

RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

ÉCOLE POLYTECHNIQUE
D'ABOMEY-CALAVI



DEPARTEMENT DE GENIE INFORMATIQUE ET TELECOMMUNICATIONS

Option: Réseaux Informatiques et Internet

MEMOIRE DE FIN DE FORMATION
POUR L'OBTENTION DU
DIPLOME D'INGENIEUR DE CONCEPTION

Thème :

Étude d'un système de vérification de signature dynamique avec les réseaux de neurones

Erick ADJE
erickadje96@gmail.com

Sous la supervision de :
Prof. Marc Kokou **ASSOGBA**, MC CAMES

Encadreur :
Dr. John **AOGA**

Année Académique : **2017-2018**

11^{e} **Promotion**

Table des matières

Dédicace **iv**

Liste des figures **vi**

Liste des tableaux **viii**

Liste des sigles et abréviations **ix**

Résumé **1**

Abstract **3**

Introduction **4**

1 État de l'art **7**

1.1 Généralités sur la signature 7

1.2 La signature manuscrite : Vue d'ensemble 8

1.2.1 Caractéristiques de la signature d'un humain 8

1.2.2 Utilité de la signature manuscrite 9

1.2.3 Les différents types de signature manuscrite 10

1.2.4 Mode de fonctionnement : statique, dynamique et hybride 11

1.3 Techniques de vérification de signature 12

1.3.1 Généralité 12

1.3.2 Système de vérification statique 13

1.3.3 Système de vérification dynamique 16

1.3.4 Application de la vérification de signature 18

1.4 Les réseaux de neurones 19

1.4.1 Machine Learning : Généralités 19

1.4.2 Neurone 25

1.4.3 Perceptron multi-couches 29

1.4.4 Apprentissage 31

1.5 Critiques de l'existant et contribution de notre travail 33

1.5.1 Critiques de l'existant 33

1.5.2 Contribution 33

2 Matériels et méthodes 35
2.1 Outils 35
2.1.1 Environnement de développement 35
2.1.2 Les données 38
2.1.3 Les caractéristiques extraites des données 40
2.1.4 Mesure de performance 47
2.2 Méthode d'étude 48
2.2.1 Évaluation de l'importance des caractéristiques 49
2.2.2 Étude de l'architecture du réseau neuronal optimal 49
2.2.3 Mise en place de l'application du système de vérification : cycle V 50

3 Résultats et discussions 52
3.1 Détails de l'implémentation d'un réseau de neurones 52
3.2 Validation des caractéristiques 54
3.3 Validation de l'architecture neuronale optimale 59
3.4 Comparaison à d'autres méthodes et à l'état de l'art 62
3.4.1 Comparaison de la méthode utilisée avec d'autres méthodes 62
3.4.2 Comparaison de notre étude à une autre utilisant la même méthode 64
3.4.3 Comparaison de notre étude à une étude de vérification statique 64
3.5 Aperçu du système final conçu 65
3.6 Discussion générale 68

Conclusion et perspectives 70

English version 71

3.7 Generality and current state of art 72
3.7.1 Nature of human signature 72
3.7.2 Signature verification system : modes of operation 73
3.7.3 Machine learning : generality 73
3.7.4 Neural network 77
3.8 Materials and methods 78
3.8.1 Tools 78
3.8.2 Study method 82
3.8.3 The different stages of our study 83
3.8.4 Establishment of our verification system 83
3.9 Results and discussions 84
3.9.1 Studies and discussions on the features 84

3.9.2 Studies on neurals network architectures 86
3.9.3 Comparison of our method with the k-NN and SVM methods 87
3.9.4 Comparison of our study to another using the same method 88
3.9.5 Comparison of our study to a static verification study 88

A Détail des tests effectués pour la sélection de caractéristiques et de l'architecture neuronale optimale **90**

Bibliographie **99**

Webographie **102**

Dédicace

A mon père Alexis A. ADJE

Pour avoir su m'inculquer les valeurs morales et le goût du travail bien fait.

A ma mère Alimatou SOUMANOU

Tu as toujours été là pour moi aussi bien dans les moments difficiles que dans les moments heureux. Chère maman, les mots me manquent pour t'exprimer toute ma gratitude.

A ma sœur Christelle et mes frères Franck, Michaël

Je salue ici la tendre complicité qui a toujours existé entre nous et qui nous a permis de surmonter tant d'obstacles. Que le Seigneur nous unisse davantage afin de relever ensemble d'autres défis qui nous attendent.

Le couronnement de ce travail n'a été possible que grâce au concours direct ou indirect de certaines personnes. J'exprime toute ma reconnaissance à tous et à toutes. Je tiens sincèrement à remercier :

- Tous les enseignants de l'EPAC pour avoir accepté de partager une partie de leurs connaissances avec moi.
- Dr Léopold DJOGBE, Chef du département de Génie Informatique et Télécommunications (GIT) de l'EPAC et toute son administration, pour m'avoir permis de suivre cette formation de qualité.
- Prof. Marc Kokou ASSOGBA qui, malgré ses nombreuses occupations, a su se rendre disponible pour superviser ce travail ;
- Ing. John AOGA, pour son encadrement et pour m'avoir accompagné tout au long de ce travail.
- Dr. Ratheil V. HOUNDJI, pour ses apports dans l'amélioration du travail.
- Mr Probus KIKI, pour avoir accepté m'accueillir dans son entreprise et pour tout le savoir-faire et les informations mises à ma disposition durant le stage.
- Le laboratoire du LETIA, pour m'avoir accueilli en son sein afin d'effectuer des recherches relatives au travail.
- Mlle Carelle ZINSOU pour son soutien et ses corrections apportées dans la rédaction.
- M. Jean-Baptiste SOSSOU, M. Harold KIOSSOU, Mlle Faïzath ZOUMAROU WALIS et M. Jacques AKOUEIKOU pour leurs apports et pour avoir été de fidèles compagnons tout au long de mon cursus universitaire.
- Tous mes camarades étudiants de promotion pour leur sens de partage.

Table des figures

1.1 Image d'une tablette wacom et d'un stylet numériseur [39] 11
1.2 Schéma fonctionnel d'un système de vérification de la signature 12
1.3 Vue d'ensemble du système de vérification de signature statique 13
1.4 Schéma synoptique des techniques d'apprentissage automatique 20
1.5 Schéma d'ensemble de quelques algorithmes d'apprentissage 21
1.6 Exemple de deux classes linéairement séparables par la méthode de la machine à vecteur de support 22
1.7 Exemple des trois(3) voisins les plus proches 23
1.8 Matrice de confusion à deux classes 24
1.9 Schéma d'un neurone naturel [29] 25
1.10 Schéma d'un neurone artificiel à 3 entrées [30]. 26
1.11 Représentation de la fonction sigmoïde et sa dérivée 27
1.12 Représentation de la fonction tangente hyperbolique et sa dérivée 28
1.13 Représentation de la fonction ReLU et sa dérivée 29
1.14 Exemple d'un réseau neurone à deux couches cachées 30

2.1 Signature, coordonnées x,y, pression, azimut et altitude associés à la signature . . 39
2.2 Phases d'apprentissage et d'évaluation par les réseaux de neurones 48
2.3 Le cycle en V [50] 50

3.1 Code python de la mise en œuvre d'un réseau de neurones avec Keras 53
3.2 Deux échantillons de la signature authentique de l'utilisateur 1 57
3.3 Deux échantillons de la signature authentique de l'utilisateur 4 57
3.4 Page de formulaire pour un entraînement 65
3.5 Page de soumission réussie et d'entraînement en cours 66
3.6 Page d'entraînement réussi et d'évaluation du modèle 66
3.7 Page de prédiction d'une signature 67
3.8 Page de prédiction réussie et évaluation de la signature 68
3.9 Machine Learning Techniques 74
3.10 Machine Learning Algorithms 75
3.11 Architecture of a neural network [53] 77
3.12 Learning and evaluation phases by neural networks 82

Liste des tableaux

2.1 Comparatif des langages de programmation Python, Java, Matlab et R 36
2.2 Ensemble des caractéristiques extraites 43

3.1 Résultats des valeurs minimales des métriques sur l'étude des caractéristiques ("__" signifie qu'on considère les caractéristiques de la ligne précédente) 56
3.3 Résultats des valeurs minimales des métriques sur l'étude de certaines caractéristiques 58
3.4 Résultats des valeurs minimales des métriques sur l'étude de l'architecture neuronale 60
3.5 Tableau comparatif des performances entre les méthodes de réseau de neurones, de machine à vecteur de support et de K-voisins les plus proches 63
3.6 Tableau comparatif de notre étude à celle de Shalini B. et al [25] 64
3.7 Study on statics features 85
3.8 Study on statics and dynamics features 85
3.9 Results of minimal values of metrics on the study of neural architecture 87

A.1 Evaluation des caractéristiques de 1 à 7 et de 1 à 12 90
A.2 Evaluation des caractéristiques de 1 à 14 et de 1 à 18 90
A.3 Evaluation des caractéristiques de 1 à 20 et de 1 à 22 91
A.4 Evaluation des caractéristiques de 1 à 24 et de 1 à 26 91
A.5 Étude des caractéristiques 1 à 27 91
A.6 Étude des caractéristiques 15 à 18 91
A.7 Étude des caractéristiques 23 à 27 92
A.8 Etude de l'architecture neuronale : nombre de couches cachées = *2* et *3*, taille du batch = *3* et *5*, optimiseur = *adam* et *rmsprop*, nombre d'époques = *1000*, *2000* et *5000* 92
A.9 Etude de l'architecture neuronale : nombre de couches cachées = *2*, taille du batch = *3* et *5*, optimiseur = *sgd*, nombre d'époques = *1000*, *2000* et *5000* 93
A.10 Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *3*, optimiseur = *sgd*, nombre d'époques = *1000* et *2000* 93
A.11 Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *5*, optimiseur = *sgd*, nombre d'époques = *1000* 94
A.12 Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *5*, optimiseur = *sgd*, nombre d'époques = *2000* 94

A.13 Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *3* et *5*, optimiseur = *sgd*, nombre d'époques = *5000* 95
A.14 Performance sur dix utilisateurs avec la méthode des réseaux de neurones 95
A.15 Performance sur dix utilisateurs avec la méthode de la machine à vecteur de support 96
A.16 Performance sur dix utilisateurs avec la méthode des K-voisins les plus proches avec K = 3 96
A.17 Tableau comparatif de chaque méthode pour chaque utilisateur 97

Liste des sigles et abréviations

AER : Acceptance Error Rate

AUC : Area Under the Curve

FAR : False Acceptance Rate

FRR : False Rejection Rate

SVC : Signature Verification Competition

SVM : Support Vector Machine

K-NN : K-Nearest Neighbors

Résumé. La **signature** est le graphisme par lequel une personne s'identifie dans un acte et par lequel elle exprime son approbation au contenu d'un document. Ainsi elle est utilisée quotidiennement sur des documents officiels et privés tels que les conventions, les chèques, etc, d'où son importance dans tous les secteurs d'activité. Ces signatures n'étant pas à l'abri des falsifications, une protection adéquate des signatures personnelles est nécessaire. S'assurer de l'authenticité de la signature à l'œil n'est pas efficace surtout lorsque la falsification est réalisée par un professionnel. Plusieurs systèmes de vérification de signature ont été développés, basés pour la plupart sur des techniques d'apprentissage automatique. Cependant elles ne sont pas forcément efficaces car les caractéristiques considérées pour procéder à la vérification ne sont pas révélatrices d'informations cruciales permettant d'identifier efficacement les traits d'une signature authentique. C'est dans cette optique que nous procédons suivant une nouvelle approche en nous basant sur de nouvelles caractéristiques afin de mettre en place un système de vérification de signature dynamique automatisé et performant. Pour y arriver, nous utilisons de l'apprentissage en profondeur afin d'apprendre à identifier les signatures authentiques et falsifiées sur une base de données de signatures dynamiques. Notre approche consiste dans un premier temps à extraire des caractéristiques morphologiques pour apprendre la forme de la signature et aussi des caractéristiques abstraites (vitesse de signature, vitesse angulaire, accélération...) pour apprendre le comportement des gens quand ils signent. Une fois l'extraction des caractéristiques effectuée, nous avons pu déterminer leurs impacts sur les performances de notre système, ensuite nous avons mené une étude sur l'architecture de réseau de neurones et ses paramètres optimaux ce qui a permis d'obtenir un modèle d'authentification de signature avec les performances de 1,25% pour le FAR(False Acceptance Rate) et de 0,75% pour le FRR(False Rejection Rate). Ce modèle a été utilisé ensuite pour mettre en place un système de vérification performant capable de détecter pour une personne si une signature est fausse ou pas. Le système final obtenu ne se limite qu'à des signatures manuscrites électroniques.

LISTE DES TABLEAUX

Mots clés : *Signature, réseau de neurones, système de vérification dynamique*

Abstract. The **signature** is the graphic design by which a person identifies himself in an act and by which he expresses his approval to the content of a document. Thus it is used daily on official and private documents namely checks, agreements etc, hence its importance in all sectors of activity. Since these signatures are not immune from falsification, adequate protection of personal signatures is necessary. Ensuring the authenticity of the signature with the naked eye is not very effective especially when the forgery is performed by a professional.Several signature verification systems have been developed, mostly based on machine learning techniques. However, they are not necessarily effective because the characteristics considered for the verification are not revealing crucial information making it possible to effectively identify the features of an authentic signature. It is with this in mind that we will proceed with a new approach based on new features in order to implement an automated and efficient online signature verification system. To achieve this, we use deep learning to learn to identify authentic and falsified signatures on an online signature database. Our approach is initially to extract morphological characteristics to learn the shape of the signature and also abstract characteristics (signature speed, angular velocity, acceleration ...) to learn the behavior of people when they sign. Once the features were extracted, we were able to determine their impact on the performance of our system, then we carried out a study on the neural network architecture and its optimal parameters which allowed us to obtain a model of signature authentication with performance of 1.25% for the FAR (False Acceptance Rate) and 0.75% for the FRR (False Rejection Rate). This model was then used to set up an efficient verification system capable of detecting for a person whether a signature is false or not. The final system obtained is limited only to electronic handwritten signatures.

Keys words: *Signature*, *Neural Network*, *dynamic system verification*

Introduction

Il existe plusieurs alternatives qui sont utilisées pour identifier les humains. Ces techniques sont notamment la reconnaissance des yeux, la reconnaissance des visages, la reconnaissance des empreintes digitales mais on se rend compte que l'utilisation de la signature reste le moyen d'authentification le plus utilisé. Le problème se pose lorsqu'une personne décide d'imiter la signature d' une autre personne dans le but de faire une fraude ou une fausse déclaration. Par conséquent, une protection adéquate des signatures personnelles est nécessaire afin d'affirmer de l'authenticité d'une signature, l'œil humain ne peut être efficace car soumis à plusieurs paramètres.

Plusieurs systèmes de reconnaissance et de vérification de signature manuscrite ont été développés de nos jours. Ces systèmes sont divisés en deux catégories à savoir le système de reconnaissance et de vérification statique et le système de reconnaissance et de vérification dynamique. Un système de vérification de signature statique utilise des caractéristiques extraites de l'image de signature numérisée pour la vérification. L'approche dynamique utilise une tablette électronique et un stylet, l'ensemble connecté à un ordinateur pour extraire des informations sur une signature et prend des informations dynamiques telles que la pression, le temps, la vitesse d'écriture, etc... à des fins de vérification.

La plupart de ces systèmes sont conçus sur la base des algorithmes d'apprentissage machine et possèdent néanmoins beaucoup de faiblesses qui ne permettent pas de procéder à une vérification efficace. C'est dans cette optique, que nous tentons de mettre en place un système de vérification dynamique de signature, en utilisant une nouvelle approche d'étude et en exploitant de nouvelles caractéristiques dynamiques de la signature manuscrite afin de combler les faiblesses des systèmes de vérification existants et qui serait à la fin capable d'identifier efficacement si une signature est falsifiée ou non grâce à une technique d'apprentissage automatique qui est dans notre cas le réseau de neurones.

Problématique

La falsification de la signature est devenue monnaie courante de nos jours. L'authenticité d'une signature ne pouvant pas être évaluée de manière efficace à l'œil nu, il incombe donc

aux structures dans le but d'assurer la sécurité et la protection des biens de leurs clients, de faire recours à des systèmes informatiques de vérification de signature efficaces. Ces systèmes possèdent pour la plupart des faiblesses qui ne permettent pas de procéder à une vérification efficace. D'où l'obligation de palier à ses faiblesses en développant une solution beaucoup plus efficace issue d'une nouvelle approche d'étude et destinée aux services juridiques, aux banques et partout où la signature est utilisée comme moyen d'authentification .

Objectifs

L'objectif principal de ce travail est de mettre en place un système facile d'accès et capable de détecter efficacement une falsification à partir d'échantillons de signatures authentiques et falsifiées fournis par un utilisateur. Cet objectif pourra être atteint grâce à la réalisation des objectifs spécifiques que sont :

- Faire une étude bibliographique des systèmes de vérifications de signature préexistants.
- Étudier l'ensemble des contours relatifs à l'architecture des réseaux de neurones ainsi que les modèles mathématiques autour et également d'autres techniques d'apprentissage automatique.
- Collecter des données de signatures originales et falsifiées.
- Étudier les outils et langage de programmations nécessaires adéquats pour faire de l'apprentissage en profondeur.
- Identifier et étudier les caractéristiques à extraire d'un échantillon d'une signature pour pouvoir qualifier efficacement cette signature (authentique ou fausse).
- Implémenter et tester plusieurs architectures de réseaux de neurones.
- Dégager la meilleure architecture et mettre en place un système de vérification basé sur l'ensemble des caractéristiques retenues au préalable.
- Mettre en place une application web capable de pouvoir utiliser le système final.

Méthodologie d'étude

Afin d'aboutir aux résultats escomptés, dans un premier temps nous procéderons aux choix des caractéristiques à extraire d'une signature afin de justifier l'authenticité de la signature tout en s'assurant qu'elles apportent un impact positif sur les performances. Ensuite on procède à la validation de l'architecture neuronale en réalisant une étude sur plusieurs architectures afin d'en dégager l'architecture la plus optimale avec le meilleur paramétrage possible. Enfin on met en place un système de vérification interfacé par une application web afin de permettre l'utilisation de ce système.

Résultats obtenus

Au vue des objectifs, les livrables de ce travail sont :

- Caractérisation d'une signature (présentation des caractéristiques qui prises ensemble permettent de décrire qualitativement une signature) ;
- Paramétrage du réseau de neurone d'apprentissage (présentation des paramètres qui permettent d'avoir un système d'apprentissage performant pour la détection de signature originale ou fausse) ;
- Application web de vérification de signatures originales et falsifiées (présentation d'une application qui permet de vérifier si la signature d'une personne est falsifiée ou pas).

Limitation et perspective de ce travail

La conception de notre vérificateur de signature se base sur des signatures obtenues à partir d'une *Tablette numérique de signature*. Cela, nous a permis d'extraire des caractéristiques morphologiques (liées à l'image) pour apprendre la forme de la signature mais aussi des caractéristiques abstraites telles que la vitesse de signature et la pression du stylo lors de la signature, pour apprendre le comportement des gens quand ils signent. Ces dernières caractéristiques ont été utiles pour faire comprendre à notre système la différence entre signature originale et falsifiée liée à une personne donnée.

Ainsi, *ce travail se limite aux signatures manuscrites électroniques* et ne tient pas compte des signatures-images (scannées, photos). Des études futures se pencherons sur la possibilité d'extraire des comportements abstraites à partir d'une image.

Il faut aussi souligner que au cours de la phase d'apprentissage en fonction des bases de données, notre application peut nécessiter des ressources machines importantes due aux réseaux de neurones qui sont connues pour être des processus gourmands en temps et en mémoire.

Plan du document

Notre document est subdivisé principalement en trois chapitres. Un premier chapitre pour donner les généralités sur la signature, présenter les systèmes de vérification de signature existants et enfin expliquer les contours des réseaux de neurones et d'autres techniques d'apprentissage. Dans le second chapitre, nous parlons essentiellement des outils dont nous aurons besoin et nous présentons une méthodologie pour pouvoir mieux exploiter ces outils afin d'atteindre nos résultats. Enfin le troisième chapitre présente l'ensemble des résultats que nous avons pu obtenir tout au long de ce travail et une discussion générale sur ce qu'on peut retenir du travail effectué.

Chapitre 1

État de l'art

Introduction

Ce chapitre présente les caractéristiques des signatures, les généralités sur les systèmes de vérification de la signature et quelques algorithmes que nous aurons à utiliser dans nos travaux. L'objectif étant de mettre en place un système de vérification nous devons donc mener une étude sur l'existant. C'est ainsi que dans un premier temps, nous parlerons des généralités concernant la signature manuscrite en elle-même c'est à dire l'origine et les recherches qui ont été faites dans ce domaine pour l'authentification de la signature manuscrite ensuite nous parlerons de l'apprentissage automatique par une machine, ses contours tout en mettant un accent particulier sur l'apprentissage en profondeur à savoir le réseau de neurones qui sera notre technique d'apprentissage sur les signatures.

1.1 Généralités sur la signature

Une signature est une marque permettant d'identifier l'auteur d'un document, d'une œuvre ou la cause d'un phénomène : ainsi un auteur signe ses écrits. Une signature peut aussi être apposée à la fin d'un document par une personne pour signifier son approbation de l'ensemble des informations contenues dans un document dont il n'est pas forcément l'auteur. C'est ainsi le cas des contrats ou de tout document commercial signé par tout un chacun.

La signature peut également être le révélateur d'une maladie ou d'un acte (parfois criminel) commis par un individu, dans le sens où elle permet d'identifier clairement la maladie, ou l'auteur de l'acte en question. Une signature a donc pour but de permettre une identification. Le paraphe est la marque visuelle abrégée de la signature complète. On distingue plusieurs types de signatures dont les plus utilisés sont :

- **La signature manuscrite** : Dans les cultures pour lesquelles les systèmes d'écriture sont basés sur un alphabet, la signature d'une personne se présente généralement comme une

forme personnalisée de l'écriture à la main de ses propres prénom et nom (pas forcément dans cet ordre). Cette forme peut être simplifiée, calligraphiée, dessinée de diverses manières, et associée à des effets de style (traits, courbes, points) qui sont mis au point par l'individu pour personnaliser cette signature et la rendre à la fois unique et si possible difficilement reproductible par d'autres que lui. Une autre variante de la signature manuscrite est : la **signature manuscrite électronique** qui consiste à cliquer sur la souris ou à tracer une signature avec le doigt ou avec un stylet numériseur sur le document numérique ou sur une tablette numérique. Notre étude portera donc uniquement sur cette dernière.

- **La signature par cachet** : Plusieurs cultures, notamment asiatiques, ne partagent pas la notion occidentale de la signature manuscrite : dans ces cultures, l'action de signer de son nom ne diffère en rien de l'écriture normale de son nom. En effet, dans ces systèmes d'écriture, écrire ou signer son nom revient au même car les caractères utilisés sont les mêmes.En pratique, dans la culture asiatique, les gens utilisent une forme de sceau nominatif où leur nom est écrit avec une graphie de cachet à la place d'une signature manuscrite.
- **La signature aveugle** : On appelle signature aveugle une signature effectuée sur un document qui a été masqué avant d'être signé.
- **La signature en peinture** : en peinture, la signature permet aux peintres de faire figurer la date, ou autres détails dans une inscription feinte (gravé sur la pierre, papier déplié...).
- **La signature en musique** : La signature rythmique désigne le type et la durée des temps dans une partition musicale.
- **La signature numérique** : En informatique, il s'agit d'un type de signature destiné à garantir l'authenticité d'un message ou d'un document informatique. Ces signatures ne sont pas conçues pour être lisibles directement par l'homme, et se présentent comme des séquences arbitraires de caractères alphanumériques ou de bits.

1.2 La signature manuscrite : Vue d'ensemble

1.2.1 Caractéristiques de la signature d'un humain

Selon le "American Heritage Dictionnary", une signature peut être définie comme «le nom d'une personne écrite de sa propre main ; l'acte de signer son nom »[1]. Une deuxième définition concerne l'ensemble du processus de signature, ce qui implique que la manière dont la signature est faite, fait partie de la signature elle-même. Cela conduit à l'hypothèse que les caractéristiques du processus de signature (pression du stylo, vitesse, etc.) sont propres à chaque individu[2].

La première définition stipule qu'une signature est une image statique bidimensionnelle ne contenant pas des informations relatives au temps et la seconde définition est basée sur les caractéristiques dynamiques du processus de signature [2].

Les signatures peuvent prendre de nombreuses formes, par exemple, des personnes utilisent leurs propres noms comme signature ou éventuellement leurs initiales, ou bien elles utilisent des signatures qui n'ont guère de lien avec leurs noms [3] et, selon Brault et Plamondon [4], certaines signatures peuvent être assez complexes alors que d'autres sont simples et peuvent être falsifiées facilement.

Ruth Rostron [5] explique que la graphologie d'une signature manuscrite est utilisée pour analyser et révéler la personnalité d'un individu. Elle explique également que la variabilité dans la signature des personnes peut être due à plusieurs facteurs, notamment en ce qui concerne l'humeur de la personne en ce moment.

Gubta [3] souligne que si deux signatures de la même personne étaient identiques, elles pourraient automatiquement être considérées comme une falsification par traçage. Du point de vue des experts, les signatures d'une même personne seront également différentes, à la fois globalement et localement et peuvent également différer par leur échelle et leur orientation. Malgré ces variations, il est dit que ces signatures auront toujours les mêmes caractéristiques, telles que l'angle oblique et la pression.

Il a été également dit que les experts formés dans la détection de fraude sur signature sont très efficaces pour identifier les faux, mais peut-être pas pour vérifier les signatures authentiques. Par exemple, dans une étude détaillée, Herbst et Liu [6], déclarent que les experts en matière de signature ont réussi à rejeter jusqu'à 25% de signatures authentiques tout en n'acceptant aucune falsification. Le personnel non formé a accepté jusqu'à 50% de contrefaçons.

1.2.2 Utilité de la signature manuscrite

Une signature est une représentation manuscrite et souvent élégante de son nom ou une certaine marque que l'on écrit sur un document pour approuver son identité. L'auteur de cette signature s'appelle un signataire. Une signature ne doit pas être confondue avec un autographe. Un autographe est une signature artistique destinée au public alors qu'une signature est toujours gardée privée ou cachée.

La signature témoigne essentiellement de :

- La provenance du document (identité) ;
- L'intention (volonté) d'un individu à l'égard de ce document ;
- etc ...

Par exemple, le rôle d'une signature dans de nombreux contrats n'est pas uniquement de prouver l'identité de la partie contractante, mais aussi de fournir la preuve de la délibération

et du consentement, ce qui signifie que la partie contractante était effectivement présente et a accepté les conditions générales.

Dans de nombreux pays, les signatures sont faites devant un notaire public pour approuver tout document juridique. Sur les documents légaux, un signataire illettré peut faire une "marque" et un témoin alphabète signe le même document. Dans certains pays, les analphabètes apposent une empreinte digitale sur des documents juridiques au lieu d'une signature écrite.

1.2.3 Les différents types de signature manuscrite

Il existe trois principaux types de signatures : les signatures authentiques, les contrefaçons et les signatures déguisées. Alors que les signatures authentiques présentent une certaine stabilité, les contrefaçons produites par le même faussaire présentent des variations significatives en fonction des compétences du faussaire.

Une **signature authentique** se produit lorsqu'un signataire authentique produit ses propres signatures dans des conditions normales. Il n'est limité par aucune règle. Ce sont des dessins gratuits et ne peuvent transmettre aucun sens. Dans de nombreux cas, les signatures authentiques sont illisibles. Bien que les signatures d'un individu puissent sembler très similaires, il est largement admis que les signatures sont produites différemment chaque fois qu'un auteur authentique signe. En quelques mots c'est dire qu'il n'y a pas deux signatures géométriquement identiques pour un même signataire. Par conséquent, lorsque deux signatures identiques sont présentées, au moins l'une d'entre elles doit être une fausse [7]. Plusieurs facteurs affectent la signature de chaque personne, tels que le pays, l'âge, l'heure, les habitudes, l'état psychologique ou mental, les conditions physiques et pratiques [8].

Les **signatures falsifiées** sont faites par un imposteur dans le but de faire passer la signature comme la signature authentique d'un autre individu. Les différences entre les contrefaçons et les signatures authentiques proviennent des différences entre les organes coordonnateurs du cerveau responsables de la génération de la signature des auteurs authentiques et des faussaires.

Comparées aux signatures authentiques, les caractères dans une signature falsifiée sont souvent très larges. Les courbes peuvent devenir des angles et inversement. Une redondance telle que des traits ou même des caractères se produit. La qualité des lignes peut être médiocre, d'autres différences qui peuvent se produire sont : la ponctuation, la pression locale ou globale, la trajectoire, l'espacement, etc. Cependant les chercheurs pensent que la plupart de ces caractéristiques ne peuvent être modélisées et calculées pour la vérification de la signature [8].

Les **signatures déguisées** sont des signatures qui sont produites dans les situations où un signataire authentique produit des signatures pour rejeter l'authenticité des documents signés plus tard dans le futur [7]. Ces signatures sont produites par des utilisateurs authentiques et

ressemblent aux signatures authentiques, mais elles contiennent des caractéristiques que l'on trouve souvent dans les contrefaçons.

