République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
UNIVERSITE Dr. TAHAR MOULAY SAIDA
FACULTE : TECHNOLOGIE
DEPARTEMENT : INFORMATIQUE

MEMOIRE DE MASTER
OPTION : SIC

Thème

Deep learning pour la Détection de Deepfakes

Présenté par :
BENYAHIA Ali
HAFIANE Amine

Encadré par :
Dr BOUARARA Hadj Ahmed

Promotion : Septembre 2020

Résumé

Le but de notre travail est de bénéficier des innovations récentes dans les technologies de l'intélligence artificielle et deep learning pour détecter les fausses images et vidéos (Deepfakes), après avoir étudié les techniques utilisées dans la création de deepfake et les solutions existantes, nous avons proposé notre modèle de deep learning avec les réseaux de neurones convolutionnels où nous ajoutons le concept d'apprentissage résiduel et après faire l'apprentissage et l'optimisation des hyper-paramètres du réseau, l'architecture nous donne des résultats acceptable par rapport aux solutions existantes.
Mots-clés : Détection de Deepfakes ; Apprentissage profond ; Réseau de neurones convolutionelle ; Apprentissage Résiduel ; Classification .

Abstract

The perpose of our work is to benefit from the recents innovations of artificial intelligence and deep learning technologies to detecte fake images and videos (Deepfakes), after studying the techniques used in the creation of deepfake and the solutions existed, we proposed our deep learning model where we add the concept of residual learning, after training the network and tuning the hyper-parameters , the architecture give us accepted results compared to the existing solutions.
Keys-words : Deepfake detection ; Deep learning ; Convolutionel neural network ; Residuel learning ; Classification.

ملخص

الهدف من عملنا هذا هو محاولة الإستفادة من آخر التطورات في مجال الذكاء الإصطناعي والتعلم العميق لكشف الصور و الفيديوهات المزورة أو ما يسمى بالتزييف العميق، بعد دراسة طرق صناعة المحتوى المزور والحلول الموجودة لكشفها، إقترحنا نموذج التعلم العميق الخاص بنا حيث استعمنا الشبكات العصبية التلافيفية و عززناها بالتعلم المتبقي، بعد القيام بتدريب الشبكة وتحسين معاملاتها، تمكنا من الحصول على نتائج مقبولة مقارنة مع الحلول الموجودة.

الكلمات المفتاحية : كشف التزييف العميق، التعلم العميق، الشبكات العصبية التلافيفية، التعلم المتبقي، التصنيف.

Remerciments

Je remercie Allah le Tout puissant qui m'a donne la force et la volonté pour réaliser ce modeste travail.

Je tiens à remercier en premier lieu Dr. Hadj Ahmed Bouarara d'avoir accepté d'être mon encadreur durant cette année de master, et pour la confiance qu'il m'a donné et ses précieux conseils.

Je remercie tous nos enseignants de département d'informatique.

Mes remerciements vont également aux membres de jury d'avoir accepté de juger mon travail.

Je remercie tout les personnes qui m'ont aide durant la préparation de ce mémoire de prés ou de loin.

Je conclurai, en remerciant vivement toute ma famille qui m'a toujours supporte pendant toutes mes longues années d'études.

BENYAHIA Ali

Je suis Redevable à un certain nombre de personnes pour leur soutien pendant que je menais les travaux relatifsà cette thèse.

J'adresse tout d'abord ma profonde reconnaissance à mon directeur de thèse, BOUARARA Hadj Ahmed , pour son soutien continu sa patience, sa motivation et ses connaissances immenses. ses conseils éclairés m'ont aidé tout au long de l'élaboration de cette thèse et qui ont permis de mieux cerner les difficultés de mes recherches.

Je remercie chaleuresement les deux êtres les plus chers au monde : mes parents qui ont bercé ma vie de tendresse, d'amour et d'encouragements et qui ont oeuvré pour ma réussite avec leur indéfectible soutien, leurs précieux conseils, leur constante présence à mes côtés aux sacrifices qu'ils ont consentis pour m'aider à mieux avancer dans la vie.

Mes remerciements vont naturellement à tous les membres du jury qui ont bien voulu prendre le temps de lire ce modeste travail et de l'évaluer.

Merci enfin à mes amis qui se reconnaîtront ici, pour leur aide si précieuse et surtout pour leur disponibilité qui m'a permis de réaliser dans de bonnes conditions ce travail avec le concours de leurs conseils et orientations.

HAFIANE Amine

Á ma mère, Á mon père
Á mon frère et mes sœurs
Á toute ma famille, Á tous mes amis ...

B.Ali

Dédie humblement ce manuscrit à mes parents, mon grand-père, ma grand- mère, mes oncles et mes tantes...

H.Amine

Table des matières

Résumé **ii**

Remerciments **iii**

1 Introduction Générale **1**

1.1 Contexte et problématique 1

1.1.1 Le deepfake 1

1.1.2 Les risques liées au deepfake 1

1.1.3 Apprentissage profond 1

1.2 Objectif de la thèse 2

1.3 Oraganisation de la thèse 2

2 La detection de Deepfakes **3**

2.1 Introduction 3

2.2 C'est quoi le Deepfake 3

2.3 Types de Deepfakes 4

2.4 La création de DFs 4

2.4.1 Deepfakes avec AE 5

2.4.2 Deepfakes avec GAN 6

2.4.3 Outils utilisés pour la création de DFs 7

2.5 Impact de la technologie Deepfakes 7

2.5.1 Les côté négatifs 7

2.5.2 Les côtés positifs 8

2.6 La détection de DFs 8

2.6.1 Datasets pour la détection de DF 8

2.6.2 Méthodes de détection 8

2.6.2.1 Détection des fausses images 9

2.6.2.2 Détection des fausses vidéos 9

2.7 Conclusion 11

3 Apprentissage profond (DL) **12**

3.1 Introduction 12

3.2 L'Intelligence Artificielle (AI) 13

3.2.1 Comprendre l'intelligence artificielle . . . 13
3.2.2 Applications de l'intelligence artificielle . . . 13
3.3 Apprentissage automatique (Machine Learning) . . . 13
3.3.1 Définition . . . 14
3.3.2 Type de ML . . . 14
3.3.2.1 Apprentissage supervisé . . . 14
3.3.2.2 Apprentissage non supervisé . . . 15
3.4 Representation Learning (RL) . . . 15
3.5 Apprentissage Profond (Deep Learning) . . . 16
3.5.1 Histoire de DL . . . 16
3.5.2 Reseaux de neurones artificiels (ANNs) . . . 16
3.5.2.1 Perceptron simple . . . 17
3.5.2.2 Perceptron Multi-class . . . 19
3.5.2.3 Gradient Descent . . . 20
3.5.2.4 Fonctions d'activation . . . 21
3.5.2.5 Perceptron multi-couche (MLP) . . . 23
3.5.2.6 Backpropagation . . . 24
3.5.3 Types d'architecture des ANNs : . . . 25
3.5.3.1 Feed-forward Neural Networks (FNN) . . . 25
3.5.3.2 AutoEncoder (AE) . . . 25
3.5.3.3 Restricted Boltzmann Machines (RBM) . . . 25
3.5.3.4 Deep Belief Networks (DBN) . . . 25
3.5.3.5 Convolution Neural Networks (CNN) . . . 25
3.5.3.6 Recurrent Neural Networks (RNN) . . . 26
3.5.3.7 Generative Adversarial Networks (GAN) . . . 26
3.5.4 Architecture profond . . . 26
3.5.5 Applications de Deep Learning . . . 26
3.5.6 Techniques d'optimisations dans DL . . . 27
3.5.6.1 Stochastic Gradient Descent (SGD) . . . 27
3.5.6.2 Momentum . . . 27
3.5.6.3 Adagrad . . . 28
3.5.6.4 Adadelta . . . 28
3.5.6.5 Adam . . . 28
3.5.7 Techniques de Regularisation . . . 28
3.5.7.1 L2 & L1 regularization . . . 28
3.5.7.2 Dropout . . . 29
3.5.7.3 Data augmentation . . . 29
3.5.7.4 Early stopping . . . 29
3.6 Conclusion . . . 30

4 Contribution **31**
4.1 Introduction . . . 31
4.2 Les réseaux de neurones convolutif (CNN) . . . 31
4.2.1 Couches dans un CNN . . . 32
4.2.2 Couche de convolution (Convolution layer) . . . 32
4.2.3 Couche de pooling (Pooling layer) . . . 33
4.2.4 Couche entièrement connectée (Fully connected layer) . . . 34
4.2.5 Architectures existantes . . . 35
4.3 Deep Residuel Learning . . . 36
4.3.1 Resnet . . . 37
4.3.2 Residual Learning . . . 38
4.4 Notre modele R-CNN . . . 39
4.5 Conclusion . . . 40

5 Implémentation **41**
5.1 Introduction . . . 41
5.2 Outils d'implémentation . . . 41
5.2.1 Les frameworks existantes . . . 41
5.2.1.1 TensorFlow . . . 42
5.2.1.2 Keras . . . 42
5.2.1.3 PyTorch . . . 42
5.2.1.4 Caffe . . . 42
5.2.1.5 Deeplearning4j . . . 43
5.2.2 Environnement d'implémentation . . . 43
5.3 Choix de Dataset . . . 43
5.4 Configuration de classification . . . 44
5.5 Expérimentations . . . 44
5.5.1 Epochs, Batch Size et Iterations . . . 44
5.5.2 Batch size . . . 45
5.5.3 Dropout . . . 45
5.5.4 Utilisation des blocs résiduels . . . 46
5.5.5 Résultats finaux . . . 46
5.5.6 Configuration finale . . . 48
5.6 Comparaisons avec autres architectures . . . 48
5.7 Utilisation de modèle . . . 48
5.7.1 Processus de détection . . . 48
5.7.2 Processus de Création . . . 49
5.8 Conclusion . . . 50

6 Conclusion Générale **51**

6.1 Conclusion 51

6.2 Travaux Futures : 51

Bibliographie **52**

Table des figures

2.1 Video généré par DF(droit :original, gauche :manipulé)[25]. 4
2.2 Principe Deepfake. En haut : les parties d'apprentissage avec l'encodeur partagé. En bas : la partie utilisation où les images de A sont décodées avec le décodeur de B.[1]. 5
2.3 Rôles du générateur et du discriminateur. 6
2.4 Création de Deepfakes avec GAN[5]. 7

3.1 Différence entre programmation traditionnel et machine learning[6] 14
3.2 Diagramme de Venn illustre la relation entre AI, ML, RL et DL[13] 17
3.3 Perception simple[9]. 19
3.4 Perceptron multi-classes[9]. 20
3.5 Representation graphique des fonctions d'activations Sigmoid et Tanh [28]. . 22
3.6 Perceptron Multi-couche[9]. 23
3.7 Etapes de Backpropagation[9]. 24
3.8 Le reseau de neurone standard et aprés l'utilisation du dropout[31]. 29

4.1 Convolution 2D[13]. 33
4.2 Opération de max-pooling. 34
4.3 Couche entièrement connectée. 34
4.4 Bloc de construction d'un CNN typique[32]. 35
4.5 Architecture de LeNet-5[23] 35
4.6 Architecture de AlexNet[21] 36
4.7 Architecture de VGGNet (configuration D, VGG16)[32] 36
4.8 Architecture de GoogleNet[34] 37
4.9 (a) Couche simple (b) Bloc résiduel[17] 37
4.10 Erreur d'apprentissage (à gauche) et erreur de test (à droite) sur CIFAR-10 avec des réseaux simples à 20 et 56 couches.[17] 38
4.11 Blocs proposés simple et avec résiduel. 39
4.12 L'architecture proposé avec et sans résiduel. 40

5.1 La précision et le taux d'erreur d'apprentissage avec diffèrents batch-size. . . 45
5.2 La précision et le taux d'erreur de validation avec différents batch-sizes. . . . 45
5.3 La précision et le taux d'erreur de validation avec diffèrents taux de dropout. 46

5.4 Résultats de validation :model0(simlpe), model1(+ dropout), model2(+blocs résiduels) 46
5.5 Model0 : résultats d'apprentissage et de validation(précision et taux d'erreur) 47
5.6 Model1 : résultats d'apprentissage et de validation(précision et taux d'erreur) 47
5.7 Model2 : résultats d'apprentissage et de validation(précision et taux d'erreur) 48
5.8 Comparaison de précision entre les models proposés et autres architectures célèbres. 49
5.9 Exemple de vidéo créée avec GAN (image gauche représente la vidéo manipulée). 49

Liste des tableaux

2.1 Outils pour créer DFs[27]. 7
2.2 Spécifications des datasets deepfake les plus pertinents de la littérature[10]. . 8

3.1 Étapes majeures de Deep Learning[35]. 18
3.2 Applications de Deep Learning[8]. 27

5.1 Précision et taux d'erreur de model 0. 47
5.2 Précision et taux d'erreur de model 1. 47
5.3 Précision et taux d'erreur de model 2. 48
5.4 Les paramètres de modèle choisis. 48

Table des sigles et acronymes

AE	**A**uto**E**ncoder
AF	**A**ctivation **F**unction
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CFFN	**C**ommon **F**ake **F**eature **N**etwork
CNN	**C**onvolution **N**eural **N**etworks
DBN	**D**eep **B**elief **N**etworks
DF	**D**eep**F**akes
DFDC	**D**eep**F**ake **D**etection **C**hallenge
DL	**D**eep **L**earning
DNN	**D**eep **N**eural **N**etwork
DRL	**D**eep **R**einforcement **L**earning
FNN	**F**eed-forward **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etworks
GD	**G**radient **D**escent
GPU	**G**raphics **P**rocessing **U**nit
IPFS	**I**nter**P**lanetary **F**ile **S**ystem
LMS	**L**east **M**ean **S**quare
LSTM	**L**ong **S**hort-**T**erm **M**emory
MCP	**M**c**C**ulloch-**P**itts
ML	**M**achine **L**earning
MLP	**M**ulti-**L**ayer **P**erceptron
MSE	**M**ean **S**quare **E**rror
NLP	**N**euro-**L**inguistic **P**rogramming
PRNU	**P**hoto **R**esponse **N**on **U**niformity
RBM	**R**estricted **B**oltzmann **M**achines
RCN	**R**ecurrent **C**onvolutional **N**etwork
ReLU	**R**ectified **L**inear **U**nit
RF	**R**andom **F**orest
RL	**R**epresentation **L**earning
RNN	**R**ecurrent **N**eural **N**etworks
SGD	**S**tochastic **G**radient **D**escent
SVM	**S**upport **V**ector **M**achine

Chapitre 1

Introduction Générale

1.1 Contexte et problématique

Pour expliquer le contexte général de notre travail et la problématique liée, il peut être utile de décomposer le sujet de notre thèse «*Deep learning pour la détection de Deepfakes*» en trois partie :

1.1.1 Le deepfake

Depuis sa première apparition en 2018, deepfake technologie a évolué d'un expérimentation des amateurs à un outil efficace et potentiellement dangereux. cette technique offre la possibilité d'échanger un visage à un autre dans une image ou une vidéo. La technologie utilisée pour créer un tel contenu numérique est rapidement devenue accessible aux publiques, avec l'utilisation de l'intelligence artificielle ou plus précisément le deep learning , qui produisent des images et des sons fabriqués qui semblent réels.

1.1.2 Les risques liées au deepfake

L'impact plus insidieux de deepfakes, ainsi que d'autres médias synthétiques, est de créer une société sans confiance, où les gens ne peuvent pas, ou ne sont plus la peine de distinguer la vérité du mensonge. Et lorsque la confiance est érodée, il est plus facile de soulever des doutes sur des événements spécifiques.

Au fur et à mesure que la technologie devient plus accessible, les deepfakes pourraient causer des problèmes aux tribunaux, où de faux événements pourraient être consignés comme preuves. Ils posent également un risque pour la sécurité personnelle : les deepfakes peuvent imiter les données biométriques et peuvent potentiellement tromper les systèmes qui reposent sur la reconnaissance du visage, de la voix, des veines ou de la marche. Le potentiel d'escroqueries est clair.

1.1.3 Apprentissage profond

La croissance et la disponibilité explosives des données et les progrès remarquables des technologies matérielles ont conduit à l'apparition de nouvelle branche de l'apprentissage automatique connu sous le nom de Deep learning ou l'apprentissage profond basé sur les réseaux de neurones artificielles. L'apprentissage profond évite la limite des algorithmes d'apprentissages traditionnels lors de l'extraction des features surtout lorsque en traitant des données non structurés comme les sons, les images et les vidéos, avec leur capacité d'automatiser la tâche de l'extraction des features et l'utilisation des réseaux de nuerons artificielles donnes des résultats très intéressant surtout dans des domaines comme la vision

par ordinateur et le traitement de langage naturel. Et comme l'apprentissage profond est utilisé pour la création des deepfakes, il peut être une outil efficace pour les détecter.

1.2 Objectif de la thèse

Notre objectif est d'utiliser l'apprentissage profond ou plus précisément les réseaux de neurones convolutif avec des blocs résiduelles pour distinguer si les images (vidéos) sont réelles ou synthétiques et donc lutter contre la mauvaise utilisation de deepfake.

1.3 Oraganisation de la thèse

Le reste de ce travail est structuré en 5 chapitres :

- **Chapitre 2 :** Explique s'est quoi le deepfake, leur utilisation, la theorie derrière cette technologie, les travaux existant pour le détecter.
- **Chapitre 3 :** Couvre ce qu'on appelle le deep learning, leur origine et le domaine d'intelligence artificiel et l'apprentissage automatique, le réseau de neurones, le fonctionnement de l'apprentissage profond et les différents models de deep learning.
- **Chapitre 4 :** Explique notre contribution pour détecter le deepfake utilisant le reseau de neurones convolutif, et comment fonctionner.
- **Chapitre 5 :**Traite le choix de plateforme pour l'implémentation de notre modèle, les expérimentations, l'évaluation de performance et l'optimisation de notre modèle.
- **Chapitre 6 :** Discute la conclusion et les futures travaux pour améliorer notre modèle.

Chapitre 2

La detection de Deepfakes

L'imagination est plus importante que la connaissance.
Car la connaissance est limitée à tout ce que nous savons et comprenons maintenant,
Tandis que l'imagination embrasse le monde entier,
Et tout ce qu'il y aura à connaître et à comprendre.

Albert Einstein - The World As I See It

2.1 Introduction

La détection de contenu visuel manipulé est un axe de recherche d'intérêt intense dans le milieu universitaire et l'industrie, et un sujet pressant de conversation dans la société plus large. Au cours des deux dernières années, les vidéos de haute qualité contenant des manipulations faciales (communément appelés *Deepfakes*), est devenu particulièrement important. Bien qu'il existe des applications bénignes et même humoristiques des vidéos deepfake, lorsqu'elles sont créées avec une intention malveillante, elles ont le potentiel d'intimider et de harceler des individus, ou de propager de la désinformation qui peut nuire aux élections ou aux systèmes financiers.

Dans ce chapitre, on va expliquer le deepfake (Df) et les techniques utilisé pour la creation de DF, les différents types de Df, aprés on va passer a la détection des videos et images crées par DF et les traveaux existants.

2.2 C'est quoi le Deepfake

Ces dernières années, AI DeepFakes ou *Deepfaking*, fait son apparition sur les réseaux sociaux. La technologie Deepfake peut facilement assembler n'importe qui dans le monde dans une vidéo ou une photo à laquelle ils n'ont jamais réellement participé. De telles capacités existent depuis des décennies, c'est ainsi que le défunt acteur Paul Walker a été ressuscité pour (Fast & Furious 7). Mais il necessite des studios entiers remplis d'experts pour créer ces effets. Désormais, les technologies Deepfake ou l'utilisation d'apprentissage automatique et deep learning peuvent synthétiser des images et des vidéos beaucoup plus rapidement.

FIGURE 2.1 – Video généré par DF(droit :original, gauche :manipulé)[25].

Le terme deepfakes origine à la fin de 2017 d'un utilisateur de Reddit nommé *deepfakes* a commencé à appliquer le deep learning pour fabriquer de fausses vidéos de célébrités et des politiciens, Reddit a interdis 'deepfakes', mais cela s'est avéré trop tard, la technologie s'était déjà répandue, c'est là que les choses deviennent un peu plus intéressantes. Deepfake (issu de deeplearning et fake) est une technique qui peut superposer des images de visage d'une personne cible à une vidéo d'une personne source pour créer une vidéo de la personne cible faisant ou disant des choses que la personne source fait. La force de DeepFake provient des réseaux de neurones profonds formés sur des images faciales qui émuler automatiquement des expressions faciales de la source à la cible.

