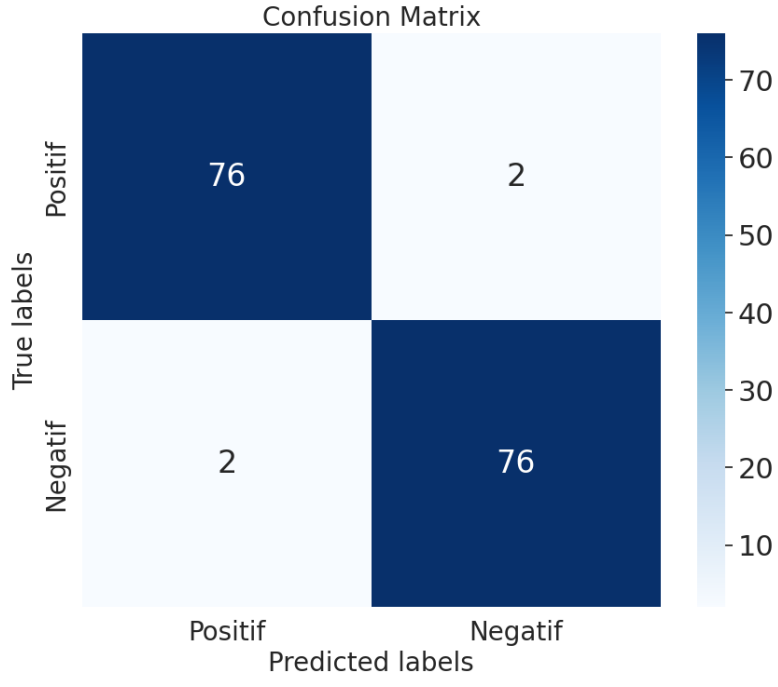

Figure 7.4 : Matrice de confusion pour le modèle Réseau Siamois VGG-19

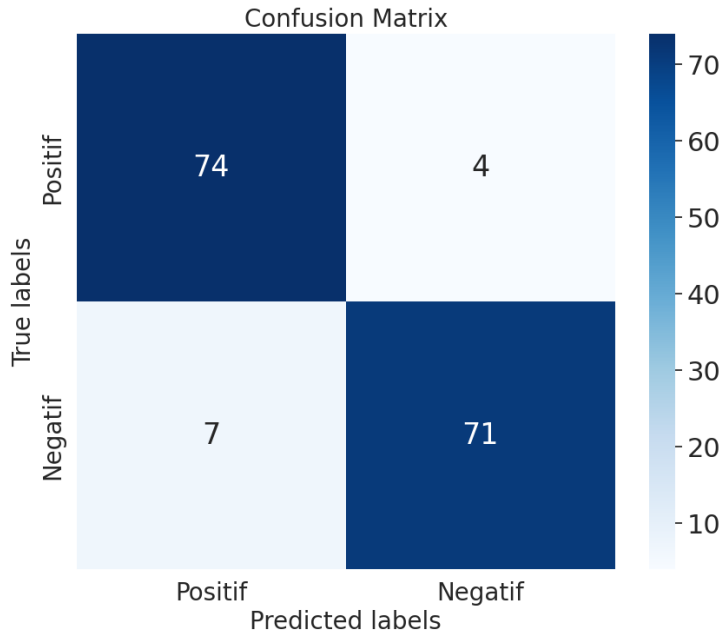

Figure 7.5 : Matrice de confusion pour le modèle Réseau Siamois Vision Transformer

