REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IA20/M2/2022

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Intelligence Artificielle (IA)**

Stéganographie d'images à l'aide de l'apprentissage automatique

---

Par :

RAHOUA AHMED

Soutenu le 26/06/2022 devant le jury composé de :

SLATNIA Sihem	MCA	Président
TIBERMACINE Ahmed	MCA	Rapporteur
TORKI Fatima Zohra	MAA	Examinateur

Année universitaire 2021-2022

Remerciements

Je tiens avant tout à exprimer ma gratitude à Allah Tout-Puissant. Grâce à Dieu, j'ai pu atteindre mes objectifs et réaliser ce travail.

Ce travail est la consécration de long mois de recherche, et il n'aurait jamais pu voire le jour sans les soutiens de mes parents et ma famille qui me motivent et me poussent à donner le meilleur de moi même. Je tiens donc à les remercier pour leurs sacrifices et leur dévouement pour que je puisse réussir.

j'ai eu l'honneur de réaliser ce travail sous la supervision d'une sommité dans le domaine informatique **Dr Tibarmacine Ahmed**. Grâce à son savoir et ses efforts, sa disponibilité à tout moment, sa motivation et son soutien, que j'ai pu réaliser ce projet.

je remercie aussi toute personne qui a contribué de prés ou de loin à ce travail, ainsi que tous mes amis, avec leurs idées ou leur soutien moral.

RAHOUA Ahmed

Résumé

Avec l'évolution du domaine de la technologie de l'information, la sécurité d'information est devenue un enjeu majeur, La stéganographie est une technique essentielle dans le domaine de l'information. La stéganographie est une technique d'encodage d'informations secrètes (comme un texte, une image ou un son) dans un objet (comme une image, un son, texte ou vidéo) connu sous le nom d'objet de couverture.
Au fil des ans, le travail dans la stéganographie d'image a été fait pour encoder une image de résolution inférieure dans une image de résolution supérieure par des méthodes simples comme la méthode de bit de poids faible (en anglais, Least Significant Bit, ou LSB). Des réseaux neuronaux profonds ont alors émergé, visant à placer une image couleur dans une autre image couleur de même taille.
Notre travail consiste à implémenter une architecture encodeur-décodeur pour l'encodage et le décodage de multiples images secrètes à l'intérieur d'une seule image de couverture de la même résolution ont utilisant l'apprentissage profond.

Mots clés : Stéganographie, Stéganalyse, Apprentissage automatique, Apprentissage en profondeur, Réseaux de neurones à convolution(CNN).

ABSTRACT

With the evolution of the field of information technology, information security has become a major issue. Steganography is an essential technique in the field of information. Steganography is a technique for encoding secret information (such as text, image, or sound) into an object (such as an image, sound, text, or video) known as a cover object. Over the years, work in image steganography has been done to encode a lower resolution image into a higher resolution image by simple methods like the Least Significant Bit (LSB) method. Deep neural networks then emerged, aiming to place a color image in another color image of the same size.
Our job is to implement an encoder-decoder architecture for encoding and decoding multiple secret images within a single cover image of the same resolution using deep learning.

Key words : Steganography, steganalysis, Machine learning, Deep leaning, Convolution neural networks(CNN).

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'information est un élément déterminant dans tous les domaines. Tout au long de l'histoire, l'humanité a essayé d'envoyer des informations d'une façon sécurisée. La dissimulation d'information a été utilisée comme instrument de sécurisation pour les stratégies militaires et échange de données secrètes. Le terme dissimulation d'information est très général ; il désigne simplement le fait de cacher une information dans un support. Il s'agit d'une libre adaptation de l'expression Anglaise "Information Hiding" ou bien "Data Hiding". Ce terme qui est apparu au moment du « First International workshop on Information Hiding" en 1996 lors de l'adoption d'un corpus relatif à la dissimulation d'information. L'une des stratégies est de rendre l'information inintelligible, par des moyens de chiffrement, l'autre est de dissimuler l'existence même de l'information. Les algorithmes et les techniques de dissimulation d'information se distinguent les uns des autres selon l'objectif et l'application utilisée. Les techniques les plus connues sont la cryptographie, le tatouage et la stéganographie .

La stéganographie est l'art de l'écriture couverte ou cachée. bien avant, il existe la stéganog raphie sur support physique ces techniques consistaient à camoufler l'information secrète dans le support physique même du message anodin. Bien entendu avec l'avènement du traitement numérique de l'information, cette stéganographie a disparu mais reste néanmoins assez intéressante voire même amusante. La stéganographie sur support physique nécessite bien entendu l'usage de courrier de physique (ou de messagers). L'histoire veut que les premières utilisations de la stéganographie datent du 5 eme siècle avant Jésus-Christ [1]. Herodotus, auteur grec, relate les communications secrètes entre deux chefs de guerre qui utilisaient des esclaves pour passer des messages et plans de batailles. L'idée était simple, ils tatouaient sur le crâne des esclaves le message, laisser repousser les cheveux.

En stéganographie moderne et avec l'avènement de l'informatique et le développement des échanges électroniques, les possibilités de cacher les messages se sont multipliées : on peut cacher des informations comme texte, image, audio,. . . , dans une autre donnée numérique : texte, image, vidéo, audio. Le procédé stéganographié place un message caché dans un support de transport, appelé le transporteur. Le transporteur peut être visible publiquement. Pour plus de sécurité, le message caché peut également être chiffré, di-

minuant ainsi la probabilité de découverte de contenu même si l'existence du message détecté.

Au fil des ans, la stéganographie a été utilisée pour coder une image de résolution inférieure en une image de résolution supérieure par des méthodes simples telles que la manipulation LSB. Ces dernières années, certaines études ont utilisé des réseaux de neurones profonds (en anglais, Deep Neural Network, ou DNN) pour cachent une image couleur pleine grandeur dans une autre image de même taille (Baluja,2017 basé en CNN)[2],(Zhangjie Fu,2020 basé en réseau contradictoire génératif (en anglais, Generative Adversarial Network, ou GAN)) [3], leur approche comprime et distribue la représentation de l'image secrète sur tous les bits disponibles.

L'objectif principal de ce travail consiste à concevoir une architecture encodeur-décodeur basée sur l'apprentissage profond pour dissimuler multiples images dans une image en utilisant les idées des articles susmentionnés.

Organisation du mémoire

Chapitre 1 : Vision par ordinateur et stéganographie

Dans ce chapitre nous avons présenté le domaine de vision par ordinateur , le système de traitement d'image et l'image numérique et ses caractéristiques, les types existants, le codage des couleurs...etc., puis nous allons présenter la stéganographie, sa définition, son principe, ses caractéristiques, et enfin, expliqué la technique de stéganalyse.

Chapitre 2 : Apprentissage machines

Ce chapitre est réservé à l'apprentissage automatique, sa définition, son objectif, ses différents types (supervisé, non-supervisé, semi supervisé et avec renforcement), puis les algorithmes d'apprentissage tel que l'apprentissage profond et ses différentes méthodes : CNN, RNN.

Chapitre 3 : Conception et implémentation

Dans ce chapitre nous présentons la conception générale de notre système développé de dissimuler multiples images dans une image basée sur l'apprentissage profond, puis la base de données et les paramètres de l'implémentation,enfin les résultats obtenus.

Chapitre 1

Vision par ordinateur et stéganographie

1.1 Introduction

Dans ce premier chapitre, on va présenter le domaine de vision par ordinateur. Ensuite, nous définissons les notions nécessaires au traitement d'image, où l'on utilise une brève description pour chaque notion. En particulier, nous considérons la définition de ce qu'une image et ses caractéristiques, les types existants, ...etc. Enfin, nous allons parler sur la stéganographie, sa définition, sa principe, ses différents types, ainsi que ses caractéristiques, et nous allons discuter sur la technique de stéganalyse qui consiste à identifier la présence d'un message secret.

1.2 Vision par ordinateur

1.2.1 Définition

La vision par ordinateur est un domaine inclus dans le rayon d'intelligence artificielle qui propose plusieurs biais et méthodes d'analyse, de classifications et de traitements dans le but de comprendre l'histoire qui se déroule dans une image.

En tant qu'humains, c'est assez simple. Mais pour les ordinateurs, la tâche est extrêmement difficile... Pour atteindre ce but et permettre cette compréhension à I'ordinateur, la vision par ordinateur se réfère à des algorithmes variés et faits appel à plusieurs technologies.

On distingue deux types de traitement d'images auxquels la vision par ordinateurs fait recours :

- La vision haut niveau dont le but d'extraire les attributs symboliques, par exemple la reconnaissance des lettres écrites à la main.
- La vision basse niveau (ou vision pré-attentives) qui traite une grande quantité de pixels et les transforme en attributs nécessaires pour la vision haut niveau, par exemple l'extraction des contours, détection des régions, texture, etc.

En effet, la vision par ordinateur fait recours aux techniques de traitement d'images vu que le principal paramètre d'entrée de cette dernière est l'image [4].