2.3 Types de Deepfakes

Différentes formes de Deepfake existe, parmi lesquelles, les DFs audios qui imitent la voix d'une personne ciblée à partir d'échantillons de sa voix, et donc permettent de lui faire prononcer un texte donné en entréé. Le face-swapping, qui remplace le visage d'une personne dans une vidéo par celui d'autre personne ciblée à partir de ses images. Une autre forme, le Deepfake lip-synching, qui adapte les mouvements du visage d'une personne ciblée à partir d'un fichier audio d'une autre personne. Il permet de lui faire prononcer le discours contenu dans le fichier audio en question sans qu'elle ne l'ait prononcé. Le Deepfake puppetry, qui génère une vidéo d'une personne ciblée à partir d'une vidéo d'acteur fournie en entrée. Il est ainsi possible de créer une vidéo dans laquelle la cible reproduit un discours joué par l'acteur. Un exemple célèbre de cette technique est une vidéo de Barack Obama énonçant un discours simulé par Jordan Peele, dans laquelle la gestuelle de l'ancien président est reproduite de manière extrêmement réaliste.

2.4 La création de DFs

Les smartphones et la croissance sophistiquée des réseaux sociaux ont conduit à une quantité gigantesque de nouveaux contenus d'objets numériques. Cette utilisation énorme d'images numériques a été suivie d'une augmentation des techniques de modifier le contenu

d'images. Jusqu'à récemment, ces techniques étaient hors de portée de la plupart des utilisateurs et nécessitaient une expertise de haut niveau en vision par ordinateur. les progrès récents de l'apprentissage automatique et l'accessibilité aux données à grand volume, ces limites ont progressivement disparu. En conséquence, le temps de fabrication et de manipulation des contenus numériques a considérablement diminué, permettant même aux utilisateurs amateurs de modifier les contenus à leur guise.

Les Deepfakes sont devenus populaires en raison de la qualité des vidéos falsifiées et de la capacité facile à utiliser de leurs applications pour un large nombre d'utilisateurs du professionnel au novice. Ces applications sont principalement développées sur la base de techniques d'apprentissage profond (DL). Le Deep Learning est bien connu pour sa capacité à représenter des données complexes et de grande dimension. Une variante des réseaux profonds avec cette capacité est les auto-encodeurs (AE) profonds, qui ont été largement appliqués pour la réduction de dimensionnalité et la compression d'image.

2.4.1 Deepfakes avec AE

L'idée centrale réside dans l'apprentissage parallèle de deux autoencodeurs (AE). Traditionnellement, un AE désigne le chaînage d'un réseau codeur et d'un réseau décodeur. Le but du codeur est d'effectuer une réduction de dimension en codant les données de la couche d'entrée en un nombre réduit de variables. Le but du décodeur est alors d'utiliser ces variables pour produire une approximation de l'entrée d'origine. La phase d'optimisation est effectuée en comparant l'entrée et son approximation produite et pénaliser la différence entre les deux. si on prend par exemple une image avec les dimensions $64 \times 64 \times 3 = 12288)$ variables, l'encodeur réduit les dimensions vers 1024 variables puis génère une image de la même taille que l'entrée avec le décodeur.

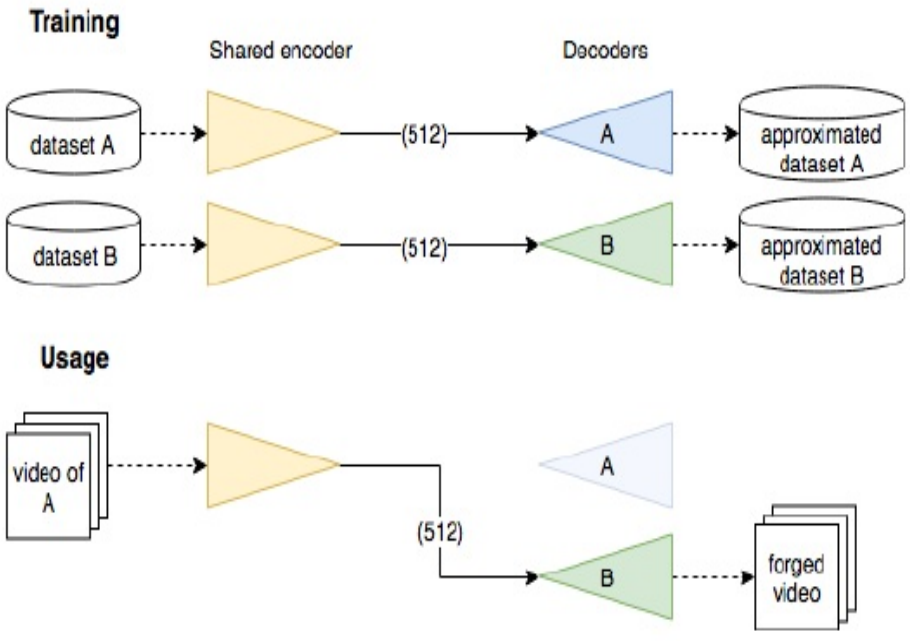

FIGURE 2.2 – Principe Deepfake. En haut : les parties d'apprentissage avec l'encodeur partagé. En bas : la partie utilisation où les images de A sont décodées avec le décodeur de B.[1].

Le processus pour générer des images Deepfake consiste à rassembler les visages de deux personnes différentes A et B, puis à apprentis un auto-encodeur E_A pour reconstruire

les visages de A à partir de l'ensemble de données d'images faciales de A, et un auto-encodeur E_B à reconstruire les visages de B à partir de l'ensemble de données d'images faciales de B. L'idée consiste a partager les poids de la partie encodage des deux auto-encodeurs E_A et E_B, mais en gardant leur décodeur respectif séparé. Une fois l'apprentissage effectuée, toute image contenant un visage de A peut être encodée via cet encodeur partagé mais décodée avec le décodeur de E_B. Ce principe est illustré dans la figure 2.2.
En pratique, les résultats sont impressionnants, ce qui explique la popularité de la technique. La dernière étape consiste à prendre la vidéo cible, extraire et aligner le visage cible de chaque image, utiliser l'encodeur automatique modifié pour générer un autre visage avec le même éclairage et la même expression, puis le fusionner à nouveau dans la vidéo.

2.4.2 Deepfakes avec GAN

L'idée de réseaux adverses génératifs (en anglais generative adversarial networks ou GANs) a été proposée par Ian GoodFellow en 2014[14]. Cette technique a permis aux chercheurs de créer des photos de visages de personnes réalistes mais entièrement générées par ordinateur. Ils ont également permis la création de vidéos controversées DFs. En fait, les GAN peuvent être utilisés pour imiter toute distribution de données (image, texte, son, etc.).
Le principe est comme un jeu à deux joueurs : un réseau de neurones appelé générateur (generator) et un réseau de neurones appelé discriminateur (discriminator). Le générateur essaie de tromper le discriminateur en générant des images réelles tandis que le discriminateur essaie de distinguer les images réelles des images fausses. Par conséquent, nous comprenons le terme *adversarial*.

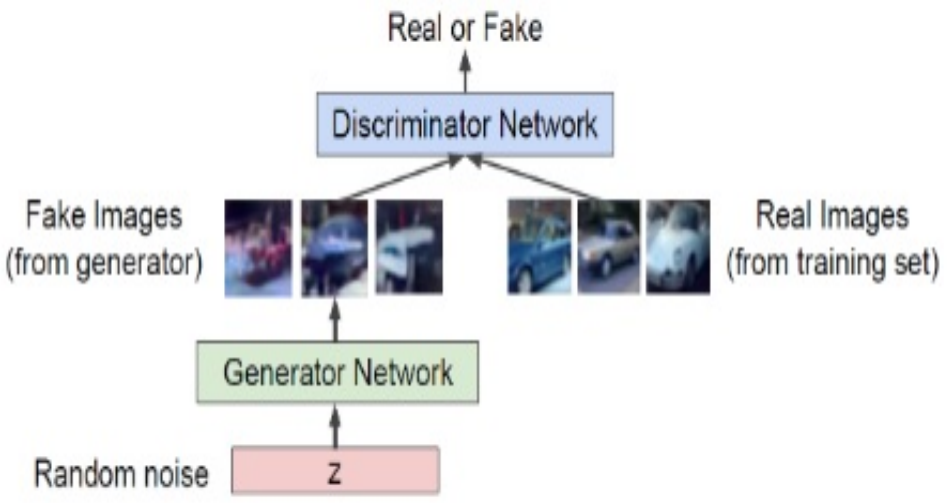

FIGURE 2.3 – Rôles du générateur et du discriminateur.

Au cours de l'apprentissage, le générateur devient progressivement mieux à créer des images qui semble réel, tandis que le discriminateur est mieux les distinguer. Le processus a atteint son équilibre lorsque le discriminateur ne peut plus distinguer les images réelles ou faux. Après cette phase d'apprentissage, nous n'avons besoin que du générateur pour produire de nouvelles (fausses) données réalistes.

Pour la creation de DFs, Une version improuvé de l'auto-encodeur, L'AE joue le role de générateur, et le discriminateur est donc classifie les images générées par le générateur comme original ou fake, la figure 2.4 ullistre le fonctionnement de l'architecture.

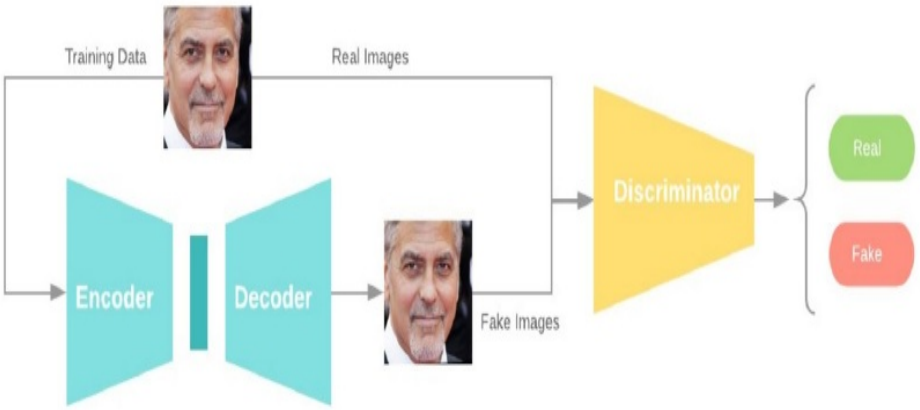

FIGURE 2.4 – Création de Deepfakes avec GAN[5].

2.4.3 Outils utilisés pour la création de DFs

Les outils populaires de creation de deepfakes et leurs fonctionnalités résumés dans le tableau2.1 suivant :

Outils	Liens	Caractéristiques
Faceswap	https ://github.com/deepfakes/faceswap	Utilisation de deux paires encodeur-décodeur. Les paramètres de l'encodeur sont partagés.
Faceswap-GAN	https ://github.com/shaoanlu/faceswap-GAN	Adversarial loss et perceptual loss (VGGface) sont ajoutés à l'architecture auto-encodeur.
DeepFaceLab	https ://github.com/iperov/DeepFaceLab	Développez à partir du modèle Faceswap avec de nouveaux modèles, par exemple H64, H128, LIAEF128, SAE. Prend en charge plusieurs modes d'extraction de visage, par exemple S3FD, MTCNN, dlib ou manuel.
DFaker	https ://github.com/dfaker/df	La fonction de perte DSSIM est utilisée pour reconstruire le visage. Implémenté basé sur la bibliothèque Keras.
DeepFake-tf	https ://github.com/StromWine/DeepFake-tf	Similaire à DFaker mais implémenté avec Tensorflow.

TABLEAU 2.1 – Outils pour créer DFs[27].

2.5 Impact de la technologie Deepfakes

Comme toute autre technologie, le DF a des utilisations positifs et négatifs :

2.5.1 Les côté négatifs

Étant donné que des personnalités publiques telles que des célébrités et des politiciens peuvent avoir un grand nombre de vidéos et d'images disponibles en ligne, elles sont les premières cibles des deepfakes. Les Deepfakes ont été utilisés pour échanger des visages de célébrités ou de politiciens dans des images et des vidéos non éthiques, Il menace la sécurité mondiale lorsque des méthodes de DFs peuvent être utilisées pour créer des vidéos de dirigeants mondiaux avec de faux discours à des fins de falsification.

Les deepfakes peuvent donc être abusés pour provoquer des tensions politiques ou religieuses entre les pays, tromper le public et affecter les résultats des campagnes électorales, ou créer le chaos sur les marchés financiers en créant de fausses nouvelles. Il peut même être utilisé pour générer de fausses images satellites de la Terre pour contenir des objets qui n'existent pas vraiment pour confondre les analystes militaires, par exemple, créer un faux pont sur une rivière bien qu'il n'y ait pas un tel pont en réalité. Cela peut induire en erreur une troupe qui a été guidée pour traverser le pont dans une bataille.

2.5.2 Les côtés positifs

La spéculation sur les utilisations potentielles des deepfakes est souvent négative, Mais il existe également une utilisation positive des deepfakes tels que la création de voix de ceux qui ont perdu le leur [27]. De grandes opportunités s'ouvrent dans l'industrie cinématographique. La technologie peut permettre de faire revivre un acteur mort, par exemple. Il est également possible de doubler des films dans différentes langues de manière plus réaliste en faisant correspondre les mouvements de la bouche des acteurs avec le dialogue oral réel. Il existe également toute une gamme de nouvelles possibilités dans le domaine des films dynamiques, qui permettraient à une personne de choisir certains acteurs ou aux téléspectateurs de se lancer dans certains rôles. Un concept similaire pourrait également être appliqué à l'industrie de la publicité.

2.6 La détection de DFs

Les deepfakes sont de plus en plus préjudiciables à la vie privée, à la sécurité de la société et à la démocratie. Des méthodes de détection des deepfakes ont été proposées dès l'introduction de cette menace. La détection de Deepfake est considérée comme un problème de classification binaire où les classificateurs sont utilisés pour classer entre les vidéos authentiques et les vidéos falsifiées. Ce type de méthode nécessite une grande nombres de vidéos réelles et fausses pour entrainer les modèles de classification.

2.6.1 Datasets pour la détection de DF

Un certain nombre dataset contenant des manipulations faciales ont été récemment publiés dans le but d'entrainer et d'évaluer la détection et la classification de DF. Ceux-ci incluent le Celeb-DF, UADFV, DeepFake-TIMIT, le dataset VTD, FaceForensics et son successeur FaceForensics ++. Le tableau 2.2 résume les spécifications des datasets les plus pertinents dans l'espace Deepfake.

Dataset	Ratio fausse : original	Total vidéos	Source	Consentement des participants
Celeb-DF	1 : 0.51	1203	YouTube	N
FaceForensics	1 : 1.00	2008	YouTube	N
FaceForensics++	1 : 0.25	5000	YouTube	N
DeepFakeDetection (partie de FaceForensics ++)	1 : 0.12	3363	Acteurs	Y
DFDC	1 : 0.28	5214	Acteurs	Y

TABLEAU 2.2 – Spécifications des datasets deepfake les plus pertinents de la littérature[10].

2.6.2 Méthodes de détection

Des vidéos de dataset VidTIMIT[7] ont été utilisées pour générer des vidéos deepfake de faible et de haute qualité, qui peuvent imiter efficacement les expressions faciales, les mouvements de la bouche et les clignements des yeux. Ces vidéos ont ensuite été utilisées pour tester différentes méthodes de détection de deepfake.
Les résultats des tests différentes méthodes de détection de deepfake montrent que les systèmes de reconnaissance faciale populaires sont incapables de détecter efficacement les deepfakes. D'autres méthodes telles que les approches de synchronisation labiale et les métriques de qualité d'image avec support vector machine (SVM) produisent un taux d'erreur

très élevé lorsqu'elles sont appliquées pour détecter les vidéos deepfake de cet ensemble de données nouvellement produit. Cela indique qu'il est nécessaire de développer des autres méthodes plus robustes capables de distinguer entre les deepfakes et les authentiques[27].

L'auteur dans [27] regroupe les méthodes de détection de DFs en deux grandes catégories : les méthodes de détection de fausses images et les méthodes de détection de fausses vidéos.

2.6.2.1 Détection des fausses images

L'échange de visage a un certain nombre d'applications convaincantes dans la composition vidéo, la transfiguration dans les portraits, et en particulier dans la protection de l'identité. Cependant, c'est aussi l'une des techniques que les cyber-attaquants utilisent pour pénétrer les systèmes d'identification ou d'authentification pour obtenir un accès illégitime. L'utilisation de l'apprentissage profond (CNN et GAN) a rendu l'échange de visage plus difficiles pour les modèles forensics car elles peuvent préserver la pose, l'expression faciale et l'éclairage des photographies.

Zhang et al[37] ont utilisé la méthode du sac de mots pour extraire un ensemble de caractéristiques compactes et l'ont introduit dans divers classificateurs tels que SVM, forêt aléatoire (RF) et perceptrons multicouches (MLP) pour la discrimination entre les images avec de visage échangées de l'authentique. Parmi les images générées par le deep learning, celles synthétisées par les modèles GAN sont probablement les plus difficiles à détecter.
Hsu et al[18] a introduit une méthode de DL en deux phases pour la détection d'images DF. La première phase est un extraction de features basé sur le réseau CFFN. Les caractéristiques discriminantes entre les images fausses et réelles sont ensuite introduites dans la deuxième phase, qui est un petit CNN concaténé à la dernière couche convolutionnelle de CFFN pour distinguer les images trompeuses des images authentiques.

2.6.2.2 Détection des fausses vidéos

Les méthodes de détection des vidéo DFs peut être classé en deux groupes : les méthodes qui utilisent des caractéristiques temporelles et celles qui explorent les artefacts visuels dans les images.

2.6.2.2.1 Features temporelles entre les images de vidéo

Avec l'observation que la cohérence temporelle n'est pas appliquée efficacement dans le processus de synthèse des deepfakes, Guera et Delp[15] ont souligné que les vidéos deepfake contiennent des incohérences intra-image et des incohérences temporelles entre les images. Ils ont ensuite proposé une méthode qui utilise CNN et LSTM pour détecter les vidéos deepfake. CNN est utilisé pour extraire les features au niveau de l'image, qui sont ensuite introduites dans le LSTM pour créer un descripteur de séquence temporelle. Un réseau entièrement connecté est utilisé par la suite pour classer les vidéos truquées de réels basé sur le descripteur de séquence.

D'autre part, l'utilisation d'un signal physiologique, le clignement des yeux, pour détecter les deepfakes a été proposée dans[24] sur la base de l'observation qu'une personne en deepfakes clignote beaucoup moins fréquemment que dans les vidéos non altérées.

2.6.2.2.2 Les artefacts visuels dans les image de vidéo

les méthodes utilisant des modèles temporels à travers des images vidéo sont principalement basées sur des modèles de réseau récurrents profonds pour détecter des vidéos deepfake. Une autre approche décompose simplement les vidéos en images et explore les artefacts visuels dans les images pour obtenir des features discriminantes. Ces features sont ensuite distribuées dans un classificateur profond ou non profond(shallow) pour différencier les vidéos fausses des vidéos authentiques.

Classificateurs profonds Les vidéos Deepfake sont normalement créées avec des résolutions limitées, qui nécessitent une approche de déformation de visage (mise à l'échelle, rotation et cisaillement) pour correspondre à la configuration des vidéos originales. En raison de l'incohérence de résolution entre la zone de visage déformée et le contexte environnant, ce processus laisse des artefacts qui peuvent être détectés par des modèles CNN.
Une méthode d'apprentissage profond pour détecter les deepfakes basées sur les artefacts observés a été proposée dans[25].