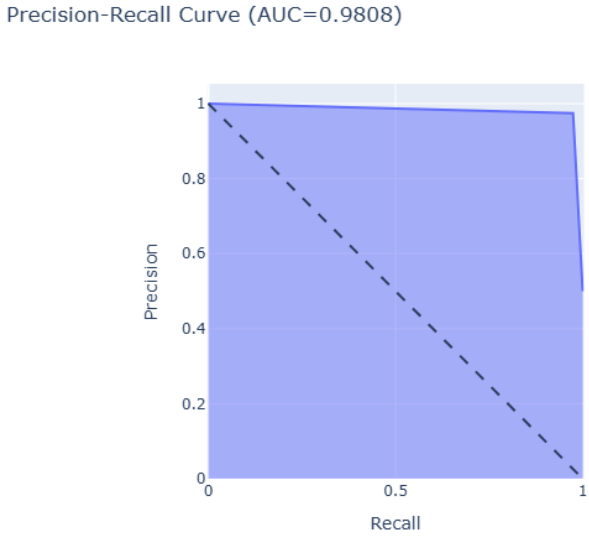

Figure 7.6 Courbe Precision-Recall (PR) et l'AUC (Area Under the Curve) du modèle Réseau Siamois VGG-19

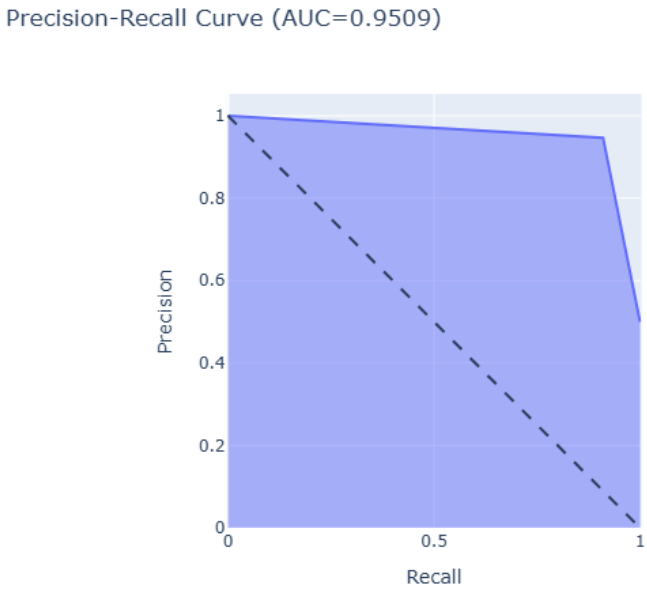

Figure 7.7 : Courbe Precision-Recall (PR) et l'AUC (Area Under the Curve) du modèle Siamois Vision Transformer

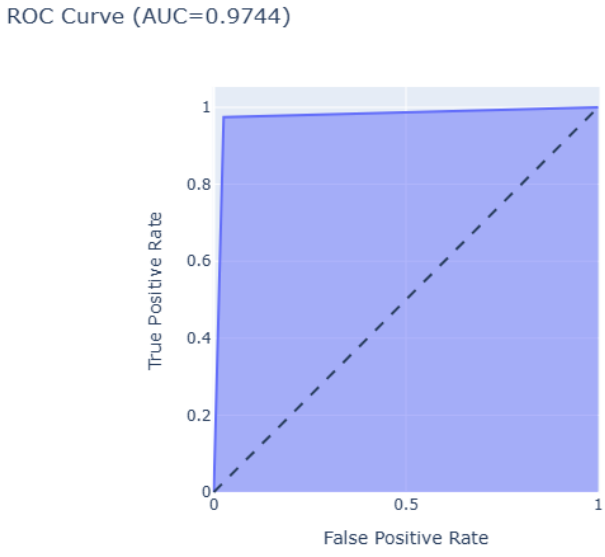

Figure 7.8 : Courbe AUC-ROC (Area Under the Curve - Receiver Operating Characteristic) du modèle Réseau Siamois VGG-19

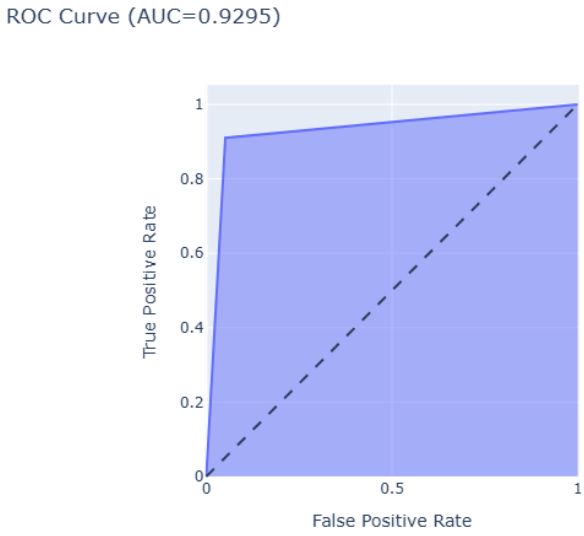

Figure 7.9: Courbe AUC-ROC (Area Under the Curve - Receiver Operating Characteristic) du modèle Siamois Vision Transformer.