1.2.2 Historique

Le développement de la vision par ordinateur a commencé dans les universités pionnières de l'intelligence artificielle à la fin des années 1960. L'objectif était d'imiter le système visuel humain, première étape pour doter les robots d'un comportement intelligent. En 1966, on croyait que cela pouvait être réalisé grâce à un projet d'été, en attachant une caméra à un ordinateur et en lui faisant « décrire ce qu'il voyait » [5].
Ce qui distinguait la vision par ordinateur du domaine prédominant du traitement d'images numériques à cette époque était le désir d'extraire une structure tridimensionnelle d'images dans le but de parvenir à une compréhension complète de la scène. Des études dans les années 1970 ont formé les premières bases de nombreux algorithmes de vision par ordinateur qui existent aujourd'hui, y compris l'extraction des bords d'images, l'étiquetage des lignes, la modélisation non polyédrique et polyédrique, la représentation d'objets sous forme d'interconnexions de structure plus petites, le flux optique et estimation de mouvement [5].
La décennie suivante a vu des études basées sur une analyse mathématique plus rigoureuse et des aspects quantitatifs de la vision par ordinateur. Ceux-ci incluent le concept d'espace d'échelle, l'inférence de la forme à partir de divers indices tels que l'ombrage, la texture et la mise au point, et les modèles de contour connus sous le nom de serpents. Les chercheurs ont également réalisé que bon nombre de ces concepts mathématiques pouvaient être traités dans le même cadre d'optimisation que la régularisation et les champs aléatoires de Markov [5].
Dans les années 1990, certains des thèmes de recherche précédentes sont devenus plus actifs que les autres. La recherche sur les reconstructions projectives 3Ds a permis de mieux comprendre l'étalonnage de caméras. Avec l'avènement des méthodes d'optimisation pour la calibration des caméras, on s'est rendu compte que de nombreuses idées avaient déjà été explorées dans la théorie de l'ajustement des faisceaux dans le domaine de la photogrammétrie. Cela a conduit à des méthodes pour des reconstructions 3Ds éparses de scènes à partir de plusieurs images. Des progrès ont été réalisés sur le problème de la correspondance stéréo dense et d'autres techniques stéréo à voir multiples. Dans le même temps, des variations de coupe graphique ont été utilisées pour résoudre la segmentation d'image [5].
Cette décennie a également marqué la première fois que des techniques d'apprentissage statistique a été utilisée dans la pratique pour reconnaître les visages dans les images (voir Eigenface). Vers la fin des années 90, un changement important s'est produit avec l'interaction accrue entre les domaines de l'infographie et de la vision par ordinateur.

Cela comprenait le rendu basé sur l'image, l'interpolation de vue, l'assemblage d'images panoramiques et le premier rendu de champ lumineux [5].
Des travaux récents ont vu la résurgence des méthodes basées sur les fonctionnalités ; utilisées en conjonction avec des techniques d'apprentissage automatique et des cadres D'optimisations complexes, Les progrès des techniques d'apprentissage en profondeur font donné une nouvelle vie au domaine de la vision par ordinateur. La précision des algorithmes d'apprentissage en profondeur sur plusieurs ensembles de données de vision par ordinateur de référence pour des tâches allant de la classification, de la segmentation et du flux optique a surpassé les méthodes antérieures [5].

1.3 Vision humaine et vision artificielle

Une des particularités des êtres vivants est de pouvoir acquérir des images via l'œil, comme une information, puis de pouvoir l'interpréter via le cerveau. L'enjeu de la vision artificielle est de permettre à un ordinateur de "voir" c'est-à-dire, comme l'homme, de récupérer l'information par l'intermédiaire d'un dispositif d'acquisition d'image puis d'exploiter [6].

1.3.1 Vision humaine

Comprendre le contenu d'une image est automatique pour l'humain mais difficile pour la machine. La raison en est la complexité du processus de vision humaine qui effectue un traitement parallèle des données dont une partie seulement est purement visuel œil (= 1 million de cellules sensibles), cerveau (= des milliards de neurones). Nommer un objet est un processus d'inférence (de déduction) qui nécessite des connaissances non contenues dans l'image. Le cerveau humain n'est pas une machine précise, Il nous est difficile de mesurer exactement la taille des objets. Cependant, nos approximations nous permettent d'aller plus loin dans la reconnaissance [6].
Les avantages de l'être humain c'est de faculté d'abstraction les modèles des objets que nous utilisons sont très abstraits, Faculté de déduction face à un nouvel objet, nous pouvons le deviner grâce à sa sémantique, pas seulement l'apparence, Faculté d'apprentissage par exemple un bébé connaît peu le monde à sa naissance, mais il apprend peu à peu à le reconnaître, Immersion dans le milieu : nous ne faisons pas que voir les objets, nous les vivons (vision active) et informations a priori : nous prenons pour acquises plusieurs informations qui simplifie l'interprétation (horizon, structure logique d'un objet, bon sens, ...) [6].

1.3.2 Vision artificielle

Une image peut montrer la vue partielle ou totale d'une scène, contenir beaucoup de détails et montrer des objets de différentes échelles (environnement, galaxies, bactéries, ...). La vision artificielle permet de faire certaines choses que la vision humaine ne permet pas et elle permet de voir l'invisible [6].

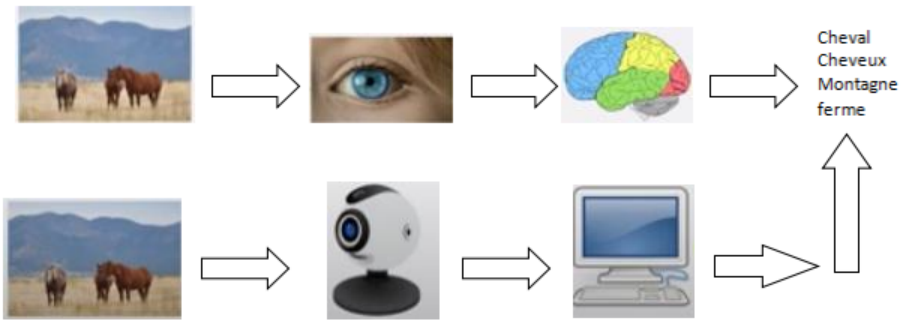

FIGURE 1.1 – Vision humaine et vision artificielle [5].

1.4 Traitement d'image

1.4.1 Définition de l'image

C'est un ensemble structuré d'informations qui peut être décrit mathématiquement sous la forme d'une fonction f(x,y), définie dans un domaine borné, tel que x et y sont les coordonnées spatiale d'un point de l'image et f est une fonction d'intensité lumineuse et de couleur. Sous cet aspect, la numérisation d'image c'était l'un des sujets primordiaux des recherches scientifiques qui visent l'exploitation d'image par la machine et les techniques modernes existantes [7].

1.4.2 L'image numérique

Le terme d'image numérique désigne, dans son sens le plus général, toute image qui a été acquise, traitée et sauvegardée sous une forme codée représentable par des nombres (valeurs numériques).
La numérisation est le processus qui permet de passer de l'état d'image physique (image optique par exemple) qui est caractérisée par l'aspect continu du signal qu'elle représente (une infinité de valeurs dans l'intensité lumineuse par exemple), à l'état d'image numérique qui est caractérisée par l'aspect discret (l'intensité lumineuse ne peut prendre que des valeurs quantifiées en un nombre fini de points distincts). C'est cette forme numérique qui permet une exploitation ultérieure par des outils logiciels sur ordinateur [7].

1.4.3 Caractéristiques de l'image numérique

L'image est une représentation d'un ensemble d'informations formalisé par des paramètres suivants :

1.4.3.1 Pixels :

Généralement on exprime la définition d'une image en indiquant le nombre de pixels répartis sur la largeur et sur la hauteur, par exemple on dira de l'image suivante que sa définition est de 40 par 30, ce que signifie qu'elle possède 40 pixels sur la largeur et 30 sur la hauteur soit 1200 pixels en tout, ce qui représente la taille de l'image.

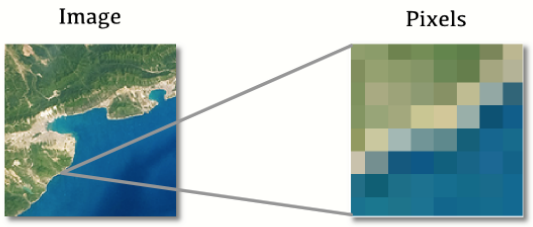

FIGURE 1.2 – Représentation Pixels d'une image [8].

L'information par pixel représenté l'intensité de l'image ou bien le niveau de gris ou la couleur, on a deux types d'images : image monochrome et l'image couleur.
La différence entre elles réside dans la quantité d'informations contenues dans chaque pixel. L'image couleur représente l'intensité sur 3 chaines (3 OCTETS RBV : ROUGE BLEU VERT) par contre l'image monochrome est sur un octet [9].

1.4.3.2 La résolution :

La résolution d'une image est le nombre de pixels par pouce qu'elle contient (1 pouce = 2.54 centimètres). Elle est exprimée en "PPP" (points par pouce). Plus il y a de pixels (ou points) par pouce et plus il y aura d'information dans l'image [10].

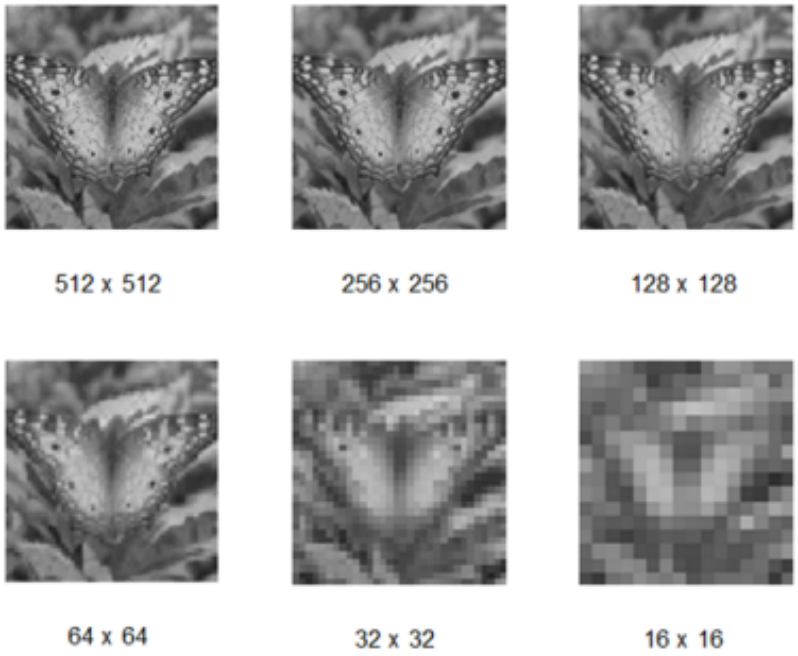

FIGURE 1.3 – Résolution d'une image [8].