Classificateurs shallow Les méthodes de détection Deepfake reposent principalement sur les artefacts ou l'incohérence des features intrinsèques entre les images ou les vidéos fausses et réelles.
Une méthode basée sur des features visuelles des yeux, des dents et des contours du visage a été étudiée dans[26]. Les artefacts visuels proviennent du manque de cohérence globale, d'une estimation erronée ou imprécise de l'éclairage incident, ou d'une estimation imprécise de la géométrie sous-jacente. Pour la détection des deepfakes, les reflets et les détails manquants dans les zones des yeux et des dents sont exploités ainsi que les features de texture extraites de la région de visage en fonction des landmarks faciaux. Après l'extraction des features, deux classificateurs, y compris la régression logistique et un réseau de neurones simple sont utilisés pour classer les deepfakes de vidéos réelles.
L'utilisation de photo response non uniformity (PRNU) a été proposée dans[20], PRNU est un motif de bruit résultant d'un défaut de fabrication de capteurs sensibles à la lumière d'un appareil photo numérique. PRNU est différent pour chaque appareil photo numérique et souvent considéré comme l'empreinte digitale des images numériques. Les vidéos sont converties en images, qui sont recadrées dans la région faciale, ensuite séparées séquentiellement en huit groupes où un motif PRNU moyen est calculé pour chaque groupe. Les scores de cross corrélation normalisés sont calculés pour les comparaisons des motifs PRNU parmi ces groupes.
Hasan et Salah[16] ont proposé l'utilisation de la blockchain et des contrats intelligents pour aider les utilisateurs à détecter les vidéos deepfake en supposant que les vidéos ne sont réelles que lorsque leurs sources sont traçables. Chaque vidéo est associée à un contrat intelligent qui est lié à sa vidéo parent et chaque vidéo parente a un lien vers son fils dans une structure hiérarchique. Grâce à cette chaîne, les utilisateurs peuvent remonter de manière crédible au contrat intelligent d'origine associé à la vidéo originale même si la vidéo a été copiée plusieurs fois. Un attribut important du contrat intelligent est le hachage unique du système de fichiers interplanétaire (IPFS), qui est utilisé pour stocker la vidéo et ses métadonnées de manière décentralisée et adressable au contenu. Cette approche est générique et peut être étendue à d'autres types de contenu numérique tels que des images, des audios et des manuscrits.

2.7 Conclusion

La qualité de Deepfakes est en croissance permanente et les performances des méthodes de détection doivent être améliorées en conséquence. L'inspiration est que ce qu'est créé par l'AI peut aussi être corrigé par l'AI. Dans le chapitre suivant, on va parler de l'apprentissage profond (DL) qui est la base de la technologie DF, et peut être la solution pour la détecter.

Chapitre 3

Apprentissage profond (DL)

Si vous ne voulez pas qu'on vous oublie, le jour où vous serez mort et pourri,
écrivez des choses qui valent la peine d'être lues
ou faites des choses qui valent la peine d'être écrites.

Benjamin Franklin - The Way to Wealth

3.1 Introduction

Depuis longtemps, les inventeurs rêvent de créer des machines pensent comme l'être humaine. Aujourd'hui, l'intelligence artificielle (AI) est un domaine florissant avec de nombreuses applications pratiques et des sujets de recherche actifs. Nous nous tournons vers des logiciels intelligents pour automatiser le travail de routine, comprendre la parole ou les images, faire des diagnostics en médecine et soutenir la recherche scientifique fondamentale.

Aux débuts de l'intelligence artificielle, le domaine a rapidement abordé et résolu des problèmes qui sont intellectuellement difficiles pour les êtres humains mais relativement simples pour les problèmes informatiques qui peuvent être décrits par une liste de règles mathématiques formelles. Le véritable défi à l'intelligence artificielle est de résoudre des tâches qui sont faciles pour l'humaine à effectuer, mais difficile de les décrire formellement, des problèmes que nous résolvons de façon intuitive, qui se sentent automatique, comme la reconnaissance des paroles ou des objets dans les images. La solution de ces problèmes intuitifs est traité dans ce chapitre. Cette solution est de permettre aux ordinateurs d'apprendre de l'expérience et de comprendre le monde en termes de hiérarchie des concepts, chaque concept défini en termes de sa relation avec des concepts plus simples. En rassemblant les connaissances de l'expérience, cette approche permet d'éviter la nécessité pour les opérateurs humains de préciser formellement toutes les connaissances que la machine les besoins. L'hiérarchie des concepts permet à l'ordinateur d'apprendre des concepts complexes en les construisant à partir de concepts plus simples. Si nous dessinons un graphique montrant comment ces concepts sont construits les uns sur les autres, le graphique est profond, avec de nombreuses couches. Pour cette raison, nous appelons cette approche de l'apprentissage profond de l'AI [13].

3.2 L'Intelligence Artificielle (AI)

L'intelligence artificielle (AI) fait référence à la simulation de l'intelligence humaine dans les machines qui sont programmés pour penser comme les humains et imiter leurs actions. Le terme peut également être appliqué à toute machine qui présente des traits associés à un esprit humain tels que l'apprentissage et la résolution de problèmes.

3.2.1 Comprendre l'intelligence artificielle

L'intelligence artificielle est basée sur le principe selon lequel l'intelligence humaine peut être définie de manière à ce qu'une machine puisse facilement l'imiter et exécuter des tâches, des plus simples à celles qui sont encore plus complexes. Les objectifs de l'intelligence artificielle comprennent l'apprentissage, le raisonnement et la perception. Avec les progrès technologiques, des critères précédents qui ont défini l'intelligence artificielle deviennent obsolètes. Par exemple, les machines qui calculent les fonctions de base ou reconnaître du texte grâce à la reconnaissance des caractères optimal ne sont plus considérés pour incarner l'intelligence artificielle, puisque cette fonction est maintenant pris pour acquis en tant que fonction informatique inhérente. L'AI évolue de façon continue au profit de nombreuses industries différentes. Les machines sont reliées selon une approche interdisciplinaire basée sur les mathématiques, l'informatique, la linguistique, la psychologie, etc.

3.2.2 Applications de l'intelligence artificielle

L'AI est utilisée partout par les grandes organisations pour simplifier la vie d'un utilisateur final. Les utilisations de l'intelligence artificielle serait largement partie de la catégorie de traitement de données, qui comprennent les éléments suivants :

- Recherche dans les données et optimisation de la recherche pour donner les résultats les plus pertinents.
- Chaînes logique pour le if-then raisonnement, qui peuvent être appliquées pour exécuter une série de commandes en fonction des paramètres.
- Détection de motif pour identifier les motifs significatifs dans un grand ensemble de données pour un perspective unique.
- Modèles probabilistes appliqués pour prédire les résultats futurs.

3.3 Apprentissage automatique (Machine Learning)

Auparavant, plusieurs des projets d'AI ont cherché à être des connaissances implémenter en utilisant des règles d'inférence logiques, ce qui est connu sous le nom d'approche de base de connaissances en intelligence artificielle. Mais malheureusement, aucun de ces projets menés à grand succès, car il est impossible à un opérateur humain de définir toutes les règles manuellement pour la machine ou pour connaître tous les cas possibles pour un travail particulier parce qu'il peut être quelque chose dans le cas du monde réel. Ces difficultés rencontrées par les anciens systèmes d'AI suggèrent la nécessité de pouvoir acquérir leurs propres connaissances, en extrayant des modèles à partir des données brutes. Cette capacité est connue sous le nom de Machine Learning[13].

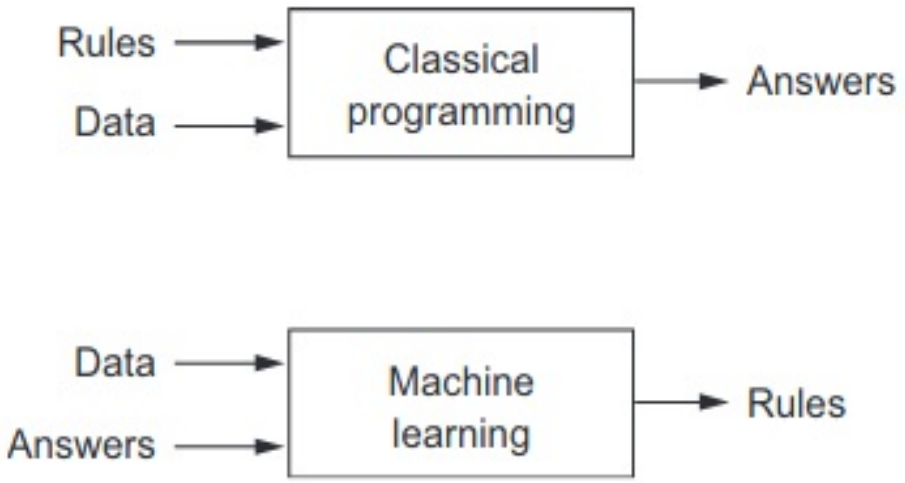

FIGURE 3.1 – Différence entre programmation traditionnel et machine learning[6]

3.3.1 Définition

Machine Learning ou l'apprentissage de la machine est un sous-ensemble de l'intelligence artificielle (AI) qui définit un des principes de base de l'intelligence artificielle - la capacité d'apprentissage à partir de l'expérience, plutôt que des instructions.

3.3.2 Type de ML

Tous les modèles d'apprentissage machine sont classés comme étant soit supervisé ou non-supervisé.Si le modèle est un modèle supervisé, il est ensuite sous-catégorisé en tant que modèle de régression ou de classification. Nous allons voir ci-dessous ce que ces termes signifient et les modèles correspondants qui entrent dans chaque catégorie.

3.3.2.1 Apprentissage supervisé

L'apprentissage supervisé implique l'apprentissage d'une fonction qui fait correspondre une entrée à une sortie sur la base des exemples des paires d'entrées-sorties. Par exemple, avec un ensemble de données de deux variables, l'âge (entrée) et la hauteur (sortie), on mettre en œuvre un modèle d'apprentissage supervisé pour prédire la hauteur d'une personne en fonction de leur âge.
Deux types d'apprentissage supervisé :

3.3.2.1.1 Régression : Dans les modèles de régression, la sortie est continue. Voici quelques-uns des types de modèles de régression les plus courants.

- **Régression linéaire :** L'idée de la régression linéaire consiste simplement à trouver une ligne qui correspond le mieux aux données.
- **Arbres de décision :** arbre de décision utilise la représentation arborescente pour résoudre le problème dans lequel chaque nœud feuille correspond à une classe et les attributs sont représentés sur le nœud interne de l'arbre.

- **Les forêts aléatoires :** ensemble technique d'apprentissage qui s'appuie sur des arbres de décision.
- **Réseau de neurones :** est un modèle à plusieurs couches de nœuds inter-connectées inspiré du cerveau humain.

3.3.2.1.2 Classification : Dans les modèles de classification, la sortie est discrète. Les plus courants sont :

- **La régression logistique :** est similaire à la régression linéaire, mais est utilisé pour modéliser la probabilité d'un nombre fini de résultats, typiquement deux.
- **Machine à vecteur de support (SVM) :** se trouve un hyperplan ou une frontière entre les deux classes de données qui maximise la marge entre les deux classes.
- Naive Bayes : est un autre modèle populaire utilisé. L'idée derrière ce modèle est entraînée par le théorème de Bayes.
- **Arbre de décision, Forêt aléatoire, Réseau de neurones :** même concepts que dans la régression. La seule différence est que la sortie est discrète plutôt que continue.

3.3.2.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé est utilisé pour tirer des inférences et trouver des modèles à partir des données d'entrée sans référence aux résultats étiquetés. Deux méthodes principales utilisées dans l'apprentissage non supervisé incluent le clustering et la réduction de la dimensionnalité.

3.3.2.2.1 Clustering : Le clustering est une technique non supervisée qui implique le regroupement ou le clustering de points de données. Il est fréquemment utilisé pour la segmentation des clients, la détection des fraudes et la classification des documents. Les techniques de clustering courantes incluent k-means, le clustering hiérarchique, mean shift et le clustering basé sur la densité.

3.3.2.2.2 La réduction de la dimensionnalité : La réduction de la dimensionnalité est le processus de réduction du nombre de variables aléatoires considérées en obtenant un ensemble de variables principales. La plupart des techniques de réduction de dimensionnalité peuvent être catégorisées en tant qu'élimination ou extraction de features[13].

3.4 Representation Learning (RL)

Les performances des algorithmes ML dépendent de la représentation des données fournies. Chaque élément d'information inclus dans la représentation est connu sous le nom de features et ces algorithmes apprennent à utiliser ces features pour extraire des modèles ou obtenir des connaissances. Parfois, il est difficile de savoir quelles fonctionnalités doivent être extraites. Par exemple, supposons que nous voulions détecter des voitures à partir d'une image, nous pourrions maintenant utiliser la présence de la roue comme une feature. Mais il est difficile de décrire à quoi ressemble une roue en termes de valeurs de pixels. Une solution à ce problème consiste à utiliser ML non seulement pour découvrir la sortie de ces features (représentation), mais les features elles-mêmes. Cette approche est connue sous le

nom representation learning. Encore une fois, il est préférable que l'algorithme apprenne les features par lui-même avec une intervention humaine minimale. [13].

Un exemple d'un algorithme de RL est l'auto-encoder qui représente est la combinaison d'une fonction de codeur qui convertit les données d'entrée en une représentation différente et une fonction de décodeur qui reconvertit la nouvelle représentation dans le format d'origine. Les AE sont apprennés pour conserver le plus d'informations possible lorsqu'une entrée est exécutée via l'encodeur puis le décodeur, mais ils sont également apprennes pour que la nouvelle représentation ait diverses propriétés intéressantes[13].

Lors de la conception de ces algorithmes, notre objectif est généralement de séparer les facteurs de variation. Maintenant, ces facteurs ne sont pas toujours directement observés, ils peuvent également être des facteurs non observés qui peuvent affecter notre algorithme. Maintenant, bien sûr, il peut être difficile d'extraire des caractéristiques abstraites de haut niveau à partir de données brutes comme l'accent du locuteur en cas de reconnaissance vocale, car elles ne peuvent être identifiées que par une compréhension sophistiquée des données au niveau humain. Le Deep Learning résoudre ce problème dans RL en introduisant des représentations exprimées en termes d'autres représentations plus simples[13].

3.5 Apprentissage Profond (Deep Learning)

Le Deep Learning est un ensemble de méthodes d'apprentissage qui tentent de modéliser des données avec des architectures complexes combinant différentes transformations non linéaires. L'idée de l'apprentissage profond est basé sur les réseaux de neurones, qui sont combinés pour former les réseaux de neurones profonds (DNN). Ces techniques ont permis des progrès significatifs dans les domaines du traitement du son et de l'image, notamment la reconnaissance faciale, la reconnaissance vocale, la vision par ordinateur, le traitement automatique du langage, la classification des textes (détection du spam). Les applications potentielles sont très nombreuses. Un spectaculaire exemple est le programme AlphaGo, qui a battu le champion du monde de jeu GO en 2016. DL est l'une des techniques d'apprentissage automatique ML qui permet d'apprend les features directement à partir des données non structuré.

3.5.1 Histoire de DL

L'histoire de l'apprentissage profond peut être résumer dans le tableau 3.1.

3.5.2 Reseaux de neurones artificiels (ANNs)

L'histoire des réseaux de neurones a commencé en 1943, lorsqu'un neurophysiologiste Warren McCulloch et un mathématicien Walter Pitts ont proposé un modèle mathématique des neurones dans le cerveau et ont démontré que leur combinaison peuvent calculer n'importe quelle fonction arithmétique et logique. Ils ont également montré que les neurones peuvent être mis en œuvre avec des circuits électriques. Donald Hebb a suggéré une loi d'apprentissage pour les connexions entre les neurones dans un cerveau ou ils deviennent plus forts chaque fois qu'ils sont utilisés[9].

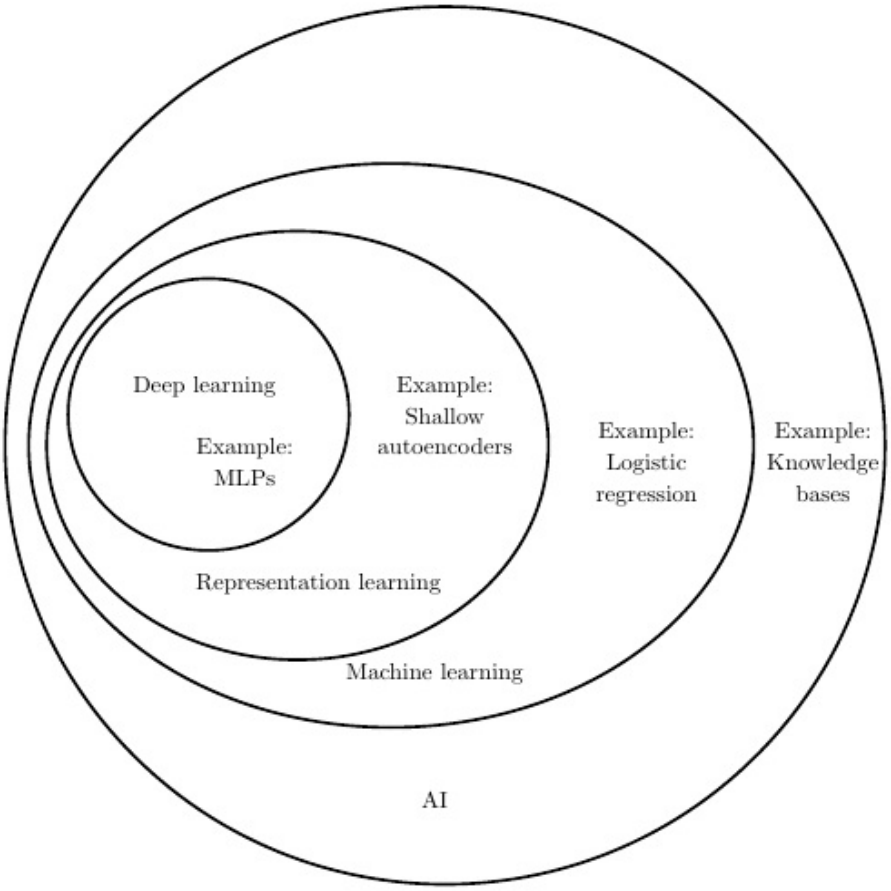

FIGURE 3.2 – Diagramme de Venn illustre la relation entre AI, ML, RL et DL[13]

3.5.2.1 Perceptron simple

Considérons un modèle d'un seul neurone. Formellement, c'est une fonction d'un argument qui est linéaire par rapport à l'entrée[9] :

$$\hat{y} = f(x) = \phi(x.w + b) = \phi(\sum_{i=1}^{N} x^i w^i + b)$$

Ici, le vecteur x est un vecteur d'entrée, w est un vecteur de paramètres également appelés $weights$, et b est le $bias$. Le vecteur w et le biais b doivent être ajustés afin de rapprocher $\hat{y}$ de la sortie désirée y.

Pour éviter le terme de bias, nous pouvons étendre le vecteur d'entrée x en ajoutant une valeur supplémentaire 1, et joindre w et b en un seul vecteur de poids(weights). Ainsi, maintenant, nous considérons toujours le vecteur étendu x avec le dernier élément égal à 1, et la matrice étendue des poids w, dans laquelle la dernière ligne est le vecteur des bias b. La fonction $\phi(z)$ est appelée une fonction d'activation et est généralement non linéaire. En 1957, Frank Rosenblatt a proposé un algorithme d'apprentissage des poids d'un seul neurone classifieur. Semblable aux idées de Donald Hebb, il met à jour les poids de telle sorte que l'argument de la fonction d'activation $x.w$ augmente pour les entrées de classe positive et diminue pour les entrées de classe négative. Formellement, la règle est la suivante[9] :

$$w \leftarrow w - (\hat{y} - y)x$$

Année	Contributeur	Contribution
300 AC	Aristotle	Introduction de l'associationnisme, a commencé l'histoire de la tentative de l'homme de comprendre le cerveau.
1873	Alexander Bain	Introduit les groupements des neurones comme les premiers modèles de réseau de neurones, inspirant la règle d'apprentissage Hebbian.
1943	McCulloch & Pitts	A introduit le modèle MCP, qui est considéré comme l'ancêtre du modèle de neurone artificiel.
1949	Donald Hebb	Considéré comme le père des réseaux de neurones, a introduit Hebbian Learning Rule, qui dresse les bases du réseau de neurones moderne.
1958	Frank Rosenblatt	A introduit le premier perceptron, qui ressemble au perceptron moderne.
1974	Paul Werbos	Introduit Backpropagation.
1980	Teuvo Kohonen	Introduit Self Organizing Map.
	Kunihiko Fukushima	Introduit Neocogitron, qui a inspiré le réseau de neurones convolutif.
1982	John Hopfield	Introduit Hopfield Network.
1985	Hilton & Sejnowski	Introduit Boltzmann Machine.
1986	Paul Smolensky	Introduit Harmonium, qui sera plus tard connu sous le nom Restricted Boltzmann Machine.
	Michael I. Jordan	Définie et introduite Recurrent Neural Network.
1990	Yann LeCun	Introduit LeNet, a montré la possibilité de réseaux de neurones profonds dans la pratique.
1997	Schuster & Paliwal	Introduit Bidirectional Recurrent Neural Network.
	Hochreiter &Schmidhuber	Introduit LSTM, résolu le problème de vanishing gradient dans les réseaux de neurones récurrents.
2006	Geoffrey Hinton	Introduit Deep Belief Networks, a également introduit la technique de pré-apprentissage par couche, a ouvert l'ère actuelle du deep learning.
2009	Salakhutdinov & Hinton	Introduit Deep Boltzmann Machines.
2012	Geoffrey Hinton	Introduit Dropout, un moyen efficace pour l'apprentissage du réseaux de neurones.