1.2.4 Mode de fonctionnement : statique, dynamique et hybride

En fonction des types d'informations disponibles, la réalisation d'un système de vérification de signature peut varier. Il est supposé qu'un système de vérification de signature serait plus efficace et donnerait plus de précision pour un système dynamique.

Si les informations d'entrée sont représentées sous la forme d'une fonction temporelle (en fonction du temps), le système est considéré comme un système de **vérification dynamique**. Ce flux d'informations est capturé à la volée, par exemple lorsqu'une personne écrit à l'aide d'un stylet, d'une tablette, d'un stylet numériseur ou d'un écran tactile. Les données obtenues peuvent être la pression locale, l'accélération, la vitesse, l'altitude du stylet, le nombre de coups du stylet et l'ordre des coups du stylets etc...

Ainsi ces différents types d'informations mises à notre disposition permettent d'avoir des performances plus élevées pour le mode de fonctionnement dynamique comparées aux autres types de fonctionnement. Ces informations peuvent également aider à générer des images de la signature statique de ce fait juste après l'acquisition de ces informations après une signature effectuée, elles ne nécessitent pas d'énormes traitements et peuvent directement être passées en entrée d'un système pour effectuer la vérification.



FIGURE 1.1 – Image d'une tablette wacom et d'un stylet numériseur [39]

Lorsque le processus de vérification est effectué en utilisant uniquement l'image d'une signature statique, le processus de vérification est appelé **vérification statique**. Cela ne nécessite aucun matériel spécialisé. L'inconvénient est que la quantité d'informations obtenues est beaucoup plus difficile à interpréter et nécessite très souvent l'utilisation de nombreux algorithmes de traitements d'image pour extraire les informations. En outre, les informations dynamiques sont difficiles à récupérer à partir d'une image statique. La récupération nécessite des compétences et des techniques professionnelles. Ces inconvénients empêchent les systèmes statiques

de donner de meilleurs résultats. Les experts en analyse de documents suggèrent donc que la détection de falsifications doit se faire sur des informations statiques et dynamiques.

En **mode hybride**, la vérification de l'image de la signature est effectuée à partir des données dynamiques précédemment citées ainsi que de données statiques telles que la taille de la signature, le centre de gravité de l'image de la signature, l'aire occupée par la signature etc. Cette approche comprend souvent la récupération de la trajectoire à partir de l'image numérisée avant de la comparer aux propriétés de la trajectoire récupérée.

1.3 Techniques de vérification de signature

1.3.1 Généralité

Les systèmes de vérification dans la plupart des cas, reçoivent en entrée une signature (originale ou falsifiée) et nous donnent en sortie le statut de l'image c'est à dire authentique ou falsifiée. Pour ce fait, ces systèmes utilisent des algorithmes d'apprentissage tels que : les réseaux de neurones, la machine à vecteur de support, les algorithmes génétiques etc. La figure 1.2 en est un schéma synoptique qui illustre cette procédure.

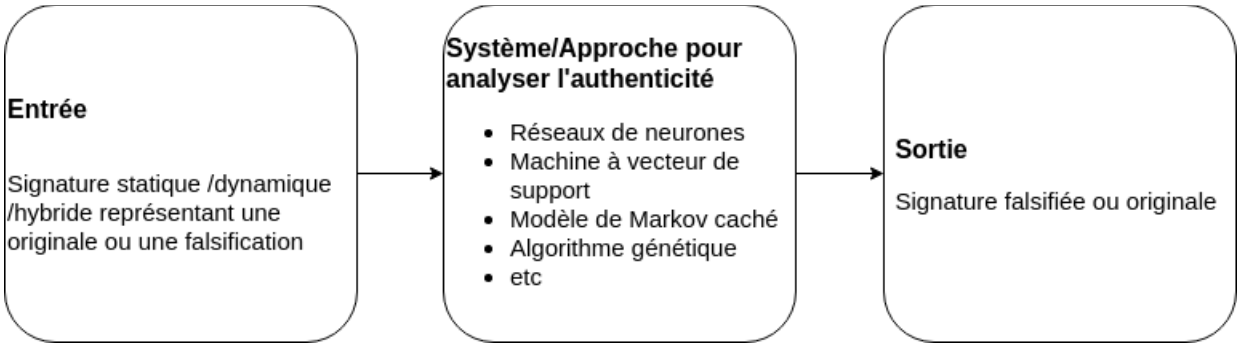


FIGURE 1.2 – Schéma fonctionnel d'un système de vérification de la signature

Ces systèmes sont divisés en deux catégories à savoir le système de vérification statique et le système de vérification dynamique. **L'approche statique** implique moins de contrôle électronique et utilise des images de signatures capturées par un scanner ou un appareil photo. Un système de vérification de signature statique utilise des caractéristiques extraites de l'image de signature numérisée pour la vérification telles que les niveaux de pixels, le centre de masse, le ratio de l'image ,etc. **L'approche dynamique** utilise une tablette électronique et un stylet, l'ensemble connecté à un ordinateur pour extraire des informations sur une signature et prend des informations dynamiques telles que la pression, le temps, la vitesse d'écriture, etc... à des fins de vérification.

1.3.2 Système de vérification statique

La première solution pour résoudre le problème de vérification de signature était la vérification de signature statique. Il distingue les signatures authentiques des signatures forgées en utilisant des images statiques. Ils ont uniquement pour entrée une image statique contenant la signature de laquelle on peut extraire uniquement des caractéristiques se basant sur la forme de la signature, sans aucune connaissance du processus de signature. Certaines difficultés rencontrées par les systèmes statiques peuvent être du bruit sur l'image, la variété des pointes de stylo peuvent également produire différentes formes [9].

La vérification de signature statique implique moins de contrôle électronique et utilise des images de signature capturées par un scanner ou un appareil photo. Un système de vérification de signature statique utilise des caractéristiques extraites de l'image de la signature numérisée.

Les caractéristiques extraites pour la vérification de signature statique peuvent être divisées en trois types principaux :

- Les **caractéristiques globales** : elles peuvent être facilement extraites. Les caractéristiques globales comprennent généralement : la zone de signature, le rapport hauteur / largeur de la signature, le centre de gravité, etc.
- Les **caractéristiques locales** : elles sont extraites dans de petites parties de l'image de la signature. Ces caractéristiques sont beaucoup plus précises que les caractéristiques globales. Les caractéristiques locales incluent la densité de pixels locaux, les caractéristiques des zones inclinées, les points critiques, etc.

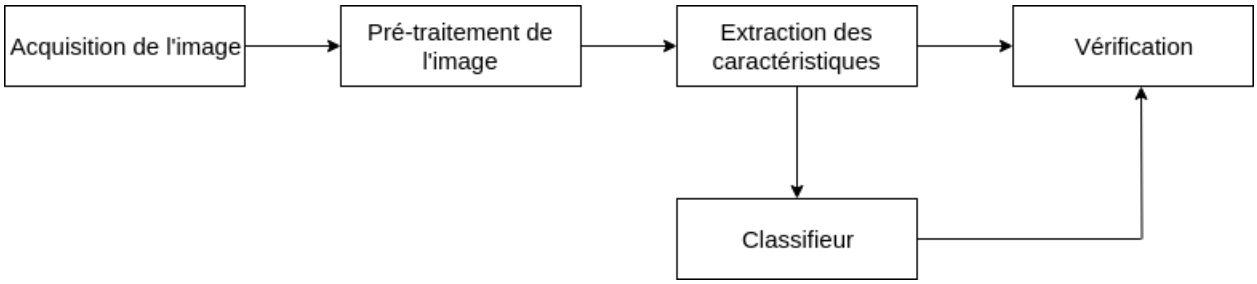


FIGURE 1.3 – Vue d'ensemble du système de vérification de signature statique

Acquisition de l'image : les signatures à traiter par le système doivent être en format d'image numérique. Les données du système de vérification de signature statique sont acquises de différentes manières, par exemple par capture photographique, scanner, etc. Les échantillons de signature sont numérisés, puis les images numérisées sont stockées numériquement pour un traitement ultérieur.

Pré-traitement : le pré-traitement des signatures a pour objectif de rendre les signatures standards et prêtes pour l'extraction de caractéristiques. Le pré-traitement est une étape nécessaire pour améliorer la précision de l'extraction et de la vérification des caractéristiques. Avant le traitement de l'image pour l'extraction des caractéristiques, certains algorithmes de pré-traitements (traitement d'image) sont appliqués à l'image numérisée, tels que la binarisation, la réduction du bruit, l'amincissement, etc.

Extraction de caractéristiques : l'efficacité d'un système de vérification de signature dépend principalement de l'étape d'extraction des caractéristiques. Les techniques d'extraction de caractéristiques doivent être rapides et faciles à calculer pour que le système soit optimal et ait une faible puissance de calcul. Les caractéristiques sélectionnées doivent faire la distinction entre une signature authentique et une signature falsifiée. Les caractéristiques extraites pour la vérification de signature statique peuvent être divisées en caractéristiques globales, locales et géométriques.

vérification : L'étape de vérification compare les caractéristiques des signatures de test avec les caractéristiques des signatures authentiques basées sur diverses techniques de classification de motifs et prend la décision finale de vérification en tant que signature authentique ou falsifiée.

Dans le domaine de la vérification de la signature manuscrite, en particulier la méthode statique, différentes technologies ont été utilisées et le domaine est toujours en pleine étude. Nous passons donc en revue certains articles récents sur les systèmes de vérifications statiques. Les approches utilisées par les différents chercheurs diffèrent par le type de caractéristiques extraites, la méthode d'entraînement et le modèle de classification et de vérification utilisé.

1.3.2.1 Modèle de Markov caché

Le modèle de Markov caché est l'un des modèles les plus utilisés pour l'analyse de séquence dans la vérification de la signature.La signature manuscrite étant une séquence de vecteurs de valeurs liés à chaque point de la signature dans sa trajectoire, il est donc impératif de choisir un ensemble bien défini de vecteurs caractéristiques pour la conception d'un système basé sur le modèle de Markov caché et ainsi avoir un système efficace de vérification de la signature.Ce type de modèle est stochastique, donc capable d'absorber la variabilité entre les modèles et les similitudes. La correspondance stochastique est largement impliquée dans ce modèle. Cet appariement est effectué par étapes de distribution de probabilité des caractéristiques impliquées dans les signatures ou par la probabilité de calcul de la signature d'origine. Si les résultats montrent une probabilité plus élevée que la probabilité des signatures de test, les signatures sont alors celles de la personne d'origine, sinon les signatures sont rejetées. Dans le papier [10], un système est introduit et utilise uniquement des caractéristiques globales. Une transformée

aléatoire discrète appelée "sinographe", est calculée pour chaque image de la signature binaire dans une plage allant de 0 à 360 et représente une fonction du pixel total de l'image et de l'intensité par pixel donné.Cette méthode a atteint un AER de 18,4% pour un ensemble de 440 signatures authentiques provenant de 32 auteurs avec 132 contrefaçons.

1.3.2.2 Les réseaux de neurones

Les principales raisons de l'utilisation répandue des réseaux de neurones dans la reconnaissance de formes sont leur puissance et leur facilité d'utilisation. Une approche simple consiste tout d'abord à extraire un ensemble de caractéristiques représentant la signature (détails tels que la longueur, la hauteur, le centre de gravité, etc.), avec plusieurs échantillons provenant de signataires différents. La deuxième étape consiste pour le réseau de neurone à apprendre la relation entre une signature et sa classe («authentique» ou «falsifiée»). Une fois que cette relation a été apprise, le réseau peut se voir présenter des signatures de test pouvant être classées comme appartenant à un signataire particulier ou comme étant une signature originale ou falsifiée. Les réseaux de neurones conviennent donc parfaitement à la modélisation des aspects globaux des signatures manuscrites.Le système proposé dans [11] utilise les caractéristiques de structure de contours des signatures, de modification de la direction, ainsi que des caractéristiques supplémentaires telles que la surface, le décalage de longueur et le centroïde, dans lesquelles une signature est divisée en deux moitiés et pour chaque moitié une position du centre de gravité est calculée par rapport à l'axe horizontal. Pour la classification et la vérification, deux approches sont comparées : le réseau de neurones à propagation résiliente et à fonction de base radiale en utilisant une base de données de 2106 signatures contenant 936 signatures authentiques et 1170 signatures falsifiées. Ces deux classificateurs enregistrent respectivement 91,21% et 88% comme taux de précision.

Un autre article, nous présente un modèle dans lequel un classifieur de réseau neuronal est utilisé pour la vérification. Les signatures de la base de données sont pré-traitées avant l'extraction des caractéristiques. Les caractéristiques sont extraites de l'image de signature prétraitée. Ces caractéristiques extraites sont ensuite utilisées pour former un réseau de neurones [12]. Les résultats obtenus pour cette étude sont de 82,66% de signatures correctement classées sur un ensemble de 300 signatures.

1.3.2.3 Approche par correspondance de modèle

Dans la correspondance de modèle, deux techniques sont proposées pour la détection des falsifications [13]. La correspondance optimale des profils de projection des motifs de signature à une dimension est une méthode, tandis que l'autre méthode consiste à utiliser des motifs de signature à deux dimensions qui dépendent de la correspondance élastique des traits. Si une signature de test doit être vérifiée, les variations de position sont comparées aux statistiques de l'ensemble des données d'entraînement et une décision est prise sur la base d'une mesure de

distance. Les images de signature binaires et de niveaux de gris sont testées. Le taux d'erreur de vérification moyen de 18,1% a été atteint lorsque les pics locaux des projections de profil vertical des images de signature en niveaux de gris ont été utilisés pour l'appariement.

1.3.2.4 Approche statistique

En utilisant les connaissances statistiques, il est facile de trouver la relation, la déviation, etc, entre deux ou plusieurs éléments de données. Pour déterminer la relation entre un ensemble d'éléments de données, nous suivons généralement le concept de coefficients de corrélation.Pour procéder à la vérification, on compare une signature en entrée à une signature moyenne obtenue grâce aux données d'entraînement, cette approche suit le concept de corrélation afin de déterminer le degré de divergence entre elles.Une méthode a été introduite dans [14]. Dans cette approche, diverses caractéristiques sont extraites, notamment des caractéristiques globales telles que le dégradé d'image, des caractéristiques statistiques dérivées de la distribution de pixels d'une signature et des descripteurs géométriques. La classification consiste à obtenir des variations entre les signatures du même auteur et à obtenir une distribution. Pour toute signature mise en cause, la méthode obtient une distribution qui est comparée à celle connue et une probabilité de similarité est obtenue en utilisant un test statistique de Kolmorogorv-Smirnov. En utilisant seulement 4 échantillons authentiques pour l'apprentissage, la méthode permet d'atteindre une précision de 84% qui peut être portée à 89% lorsque la taille de l'échantillon des données d'entraînement est augmentée. Cette méthode n'utilise aucune signature falsifiée lors de l'apprentissage.

1.3.2.5 Machine à vecteur de support

Les machines à vecteurs de support sont des algorithmes d'apprentissage automatique qui utilisent un très grand espace dimensionnel de fonctions et évaluent les différences entre les classes de données afin de faire une généralisation. Le système de [15] utilise les caractéristiques globales, directionnelles et de grille de la signature avec un algorithme de machine à vecteur de support pour la classification et la vérification. Une base de données de 1320 signatures a été utilisée et nous a permis d'obtenir un FRR de 2% et une FAR de 11%.

1.3.3 Système de vérification dynamique

La vérification de signature dynamique est basée sur les caractéristiques dynamiques du processus de la signature. Le fait que la vérification dynamique contienne plus d'informations, le taux de précision du processus d'authentification de la signature, est nettement supérieur à celui d'une méthode statique qui ne contient aucune information sur le processus lors de signature [16].

Un matériel spécial permettant de mesurer les caractéristiques dynamiques du processus de la signature est donc nécessaire pour les méthodes de vérification dynamique telles qu'une tablette à numériser, qui enregistre principalement la trajectoire et la vitesse du processus ainsi que la pression, la position de la pointe du stylet et d'autres caractéristiques. La combinaison de ces caractéristiques est ainsi propre à chaque individu [16].

La méthodologie comprend l'acquisition de données, le prétraitement, l'extraction de caractéristiques, la prise de décision et l'évaluation de la performance [3]. Les méthodes statiques semblent plus pratiques que les méthodes dynamiques, mais plus difficiles, car seules les caractéristiques statiques sont disponibles à partir de l'image de signature [17].

Les méthodes d'authentification de signature dynamiques utilisent des tablettes de capture de pression et un stylet pour capturer chaque point de la signature. Ainsi, à chaque point, en utilisant une tablette, les coordonnées x et y du point, la pression appliquée lors de la signature, le temps, ainsi que les angles d'azimut et l'altitude sont enregistrés. Ces caractéristiques dynamiques sont ensuite utilisées pour la vérification des signatures.

L'une des techniques les plus courantes pour la vérification des signatures implique l'utilisation de techniques de traitement d'image. Liu Dong et.al [18] ont présenté une nouvelle méthode basée sur une approche par appariement de modèles et sur une description de données vectorielles pour résoudre la difficulté de choisir le seuil de vérification dans la vérification de la signature. Chang P. et al. [19] ont proposé une approche basée sur plusieurs algorithmes de correspondance de modèles pour identifier l'individu au moyen de quelques échantillons d'apprentissage.

La déformation temporelle dynamique (DTW) est un algorithme permettant de mesurer la similarité entre deux séquences pouvant varier dans le temps ou la vitesse. Basé sur la programmation dynamique, l'algorithme de déformation temporelle dynamique trouve une correspondance optimale entre deux séquences de vecteurs de caractéristiques en permettant l'étirement et la compression de sections des séquences. Daramola et Ibiyemi [20] ont développé un système de vérification de signature dynamique sur la déformation temporelle dynamique.

La machine à vecteur de support est un classifieur dérivé de la théorie de l'apprentissage statistique présenté pour la première fois dans [21] . Le problème que ce classifieur essaie de résoudre est de trouver un hyperplan optimal pour classifier correctement les points de données en séparant autant que possible les points de deux ou plusieurs classes. Kour J. et al. [22] ont mis au point un système de vérification dynamique de signature utilisant des algorithmes génétiques et la méthode de la machine à vecteur de support. Fauziyah S. et al. [23] ont également mis au point un système de vérification de signature dynamique utilisant la méthode de la machine à vecteur de support et VBTablet 2.0.

L'approche de Justino et al [24] utilise les caractéristiques graphométriques, c'est-à-dire des caractéristiques statiques telles que la densité de pixels et les caractéristiques pseudo-dynamiques représentées par l'inclinaison axiale. Un modèle de Markov caché est utilisé pour le processus d'apprentissage et de vérification.

L'article [25] présente un système de prévention de contrefaçon de signature en dynamique mis en place à l'aide de réseaux de neurones à partir d'un algorithme de rétro-propagation de l'erreur. La vérification des signatures est effectuée à l'aide de leurs caractéristiques statiques et dynamiques.

1.3.4 Application de la vérification de signature

Les techniques de vérifications de signature peuvent être appliquées dans de nombreux domaines, tels que :

- **Les transactions financières** : la signature est la méthode d'authentification la plus préférée en raison de sa commodité. Mais des événements récents ont montré qu'il est devenu facile de falsifier une signature manuscrite ce qui a finalement augmenté les pertes financières. Par exemple, pour les fraudes par chèques et cartes de crédit, selon A.Kholmatov [26], MasterCard estime chaque année une perte de 450 millions de dollars liée à la fraude aux cartes de crédit.

- **Les transactions bancaires en ligne** : afin d'acquérir la signature manuscrite de l'utilisateur, une tablette à numériser est utilisée. Cela évite la peine aux utilisateurs de se souvenir du mot de passe et des codes PIN. Ainsi, la capture de caractéristiques statiques et dynamiques confère à la signature manuscrite un caractère unique pour chaque individu qui est également très difficile à imiter.

- **Le traitement de chèque** : le traitement de chèque comprend également un système de vérification de signature dans lequel le chèque après avoir été scanné compare la signature présente sur le chèque à l'ensemble des signatures authentiques présentes dans la base de données des utilisateurs.La majeure partie du processus est automatisée et numérisée ainsi une intervention humaine ne sera requise que si la signature en cours de traitement aboutit à un seuil supérieur à un seuil spécifique.

- **L'authentification d'utilisateur sur un ordinateur** : les systèmes de vérification de signature peuvent également être utilisés pour accéder à des systèmes informatiques tels que un système d'exploitation et un système d'information, en remplacement du système de mot de passe traditionnel. L'exigence de base serait de connecter une tablette à numériser à chaque poste de travail pour capturer les informations de la signature [3].

- **Les passeports** : Les systèmes de vérification peuvent également être utilisés dans le processus de validation du passeport. Au moment de la délivrance du passeport à une personne, celle-ci sera tenue de se rendre dans un bureau agréé où elle devra fournir un échantillon de signature. Cette signature sera effectuée électroniquement sur la bande magnétique du passeport. Ainsi au point d'entrée d'un autre pays, le client devra signer sur une tablette graphique, qui sera comparée à la signature de référence stockée sur la bande magnétique [3].

1.4 Les réseaux de neurones

1.4.1 Machine Learning : Généralités

L'apprentissage automatique enseigne aux ordinateurs à apprendre comme ça se fait de manière naturelle aux humains et aux animaux : apprendre de l'expérience. Les algorithmes d'apprentissage automatique utilisent des méthodes de calcul pour «apprendre» des informations directement à partir de données sans utiliser une équation prédéterminée comme modèle. Les algorithmes améliorent de manière adaptative leurs performances lorsque le nombre d'échantillons disponibles pour l'apprentissage augmente.

Les algorithmes d'apprentissage automatique recherchent des modèles naturels dans les données qui génèrent des informations et aident à prendre de meilleures décisions. Ils sont utilisés quotidiennement pour prendre des décisions critiques en matière de diagnostic médical, d'échange d'actions, de prévision de la charge énergétique, etc. Les sites de médias font appel à l'apprentissage automatique pour passer en revue des millions d'options et vous proposer des recommandations de chansons ou de films. Les détaillants l'utilisent pour mieux comprendre le comportement d'achat de leurs clients.Avec l'essor du Big Data, l'apprentissage automatique est devenu particulièrement important pour résoudre des problèmes dans des domaines tels que :

- L'informatique dans les finances, pour l'attribution de crédit et le trading algorithmique ;
- Le traitement d'images et la vision par ordinateur, pour la reconnaissance faciale, la détection de mouvement et la détection d'objets ;
- La biologie numérique, pour la détection des tumeurs, la découverte de médicaments et le séquençage de l'ADN ;
- La production d'énergie, pour la prévision des prix et de la charge ;
- L'automobile, l'aérospatiale et la fabrication, pour la maintenance prédictive ;
- Le traitement du langage naturel.

L'apprentissage automatique utilise deux types de techniques : l'apprentissage supervisé, qui forme un modèle sur des données d'entrée et de sortie connues afin qu'il puisse prédire les sorties futures, et l'apprentissage non supervisé, qui trouve des modèles cachés ou des structures intrinsèques dans les données d'entrée.

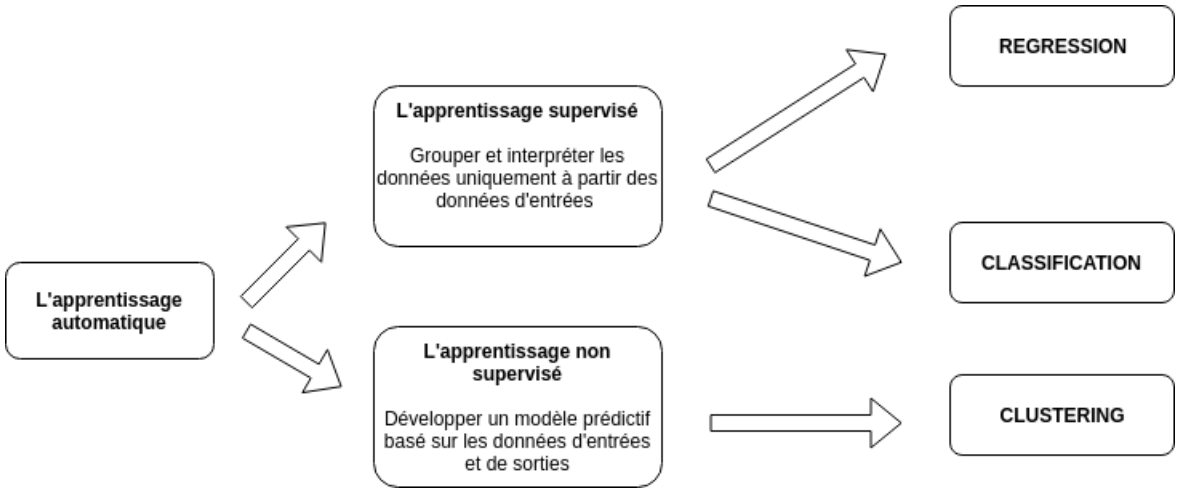


FIGURE 1.4 – Schéma synoptique des techniques d'apprentissage automatique

L'apprentissage supervisé a pour objectif de créer un modèle permettant de faire des prédictions à partir de données probantes en présence d'incertitude. Un algorithme d'apprentissage supervisé prend un ensemble connu de données d'entrée et de réponses connues aux données (sortie) et entraîne un modèle afin de générer des prédictions raisonnables pour la réponse à de nouvelles données[27].

L'apprentissage supervisé utilise des techniques de classification et de régression pour développer des modèles prédictifs.

- **Les techniques de classification** permettent de prévoir des réponses discrètes, par exemple, si un courrier électronique est authentique ou un spam, ou si une tumeur est cancéreuse ou non. Les modèles de classification classent les données d'entrée en catégories. Des applications typiques incluent l'imagerie médicale, la reconnaissance vocale et la notation de crédit.

- **Les techniques de régression** prédisent des réponses continues, par exemple des changements de température ou des fluctuations de la demande de puissance. Des applications typiques incluent la prévision de la charge d'électricité et le trading algorithmique.

L'apprentissage non supervisé trouve des modèles cachés ou des structures intrinsèques dans les données. Il est utilisé pour tirer des inférences à partir d'ensembles de données non étiquetées. La technique la plus utilisée pour ce type d'apprentissage est le clustering

- **Le clustering** est la technique d'apprentissage non supervisée la plus courante. Il est utilisé pour l'analyse exploratoire des données afin de trouver des modèles ou des regroupements cachés dans les données.Les applications de clustering incluent l'analyse de la séquence des gènes, les études de marché et la reconnaissance d'objets.

Choisir le bon algorithme peut sembler fastidieux : il existe des dizaines d'algorithmes d'apprentissage automatique supervisés et non supervisés, et chacun adopte une approche différente en matière d'apprentissage.

Il n'y a pas de meilleure méthode ni de solution unique. Trouver le bon algorithme repose sur un grand nombre d'essais et d'erreurs. Même les scientifiques expérimentés en données ne peuvent pas savoir si un algorithme fonctionnera sans l'essayer. Mais la sélection des algorithmes dépend également de la taille et du type de données avec lesquelles vous travaillez, des informations que vous souhaitez obtenir des données et de la manière dont ces informations seront utilisées.

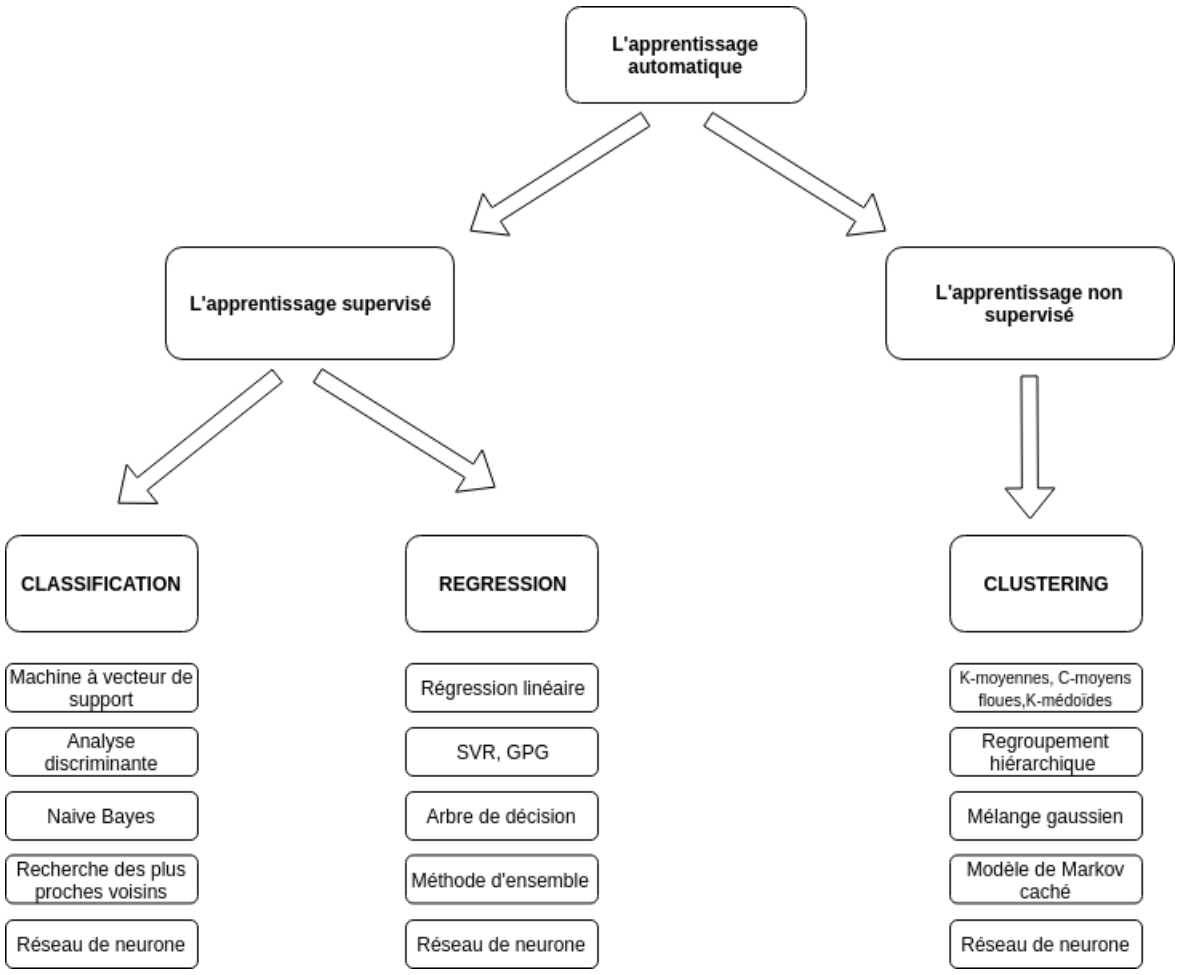


FIGURE 1.5 – Schéma d'ensemble de quelques algorithmes d'apprentissage

Cependant pour nos travaux de recherche nous avons choisir d'utiliser de l'apprentissage supervisé plus précisément la méthode des réseaux de neurones contenue dans la classe des

algorithmes de classification dont nous parlerons de long en large dans les prochaines sections mais nous définirons quand même certains algorithmes de classification que nous utiliserons par la suite.