1.4.3.3 Voisinage :

La surface d'une image est représentée par l'ensemble des surfaces rectangulaires On distingue deux types de voisinage [10] :

- Voisinage à 4 : pixels qui ont un côté commun avec le pixel centré.
- Voisinage à 8 : tout pixel situé autour du pixel centré.

1.4.3.4 Contraste :

Le contraste est une propriété intrinsèque d'une image qui quantifie la différence de luminosité entre les parties claires et sombres d'une image.

- **Une image contrastée :** présente une bonne dynamique de la distribution des valeurs de gris sur tout l'intervalle des valeurs possibles, avec des blancs bien clairs et des noirs profonds.
- **Une image faible contrastée :** a une faible dynamique, la plupart des pixels ayant des valeurs de gris très proches [9].

FIGURE 1.4 – Contraste d'une image [11].

1.4.3.5 Niveau de gris (Grayscale) :

Une intensité lumineuse d'un pixel situé dans la surface de l'image, la plage de l'intensité entre 0 (noire) et 255 (blanc) elle représente aussi la quantité de la lumière réfléchie pour 8 bits, on dispose 256 niveaux de gris, plus le nombre de bits augmente plus les niveaux sont nombreux.

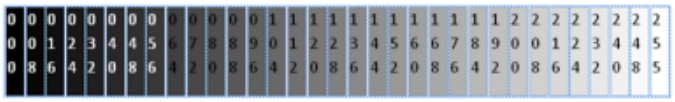

FIGURE 1.5 – Les niveaux de gris (0, 255) [12].

1.4.3.6 Luminance :

La luminance est une mesure photométrique de l'intensité lumineuse par unité de surface de lumière se déplaçant dans une direction donnée, pour une bonne luminance :

- Il faut que l'image soit lumineuse.
- Il faut un bon contraste sa qui veut dire que le contraste d'une image ne tend ni vers le noir ni vers le blanc, pour éviter des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de **bruit** [9].

1.4.3.7 Bruit :

Dans une image, le bruit est toute fluctuation parasite ou dégradation que subit l'image de l'instant de son acquisition jusqu'à son enregistrement à cause de variation de l'intensité d'un pixel par rapport à ses voisins. Ce qui donne certains défauts (petits nuages, poussière et la diminution de l'intensité électrique sur les capteurs), Les sources de bruit sont multiples, certaines sont physiques liées à la qualité de l'éclairage de la scène, et électroniques liées à la stabilité du capteur de l'image durant l'acquise [13].

FIGURE 1.6 – Image bruitée [14].

1.4.3.8 Contour :

Le contour est la frontière qui sépare des objets dans une image qui ont des pixels dont les niveaux de gris différents, ou la limite des objets qui marquant des changements d'intensité [9].

FIGURE 1.7 – Contour d'une image [14].

1.4.3.9 Histogramme :

L'histogramme des niveaux de gris ou des couleurs d'une image est fonction de la fréquence à laquelle chaque niveau de gris (couleur) apparaît dans l'image affichée. Il fournit de nombreuses informations sur la distribution des niveaux de gris (couleurs) et vous pouvez voir dans quelles limites la plupart des niveaux de gris (couleurs) est distribué lorsque l'image est trop sombre [14].

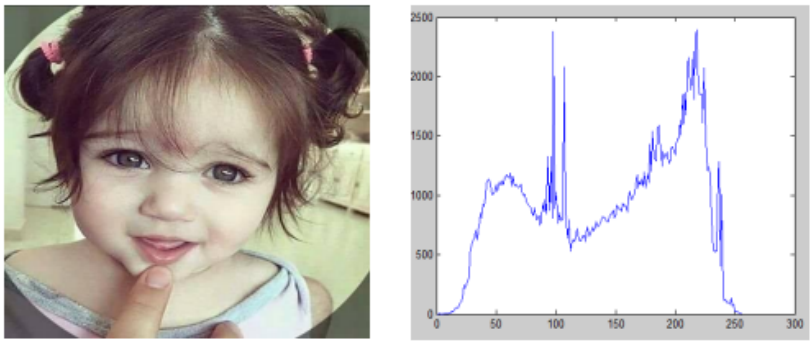

FIGURE 1.8 – Image avec histogramme [14].

1.4.4 Types d'images numériques

On distingue deux types d'images à la composition, et au comportement différent : images matricielles et les images vectorielles.

1.4.4.1 Images matricielles

Une image matricielle, ou « carte de points (de l'anglais bitmap » est composée comme son nom l'indique d'une matrice (tableau) de points à plusieurs dimensions, chaque dimension représente une dimension spatiale (hauteur, largeur), ou autres (niveau de résolution). Dans le cas des images à deux dimensions, les points sont appelés pixels [7].

1.4.4.2 Images vectorielles

Elle est composée de différents objets repérés par leurs coordonnées et comportant différents attributs (bordure, fond, forme, coordonnées). Leur avantage c'est qu'elles peuvent être facilement redimensionnées. Leur codage dépend directement du logiciel qui a permis de les créer [7].

1.4.5 Codages des couleurs

Nous avons vu qu'une image apparaît comme une matrice où chaque case contient des nombres associés à une couleur. Usuellement on distingue 3 grands types de couleurs pour une image numérique :

- Le noir et blanc.
- Les niveaux de gris
- La couleur.

Ces types sont généralement à choisir lors d'une numérisation par scanner ou lors de la configuration d'un appareil photographique .

1.4.5.1 Image noir et blanc (binaire)

Le noir et blanc est le plus simple. Le contenu de chaque case de la matrice est soit un 0 (noir) soit 1 (blanc). Le nombre de couleurs n'est que de 2 et le rendu de l'image le moins performant mais parfois suffisant dans le cadre par exemple de documents scripturaux (Figure 1.9) [13].

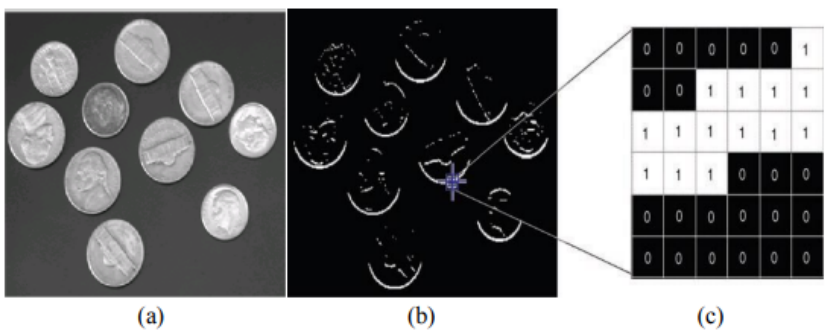

FIGURE 1.9 – Image binaire.
a) Image originale, b) Une image binaire, c) Le tableau de valeurs correspondant [15].

1.4.5.2 Niveaux de gris

Le codage dit en niveaux de gris permet d'obtenir plus de nuances que le simple noir et blanc. Il offre des possibilités supplémentaires pour coder le niveau de l'intensité lumineuse. La couleur est codée souvent sur un octet soit 8 bits ce qui offre la possibilité d'obtenir 256 niveaux de gris (0 pour le noir et 255 pour le blanc). On peut aussi le faire avec 16 niveaux de gris (4 bits) (Figure 1.10) [13].

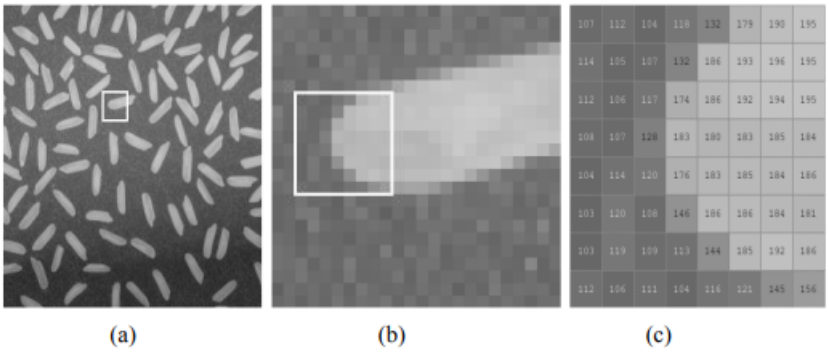

FIGURE 1.10 – Quantification des niveaux de gris.
a) Une image en niveaux de gris, b) Agrandissement d'une zone de l'image ,c) Affichage des valeurs constituant la matrice image [15].

1.4.5.3 Image couleur

Principe : La couleur d'un pixel est obtenue, comme le ferait un peintre, par le mélange de couleurs fondamentales. Il ne s'agit pas ici de décrire toutes les techniques utilisées. Nous allons décrire un des principes les plus couramment utilisés qui est celui de la synthèse additive [13].

Codage RVB : Le principe consiste à mélanger les 3 couleurs : rouge, vert et bleu (noté RVB ou RGB en anglais). À l'aide de ces 3 couleurs, on obtient toute une palette de nuances allant du noir au blanc. À chaque couleur est associé un octet (donc 256 niveaux de luminosité) de chacune des couleurs fondamentales (Figure 1.11) [13].

Rouge	Vert	Bleu	Couleur
0	0	0	Noir
0	0	1	Nuance de noir
255	0	0	Rouge
0	255	0	Vert
0	0	255	Bleu
128	128	128	Gris
255	255	255	Blanc

FIGURE 1.11 – Principe codage de la couleur [13].

Un pixel "couleur" est alors codé avec 3 octets et on a alors la possibilité d'obtenir 224 possibilités de couleurs soit de l'ordre de 16 millions de couleurs différentes (Figure 1.12) [13].

FIGURE 1.12 – Exemple d'une image en couleur [15].

1.4.6 Traitement d'images numérique

Le traitement d'image est une discipline issue de deux domaines qui se croisent et qui sont l'informatique et les mathématiques appliquées, cette dernière étudie les caractéristiques et les propriétés des images numériques et leurs transformations dans le but soient d'améliorer leur qualité ou d'en extraire de l'information. Dans ce qui va suivre, nous allons présenter trois opérations de traitement d'image visant à améliorer la qualité ou extraire de l'information de cette dernière [4].