RÉFÉRENCES

Ahmed, N. K., Hemayed, E. E., & Fayek, M. B. (2020). Hybrid Siamese Network for Unconstrained Face Verification and Clustering under Limited Resources. *Big Data and Cognitive Computing*, *4*(3).

Al-Ramahi, M., Elnoshokaty, A., El-Gayar, O., Nasralah, T., & Wahbeh, A. (2021). Public discourse against masks in the COVID-19 era: Infodemiology study of Twitter data. *JMIR Public Health and Surveillance*, *7*(4), e26780. https://doi.org/10.2196/26780

Aufar, Y., & Sitanggang, I. S. (2022). Face recognition based on Siamese convolutional neural network using Kivy framework. *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, *26*(2), 764-772. https://doi.org/10.11591/ijeecs.v26.i2.pp764-772

Aydemir, E., Yalcinkaya, M. A., Barua, P. D., Baygin, M., Faust, O., Dogan, S., Chakraborty, S., Tuncer, T., & Acharya, U. R. (2022). Hybrid Deep Feature Generation for Appropriate Face Mask Use Detection. *International Journal of Environmental Research and Public Health*, *19*(4).

Choi, H., Ryu, S., & Kim, H. (2018). Short-term load forecasting based on ResNet and LSTM. 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm),

Colab, G. (2024). *Bienvenue dans Colaboratory - Colab*. https://colab.research.google.com/

Damer, N., Boutros, F., Süßmilch, M., Kirchbuchner, F., & Kuijper, A. (2021). Extended evaluation of the effect of real and simulated masks on face recognition performance. *Iet Biometrics*, *10*(5), 548-561. https://doi.org/10.1049/bme2.12044

Dasun, T. (2020). *Algorithme de détection de masque facial utilisant le réseau neuronal convolutif - AI - Vision par ordinateur*. https://ichi.pro/fr/algorithme-de-detection-de-masque-facial-utilisant-le-reseau-neuronal-convolutif-ai-vision-par-ordinateur-24122843747134

Din, N. U., Javed, K., Bae, S., & Yi, J. (2020). A Novel GAN-Based Network for Unmasking of Masked Face. *IEEE Access*, *8*, 44276-44287. https://doi.org/10.1109/ACCESS.2020.2977386

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ArXiv*, *abs/2010.11929*. https://arxiv.org/abs/2010.11929

Du, H., Shi, H., Liu, Y., Zeng, D., & Mei, T. (2021). Towards NIR-VIS Masked Face Recognition. *IEEE Signal Processing Letters*, *28*, 768-772. https://doi.org/10.1109/LSP.2021.3071663

Dzlab. (2021). *Vision Transformer in TensorFlow*. https://colab.research.google.com/github/dzlab/notebooks/blob/master/_notebooks/2021-10-01-vision_transformer.ipynb#scrollTo=ovSbj8vj8c1w

Elkhaloui, A. (2018). *Fabrique & comprend ton premier réseau de neurones en partant de zéro !* https://datafuture.fr/post/fabrique-ton-premier-reseau-de-neurones/

Ge, Y., Liu, H., Du, J., Li, Z., & Wei, Y. (2023). Masked face recognition with convolutional visual self-attention network. *Neurocomputing*, *518*, 496-506. https://doi.org/10.1016/j.neucom.2022.10.025

Google Drive. (2024). *Google Drive*. https://drive.google.com/drive/u/0/home

Gruet, M. (2018). *Artificial Intelligence and prediction of the impact of solar activity on the Earth magnetic environment* UNIVERSITE DE TOULOUSE]. https://hal.science/tel-01987697