Machine à vecteur de support

Un hyperplan est une ligne qui divise l'espace variable d'entrée. Au niveau des machines à vecteur de support, un hyperplan est sélectionné pour séparer au mieux les points de la variable d'entrée par leur classe, soit la classe 0, soit la classe 1.Ainsi en deux dimensions, vous pouvez visualiser cela sous forme de ligne et en supposant que tous nos points d'entrée peuvent être complètement séparé par cette ligne. Cet algorithme d'apprentissage recherche les coefficients qui permettent la meilleure séparation des classes par l'hyperplan.

La distance entre l'hyperplan et les points de données les plus proches est appelée la marge. L'hyperplan optimal pouvant séparer les deux classes est la ligne qui présente la marge la plus grande. Seuls ces points sont pertinents dans la définition de l'hyperplan et dans la construction du classifieur. Ces points s'appellent les vecteurs de support. Ils supportent ou définissent l'hyperplan. En pratique, un algorithme d'optimisation est utilisé pour trouver les valeurs des coefficients qui maximisent la marge.

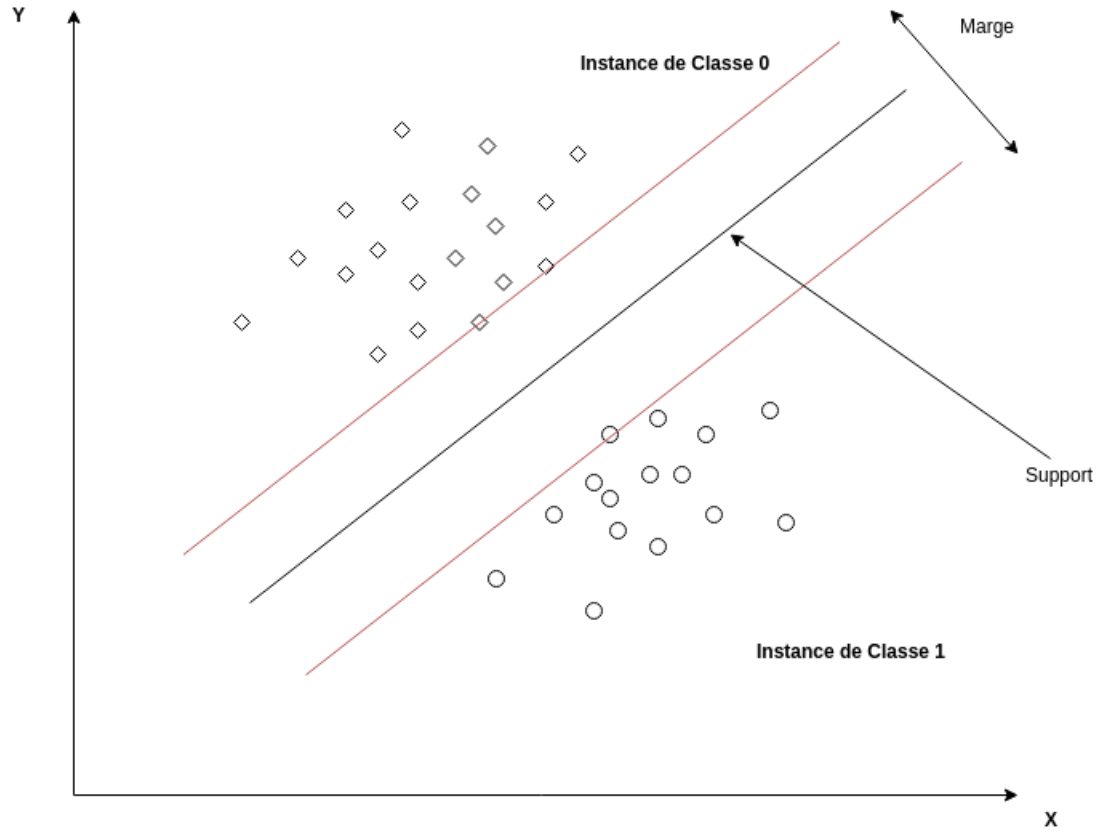


FIGURE 1.6 – Exemple de deux classes linéairement séparables par la méthode de la machine à vecteur de support

Les K voisins les plus proches

Les prédictions sont établies pour une nouvelle instance des données en effectuant une recherche dans l'ensemble complet d'apprentissage pour les K instances les plus similaires (les voisins) et en résumant la variable de sortie pour ces K instances. Pour les problèmes de régression, il peut s'agir de la variable de sortie moyenne. Pour les problèmes de classification, il peut s'agir de la valeur de la classe modale (ou la plus courante).L'astuce consiste à déterminer la similarité entre les instances des données. La technique la plus simple si vos attributs ont tous la même échelle (tous en pouces, par exemple) consiste à utiliser la distance euclidienne, un nombre que vous pouvez calculer directement en fonction des différences entre chaque variable en entrée.

Cette méthode peut nécessiter beaucoup de mémoire ou d'espace pour stocker toutes les données, mais effectue uniquement un calcul (ou un apprentissage) lorsqu'une prévision est nécessaire. Vous pouvez également mettre à jour et gérer vos instances de formation au fil du temps pour que les prévisions demeurent le plus souvent exactes.L'idée de distance ou de proximité peut être inefficace pour de très grandes dimensions (beaucoup de variables d'entrée), ce qui peut affecter négativement les performances de l'algorithme sur le problème. C'est ce qu'on appelle la malédiction de la dimensionnalité. Il suggère d'utiliser uniquement les variables d'entrée les plus pertinentes pour prédire la variable de sortie.

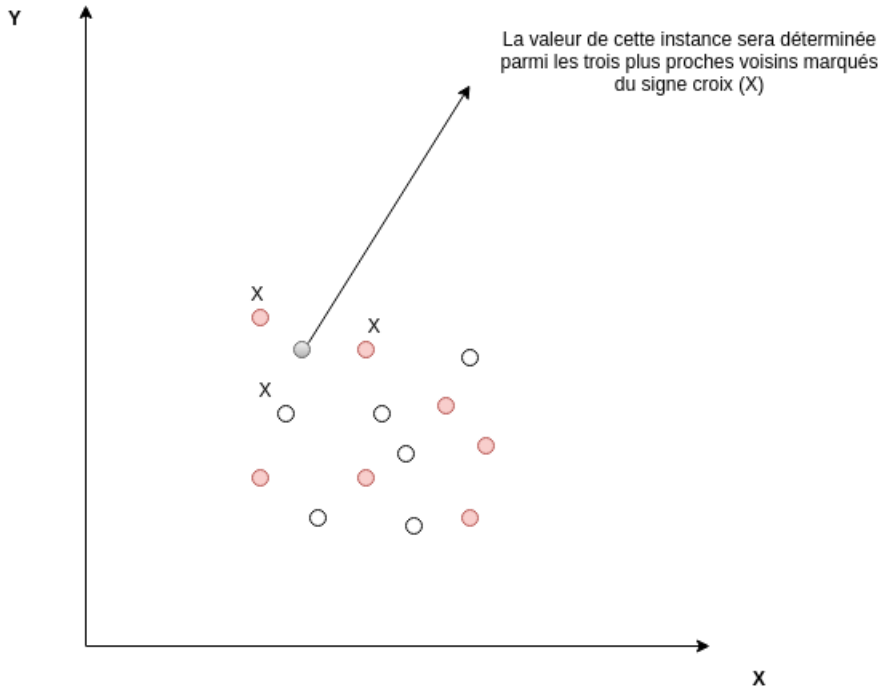


FIGURE 1.7 – Exemple des trois(3) voisins les plus proches

Finissons cette section en parlant de l'évaluation d'un système de classification. Prenons l'exemple d'un classifieur binaire, c'est-à-dire, qui prédit deux classes notées classe 0 et classe 1. Pour mesurer les performances de ce classifieur, il est d'usage de distinguer 4 types d'éléments

classés pour la classe voulue :

- Vrai positif VP : Elément de la classe 1 correctement prédit
- Vrai négatif VN : Elément de la classe 0 correctement prédit
- Faux positif FP : Elément de la classe 0 mal prédit
- Faux négatif FN : Elément de la classe 1 mal prédit

Ces informations peuvent être rassemblées et visualisées sous forme de tableau dans une matrice de confusion. Dans le cas d'un classifieur binaire, on obtient :

		Classe prédite	
		Classe 0	Classe 1
Classe réelle	Classe 0	VN	FN
	Classe 1	FP	VP

FIGURE 1.8 – Matrice de confusion à deux classes

En particulier, si la matrice de confusion est diagonale, le classifieur est parfait. Notons que la matrice de confusion est aussi généralisable lorsqu'il y a k > 2 classes à prédire. Il est possible de calculer plusieurs indicateurs résumant la matrice de confusion. Par exemple si nous souhaitons rendre compte de la qualité de la prédiction sur la classe 1, on définit :

- **la précision** : le nombre de vrais positifs sur le nombre d'exemples prédits comme vrais par le modèle. Comme son nom l'indique, elle permet de mesurer la précision du modèle, soit la confiance que l'on peut attribuer au modèle. Plus la précision est forte, plus une prédiction 'vrai' du modèle peut être considérée comme exacte.

$$precision = \frac{VP}{VP + FP} [28] \tag{1.1}$$

- **le rappel** : est le nombre de vrais positifs sur le nombre d'exemples effectivement positifs dans l'ensemble de données. Le rappel permet de mettre en évidence l'exhaustivité du modèle.

$$rappel = \frac{VP}{VP + FN} [28] \tag{1.2}$$

- **la F-mesure** : est une combinaison de la précision et du rappel. Elle mesure l'exactitude du modèle.

$$F - mesure = 2 \times \frac{precision \times rappel}{precision + rappel}[28] \tag{1.3}$$

L'efficacité des classifieurs est représentée quantitativement en termes de taux d'acceptation erronée (FAR) et de taux de rejet erroné (FRR). Le taux d'acceptation erronée représente le pourcentage de classe 0 prédit comme étant 1 par le système et le taux de rejet erroné représente le pourcentage de classe 1 prédit comme étant 0 par le système. En général plus ces valeurs sont proches de 1 et plus notre système est fiable.

$$FAR = \frac{FP}{VP + VN + FP + FN} \times 100[28] \tag{1.4}$$

$$FRR = \frac{FN}{VP + VN + FP + FN} \times 100[28] \tag{1.5}$$

1.4.2 Neurone

Un neurone artificiel est un modèle informatique inspiré des neurones naturels. Les neurones naturels reçoivent des signaux via des synapses situées sur les dendrites ou la membrane du neurone. Lorsque les signaux reçus sont suffisamment puissants (dépassent un certain seuil), le neurone est activé et émet un signal via l'axone. Ce signal peut être envoyé à une autre synapse et activer d'autres neurones [29].

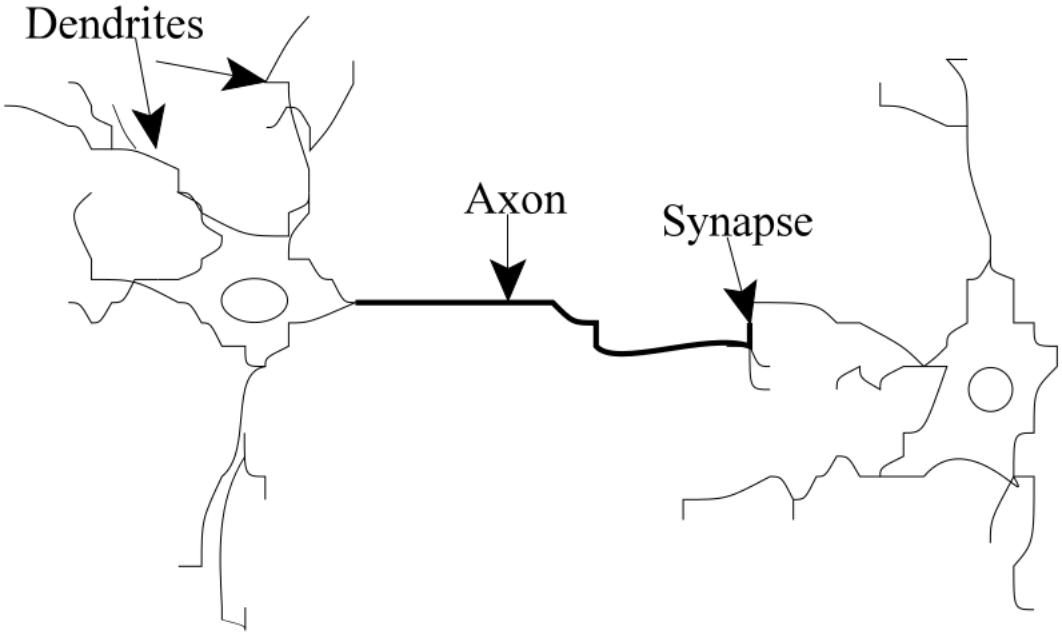


FIGURE 1.9 – Schéma d'un neurone naturel [29]

La modélisation simplifiée de McCulloch et Pitts datant de 1943 le conçoit comme recevant, d'autres neurones, des impulsions électriques de poids variables par ses dendrites. En passant dans le neurone, et si la somme des impulsions dépasse un certain seuil, l'influx électrique reçu est transformé, dans le corps du neurone, en une nouvelle impulsion électrique, que l'on peut modéliser comme un 1 ou un 0 (activation ou non activation). Cette activation est ensuite transmise par l'axone aux neurones connectés à celui-ci [30]. On construit, en se calquant sur cette modélisation, un neurone formel avec :

- Les **entrées** x du neurone, se présentant comme un vecteur de dimension N
- Un ensemble de **poids** W , modélisant les poids variables des différentes entrées. W est un vecteur de dimension N également. À chaque entrée correspond un poids.
- Un **biais** b représentant le seuil d'activation du neurone.
- La somme a des N "impulsions électriques" reçues est donc obtenue par :

$$a = \sum_{n=1}^{N} W_n x_n + b \tag{1.6}$$

- Une fonction d'activation h, rendant compte de l'influx électrique de sortie, appliquée à la somme des entrées pondérées par W et du biais b. On utilise en général la fonction sigmoïde.

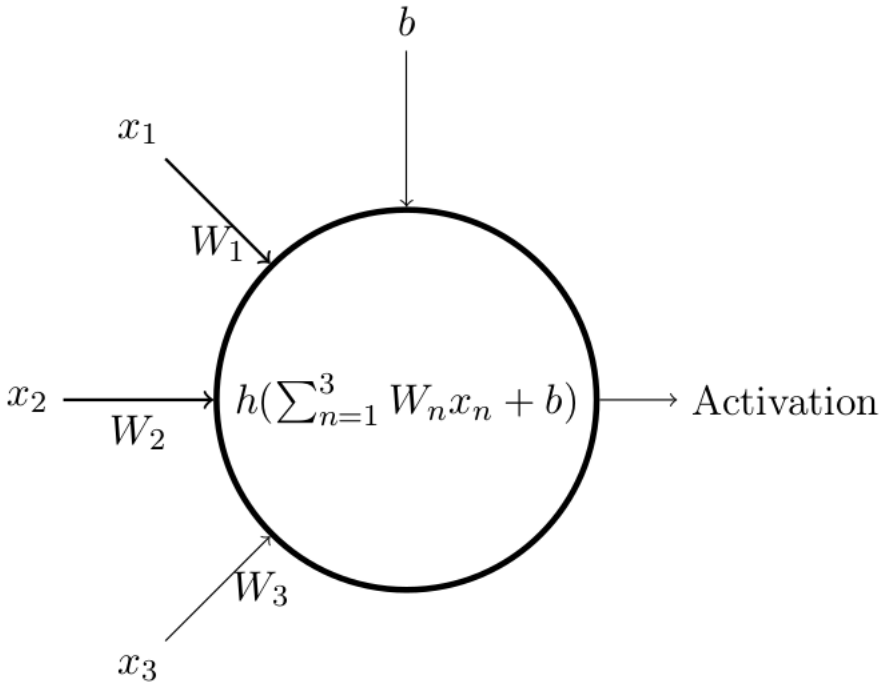


FIGURE 1.10 – Schéma d'un neurone artificiel à 3 entrées [30].

Cependant nous distinguons plusieurs fonctions d'activation telles que :

La fonction sigmoïde : elle prend ses valeurs entre]0,1[et est définie par :

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.7}$$

et sa dérivée par :

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \tag{1.8}$$

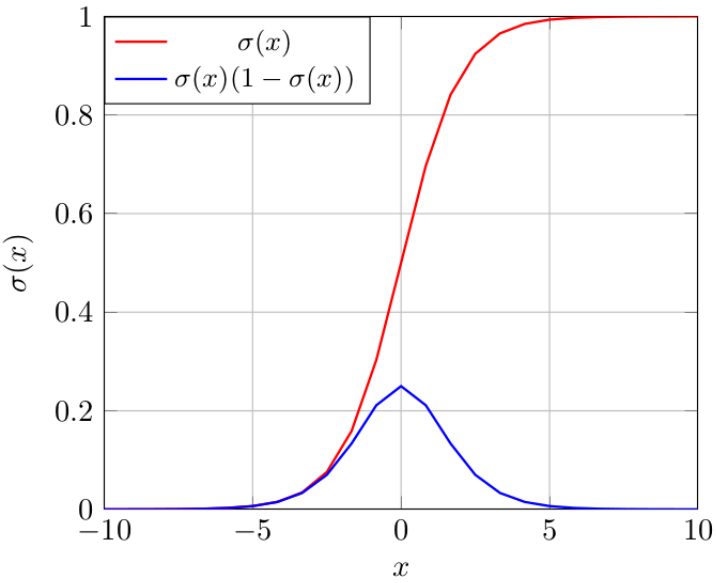


FIGURE 1.11 – Représentation de la fonction sigmoïde et sa dérivée

La fonction tangente hyperbolique : elle prend ses valeurs entre]-1,1[et est définie par :

$$g(x) = \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{2x}} \tag{1.9}$$

et sa dérivée par :

$$\tanh'(x) = 1 - \tanh^2(x) \tag{1.10}$$

Cette fonction d'activation a vu sa popularité baisser en raison de l'utilisation de la fonction d'activation présentée dans le paragraphe suivant.

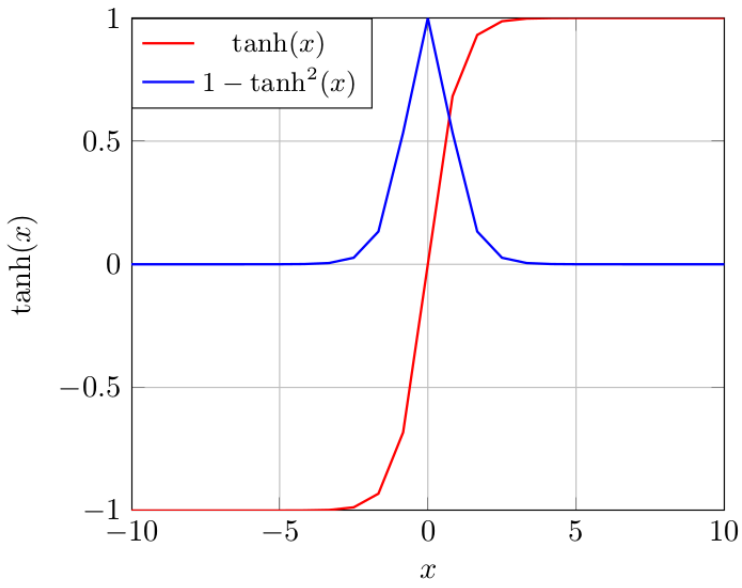


FIGURE 1.12 – Représentation de la fonction tangente hyperbolique et sa dérivée

La fonction ReLU ou unité linéaire rectifiée prend sa valeur entre]0,+∞[et est définie par :

$$g(x) = ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{1.11}$$

et sa dérivée par

$$ReLU'(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{1.12}$$

Cette fonction d'activation est la plus utilisée de nos jours. On peut également trouver deux de ses variantes les plus courantes : les systèmes à fuites ReLU et ELU (Unité linéaire exponentielle). Ils ont été introduits parce que la fonction d'activation ReLU a tendance à "tuer" certains neurones cachés : une fois désactivée (valeur nulle), elle ne peut plus jamais être réactivée.

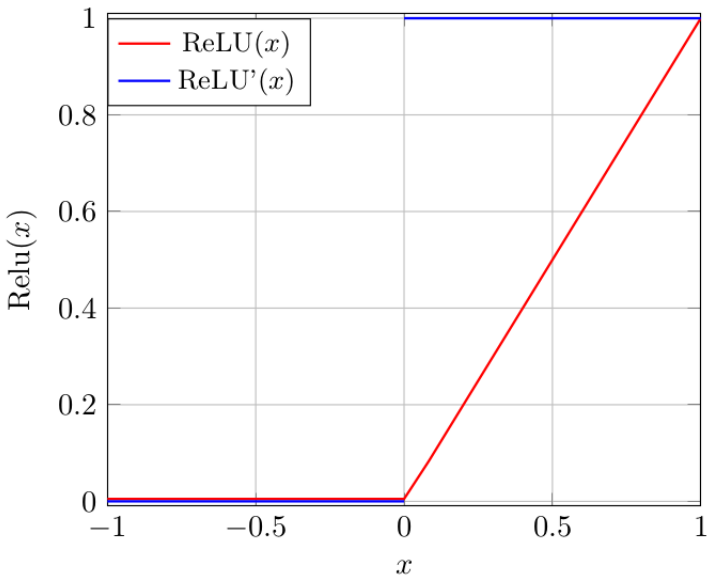


FIGURE 1.13 – Représentation de la fonction ReLU et sa dérivée

1.4.3 Perceptron multi-couches

Afin de complexifier la frontière de décision d'un perceptron unique, on peut alors former un ensemble de perceptrons inter-connectés, appelé réseau de neurones ou perceptron multi-couche. Une couche d'un tel réseau est constituée de N neurones, produisant N activations. Ces sorties peuvent devenir alors de nouvelles entrées pour une nouvelle couche de neurones.Avec les réseaux de neurones à couches multiples, nous avons des neurones artificiels organisés en groupes appelés couches. En nous appuyant sur le concept de couche, nous constatons que le réseau de neurones multicouches possède les caractéristiques suivantes :

- Une seule couche d'entrée ;
- Une ou plusieurs couches cachées, entièrement connectées ;
- une seule couche de sortie

Dans un réseau de neurone, les neurones de chaque couche (représentés par des cercles) sont entièrement connectés à tous les neurones de toutes les couches adjacentes.Les neurones de chaque couche utilisent tous le même type de fonction d'activation (la plupart du temps). Pour la couche d'entrée, on a en entrée un ensemble de valeurs sous forme vectorielle brute. L'entrée des neurones des autres couches est la sortie des neurones de la couche précédente. Lorsque les données transitent par le réseau, elles sont influencées par les poids de connexion et le type de fonction d'activation.

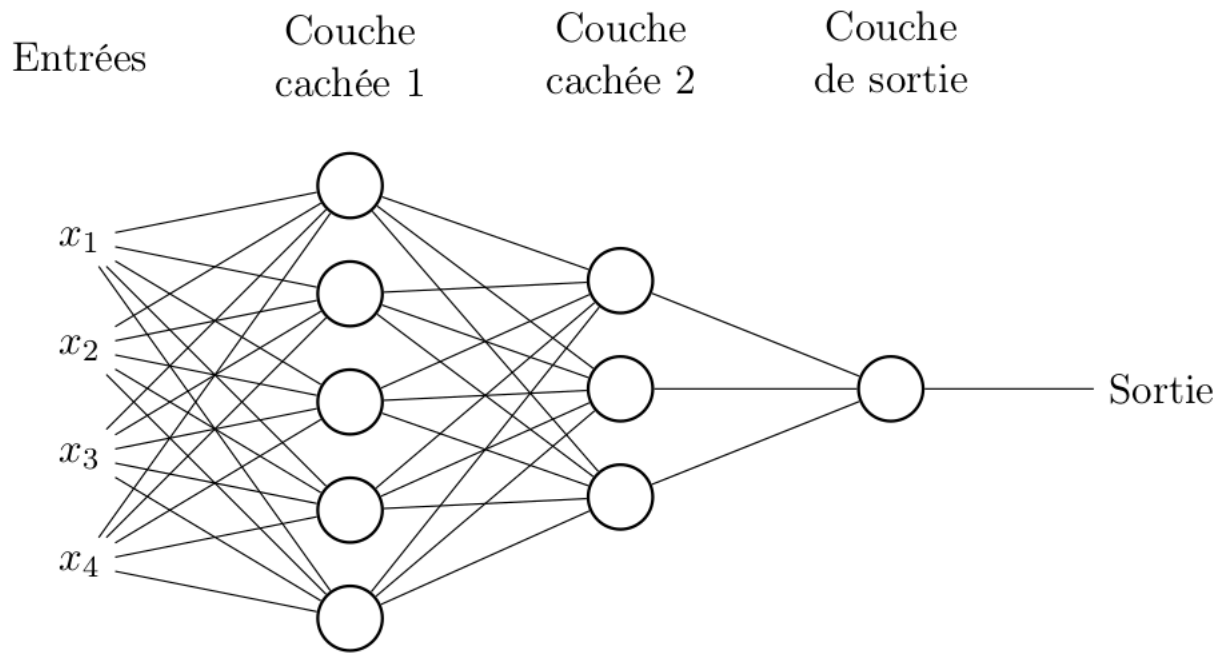


FIGURE 1.14 – Exemple d'un réseau neurone à deux couches cachées

La couche d'entrée : Cette couche est la façon dont nous obtenons des données d'entrée (vecteurs) qui seront introduites dans notre réseau. Le nombre de neurones dans une couche d'entrée est généralement le même nombre que l'entité d'entrée sur le réseau. Les couches en entrée sont suivies d'une ou de plusieurs couches cachées (explications dans le paragraphe suivant). Les couches d'entrée dans les réseaux de neurones à "feed-forward" classiques sont entièrement connectées à la couche cachée suivante, mais dans d'autres architectures de réseau, la couche d'entrée peut ne pas être entièrement connectée.

Les couches cachées : Il existe une ou plusieurs couches cachées dans un réseau de neurones à "feed-forward". Les valeurs de poids sur les connexions entre les couches indiquent comment les réseaux de neurones codent les informations apprises extraites des données brutes d'apprentissage. Les couches cachées sont la clé pour permettre aux réseaux de neurones de modéliser des fonctions non linéaires.

La couche de sortie : Nous obtenons la réponse ou la prédiction de notre modèle grâce à la couche de sortie. Étant donné que nous mappons un espace d'entrée vers un espace de sortie avec le modèle de réseau neuronal, la couche de sortie nous fournit une sortie en fonction des données reçues par la couche d'entrée. Selon la configuration du réseau de neurones, la sortie finale peut être une sortie à valeur réelle (régression) ou un ensemble de probabilités (classification). Ceci est contrôlé par le type de fonction d'activation que nous utilisons sur les neurones de la couche de sortie. La couche de sortie utilise généralement une fonction d'activation softmax ou sigmoïde pour la classification.

les connexions entre les couches : Dans un réseau entièrement connecté, les connexions entre les couches sont les connexions sortantes de tous les neurones de la couche précédente à tous les neurones de la couche suivante. Les poids de chaque connexion sont modifiés progressivement au fur et à mesure que notre algorithme trouve la meilleure solution possible avec l'algorithme d'apprentissage de la "rétro-propagation". Ainsi ces poids sont mis à jour au fur et à mesure de l'entraînement afin de minimiser le plus possible les erreurs.

Cependant jetons un coup d'œil sur la manière dont se fait l'entraînement dans un réseau de neurone. Un réseau neuronal artificiel bien entraîné a des poids qui amplifient le signal et atténuent le bruit. Un poids plus important signifie une corrélation plus étroite entre un signal et le résultat du réseau. Les entrées associées à des poids élevés affecteront davantage l'interprétation des données par le réseau que les entrées associées à des poids plus faibles.