1.4.6.1 Acquisition

L'acquisition d'images constitue un des maillons essentiels de toute chaîne de conception et de production d'images. Pour pouvoir manipuler une image sur un système informatique, il est avant tout nécessaire de lui faire subir une transformation qui la rendra lisible et manipulable par ce système. Cette opération se fait par des systèmes de saisie qui peuvent être classés en deux catégories principales : les caméras numériques et les scanners [16].

1.4.6.2 Segmentation

La segmentation consiste en la répartition de l'ensemble des pixels de l'image en groupe, chacun d'eux formera une région de l'image selon un critère d'homogénéité (chaque

région est caractérisée par un critère d'homogénéité). Cette technique de traitement d'image vise à séparer le plus précisément possible les différents objets se trouvant dans l'image traitée et ainsi extraire l'information présente dans cette dernière. Il existe plusieurs méthodes et approches pour la segmentation d'images [4].

FIGURE 1.13 – Illustration de la segmentation d'images (figure1 : image d'origine, figure2 : l'image segmentée) [4].

1.4.6.3 Filtrage

Le principe du filtrage est de modifier la valeur des pixels d'une image, généralement dans le but d'améliorer son aspect. En pratique, il s'agit de créer une nouvelle image en se servant des valeurs des pixels de l'image d'origine. Il existe plusieurs filtres effectuant différents traitements selon le besoin de l'analyse, on cite l'exemple de filtrage d'une image bruité par un filtre opérant de telle sorte à supprimer les pixels bruits présents dans l'image [4].

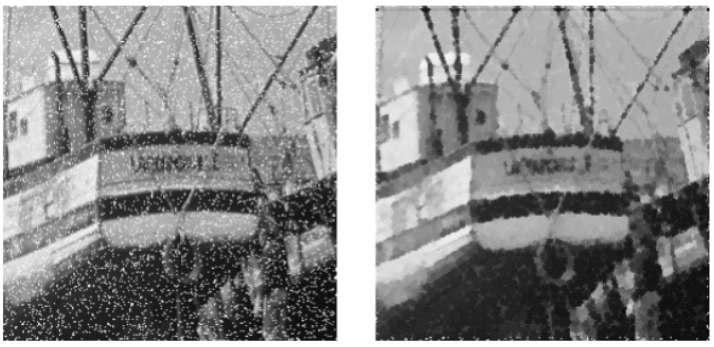

FIGURE 1.14 – Exemple de filtrage d'une image bruitée (à gauche l'image d'origine, à droite l'image filtrée) [4].

1.4.7 Domaines D'application

Le traitement d'images possède l'aspect multidisciplinaire. On trouve ses applications dans des domaines très variés tels que :

1.4.7.1 Imagerie aérienne et spatiale

Dans laquelle les traitements concernent l'étude des images satellites, l'analyse des ressources terrestres, la cartographie automatique, les analyses météorologiques [13].

1.4.7.2 Technologies biomédicales

Nous trouvons des utilisations de cette technique dans l'échographie, la résonance magnétique nucléaire, ainsi que dans le domaine de la reconnaissance automatique des cellules ou des chromosomes [13].

1.4.7.3 La robotique

Qui connaît actuellement le plus grand développement et dont les tâches usant de l'imagerie sont principalement l'assemblage (pièce mécanique, composants électroniques,...), le contrôle de qualité, ainsi que la robotique mobile [13].

1.4.7.4 La télésurveillance

Exemple, radar automatique : recherche en temps réel d'un véhicule par reconnaissance de son immatriculation parmi un flot de véhicules circulant sur le boulevard périphérique par caméra fixe [13].

1.5 Stéganographie

1.5.1 Définition

Le mot stéganographie vient du mot grec "steganos" qui veut dire "couvert" et du mot "graphe in" pour "écriture".
La stéganographie (Steganography) est l'art de faire passer de l'information en dissimulant son existence même. Le but de la stéganographie est d'éviter d'attirer l'attention sur la transmission d'un message caché.
On rend anonyme l'information utile dans un flot d'informations anodines. Si des soupçons existent, alors l'objectif n'est pas atteint [17].

1.5.2 Types de stéganographie

On distingue deux types de stéganographie :

1.5.2.1 La stéganographie linguistique

On peut définir la stéganographie linguistique tout simplement comme étant toute forme de stéganographie qui emploie la langue dans le processus de dissimulation.
L'objectif de la stéganographie linguistique consiste à dissimuler un message sous une lettre anodine ou dans la disposition d'un objet qui n'éveille pas les soupçons afin de ne pas attirer les regards de potentiels observateurs.
Différentes méthodes linguistiques, pour cacher un message dans un texte, ont peut jouer sur l'espace entre les mots, la ponctuation, ou encore l'orthographe.
A l'origine, c'est l'acrostiche qui permettait de cacher ces messages. L'acrostiche est un poème dont la première lettre de chaque vers compose un mot ou une phrase.
Mais il existe également des techniques plus évoluées où, pour pouvoir comprendre le message caché, il faut savoir quels mots ou quelles lettres lire.
Par exemple, ci-dessous, un texte à première vue innocent envoyé par un espion allemand pendant la Seconde Guerre mondiale renfermait un message très important.
" A**P**PARENTLY N**E**UTRAL'S P**R**OTEST I**S** T**H**OROUGHLY D**I**SCOUNTED A**N**D IGNORED. I**S**MAN H**A**RD IT. B**L**OCKADE I**S**SUE A**F**FECTS P**R**ETEXT F**O**R EM**B**ARGO O**N** B**Y**PRODUCTS, E**J**ECTING S**U**ETS A**N**D V**E**GETABLE O**I**LS. "
Dans le cas présent, en prenant la deuxième lettre de chaque mot, on voit apparaitre le message suivant : **Pershing sails from NY June 1.**[17].

1.5.2.2 La stéganographie technique

Elle intéresse le plus les informaticiens, ce type de stéganographie est défini comme étant l'art et la science de "cacher" une information privée ou secrète dans un support numérique apparemment anodin. La caractéristique principale est que le support doit sembler ne contenir aucune information sensible, c.à.d ne renfermer aucune information secrète comme le montre.
Le support associé à son message porte le nom de "stégo-médium" [17].

1.5.3 Les types de support

1.5.3.1 Le texte

On peut cacher un message, non pas en utilisant le support lui-même, mais le texte lui-même. Ainsi, une personne confrontée à une lettre dont l'auteur a fait usage de stéganographie, mais ne le sachant pas, ne peut intuitivement s'en douter. La dissimulation d'information dans un support de type texte est généralement associé à la stéganographie linguistique [17].

1.5.3.2 L'audio

La dissimulation d'informations dans des signaux audio est basée sur le fait que l'oreille humaine ne peut percevoir qu'une certaine gamme de fréquences, et n'est pas capable de percevoir un faible changement d'amplitude du signal. Cependant, elle ne peut détecter une faible distorsion qu'aurait entraîné une modification du signal.
Le support audio est le moins utilisé pour la dissimulation d'informations ce qui est du au manque de logiciels qui sont dédiés pour support du type audio, de même ce type de format ne permet pas de dissimuler une grande quantité d'information [17].

1.5.3.3 La vidéo

Plusieurs techniques existent pour dissimuler une information dans un support du type vidéo, mais sont pour la plupart des améliorations ou modifications de la technique de la transformée en cosinus discret (DCT). Celle-ci utilise la quantification des parties les moins importantes des images (arrondies aux valeurs supérieures par exemple). L'oeil était relativement peu sensible aux hautes fréquences, la DCT pourra appliquer des modifications plus importantes dans cette plage, sans pour autant créer de modifications visibles dans l'image [17].

1.5.3.4 L'image

La dissimulation d'information dans les supports de type image est la plus répondue et la plus sollicitée, mais pourquoi la stéganographie image est la technique la plus populaire de nos jours ? Le support de type image est le support populaire. À la base, il s'agit d'exploiter les capacités limitées du système visuel humain afin de dissimuler des données dans une image. Ces données sont généralement du texte ou d'autres images, mais on peut dissimuler tout ce qui peut être codé en binaire [17].

1.5.4 Structure d'une communication secrete

Le processus complet de dissimulation d'informations repose sur deux processus :

1. **la dissimulation** qui consiste à insérer une information dans le medium.
2. **l'extraction** qui consiste à restituer l'information cachée.

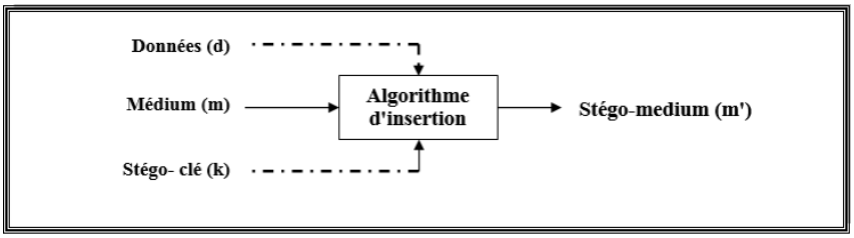

FIGURE 1.15 – Schéma de dissimulation des données dans un médium [17].

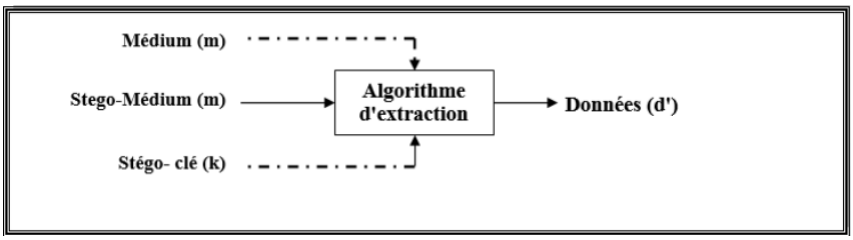

FIGURE 1.16 – Schéma d'extraction des données d'un médium [17].

1.5.5 Classification des schémas de stéganographie

1.5.5.1 Stéganographie pure

La stéganographie pure est le processus d'incorporation des données dans l'objet sans utiliser de clés privées. Ce type de stéganographie dépend entièrement du secret.