TABLEAU 3.1 – Étapes majeures de Deep Learning[35].

Ici, $y \in \{0, 1\}$ et $\hat{y} \in \{0, 1\}$ sont les étiquettes correctes et prédites. S'il n'y a pas d'erreur, c'est-à-dire $y = \hat{y}$, l'algorithme ne met pas à jour les pondérations. Il a été démontré que dans le cas de deux classes linéairement séparables, l'algorithme converge en un nombre fini d'étapes. En revanche, si les classes ne sont pas linéairement séparables, elle ne converge jamais[9].

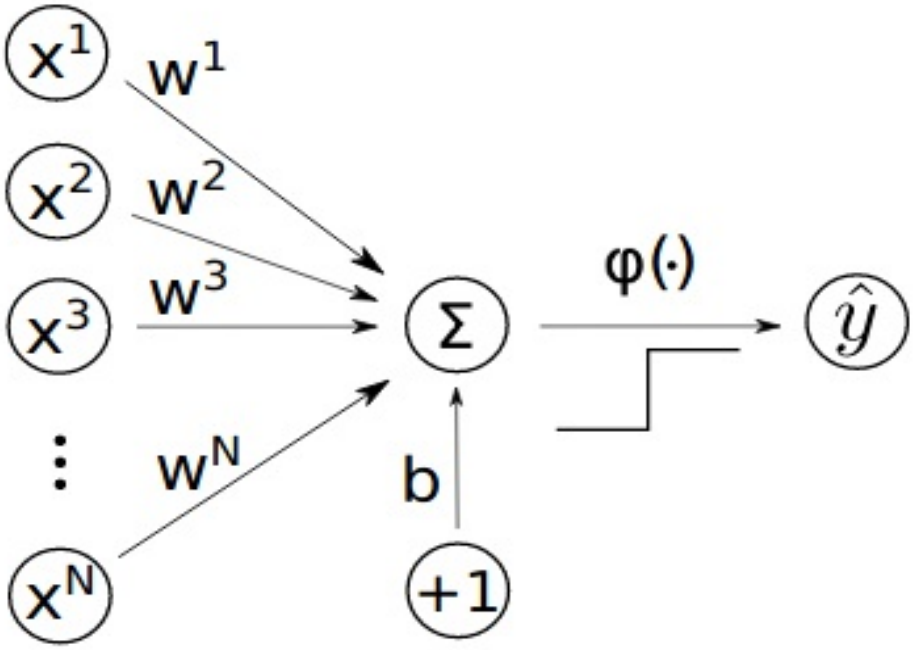

FIGURE 3.3 – Perception simple[9].

3.5.2.2 Perceptron Multi-class

Il est possible d'étendre un perceptron pour qu'il fonctionne avec plusieurs classes. Si le nombre de classes $M > 2$, le nombre de neurones doit également être M. Chacun d'eux doit être connecté au vecteur d'entrée par son propre ensemble de poids et donner sa propre prédiction. Ainsi, nous obtenons un vecteur de prédictions :

$$\hat{y} : \hat{y}^i = f^i(x), \forall i = 1, \ldots, M$$

Dans ce cas, la classe prédite est donnée par la valeur maximale de ce vecteur :

$$k : \hat{y}^k \geqslant \hat{y}^i, \forall i = 1, \ldots, M$$

La règle d'apprentissage du perceptron reste la même. Il suffit de l'appliquer indépendamment à chaque neurone de sortie. Il peut être écrit sous forme matricielle comme :

$$w \leftarrow w - x^T \cdot (\hat{y} - y) \tag{3.1}$$

où w est la matrice de taille $N \times M$, x est un vecteur $1 \times N$ et $\hat{y} - y$ est le vecteur d'erreurs de taille $1 \times M$. Ici $y_{i=1,\ldots,M} \in \{0, 1\}$ est vecteur d'étiquettes(labels), où 1 correspond à la classe correcte. Habituellement, l'objet x appartient à une seule classe, c'est-à-dire $\sum_{i=1}^{M} y_i = 1$.

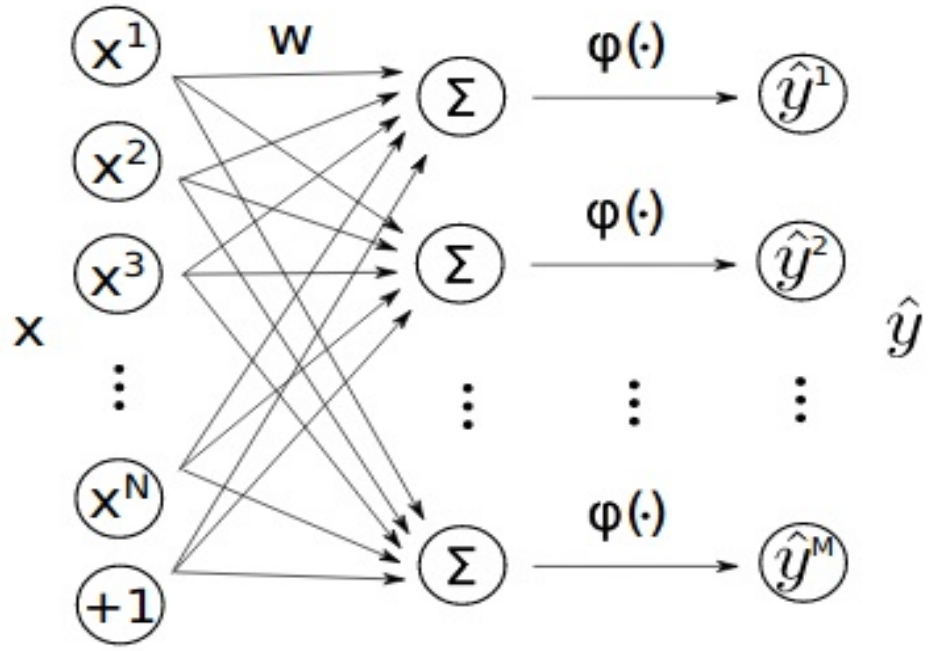

FIGURE 3.4 – Perceptron multi-classes[9].

3.5.2.3 Gradient Descent

En même temps, Bernard Widrow et son élève Ted Hoff ont introduit une règle d'apprentissage légèrement modifiée, appelée $\mu - LMS$. Au lieu de corriger les erreurs de classification, ils ont proposé d'utiliser l'erreur carré comme mesure de la qualité[9].

$$l(y, f(x)) = \frac{1}{2}\|y - f(x)\|_2^2 = \frac{1}{2}\sum_{i=1}^{M}(y^i - f^i(x))^2 \tag{3.2}$$

et minimiser l'erreur carré moyenne (MSE)

$$L(w) = \frac{1}{N}\sum_{i=1}^{N} l(y_i, f(x_i)) \tag{3.3}$$

sur les objets de l'ensemble d'apprentissage. A cet effet, ils ont suggéré d'utiliser une méthode de gradient descent

$$w \leftarrow w - \alpha\frac{\partial L(w)}{\partial w} \tag{3.4}$$

où α est un taux d'apprentissage, qui a régulé la vitesse de convergence. L'algorithme déplace le vecteur de poids w dans la direction qui diminue la valeur de la fonction de perte $L(w)$. En fait, il n'est pas nécessaire de calculer le gradient $\nabla_w L(w)$ sur l'ensemble d'apprentissage entière. À chaque itération, une bonne approximation peut être obtenue en faisant la moyenne sur son sous-ensemble choisi au hasard d'une taille plus petite, appelée batch. Il augmente les chances de converger vers un minimum global si $L(w)$ n'est pas convexe et réduit considérablement le temps de calcul. Cette modification est appelée descente de gradient stochastique (SGD). Pour utiliser SGD, il est nécessaire de calculer les gradients de

poids $\nabla_w l(x; w)$. Ils peuvent être calculés à l'aide de la règle[9] :

$$\frac{\partial L}{\partial w^{ij}} = \frac{\partial L}{\partial \hat{y}^j} \frac{\partial \hat{y}^j}{\partial w^{ij}}, \forall i = 1, \ldots, N, j = 1, \ldots, M \tag{3.5}$$

Pour la fonction de perte au carré $l(y, f(x))$, les dérivées par rapport aux prédictions $\hat{y}$ sont calculées comme

$$\frac{\partial L}{\partial \hat{y}^i} = \hat{y}^i - y^i \Rightarrow \frac{\partial L}{\partial \hat{y}} = \hat{y} - y \tag{3.6}$$

Cependant, pour calculer $\partial \hat{y}^j / \partial w^{ij}$, la fonction $\phi(z)$ doit être différenciable, Si aucune fonction $\phi(z)$ n'est utilisée, les gradients $\nabla_w \hat{y}$ sont simplement les éléments du vecteur d'entrée :

$$\frac{\partial \hat{y}^j}{\partial w^{ij}} = x^i, \forall i = 1, \ldots, N, j = 1, \ldots, M$$

Ainsi, nous pouvons écrire la règle $\mu - LMS$ sous la forme matricielle comme :

$$w \leftarrow w - \alpha \frac{\partial L}{\partial w} = w - \alpha x^T \cdot (y - \hat{y})$$

3.5.2.4 Fonctions d'activation

Les fonctions d'activation sont des fonctions utilisées dans les réseaux de neurones pour calculer la somme pondérée des entrées et des bias, qui sert à décider si un neurone peut être déclenché ou non. Il manipule les données présentées à travers un traitement de gradient généralement gradient descent (GD) et produit ensuite une sortie pour le réseau de neurones, qui contient les paramètres dans les données. Les fonctions d'activation les plus connues sont les suivantes :

3.5.2.4.1 Sigmoid : Le Sigmoid est un AF non linéaire utilisé principalement dans les réseaux de neurones à action directe. La fonction sigmoid est donnée par la relation :

$$f(x) = \frac{1}{1 + \exp^{-x}} \tag{3.7}$$

La fonction sigmoid apparaît dans les couches de sortie des architectures DL, et elles sont utilisées pour prédire la sortie basée sur les probabilités et a été appliquée avec succès dans les problèmes de classification binaire, en modélisant les tâches de régression logistique ainsi que d'autres domaines de réseau neurones.

3.5.2.4.2 Hyperbolic Tangent (Tanh) : La fonction tangente hyperbolique est un autre type d'AF utilisé en DL et il a quelques variantes utilisées dans les applications DL. Il est connue sous le nom de fonction tanh, est une fonction centrée sur zéro dont la plage se situe entre -1 et 1, ainsi la sortie de la fonction tanh est donnée par :

$$f(x) = \frac{exp^x - exp^{-x}}{exp^x + exp^{-x}} \tag{3.8}$$

La fonction tanh est devenue la fonction préférée par rapport à la fonction sigmoïd en ce qu'elle donne de meilleures performances d'apprentissage pour les réseaux de neurones multicouches. Les fonctions tanh ont été utilisées principalement dans les réseaux de neurones récurrents pour le traitement du langage naturel et les tâches de reconnaissance vocale [28].

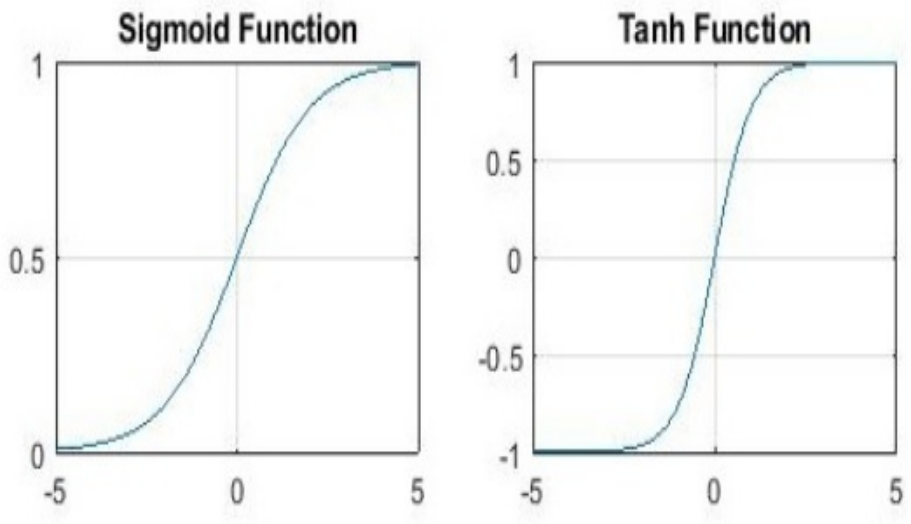

FIGURE 3.5 – Representation graphique des fonctions d'activations Sigmoid et Tanh [28].

3.5.2.4.3 Rectified Linear Unit (ReLU) : La fonction d'activation d'unité linéaire rectifiée (ReLU) a été proposée par Nair et Hinton 2010, et depuis lors, a été la fonction d'activation la plus largement utilisée pour les applications d'apprentissage profond avec des résultats de pointe à ce jour. Le ReLU représente une fonction quasi linéaire et préserve donc les propriétés des modèles linéaires qui les rendaient faciles à optimiser, avec des méthodes de gradient descent[13]. La fonction d'activation ReLU effectue une opération de seuil pour chaque élément d'entrée où les valeurs inférieures à zéro sont mises à zéro, ainsi ReLU est donné par :

$$f(x) = max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases} \tag{3.9}$$

L'avantage d'utiliser ReLU dans le calcul est que, elles garantissent un calcul plus rapide car il ne calcule pas les exponentielles et les divisions, avec une vitesse globale de calcul améliorée[36].

3.5.2.4.4 Softmax : La fonction Softmax est un autre type de fonction d'activation. Il est utilisé pour calculer la distribution de probabilité à partir d'un vecteur de nombres réels. La fonction Softmax produit une sortie qui est une plage de valeurs entre 0 et 1, la somme des probabilités étant égale à 1. La fonction Softmax[13] est calculée en utilisant la relation :

$$f(x_i) = \frac{exp^{x_i}}{\sum_j exp^{x_j}} \tag{3.10}$$

La fonction Softmax est utilisée dans les modèles multi-classes où elle renvoie les probabilités de chaque classe, la classe cible ayant la probabilité la plus élevée. La principale différence entre le Sigmoid et Softmax AF est que le Sigmoid est utilisé dans la classification binaire tandis que le Softmax est utilisé pour les tâches de classification multivariées.

3.5.2.5 Perceptron multi-couche (MLP)

Alors que les perceptrons implémentent une fonction linéaire dans l'espace vectoriel d'entrée, dans les applications réelles, la frontière de séparation entre les classes est rarement linéaire. Cependant, il pourrait être possible que les classes soient linéairement séparables dans un autre espace. Dans ce cas, une transformation supplémentaire est requise. Dans le perceptron multiclasses, le vecteur de prédictions y est une fonction linéaire du vecteur d'entrée x et peut être écrit sous forme matricielle comme[9] :

$$\hat{y} = x \cdot w$$

Ici, x et y peuvent être considérés comme deux couches, entièrement reliées par les poids dans la matrice w. La transformation supplémentaire peut être effectuée par une autre couche de neurones située entre l'entrée et la sortie, et entièrement connectée avec les deux. Cela s'appelle une couche cachée. Si nous énumérons les couches comme $y_0 = x, y_1$ et $y_2 = \hat{y}$, nous pouvons écrire :

$$y_1 = y_0 \cdot w_1, y_2 = y_1 \cdot w_2$$

Cependant, un tel perceptron est équivalent à un perceptron sans couche cachée avec la matrice de poids $w = w_1 \cdot w_2$. Afin de rendre utile la couche cachée, la fonction de transformation de ses neurones doit être non linéaire[9].

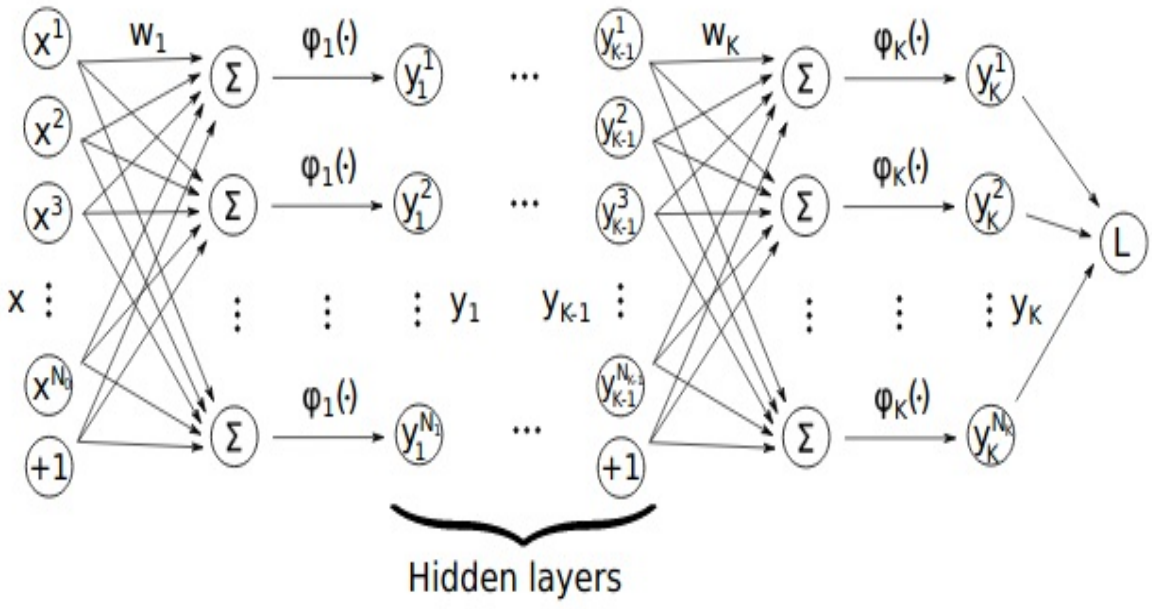

FIGURE 3.6 – Perceptron Multi-couche[9].

Les perceptrons multicouches ne contiennent qu'un type particulier de couches, appelées couches entièrement connectées, qui ont une connexion indépendante entre chaque paire de neurones. Les perceptrons généralisés qui pourraient contenir des couches d'autres types sont appelés réseaux de neurones.

3.5.2.6 Backpropagation

Appliquant les idées de Bryson et Kelley aux perceptrons multicouches, Paul Werbos [9] a proposé une extension de la règle d'apprentissage $\mu - LMS$, appelée algorithme de backpropagation.

Considérons un perceptron multicouche avec K couches de neurones : $y_0 = x$ vecteur d'entrée de taille $1 \times N$, et les K couches suivantes :

$$y_k = \phi_k(y_{k-1} \cdot w_k), \forall k = 1, \ldots, K$$

avec les fonctions d'activation $\phi_k(z)$, où y_K est le vecteur de sortie de la taille $1 \times M$. Si les fonctions $\phi_k(z)$ sont différenciables, nous pouvons calculer les gradients pour la couche y_{k-1} en utilisant les gradients pour la couche y_k en utilisant la règle :

$$\frac{\partial L}{\partial y_{k-1}^i} = \sum_{j=1}^{M} \frac{\partial L}{\partial y_k^j} \cdot \frac{\partial y_k^j}{\partial y_{k-1}^i} = \sum_{j=1}^{M} \frac{\partial L}{\partial y_k^j} \cdot \left. \frac{\partial \phi_k(z)}{\partial z} \right|_{z=y_{k-1} w_k^j} \cdot w_k^{ij} \tag{3.11}$$

Ici w_k^j est le vecteur de poids, reliant la couche $k-1$ au neurone j sur la couche k. Comme nous pouvons le voir, les gradients de la couche précédente $\partial L / \partial y_{k-1}$ dépendent uniquement des gradients de la couche actuelle $\partial L / \partial y_k$, de la dérivée de la fonction d'activation $\nabla_z \phi_k(z)$ et des poids de couche w_k. Par conséquent, il est possible de calculer de manière itérative les gradients pour toutes les couches de $K-1$ à 0, en utilisant $\partial L / \partial y_K$3.6 lors de la première itération. Les gradients de poids $\partial L / \partial w_k$ peuvent également être calculés à l'aide de la règle :

$$\frac{\partial L}{\partial w_k^{ij}} = \frac{\partial L}{\partial y_k^j} \cdot \frac{\partial y_k^j}{\partial w_k^{ij}} = \frac{\partial L}{\partial y_k^j} \cdot \left. \frac{\partial \phi_k(z)}{\partial z} \right|_{z=y_{k-1} w_k^j} \cdot y_{k-1}^i \tag{3.12}$$

et peut donc être obtenu de manière itérative pour toutes les couches de K à 1 une fois que les gradients $\partial L / \partial y_k$ sont disponibles.