Le processus d'apprentissage pour tout algorithme d'apprentissage utilisant des poids consiste à réajuster les poids et les biais, en rendant certains plus petits et d'autres plus grands, attribuant ainsi une signification à certaines informations et en minimisant d'autres. Cela aide notre modèle à déterminer quelles caractéristiques sont plus liées aux résultats et ajuste les poids et les biais en conséquence.

Dans la plupart des jeux de données, certaines caractéristiques sont fortement corrélées avec certaines étiquettes (par exemple, la superficie en pieds carrés correspond au prix de vente d'une maison). Les réseaux de neurones apprennent ces relations à l'aveuglette en faisant des suppositions en fonction des entrées et des pondérations, puis en mesurant la précision des résultats. Les fonctions de perte dans les algorithmes d'optimisation, tels que la descente de gradient stochastique (SGD), permettent au réseau de faire de bonnes suppositions et le pénalisent pour les mauvaises. Cette fonction de perte déplace les paramètres du réseau vers de bonnes prévisions s'éloignant ainsi des mauvaises [31].

Une autre façon de regarder le processus d'apprentissage consiste à considérer les étiquettes comme des théories et l'ensemble des caractéristiques comme une preuve. Nous pouvons ensuite faire l'analogie voulant que le réseau cherche à établir la corrélation entre la théorie et la preuve. Le modèle tente de répondre à la question «quelle théorie les données probantes soutiennent-elles ?». C'est là qu'intervient la méthode d'apprentissage appelée : **l'apprentissage par rétro-propagation** [31].

1.4.4 Apprentissage

Dans un réseau de neurone nous savons à présent qu'il existe un processus appelé la propagation dans lequel des informations sont introduites dans le réseau à partir de la couche d'entrée puis propagées pour obtenir nos valeurs de sortie qui sont ensuite comparées aux valeurs réelles que nous avons dans notre ensemble de données d'entraînement, nous pouvons

ainsi calculer les erreurs, qui sont propagées dans le réseau dans la direction opposée, ce qui nous permet d'entraîner le réseau en ajustant les poids. Donc, la chose la plus importante à retenir ici est que la rétro-propagation est un algorithme avancé basé sur des calculs mathématiques très intéressants et sophistiqués qui nous permettent d'ajuster les poids. Ces poids sont alors mis à jour simultanément et automatiquement de manière indépendante.

L'énorme avantage de la rétro-propagation, c'est que pendant le processus de propagation en raison de la structure de l'algorithme, il est possible d'ajuster tout un chemin en même temps de telle sorte qu'on puisse déterminer l'influence des erreurs sur les poids du réseau de neurones. D'où le principe fondamental de la rétro-propagation. Ce principe est piloté par un ensemble de formule mathématiques décrit dans le papier [40].

Résumons donc le processus de rétro-propagation à travers les étapes suivantes :

- Etape 1 : initialiser aléatoirement les poids à de petits nombres proches de zéro mais non nuls. Car vu que c'est la première étape il nous faut des poids pris aléatoirement afin d'être corrigés dans les étapes suivantes jusqu'à ce que l'erreur soit minimisée.

- Etape 2 : faire rentrer nos données d'observation dans le réseau à partir de notre couche d'entrée en attribuant à chaque neurone une et une seule caractéristique parmi les données d'observation.

- Etape 3 : faire la propagation en avant : de gauche à droite, ainsi les neurones sont activés de telle sorte que l'impact de l'activation de chaque neurone soit limité par les poids. Fondamentalement il s'agit de déterminer l'importance de l'activation de chaque neurone puis de la propagée de gauche à droite jusqu'à obtenir le résultat de prédiction.

- Etape 4 : comparer le résultat prédit au résultat réel et calculer l'erreur générée.

- Etape 5 : faire la propagation en arrière : de droite à gauche, et les poids sont mis à jour en fonction de leur responsabilité dans l'erreur générée. Le taux d'apprentissage décide alors de comment ajuster les poids.

- Etape 6 : répéter l'étape 1 à 5 et mettre à jour les poids après chaque observation (apprentissage par renforcement) ou après un ensemble d'observation (apprentissage par lots). Cette méthode de faire est appelé le gradient descendant qui consiste à minimiser l'erreur après une ou plusieurs observations grâce à une fonction d'optimisation qui récompense le réseau pour les bonnes suppositions et le pénalise pour les mauvaises en estimant l'erreur entre la valeur réelle d'un élément et la valeur prédite par le réseau pour cet élément.

- Etape 7 : lorsque l'ensemble des observations est passé par le réseau, cela fait une époque. Refaire plusieurs époques afin de minimiser encore plus l'erreur

1.5 Critiques de l'existant et contribution de notre travail

1.5.1 Critiques de l'existant

Malgré les bons résultats fournis, les systèmes existants, présentés dans ce chapitre, ne sont pas parfaits. Il convient donc de souligner quelques limitations qui situent et justifie notre travail.

La caractérisation du phénomène à comprendre ou de l'objet à apprendre dans un système d'apprentissage est une étape critique pour avoir un classifieur performant. Cependant, la plupart des méthodes existantes ne présentent pas une étude formelle permettant de s'assurer que les caractéristiques utilisées sont informatives et discriminantes.

En conséquence,

- Il devient parfois difficile d'identifier les caractéristiques et les propriétés du phénomène observé ou de l'objet à apprendre qui ont contribué aux résultats obtenus.
- Il devient parfois difficile de savoir quelles caractéristiques peuvent être combinées pour arriver à un système performant qui apprend bien (ne sur-apprend pas surtout).
- Il devient souvent évident qu'une collecte conséquente de données est nécessaire pour avoir la chance de couvrir tous les paramètres nécessaires à l'apprentissage d'un phénomène et pouvoir construire un système d'apprentissage performant.

1.5.2 Contribution

Notre travail s'inscrit dans la logique de pallier ces limites de l'existant, en faisant une étude préalable des caractéristiques pour ne retenir que celle qui sont pertinentes et informatives pour l'apprentissage automatique d'un système de vérification de signature afin de s'assurer de la conception d'un système performant. Cette étude sera suivit d'une étude de l'architecture des réseaux de neurones (notre méthode d'apprentissage) en vue de choisir les paramétrages adéquats pour un vérificateur de signature performant. Finalement, en conséquence d'une étude de caractéristiques adéquate une collecte importante de signatures ne sera pas nécessaires pour notre vérificateur.

Conclusion

Dans ce chapitre nous avons pu retenir que la signature manuscrite est une caractéristique propre à chaque être humain mais qu'elle n'a pas été à l'abri de falsifications. C'est dans ce sens que sont apparus les systèmes de vérification statique (vérification effectuée à partir d'une image brute numérisée) et de vérification dynamique (vérification à partir d'éléments dynamiques recueillis pendant que la signature est effectuée tels que le temps, la pression, la vitesse

etc.). On a également donné les généralités sur l'apprentissage automatique , sa manière de fonctionner et l'évaluation de ses performances en mettant un accent sur les réseaux de neurones : sa conception et sa manière d'apprendre.

Chapitre 2

Matériels et méthodes

Introduction

L'objectif étant d'avoir un système de vérification de signature dynamique performant nous aurons besoin d'un certain nombre d'outils accompagnés d'une méthodologie d'étude adéquate pour atteindre cet objectif. C'est ainsi que dans un premier temps, nous aborderons ces différents outils à utiliser et ensuite nous mettrons en place une méthodologie permettant de combiner ces différents outils afin d'atteindre nos résultats.

2.1 Outils

2.1.1 Environnement de développement

Système d'apprentissage

Kit système

Pour nos travaux d'étude et d'application nous avions eu à utiliser un ordinateur portatif dont les caractéristiques sont les suivantes :

- **Modèle** : OMEN HP Laptop
- **Système d'exploitation** : Ubuntu 18.04 LTS
- **Architecture** : 64 bits
- **Processeur** : Intel(R) Core(TM) i7-7700HQ CPU @ 2,80GHz
- **Mémoire RAM** : 8 Go

Kit d'apprentissage

Langage de programmation : alors que l'apprentissage automatique gagne de plus en plus de terrain, presque tous les langages populaires ajoutent des modules et des bibliothèques pour faciliter les tâches d'apprentissage machine. Les quatre langages que nous considérons être les meilleurs sont python, Java, Matlab et R. Cependant nous allons nous baser sur les caractéristiques telles que : la communauté afin de pouvoir trouver rapidement des solutions en cas de dysfonctionnements dans les codes développés ; la disponibilité des bibliothèques d'apprentissage automatique afin d'avoir un choix très large et de pouvoir choisir le meilleur ; la rapidité d'exécution car les algorithmes d'apprentissage machine mettent beaucoup de temps à s'exécuter pendant la phase d'entraînement et enfin le coût d'utilisation car nous ne disposons pas de beaucoup de moyens financiers. Le Tableau 2.1 nous montre un comparatif de ces quatre langages de programmation.

TABLEAU 2.1 – Comparatif des langages de programmation Python, Java, Matlab et R

	Python	**Java**	**Matlab**	**R**
Coût d'utilisation	Gratuit	Gratuit	Payant	Gratuit
Communauté	Très large	Très large	Moyenne	Petite
Rapidité d'exécution	Rapide	Rapide	Rapide	Rapide
Outils d'apprentissage	Très nombreux	Nombreux	Nombreux	Nombreux

Pour nos travaux *Python* reste un bon compromis par rapport à nos critères. C'est un langage de programmation orienté objet et qui a un caractère multi-plateforme. C'est un langage puissant, à la fois facile à apprendre et riche en possibilités. Il est en outre, très facile d'étendre les fonctionnalités existantes, ainsi, il existe des bibliothèques qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées. Il est bien évidemment très adapté pour des applications d'apprentissage automatique.

Anaconda : est une distribution libre et gratuite des langages de programmation Python et R ainsi que de nombreux autres outils tel que spyder(un environnement de développement intégré abrégé EDI en français pour écrire du code python), matplotlib(pour visualiser des données à travers les courbes)..., très utiles pour des applications liées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique) et qui vise à simplifier la gestion et le déploiement d'applications relatives à ce dernier [41].

Tensorflow : est un framework de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multi-dimensionnelles, appelées Tenseurs (Tensor). Un Tenseur à deux dimensions est l'équivalent d'une matrice. Ainsi sa documentation très riche fait d'elle une bibliothèque facile à utiliser [42].

Keras : est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow ou Theano. Il a été développé en mettant l'accent sur l'expérimentation rapide. Être capable d'aller de l'idée à un résultat avec le moins de délai possible est la clé pour faire de bonnes recherches. Keras a été conçue comme une interface plutôt que comme un cadre d'apprentissage end to end. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilite la configuration des réseaux neuronaux [43].

Scikit-learn : est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec d'autres bibliothèques libre Python, notamment NumPy et SciPy [44].

Application du système de vérification dynamique

Serveur web pour le back-end

Afin de pouvoir mettre en place le serveur qui va se charger de faire la liaison entre notre système de vérification et le client web nous avons opté pour un ensemble de bibliothèques python permettant de mettre en place notre serveur que sont :

Flask : est un framework web python qui va nous permettre de simuler un serveur web en local sur notre machine.

SocketIO : est une bibliothèque basée sur un système d'émission et de réception d'évènements afin de déclencher une action donnée. Elle est disponible en JavaScript et en Python afin de pouvoir faire communiquer le client et le serveur. Elle nous permettra principalement d'envoyer une notification nous signalant que la phase d'entraînement d'un modèle est terminée.

Kit web pour le front-end

Pour pouvoir communiquer avec le serveur, nous mettrons en place une application basée sur les technologies web à savoir :

HTML (HyperText Markup Language) : Son rôle est de gérer et organiser le contenu. C'est donc en HTML que nous écrirons ce qui doit être affiché sur la page : du texte, des liens, des images etc [45].

CSS Cascading Style Sheets, aussi appelées Feuilles de style : le rôle du CSS est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, taille du texte…) [45].

JavaScript : c'est un langage de programmation interprété, c'est-à-dire qu'il a besoin d'un interpréteur pour pouvoir être exécuté. Il est utilisé majoritairement au sein des pages web. Tout comme le HTML, le Javascript est exécuté par le navigateur de l'internaute : on parle d'un comportement client-side, par opposition au server-side lorsque le code est exécuté par le serveur. Il va nous permettre de gérer les évènements au niveau de l'internaute comme par exemple au niveau du formulaire d'inscription pour vérifier si les champs sont bien remplis [46].

Bootstrap : c'est une bibliothèque CSS, mais pas seulement, puisqu'il embarque également des composants HTML et JavaScript. Il comporte un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style pour les boutons, les formulaires, la navigation… elle prend en compte également les tailles des écrans très variées [47].

Angularjs : est une bibliothèque javascript construite autour de concepts clés à savoir : L'Architecture MVC (Modèle-Vue-Contrôleur) qui permet de séparer la vue de l'ensemble logique de l'application Elle aura pour rôle d'afficher les informations reçues par le serveur [48].

2.1.2 Les données

Ensemble de données (SVC 2004) : SVC est l'abréviation du premier concours international de vérification de la signature qui consistait en deux tâches séparées en deux dossiers que sont "Task 1" pour la tâche 1 et "Task 2" pour la tâche 2 [49]. Chaque base de données de signatures créée pour la compétition comprend 40 séries de signatures, dont 20 authentiques (enregistrées en deux sessions) et 20 contrefaçons qualifiées prélevées sur au moins 4 autres contributeurs autorisés à voir la signature originale et à s'exercer avant d'effectuer les contrefaçons. La base de données ne contient que des informations de coordonnées pour la tâche 1 ; pour la tâche 2 la pression et l'orientation du stylet sont également incluses. Ainsi nos travaux d'études se feront uniquement sur la base de données de la tâche 2 vu que les données de la tâche 1 sont

incomplètes. Il faut noter que le dossier "Task 2" de la tâche 2 est un ensemble de signatures de 40 utilisateurs dont 20 authentiques et 20 falsifications pour chaque utilisateur ce qui fait 40 signatures pour un utilisateur. Chaque signature (authentique ou falsifiée) est stockée dans un fichier texte séparé. Les noms de fichier sont au format USERx_y.txt, où x (1..40) indique l'utilisateur et y (1..40) indique une instance de signature de l'utilisateur correspondant, avec les 20 premiers (1..20) représentants les signatures authentiques et le reste (21..40) représentant des contrefaçons qualifiées fournies par les autres utilisateurs. Dans chaque fichier texte, la signature est simplement représentée par une séquence de points. La première ligne stocke un entier unique qui est le nombre total de points dans la signature. Chacune des lignes suivantes correspond à un point de la signature caractérisé par sept caractéristiques énumérées dans l'ordre suivant :

- **Abscisse X** : position du curseur mise à l'échelle le long de l'axe des X
- **Ordonné Y** : position du curseur mise à l'échelle le long de l'axe des Y.
- **Horodatage** : système de temps à partir duquel l'événement a été effectué.
- **Statut de la pointe du stylet** : statut courant de la pointe (0 pour la pointe levée et 1 pour la pointe en bas).
- **Azimut** : angle du stylet avec l'axe z (sens horaire).
- **Altitude** : position du bout supérieur du stylet par rapport à l'axe z.
- **Pression** : valeur de la pression normale.

La figure 2.1 présente un exemple d'une signature prise de la base de données de la tâche 2 et les différentes caractéristiques qui lui sont associées.

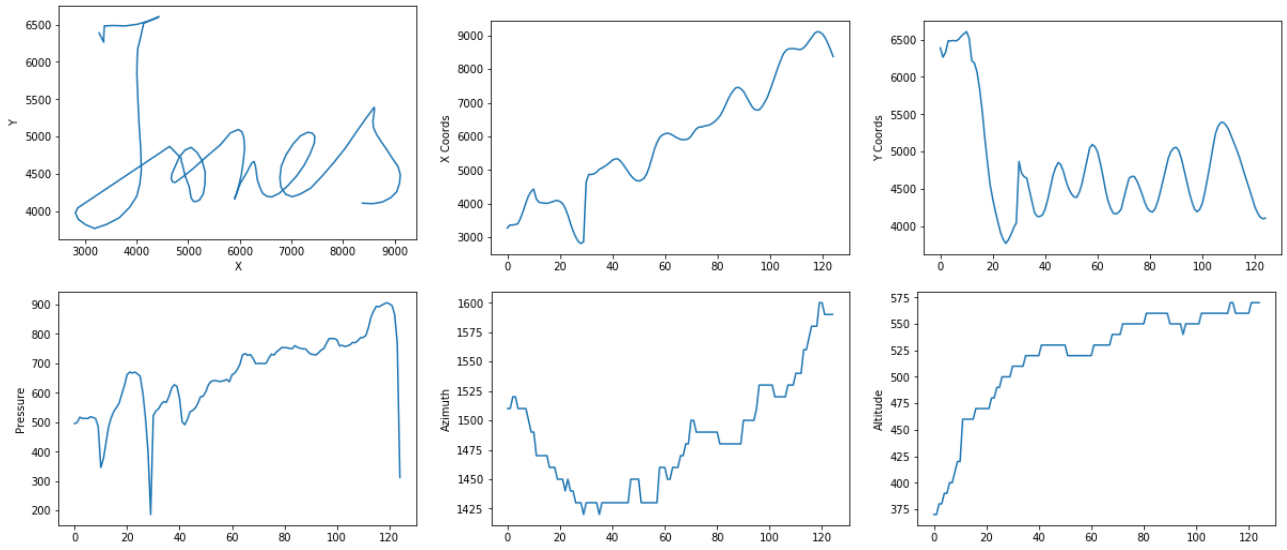


FIGURE 2.1 – Signature, coordonnées x,y, pression, azimut et altitude associés à la signature

2.1.3 Les caractéristiques extraites des données

La sélection des caractéristiques informatives est au cœur de la conception d'un classifieur performant. Dans le cas d'une signature ces caractéristiques à considérer doivent nous permettre d'évaluer si une signature est falsifiée ou originale. Pour se faire nous avons besoin de caractériser le comportement d'un être humain qui signe. Cette caractérisation peut être effectuée essentiellement de deux manières :

- On peut prendre l'image de la signature (version scannée ou par prise de vue) et extraire des caractéristiques comme la texture, la quantité de couleurs,... Ces caractéristiques sont considérées comme des caractéristiques morphologiques.
- On peut aussi utiliser une tablette avec un stylet et quand la personne signe on récupère directement les caractéristiques. Ainsi, des caractéristiques morphologiques comme abstraites comme la pression et la position du stylo, le temps qu'il a fallut pour signer,... peuvent être extraites.

Dans ce travail, nous avons choisi cette deuxième approche car nous faisons l'hypothèse que la caractérisation de la signature d'une personne prend en compte des *critères morphologiques ou statiques* mais aussi des *critères abstraites ou dynamiques* qu'on ne peut obtenir facilement à partir d'une image.

Après une signature avec la tablette et le stylet, nous disposons des informations de base telles que décrite dans la section 2.1.2 à savoir :

- Une distribution des abscisses x.
- Une distribution des ordonnées y.
- Une distribution des horodatages t.
- Une distribution des statuts du stylets.
- Une distribution des azimuts az.
- Une distribution des altitudes al.
- Une distribution des pressions p.

A partir de ces informations nous pouvons calculer d'autres distributions comme :

- La distribution des coordonnées polaires r et θ à partir des coordonnées cartésiens de chaque point grâce à la formule :

$$\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases} \tag{2.1}$$

- La distribution des temps relatifs entre deux points consécutifs tr à partir de l'horodatage de chaque point grâce à la formule :

$$tr = \frac{t_2 - t_1}{t_{fin} - t_{debut}} \tag{2.2}$$

- La distribution des vitesse linéaires v entre deux points qui est fonction des coordonnées cartésiens et du temps grâce à la formule :

$$v = \frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{t_2 - t_1} \times (t_{fin} - t_{debut}) \tag{2.3}$$

- La distribution des accélérations a entre deux points qui est fonction des vitesses linéaires et du temps grâce à la formule :

$$a = \frac{\mid v_2 - v_1 \mid}{t_2 - t_1} \times (t_{fin} - t_{debut}) \tag{2.4}$$

- La distribution des vitesses angulaires entre deux points qui est fonction des coordonnées polaires θ et du temps grâce à la formule :

$$\omega = \frac{\mid \theta_2 - \theta_1 \mid}{t_2 - t_1} \times (t_{fin} - t_{debut}) \tag{2.5}$$

. Ensuite, nous pourrons calculer les caractéristiques de chaque signature en utilisant les opérations statistiques telles que la *moyenne*, la *médiane* et l'*écart-type* de ces distributions.

Moyenne : La moyenne d'une distribution, de formule $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, est l'une des plus anciennes méthodes utilisée pour caractériser un ensemble de données et indiquer une tendance central. Ainsi, elle représente un bon indicateur pour résumer une distribution de plusieurs éléments.

Médiane : La médiane d'une distribution se calcule (i) en triant les valeurs de la distribution par ordre, puis (ii) en prenant la valeur du milieu si la taille de la distribution est impaire et ou la demi-somme des deux valeurs du milieu si la taille est paire. La médiane permet de situer la valeur centrale de la distribution. Elle est une mesure qui permet de juger de la qualité de la moyenne.

Ecart-type : L'écart-type d'une distribution, de formule $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$, mesure la dispersion des valeurs de la distribution autour de la moyenne (et de la médiane). Elle est une mesure qui permet de juger de la qualité de la moyenne.

La moyenne nous permettra de caractériser chaque signature avec une seule valeur qui sera sa valeur principale. A cette valeur est associée sa valeur médiane et l'écart-type qui permet-

trons de juger si cette moyenne est une vraie valeur centrale et combien de fois les valeurs de la distribution sont regroupées ou dispersée par rapport à elle. Ceci constitue un indicateur intéressant pour faire la différence entre une signature originale et sa falsification.

Cette manipulation est importante car elle permettra non seulement d'extraire de bon indicateurs mais aussi de garder la même proportion dans la caractérisation des signatures. En effet, une personne qui appose une signature a tendance à signer de manière proportionnelle au support qu'il a devant lui. Cette variation peut avoir un impact néfaste sur l'apprentissage puisque les bases de comparaison ne seront pas forcément les mêmes.

Le Tableau 2.2 présente l'ensemble des vingt-sept(27) caractéristiques retenues pour l'étude. Ces caractéristiques sont regroupées en caractéristiques statiques et dynamiques en fonction du fait qu'elles tiennent compte des caractéristiques morphologiques ou abstraites. Pour extraire ces caractéristiques on parcours le fichier de signature, issue de la tablette, et pour chaque point dans ce fichier on effectue les opérations escomptées. Pour chaque caractéristique nous donnons la description mais aussi l'importance quand cela est applicable.

TABLEAU 2.2 – Ensemble des caractéristiques extraites

No	**Type caractéristique**	**Nom de la caractéristique**	**Description/Formulation**	**Importance**
1	Statique	Moyenne des abscisses relatives	$\overline{x} = \frac{1}{n}\sum_{i=1}^{n}\frac{x_i}{x_{max}}$	Avoir une idée de la distribution des coordonnées cartésiennes sans laisser influencer la taille de la signature.
2	Statique	Ecart-type des abscisses relatives	$\sigma_x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{x_i}{x_{max}} - \overline{x})^2}$	
3	Statique	Moyenne des ordonnées relatives	$\overline{y} = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i}{y_{max}}$	
4	Statique	Ecart-type des ordonnées relatives	$\sigma_y = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{y_i}{y_{max}} - \overline{y})^2}$	
5	Statique	Moyenne du ratio des coordonnées	$\overline{xy} = \frac{1}{n}\sum_{i=1}^{n}\frac{x_i}{y_i}$	Déterminent les relations éventuelles entre les abscisses X et les ordonnées Y.
6	Statique	Ecart-type du ratio des coordonnées	$\sigma_{xy} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{x_i}{y_i} - \overline{xy})^2}$	
7	Statique	Ratio de l'aire occupée par la signature	$ratio = \frac{x_{max}-x_{min}}{y_{max}-y_{min}}$	Connaître la proportion de la signature.
8	Statique	Moyenne des angles θ	$\overline{\theta} = \frac{1}{n}\sum_{i=1}^{n}\theta_i$	Il serait judicieux d'avoir une vue globale de la signature suivant un autre repère et d'y extraire des caractéristiques.
9	Statique	Ecart-type des angles θ	$\sigma_\theta = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\theta_i - \overline{\theta})^2}$	
10	Statique	Médiane des angles θ	θ_{median}	
11	Statique	Moyenne des rayons r relatives	$\overline{r} = \frac{1}{n}\sum_{i=1}^{n}\frac{r_i}{r_{max}}$	

Table 2.2 Ensemble des caractéristiques extraites(suite)

12	Statique	Ecart-type des rayons r relatives	$\sigma_r = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{r_i}{r_{max}} - \overline{r})^2}$	
13	Dynamique	Moyenne des temps relatifs	$\overline{tr} = \frac{1}{n}\sum_{i=1}^{n} tr_i$	Un falsificateur aura tendance à mettre plus de temps à effectuer une signature qu'une personne authentique.
14	Dynamique	Ecart-type des temps relatifs	$\sigma_{tr} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(tr_i - \overline{tr})^2}$	
15	Dynamique	Moyenne des vitesses linéaires	$\overline{v} = \frac{1}{n}\sum_{i=1}^{n} v_i$	Même un faussaire expérimenté peut avoir à ralentir le plus souvent pour avoir une imitation géométrique parfaite, alors que l'utilisateur légitime reste constant pour tous ses échantillons.
16	Dynamique	Ecart-type des vitesses linéaires	$\sigma_v = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \overline{v})^2}$	
17	Dynamique	Moyenne des accélérations	$\overline{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$	Ces caractéristiques dynamique tirent également parti de la faible confiance de l'imposteur lors de la falsification.
18	Dynamique	Ecart-type des accélérations	$\sigma_a = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \overline{a})^2}$	

Table 2.2 Ensemble des caractéristiques extraites(suite)

19	Dynamique	Moyenne des vitesses angulaires	$\overline{\omega} = \frac{1}{n}\sum_{i=1}^{n}\omega_i$	En considérant qu'on fait une analyse suivant deux types de repères considérons également la vitesse dans le système polaire appelée vitesse angulaire.
20	Dynamique	Ecart-type des vitesses angulaires	$\sigma_\omega = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\omega_i - \overline{\omega})^2}$	
21	Dynamique	Moyenne des pressions	$\overline{p} = \frac{1}{n}\sum_{i=1}^{n}p_i$	La pression est exercée au bout du stylo et ne peut pas être facilement imitée par le faussaire.
22	Dynamique	Ecart-type des pressions	$\sigma_p = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - \overline{p})^2}$	
23	Dynamique	Moyenne des azimuts	$\overline{az} = \frac{1}{n}\sum_{i=1}^{n}az_i$	Certains individus inclinent le stylet tandis que d'autres le tiennent beaucoup plus droit.
24	Dynamique	Ecart-type des azimuts	$\sigma_{az} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(az_i - \overline{az})^2}$	
25	Dynamique	Moyenne des altitudes	$\overline{al} = \frac{1}{n}\sum_{i=1}^{n}al_i$	Ces caractéristiques déterminent comment chaque individu tien le stylet. Soit en hauteur ou soit bas.
26	Dynamique	Ecart-type des altitudes	$\sigma_{al} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(al_i - \overline{al})^2}$	

Table 2.2 Ensemble des caractéristiques extraites(suite)

27	Dynamique	La transition	Le nombre de transitions de la valeur d'état du stylet de 0 à 1 est compté, ce qui indique le nombre de fois que le stylet a été levé lors de la signature.	Le nombre de fois qu'une personne lève le stylo lors de la signature est exactement le même pour deux échantillons authentiques du même utilisateur.

Ce qui nous fait en fin de compte un total de vingt-sept (27) caractéristiques à exploiter pour notre étude afin d'atteindre les meilleures performances possibles.

2.1.4 Mesure de performance

Afin de pouvoir évaluer nos performances tout au long de l'étude on décide de considérer les métriques que sont **la F-mesure**, **l'AUC**, **le rappel**, **la précision**. Cependant essayons de comprendre l'information qu'apporte chacune de ces métriques

- La précision : est le rapport entre les observations positives correctement prédites et le total des observations positives prédites (cf formule 1.1). L'information qu'apporte cette métrique lorsqu'on considère l'objet de cette étude est que de toutes les signatures prédites comme authentiques, combien sont réellement authentiques. De plus on remarque que sa formule est fonction du nombre de Faux Positifs. Plus ce nombre est grand plus la précision s'éloigne de sa valeur idéale 1. La précision est un bon moyen d'évaluation lorsqu'on cherche à éviter des éléments Faux Positifs.

  Pour notre système on peut accepter à la limite qu'une signature authentique soit considérée comme étant fausse car même un signataire authentique est soumis à plusieurs conditions lors de la signature ainsi le stylet utilisé, le support sur lequel il signe, son humeur etc, peuvent faire que la signature qu'il produit ait l'air d'une falsification. Mais il serait inconcevable tout de même qu'une signature falsifiée soit considérée comme étant vraie (élément classé comme étant un Faux Positif). Ainsi lors de l'évaluation de nos performances, nous mettons un accent particulier sur la précision.