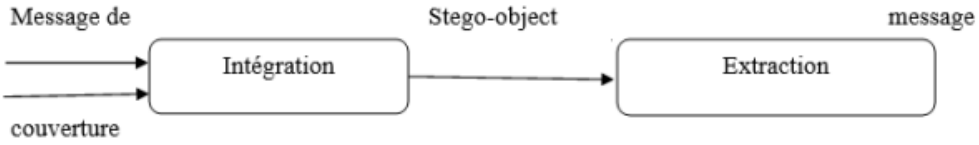

FIGURE 1.17 – Processus de stéganographie pure [18].

Ce type de stéganographie ne peut pas fournir la meilleure sécurité parce qu'il est facile pour extraire le message si la personne non autorisée connaît la méthode d'incorporation. Il a un avantage à réduire la difficulté de partage des clés [18].

1.5.5.2 Stéganographie à clé secrète

La stéganographie clé secrète est un autre procédé de stéganographie qui utilise la même procédure autre que l'utilisation de clés sécurisées. Il utilise la clé individuelle pour incorporer les données dans l'objet qui est similaire à la clé symétrique. Pour le décryptage il utilise la même clé qui est utilisée pour le cryptage.

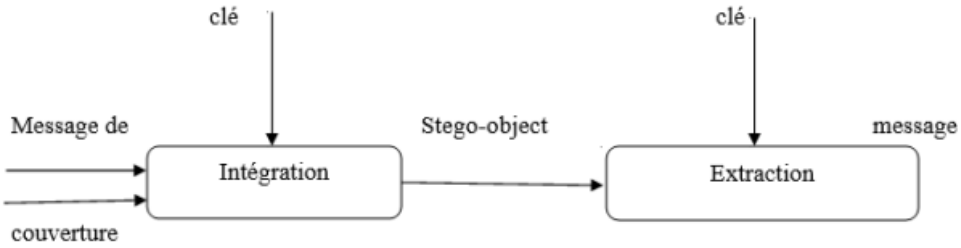

FIGURE 1.18 – Stéganographie à clé secrète [18].

Ce type de stéganographie offre une meilleure sécurité par rapport à la stéganographie pure. Le problème principal de l'utilisation de ce type de système stéganographie est le

partage de la clé secrète. Si l'attaquant connaît la clé, il sera plus facile de déchiffrer et d'accéder à l'information originale [18].

1.5.5.3 Stéganographie à clé publique

La stéganographie à clé publique utilise deux types de clés : une pour le chiffrement et une autre pour le décryptage. La clé utilisée pour le décryptage est une clé privée et pour le cryptage, c'est une "clé publique" et est stockée dans une base de données publique [18].

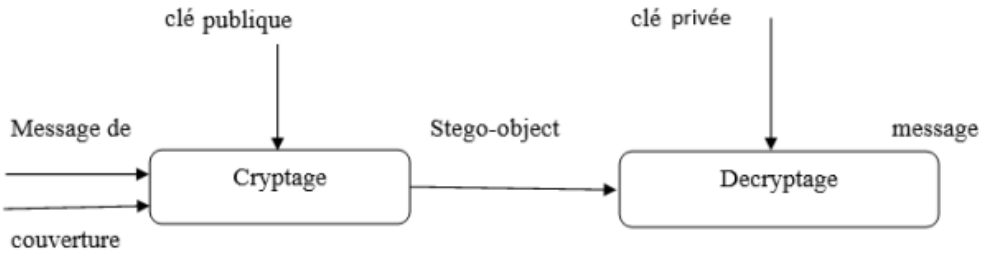

FIGURE 1.19 – Stéganographie à clé publique [18].

1.5.6 Efficacité d'un systéme stéganographique

- Le support doit être utilisé qu'une seule fois et détruit après utilisation, afin d'éviter les attaques par différence. En effet, tout attaquant possédant le support original est capable avec une probabilité égale à 1 de détecter tout Stégo-médium issu dû support.
- Pour éviter les attaques visuelles, l'algorithme de stéganographie ne doit pas détériorer visuellement le support.
- La quantité d'informations à dissimuler doit être petite Intuitivement ,plus ont dissimule d'information dans un support, plus celui-ci subit de changements et plus le Stégo-médium risqué d'être détecté [17].

1.5.7 Caractéristique d'un processus de stéganographie

La stéganographie possède trois grandes caractéristiques qui dirigent son utilisation : l'indéctectabilité, la capacité et la robustesse.
Lors de la mise en place d'un projet intégrant l'utilisation de la stéganographie, un contrôle de la qualité et de la pertinence du projet vis -à-vis des objectifs de sécurité doit être envisagée et il peut reposer sur ces trois points :

1.5.7.1 l'indétectabilité

Une fois intégrées dans le stego-medium les données ne doivent pas être perceptibles. Il s'agit de ne pas détériorer le support et de rendre la donnée cachée indétectable à une personne surveillant le canal de transmission du message.
Ce concept est lié aux propriétés des systèmes visuels et auditifs humains.
Afin de garantir l'imperceptibilité on doit se poser la question qui suit :
"Les données dissimulées peuvent elles être détectées par l'homme ?".[17].

1.5.7.2 La capacité

C'est la quantité de bits significatifs dissimulés dans le stégomédium par unité d'accès ; c'est-à-dire prévoir la quantité de bits significatifs pouvant être utilisées. Pour savoir si un support peut-il être un bon dissimulateur de données on doit connaitre la réponse de la question suivante :
"Quelle quantité de données le support peut-il dissimuler ?".[17].

1.5.7.3 La robustesse

Ou l'aptitude de préservation des données cachées face aux modifications appliquées au stego-medium, condition absolue pour garantir la discrétion de la transmission de la donnée cachée, c'est à dire le fondement même du projet de sécurité. En d'autres termes pour connaître le degré de robustesse, l'utilisateur doit se poser la question suivante :
" Le message résiste-t-il à des modifications du support (redimensionnement, compression) ?".[17].

1.5.8 Classification des techniques des systèmes steganographies

Plusieurs approches sont possibles pour classifier les techniques de stéganographie. Celles-ci sont en fonction du type de couvertures, du type d'algorithmes ou du schéma d'insertion.

1.5.8.1 Substitution

Les méthodes classées dans ce groupe remplacent les parties redondantes de la couverture par le message. Les algorithmes sont simples à mettre en oeuvre, mais sont vulnérables à des modifications les plus simples. Parmi les nombreuses méthodes de substitution, nous citons le remplacement du LSB-1, bit de poids le plus faible, par le bit du message [17].

1.5.8.2 Transformation de domaines

Les procédés insèrent le message dans des espaces transformés comme DCT, ou DWT Cette stratégie rend le message plus robuste aux attaques parce que l'information est

dissimulée dans la portion la plus significative du signal.
La DCT appliquée aux images est devenue très populaire après la création de la norme JPEG. Par conséquence, beaucoup de stratégies d'insertion de données cachées ont été développées pour ce format d'image [17].

1.5.8.3 Distorsion

L'idée est de cacher le message en effectuant une distorsion du signal original. L'extraction du message est faite par la différence entre l'image porteuse et la couverture. Cette catégorie n'est pas adaptée à la stéganographie à cause de la quantité importante de distorsions apportées à l'image et la nécessité d'avoir l'image originale lors de l'extraction [17].

1.5.9 Stéganalyse

Malheureusement le bienveillant peut aussi cacher une certaine malveillance. Du moment que la stéganographie est la science permettant la dissimulation des données pour la bonne cause, elle peut aussi être usée par des malintentionnés en vue de s'en servir pour des fins destructives.
Pour cette raison la présence des méthodes détectant la stéganographie devient une nécessité. La science qui s'intéresse aux méthodes de détection de la stéganographie s'appelle la stéganalyse et qui évolue en fonction des nouvelles techniques stéganographies [17].

1.5.9.1 Description de la stéganalyse

Contrairement au but du "Data- hiding", la stéganalyse consiste à attaquer les méthodes stéganographies par détection, destruction, extraction ou modification des données encapsulées.
La stéganalyse consiste à identifier la présence d'un message. Seule cette présence doit être déterminée. Dans un second temps, le message lui-même pourra être retrouvé, quoique ceci ne soit pas l'objectif principal de la stéganalyse [17].

1.5.9.2 Processus de la stéganalyse

Le processus de stéganalyse s'effectue en trois étapes :

— **La détection :** consiste à établir l'existence d'un message caché, sans que l'on sache, à ce stade, quel est ce message.
— **L'extraction :** consiste à isoler l'information cachée. Cette étape peut nécessiter, outre l'emploi de techniques de stéganalyse, l'usage de techniques de cryptanalyse

pour donner un sens au message caché préalablement extrait, mais rendu a priori inintelligible via l'utilisation de méthodes de cryptage.

— **Le filtrage :** consiste à détruire le message caché, sans nécessairement pouvoir détecter son existence, ou déchiffrer son contenu, mais également sans que le récipient ne soit affecté de manière décelable (soit par un utilisateur humain, ou soit par une machine) [17].

1.5.9.3 Types de stéganalyse

Il existe deux catégories permettant de classer la stéganalyse :

— **La stéganalyse active :** Dans ce type d'attaque on souhaite non seulement détecter le message caché mais, en plus, on va chercher à extraire, modifier ou supprimer ces données. ou en tout cas, à le rendre inutilisable. Cette destruction aura souvent lieu par l'intermédiaire de modifications du support (redimensionnement, compression,...).

— **La stéganalyse passive :** il s'agit simplement ici de détecter la présence de données dissimulées. On peut par exemple imaginer que ce type d'attaque soit utilisé pour voir si des données stéganographiées sortent d'une entreprise et prévenir les risques encourus par ces fuites [17].