Hariri, W. (2022). Efficient masked face recognition method during the covid-19 pandemic. *Signal, image and video processing*, *16*(3), 605-612. https://link.springer.com/article/10.1007/s11760-021-02050-w

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. https://arxiv.org/abs/1512.03385

Hosen, M. I., & Islam, M. B. (2022). HiMFR: A Hybrid Masked Face Recognition Through Face Inpainting. *ArXiv*, *abs/2209.08930*. https://arxiv.org/abs/2209.08930

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv*, *abs/1704.04861*. https://arxiv.org/abs/1704.04861

Jupyter, P. (2024). *Projet Jupyter*. https://jupyter.org/

kaggle. (2025). *Détection de la somnolence | Kaggle*. https://www.kaggle.com/datasets/kutaykutlu/drowsiness-detection?resource=download

Keras. (2024). *Keras: Deep Learning for humans*. https://keras.io/

Keresh, A., & Shamoi, P. (2024). Liveness Detection in Computer Vision: Transformer-based Self-Supervised Learning for Face Anti-Spoofing. *ArXiv*, *abs/2406.13860*. https://arxiv.org/abs/2406.13860

Kertous, M., Rezai, A., & Bouraoui, I. (2021). *Classification des empreintes palmaires multi-spectrales par les réseaux de neurones convolutionnels* Université de Jijel].

Kim, M., Su, Y., Liu, F., Jain, A., & Liu, X. (2024). KeyPoint Relative Position Encoding for Face Recognition. *ArXiv*, *abs/2403.14852*. https://arxiv.org/abs/2403.14852

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. https://doi.org/10.1109/5.726791

Liang, Y., Cui, Q., Luo, X., & Xie, Z. (2021). Research on Classification of Fine-Grained Rock Images Based on Deep Learning. *Computational Intelligence and Neuroscience*, *2021*(1), 5779740. https://doi.org/10.1155/2021/5779740

Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement*, *167*, 108288. https://doi.org/10.1016/j.measurement.2020.108288

Matplotlib. (2024). *Matplotlib — Visualisation avec Python*. https://matplotlib.org/

McDonald, F., Horwell, C. J., Wecker, R., Dominelli, L., Loh, M., Kamanyire, R., & Ugarte, C. (2020). Facemask use for community protection from air pollution disasters: An ethical overview and framework to guide agency decision making. *International Journal of Disaster Risk Reduction*, *43*, 101376. https://doi.org/10.1016/j.ijdrr.2019.101376

Militante, S. V., & Dionisio, N. V. (2020, 8-8 Aug. 2020). Real-Time Facemask Recognition with Alarm System using Deep Learning. 2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC),

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. *Sustainable Cities and Society*, *66*, 102692. https://doi.org/10.1016/j.scs.2020.102692

Neuromedia. (2021). *La synapse* https://www.neuromedia.ca/la-synapse/

Numpy. (2024). *numpy*. https://numpy.org/

Oliveira, R. M. d., Araújo, R. C., Barros, F. J., Segundo, A. P., Zampolo, R. F., Fonseca, W., Dmitriev, V., & Brasil, F. S. (2017). A system based on artificial neural networks for

automatic classification of hydro-generator stator windings partial discharges. *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, *16*(03), 628-645.

OMS. (2020). *Lignes directrices pour le nouveau coronavirus (2019-nCov).* https://www.who.int/fr/emergencies/diseases/novel-coronavirus-2019/technical-guidance

PA, S. (2020). *An Overview on MobileNet: An Efficient Mobile Vision CNN*. https://medium.com/@godeep48/an-overview-on-mobilenet-an-efficient-mobile-vision-cnn-f301141db94d

Poonpinij, P., Tarnpradab, S., Lumpoon, N. N., & Wattanapongsakorn, N. (2023, 20-21 Sept. 2023). Masked Face Recognition and Identification Using Convolutional Neural Networks. 2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI),