- Le rappel : est le nombre de Vrai Positifs sur le nombre d'exemples effectivement positifs dans l'ensemble de données (cf formule 1.2). L'information qu'apporte cette métrique lorsqu'on considère l'objet de cette étude est que de toutes les signatures authentiques, combien ont été prédites comme authentiques. Contrairement à la précision, cette métrique est idéale pour vérifier la performance lorsqu'on cherche à avoir le moins d'éléments Faux Négatifs.

- La F-mesure : est une combinaison de la précision et du rappel (cf formule 1.3). Elle mesure l'exactitude du modèle. Elle est intéressante lorsqu'on recherche un équilibre entre la précision et le rappel.

- L'AUC (ou en français surface sous la courbe) est une métrique pour une classification binaire. Il indique l'efficacité du modèle à distinguer les classes. Plus l'AUC est élevée, meilleur est le modèle qui distingue correctement les signatures authentiques et falsifiées.

2.2 Méthode d'étude

Tout au long de l'étude, les différentes phases pour l'apprentissage et l'évaluation du modèle par les réseaux de neurones, de manière générale sont résumées suivant les étapes de la figure 2.2 :

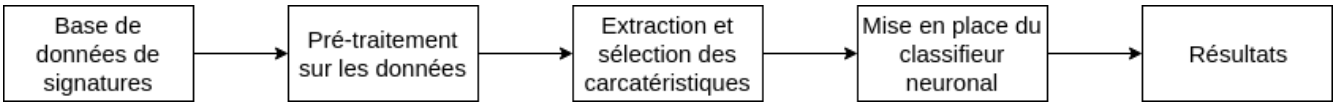


FIGURE 2.2 – Phases d'apprentissage et d'évaluation par les réseaux de neurones

Base de données de signatures : l'ensemble des signatures provenant du dossier "Task 2" contenues dans la base de données SVC 2004 dans la sous section 2.1.2 portant sur les données.

Pré-traitement sur les données : En analysant l'ensemble de données obtenu, on observe que les horodatages de quelques points consécutifs pour certains échantillons de signature étaient les mêmes. Ceci est une anomalie car la pointe du stylet ne peut pas se trouver à deux endroits différents simultanément sur la surface de la tablette. En raison de cette anomalie, on aura du mal à calculer certaines caractéristiques dynamiques.Une autre anomalie observée dans l'ensemble de données est que certains points ayant pour statut de stylet 0 signifiant que la pointe est levée ont été rajoutés après l'obtention du dernier point de la signature (dernier point ayant pour statut du stylet 1). L'élimination de ces points supplémentaires qui ne faisaient pas partie de la signature était donc nécessaire. Une fois ces points éliminés, on doit procéder à une normalisation de toutes les signatures. Les étapes mises en œuvres pour palier à ces anomalies et procéder à la normalisation sont les suivantes :

- Pour tous les échantillons de signatures, supprimez les jetons à la fin de l'échantillon de signature pour lesquels le statut du stylet est égale à 0.
- Pour tous les échantillons de signature, si l'horodatage de deux jetons consécutifs est le même, la valeur d'horodatage du deuxième jeton est égale à la moyenne des valeurs d'horodatage des jetons situés au-dessus et au-dessous de lui.
- Il est souhaitable que les valeurs des coordonnées de tous les points de la signature soient considérées en fonction de leur distance par rapport à un point de référence situé sur la signature elle-même. Une solution à ce problème consiste à déplacer l'origine (0,0) sur la coordonnée qui a les valeurs de coordonnées x et y minimales. Cela peut être fait en soustrayant la valeur X minimale et la valeur Y minimale de la signature des valeurs X et Y. Soit pour chaque point (X-Xmin, Y-Ymin).

Extraction et sélection des caractéristiques : l'ensemble des vingt-sept(27) caractéristiques (dynamiques et statiques) calculées dans la sous section 2.1.3 seront considérées tout au long

de notre étude.

Mise en place du classifieur neuronal : les caractéristiques extraites des signatures sont utilisées pour créer un réseau de neurones entraîné à l'aide de l'algorithme de rétro-propagation d'erreur pour chaque utilisateur. Le nombre de couches cachées et de neurones dans ces dernières, le nombre d'époques, les fonctions d'activation et d'autres paramètres du réseau de neurones sont définis et le réseau de neurones est créé sur la base de ces informations.

L'ensemble des signatures concernant chaque utilisateur est divisé en deux parties : l'ensemble d'apprentissage et l'ensemble de test. Dans la phase de l'apprentissage ou d'entraînement, 75% des échantillons de signature (30 signatures) de l'utilisateur sont utilisés et les 25% restants (10 signatures) sont mis de côté pour les tests. Il y a un nombre égal d'échantillons de signatures originaux et falsifiés pour chaque utilisateur.

Du fait qu'on soit dans un apprentissage supervisé, les réseaux seront entraînés jusqu'à ce que la sortie qu'ils génèrent soit aussi proche que possible de la sortie souhaitée. Une fois l'entraînement fini, on estime l'efficacité du réseau de neurones à partir des données de signatures réservées pour le test.

Résultats : A ce niveau on évalue les performances de notre modèle entraîné à partir des métriques calculées telles que la F-mesure, la précision, le rappel et l'AUC.

2.2.1 Évaluation de l'importance des caractéristiques

Pendant cette étape, nous procéderons à la validation des caractéristiques utilisées, c'est à dire pour l'ensemble des caractéristiques, partir de la première caractéristique et rajouter les autres caractéristiques au fur et à mesure dans un réseau de neurones afin de comprendre l'importance de chaque caractéristique en y relevant la F-mesure, la précision, le rappel et l'AUC afin d'évaluer les performances à chaque ajout de nouvelles caractéristiques.

2.2.2 Étude de l'architecture du réseau neuronal optimal

Lorsque les caractéristiques finales seront retenues ,nous évaluerons la méthode, c'est à dire pour la méthode de réseau de neurones qui va être utilisée, tester plusieurs architectures de réseau de neurones et plusieurs paramètres avec l'ensemble des caractéristiques retenues pour l'étude sur chaque architecture et relevé la F-mesure, la précision, le rappel et l'AUC afin de pouvoir évaluer notre architecture à chaque fois et ainsi de dégager l'architecture de réseau le mieux adapté à notre étude.

2.2.3 Mise en place de l'application du système de vérification : cycle V

Le modèle de gestion pour laquelle nous optons pour mettre en place notre application est **le cycle en V**. Le modèle V spécifie une série d'étapes linéaires qui devraient se produire tout au long du cycle de vie, une à la fois, jusqu'à ce que le projet soit terminé. Il est composé d'une phase descendante puis montante, la phase montante envoie des informations vis-à-vis de la phase descendante [50]. Il est composé de trois(3) grandes phases contenant huit(8) étapes de conception d'une application :

- La phase de conception
  - Analyse des besoins et faisabilité
  - Spécifications
  - Conception Architecturale
  - Conception détaillée
- La phase de réalisation
  - Codage
  - Tests unitaires
- La phase de validation
  - Tests d'intégration
  - Test de validation
  - Recette

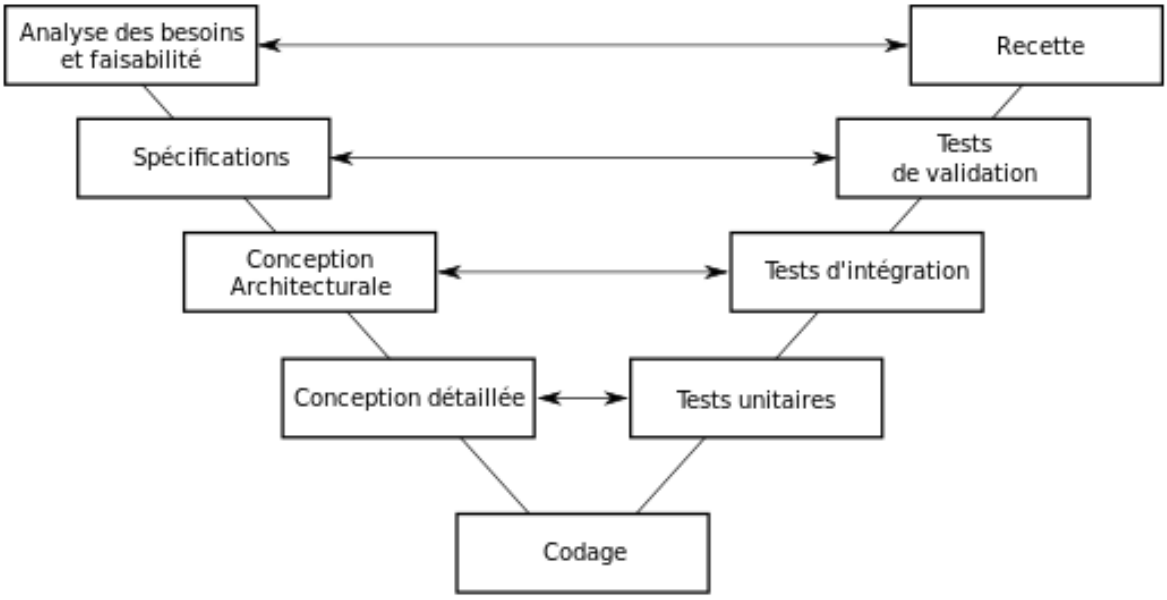


FIGURE 2.3 – Le cycle en V [50]

La phase de conception : le besoin essentiel pour notre réalisation est de mettre en place une application web d'un système de vérification de signature dynamique performant qui saura distinguer une signature authentique d'une falsification. Pour cela grâce à notre méthode d'apprentissage que représente les réseaux de neurones on peut être capable de construire un modèle entraîné pour chaque individu qui soumet un ensemble de ses signatures (authentiques et falsifiées). Ce modèle est ainsi sauvegardé et réutilisé pour prédire l'authenticité lorsqu'on lui soumet dans le futur une signature de cet utilisateur. Afin de pouvoir automatiser tout ce système, nous mettons en place un serveur capable de faire la liaison entre un client web et le système d'apprentissage. Au niveau du client web, on doit pouvoir mettre en place une interface constituée d'un formulaire de soumission de données de signatures pour un entraînement de modèle et pour une prédiction. Également on doit pouvoir aménager un espace qui nous informe des résultats obtenus après un entraînement de modèle ou une prédiction.

La phase de réalisation : dans un premier temps nous installons les bibliothèques qui doivent nous permettre de concevoir ce système. Nous développons ensuite l'interface web qui doit être constitué principalement de deux pages, une page pour l'entraînement du modèle et une autre page pour la prédiction. Chaque page dispose de son formulaire de soumission de données et de son espace pour afficher les résultats. Après ceci nous développons le module capable d'entraîner un modèle à partir des données qui lui ont été soumises et d'enregistrer ce modèle et enfin nous développons le module capable de retrouver le modèle d'un utilisateur et de prédire si oui ou non une signature est fausse. Bien sûr à chaque fois qu'un module est développé, il est testé pour s'assurer de son bon fonctionnement.

La phase de validation : une fois tous nos modules développés, on effectue la liaison grâce à notre serveur et on fait un test global pour s'assurer du bon fonctionnement de l'ensemble du système.

Conclusion

Nous avons présenté dans ce chapitre les différents outils et matériels physiques comme logiciels ainsi que les technologies et la base de données qui vont nous permettre d'effectuer nos travaux. Ensuite nous avons présenté des caractéristiques concrètes qu'on pourrait utiliser pour atteindre des performances satisfaisantes sans oublier la mise en place d'une méthode d'étude afin d'atteindre de bons résultats.

Chapitre 3

Résultats et discussions

Introduction

Ce chapitre nous présente en général les résultats obtenus pour l'étude menée dans l'atteinte de notre objectif qui est de mettre un système de vérification dynamique de signature. Dans un premier temps, nous montrons l'impact des caractéristiques prises par groupe selon leurs natures sur les performances d'un système basé sur les réseaux de neurones. Dans un second temps , une fois que l'impact de ces caractéristiques est évalué et qu'on remarque un apport positif, ces caractéristiques seront utilisées pour étudier les différentes architectures de réseaux de neurones possibles afin d'en dégager l'architecture optimale qui sera utilisée à son tour pour mettre en place notre système de vérification. Nous allons également utiliser d'autres méthodes d'apprentissage automatique comme la machine à vecteur de support et les K-voisins les plus proches, afin de faire une comparaison avec la méthode des réseaux de neurones et aussi comparer nos résultats à ceux des travaux qui ont déjà été menés .

3.1 Détails de l'implémentation d'un réseau de neurones

Suivant les étapes de notre étude, notre réseau de neurones utilisera l'algorithme de rétropropagation pour l'apprentissage. Voici un extrait de code pour construire un réseau de neurones avec quelques explications. Nous en profiterons pour expliquer les différents paramétrages d'un réseau de neurones.

```
1
2 classifier = Sequential()
3
4 ##Couche d'entrée
5 classifier.add(Dense(
6           units = 14 ,
7           activation = 'relu' ,
8           kernel_initializer = 'uniform',
9           input_dim = 27 ))
10
11 ##Couche cachée
12 classifier.add(Dense(
13           units = 14 ,
14           activation = 'relu' ,
15           kernel_initializer = 'uniform' ))
16
17 ##Couche de sortie
18 classifier.add(Dense(
19           units = 1,
20           activation = 'sigmoid' ,
21           kernel_initializer = 'uniform' ))
22
23 ##Définition de l'optimizeur
24 ##et des fonctions d'error et d'évaluation
25 classifier.compile(
26           optimizer = 'sgd',
27           loss = 'binary_crossentropy',
28           metrics = ['accuracy'])
29
30 ##Phase d'entrainement sur une certaine époque
31 classifier.fit(X_train,
32           y_train,
33           batch_size = 3,
34           epochs = 5000)
35
36 ##Evaluation de l'entrainement
37 y_pred = classifier.predict(X_test)
38 y_pred = (y_pred > 0.5)
39 f1_score =  f1_score(y_test,y_pred)
40 recall =  recall_score(y_test, y_pred)
41 precision = precision_score(y_test,y_pred)
42 auc_score = roc_auc_score(y_test,y_pred)
```

FIGURE 3.1 – Code python de la mise en œuvre d'un réseau de neurones avec Keras

- **Ligne 2** : elle s'occupe de l'initialisation du classifieur de notre réseau de neurones.

- **Ligne 4 à 9** : elle consiste à ajouter à la fois notre couche d'entrée constituée de 27 neurones dans ce cas grâce au paramètre *input_dim* et aussi notre première couche cachée constituée de 14 neurones grâce au paramètre *units*, sans oublier la fonction d'activation pour chaque neurone qui est la fonction *relu* (fixée tout au long de notre étude) précisée grâce au paramètre *activation*. Notons également la présence du paramètre *kernel_initializer* qui va nous permettre de fixer des poids aléatoires à aux connexions reliant les neurones au début de l'entraînement à des valeurs très proches de zéro mais non nulles.

- **Ligne 11 à 15** : elle permet de rajouter une nouvelle couche cachée, elle est identique à la ligne décrite plus haut sauf qu'on a plus besoin de préciser les paramètres pour le nombre de neurones dans la couche d'entrée.

- **Ligne 17 à 21** : elle représente notre couche de sortie constituée d'un seul neurone et dont la fonction d'activation est le sigmoïde (fixée tout au long de notre étude).

- **Ligne 23 à 28** : permet de préciser la fonction d'optimisation (de calcul d'erreur) après chaque propagation de gauche à droite grâce au paramètre *optimizer*, dans ce code nous avons utilisé le paramètre *sgd*.

- **Ligne 30 à 34** : cette ligne nous permet de préciser le paramètre *epochs* qui représente le nombre de fois que l'entièreté de la base de données se propage et se rétro-propage à travers le réseau de neurones. Notons aussi la présence du paramètre *batch_size* qui représente le nombre de données dans l'ensemble de la base de données après lequel les poids des connexions sont mis à jour.

- **Ligne 37** : elle nous permet de classifier nos données de tests à partir du modèle entraîné.

- **Ligne 38** : elle permet de fixer le seuil de probabilité de 0,5 pour lequel une sortie en dessus de ce seuil est arrondie à 1 sinon 0 en dessous.

- **Les lignes 39 à 42** : elles évaluent les performances grâce à la F-mesure, le rappel, la précision et l'AUC sur la base des données de test.

3.2 Validation des caractéristiques

Dans cette section, nous menons une étude sur les caractéristiques extraites qui ont été présentées dans la section 2.1.3 afin d'évaluer leur influence dans la conception de notre système de vérification. Ainsi, à chaque étape nous rajoutons un ensemble de caractéristiques de mêmes natures afin d'évaluer leur impact sur les performances du système. Pour ce faire, on utilise les cinq premiers utilisateurs du dossier *Task 2* de la base de données de l'étude : 75% de cette base de données (30 signatures originales et fausses) est utilisée pour l'entraînement et les 25%

restante (10 signatures originales et fausses) pour l'évaluation. L'architecture du réseau de neurones utilisée pour cette étude se présente comme suit :

Couche d'entrée : le nombre de neurones dans cette couche équivaut au nombre de caractéristiques en entrées.

Couches cachées : nous fixons le nombre de couches cachées à deux, la fonction d'activation qu'on utilisera sera la fonction *relu* pour l'ensemble des neurones des couches cachées car elle représente la fonction la plus utilisée dans l'apprentissage en profondeur pour ses bons résultats. Pour chaque couche le nombre de neurones sera déterminé par la moyenne de la couche d'entrée et de sortie qui est un bon compromis, soit le nombre de neurone sur la couche d'entrée additionné au nombre de neurone sur la couche de sortie le tout divisé par deux.

Couche de sortie : le nombre de neurones dans cette couche sera de un (1) qui donnera la probabilité d'une signature authentique plus la sortie est proche de 1, plus on est sûr de l'authenticité grâce à la fonction d'activation *sigmoïde*. La fonction d'optimisation est fixée sur la fonction *adam*. Nous fixons également une probabilité de seuil de **0,5** pour laquelle si on obtient une probabilité en sortie supérieure à ce seuil on considère que cette signature est authentique tandis que pour une probabilité inférieure à ce seuil, on considère la signature en entrée comme étant une falsification.

Autres paramètres : L'entraînement se fera sur 1000 époques.

Pour l'évaluation des performances du système, nous utilisons la **F-mesure**, le **rappel**, la **précision** et l'**AUC**. Nous baserons fondamentalement nos analyses sur la *précision* car cette mesure nous permet de savoir le nombre de signatures falsifiées prédites comme originales. En effet, ce comportement est *inadmissible*. Le vérificateur peut se tromper en supposant qu'une signature originale est fausse. Ce comportement serait admissible en ce sens qu'il peut arriver qu'un propriétaire signe mal en fonction des conditions où la signature a été apposée. Nos tests visent donc une **précision de 1** qui signifie que *notre système ne prédit pas de fausse signature comme vraie*. Nous analyserons aussi le système au regard de l'AUC et de la F-mesure qui témoigne de la performance globale de notre système.

Le Tableau 3.1 présente les résultats obtenus à parti de nos tests. Dans ce tableau, les caractéristiques considérées sont référencées par leur numéro dans le Tableau 2.2. Nous consignons ici uniquement les valeurs minimales de chaque mesure. Nous avons choisi de montrer uniquement les minima car ils représentent le pire des cas de notre système. Voir l'Annexe A du Tableau A.1 au Tableau A.5 pour le détail des résultats sur les cinq utilisateurs de ce test.

TABLEAU 3.1 – Résultats des valeurs minimales des métriques sur l'étude des caractéristiques ("__" signifie qu'on considère les caractéristiques de la ligne précédente)

Caractéristiques considérées	Nombre de neurones par couche cachée	F-mesure	Rappel	Précision	AUC
1, **2**, **3**, **4**, **5**, **6**, **7**	4	0,666	0,666	0,5714	0,6
__, **8**, **9**, **10**, **11**, **12**	7	0,5	0,5	0,5	0,5
__, **13**, **14**	8	0,8888	0,8333	0,8	0,9166
__, **15**, **16**, **17**, **18**	10	0,8888	0,8333	0,8	0,9166
__, **19**, **20**	11	0,8888	0,8333	0,8	0,9166
__, **21**, **22**	12	0,9090	0,8333	1	0,9166
__, **23**, **24**	13	0,9090	0,8333	1	0,9166
__, **25**, **26**	14	0,9090	0,8333	1	0,9166
__, **27**	14	0,9090	0,8333	1	0,9166

Analyse par rapport aux caractéristiques statiques

Lorsqu'on considère les sept premières caractéristiques issues des coordonnées cartésiennes, c'est à dire la première ligne du Tableau 3.1, on remarque une précision minimale de 0,5714 qui ne semble pas satisfaisante comme performance à cause du fait qu'il y a beaucoup d'éléments Faux Positifs ce qui n'avantage pas vraiment notre système. Ensuite on rajoute les cinq autres caractéristiques statiques issues des coordonnées polaires au niveau de la deuxième ligne, on remarque que en plus de la précision , on a une baisse également au niveau des autres métriques. Cela est provoqué à cause du fait que ces caractéristiques apportent du bruit vu qu'elles ont été recalculées à partir des coordonnées cartésiennes qui n'ont pas donné des résultats satisfaisants au préalable. Mais en observant les performances sur tous les utilisateurs, on remarque que les utilisateurs ayant de simples signatures ont des précisions allant jusqu'à 1 tandis que pour ceux dont la signature est très complexe la précision est très basse. Ainsi, la complexité d'une signature provoque une forte variation sur la distribution des coordonnées d'un échantillon à un autre ce qui n'avantage pas notre système d'apprentissage. A partir de là on peut conclure que des performances élevées sont obtenues principalement pour de simples signatures lorsqu'on considère uniquement des caractéristiques statiques.

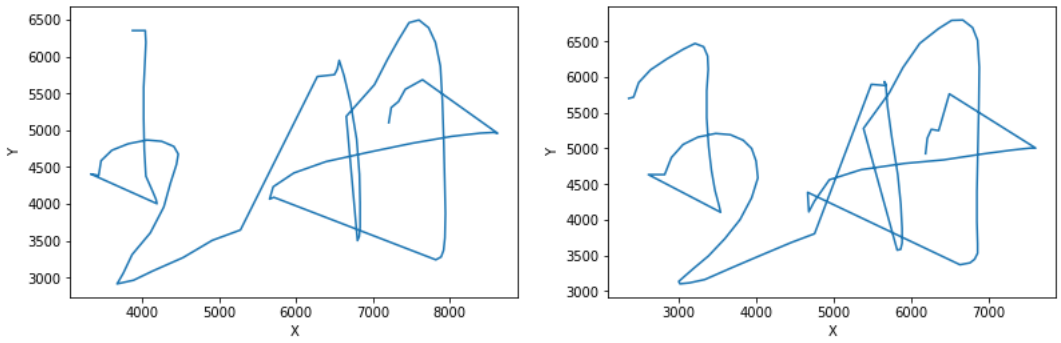


FIGURE 3.2 – Deux échantillons de la signature authentique de l'utilisateur 1

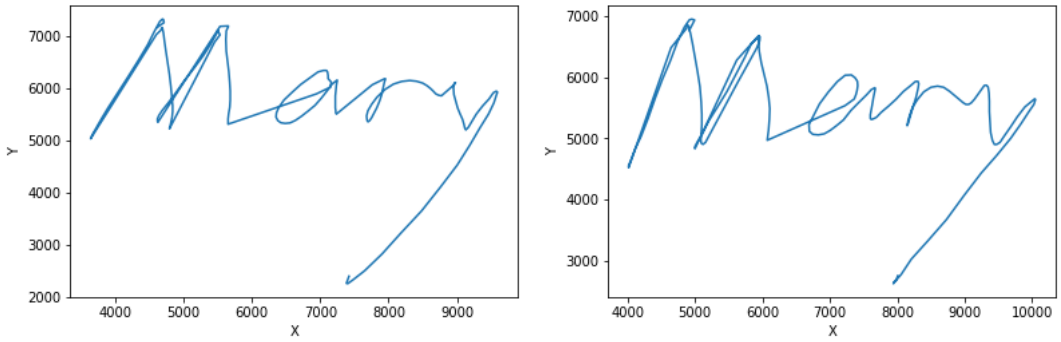


FIGURE 3.3 – Deux échantillons de la signature authentique de l'utilisateur 4

Suivant la logique de notre conclusion, lorsqu'on analyse deux échantillons de l'utilisateur 4, il a plus de chance d'avoir un modèle propre à lui très performant vu la simplicité de sa signature ce qui à été le cas. Par contre lorsqu'on analyse deux échantillons de signature de l'utilisateur 1, on remarque une signature beaucoup plus complexe provoquant ainsi une grande variation de la distribution des coordonnées des points représentants la signature d'un échantillon à un autre, ce qui ne rend pas la tâche facile et ne favorise pas une bonne performance pour le modèle de cet utilisateur.

Analyse par rapport aux caractéristiques dynamiques

Lorsqu'on commence par rajouter nos premières caractéristiques dynamiques à l'ensemble des caractéristiques statiques précédemment citées, c'est à dire à partir de la troisième ligne du Tableau 3.1, on remarque une forte augmentation de la précision minimale qui passe à 0,8 et des performances générales assez élevées, on remarque au fur et à mesure qu'on ajoute les caractéristiques dynamiques, la précision passe de 0,8 à 1. On en conclut que les caractéristiques dynamiques sont très déterminantes pour l'obtention de très bonnes performances, elles auront beaucoup d'impacts dans la distinction d'une signature authentique et falsifiée, de ce fait elles doivent être conservées comme étant des caractéristiques importantes pour la réalisation d'un système de vérification très performant.

Autre Analyse générales

En analysant de manière générale, on remarque que certaines caractéristiques lorsqu'elles sont rajoutées n'apportent aucun changement sur les performances comparées aux performances obtenues lorsqu'elles n'y étaient pas. Ainsi on n'arrive pas à réaliser l'importance de ces caractéristiques. Il serait donc judicieux de mener des tests en utilisant uniquement ces caractéristiques afin d'évaluer leurs apports sur les performances. Dans un premier temps, l'ajout des caractéristiques 15, 16, 17, 18 (moyenne des vitesses linéaires, écart-type des vitesses linéaires, moyenne des accélérations, écart-type des accélérations) au niveau de la troisième ligne du Tableau 3.1, n'apporte aucune amélioration par rapport à la ligne précédente. Ensuite lorsqu'on analyse la quatrième ligne, il n'y a également pas de changement mais quand on regarde l'étude effectuée à ce niveau en détails c'est à dire sur les cinq utilisateurs, on remarque une amélioration chez certains utilisateurs tandis que pour d'autres on constate que du bruit s'est rajouté, nous n'effectuerons pas donc de tests à ce niveau puisqu'on remarque déjà leur importance dans certains cas. Enfin lorsqu'on obtient une précision minimale de 1 au niveau de la ligne 5 de notre tableau, l'ajout des caractéristiques de 23 à 27 (moyenne des azimuts, écart-type des azimuts, moyenne des altitudes, écart-type des altitudes, transition) au niveau des lignes qui suivent n'apporte pas de changement sur les performances. Nous ferons également une étude en considérant uniquement ces caractéristiques.

Pour résumer nous ferons d'abord une étude en considérant uniquement les caractéristiques 15, 16, 17 et 18, ensuite nous ferons une autre étude en considérant les caractéristiques 23, 24, 25, 26 et 27. Cependant nous consignons uniquement la valeur minimale de chaque mesure dans le Tableau 3.3. Les résultats des cinq utilisateurs de ce test sont disponibles dans l'Annexe A dans les Tableau A.6 et A.7. Les numéros de ligne seront utilisés pour désigner chaque caractéristique (cf Tableau 2.2).

TABLEAU 3.3 – Résultats des valeurs minimales des métriques sur l'étude de certaines caractéristiques

Caractéristiques considérées	**Nombre de neurones par couche cachée**	**F-mesure**	**Rappel**	**Précision**	**AUC**
15, 16, 17, 18	3	0,7272	0,6666	0,5714	0,7083
23, 24, 25, 26, 27	3	0,75	0,75	0,75	0,75

En analysant le Tableau 3.3, on constate que les caractéristiques 15, 16, 17 et 18 donnent une précision de 0,5714. Cette valeur n'est tout de même pas très rassurante car elle tourne beaucoup autour de la moyenne mais sur les autres métriques nous avons quand même des

valeurs satisfaisantes. Même si ces caractéristiques nous donnent des performances moyennes elles peuvent s'avérer importantes pour la suite.

Cependant les caractéristiques 23, 24, 25, 26 et 27 quant à elles donnent une précision minimale de 0,75 qui est quand même une valeur assez satisfaisante qui prouve l'importance de ces caractéristiques.

Pour la suite on décide de conserver toutes nos vingt-sept(27) caractéristiques car elles se sont avérées importantes dans l'évaluation des performances.

3.3 Validation de l'architecture neuronale optimale

Maintenant que nous avions évalué l'impact de nos caractéristiques sur notre système final et que nous avions retenu les caractéristiques à utiliser, passons à présent à la validation de l'architecture optimale, c'est à dire l'architecture finale du réseau de neurones ainsi que les paramètres optimaux de ce dernier qu'on retiendra pour avoir un système performant et efficace pour n'importe quel utilisateur. Pour ce faire, on utilise les dix premiers utilisateurs du dossier *Task 2* de la base de données de l'étude : 75% de cette base de données (30 signatures originales et fausses) est utilisée pour l'entraînement et les 25% restante (10 signatures originales et fausses) pour l'évaluation. Nous décidons de mener l'étude suivant deux architectures principales, une architecture constituée de deux couches cachées avec quatorze (14) neurones sur chaque couche cachée et une architecture constituée de trois couches cachées avec quatorze (14) neurones sur chaque couche cachée. Pour chaque architecture, on fera varier les paramètres tout au long des expériences. L'architecture du réseau de neurones utilisé pour cette étude se présente comme suit :

Couche d'entrée : le nombre de neurones dans cette couche équivaut au nombre de caractéristiques en entrées soit vingt-sept(27) neurones car dans l'étude sur la validation de nos caractéristiques, on a décidé de garder toutes les caractéristiques décrites dans le Tableau 2.2 qui sont au nombre de vingt-sept(27).