1.6 Conclusion

Dans ce chapitre, on a essayé de faire un récapitulatif sur les notions élémentaires de vision par ordinateur et de traitement d'image, et nous avons présenté les notions de l'image numérique et ses différentes caractéristiques (pixels, résolution, voisinage,. . .etc.) ainsi que le codage des couleurs, enfin on a expliqué la technique de dissimulation d'informations (stéganographie).
Dans ce qui suit, nous allons parler sur le domaine d'apprentissage machines et ses caractéristiques principales et nous avons introduit ses composants.

Chapitre 2

Apprentissage machines

2.1 Introduction

Ce chapitre contient des généralités relatives à l'apprentissage automatique (machine learning en anglais). nous définissons l'apprentissage automatique, son objectif . Ensuite, nous présentons les principaux types de systèmes d'apprentissage et types de modèles, et enfin une brève description sur les algorithmes l'apprentissage.

2.2 Apprentissage machines

2.2.1 Définition

L'apprentissage automatique (" machine learning ") est une méthode utilisée en intelligence artificielle. Il s'agit des techniques qui analysent un ensemble de données afin de déduire des règles qui constituent de nouvelles connaissances permettant d'analyser de nouvelles situations. C'est une technique qui sort un "modéle" à partir des "données" pouvant être selon plusieurs formats (des images, des sons, des vidéos, des valeurs numériques ou des signaux). Le concept de "l'apprentissage" apparaît dans le fait que la technique analyse les données en entrée et trouve le modèle par soi-même au lieu d'avoir un humain pour le lui faire. On l'appelle apprentissage car la procédure ressemble à être entrainé avec les données (appelées données d'entrainement ou Training Data en anglais) pour résoudre le problème de trouver un modèle, qui à son tour sera utilisé pour traiter d'autres données jamais vues avant. La figure suivante schématise le processus de la machine learning [19].

FIGURE 2.1 – Schèma général du Machine Learning [19].

2.2.2 Modélisation

L'apprentissage automatique d'une machine concerne toujours un ensemble de tâches concrètes - T. Pour déterminer la performance de la machine, on utilise une mesure de la performance P. La machine peut avoir à l'avance un ensemble d'expérience E ou elle va enrichir cet ensemble plus tard.

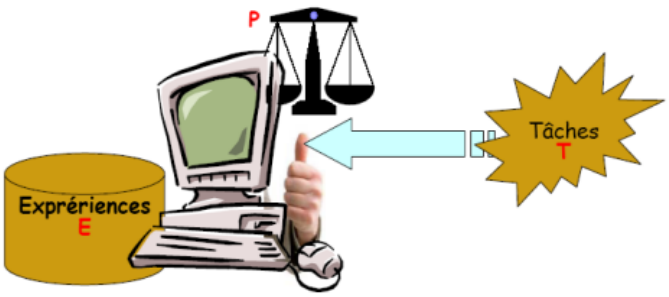

FIGURE 2.2 – Schéma de modélisation d'une machine d'apprentissage [20].

Donc, l'apprentissage automatique pour la machine est qu'avec l'ensemble de tâches T que la machine doit réaliser, elle utilise l'ensemble d'expériences E telle que sa performance sur T est améliorée [20].

2.2.3 Objectif

L'apprentissage machine est une sous discipline de l'intelligence artificielle dont l'objectif ultime est de reproduire chez l'ordinateur les capacités cognitives de l'être humain, par le biais de l'apprentissage. Ici, le terme apprentissage est à mettre en opposition avec des techniques d'intelligence artificielle basées sur des comportements intelligents "précodés", comme dans le fameux programme ELIZA [21], où l'ordinateur donnait l'illusion de pouvoir poursuivre une conversation intelligente avec un humain à partir d'un système en fait très basique de détection de mots-clés et de réponses prédéfinie. L'approche "apprentissage machine" consiste plutôt à programmer des mécanismes qui permettent de développer les connaissances c'est-à-dire d'apprendre à partir d'observations (les exemples d'apprentissage) de manière automatique [22].

Les êtres humains faisant preuve de capacités d'apprentissage impressionnants, il est naturel que l'apprentissage machine s'inspire d'eux pour tenter de reproduire leurs comportements. En particulier, les travaux en neurosciences visant à comprendre les mécanismes d'apprentissage dans le cerveau. Une classe d'algorithmes d'apprentissage très populaire, les réseaux de neurones, a ainsi été inspirée de telles observations biologiques. Mais les détails du fonctionnement du cerveau restent pour l'instant en grande partie un mystère. Les algorithmes développés en apprentissage machine ont des objectifs moins ambitieux. Ils ont chacun leurs propres forces et faiblesses – souvent très différentes de celles des humains – et sont généralement prévus pour résoudre certains types de problèmes spécifiques [22].

2.3 Types de système d'apprentissage

Il existe plusieurs types de systèmes d'apprentissage et cela varie en fonction du type de problème que l'on se pose. Il est alors utile de les classer en différentes catégories. Les systèmes de machine learning peuvent être classés en fonction de l'importance et de la nature de la supervision qu'ils requièrent durant la phase d'entraînement. On distingue alors quatre grandes catégories : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage avec renforcement.

2.3.1 Apprentissage supervisé

L'apprentissage supervisé consiste en la conception d'un modèle reliant des données d'apprentissage à un ensemble de valeurs de sortie. C'est-à-dire que les données d'entraînement qu'on fournit à l'algorithme comportent les solutions désirées, appelées étiquettes (en anglais, labels). Cette méthode permet donc à l'algorithme d'apprendre en comparant sa sortie réelle avec les sorties enseignées, afin de trouver les erreurs et modifier le modèle en conséquence. L'apprentissage supervisé confère au modèle la possibilité de prédire des valeurs d'étiquette sur des données non étiquetées supplémentaires.
Soit D un ensemble de données, décrit par un ensemble de caractéristiques X, un algorithme d'apprentissage supervisé va trouver une fonction de mapping entre les variables prédictives en entrée X et la variable à prédire Y. La fonction de mapping décrivant la relation entre X et Y s'appelle un modèle de prédiction. Les caractéristiques X peuvent être des valeurs numériques, alphanumériques ou des images.
Un exemple d'utilisation de l'apprentissage supervisé est le filtre antispam, l'apprentissage s'effectue à l'aide de nombreux exemples d'e-mails qu'on a étiqueté spam ou normal. A partir de cela, le filtre doit alors être capable de classer de nouveaux e-mails.
Un autre exemple consiste à prédire le prix d'une voiture à partir des valeurs d'un certain nombre d'attributs ou variables qu'on appelle caractéristiques d'une observation ou

features en anglais. Ces variables peuvent être le kilométrage, l'âge, la marque, etc. Ils sont également appelés variables explicatives ou prédictives. L'entrainement se fait alors à partir de ces variables et des étiquettes. La catégorie de la variable prédite Y fait décliner l'apprentissage supervisé en deux sous-catégories : la classification et la régression, ces deux concepts seront abordés plus tard [23].

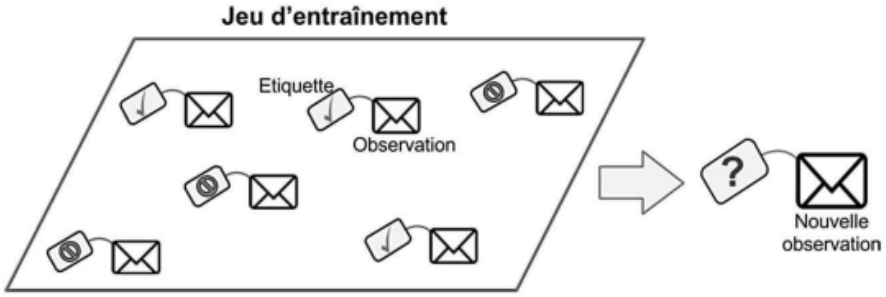

FIGURE 2.3 – Un jeu d'entraînement étiqueté pour un apprentissage supervisé [23].

2.3.2 Apprentissage non supervisé

L'apprentissage non supervisé consiste en la conception d'un modèle structurant l'information, c'est-à-dire les données d'apprentissage ne sont pas étiquetées. Cette méthode permet donc à l'algorithme de trouver tout seul des points communs parmi les données d'entrée, le système apprend alors sans professeur.
Comme l'étiquetage de données requiert beaucoup de temps, les méthodes d'apprentissage utilisant l'apprentissage non supervisé sont particulièrement utiles. L'apprentissage non supervisé peut-être utilisé pour la réduction de dimension ou l'extraction de variable. Cette tâche consiste à simplifier les données sans perdre trop d'informations, il pourra ensuite être fourni à un autre algorithme d'apprentissage automatique (tel qu'un algorithme d'apprentissage supervisé). Le kilométrage d'une voiture, par exemple, peut être fortement corrélé à son âge, de sorte que l'algorithme de réduction de dimension les combine en une seule variable représentant la vétusté de la voiture.
A première vue, on pourrait penser que l'apprentissage non supervisé a peu d'utilité dans les applications de la vraie vie, mais les applications de cette technique sont nombreuses. Les sites comme Amazon, Netflix ou encore Youtube utilisent les algorithmes de partitionnement ou Clustering en anglais pour faire des recommandations de produits ou de films. Il peut être également utilisé pour explorer de larges ensembles de données et de découvrir d'intéressantes relations entre les variables : pour un supermarché par exemple, exécuter une règle d'association sur les journaux de vente permettrait peut-être de découvrir que les personnes achetant de la sauce barbecue et des chips ont aussi tendance à acheter des grillades. Cela permettrait de réorganiser les rayons afin de présenter ces articles à proximité les uns des autres [23].