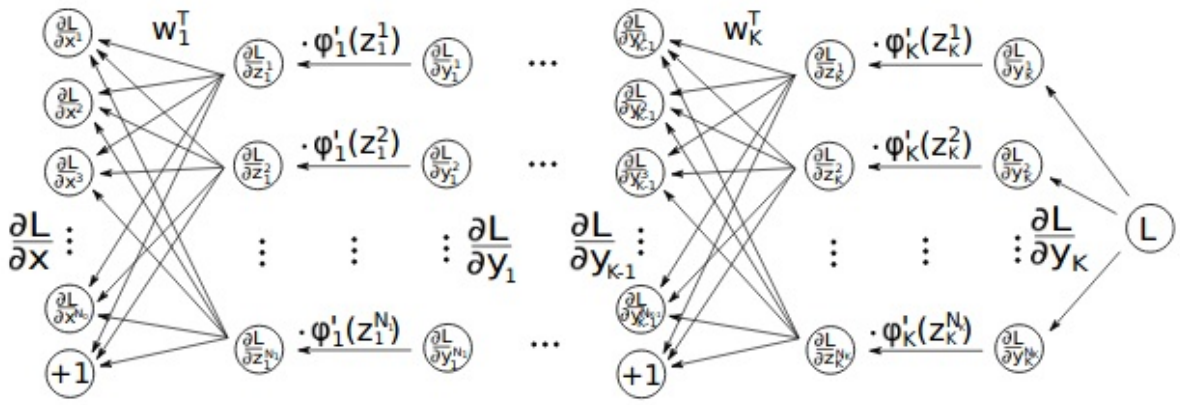

FIGURE 3.7 – Etapes de Backpropagation[9].

L'algorithme peut être divisé en trois parties : forward étape, backward étape et calcul du gradient de poids. Sur le forward étape, nous initialisons le vecteur d'entrée y_0 avec un objet d'apprentissage x et calculons de manière itérative les valeurs de couche suivantes y_i de 1 à K. Sur la passe arrière, nous initialisons les gradients de prédiction

$\partial L/\partial y_K$ avec l'équation 3.6, et les propagons à travers le réseau pour les couches de $K-1$ à 0, obtenant les gradients $\partial L/\partial y_i$.

Les gradients de poids $\partial L/\partial w_i$ peuvent ensuite être calculés en parallèle ou après le backward étape, en utilisant les valeurs de couche y_i et leurs gradients $\partial L/\partial y_i$. Les gradients de poids sont ensuite utilisés pour mettre à jour les poids utilisant l'équation 3.4. Lorsque la taille d'un batch est supérieure à 1, la moyenne des gradients de poids est utilisée. Habituellement, l'algorithme de backpropagation traite plusieurs fois le même ensemble d'apprentissage, jusqu'à ce que l'erreur de test cesse de diminuer. Chaque itération de traitement est appelée $epoch$[9].

3.5.3 Types d'architecture des ANNs :

Les ANNs sont développé très rapidement, de nombreux nouveaux réseaux et nouvelles architectures apparaissent chaque mois, dans ce qui suit on résume les architectures les plus populaires

3.5.3.1 Feed-forward Neural Networks (FNN)

C'est le réseau hautement interconnecté de neurones artificiels inspiré par le système nerveux humain, travaillant en union pour effectuer une tâche donnée et, à la fin, donne la décision basée sur les poids et les bias pour les données d'entrée complexes.

3.5.3.2 AutoEncoder (AE)

Il s'agit généralement d'un réseau de neurones de type Feed-Forward qui vise à apprendre une représentation compressée et distribuée (encoding) d'un dataset.

3.5.3.3 Restricted Boltzmann Machines (RBM)

RBM est un ANN stochastique génératif qui peut apprendre la distribution de probabilité sur ses jeux de données d'entrée. Ils peuvent être apprendre de façon supervisés ou non supervisés selon la tâche avec une restriction où la paire de nœuds des unités visibles et cachées peut avoir une connexion symétrique entre eux et il n'y a pas de connexion entre les nœuds au sein d'un groupe.

3.5.3.4 Deep Belief Networks (DBN)

Il s'agit d'un modèle génératif composé de plusieurs couches de RBM et d'autoencodeurs. Lorsqu'il est apprend sans supervision, il peut apprendre à restreindre ses entrées de manière probabiliste et agit comme des détecteurs de features et avec supervision pour effectuer la classification.

3.5.3.5 Convolution Neural Networks (CNN)

Les CNNs sont un type spécifique de réseau neurones conçu pour la reconnaissance d'images Tout comme les ANNs ont des nœuds (neurones), les CNN utilisent des filtres(Kernels)[21], qui sont essentiellement des grilles de neurones qui peuvent apprendre des motifs dans les images.

Les filtres des couches initiales de CNN apprennent des features de base comme les lignes horizontales et verticales. Alors que nous avançons plus profondément dans les couches, le filtre commence à reconnaître les caractéristiques complexes comme les oreilles, le nez, la bouche, etc., et dans la couche finale, le réseau peut identifier l'image entière basée sur ce réseau CNN.

3.5.3.6 Recurrent Neural Networks (RNN)

Ceux-ci sont conçus pour reconnaître des textes manuscrits, des paroles, etc. C'est l'un des types les plus puissants et les plus utiles, applicable même aux images qui peuvent être décomposées en séries de patchs et traitées comme des séquences.

3.5.3.7 Generative Adversarial Networks (GAN)

Les GANs sont des modèles génératifs qui génèrent de nouvelles données similaires aux données sur lesquelles ils ont été apprenti. Les GANs sont essentiellement des systèmes à double agent dans lesquels il y a deux réseaux de neurones qui sont entraînés l'un contre l'autre. L'un de ces réseaux de neurones est le générateur, qui essaie de générer de nouvelles données, tandis que l'autre est le discriminateur, dont la tâche est de déterminer si les données générées sont fausses ou non.

3.5.4 Architecture profond

L'architecture profond fait référence au nombre de niveaux de composition d'opérations non linéaires dans la fonction apprise. Inspirés par la profondeur architecturale du cerveau, les chercheurs en réseaux de neurones souhaitaient depuis des décennies apprennent des réseaux de neurones multicouches profonds[3], mais aucune tentative réussie n'a été signalée avant 2006 (Sauf pour les CNNs) : les chercheurs ont rapporté des résultats expérimentaux positifs avec généralement deux ou trois couches (c-à-d. une ou deux couches cachées), mais l'apprentissage des réseaux plus profonds ont donné des résultats toujours plus faibles.

En 2006, Hinton et al. à l'Université de Toronto a introduit Deep Belief Networks (DBNs), avec un algorithme d'apprentissage qui apprenne avec avidité une couche à la fois, exploitant un algorithme d'apprentissage non supervisé pour chaque couche, une Restricted Boltzmann Machine (RBM). Peu de temps après, des algorithmes connexes basés sur des autoencodeurs ont été proposés[4], exploitant apparemment le même principe : guider l'apprentissage de niveaux intermédiaires de représentation à l'aide d'un apprentissage non supervisé, qui peut être effectué localement à chaque niveau. Bien que les auto-encodeurs, les RBM et les DBN puissent être entraînés avec des données sans étiquette, dans de nombreuses applications, ils ont été utilisés avec succès pour initialiser des réseaux de neurones feedforward profond supervisés appliqués à une tâche spécifique[13].

3.5.5 Applications de Deep Learning

Les différentes types de DL et leur possibles utilisations peut etre résumé dans le tableau suivant :

Application	Type de DL
Classification	Neural Networks
Reconnaissance Faciale	Convolutional Neural Networks(CNNs)
Véhicules autonomes	CNNs
Recherche vocale	Recurrent Neural Networks(RNNs)
Traduction automatique	RNNs
Deep Fakes	Generative Adversarial Networks (GANs)
Prévision météo	Neural Networks
Reconstruction 3D	Deep Reinforcement Learning(DRL)
Systèmes multi-agents (Jeux)	DRL
Robotique	DRL

TABLEAU 3.2 – Applications de Deep Learning[8].

3.5.6 Techniques d'optimisations dans DL

L'apprentissage d'un DNN est un processus d'optimisation pour trouver les paramètres dans le réseau qui minimisent la fonction de perte (Loss function).

3.5.6.1 Stochastic Gradient Descent (SGD)

En pratique, la méthode SGD[33] est un algorithme fondamental appliqué à l'apprentissage profond, qui ajuste itérativement les paramètres en fonction du gradient pour chaque échantillon d'apprentissage. La complexité de calcul de SGD est inférieure à celle de la méthode originale de gradient descent (GD), dans laquelle les paramètres sont mis à jour chaque itération pour le dataset complet.

Dans le processus d'apprentissage, la vitesse de mise à jour est contrôlée par l' hyperparamètre de taux d'apprentissage. Les taux d'apprentissage plus faibles conduire à un état optimal après une longue période, tandis que des taux d'apprentissage plus élevés diminueront la perte plus rapidement, mais peuvent provoquer des variations pendant l'apprentissage[29].

3.5.6.2 Momentum

Afin de contrôler l'oscillation de SGD, l'idée d'utiliser momentum est introduite. Inspirée par la première loi du mouvement de Newton, cette technique obtient une convergence plus rapide et une momentum appropriée qui peuvent améliorer les résultats d'optimisation de SGD [33].

D'autre part, plusieurs techniques sont proposées pour déterminer le taux d'apprentissage approprié. Principalement, la décroissance du poids et la décroissance du taux d'apprentissage sont introduites pour ajuster le taux d'apprentissage et accélérer la convergence. Une décroissance de poids fonctionne comme un coefficient de pénalité dans la fonction de coût pour éviter le sur-ajustement (overfitting), et une décroissance du taux d'apprentissage peut réduire le taux d'apprentissage de manière dynamique pour améliorer les performances. De plus, l'adaptation du taux d'apprentissage par rapport au gradient des étapes précédentes est utile pour éviter la fluctuation.

3.5.6.3 Adagrad

Est le premier algorithme adaptatif utilisé avec succès en apprentissage profond. Il amplifie le taux d'apprentissage pour les paramètres rarement mis à jour et supprime le taux d'apprentissage pour les paramètres fréquemment mis à jour en enregistrant les gradients carrés accumulés[11].

3.5.6.4 Adadelta

Les gradients au carré étant toujours positifs, le taux d'apprentissage d'Adagrad peut devenir extrêmement faible et n'optimise plus le modèle. Adadelta[19] est proposé pour résoudre ce problème où une fraction de diminution est introduite pour limiter l'accumulation des gradients au carré.

3.5.6.5 Adam

l'Adadelta est encore améliorée en introduisant une autre fraction de diminution pour enregistrer l'accumulation des gradients [19]. Il est démontré que Adam donne de meilleurs résultats dans la pratique que les autres algorithmes avec un taux d'apprentissage adaptatif.

3.5.7 Techniques de Regularisation

Le sur-apprentissage (overfitting) fait référence au phénomène où le réseau de neurone donne de bon résultat avec les données d'apprentissage mais échoue lorsqu'il voit de nouvelles données du même catégorie. Le rôle de la régularisation est de modifier un modèle d'apprentissage profond pour bien fonctionner avec des entrées en dehors de données d'apprentissages. Plus précisément, la régularisation se concentre sur la réduction de taux d'erreur dans la phase de test sans affecter le taux d'erreur d'apprentissage initiale.

3.5.7.1 L2 & L1 regularization

L1 et L2 sont les types de régularisation les plus courants. Celles-ci mettent à jour la fonction de coût général en ajoutant un autre terme appelé terme de régularisation.

$$\text{Loss} = \text{Loss} + \text{terme de régularisation}$$

Avec ce terme de régularisation, les valeurs des matrices de poids diminuent car cela suppose qu'un réseau de neurone avec des matrices de poids plus petites conduit à des modèles plus simples. Par conséquent, cela réduira également le sur-ajustement. Ce terme de régularisation diffère en L1 et L2

Dans L2 nous avons :

$$\hat{L}(W) = L(W) + \frac{\alpha}{2}\|W\|_2^2 = L(W) + \frac{\alpha}{2}\sum_i \sum_j w_{ij}^2$$

α est le paramètre de régularisation. C'est l'hyperparamètre dont la valeur est optimisée pour de meilleurs résultats. La régularisation L2 est également connue sous le nom de décroissance de poids (weight decay) car elle oblige les poids à se diminuer vers zéro (mais

pas exactement zéro). Et pour L1 :

$$\hat{L}(W) = L(W) + \alpha\|W\|_1 = L(W) + \alpha \sum_i \sum_j |w_{ij}|$$

Dans ce cas, nous pénalisons la valeur absolue des poids. Contrairement à L2, les poids peuvent être réduits à zéro.

3.5.7.2 Dropout

L'une des méthodes de régularisation les plus populaires est dropout, Lors de l'utilisation du dropout pendant l'apprentissage, un paramètre est défini sur une valeur comprise entre 0 et 1. Si cette valeur est inférieure à la valeur du paramètre, ce neurone n'est pas activé. Cette méthode consiste à supprimer aléatoirement les neurones de la couche avec une probabilité de $1 - p$, ce qui rend les valeurs de sortie des neurones égales à 0. Au moment du test, les neurones ne sont pas supprimés, mais les poids de sortie sont multipliés par p.

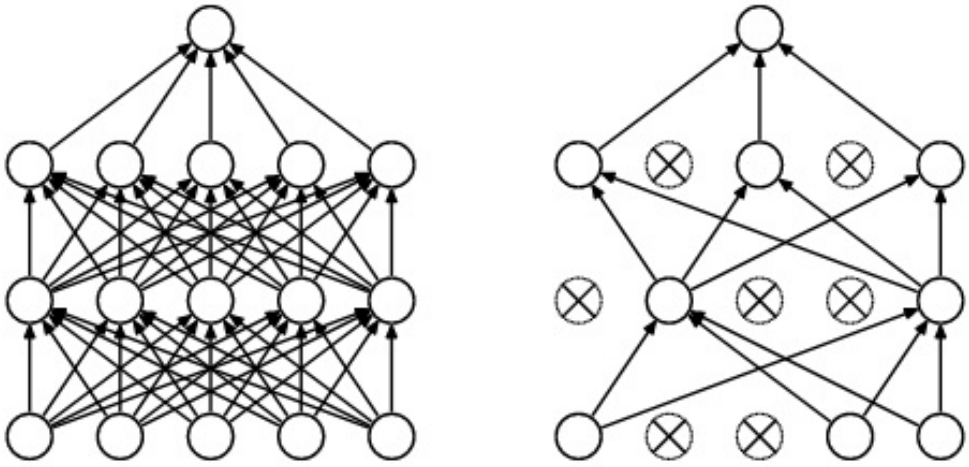

FIGURE 3.8 – Le reseau de neurone standard et aprés l'utilisation du dropout[31].

3.5.7.3 Data augmentation

Le moyen le plus simple de réduire le sur-apprentissage est d'augmenter la taille des données d'apprentissage. Cette technique peut etre utilisée avec les données de type images. Dans ce cas, il existe plusieurs façons d'augmenter la taille des données d'apprentissage comme la rotation de l'image, retournement, mise à l'échelle, décalage, etc.

3.5.7.4 Early stopping

C'est une technique de régularisation qui utilise les performances de l'ensemble de validation (validation set) comme test pour éviter le sur-apprentissage. La façon dont il le fait en arrêtant le processus d'apprentissage chaque fois que les performances du réseau sur l'ensemble de validation cessent de s'améliorer.

3.6 Conclusion

Dans ce chapitre, nous avons vu qu'est ce que le DL et leur historique, nous avons vu aussi les notions de base concernant les réseaux de neurones artificielles, leurs différentes architectures, les domaines d'applications, et finalement nous avons introduit les différentes techniques d'optimisations et de régularisations.

Chapitre 4

Contribution

La théorie, c'est quand on sait tout et que rien ne fonctionne.
La pratique, c'est quand tout fonctionne et que personne ne sait pourquoi.

Albert Einstein - The World As I See It

4.1 Introduction

Dans le chapitre précédent, nous avons vu des différents types des DNNs et leurs utilisations, Les réseaux de neurones convolutifs (CNNs) est l'un de ces types qui a démontré une excellente réussite dans les problèmes de vision par ordinateur, en particulier dans la classification d'images. Pour cela, on a choisi d'utiliser le CNN pour détecter les deepfakes.

Premièrement, on va expliquer l'architecture de réseaux CNN, citons quelques reseaux célèbres, puis on va parler d'un réseau résiduel, ce dernier utilise une représentation résiduelle pour améliorer le réseau de neurones convolutifs, et finalement faire une description de notre modèle et son architecture.

4.2 Les réseaux de neurones convolutif (CNN)

Le réseau de neurones convolutionnels (CNN ou ConvNet) est un type particulier de réseau de neurones multicouche inspiré par le mécanisme du système optique des créatures vivantes. En 1980, Kunihiko Fukushima a introduit le néocognitron [12] qui est un réseau de neurone multicouche capable de reconnaître hiérarchiquement le modèle visuel par l'apprentissage. Ce réseau est considéré comme l'inspiration théorique de CNN.
Les réseaux de neurones convolutifs (CNN) introduits par LeCun[22] ont révolutionné le traitement d'image et remplacé la tâche de l'extraction manuelle des fonctionnalités. CNN agit directement sur les matrices, voire sur les tenseurs pour les images avec trois canaux de couleurs RVB.

4.2.1 Couches dans un CNN

Un réseau de neurones convolutifs est composé de plusieurs types de couches, les couches convolutionnelles, couches de pooling et couches entièrement connectées (fully connected).

4.2.2 Couche de convolution (Convolution layer)

La convolution discrète entre deux fonctions f et g est définie comme :

$$(f * g)(x) = \sum_t f(t)g(x+t).$$

Pour les signaux bidimensionnels tels que les images, nous considérons les convolutions 2D

$$(K * I)(i,j) = \sum_{m,n} K(m,n)I(i+n, j+m)$$

K est un filtre (kernel) de convolution appliqué à un signal 2D (ou image) I.

Comme le montre la figure 4.1, le principe de la convolution 2D est de faire glisser un filtre de convolution sur l'image.

À chaque position, nous obtenons la convolution entre le filtre et la partie de l'image qui est actuellement traitée. Ensuite, le filtre se déplace d'un nombre s de pixels, s est appelé *stride*. Lorsque le stride est de petit, nous obtenons des informations redondantes. Parfois, nous ajoutons également un *zero padding*, qui est une marge de taille p contenant des valeurs nulles autour de l'image afin de contrôler la taille de la sortie.
Supposons que nous appliquons des filtres C_0 (également appelés kernels), chacun de taille $k \times k$ sur une image. Si la taille de l'image d'entrée est $W_i \times H_i \times C_i$ (W_i désigne la largeur, H_i la hauteur et C_i le nombre de canaux, généralement $C_i = 3$), le volume de la sortie est $W_0 \times H_0 \times C_0$, où C_0 correspond au nombre de filtres que nous considérons, et

$$W_0 = \frac{W_i - k + 2p}{s} + 1$$

$$H_0 = \frac{H_i - k + 2p}{s} + 1$$

Si l'image a 3 canaux et si K_l ($l = 1, ..., C_0$) dénote $5 \times 5 \times 3$ filtres (où 3 correspond au nombre de canaux de l'image d'entrée), la convolution avec l'image I et le filtre K_l correspond à la formule :

$$K_l * I(i,j) = \sum_{c=0}^{2} \sum_{n=0}^{4} \sum_{m=0}^{4} K_l(n,m,c)I(i+n-2, i+m-2, c).$$

Plus généralement, pour les images avec des canaux C_i, la forme du filtre est (k, k, C_i, C_0) où C_0 est le nombre de canaux de sortie (nombre de filtres) que l'on considère. Le nombre de paramètres associés à un filtre de forme (k, k, C_i, C_0) est $(k \times k \times C_i + 1) \times C_0$.