Couches cachées : nous ferons varier le nombre de couches cachées entre deux et trois, la fonction d'activation qu'on utilisera sera la fonction *relu* pour l'ensemble des neurones des couches cachées car elle représente la fonction la plus utilisée dans l'apprentissage en profondeur pour ses bons résultats. Pour chaque couche le nombre de neurones sera déterminé par la moyenne des neurones de la couche d'entrée et de sortie qui est un bon compromis, soit le nombre de neurone sur la couche d'entrée additionné au nombre de neurone sur la couche de sortie le tout divisé par deux. Vu qu'on aura vingt-sept(27) neurones sur la couche d'entrée et un(1) neurone sur la couche de sortie en suivant notre formule on a alors un total de quatorze(14) neurones sur chaque couche cachée.

Couche de sortie : le nombre de neurones dans cette couche sera de un (1) qui nous donnera la probabilité d'être une signature authentique plus la sortie est proche de 1 plus on est sûr de l'authenticité grâce à notre fonction d'activation *sigmoïde*, on a une fonction d'optimisation qui variera entre *adam*, *rmsprop* et *sgd*. Nous décidons donc de fixer une probabilité de seuil de **0,5** pour laquelle si on obtient une probabilité en sortie supérieure à ce seuil on considère que cette signature est authentique tandis que pour une probabilité inférieure à ce seuil on considère la signature en entrée comme étant une falsification.

Autres paramètres : L'entraînement se fera sur plusieurs nombres d'époques varié.

Pour l'évaluation des performances du système, nous utilisons la **F-mesure**, le **rappel**, la **précision** et l'**AUC**. Nous baserons fondamentalement nos analyses sur la *précision* car cette mesure nous permet de savoir le nombre de signatures falsifiées prédites comme originales. Nous restons dans la logique de l'analyse précédente.

Le Tableau 3.4 présente les résultats obtenus à parti de nos tests. Nous consignons ici uniquement les valeurs minimales de chaque mesure. Nous avons choisi de montrer uniquement les minima car ils représentent le pire des cas de notre système. Les résultats des dix utilisateurs de ce test sont disponibles dans l'Annexe A du Tableau A.8 au Tableau A.13.

TABLEAU 3.4 – Résultats des valeurs minimales des métriques sur l'étude de l'architecture neuronale

Nombre de couches cachées	Taille du batch	Nombre d'époques	Optimiseur	F-mesure	Rappel	Précision	AUC
2	3, 5	1000, 2000, 5000	adam	0,75	0,6	1	0,8
			rmsprop	0,75	0,6	1	0,8
			sgd	0,8888	0,8	1	0,9
3	3	1000, 2000	adam	0,75	0,6	1	0,8
			rmsprop	0,75	0,6	1	0,8
			sgd	0, 6153	0, 8	0, 5	0, 5
		5000	adam	0,75	0,6	1	0,8
			rmsprop	0,75	0,6	1	0,8
			sgd	0, 8888	0, 8	1	0, 9
	5	1000, 2000	adam	0,75	0,6	1	0,8
			rmsprop	0,75	0,6	1	0,8
			sgd	0	0	0	0
		5000	adam	0,75	0,6	1	0,8
			rmsprop	0,75	0,6	1	0,8
			sgd	0,8888	0,8	1	0,9

En analysant le Tableau 3.4, on remarque que, quelque soit le nombre de couches cachées, la taille du batch ou le nombre d'époques, les optimiseurs *adam* et *rmsprop* convergent beaucoup plus vite vers leurs résultats optimaux, même si dans l'ensemble ils ont permis d'atteindre une précision de 1 pour l'ensemble des dix(10) utilisateurs, de manière générale, ils n'ont pas donné les meilleures performances. Cependant lorsqu'on considère l'optimiseur *sgd*, on remarque que dans une architecture à deux couches cachées, quelque soit la taille du batch ou le nombre d'époques, il converge très vite vers son résultat optimal mais lorsqu'on vient sur l'architecture à trois couches cachées, elle atteint des résultats satisfaisants qu'à partir de 5000 époques, mais de manière générale on constate également que l'optimiseur *sgd* a également permis d'avoir une précision minimale de 1 mais sur les autres métriques il fait mieux que les deux autres optimiseurs.

En conclusion, pour notre architecture finale, on retient pour notre architecture optimale l'optimiseur *sgd*, pour le nombre de couches cachées on retiendra deux(2) couches car avec cette architecture il converge mieux. Pour la taille du batch on choisit 3 on aurait également pu choisir 5 mais ce qu'on remarque est que dans l'architecture à trois couches lorsqu'on a utilisé 5 comme taille de batch, on a eu des valeurs de métrique minimales qui sont de 0 témoignant ainsi de la pire mauvaise performance qu'on puisse avoir donc par précaution on retient un batch de 3. Quant au nombre d'époques, on a préféré choisir un nombre d'époques de 5000, mais vu les paramètres qu'on a choisi jusque-là on aurait pu garder 1000 ou 2000 comme nombre d'époques mais lorsqu'on se réfère à une architecture à trois (3) couches cachées, les meilleurs résultats de l'optimiseur *sgd* n'ont été atteints que pour un nombre d'époques de 5000. On suppose donc que l'optimiseur *sgd* à besoin d'un nombre d'époques assez élevé pour mieux converger, par précaution on retient donc 5000 comme nombre d'époques.

L'architecture finale retenue est la suivante :

- **Couche d'entrée** : nombre de neurones = *27*
- **Couche cachées** : nombre de couches = *2*, nombre de neurones par couche = *14*, fonction d'activation = *relu*
- **Couche de sortie** : nombre de neurone = *1*, fonction d'optimisation = *sgd*
- **Autre paramètres** : taille du batch = *3*, nombre d'époques = *5000*

Nous n'avons cependant pas arrêté notre étude sur les deux architectures décrites plus haut. Nous avons effectué des études également sur d'autres architectures, même si elles nous ne les décrirons pas de manière exhaustive car elles se sont révélées peu satisfaisantes et dans d'autres cas pas du tout intéressantes. En outre nous pouvons citer :

- **L'étude sur une couche cachée** : dans cette architecture on remarque des performances très basses à certains niveaux et des taux d'erreurs d'entraînements très grands par rapport aux deux architectures principales considérées. Les résultats avec cette architecture ont été très peu satisfaisants par rapport à ce qu'on a déjà obtenu.

- **L'étude sur quatre couche cachées** : dans cette architecture le problème principal est que la convergence n'est pas vite atteinte, on pourrait avoir des résultats proches des architectures de deux et trois couches cachées mais sur un nombre d'époques très grand (plus de 5000) ce qui ne nous favorise pas réellement en terme de ressources et de durée d'exécution ainsi on a pu généraliser sur le fait que plus nous augmenterons le nombre de couches cachées plus les ressources du matériel seront utilisées également plus on aura une très grande durée d'exécution pour au final avoir un même résultat qu'une architecture à deux ou trois couches qui consomme moins de ressources physiques et s'exécute sur une durée de temps inférieure ce qui ne serait pas du tout optimal.

- **Variation du nombre de neurones à l'intérieur des couches cachées** : on a remarqué que lorsque le nombre de neurones était soit largement inférieur ou largement supérieur à la moyenne des neurones en entrée et en sortie, on a des résultats très insuffisants sur l'ensemble des utilisateurs, on a jusqu'à 0 parfois pour des évaluateurs de performance. On a donc eu la bonne idée de trouver un bon compromis pour un nombre de neurones dans les couches cachées qui tourne autour de la moyenne des neurones en entrée et en sortie.

3.4 Comparaison à d'autres méthodes et à l'état de l'art

3.4.1 Comparaison de la méthode utilisée avec d'autres méthodes

A présent que nous avons réalisé notre étude avec la méthode des réseaux de neurones essayons de refaire une étude brève avec d'autres méthodes afin de faire une comparaison et d'y relever d'éventuelles faiblesses ou d'éventuels points forts par rapport à notre méthode. Pour ce travail on effectue l'étude sur l'ensemble des quarante(40) utilisateurs du dossier "Task 2" de notre base de données sachant que 75% des signatures ont été utilisées pour l'entraînement soit trente(30) et 25% pour le test soit dix(10) pour le test par utilisateur ce qui nous fait au total pour tous les utilisateurs 1200 signatures pour l'entraînement et 400 pour le test, nous utiliserons notre architecture principale retenue lors de l'étude sur l'architecture du réseau de neurones optimal dans un premier temps ensuite nous utiliserons la méthode de la machine à vecteur de support et la méthode des K voisins les plus proches afin d'établir une comparaison à partir des indices de performances que sont le FAR et le FRR. Cependant nous effectuons dans un premier temps une étude sur les dix(10) premiers utilisateurs dont nous consignons uniquement la valeur minimale de chaque mesure dans le Tableau 3.5. Les résultats des dix(10)

premiers utilisateurs sont disponibles dans l'Annexe A du Tableau A.14 au Tableau A.16.

TABLEAU 3.5 – Tableau comparatif des performances entre les méthodes de réseau de neurones, de machine à vecteur de support et de K-voisins les plus proches

Méthode	**F-mesure**	**Rappel**	**Précision**	**AUC**
Réseau de neurones	0, 8888	0,8	1	0,9
Machine à vecteur de support	0, 9091	0, 8333	0, 8750	0, 8333
K-voisins les plus proches	0, 8888	0, 8	0, 8	0, 8750

- L'utilisation de l'architecture optimale de notre **réseau de neurones** nous a permis d'avoir un FAR de 1,25% et un FRR de 0,75% sur les quarante(40) utilisateurs. Avec une précision de 1 pour l'ensemble des dix(10) premiers utilisateurs, soit une précision minimale de 1.
- L'utilisation de la méthode de la **machine à vecteur de support** avec comme paramètre de base un hyperplan linéaire a permis d'avoir un FAR de 1,5% et un FRR de 0,25% sur les quarante(40) utilisateurs. Avec une précision minimale de 0,8750 pour l'ensemble des dix(10) premiers utilisateurs.
- L'utilisation de la méthode des **K-voisins les plus proches avec k = 3** a permis d'avoir un FAR de 3,5% et un FRR de 1% sur les quarante(40) utilisateurs. Avec une précision minimale de 0,8 pour l'ensemble des dix(10) premiers utilisateurs.

On remarque que la méthode des K-voisins les plus proches n'est pas très adaptée pour notre système de vérification à cause de ses performances en dessous des deux autres c'est à dire celles des réseaux de neurones et de la machine à vecteur de support. Cependant lorsqu'on considère ces dernières on remarque que pour les réseaux de neurones on a un FAR de 1,25% qui est plus bas par rapport au FAR de la machine à vecteur de support qui est de 1,5%. A partir de là on peut déjà voir l'efficacité du réseau de neurones par rapport à notre système avec son taux de FAR plus bas que celui de la machine à vecteur de support, elle aura tendance à moins générer des éléments Faux Positifs c'est à dire qu'elle ne considérera pas une falsification de signature comme étant une authentique beaucoup moins souvent que la méthode de la machine à vecteur de support. Lorsqu'on considère le FRR on remarque que la méthode de la machine à vecteur de support a une valeur de 0,25% qui est plus basse que celle du réseau de neurones qui est de 0,75%. On en conclut alors que les réseaux de neurones sont plus efficaces pour détecter une signature falsifiée comparés à la machine à vecteur de support qui est particulièrement plus efficace pour détecter une signature authentique.

3.4.2 Comparaison de notre étude à une autre utilisant la même méthode

Au cours de notre travail sur l'état de l'art nous avons pu constater un travail qui se rapprochait du nôtre. Ce travail présente un système de prévention des falsifications de signatures dynamiques créé à l'aide des réseaux de neurones, entraîné également grâce à l'algorithme de rétro-propagation [25]; de plus la base de données considérée pour notre étude a été également utilisée dans ce travail. Ainsi les meilleurs résultats obtenus dans ce travail de l'état de l'art sur les utilisateurs du dossier Task 2 ont permis de classifier correctement vingt-cinq(25) utilisateurs sur l'ensemble des quarante(40) utilisateurs (l'ensemble les utilisateurs qui ont un nombre de faux positif et un nombre de faux négatif nuls pendant la phase de test) avec un FAR de 17% et un FRR de 1,5% alors que notre étude nous a permis d'obtenir un meilleur résultat qui est de trente-deux(32) utilisateurs correctement classifiés sur quarante(40) utilisateurs avec un FAR de 1,25% et un FRR de 0,75%.

TABLEAU 3.6 – Tableau comparatif de notre étude à celle de Shalini B. et al [25]

	Notre étude	**Etude de Shalini B. et al**
FAR	1,25%	17%
FRR	0,75%	1,5%
Nombre d'utilisateurs bien classifiés	32/40	25/40

3.4.3 Comparaison de notre étude à une étude de vérification statique

Dans l'état de l'art nous avons eu à aborder certains travaux effectués grâce à la vérification statique qui donne des résultats intéressants. Lorsqu'on considère les travaux effectués dans les papiers [11] et [12] abordés dans l'état de l'art et qui utilisent également la méthode des réseaux de neurones on a des taux de classifications réussies de 91,21% et de 82,66% respectivement. Alors que pour notre système dynamique étudié et utilisant également les réseaux de neurones lorsqu'on déduit le taux de FAR et de FRR qui sont respectivement de 1,25% et 0,75% de l'ensemble de nos données de test, il nous reste donc 98% de signatures bien classifiées sur l'ensemble des signatures de test qui est bien meilleur que les résultats obtenus pour une vérification statique malgré l'utilisation d'une même méthode dans les deux cas. De même le système de vérification statique présenté dans le papier [15] en utilisant la méthode de la machine à vecteur de support à permis d'obtenir sur un ensemble de 1320 signatures un FAR et un FRR qui sont respectivement de 2% et 11%. On remarque bien que nos résultats pour notre étude sont bien meilleurs que ces derniers. On peut en déduire que la vérification dynamique de part ses caractéristiques dynamiques (pression, azimut, temps etc...) nous donne des informations très indispensables et beaucoup plus fiables pour détecter une signature falsifiée et une signature authentique, cependant elle doit être beaucoup plus considérée par rapport à

un système de vérification statique.

3.5 Aperçu du système final conçu

Dans cette section nous présentons juste un aperçu du système web mis en place à partir des résultats optimaux de notre étude qui nous permet d'entraîner un modèle de classification de signature (authentique ou falsification) de manière automatique juste en lui donnant les informations et les ressources nécessaires, elle nous permet également lorsqu'on lui soumet une signature de pouvoir distinguer à partir du modèle entraîné si la signature est une falsification ou une authentique.

Lorsqu'on démarre notre serveur et qu'on tente d'accéder à l'application, on est redirigé vers la page suivante sur laquelle on distingue deux champs : un champ pour renseigner le nom à qui appartiendra le modèle qui va être entraîné et un autre champ pour soumettre l'ensemble des signatures (falsifiées ou authentiques) de ce dernier dans un dossier compressé (zip) tout en respectant le format de présentation des fichiers contenus dans le dossier compressé.

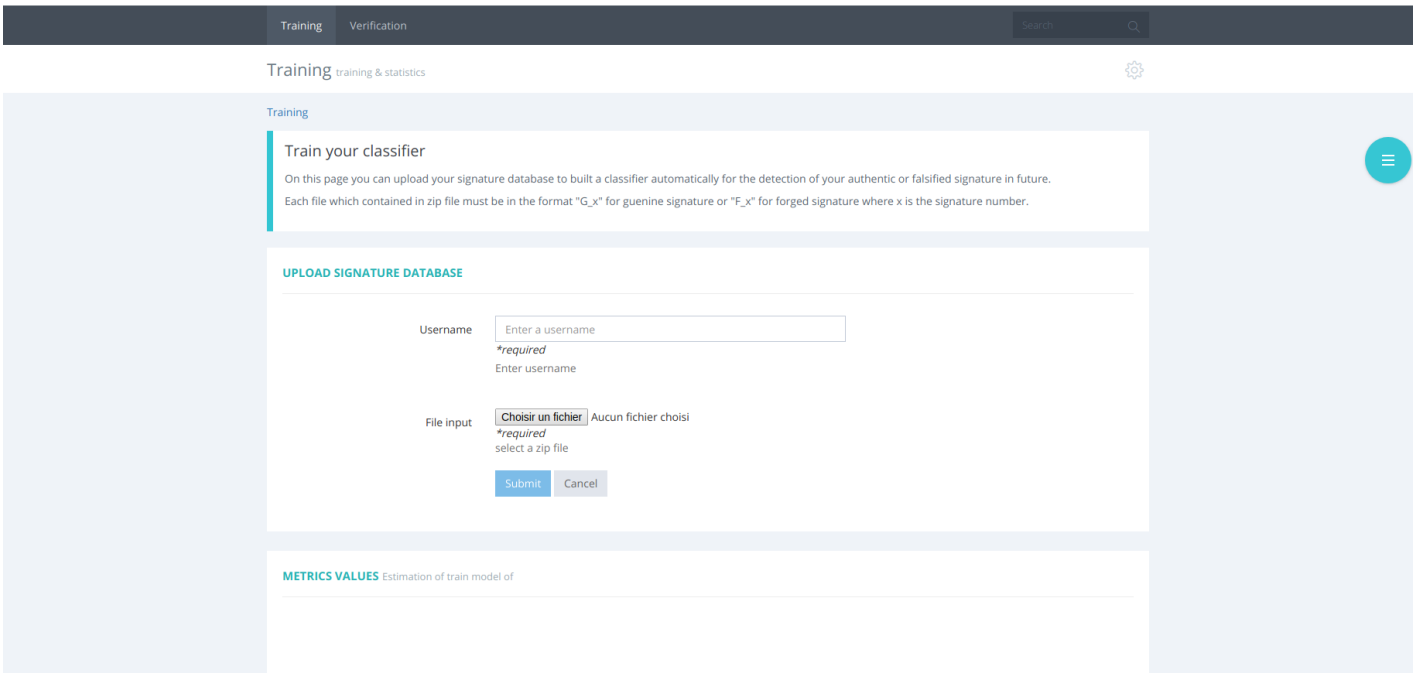


FIGURE 3.4 – Page de formulaire pour un entraînement

Ensuite lorsqu'on lui soumet les informations nécessaires on remarque l'apparition d'une barre de chargement qui signifie que les informations ont été enregistrées avec succès et que le modèle est en cours d'entraînement.

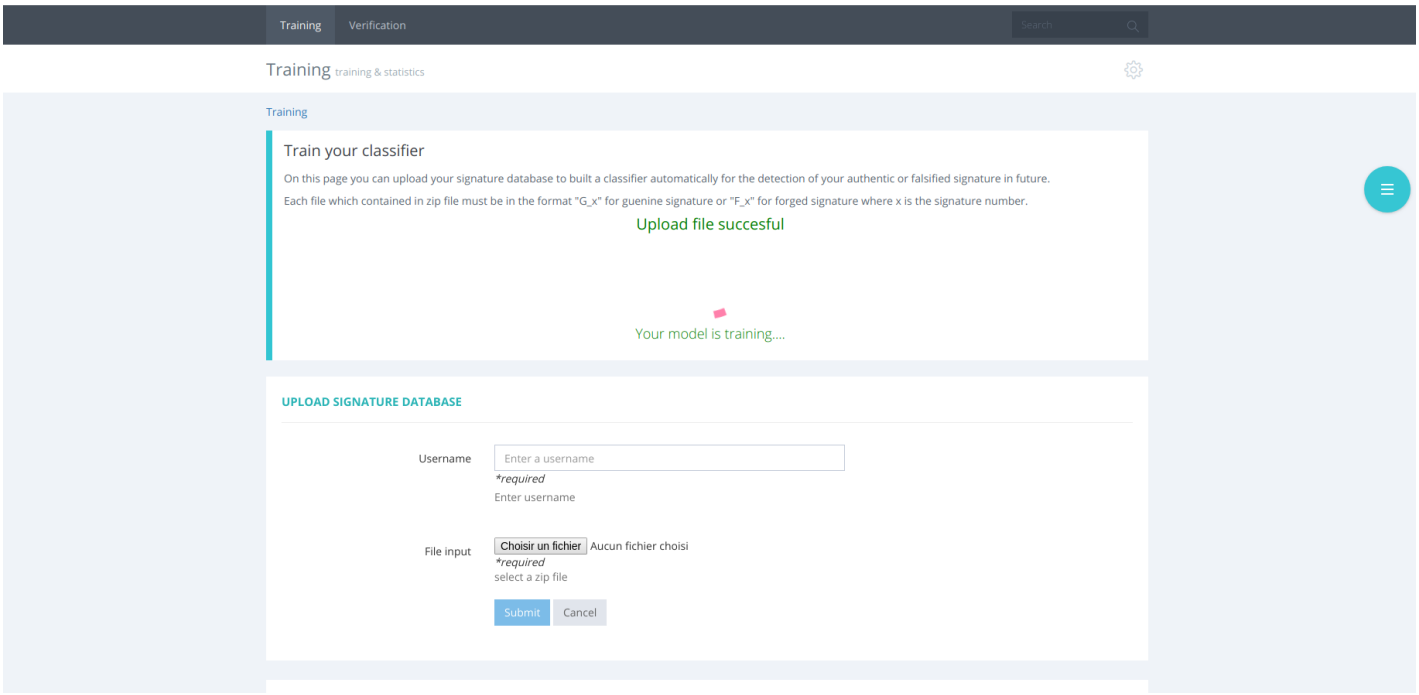


FIGURE 3.5 – Page de soumission réussie et d'entraînement en cours

Une fois l'entraînement de notre modèle terminé pour l'utilisateur "Franck", une notification apparaît pour nous le signaler et on a l'ensemble des valeurs des métriques qui nous permet d'évaluer et de conclure si le modèle entraîné est fiable pour une utilisation future.

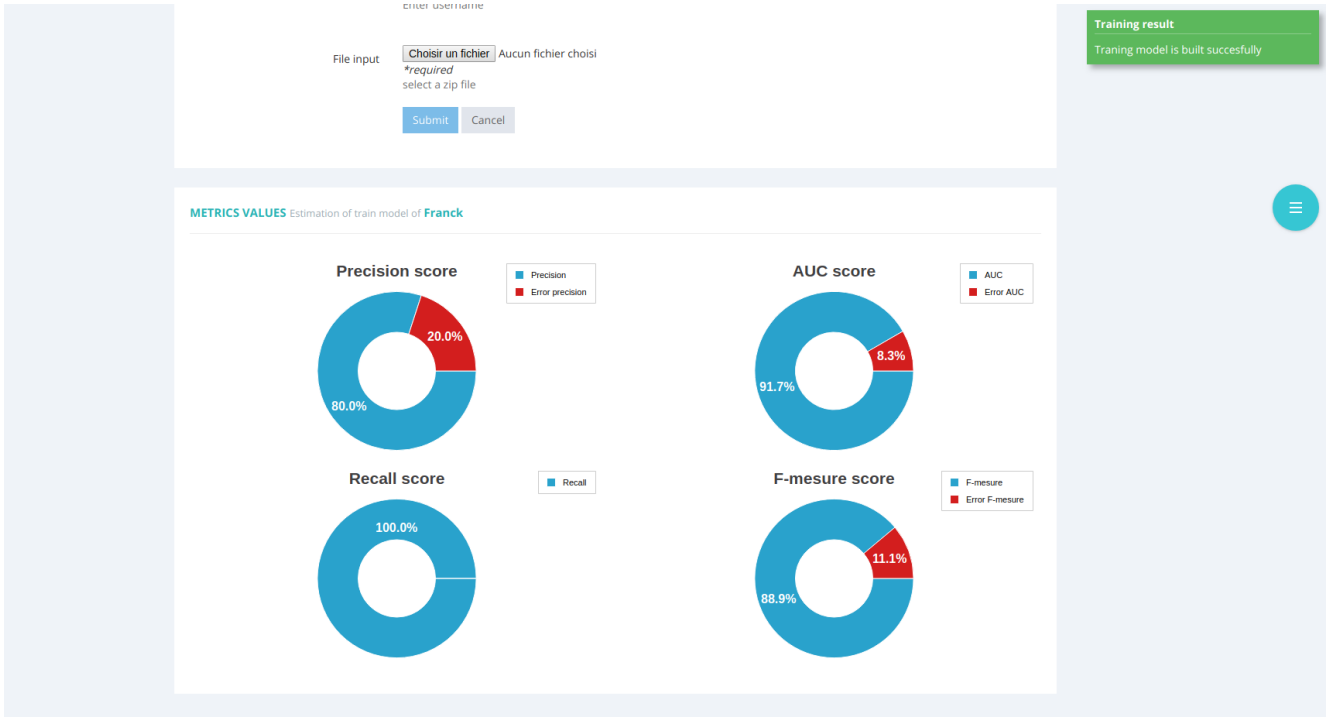


FIGURE 3.6 – Page d'entraînement réussi et d'évaluation du modèle

A présent jetons un coup d'œil sur la page qui va nous permettre de pouvoir prédire si une signature est falsifiée ou pas. Comme on le constate bien on a un formulaire presque identique à celui de l'entraînement c'est à dire un champ pour renseigner le modèle de l'utilisateur avec lequel nous effectuons la classification et un champ de soumission de fichier à la différence que ce champ ne prendra qu'un fichier de type texte (txt) qui est un singleton de signature afin d'effectuer la classification.

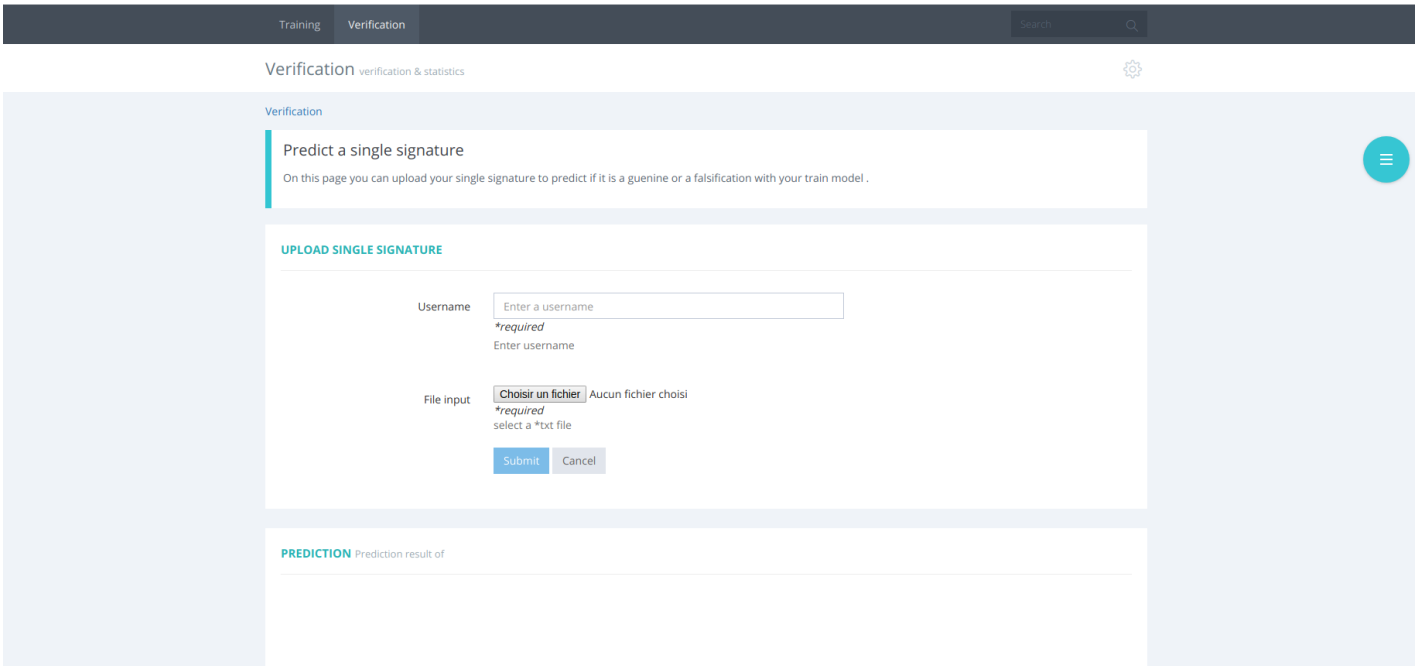


FIGURE 3.7 – Page de prédiction d'une signature

Une fois la signature soumise, le système procède ensuite à une vérification à partir du modèle de l'utilisateur renseigné et nous ressort une valeur de probabilité que la signature soit authentique ainsi qu'une représentation en fonction du temps de la distribution des valeurs de l'altitude, la pression, l'azimut également une représentation de la signature soumise.