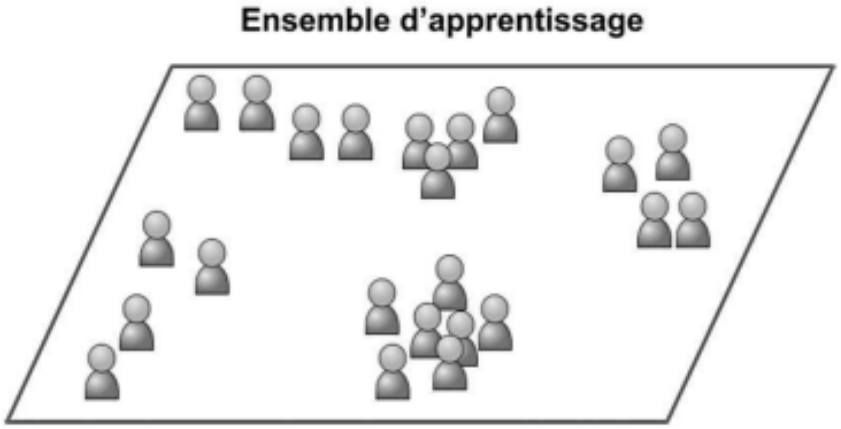

FIGURE 2.4 – Un jeu d'entraînement non étiqueté pour un apprentissage non supervisé [23].

2.3.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé vise à résoudre les problèmes avec relativement peu de données étiquetées et une grande quantité de données non étiquetées. L'apprentissage semi-supervisé réduit également le temps d'étiquetage de grandes quantités de données par rapport à un apprentissage supervisé. Il a été démontré que l'utilisation de données non étiquetées, en combinaison avec des données étiquetées, permet d'améliorer significativement la qualité de l'apprentissage. Ce type d'apprentissage a pour objectif de classer certaines des données non étiquetées à l'aide de l'ensemble d'informations étiquetées.
Un exemple illustrant l'utilisation d'un apprentissage semi-supervisé est le service d'hébergement d'image Google Photos. Une fois avoir téléchargé des photos de famille sur ce service, le système arrive à reconnaître qu'une personne A apparaît sur telle ou telles photos et qu'une personne B sur telle autre. Cela est dû à la partie non supervisée de l'algorithme. Une fois que vous avez identifié ces personnes, juste une étiquette par personne, le système sera capable de nommer les personnes figurant sur chaque photo, ce qui est utile pour des recherches ultérieures [23].

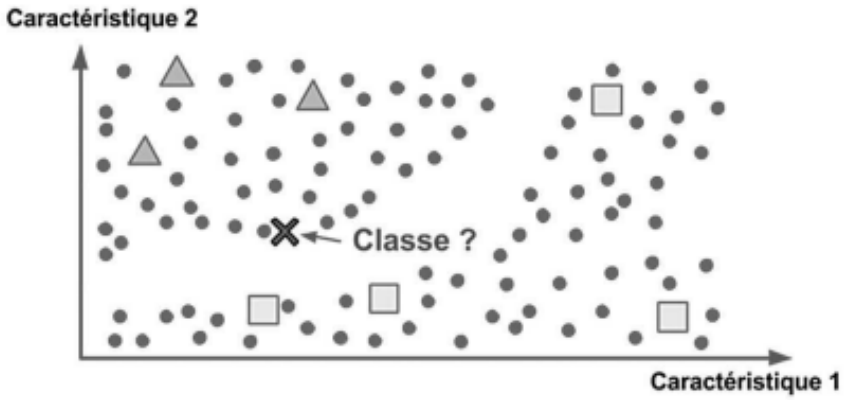

FIGURE 2.5 – Apprentissage semi-supervisé [23].

2.3.4 Apprentissage avec renforcement

L'apprentissage par renforcement est très différent des types d'apprentissages vus jusqu'ici. Il consiste à apprendre, à partir d'expériences successives, ce qu'il convient de faire de façon à trouver la meilleure solution. Le système d'apprentissage, que l'on appelle ici « agent », interagit avec l'environnement, en sélectionnant et accomplissant des actions afin de trouver la solution optimale et obtenir en retour des récompenses. L'agent essaie plusieurs solutions, on parle d'« exploration », observe la réaction de l'environnement et adapte son comportement, c'est-à-dire les variables pour trouver la meilleure stratégie. Pour ce type d'apprentissage, les données d'entraînement proviennent directement de l'environnement.

L'apprentissage par renforcement peut être utilisé pour apprendre à un robot à marcher, ou à un programme à jouer, comme AlphaGo de DeepMind qui a vaincu l'un des meilleurs joueurs de go au niveau mondial. En 2018, une startup issue des travaux de l'université de Cambridge a formé une intelligence artificielle à la conduite automobile. Au bout de seulement vingt minutes, l'IA est parvenue à savoir comment maintenir la voiture sur sa voie de circulation. Durant cette expérience un chauffeur était présent à bord de la voiture pour corriger les écarts de trajectoire causés par le logiciel, en arrêtant la voiture. Le programme a rapidement progressé en suivant une logique pénalité-récompense, en l'occurrence, respectivement, une intervention humaine et la distance maximale parcourue sans correction par le conducteur [23].

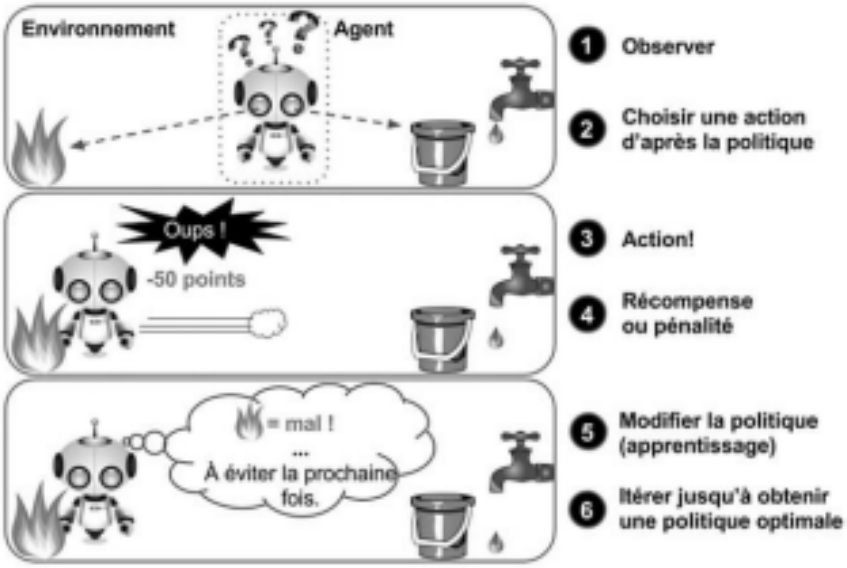

FIGURE 2.6 – Apprentissage par renforcement [23].

2.4 Généralisation

2.4.1 Sur-apprentissage

L'entraînement d'un algorithme d'apprentissage consiste à extraire, de manière explicite ou implicite, des caractéristiques de la distribution de probabilité P qui génère les données. Mais P étant inconnu, l'algorithme se base à la place sur un nombre fini d'exemples d'entraînement, c'est-à-dire. Sur la distribution discrète P des exemples disponibles dans D (appelée la distribution empirique). Lorsque certaines caractéristiques apprises sur P ne s'appliquent pas à P, on parle de sur-apprentissage, et on risque une mauvaise généralisation, c'est-à-dire. Que l'algorithme ne va pas obtenir une bonne performance sûre de nouveaux exemples tirés de P. Prenons l'exemple de la classification, lorsqu'un modèle estimé $P(y \mid x)$ par une fonction $q_x(y)$. Afin de mesurer la similarité entre q_x et $P(. \mid x)$, on peut par exemple utiliser la divergence de Kullback-Leibler DKL[24], définie par [22] :

$$D_{KL}\left(P(x) \| q_x\right) = \sum_y P(x) \ln \frac{P(x)}{q_x(y)} \tag{2.1}$$

Cette quantité est toujours supérieure ou égal à zéro, et est égale à zéro si et seulement si q_x est égale à P(x). La minimisation de la divergence de Kullback-Leibler est donc un critère raisonnable pour obtenir une fonction q_x qui approxime $P(. \mid x)$. Vu que le but est d'obtenir une bonne approximation pour toutes les valeurs de x qui pourraient être

générées par P, il est logique de considérer la minimisation du critère.

$$C(q) = E_X \left[D_{KL} \left(P(x) \| q_x \right) \right] = \int_x \sum_y P(x) P(x) \ln \ln \frac{P(x)}{q_x(y)} dx \tag{2.2}$$

C(q) Est ici ce que l'on appelle l'erreur de généralisation, c'est-à-dire. L'erreur moyenne sur des exemples tirés de P. Puisque P est inconnu, on minimise en pratique un critère $\hat{C}$ défini de la même manière en remplaçant P par $\hat{P}$. C'est le principe de minimisation du risque empirique [25], et $\hat{C}$ s'écrit ici [22] :

$$\hat{C}(q) = \sum_{i=1}^{n} \frac{1}{n} \ln \ln \frac{1}{q_{x_i}(y_i)} = -\frac{1}{n} \sum_{i=1}^{n} \frac{1}{n} \ln \ln q_{x_i}(y_i) \tag{2.3}$$

Qui est appelée la log-vraisemblance négative (en anglais NLL, pour "Negative Log-Likelihood"). Ce critère est minimisé dès que $q_x(y_i) = 1$, pour tous les exemples d'entraînement $(x_i, y_i) \in D$ et ce quelle que soit la valeur de $q_x(y)$ pour des valeurs de x non observées dans D [22].

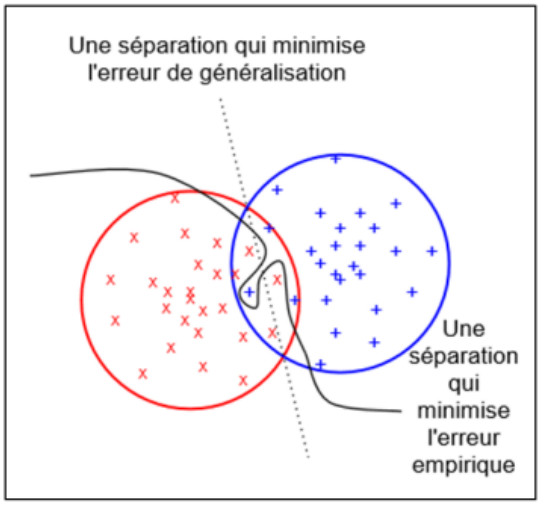

FIGURE 2.7 – Situation de sur-apprentissage [22].