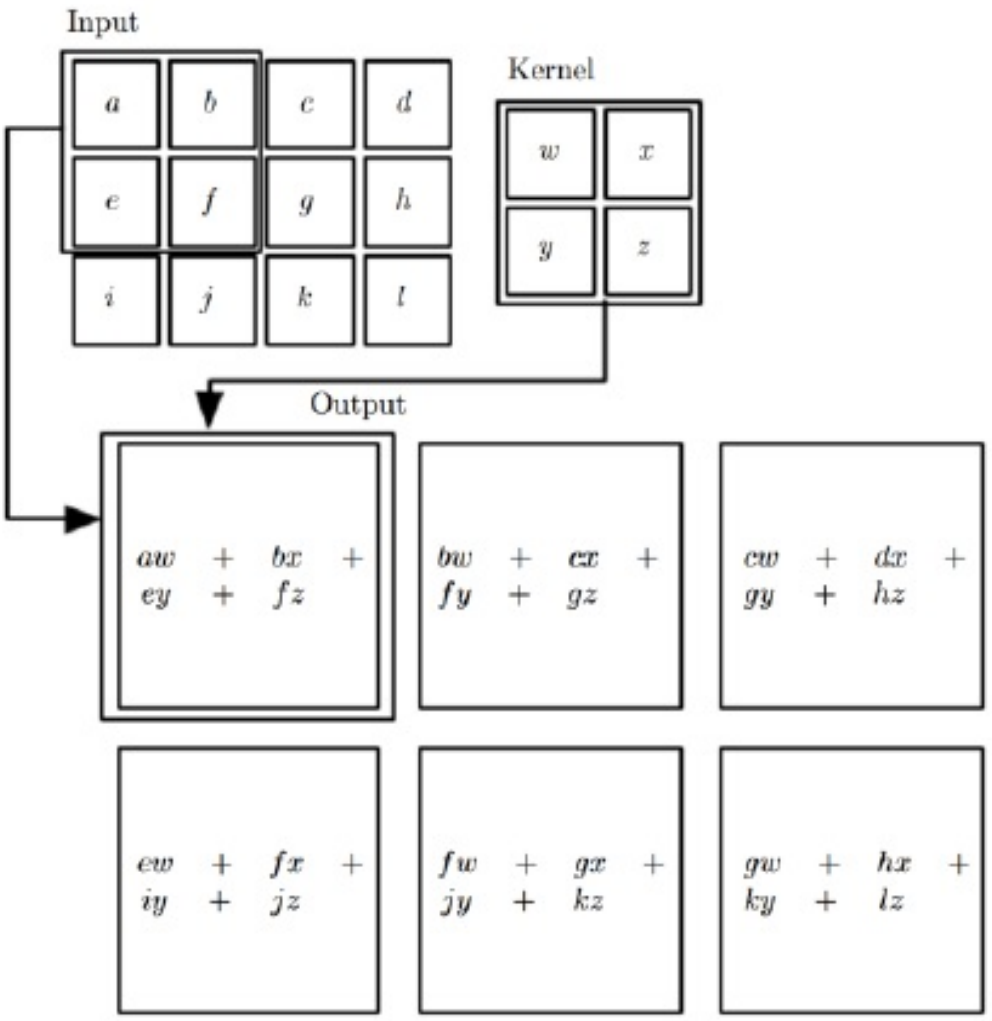

FIGURE 4.1 – Convolution 2D[13].

Les opérations de convolution sont combinées avec une fonction d'activation ϕ (généralement la fonction d'activation Relu) : si l'on considère un filtre K de taille $k \times k$, si x est un patch a $k \times k$ d'image, l'activation est obtenue en faisant glisser le $k \times k$ fenêtre et calcul $z(x) = \phi(K * x + b)$, où b est un biais.

C'est dans la couche de convolution que nous trouvons la force du CNN, en effet, le CNN apprendra les filtres (ou noyaux) qui sont les plus utiles pour la tâche que nous devons faire (comme la classification). Un autre avantage est que l'on peut considérer plusieurs couches de convolution : la sortie d'une convolution devient l'entrée de la suivante.

4.2.3 Couche de pooling (Pooling layer)

L'idée principale de pooling est le downsampling afin de réduire la complexité pour d'autres couches. Dans le domaine du traitement d'image, cela peut être considéré similaire à la réduction de la résolution., le pooling n'affecte pas le nombre de filtres. en prenant la moyenne ou le maximum sur les patchs de l'image (mean-pool ou max-pool).

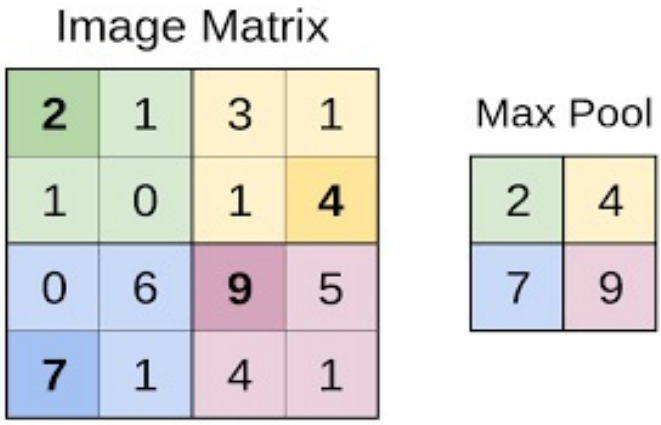

FIGURE 4.2 – Opération de max-pooling.

Comme les couches convolutifs, couche de pooling agit sur de petites parcelles de l'image, nous avons aussi un stride. Si nous considérons 2×2 patchs, sur lesquels nous prenons la valeur maximale pour définir la couche de sortie, et un stride $s = 2$, nous divisons par 2 la largeur et la hauteur de l'image. (Rq :il est également possible de réduire la dimension avec la couche convolutif, en prenant un stride supérieure à 1, et sans zero padding).
Il convient de considérer que le pooling ne préserve pas la position des informations. Par conséquent, elle ne devrait être appliquée que lorsque la présence d'informations est importante (plutôt que d'informations spatiales) [2].

4.2.4 Couche entièrement connectée (Fully connected layer)

Après plusieurs couches de convolution et de pooling, le CNN se termine généralement un ou plusieurs couches entièrement connectées. La couche entièrement connectée est similaire à la façon dont les neurones sont disposés dans un réseau de neurones traditionnel. Par conséquent, chaque nœud dans une couche entièrement connectée est directement connecté à chaque nœud de la couche précédente et de la couche suivante, comme indiqué dans la figure 4.3

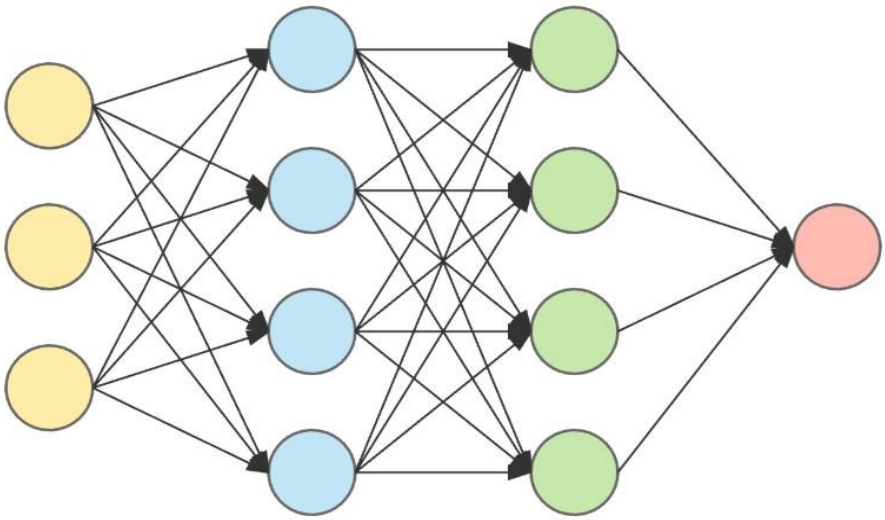

FIGURE 4.3 – Couche entièrement connectée.

Chacun des nœuds de dernière couche de convolution ou pooling est connecté en tant que vecteur à la première couche de la couche entièrement connectée. Les paramètres les plus utilisés avec le CNN sont dans la couche fully-connected, et prennent beaucoup de temps dans l'apprentissage.

4.2.5 Architectures existantes

Nous avons décrit les différents types de couches composant un CNN. Nous présentons maintenant comment ces couches sont combinées pour former l'architecture du réseau. Le choix d'une architecture est très complexe et c'est plus de l'ingénierie qu'une science exacte. Il est donc important d'étudier les architectures qui se sont révélées être efficaces et d'inspirer de ces exemples célèbres. Dans le CNN le plus classique, nous enchaînons plusieurs fois une couche de convolution suivie d'une couche de pooling et nous ajoutons à la fin des couches entièrement connectées comme illustré dans la figure 4.4

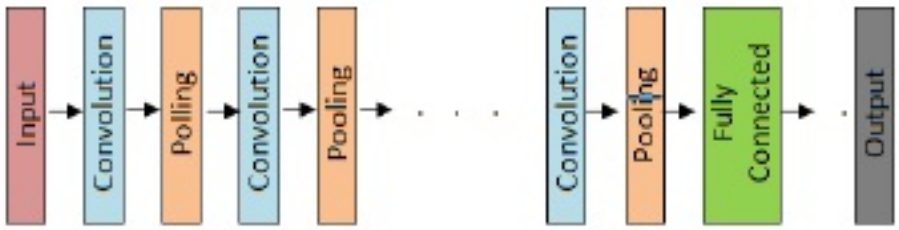

FIGURE 4.4 – Bloc de construction d'un CNN typique[32].

Le réseau LeNet, proposé par l'inventeur du CNN, Yann LeCun[23] est de ce type, comme le montre la figure 4.5. Ce réseau était dédié à la reconnaissance des chiffres. Il est composé seulement de quelques couches et de quelques filtres, en raison des limitations informatiques de l'époque.

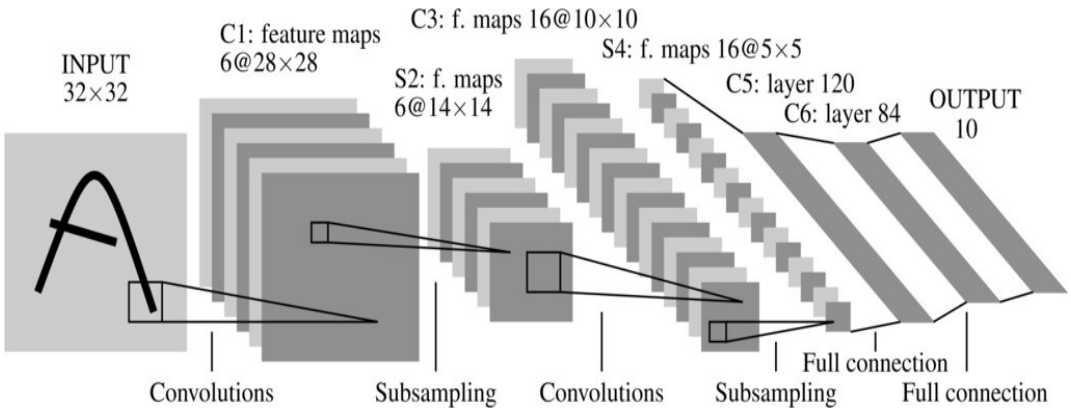

FIGURE 4.5 – Architecture de LeNet-5[23]

Quelques années plus tard, avec l'apparition des cartes GPU (Graphical Processor Unit), des architectures beaucoup plus complexes pour CNN ont été proposées, comme le réseau AlexNet [21] qui a remporté le concours ImageNet et pour lequel une version simplifiée est présentée dans la figure 4.6. Ce concours a été consacrée à la classification d'un million d'images en couleurs sur 1000 classes. La résolution des images était de 224×224. AlexNet est composé de 5 couches de convolution, de 3 couches de max pooling 2×2 et

de couches entièrement connectées. Comme le montre la figure 4.6, la forme du filtre de la première couche de convolution est $(11, 11, 3, 96)$ avec une stride de $s = 4$, et la première forme de sortie est $(55, 55, 96)$.

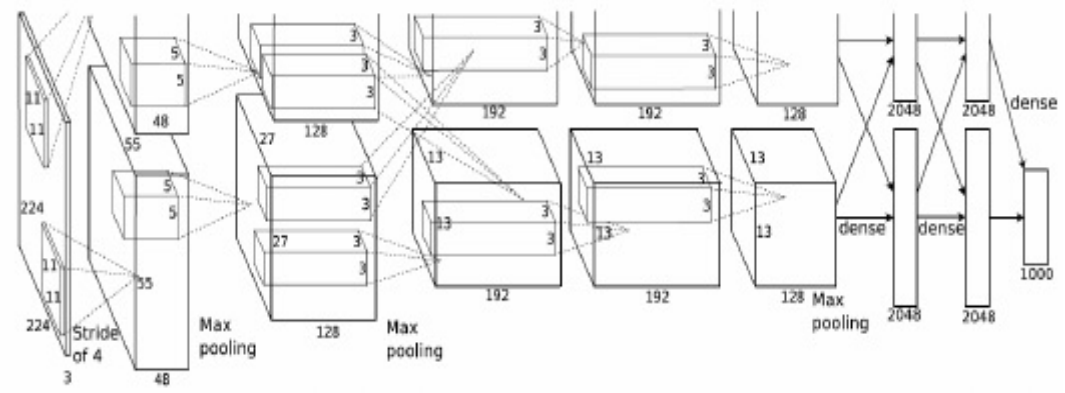

FIGURE 4.6 – Architecture de AlexNet[21]

Simonyan et Zisserman ont utilisé une configuration plus profonde d'AlexNet[21], et ils l'ont proposé comme VGGNet[30]. Ils ont utilisé de petits filtres de taille 3×3 pour toutes les couches et ont rendu le réseau plus profond en gardant les autres paramètres fixes. Ils ont utilisé un total de 6 configurations CNN différentes : A, A-LRN, B, C, D (VGG16) et E (VGG19) avec $11, 11, 13, 16, 16, 19$ couches pondérées respectivement. La figure 4.7 montre la configuration du modèle D.

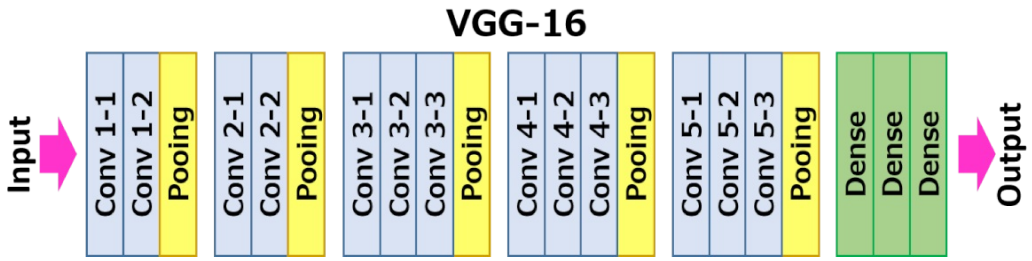

FIGURE 4.7 – Architecture de VGGNet (configuration D, VGG16)[32]

Le réseau qui a remporté le concours en 2014 est le réseau GoogLeNet[34], qui est un nouveau type de CNN, composé non seulement de couches de convolution et de pooling successives, mais également de nouveaux modules appelés Inception, qui sont une sorte de réseau dans le réseau. Un exemple est représenté à la figure 4.8.

Les innovations les plus récentes concernent les réseaux ResNet[34]. L'originalité des ResNets est d'ajouter une connexion reliant l'entrée d'une couche (ou un ensemble de couches) à sa sortie. Afin de réduire le nombre de paramètres, les ResNets n'ont pas de couches entièrement connectées. GoogleNet et ResNet sont beaucoup plus profonds que le CNN précédent, mais contiennent beaucoup moins de paramètres. Ils sont néanmoins beaucoup plus coûteux en mémoire que CNN plus classique, comme VGG ou AlexNet.

4.3 Deep Residuel Learning

Dans cette section, on va parlé plus profond sur le concept Residuel dans CNN.

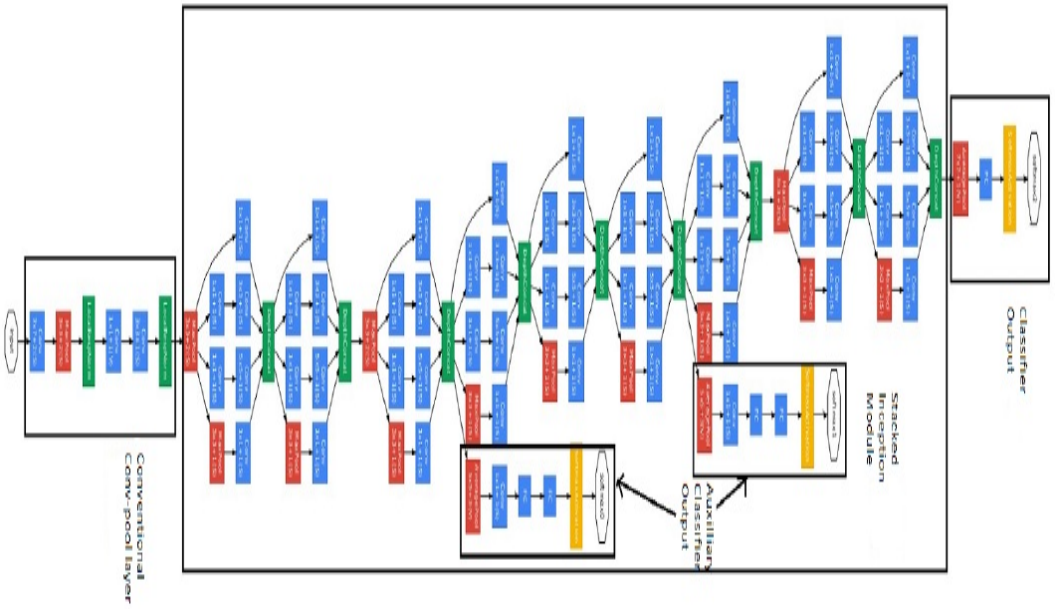

FIGURE 4.8 – Architecture de GoogleNet[34]

4.3.1 Resnet

He et al. expérimenté qu'un CNN plus profond avec plus de couches souffre d'un problème de vanishing gradient. Bien que ce problème soit géré par une initialisation normalisée et intermédiaire, le modèle plus profond montre des performances faibles sur les erreurs d'apprentissage et de validation et il n'est pas causé par un sur-apprentissage. Cela indique que l'optimisation d'un réseau plus profond est difficile. Pour résoudre ce problème, les auteurs ont utilisé un modèle moins profond pré-apprentis avec des couches supplémentaires pour effectuer identity mapping. Pour que les performances d'un réseau plus profond et d'un réseau moins profond soient similaires. Ils ont proposé un framework d'apprentissage résiduel profond[17] comme solution au problème de dégradation. Ils ont inclus residual mapping $(H(x) = F(x) + x)$ au lieu de mapping sous-jacente souhaitée $(H(x))$ dans leur réseau et ont nommé leur modèle ResNet[17].

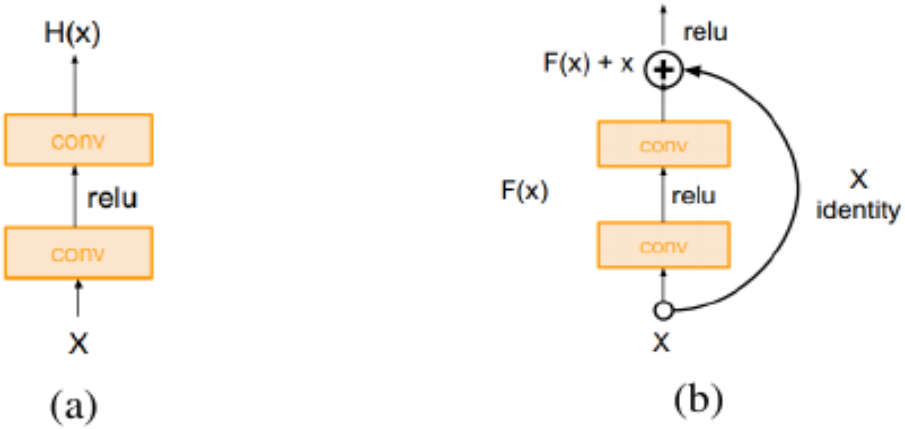

FIGURE 4.9 – (a) Couche simple (b) Bloc résiduel[17]

L'architecture ResNet se compose de blocs résiduels empilés de 3×3 couches convolutionnelles. Ils ont périodiquement doublé le nombre de filtres et utilisé une strides de 2. Les figures a et b de 4.9 montrent une couche simple et un bloc résiduel. Comme première couche, ils ont utilisé une couche conv 7×7. Ils n'ont pas utilisé de couches entièrement connectées à la fin. Ils ont utilisé différentes profondeurs $(34, 50, 101et152)$ ResNet dans la compétition ILSVRC-2014. Pour le CNN avec une profondeur supérieure à 50, ils ont utilisé une couche bottleneck pour réduire la dimensionnalité et améliorer l'efficacité en tant que GoogLeNet. Leur conception de bottleneck est constitué de $1 \times 1, 3 \times 3$ et 1×1 couche de convolution. Bien que le réseau 152 couches ResNet soit 8 fois plus profond que les réseaux VGG, il est moins complexe que les réseaux VGG $(16/19)$[32].