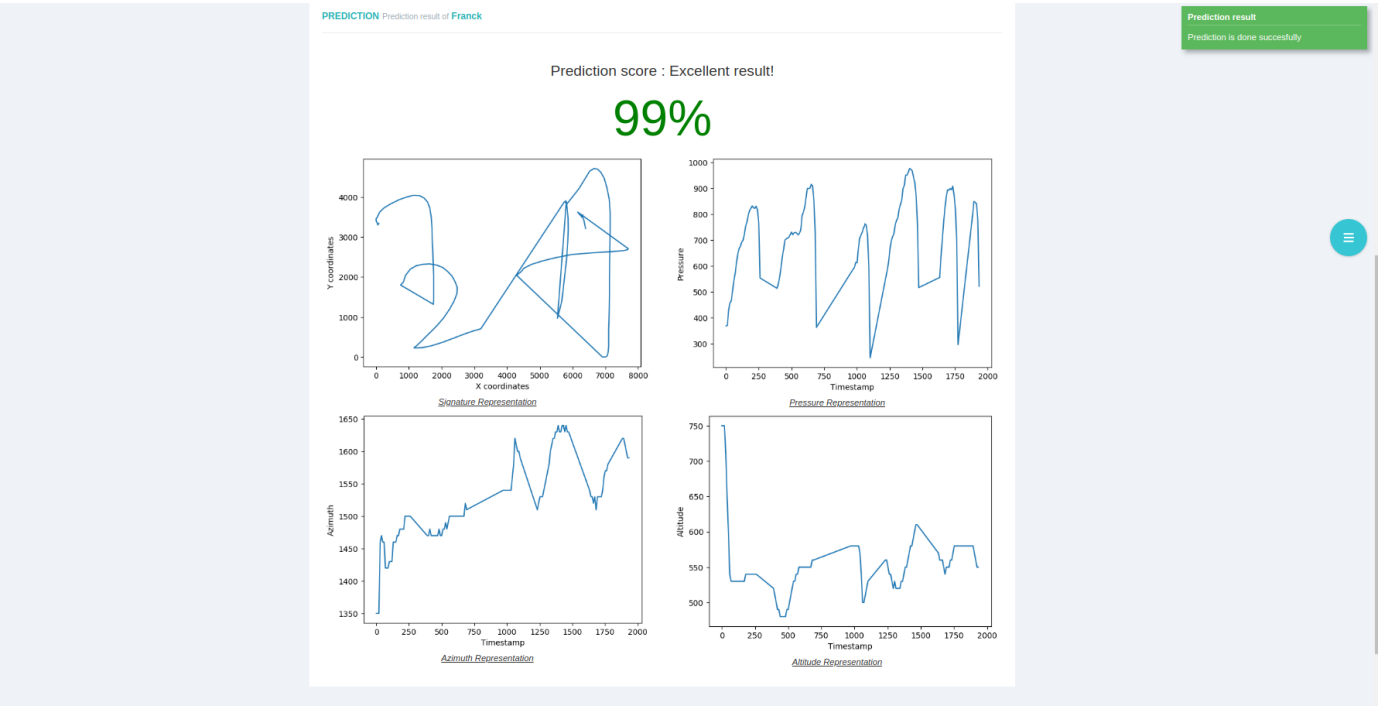


FIGURE 3.8 – Page de prédiction réussie et évaluation de la signature

3.6 Discussion générale

L'étude sur l'impact des caractéristiques retenues nous a permis de constater que lorsqu'on ne considère que les caractéristiques statiques, il a été constaté que des performances élevées sont obtenues principalement pour les utilisateurs dont les signatures sont simples et ne sont pas très complexes à exécuter. Ces caractéristiques utilisées sont donc efficaces pour des signatures authentiques très simples et dont la forme varie très peu d'un échantillon à l'autre mais restent insuffisantes pour des signatures beaucoup plus complexes. Dans certains cas particuliers, il est possible d'amener l'utilisateur authentique à refaire de nouvelles signatures afin de disposer d'une base de données solide pour les différentes variations possibles de ses signatures et ainsi d'obtenir un système de vérification très efficace pour cet utilisateur.

Par ailleurs nous remarquons l'amélioration des performances lorsque nous ajoutons les caractéristiques dynamiques. Nous pouvons en conclure que les caractéristiques dynamiques sont très importantes pour l'obtention de très bonnes performances. Elles sont déterminantes pour la distinction d'une signature authentique et falsifiée car elles sont uniques à chaque être humain. Elles doivent donc être conservées comme caractéristiques importantes pour la réalisation d'un système de vérification performant. L'importance de nos caractéristiques ayant été prouvées, elles ont été utilisées pour réaliser plusieurs architectures de réseau de neurones. On remarque que l'architecture à deux couches cachées est beaucoup plus stable car elle permet de converger très vite cependant dans le choix de l'optimiseur, la fonction *sgd* même si on a remarqué dans certains cas qu'elle converge moins vite nous a permis d'obtenir les meilleurs

résultats possibles. L'utilisation d'une taille de batch de 3 a été préférée à l'utilisation d'une taille de batch 5 car on a remarqué avec cette dernière de mauvaises performances à certains niveaux. Pour assurer une meilleure convergence de l'optimiseur *sgd* on a décidé de faire les entraînements sur cinq-mille(5000) époques. On a essayé de se comparer aux autre travaux, on constate que en général, notre système de vérification dynamique donne de meilleurs résultats comparé aux systèmes de vérification statique d'où leur efficacité. Le système conçu tout au long de notre travail, à été intégré dans une application web et les essais menés à partir de notre application ont été satisfaisants.

Conclusion

Tout au long de ce chapitre nous avions pu mesurer nos performances à travers les métriques tels que la précision, l'AUC, la F-mesure et le rappel et nous avions constaté l'importance de toutes nos caractéristiques abordées dans le chapitre 2. De ce fait nous avions utilisé ces caractéristiques pour dégager notre architecture optimale après des études sur plusieurs architectures ce qui nous a permis de construire notre petit système de vérification. Également on a pu dégager les faiblesses d'utiliser un réseau de neurones par rapport à la machine à vecteur de support, c'est à dire comparé à cette dernière, le réseau de neurones est plus efficace pour détecter une signature falsifiée alors que la machine à vecteur de support est plus efficace pour détecter une signature authentique. On a aussi remarqué que notre système atteignait des performances plus élevées que d'autres travaux effectués utilisant également les réseaux de neurones et de même qu'en général, le système de vérification dynamique est beaucoup plus efficace que le système de vérification statique.

Conclusion et perspectives

Tout au long de notre travail, nous avons mené une étude de performance sur un système dynamique de vérification de la signature manuscrite grâce à la méthode des réseaux de neurones, cette étude a été divisée essentiellement en deux parties tout d'abord nous avons mesuré l'impact de chaque caractéristique considérée sur les performances du système ensuite une fois que nous avons réalisé l'importance de chacune de nos caractéristiques, nous avons utilisé ces caractéristiques pour mener une étude sur plusieurs architectures de réseau de neurones afin d'en dégager l'architecture optimale et ainsi d'avoir les meilleurs résultats possibles. Grâce à notre étude nous avons obtenu sur la base de données utilisée (SVC 2004) un FAR de 1,25% et un FRR de 0,75% sur un total de 400 signatures pour 40 utilisateurs différents. Ces résultats obtenus nous ont permis de constater que nous avons des résultats meilleurs que d'autres travaux ainsi nous avons implémenté un système de vérification de signature dynamique afin de mettre en application les résultats de nos travaux d'étude.

En guise de perspective, nous prévoyons dans un premier temps, améliorer nos résultats par rapport à l'étude menée en faisant une analyse encore plus profonde des caractéristiques qu'on pourrait rajouter ou supprimer afin d'avoir moins de bruits dans nos caractéristiques considérées et ainsi d'impacter de manière positive nos performances. Dans le même objectif nous prévoyons également mener une étude sur l'architecture des réseaux de neurones en faisant varier de manière plus exhaustive les paramètres du réseau tels que les fonctions d'activation, le nombre de couches cachées ainsi que le nombre de neurones sur chaque couche cachée, la fonction d'erreur, le nombre d'époques sur lequel faire l'entraînement, la taille du batch etc, afin d'avoir une architecture meilleure et avoir de bonnes performances. Dans un second temps nous prévoyons améliorer notre système de vérification en trouvant plusieurs architectures de réseau de neurones donnant des résultats très intéressants car une même architecture ne peut être adaptée à toutes nos données d'entraînements, de ce fait on pourrait avoir un système beaucoup plus fiable qui pourrait être déployé en production.

English version

Introduction

There are several alternatives that are used to identify humans. These techniques include eye recognition, face recognition, fingerprint recognition, but we realize that the use of the signature remains the most used for authentication. The problem arises when a person decides to imitate the person's signature for the purpose of fraud or misrepresentation. Therefore, adequate protection of personal signatures is necessary. In order to assert the authenticity of a signature, the human eye can not be effective because subject to several parameters.

Several handwritten signature recognition and verification systems have been developed nowadays. These systems are divided into two categories : static verification system and dynamic verification system. A static signature verification system uses features extracted from the digitized signature image for verification. The dynamic approach uses an electronic tablet and an electronic pen, the set connected to a computer to extract information about a signature and takes dynamic information such as pressure, time, writing speed, etc for verification purposes.

Most of these systems are based on machine learning algorithms and have many weaknesses that do not allow for effective verification. It is with this in mind that we are attempting to implement a dynamic signature verification system, using a new approach of study and exploiting new dynamic characteristics of the handwritten signature in order to fill the weaknesses of the verification systems. existing and that would ultimately be able to effectively identify whether a signature is falsified or not through a machine learning technique which is in our case the neural network.

3.7 Generality and current state of art

3.7.1 Nature of human signature

According to the American Heritage Dictionary, a signature can be defined as "the name of a person written with his or her own hand ; the act of signing one's name" [1]. A second definition refers to the whole process of signing which implies the way the signature is made is part of the signature itself. This leads to the hypothesis that the characteristics of the process of signing (i.e. pen pressure, velocity, stroke, etc.) are unique to every individual [2].

The first definition states that a signature is a static two-dimensional image not containing any time-related information and the second definition is based on the dynamic features of the process of signing [2].

Signatures can have many forms such as people use their own names as a signature or maybe initials, or using signatures that are hardly related to their names [3] and, as Brault and Plamondon [4] said, some signatures may be quite complex while others are simple and can be forged easily.

Ruth Rostron [5] explains the graphology of a handwritten signature used to analyze and

reveal the personality of an individual. She describes that the variability in the signatures of people may vary in a number of ways, including the individual's mood at the moment.

Gubta [3] points out that if two signatures of the same person were identical, they could be automatically considered as a forgery by tracing. From experts' point of view, successive signatures of the same person will also differ, both globally and locally and may also differ in scale and orientation. Despite these variations, it is said that these signatures will still have the same characteristics, such as the slant angle and the pressure, classifying them as genuine signatures.

It has also been suggested that human experts are very good in identifying forgeries but perhaps not so in verifying genuine signatures. For instance, in a detailed study, Herbst and Liu [6] cite references, state that signature experts managed to reject or classify as no-opinion as high as 25% genuine signatures while accepting no forgeries. Untrained personnel accepted up to 50% forgeries.

3.7.2 Signature verification system : modes of operation

Signature authentication methods have been in use since many years and the research to improve the existing techniques is still in progress. Generally speaking, signature authentication methods can be divided into two categories : Offline methods and Online methods.

Offline signature authentication methods are based on images of the signature captured after the entire signature has been written down. Thus, only the static features of the image such as the shape, size and curves in the image of the signature are visible and hence, these features are the only ones that can be used for verification.

Online signature authentication methods use pen pressure pads to capture each and every point of the signature. Thus, at every point, using a tablet, the x and y co-ordinate of the point, the pressure applied while signing, the timestamp, and the azimuth and altitude angles at which the pen was held are recorded. These dynamic features are further used for verification of signatures.

Most of the works are based on machine learning algorithm like SVM, neural network, decision tree etc for training.

3.7.3 Machine learning : generality

Machine learning teaches computers to do what comes naturally to humans and animals : learn from experience. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases.

Machine learning uses two types of techniques : supervised learning, which trains a model on known input and output data so that it can predict future outputs, and unsupervised

learning, which finds hidden patterns or intrinsic structures in input data.

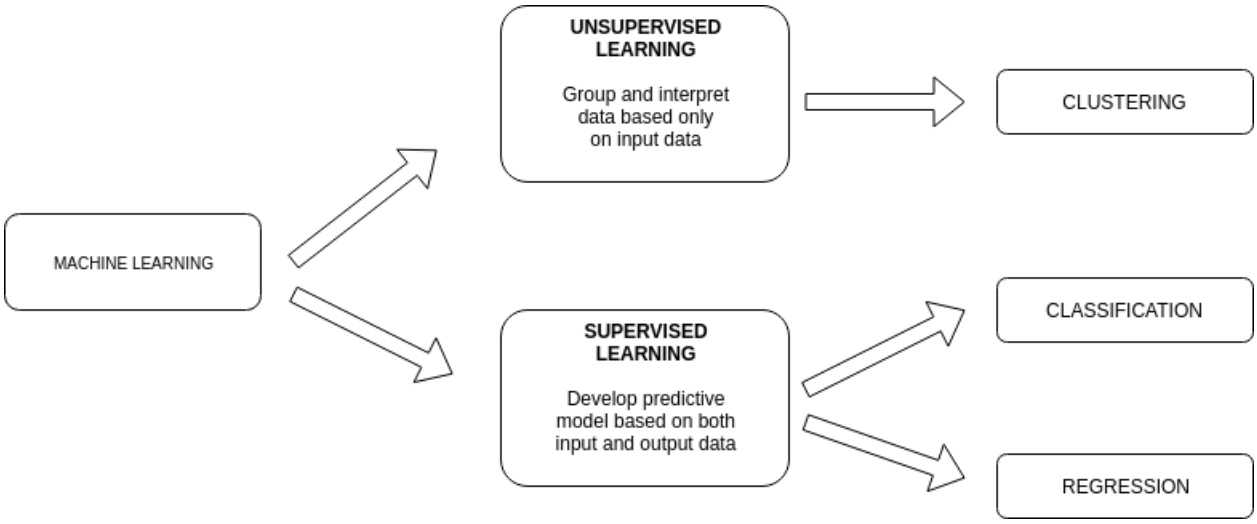


FIGURE 3.9 – Machine Learning Techniques

The aim of supervised machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data. Supervised learning uses classification and regression techniques to develop predictive models.

- **Classification techniques** : predict discrete responses for example, whether an email is genuine or spam, or whether a tumor is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring.
- **Regression techniques** : predict continuous responses for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labeled responses.

- **Clustering** : is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data.

Choosing the right algorithm can seem overwhelming, there are dozens of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning. There is no best method or one size fits all. Finding the right algorithm is partly just trial and error—even highly experienced data scientists can't tell whether an algorithm will work without trying it out. But algorithm selection also depends on the size and type of data you're working with, the insights you want to get from the data, and how those insights will be used.

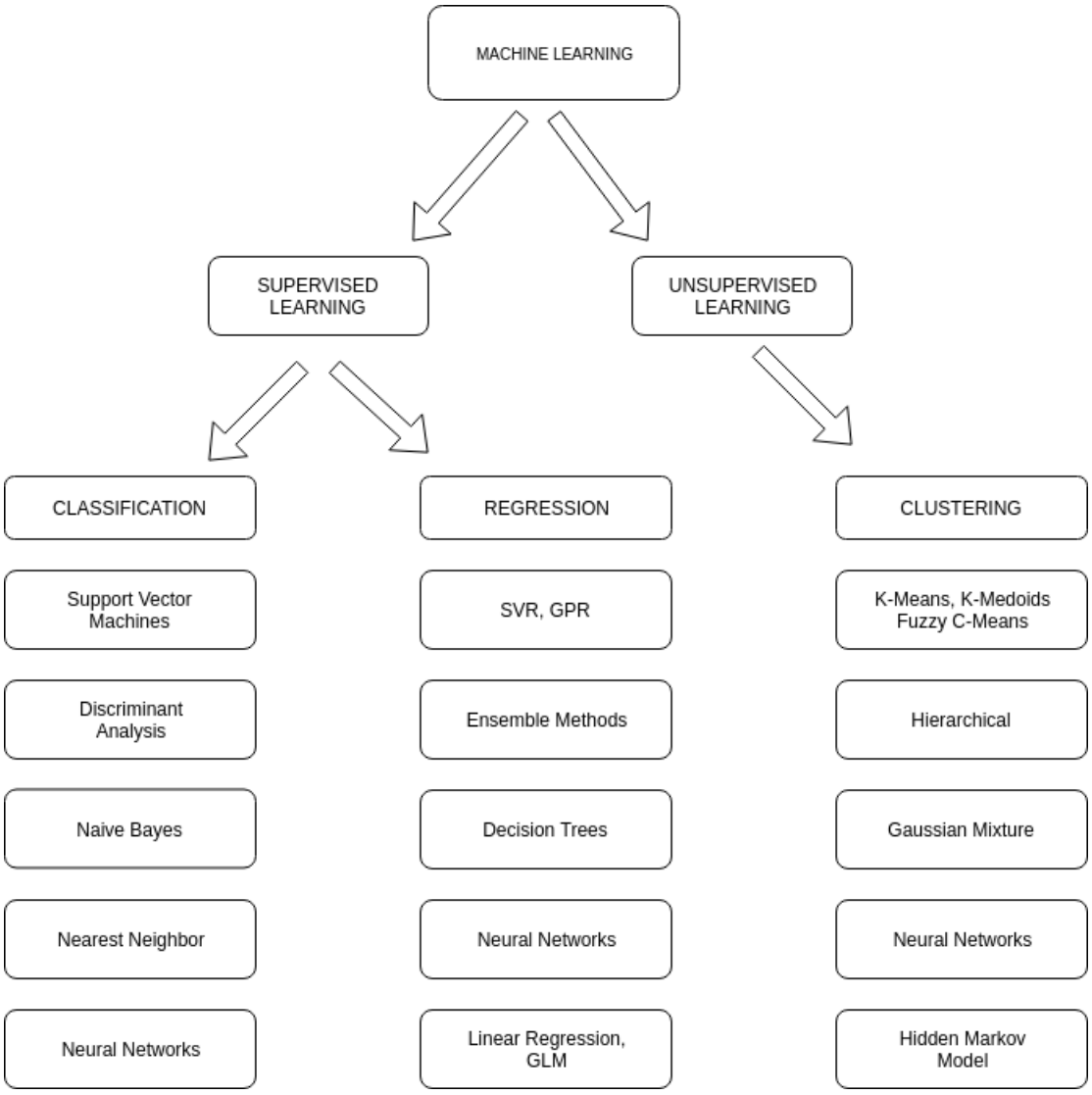


FIGURE 3.10 – Machine Learning Algorithms

Now that we have an idea of the different machine learning algorithms, let's define the ones we will have to use in our work.

Support Vector Machine(SVM) : is a supervised learning algorithm that sorts data into two categories. It is trained with a series of data already classified into two categories, building the model as it is initially trained. The task of an SVM algorithm is to determine which category a new data point belongs in. This makes SVM a kind of non-binary linear classifier. An SVM algorithm should not only place objects into categories, but have the margins between them on a graph as wide as possible [51]. Some applications of SVM include :

- Text and hypertext classification
- Image classification
- Recognizing handwritten characters

- Biological sciences, including protein classification

K-Nearest Neighbor : is a data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it.An algorithm, looking at one point on a grid, trying to determine if a point is in group A or B, looks at the states of the points that are near it. The range is arbitrarily determined, but the point is to take a sample of the data. If the majority of the points are in group A, then it is likely that the data point in question will be A rather than B, and vice versa.The k-nearest-neighbor is an example of a "lazy learner" algorithm because it does not generate a model of the data set beforehand. The only calculations it makes are when it is asked to poll the data point's neighbors. This makes k-nn very easy to implement for data mining [52].

Let's finish this section by discussing about evaluation of a classification system. Let's take an example of a binary classifier, which predicts two classes : class 0 and class 1. To measure the performance of this classifier, it is customary to distinguish four(4) types of classified elements for the desired class :

- True positive TP : Element of class 1 correctly predicted.
- True negative TN : Element of class 0 correctly predicted.
- False positive FP : Element of class 0 poorly predicted.
- False negative FN : Element of class 1 poorly predicted.

It is possible to calculate several indicators from this information. Let's quote a few :

$$precisionscore = \frac{TP}{TP + FP} \tag{3.1}$$

$$recallscore = \frac{TP}{TP + FN} \tag{3.2}$$

$$f1score = 2 \times \frac{precision \times rappel}{precision + rappel} \tag{3.3}$$

$$FAR = \frac{FP}{VP + VN + FP + FN} \times 100 \tag{3.4}$$

$$FRR = \frac{FN}{VP + VN + FP + FN} \times 100 \tag{3.5}$$

The efficiency of the classifier is quantitatively represented in terms of false acceptance rate (FAR) and false rejection rate (FRR). The false acceptance rate represents the percentage of class 0 predicted as 1 by the system and the false rejection rate represents the percentage of class 1 predicted to be 0 by the system. In general, if these values are close to 1 so our system is reliable.

3.7.4 Neural network

Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Neural neworks are typically organized in layers. Layers are made up of a number of interconnected *nodes* which contain an *activation function*. Patterns are presented to the network via the *input layer*, which communicates to one or more *hidden layers* where the actual processing is done via a system of weighted *connections*. The hidden layers then link to an *output layer* where the answer is output as shown in the figure 4.3 [53].

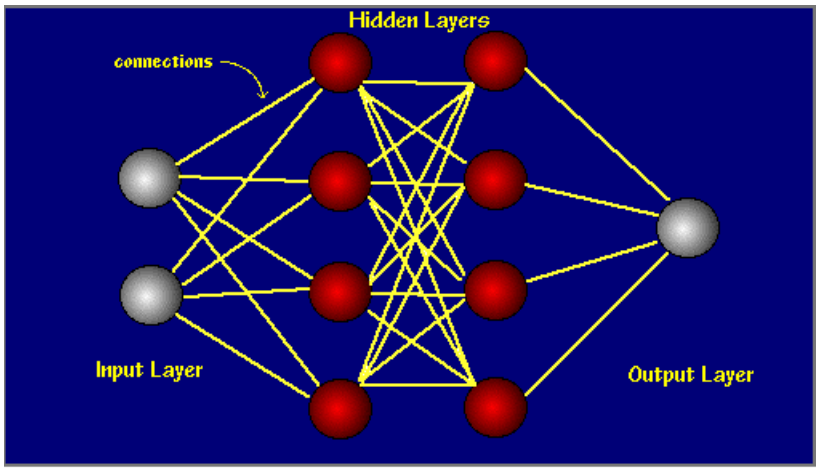


FIGURE 3.11 – Architecture of a neural network [53]

Most neural network contain some form of "learning rule" which modifies the weights of the connections according to the input patterns that it is presented with. In a sense, neural network learn by example as do their biological counterparts, a child learns to recognize dogs from examples of dogs.

Although there are many different kinds of learning rules used by neural networks. The delta rule is often utilized by the most common class of neural networks called **backpropagational neural networks** (BPNNs). Backpropagation is an abbreviation for the backwards propagation of error.

With the delta rule, as with other types of backpropagation, learning is a supervised process that occurs with each *cycle* or *epoch* (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments. More simply, when a neural network is initially presented with a pattern it makes a random guess as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights.

Since the nature of the error space can not be known a prioi, neural network analysis often requires a large number of individual runs to determine the best solution [53].

3.8 Materials and methods

3.8.1 Tools

3.8.1.1 Development environment

Learning system

System kit

For our work of study and application we had to use a laptop which characteristics are as follows :

- **Model** : OMEN HP Laptop
- **Operating system** : Ubuntu 18.04 LTS
- **Architecture** : 64 bits
- **Processor** : Intel(R) Core(TM) i7-7700HQ CPU @ 2,80GHz
- **RAM memory** : 8 Go

Learning kit

The software tools we will need are :

- **Python** : it is an object-oriented programming language that has a multi-platform character. It is a powerful language, both easy to learn and rich in possibilities. It is also very easy to extend the existing features, so there are libraries that help the developer to work on particular projects. Several libraries can be installed. It is obviously very suitable for machine learning applications.
- **Anaconda** : is free distribution of Python and R programming languages and has many other tools such as spyder (an abbreviated EDI integrated development environment in French for writing python code), matplotlib (for visualizing data across curves) . . ., for applications related to data science and machine learning (large-scale data processing, predictive analysis, scientific computing), which aims to simplify the management and deployment of applications related to machine learning [41].
- **Tensorflow** : is a programming framework for numerical computation that was made Open Source by Google in November 2015. TensorFlow has been growing in popularity, quickly becoming one of the most used frameworks for Deep Learning and therefore the neural networks. Its name is inspired by the fact that the current operations on

neural networks are mainly done via multi-dimensional data tables, called Tensors. A two-dimensional tensor is the equivalent of a matrix. Its rich documentation makes it an easy-to-use library [42].

- **Keras** : is a high-level neural network API, written in Python and able to run on TensorFlow or Theano. It has been developed with a focus on rapid experimentation. Being able to go from the idea to a result with a little delay as possible is the key to doing good research. Keras was designed as an interface rather than an end-to-end learning environment. It presents a set of higher level and more intuitive abstractions that make easy the configuration of neural networks [43].

- **Scikit-learn** : is a free Python library dedicated to machine learning. It includes functions for estimating random forests, logistic regressions, classification algorithms, and support vector machines. It is designed to harmonize with other Python free libraries, including NumPy and SciPy [44].

Application of the dynamic verification system

Web server for the back end

In order to set up the server that will be responsible for linking our verification system to the web client we have opted for a set of python framework to set up our server that are :

- **Flask** : is a python web framework that will allow us to simulate a web server locally on our machine.

- **SocketIO** : is a library based on a system of sending and receiving events in order to trigger a given action. It is available in JavaScript and in Python so that you can communicate the client and the server. It will mainly allow us to send a notification informing us that the training phase of a model is complete.

Web kit for the front-end

To communicate with the server, we will implement an application based on web technologies namely :

- **HTML** (HyperText Markup Language) : It allow us to manage and organize the content. It is in HTML that we will write what must be displayed on the page : text, links, images, etc.

- **CSS** (Cascading Style Sheets) : also called Stylesheets, the role of CSS is to manage the appearance of the web page (layout, positioning, decoration, colors, text size ...). This language came to complete the HTML in 1996. We will therefore use the latest version

CSS3 which provides particularly expected features such as rounded borders, gradients, shadows, etc.

- **JavaScript** : it is an interpreted programming language, that is, it needs an interpreter in order to be executed. It is used mainly in web pages. Just like HTML, Javascript is executed by the browser of the client : we speak of a client-side behavior, as opposed to the server-side when the code is executed by the server with the example of php. It will allow us to manage events at the user level.

- **Bootstrap** : it's a CSS library, but not only, since it also includes HTML and JavaScript components. It includes a simple and effective grid system to put in order the visual aspect of a web page. It brings style for buttons, forms, navigation. . . it also takes into account the sizes of screens very varied.

- **AngularJS** : is a javascript library built around key concepts namely : The MVC (Model-View-Controller) architecture that separates the view from the logical set of the application.

3.8.1.2 Dataset

For our study, we will use **SVC 2004** database. SVC 2004 consists of two separate signature verification tasks using two different signature databases [49]. The signature data for the first task contain coordinate information only, but the signature data for the second task also contain additional information including pen orientation and pressure. Thus our studies will be done only on Task 2 data since the data of Task 1 are incomplete. It should be noted that Task 2 is a set of signatures of 40 users including 20 authentics and 20 falsifications which made 40 signatures for each user.Each genuine/forgery signature is stored in a separate text file. The file names are in the format "USERx_y.txt", where x (1..5) indicates the user and y (1..40) indicates one signature instance of the corresponding user, with the first 20 (1..20) representing genuine signatures and the rest (21..40) representing skilled forgeries provided by the other users.

In each text file, the signature is simply represented as a sequence of points. The first line stores a single integer which is the total number of points in the signature. Each of the subsequent lines corresponds to one point characterized by seven features listed in the following order :

- **X-coordinate** : scaled cursor position along the x-axis

- **Y-coordinate** : scaled cursor position along the y-axis

- **Time stamp** : system time at which the event was posted

- **Button status** : current button status (0 for pen-up and 1 for pen-down)

- **Azimuth** : clockwise rotation of cursor about the z-axis
- **Altitude** : angle upward toward the positive z-axis
- **Pressure** : adjusted state of the normal pressure

3.8.1.3 Features extracted from the data

This part is to extract features from the signatures which will be provided as input to the neural networks. The set of features retained can be divided into two classes : static features and dynamic features. Static features are the features calculated with x and y coordinates of each token while the dynaminque features relate to features calculated from data such as pressure, aziumuth, altitude, and stylus state. Thus we retained a total of twenty-seven (27) features.

3.8.1.4 Performance measurement

In order to be able to evaluate our performances throughout the study we decide to consider the metrics that are the F1 score, the AUC, the recall, the precision. However, let's try to understand the information that each of these metrics bring :

- **Precision** : is the ratio between the correctly predicted positive observations and the total predicted positive observations (see formula 3.1). The information this metric provides when considering the purpose of this study is that of all the signatures predicted as authentic, how many are truly authentic. Moreover, we note that its formula is a function of the number of FP. Precision is a good metric evaluation when trying to avoid false positive elements.