Python. (2024). *langage de programmation python*. https://www.python.org/

Rahman, M. H., Jannat, M. K. A., Islam, M. S., Grossi, G., Bursic, S., & Aktaruzzaman, M. (2023). Real-time face mask position recognition system based on MobileNet model. *Smart Health*, *28*, 100382. https://doi.org/10.1016/j.smhl.2023.100382

Rihabziani. (2022). *facial-recognition-Siamese*. https://github.com/rihabziani/facial-recognition-Siamese/?tab=readme-ov-file

Scikit-learn. (2024). *scikit-learn : apprentissage automatique en Python — documentation scikit-learn 1.6.1*. https://scikit-learn.org/stable/index.html

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, *abs/1409.1556*. https://arxiv.org/abs/1409.1556

Singh, S., Ahuja, U., Kumar, M., Kumar, K., & Sachdeva, M. (2021). Face mask detection using YOLOv3 and faster R-CNN models: COVID-19 environment. *Multimedia Tools and Applications*, *80*(13), 19753-19768. https://doi.org/10.1007/s11042-021-10711-8

Solomon, E., Woubie, A., & Emiru, E. S. (2023). Deep Learning Based Face Recognition Method using Siamese Network. *ArXiv*, *abs/2312.14001*. https://arxiv.org/abs/2312.14001

Song, C., & Ji, S. (2022). Face Recognition Method Based on Siamese Networks Under Non-Restricted Conditions. *IEEE Access*, *10*, 40432-40444. https://doi.org/10.1109/ACCESS.2022.3167143

Sun, Z., & Tzimiropoulos, G. (2022). Part-based Face Recognition with Vision Transformers. British Machine Vision Conference,

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition,

TensorFlow. (2024). *TensorFlow*. https://www.tensorflow.org/?hl=fr

Ullah, N., Javed, A., Ali Ghazanfar, M., Alsufyani, A., & Bourouis, S. (2022). A novel DeepMaskNet model for face mask detection and masked facial recognition. *Journal of King Saud University - Computer and Information Sciences*, *34*(10, Part B), 9905-9914. https://doi.org/10.1016/j.jksuci.2021.12.017

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. Neural Information Processing Systems,

Venkateswarlu, I. B., Kakarla, J., & Prakash, S. (2020, 3-5 Dec. 2020). Face mask detection using MobileNet and Global Pooling Block. 2020 IEEE 4th Conference on Information & Communication Technology (CICT),

Wang, C., Horby, P. W., Hayden, F. G., & Gao, G. F. (2020). A novel coronavirus outbreak of global health concern. *The lancet*, *395*(10223), 470-473.

Wiskott, L., Fellous, J.-M., Krüger, N., & Von Der Malsburg, C. (2022). Face recognition by elastic bunch graph matching. In *Intelligent biometric techniques in fingerprint and face recognition* (pp. 355-396). Routledge.

Xu, X. F., Zhang, L., Duan, C. D., & Lu, Y. (2020). Research on Inception Module Incorporated Siamese Convolutional Neural Networks to Realize Face Recognition. *IEEE Access*, *8*, 12168-12178. https://doi.org/10.1109/ACCESS.2019.2963211

Yang, G., Feng, W., Jin, J., Lei, Q., Li, X., Gui, G., & Wang, W. (2020, 11-14 Dec. 2020). Face Mask Recognition System with YOLOV5 Based on Image Recognition. 2020 IEEE 6th International Conference on Computer and Communications (ICCC),

YouTube. (2024). *YouTube*. https://www.youtube.com/

Zeng, W., Huang, J., Wen, S., & Fu, Z. (2023). A masked-face detection algorithm based on M-EIOU loss and improved ConvNeXt. *Expert Systems with Applications*, *225*, 120037. https://doi.org/10.1016/j.eswa.2023.120037

Zhang, Y., Wang, X., Shakeel, M. S., Wan, H., & Kang, W. (2022). Learning upper patch attention using dual-branch training strategy for masked face recognition. *Pattern Recognition*, *126*, 108522. https://doi.org/10.1016/j.patcog.2022.108522

Zheng, Y., Yang, C., & Merkulov, A. (2018). Breast cancer screening using convolutional neural network and follow-up digital mammography. Computational Imaging III,