4.3.2 Residual Learning

Considérons $H(x)$ comme un mappage sous-jacent devant être ajusté par quelques couches empilées (pas nécessairement le réseau entier), avec x désignant les entrées de la première de ces couches. Si l'on suppose l'hypothèse que plusieurs couches non linéaires peuvent approcher asymptotiquement des fonctions complexes, alors cela équivaut à supposer qu'elles peuvent approximer asymptotiquement les fonctions résiduelles, c'est-à-dire $H(x) - x$ (en supposant que l'entrée et la sortie sont de mêmes dimensions). Ainsi, plutôt que de nous attendre à ce que les couches empilées approchent $H(x)$, nous laissons explicitement ces couches approcher une fonction résiduelle $F(x) := H(x) - x$. La fonction d'origine devient ainsi $F(x) + x$. Bien que les deux formes devraient être capables d'approximer asymptotiquement les fonctions souhaitées (l'hypothèse), la facilité d'apprentissage peut être différente[17].

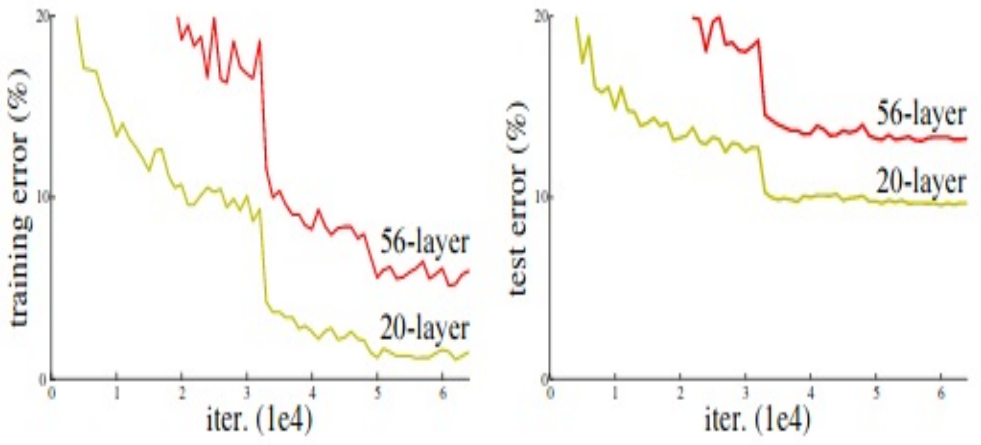

FIGURE 4.10 – Erreur d'apprentissage (à gauche) et erreur de test (à droite) sur CIFAR-10 avec des réseaux simples à 20 et 56 couches.[17]

Cette reformulation est motivée par les phénomènes contre-intuitifs sur le problème de la dégradation comme le montre la figure 4.10, le réseau plus profond a une erreur d'apprentissage et de test plus élevée. si les couches ajoutées peuvent être construites en tant que d'identity mapping, un modèle plus profond ne devrait pas avoir d'erreur d'apprentissage supérieure à son homologue moins profond. Le problème de dégradation suggère que les solveurs pourraient avoir des difficultés d'approximation d'identité mapping par multiples couches non linéaires. Avec la reformulation de l'apprentissage résiduel, si les identity mappings sont optimales, les solveurs peuvent simplement conduire les poids des multiples couches non linéaires vers zéro pour approcher les identity mappings[17].

4.4 Notre modele R-CNN

En général, les vidéos DeepFake sont de courte durée et de faible résolution. Par conséquent, nous pouvons logiquement déduire que si une courte vidéo contient plus d'images floues, en particulier cela se produit autour de la zone du visage, indiquant une forte probabilité que cette vidéo soit une vidéo synthétisé. La plupart des vidéos générées par les applications DeepFake présentent certains défauts. Tels que des artefacts visibles, une résolution distincte autour de la zone du visage et un changement soudain de couleur lorsque le module a un mouvement brusque du visage. Ces défauts sont causés par l'algorithme d'apprentissage DeepFake lorsque l'ensemble de données d'apprentissages est insuffisant pour couvrir tous les angles ou l'éclairage du visage. D'autre part, la plupart des utilisateurs des applications de DeepFake ne possèdent pas assez de ressources de calcul ou le temps de produire un modèle bien apprentis.

Pour detecter le DF, nous avons proposé une architecture avec des couches simples puis ajouter des blocs résiduels pour améliorer notre modèle, la figure 4.11 suivante représente les deux blocs proposés :

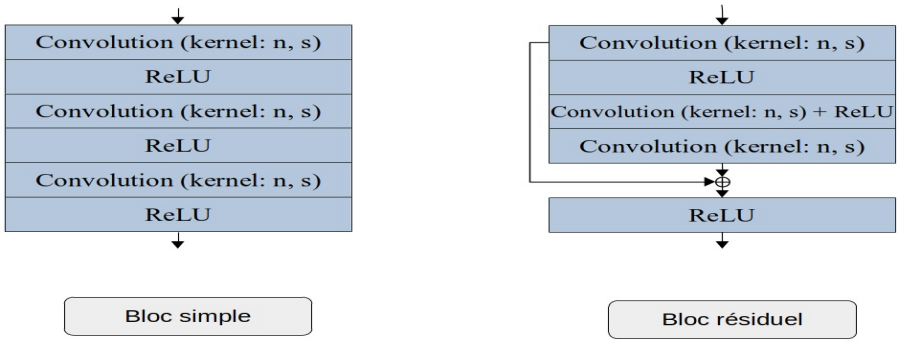

FIGURE 4.11 – Blocs proposés simple et avec résiduel.

Les deux blocs se composes de 3 couches convolutionnelles avec des fonctions d'activations ReLU et padding *same*, les symboles n et s représentent le nombre et la dimension des filtres. Dans le bloc résiduel, la sortie de premier couche de convolution est ajouter à la sorite de dernier couche de convolution, une fonction d'activation ReLU est appliqué au résultat obtenu.

L'architecture global de notre modèle 4.12 est composé de 5 blocs simples (modèle 1) ou résiduels (modèle 2) avec : $s = (1, 3, 3, 6, 9)$ et $n = (16, 32, 32, 64, 128)$, aprés chaque bloc, un couche de Max pooling est ajouté (sauf le premier bloc).

Deux couches entièrement connectées sont ajoutées, l'une avec 32 noeuds et le dernier avec un seul noeud suivi par la fonction Sigmoid pour obtenir le résultat de classification (classification binaire). Nous avons 4.7 millions de paramètres pour chaque modèle.

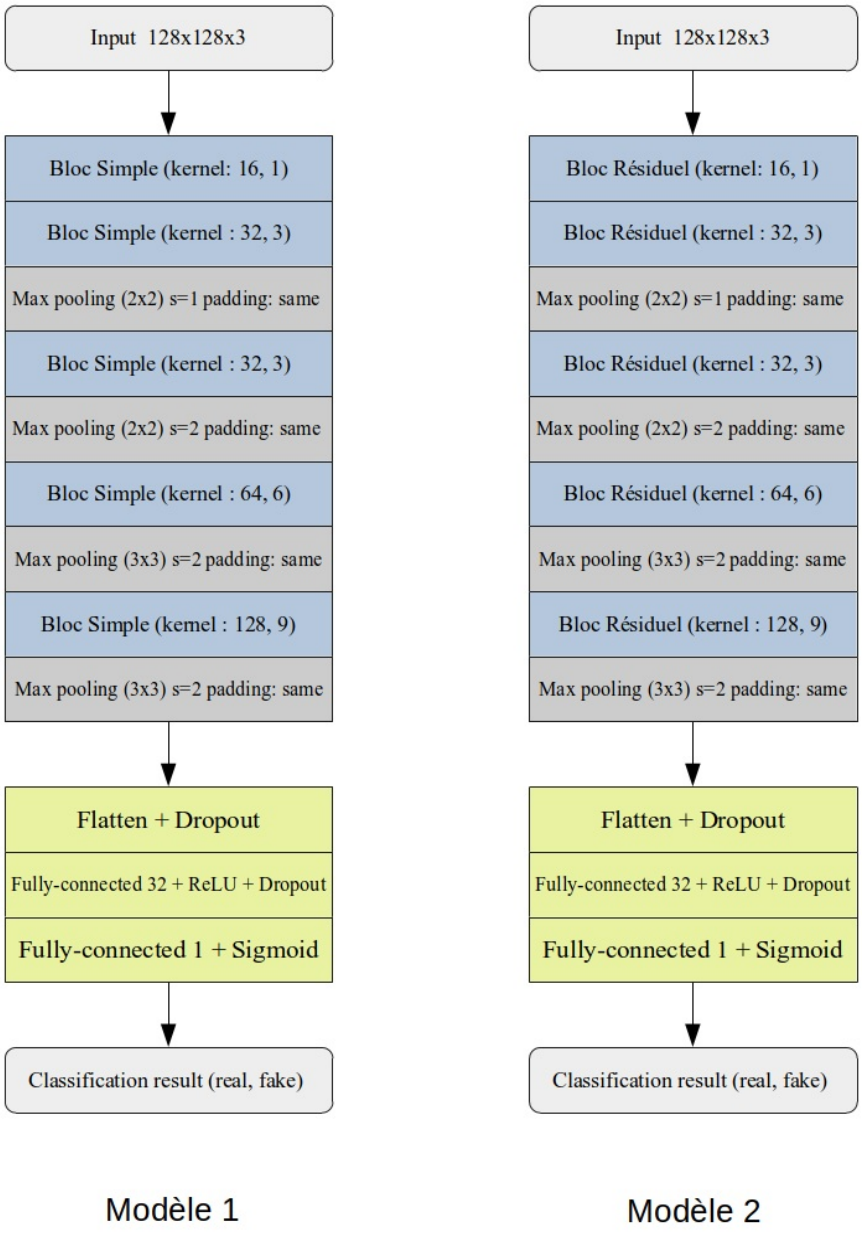

FIGURE 4.12 – L'architecture proposé avec et sans résiduel.

4.5 Conclusion

La conception de l'architecture de modéle est important pour améliorer les résutats de classification, mais les autres facteurs comme les datasets et les paramètres d'apprentissage jouent un rôle trés important, le choix de dataset et les expérimentations pour optimiser le modèle sont présentés dans le chapitre d'implémentation.

Chapitre 5

Implémentation

C'est l'innovation qui fait la différence entre un leader et un suiveur.

Steve Jobs - The Innovators

5.1 Introduction

Dans le chapitre précédent, nous avons vu l'architecture CNN proposé pour détecter les DFs, mais avant de valider notre modèle, nous avons fait une série d'expérimentations pour choisir les meilleurs hyper-paramétrés afin optimiser la performance du modèle.
Dans ce qui suitent, on va commencer par la description des utiles d'implémentations, le choix de Dataset et les paramètres de classification, après on va passé à l'étape des expérimentations et la comparaison avec d'autres architectures.

5.2 Outils d'implémentation

Pour implémenter et optimiser le modèle proposé, nous avons maintenant des frameworks de DL open source faciles à utiliser qui visent à simplifier la mise en œuvre des modèles complexes et à grande échelle.

5.2.1 Les frameworks existantes

Une framework de DL est une interface, une bibliothèque ou un outil qui nous permet de construire des modèles de DL plus facilement et plus rapidement, sans entrer dans les détails des algorithmes sous-jacents. Ils fournissent un moyen clair et concis de définir des modèles à l'aide d'une collection de composants préconfigurés et optimisés.
Parmi les frameworks les plus utilisés, on peut citer les suivants :

5.2.1.1 TensorFlow

TensorFlow a été développé par des chercheurs et des ingénieurs de l'équipe Google Brain. C'est la bibliothèque la plus utilisée dans le domaine du DL. L'une des raisons pour lesquelles TensorFlow est si populaire est la prise en charge de plusieurs langages pour créer des modèles d'apprentissage profond, tels que Python, C++ et R.
L'architecture flexible de TensorFlow nous permet de déployer nos modèles d'apprentissage en profondeur sur un ou plusieurs processeurs (ainsi que sur des GPU). Voici quelques cas d'utilisation populaires de TensorFlow :

- Applications textuelles : détection de langue, résumé de texte.
- Reconnaissance d'image : sous-titrage d'image, reconnaissance faciale, détection d'objets.
- Reconnaissance sonore.
- Analyse des séries chronologiques.
- Analyse vidéo.

5.2.1.2 Keras

Keras est écrit en Python et peut s'exécuter sur TensorFlow (ainsi que CNTK et Theano). C'est une API de haut niveau, développée dans le but de permettre une expérimentation rapide. Les réseaux de neurones convolutifs et les réseaux de neurones récurrents sont pris en charge par Keras. Il fonctionne de façon transparente sur les processeurs ainsi que les GPU. Keras possède plusieurs architectures, mentionnées ci-dessous, pour résoudre une grande variété de problèmes (ex : classification des images).

- VGG16.
- VGG19.
- InceptionV3.
- Mobilenet, .. etc.

5.2.1.3 PyTorch

PyTorch est un port d'accès au framework d'apprentissage profond Torch qui peut être utilisé pour créer des réseaux de neurones profonds et exécuter des calculs de tenseurs. Torch est basé sur Lua tandis que PyTorch fonctionne sur Python.
PyTorch est un package Python qui fournit des calculs Tensor. Les tenseurs sont des tableaux multidimensionnels tout comme les ndarrays de numpy qui peuvent également fonctionner sur GPU. Nous pouvons travailler sur toutes sortes de défis d'apprentissage profond à l'aide de PyTorch, notamment :

- Images (détection, classification, etc.)
- Texte (NLP).
- Apprentissage par renforcement.

5.2.1.4 Caffe

Caffe est un autre framework d'apprentissage profond populaire orienté vers le domaine du traitement d'image. Il a été développé par Yangqing Jia lors de son PhD à l'Université de Claifornia, Berkeley. Il fournit un support solide pour les interfaces comme C, C

++, Python, MATLAB ainsi que la ligne de commande traditionnelle. Le framework *Caffe Model Zoo* nous permet d'accéder à des réseaux, des modèles et des poids pré-entraînés qui peuvent être appliqués pour résoudre des problèmes de DL. Ces modèles fonctionnent sur les tâches ci-dessous :

- Régression simple.
- Classification visuelle à grande échelle.
- Réseaux Siamese pour la similitude d'images.
- Applications vocales et robotiques

5.2.1.5 Deeplearning4j

Deeplearning4j est implémenté en Java. Il utilise la bibliothèque de tenseurs appelée *ND4J* qui offre la possibilité de travailler avec des tableaux à n dimensions (également appelés tenseurs) et permet également l'utilisation des GPUs. Deeplearning4j traite la tâche de chargement des données et d'apprentissage comme des processus séparés. Différents types de données sont supportées :

- Images.
- csv.
- texte brut, etc.

Plusieurs types de modèles peut etre créer à l'aide de Deeplearning4j comme CNNs, RNNs, LSTMs et autres.

5.2.2 Environnement d'implémentation

Nous avons implémentés notre modèle et faire toutes les expérimentation en google cloud et à l'aide de plateform Kaggle (www.kaggle.com) avec les paramètres suivants :

Langage de programmation Python 3 (www.python.org).

Framework Keras avec Tensorflow comme backend (www.tensorflow.org).

Hardware spécification

— **OS :** Linux 4.19.112+ PDT 2020 GNU/Linux x86_64.
— **CPU :** Intel(R) Xeon(R) CPU @ 2.00GHz
— **RAM :** 13 GB
— **GPU :** NVIDIA TESLA P100 GPU
— **GPU Memory :** 16 GB

.

5.3 Choix de Dataset

Le dataset utilisé est *Deepfakes Dataset* par Paolo Mazza (www.kaggle.com/meraxes10). Il a été extrait le premier frame pour tous les vidéos de Deepfake Detection Challenge Dataset (DFDC 2.2). L'étape d'extraction des visages est realisé avec l'utilité facenet-pytorch (www.github.com/timesler/facenet-pytorch). Le dataset contient 119122 images 224×224 (Fake :84%, Real :16%).

5.4 Configuration de classification

La configuration suivante a été considéré dans toutes les étapes de l'apprentissage de notre modèle :

Prétraitement des données de dataset Nous avons utilisé 19148 images réeles et 19148 images fausses, l'ensemble des données est divisé en deux sous-ensembles où 80% représente l'ensemble d'apprentissage (sous-ensemble destiné à l'apprentissage du modèle) et le reste est l'ensemble d'évaluation (sous-ensemble destiné à l'évaluation). La taille des images est 128×128.

L'apprentissage de modèle Puisque le type de classification est binaire, nous avons choisi la fonction *binary-crossentropy* 5.4 pour calculer le taux d'erreur pour chaque itération de l'apprentissage.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Pour l'optimisation de modèle, l'algorithme Adam[19] a été utilisé pour mettre à jour les poids du réseau durant l'apprentissage du modèle.

Le taux d'apprentissage (learning rate) avec une valeur initiale de 10^-4 est divisé par 10 si la précision de l'apprentissage n'augmente durant deux epochs successifs jusqu'à la valeur minimale 10^6.

Les autres paramètres ont été fixer aprés les expérimentations.

5.5 Expérimentations

Nous avons faire plusieurs expérimentations avec des diffèrents batch size et dropout(problème de sur-apprentissage), nous avons aussi ajouté des blocks résiduels et comparé le modèle avec et sans ces blocks.
La précision 5.5 (accuracy) et le taux d'erreur (Loss) 5.4 d'apprentissage et de validation sont utilisés pour évaluer la performance du modèle.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP :vrai positive, TN : vrai négative, FP : faux positive, FN : faux négative.

5.5.1 Epochs, Batch Size et Iterations

A cause du taille de dataset et les limitations de mémoire de l'unité de calcul utilisée pour l'apprentissage, il n'est pas possible de prendre toutes les données d'apprentissage dans un algorithme en une seule passe dans la plupart des cas. Epochs, Batchsize et Iterations sont des terminologie nécessaire pour mieux comprendre comment les données sont mieux divisées en plus petits morceaux.

Epoch s'écoule lorsqu'un ensemble de données (dataset) entier est transmis à travers le réseau de neurones et les poids sont mis à jour exactement une fois. Si l'ensemble de données ne peut pas être transmis à l'algorithme en une seule fois, il doit être divisé en mini-batchs.

Batch size est le nombre total d'échantillons d'apprentissage présents dans un seul mini-batch.

Iteration indique le nombre de mises à jour des paramètres de réseau.

5.5.2 Batch size

Dans notre cas, c'est le nombre des images apprentis dans une itération avant la mise à jour des poids de réseau.

La figure 5.1 présente la précision et le taux d'erreur d'apprentissage optenues avec batch-size 64, 128, 256

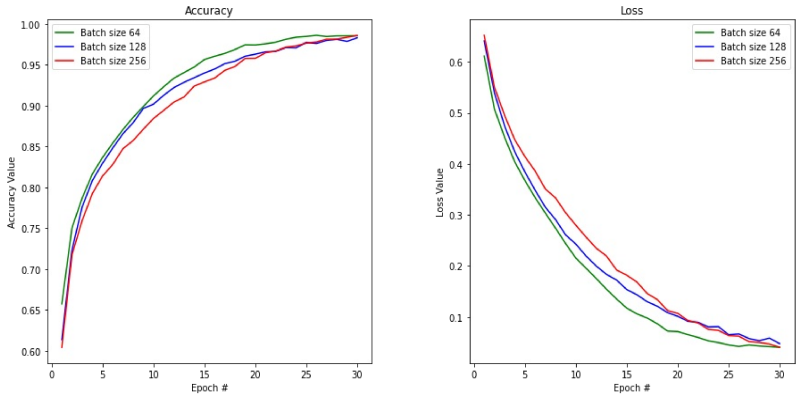

FIGURE 5.1 – La précision et le taux d'erreur d'apprentissage avec diffèrents batch-size.

Pour la validation, les résultats dans la figure 5.2 sont obtenues.