  For our system we can accept that an authentic signature is considered to be false because even an authentic signer is subject to several conditions during the signature so the stylus used, the support on which he signs, his mood etc, can make the signature he produces look like a forgery. But it would be inconceivable anyway that a falsified signature is considered to be true (elements classified as False Positives). So when evaluating our performance, we put a special emphasis on precision.

- **Recall** : is the ratio between number of true positives and the number of actually positive examples in the data set (see formula 3.2). The information provided by this metric when considering the object of this study is that of all the authentic signatures, how many have been predicted as authentic. Unlike precision, this metric is ideal for checking performance when trying to have as few False Negative elements as possible.

- **F1 score** : is a combination of precision and recall (see formula 3.3). It measures the accuracy of the model. It is interesting when looking for a balance between precision and recall.

- **AUC** : is a metric for a binary classification. It indicates the effectiveness of the model in distinguishing classes. The higher the AUC, better the model can correctly distinguishes authentic and falsified signatures.

3.8.2 Study method

Throughout the study, the various phases of model learning and evaluation by neural networks, in general, are summarized below the steps of the figure 4.4 :

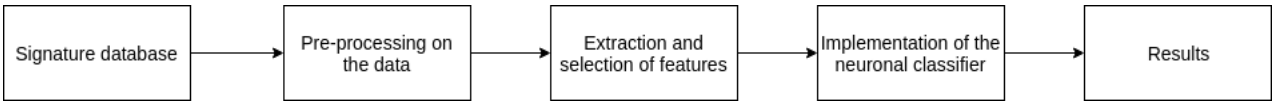


FIGURE 3.12 – Learning and evaluation phases by neural networks

Signature database

For our study we will use data of the "Task 2" folder contained in the SVC 2004 database presented previously.

Pre-processing on the data

This part is to pre-process the signatures and make them ready for feature extraction. The steps to perform the pre-processing are as follows :

- For all signatures samples, remove the tokens at the end of the signature sample which have pressure applied (or pen status) equal to 0.
- For all signatures samples, if the timestamp for any two consecutive tokens are the same, then make the timestamp value of the second token equal to the average of the timestamp values of the tokens above and below it.
- In order to ensure uniformity between two signature samples of the same user, it is desirable that the values of co-ordinates for all the points of the signature are decided on the basis of their distance from a reference point located on the signature itself. A solution to this problem is to shift the origin to the co-ordinate which has the minimum x and y co-ordinate values. This can be done by subtracting the minimum X value and the minimum Y values from the X and the Y columns.

Extraction and selection of features

We will use a set of twenty-seven(27) features (statics and dynamics) extract from each signature.

Implementation of the neuronal classifier

Features extracted from all signatures are used to create a trained neural network using the back-error propagation algorithm for each user. The number of hidden layers and neurons in the latter, the number of epochs, the activation functions and other parameters of the neural network are defined and the neural network is created on the basis of this information.

The set of signatures for each user is divided into two parts : the training set and the test set. In the learning or training phase, 75% of the user's signatures samples (30 signatures) are used and the remaining 25% (10 signatures) are set aside for testing. There is an equal number of original and falsified signatures samples for each user.

Results

At this level we evaluate the performance of our model by calculating metrics such as F1 score, precision score, recall score and AUC for each user.

3.8.2.1 Evaluation of the importance of extracted features

During this step, We will proceed with the validation of the features used, ie for all features, starting from the first feature and adding the other feature as we go along in a network of neurons in order to understand the importance of each feature by measuring, precision, recall and AUC to evaluate performance with each new feature added.

3.8.2.2 Study of the architecture of the optimal neural network

During this step, We will proceed with the validation of the features used, ie for all features, starting from the first feature and adding the other feature as we go along in a network of neurons in order to understand the importance of each feature by measuring, precision, recall and AUC to evaluate performance with each new feature added.

3.8.3 The different stages of our study

Then, when the final features are retained, we will evaluate the method, ie for the neural network method that will be used, test several neurals network architectures and several parameters with all features retained for the study on each architecture by measuring, precision, recall and AUC to evaluate performance.

3.8.4 Establishment of our verification system

On the basis of our study, we realize an autonomous system based on web technologies able to set up a classification model (authentic or falsified signature) with a neural network for each user who provides to the system a set of data consisting of authentic signatures and falsified

signature. The system must also be able in the future when given a single signature sample to evaluate according to a probability value the authenticity of a signature of a user who has already trained a model before.

3.9 Results and discussions

3.9.1 Studies and discussions on the features

Throughout this section we will conduct a study on the extracted features that were presented in the previous section to understand their importance for a good performance of the verification system. So at each step we will add a set of features in relation to their nature to evaluate their contribution to the performance each time. For this work we made our study on the first five users of the "Task 2" folder of our database knowing that 75% of the signatures are used for training either 30 and 25% for the test either 10 for the test per user. Our neurons network will use the back propagation algorithm for learning and will be built with the following basic architecture :

- **Input layer** : The number of neurons in this layer equals to the number of input features.
- **Hidden layers** : we set the number of hidden layers to two, the activation function we will use will be the *relu* function for all the neurons of the hidden layers and for each layer the number of neurons will be determined by the following formula :
- **Output layer** : the number of neurons in this layer will be one which will give the probability of an authentic signature so if the output is close to 1, we are sure of the authenticity thanks to the *sigmoid* activation function. The optimization function is fixed on the *adam* function. We also set a threshold probability of 0.5 for which if we obtain an output probability greater than this threshold, we consider that this signature is authentic and for a probability lower than this threshold, we consider the input signature as being a falsification.

To measure the performance each time we consider the following metrics : F1 score, recall score, precision score and AUC score.

Performance of all static features

TABLEAU 3.7 – Study on statics features

User	F1 score	Recall	Precision	AUC
1	0.5454	0.6	0.5	0.5
2	0.9231	0.8571	1	0.9286
3	1	1	1	1
4	1	1	1	1
5	0.5	0.5	0.5	0.5833

When considering only the static features, it has generally been found that high performance is obtained mainly for simple signatures that are not very complex to perform. These features used are therefore effective for very simple authentic signatures that vary slightly in their coordinates for one sample to another but remain insufficient for much more complex signatures. In certain special cases, it is possible to bring the authentic user to redo new signatures in order to have a solid database for the different possible variations of his signatures and thus obtain a very efficient verification system for this user.

Performance of all features (statics and dynamics)

TABLEAU 3.8 – Study on statics and dynamics features

User	F1 score	Recall	Precision	AUC
1	1	1	1	1
2	1	1	1	1
3	0.9090	0.8333	1	0.9166
4	1	1	1	1
5	1	1	1	1

we notice the improvement of the performances also almost all the precisions of the users are at 1 when we add the dynamic features. We can conclude that the dynamic features are very importants for the obtaining of very good performances, they have many impacts in the distinction of an authentic and falsified signature because they are unique for each human, so they must be retained as important features for the realization of a powerful verification system.

3.9.2 Studies on neurals network architectures

Now that we have retained the features to use, let's go now to the validation of the optimal architecture, which is the final architecture of the neural network and the optimal parameters of the latter that will be retained to have a system efficient and effective for any user. For this work we made our study on the first ten users of the "Task 2" folder of our database knowing that 75% of the signatures are used for training either 30 and 25% for the test either 10 for the test per user. Our neurons network will use the back propagation algorithm for learning and will be built with the following basic architecture :

- **Input layer** : The number of neurons in this layer is twenty-seven(27).
- **Hidden layers** : we will vary the number of layers hidden between two and three, the activation function that will be used will be the *relu* function for all the neurons of the hidden layers with fourteen (14) neurons on each hidden layer.
- **Output layer** : the number of neurons in this layer will be one which will give the probability of an authentic signature so if the output is close to 1, we are sure of the authenticity thanks to the *sigmoid* activation function. The optimization function will vary between *adam*, *rmsprop* and *sgd*. We also set a threshold probability of 0.5 for which if we obtain an output probability greater than this threshold, we consider that this signature is authentic and for a probability lower than this threshold, we consider the input signature as being a falsification.

The elements that will allow us to measure performance each time will be F1 score, recall, precision and AUC. However, we only record the minimum value of each metric in table 3-8 :

TABLEAU 3.9 – Results of minimal values of metrics on the study of neural architecture

Number of hidden layers	Batch size	Epochs	Optimizer	F1 score	Recall	Precision	AUC
2	3, 5	1000, 2000, 5000	adam	0.75	0.6	1	0.8
			rmsprop	0.75	0.6	1	0.8
			sgd	0.8888	0.8	1	0.9
3	3	1000, 2000	adam	0.75	0.6	1	0.8
			rmsprop	0.75	0.6	1	0.8
			sgd	0.6153	0.8	0.5	0.5
		5000	adam	0.75	0.6	1	0.8
			rmsprop	0.75	0.6	1	0.8
			sgd	0.8888	0.8	1	0.9
	5	1000, 2000	adam	0.75	0.6	1	0.8
			rmsprop	0.75	0.6	1	0.8
			sgd	0	0	0	0
		5000	adam	0.75	0.6	1	0.8
			rmsprop	0.75	0.6	1	0.8
			sgd	0.8888	0.8	1	0.9

The optimal architecture obtained by varying all of these parameters is : a neurals network with two hidden layers and fourteen (14) neurons on each layer. For the parameters we have a batch of 3 and an optimizer 'sgd' over 5000 epochs. Hence the following architecture :

- **Input layer** : consisting of twenty-seven(27) neurons.
- **Hidden layers** : two(2) hidden layers, fourteen(14) neurons per layer and *relu* function for activation.
- **Output layer** : consisting of one(1) neuron with *sgd* for optimization.

3.9.3 Comparison of our method with the k-NN and SVM methods

For this work the study is carried out on all forty (40) users of the "Task 2" file of our database, knowing that 75% of the signatures were used for training, ie thirty (30) and 25% for the test is ten (10) for the test per user which in total for all users 1200 signatures for the training and 400 for the test, we will use our main architecture retained during the study on the optimal neural network architecture after that we will use the support vector machine method and the K nearest neighbor method in order to make a comparison based on the performance indices FAR and FRR.

- The use of the optimal architecture of our neural network allowed us to have a FAR of 1.25% and a FRR of 0.75% on the 40 users.
- Using the support vector machine method with basic parameters (for a linear hyperplane) resulted in a FAR of 1.5% and a FRR of 0.25% on the 40 users.
- Using the k nearest neighbor method with k = 3 resulted in a FAR of 3.5% and a FRR of 1% on the forty (40) users.

We note that the k nearest neighbor method is not very suitable for our verification system and also that the neural networks are more efficient to detect a falsified signature compared to the support vector machine which is particularly efficient to detect an authentic signature.

3.9.4 Comparison of our study to another using the same method

The best results obtained in paper[25] on "Task 2" users allowed us to correctly classify twenty-five (25) users across all forty (40) users (all users who have zero false positive and zero false negative during the test phase) with a FAR of 17% and a FRR of 1.5% whereas our study allowed us to obtain a better result which is thirty-two(32) users correctly classified on forty (40) users with a FAR of 1.25% and a FRR of 0.75%.

3.9.5 Comparison of our study to a static verification study

Generally, when we compare our performance to static verification systems, we note that our results for our study are much better than these. We can deduce that the dynamic verification by its dynamic feautures (pressure, azimuth, time etc ...) gives us very indispensable and much more reliable information to detect a falsified signature and an authentic signature, however it must be much more considered than a static verification system.

Conclusion

Throughout our work, we conducted a performance study on a dynamic handwritten signature verification system using the neural network method. This study was essentially divided into two parts. First, we measured the impact of each feature considered on the performance of the system then once we have realized the importance of each of our features, we used these features to conduct a study on several neural network architectures in order to identify optimal architecture and thus to have the best possible results. Based on our study, we obtained on the database used (SVC 2004), 1.25% of FAR and 0.75% of FRR out of a total of 400 signatures for 40 different users. These results show that we have better results than other works so we have implemented a dynamic signature verification system to implement the results of our study work.

As a perspective, we initially plan to improve our results compared to the study by doing an even deeper analysis of the features that could be added or deleted in order to have less noise in our features and thus positively impact our performance. For the same purpose we also plan to carry out a study on the architecture of neural networks by more extensively varying network parameters such as activation functions, the number of hidden layers as well as the number of neurons on each hidden layer, the optimiaztion function, the number of epochs on which to do the training, the size of the batch etc, in order to have a much better architecture and to have good performances. In the second time we plan to improve our verification system by finding several neural network architectures giving very interesting results because the same architecture can not be adapted to all our training data, so we could have a much more system reliable that could be deployed in production.

Annexe A

Détail des tests effectués pour la sélection de caractéristiques et de l'architecture neuronale optimale

(a) Étude des caractéristiques 1 à 7

Utilisateur	F-mesure	Rappel	Précision	AUC
1	$0,6666$	$0,8$	$0,5714$	$0,6$
2	$0,8333$	$0,7143$	1	$0,8571$
3	$0,8$	$0,6666$	1	$0,8333$
4	$0,9091$	$0,8333$	1	$0,9166$
5	$0,8$	1	$0,6666$	$0,8333$

(b) Étude des caractéristiques 1 à 12

Utilisateur	F-mesure	Rappel	Précision	AUC
1	$0,5454$	$0,6$	$0,5$	$0,5$
2	$0,9231$	$0,8571$	1	$0,9286$
3	1	1	1	1
4	1	1	1	1
5	$0,5$	$0,5$	$0,5$	$0,5833$

TABLEAU A.1 – Evaluation des caractéristiques de 1 à 7 et de 1 à 12

(a) Étude des caractéristiques 1 à 14

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	$0,9090$	$0,8333$	1	$0,9166$
4	$0,9090$	$0,8333$	1	$0,9166$
5	$0,8888$	1	$0,8$	$0,9166$

(b) Étude des caractéristiques 1 à 18

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	$0,9090$	$0,8333$	1	$0,9166$
4	$0,9090$	$0,8333$	1	$0,9166$
5	$0,8888$	1	$0,8$	$0,9166$

TABLEAU A.2 – Evaluation des caractéristiques de 1 à 14 et de 1 à 18

(a) Étude des caractéristiques 1 à 20

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	0,9090	0,8333	1	0,9166
4	1	1	1	1
5	0,8888	1	0,8	0,9166

(b) Étude des caractéristiques 1 à 22

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	0,9090	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1

TABLEAU A.3 – Evaluation des caractéristiques de 1 à 20 et de 1 à 22

(a) Étude des caractéristiques 1 à 24

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	0,9090	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1

(b) Étude des caractéristiques 1 à 26

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	0,9090	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1

TABLEAU A.4 – Evaluation des caractéristiques de 1 à 24 et de 1 à 26

TABLEAU A.5 – Étude des caractéristiques 1 à 27

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	0,9090	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1

TABLEAU A.6 – Étude des caractéristiques 15 à 18

Utilisateur	F-mesure	Rappel	Précision	AUC
1	0.8	0.8	0.8	0,8
2	1	1	1	1
3	1	1	1	1
4	0,7272	0,6666	0,8	0,7083
5	0,7272	1	0,5714	0,75

TABLEAU A.7 – Étude des caractéristiques 23 à 27

Utilisateur	F-mesure	Rappel	Précision	AUC
1	0.9090	1	1	0, 9
2	0, 9230	0, 8571	1	0, 9285
3	0, 8571	1	0.75	0, 75
4	0, 8333	0, 8333	0, 8333	0, 7916
5	0, 75	0, 75	0, 75	0, 7916

TABLEAU A.8 – Etude de l'architecture neuronale : nombre de couches cachées = *2* et *3*, taille du batch = *3* et *5*, optimiseur = *adam* et *rmsprop*, nombre d'époques = *1000*, *2000* et *5000*

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	0, 9091	0, 8333	1	0, 9166
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	0, 75	0, 6	1	0, 8

TABLEAU A.9 – Etude de l'architecture neuronale : nombre de couches cachées = *2*, taille du batch = *3* et *5*, optimiseur = *sgd*, nombre d'époques = *1000*, *2000* et *5000*

Utilisateur	**F-mesure**	**Rappel**	**Précision**	**AUC**
1	1	1	1	1
2	1	1	1	1
3	0,9091	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	0,8888	0,8	1	0,9

TABLEAU A.10 – Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *3*, optimiseur = *sgd*, nombre d'époques = *1000* et *2000*

Utilisateur	**F-mesure**	**Rappel**	**Précision**	**AUC**
1	1	1	1	1
2	1	1	1	1
3	0,9091	0,8333	1	0,9166
4	0,9230	1	0,8571	0,8750
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	0,6153	0,8	0,5	0,5

TABLEAU A.11 – Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *5*, optimiseur = *sgd*, nombre d'époques = *1000*

Utilisateur	F-mesure	Rappel	Précision	AUC
1	$0,8$	$0,8$	$0,8$	$0,8$
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	$0,5714$	1	$0,4$	$0,5$
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	$0,8333$	1	$0,7143$	$0,8$
10	0	0	0	0

TABLEAU A.12 – Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *5*, optimiseur = *sgd*, nombre d'époques = *2000*

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	$0,5714$	1	$0,4$	$0,5$
6	$0,9230$	1	$0,8571$	$0,875$
7	0	0	0	0
8	1	1	1	1
9	1	1	1	1
10	$0,8333$	1	$0,7143$	$0,8$

TABLEAU A.13 – Etude de l'architecture neuronale : nombre de couches cachées = *3*, taille du batch = *3* et *5*, optimiseur = *sgd*, nombre d'époques = *5000*

Utilisateur	**F-mesure**	**Rappel**	**Précision**	**AUC**
1	1	1	1	1
2	1	1	1	1
3	0,9091	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	0,8888	0,8	1	0,9

TABLEAU A.14 – Performance sur dix utilisateurs avec la méthode des réseaux de neurones

Utilisateur	**F-mesure**	**Rappel**	**Précision**	**AUC**
1	1	1	1	1
2	1	1	1	1
3	0,9091	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	0,8888	0,8	1	0,9

TABLEAU A.15 – Performance sur dix utilisateurs avec la méthode de la machine à vecteur de support

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	0,9091	0,8333	1	0,9166
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	0,9333	1	0,8750	0,8333
8	1	1	1	1
9	1	1	1	1
10	1	1	1	1

TABLEAU A.16 – Performance sur dix utilisateurs avec la méthode des K-voisins les plus proches avec K = 3

Utilisateur	F-mesure	Rappel	Précision	AUC
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	0,9230	1	0,8571	0,8750
5	1	1	1	1
6	0,8888	1	0,8	0,9166
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	0,8888	0,8	1	0,9

TABLEAU A.17 – Tableau comparatif de chaque méthode pour chaque utilisateur

	Réseau de neurones		Machine à vecteur de support		K-voisins plus proches	
Utilisateur	FP	FN	FP	FN	FP	FN
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	1	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	1	0
6	0	0	0	0	0	0
7	0	0	1	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	2	0	0	0	1
11	0	0	0	0	0	0
12	1	0	0	0	3	0
13	0	0	0	0	0	0
14	0	0	0	0	0	1
15	0	0	0	0	0	0
16	0	0	0	0	0	0
17	0	0	0	0	2	0
18	1	0	1	0	2	0
19	1	0	1	0	0	0
20	0	0	0	0	0	0
21	0	0	0	0	0	1
22	0	0	0	0	0	0
23	0	0	0	0	0	0
24	1	0	1	0	2	0
25	0	0	0	0	0	0
26	0	0	0	0	0	0
27	0	0	0	0	0	0
28	0	0	0	0	0	0
29	0	0	0	0	0	0
30	0	0	0	0	0	0
31	0	0	0	0	0	0
32	0	0	0	0	1	1
33	1	0	1	0	0	0
34	0	0	0	0	0	0

Table A.17 Tableau comparatif des méthodes pour chaque utilisateur

35	0	0	0	0	0	0
36	0	0	0	0	1	0
37	0	0	0	0	0	0
38	0	0	0	0	0	0
39	0	0	1	0	1	0
40	0	0	0	0	0	0
Totaux	5	3	6	1	14	4
Taux	FAR = 1,25%	FRR = 0,75%	FAR = 1,5%	FRR = 0,25%	FAR = 3,5%	FRR = 1,0%

Bibliographie

[1] American Heritage Dictionary, 3rd Edition, ver. 3.6a, (SoftKey Intl. Inc., 1994).

[2] Pacut A. and Czajka A., *Recognition of Human Signatures*. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2, 1560-1564 (2001).

[3] Gubta G. and McCabe A., *A Review of Dynamic Handwritten Signature Verification*. Department of Computer Science, James Cook University Townsville, Qld 4811, Australia, (1997).

[4] Brault J. and Plamondon R., *A Complexity Measure of Handwritten Curves : Modeling of Dynamic Signature Forgery* IEEE Transactions on Systems, Man, and Cybernetics, 23(2), 400-413 (1993).

[5] Rostron R., *The Graphologist*, The Journal of the British Institute of Graphologists, 22(2), 28-38 (2004).

[6] Herbst N.M. and Liu C.N., *Automatic Signature Verification Based on Accelerometry*. IBM J Res Dev, 21, 245-253, 1977.

[7] R. A. Huber and A. M. Headrick, *Handwriting Identification : Facts and Fundamentals*. CRC Press, Boca Roton, 1999.

[8] R. Plamondon and F. J. Maarse, *An evaluation of motor models of handwriting*. Systems, Man and Cybernetics, IEEE Transactions on, 19(5) :1060–1072, 1989.

[9] Anatolyevich Kholmatov Alisher, *Biometric Identity Verification Using On-Line and Off-Line Signature Verification*. MSc Sabanci University, (2003).

[10] B. Herbst. J. Coetzer. and J. Preez, *Online Signature Verification Using the Discrete Radon Transform and a Hidden Markov Model*. EURASIP Journal on Applied Signal Processing, vol. 4, pp. 559–571, 2004.

[11] M. Blumenstein. S. Armand. and Muthukkumarasamy, *Off-line Signature Verification using the Enhanced Modified Direction Feature and Neural basedClassification*. International Joint Conference on Neural Networks, 2006.

[12] Ashwini P. and Shalini B., *Handwritten Signature Verification using Neural Network*. International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 1– No.2, January 2012 – www.ijais.org

[13] S.Srihari. K. M. Kalera. and A. XU, *Offline Signature Verification and Identification Using Distance Statistics*. International Journal of Pattern Recognition And Artificial Intelligence ,vol. 18, no. 7, pp. 1339–1360, 2004.

[14] H. S. Srihari and M. Beall, *Signature Verifcation Using Kolmogrov Smirnov Statistic*. Proceedings of International Graphonomics Society,Salemo Italy , pp.152–156, june, 2005.

[15] T.S. enturk. E. O zgunduz. and E. Karshgil, *Handwritten Signature Verification Using Image Invariants and Dynamic Features*. Proceedings of the 13th European Signal Processing Conference EUSIPCO 2005,Antalya Turkey, 4th-8th September, 2005.

[16] Kalenova Diana, *Personal Authentication Using Signature Recognition*. Department of Information Technology, Laboratory of Information Processing, Lappeenranta University of Technology, (2004).

[17] Likforman-Sulem L., Garcia-Salicetti S., Dittmann J., Ortega-Garcia J., Pavesic N., Gluhchev G., Ribaric S. and Sankur B, *Report on the hand and other modalities stated of the art*. Biometrics for Secure Authentication, (2005).

[18] Dong L., Yun-Jian G., Xue-Yong Z., *On-line signature verification based on template matching approach and support vector data description*, International Conference on Computer Application and System Modeling (ICCASM), Vol. 12, pp. 681-685, (2010)

[19] Han, C.C., Chang P.C., Hsu C.C., Jeng B.S., *An on-line signature verification system using multi-template matching approaches*, IEEE Proceedings - 33rd Annual International Carnahan Conference on Security Technology, pp. 477-480, (1999).

[20] Daramola S.A., Ibiyemi T.S., *Efficient on- line signature verification system*, International Journal of Engineering and Technology IJET-IJENS, Vol. 10, No. 4, pp. 48-52, (2010)

[21] Boser B.E., Guyon I.M., Vapnik V.N., *training algorithm for optimal margin classifiers*, Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144-152, (1992).

[22] Kour J., Hanmandlu M., Ansari A.Q., *Online signature verification using GA-SVM*, International Conference on Image Information Processing (ICIIP), pp. 1-4, (2011)

[23] Fauziyah S., Azlina O., Mardiana B., Zahariah A.M., Haroon H., *Signature verification system using Support Vector Machine*, 6th International Symposium on Mechatronics and its Applications (ISMA), pp.1-4, (2009)

[24] Justino E.R., Yocoubi A.E., Bortolozi F., Sabourin R., *An Off-line Signature Verification System Using HMM and Graphometric features* ,4th IAPR International on Document Analysis Systems, 2000.

[25] Shalini B., Pratik B., Dheeraj N., Sandhya N., *Online Signature Forgery Prevention*, International Journal of Computer Applications (0975 – 8887), Volume 75 – No.13, August 2013

[26] Anatolyevich Kholmatov Alisher, *Biometric Identity Verification Using On-Line and Off-Line Signature Verification*. MSc Sabanci University, (2003).

[27] MathWorks, *Introducing Machine Learning*, 2016

[28] Eric Biernat, Michel Lutz *Data science : fondamentaux et études de cas. Machine learning avec Python et R*, 2015, Eyrolles

[29] Carlos Gershenson, *Artificial Neural Networks for Beginners*, 2003.

[30] Alice Cohen-Hadria, *Apprentissage de Structure Musicale par Réseaux Neuronaux Convolutionnels*, rapport de stage, année universitaire 2015-2016

[31] Josh Patterson and Adam Gibson, *Deep Learning*,First Edition, Published by O'Reilly Media, Inc. 2017

[32] Vamsi K. Madasu, *Automatic Bank Check Processing and Authentication using Signature Verification*. Ph.D. thesis, Queensland University, Brisbane, Australia, 2006.

[33] Graham Leedham and Vladimir Pervouchine. *Validating the use of handwriting as a biometric and its forensic analysis*. In Umapada Pal, Swapan K. Parui, and Bidyut B. Chaudhuri, editors, Document Analysis, pages 175– 192. Allied Publishing Ltd., 2005.

[34] R. Sabourin and R. Plamondon, *On the implementation of some graphometric techniques for automated signature verification systems : a feasibility study*. In Third Intl. Symp. on Handwriting and Comp. App., pages 160–162, Montreal, 1987.

[35] A. J. Mauceri, *Feasibility study of personnel identification by signature verification*. Technical report, Anaheim, CA, 1965.

[36] M. Ammar, Y. Yoshida, and T. Fukumura *A new effective approach for off-line verification of signatures by using pressure features*. In 8th Intl. Conf. on Pattern Recognition, pages 566–569, Paris, 1986.

[37] R. A. Huber and A. M. Headrick.*Handwriting Identification : Facts and Fundamentals*. CRC Press, Boca Roton, 1999.

[38] Marcus Liwicki, C. Elisa van den Heuvel, Bryan Found, and Muhammad Imran Malik, *Forensic signature verification competition 4nsigcomp2010 - detection of simulated and disguised signatures*, In ICFHR, pages 715–720, 2010.

Webographie

[39] Wacom Intuos3 6 x 8-Inch Pen Tablet https://www.amazon.com/Wacom-Intuos3-8-Inch-Pen-Tablet/dp/B00030097G consulté le 10 Octobre 2018.

[40] Neural Networks and Deep Learning http://neuralnetworksanddeeplearning.com/ consulté le 10 Octobre 2018.

[41] Anaconda Documentation https://docs.anaconda.com/ consulté le 09 Septembre 2018.

[42] https://blog.xebia.fr/2017/03/01/tensorflow-deep-learning-episode-1-introduction/ consulté le 09 Septembre 2018.

[43] Débuter avec Keras https://www.actuia.com/keras/ consulté le 09 Septembre 2018.

[44] FORMATION PYTHON MACHINE LEARNING AVEC SCIKIT.LEARN https://www.ambient-it.net/formation/formation-python-machine-learning/ consulté le 09 Septembre 2018.

[45] Présentation au sujet : "Html et Css." https://slideplayer.fr/slide/5565137/ consulté le 09 Septembre 2018.

[46] Dynamisez vos sites web avec JavaScript https://openclassrooms.com/fr/courses/1916641-dynamisez-vos-sites-web-avec-javascript/1916776-introduction-au-javascript consulté le 09 Septembre 2018.

[47] Bootstrap 4 : présentation https://laravel.sillo.org/bootstrap-4-presentation/ consulté le 09 Septembre 2018.

[48] Développez vos applications web avec AngularJS https://openclassrooms.com/courses/developpez-vos-applications-web-avec-angularjs consulté le 09 Septembre 2018.

[49] http://www.cse.ust.hk/svc2004/download.html consulté le 07 Septembre 2018.

[50] Le cycle en V https://www.supinfo.com/articles/single/6278-cycle-v consulté le 24 Janvier 2019.

[51] Support Vector Machine (SVM) https://www.techopedia.com/definition/30364/support-vector-machine-svm consulté le 15 Janvier 2019.

[52] K-Nearest Neighbor (K-NN) https://www.techopedia.com/definition/32066/k-nearest-neighbor-k-nn consulté le 15 Janvier 2019.

[53] A Basic Introduction To Neural Networks http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html consulté le 15 Janvier 2019.

[54] SERRURE BIOMÉTRIQUE http://www.guide-serrurerie.fr/serrurerie/serrure-biometrique consulté le 10 Octobre 2018.

[55] Signature manuscrite https://pixabay.com/fr/signature-écriture-manuscrite-523237/ consulté le 10 Octobre 2018.