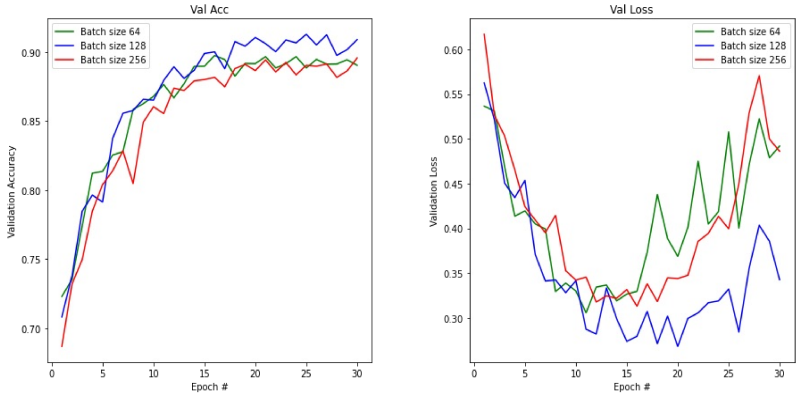

FIGURE 5.2 – La précision et le taux d'erreur de validation avec différents batch-sizes.

Après la comparaison, le batch-sise avec la valeur 128 donne le meilleur résultat.

5.5.3 Dropout

Afin d'improuver les résultats de validation, nous avons utilisé la technique de régulation Dropout, L'idée clé est de supprimer au hasard des unités (avec leurs connexions) du réseau de neurones pendant l'apprentissage. Les résultats dans la figure 5.3 représentent la précision et le taux d'erreur durant la phase de validation avec les taux de dropout 0, 0.25, 0.5.

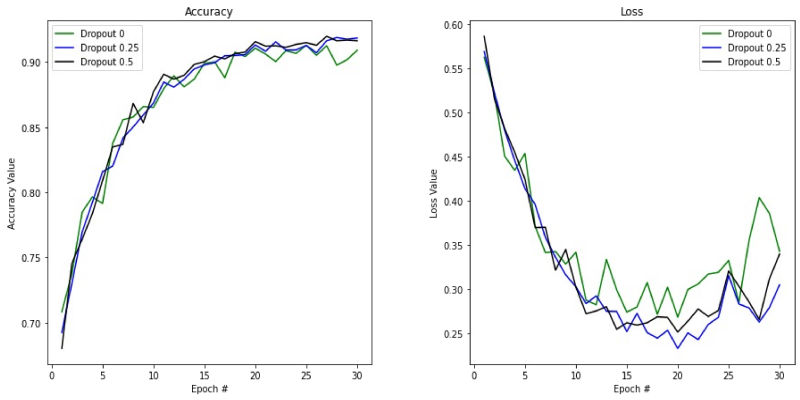

FIGURE 5.3 – La précision et le taux d'erreur de validation avec diffèrents taux de dropout.

Il est claire que l'utilisation de dropout minimise le taux d'erreur et le broblème de sur-ajustement. on remarque également une amélioration de la précision. Le taux 0.25 est choisis pour notre modèle.

5.5.4 Utilisation des blocs résiduels

Aprés l'utilisation des blocs résiduels au modèle, on obtient les résultats suivantes :

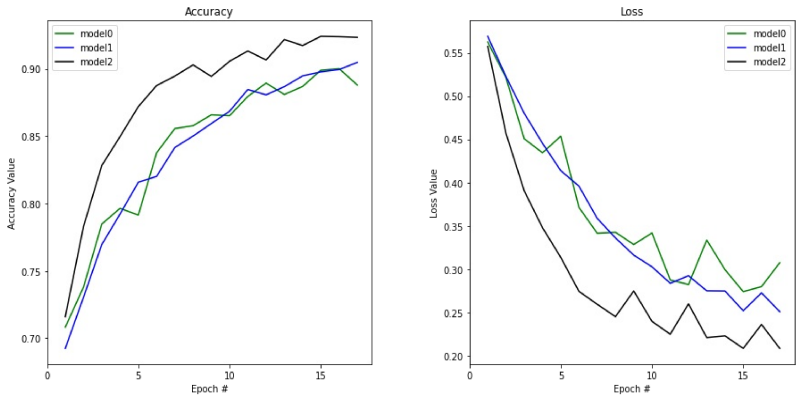

FIGURE 5.4 – Résultats de validation :model0(simlpe), model1(+ dropout), model2(+blocs résiduels)

L'utilisation des blocs résiduels improve la performance de modèle dans la précision et réduit le taux d'erreurs, l'effet apparaît aussi dans la rapidité d'apprentissage de modèle.

5.5.5 Résultats finaux

Les courbes 5.5, 5.6 et 5.7 représentent la précision et le taux d'erreur des trois modèles :

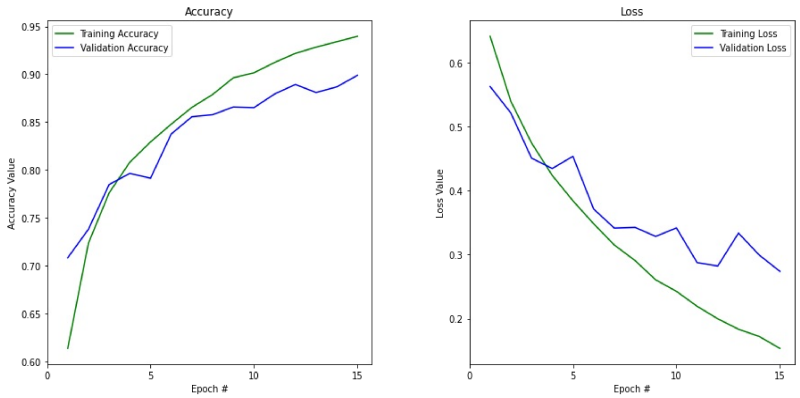

FIGURE 5.5 – Model0 : résultats d'apprentissage et de validation(précision et taux d'erreur)

Model 0 représente le modèle initial, on obtient les résultats dans le tableau 5.1 après 15 epochs :

Précision	**Taux d'erreur**
90.12 %	27.39 %

TABLEAU 5.1 – Précision et taux d'erreur de model 0.

Dans *Model 1*, la technique de régularisation Dropout est utilisé.

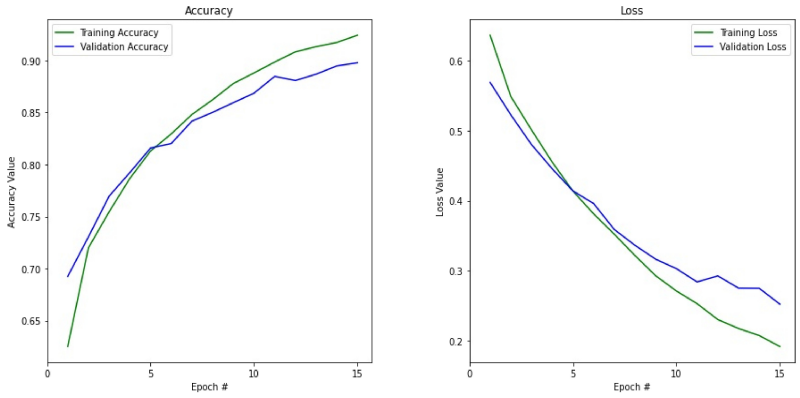

FIGURE 5.6 – Model1 : résultats d'apprentissage et de validation(précision et taux d'erreur)

Le meilleur résultat de modèle 1 est obtenue dans le tableau 5.2.

Précision	**Taux d'erreur**
91.31 %	25.18 %

TABLEAU 5.2 – Précision et taux d'erreur de model 1.

Finalement, le model 2 est l'architecture avec les blocs résiduels.

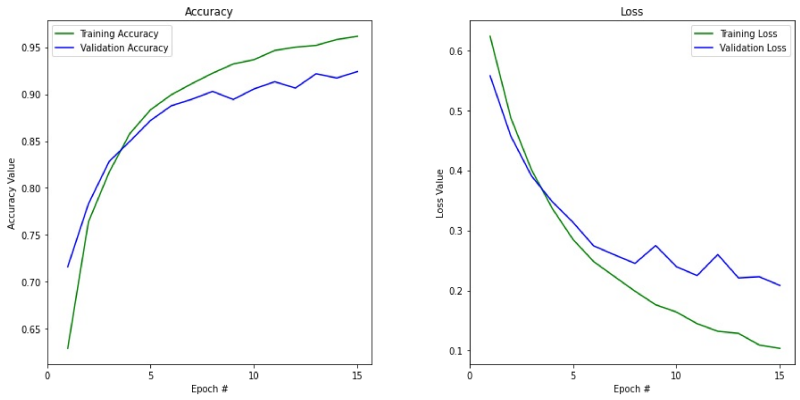

FIGURE 5.7 – Model2 : résultats d'apprentissage et de validation(précision et taux d'erreur)

La précision et le taux d'erreur de model 3 obtenue est dans le tableau 5.3

Précision	Taux d'erreur
92.41 %	20.83 %

TABLEAU 5.3 – Précision et taux d'erreur de model 2.

5.5.6 Configuration finale

Après les expérimentations, nous avons atteidre au configurations des paramètres dans le tableau 5.4 :

Batch size	Taux d'apprentissage	Taux de dropout	Nombres d'epochs	Modèle
128	0.0001	0.25	15	model 2

TABLEAU 5.4 – Les paramètres de modèle choisis.

5.6 Comparaisons avec autres architectures

Dans la figure 5.8, nous avons fait une comparaison en matière de précision avec quelques architectures classiques.

Les résultats obtenus montrent l'efficacité de notre model par rapport les autres architectures.

5.7 Utilisation de modèle

Notre objectif principale est de détecter les vidéos et les images de deepfakes, mais il est possible également d'utiliser le modèle dans le processus de création de deepfakes.

5.7.1 Processus de détection

Un étape de prétraitement est nécessaire pour faire la classification :

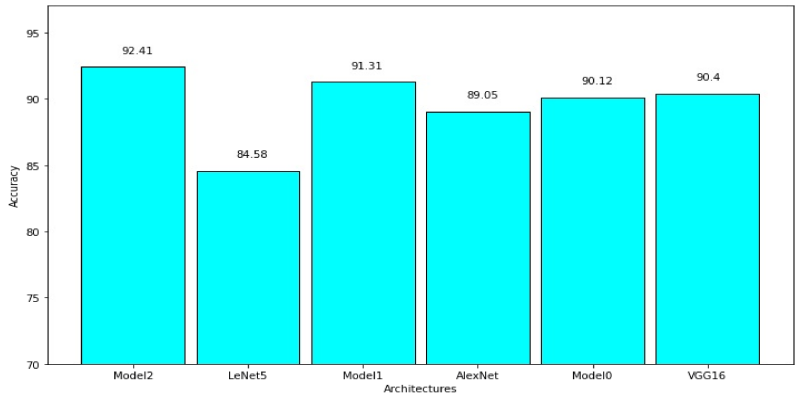

FIGURE 5.8 – Comparaison de précision entre les models proposés et autres architectures célèbres.

> Pour les vidéos, if faut convertir le vidéo à des images.

> Faire la detection des visages des persones dans les images.

Aprés le traitement de vidéo(image), on prend les images(visages) comme entrés à notre notre classificateur et faire la prédiction. Avec une seul image, la prédiction est donc le résultat de classification (réel ou fake), mais avec plusieurs images (dans le cas de vidéo) nous avons utilisé la formule suivante pour faire la classification :

$$resultat = \frac{NP_{p>0.5}}{NI}$$

où NP : le nombre des prédictions superieur à 0.5 et NI : le nombre total des prédictions.

5.7.2 Processus de Création

Pour la création des deepfakes, notre modèle proposé peut étre utilisé comme un discriminateur dans un architecture GAN, et donc améliorer la qualité des vidéos deepfakes. La figure 5.9 représente le resultat obtenu après le processus de creation de DF.

FIGURE 5.9 – Exemple de vidéo créée avec GAN (image gauche représente la vidéo manipulée).

5.8 Conclusion

Les expérimentations que nous avons faite avec les différents paramètres nous permettent d'améliorer la performance de notre modèle en matière de précision et taux d'erreur. L'utilisation de dropout permet d'augmenter la précision et de minimiser le taux d'erreur. Avec l'utilisation des liens résiduels au modèle, le modèle apprenti plus rapide et donne des résultats acceptables par rapport au modèle classique. Notre modèle donne de meilleure précision dans la détection de DF comparant aux autres architectures classiques (Alexnet, VGG16...).

Chapitre 6

Conclusion Générale

6.1 Conclusion

Les avancements de la technologie des caméras mobiles et la portée sans cesse croissante des médias sociaux et des portails de partage de médias ont rendu la création et la diffusion de vidéos numériques plus pratiques que jamais. Jusqu'à recement, le nombre de fausses vidéos et leur degré de réalisme étaient limités par le manque d'outils d'édition sophistiqués, le besoin des experts et le processus complexe impliqué. Cependant, le temps de fabrication et de manipulation des vidéos a considérablement diminué ces dernières années, grâce à l'accessibilité aux données d'entraînement à grand volume et à la puissance de calcul à haut débit, mais plus à la croissance des techniques d'apprentissage profond. Pour cela, il est trés important de trouver une solution efficaces pour détecter les vidéos manipulés qui representent une menace au sécurité de l'information.

Dans ce travail, nous avons proposé une solution pour détecter les vidéos et les images manipulés (Deepfakes), la méthode de détection est basé sur l'apprentissage profond ou plus précisement les réseaux de neurones convolutionnel, aprés l'optimisation de modele par des nombreuses expérimentations, nous avons atteindre a un classifieur avec des résultats acceptables en comparant avec les autres solutions existants.

La méthode proposé repose sur les imperfections des techniques de Deepfake utilisées, et restent efficaces jusqu'à ce que les attaquants comprennent et adaptent leurs méthodes pour passer sous les seuils de détection.

Finalement, il est nécessaire de connaitre que nous existons maintenant dans un monde où la fausse réalité est aussi réelle que la réalité, ai-technologie avait avancé au point de créer une fausse vidéo qui peut être si convaincante à l'œil humain, causant des dommages massifs à la société, la génération de deepfake et les modèles de détection progressent d'une manière parallèle, nous espérons que notre modèle a fait au moins une légère amélioration dans ce sujet de recherche particulière.

6.2 Travaux Futures :

Trouver une solution efficace pour un problème comme le deepfakes nécessite une connaissance très solide dans le domaine d'apprentissage profond et aussi dans la sécurité des informations.

Dans notre future travaux, l'hybridation entre ces deux domaines deviendra le nouvel axe pour limiter les dangers de Deepfakes.

Bibliographie

[1] Darius AFCHAR et al. « Mesonet : a compact facial video forgery detection network ». In : *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2018, p. 1-7.

[2] Saad ALBAWI, Tareq Abed MOHAMMED et Saad AL-ZAWI. « Understanding of a convolutional neural network ». In : *2017 International Conference on Engineering and Technology (ICET)*. IEEE. 2017, p. 1-6.

[3] Yoshua BENGIO, Yann LECUN et al. « Scaling learning algorithms towards AI ». In : *Large-scale kernel machines* 34.5 (2007), p. 1-41.

[4] Yoshua BENGIO et al. « A neural probabilistic language model ». In : *Journal of machine learning research* 3.Feb (2003), p. 1137-1155.

[5] Jinil Jang CHI WANG. « CS230 : Deep Face Swap with GAN ». In : *URL https ://cs230.stanford.edu* (2019).

[6] François CHOLLET. « Deep Learning with Python ». In : 2017.

[7] *Conrad Sanderson - VidTIMIT dataset*. URL : http://www.conradsanderson.id.au/vidtimit/.

[8] Harshavardhan CA DEEKSHITH SHETTY et al. « Diving Deep into Deep Learning : History, Evolution, Types and Applications ». In : ().

[9] Sergey DEMYANOV. « Regularization methods for neural networks and related models ». Thèse de doct. 2015.

[10] Brian DOLHANSKY et al. « The Deepfake Detection Challenge (DFDC) Preview Dataset ». In : *arXiv preprint arXiv :1910.08854* (2019).

[11] John DUCHI, Elad HAZAN et Yoram SINGER. « Adaptive subgradient methods for online learning and stochastic optimization ». In : *Journal of machine learning research* 12.Jul (2011), p. 2121-2159.

[12] Kunihiko FUKUSHIMA. « Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position ». In : *Biological cybernetics* 36.4 (1980), p. 193-202.

[13] Ian GOODFELLOW, Yoshua BENGIO et Aaron COURVILLE. *Deep learning*. MIT press, 2016.

[14] Ian GOODFELLOW et al. « Generative adversarial nets ». In : *Advances in neural information processing systems*. 2014, p. 2672-2680.

[15] David GÜERA et Edward J DELP. « Deepfake video detection using recurrent neural networks ». In : *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2018, p. 1-6.

[16] Haya R HASAN et Khaled SALAH. « Combating deepfake videos using blockchain and smart contracts ». In : *Ieee Access* 7 (2019), p. 41596-41606.

[17] Kaiming HE et al. « Deep residual learning for image recognition ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770-778.

[18] Chih-Chung HSU, Yi-Xiu ZHUANG et Chia-Yen LEE. « Deep fake image detection based on pairwise learning ». In : *Applied Sciences* 10.1 (2020), p. 370.

[19] Diederik P KINGMA et Jimmy BA. « Adam : A method for stochastic optimization ». In : *arXiv preprint arXiv :1412.6980* (2014).

[20] Marissa KOOPMAN, Andrea Macarulla RODRIGUEZ et Zeno GERADTS. « Detection of deepfake video manipulation ». In : *The 20th Irish Machine Vision and Image Processing Conference (IMVIP)*. 2018, p. 133-136.

[21] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON. « Imagenet classification with deep convolutional neural networks ». In : *Advances in neural information processing systems*. 2012, p. 1097-1105.

[22] Yann LE CUN et al. « Handwritten digit recognition : Applications of neural network chips and automatic learning ». In : *IEEE Communications Magazine* 27.11 (1989), p. 41-46.

[23] Yann LECUN et al. « Gradient-based learning applied to document recognition ». In : *Proceedings of the IEEE* 86.11 (1998), p. 2278-2324.

[24] Yuezun LI, Ming-Ching CHANG et Siwei LYU. « In ictu oculi : Exposing ai created fake videos by detecting eye blinking ». In : *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2018, p. 1-7.

[25] Yuezun LI et Siwei LYU. « Exposing deepfake videos by detecting face warping artifacts ». In : *arXiv preprint arXiv :1811.00656* (2018).

[26] Falko MATERN, Christian RIESS et Marc STAMMINGER. « Exploiting visual artifacts to expose deepfakes and face manipulations ». In : *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*. IEEE. 2019, p. 83-92.

[27] Thanh Thi NGUYEN et al. « Deep Learning for Deepfakes Creation and Detection ». In : *arXiv preprint arXiv :1909.11573* (2019).

[28] Chigozie NWANKPA et al. « Activation functions : Comparison of trends in practice and research for deep learning ». In : *arXiv preprint arXiv :1811.03378* (2018).

[29] Samira POUYANFAR et Shu-Ching CHEN. « T-LRA : Trend-based learning rate annealing for deep neural networks ». In : *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*. IEEE. 2017, p. 50-57.

[30] Karen SIMONYAN et Andrew ZISSERMAN. « Very deep convolutional networks for large-scale image recognition ». In : *arXiv preprint arXiv :1409.1556* (2014).

[31] Nitish SRIVASTAVA et al. « Dropout : a simple way to prevent neural networks from overfitting ». In : *The journal of machine learning research* 15.1 (2014), p. 1929-1958.

[32] Farhana SULTANA, Abu SUFIAN et Paramartha DUTTA. « Advancements in image classification using convolutional neural network ». In : *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE. 2018, p. 122-129.

[33] Ilya SUTSKEVER et al. « On the importance of initialization and momentum in deep learning ». In : *International conference on machine learning*. 2013, p. 1139-1147.

[34] Christian SZEGEDY et al. « Inception-v4, inception-resnet and the impact of residual connections on learning ». In : *Thirty-first AAAI conference on artificial intelligence*. 2017.

[35] Haohan WANG et Bhiksha RAJ. « On the Origin of Deep Learning ». In : (fév. 2017).

[36] Matthew D ZEILER et al. « On rectified linear units for speech processing ». In : *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, p. 3517-3521.

[37] Ying ZHANG, Lilei ZHENG et Vrizlynn LL THING. « Automated face swapping and its detection ». In : *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*. IEEE. 2017, p. 15-19.