Une fonction q peut donc minimiser $\hat{C}(q)$ sans nécessairement minimiser $C(q)$, si c'est le cas, on est en situation de sur-apprentissage, illustrée en figure 2.7. Pour une distribution fixe des données, deux facteurs principaux augmentent le risque de sur-apprentissage [22] :

- Le manque d'exemples d'entraînement : moins il y a d'exemples, plus il existe de fonctions minimisant le critère $\hat{C}(q)$ (éq. 3), parmi lesquelles seulement un petit nombre seront vraiment proches de la "vraie" solution au problème.
- Pas assez de contraintes dans la forme de la fonction q : moins la classe de fonctions à laquelle q appartient est restreinte, plus l'algorithme d'apprentissage risque de tirer parti de la flexibilité de q pour apprendre des "détails" des exemples d'entraînement, qui ne se généralisent pas à la vraie distribution P. C'est le cas par

exemple dans la figure 2.7 où la séparation tarabiscotée des exemples par la ligne pleine permet de minimiser l'erreur empirique, mais va mener à une plus grande erreur de généralisation qu'une simple ligne droite [22].

2.4.2 Régularisation

Un moyen de lutter contre le sur-apprentissage est d'utiliser une technique dite de régularisation. Il existe plusieurs méthodes de régularisation, mais elles partagent le même principe : rajouter au processus d'apprentissage des contraintes qui, si elles sont appropriées, vont améliorer les capacités de généralisation de la solution obtenue.
Reprenons par exemple le cas de la classification, où l'on cherche à minimiser le critère $C(q)$ (éq. 2), que l'on approxime par $\hat{C}(q)$(éq. 3). Comme nous venons de le voir, ce problème est mal défini car il existe une infinité de fonctions qui minimisent C sans donner aucune garantie sur la valeur de C. Une première façon de régulariser le problème est donc de restreindre la forme de q : par exemple $x \in R^d$ et $y \in \{0, 1\}$ on peut se limiter aux fonctions de la forme [22].

$$q_x(1) = \frac{1}{1 + e^{-w^V x}} \tag{2.4}$$

Où $w \in R^d$ est le vecteur de paramètres du modèle. Notons que si les x_i de l'ensemble d'entraînement sont linéairement indépendants, alors cette contrainte sur la forme de q n'est pas suffisante, puisqu'il est toujours possible que $\hat{C}(q)$ soit arbitrairement proche de zéro sans pour autant avoir de garantie sur la valeur de $C(q)$. Une technique classique de régularisation consiste alors à rajouter au critère $\hat{C}$ une mesure qui pénalise la complexité de la solution, suivant le principe du rasoir d'Occam qui dit que les hypothèses les plus simples sont les plus vraisemblables voir [26]. Une possibilité est de minimiser [22].

$$\tilde{C}(q) = \hat{C}(q) + \lambda \|w\|^2 \tag{2.5}$$

Au lieu de $\hat{C}$, pour q défini comme dans (l'éq. 4), afin d'empêcher le vecteur w de contenir des valeurs arbitrairement grandes (en valeur absolue). Le paramètre λ contrôle la force de cette contrainte (lorsque $\lambda \to +\infty$ la seule solution possible est la fonction constante $q_x(1) = q_x(0) = 0,5$, qui est la plus simple qu'on puisse imaginer).
Le critère empirique $\hat{C}\,(q^*)$ pour la fonction q^* qui minimise le critère régularisé $\tilde{C}$ pourrait ne pas être proche de zéro, mais on peut souvent ainsi - pour certaines valeurs de λ - obtenir des valeurs plus basses du critère de généralisation C (celui qui nous intéresse vraiment). C'est le principe de la minimisation du risque structurel [25, 22].
Dans cet exemple, nous avons utilisé $\|w\|^2$ pour mesurer la complexité de la fonction q définie à partir de w par (l'éq. 4). En général, il n'existe pas une seule mesure de complexité universelle qui soit appropriée pour tous les algorithmes d'apprentissage, et le

choix de la mesure de complexité à pénaliser joue un rôle très important. La complexité de Kolmogorov [27, 28], est une mesure de complexité très générique qui est intéressante en théorie, même si en pratique elle est souvent impossible à utiliser directement. Elle consiste à dire que la complexité d'une fonction est la taille du plus petit programme qui l'implémente. Un premier obstacle à l'utilisation de cette complexité est le fait qu'il faille choisir un langage de programmation approprié : par exemple si le langage choisi contient une fonction primitive qui calcule le produit scalaire, alors, dans notre exemple ci-dessus la plupart des fonctions q définies par (l'éq. 4) ont la même complexité de Kolmogorov. Par contre, si le produit scalaire n'est pas une primitive du langage (et qu'il n'y a pas d'instruction de boucle), alors il faut l'écrire comme une somme de produits et q est d'autant plus complexe que w a d'éléments non nuls. Une autre difficulté est qu'il n'est en général pas possible d'optimiser la complexité de Kolmogorov de manière efficace, ce qui rend vaine son utilisation directe dans un processus d'optimisation. Elle a malgré tout de nombreuses applications, comme décrit dans le livre de [29, 22].

2.4.3 Malédiction de la dimensionnalité

On peut observer empiriquement - et dans certains cas justifier mathématiquement -que plus la dimension d de l'entrée x est élevée, plus les tâches d'apprentissage machine ont tendance à être difficiles à résoudre. C'est ce qu'on appelle la malédiction de la dimensionnalité [30]. II existe plusieurs manifestations de cette malédiction. La plus importante dans le contexte de cette thèse est le fait que le nombre de combinaisons possibles des entrées augmente exponentiellement avec la dimensionnalité d : en notant x_{ij} la valeur associée à la j-ème dimension de l'entrée x_i, si l'on suppose que ces entrées ne peuvent prendre qu'un nombre finit k de valeurs, alors le nombre de combinaisons possibles est égal à k^d. Un algorithme qui apprend "bêtement" à associer une valeur à chaque combinaison sans partager d'information entre les différentes combinaisons n'a aucune chance de fonctionner en haute dimension, car il ne pourra pas généraliser aux multiples combinaisons qui n'ont pas été vues dans l'ensemble d'entraînement. Dans le cas où x_{ij} ne serait pas contraint dans un ensemble fini de valeurs, l'intuition reste la même pour certains algorithmes qui consistent à "partitionner" R^d en régions indépendantes (possiblement de manière implicite) : si le nombre de ces régions augmente exponentiellement avec d, alors un tel algorithme aura de la difficulté à généraliser pour de grandes valeurs de d. La figure 2.8 illustre ce phénomène en une et deux dimensions, et il faut garder à l'esprit que la situation peut s'avérer encore bien pire lorsque l'on manipule des entrées à plusieurs centaines de dimensions [22].

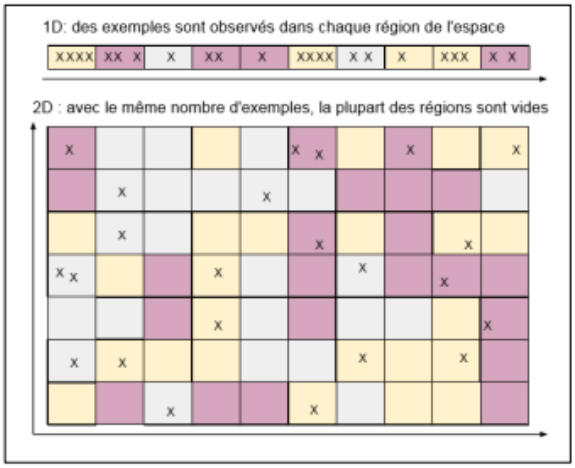

FIGURE 2.8 – Malédiction de la dimensionnalité [22].

2.5 Différents types de modèles

2.5.1 Modèles paramétriques

En apprentissage machine, un modèle paramétrique est défini par un ensemble Θ de paramètres de dimension fini, et l'algorithme d'apprentissage associé consiste à trouver la meilleure valeur possible de Θ. La figure 2.9 montre un exemple de régression linéaire en une dimension, sans régularisation.

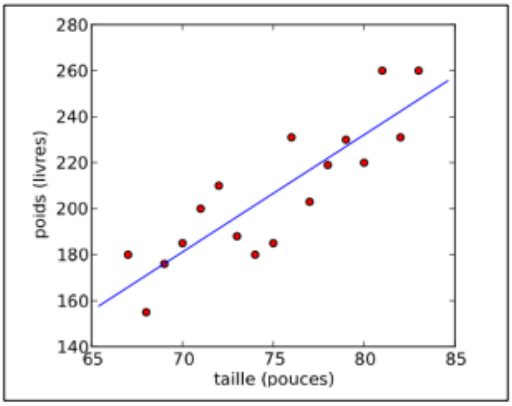

FIGURE 2.9 – Modèle paramétrique : la régression linéaire. Les données (points rouges) sont un sous-ensemble, et la tâche est ici de prédire le poids d'un joueur de baseball en fonction de sa taille [22].

Les modèles paramétriques peuvent également être statistiquement inefficaces. S'il y a moins d'exemples d'apprentissage, alors le problème est sur-paramétré et on risque le sur-apprentissage : le modèle pourrait apprendre des paramètres taillés "sur mesure" pour les données d'entraînement, mais qui mèneront à une mauvaise généralisation. La

conséquence de cette observation est qu'en général, un modèle avec un grand nombre de paramètres est statistiquement inefficace [22].

2.5.2 Modelés non paramétriques

Un modèle non paramétrique n'a au contraire pas d'ensemble exact de paramètres : le nombre de variables utilisées par le modèle augmente généralement avec le nombre d'exemples dans l'ensemble d'entraînement. Un exemple de modèle non paramétrique pour résoudre le même problème de régression que celui décrit en 2.5.1 est l'algorithme des fenêtres de Parzen, aussi appelé régression de Nadaraya-Watson [31, 32]. La figure 2.10 montre un exemple de régression par fenêtres de Parzen en une dimension [22].

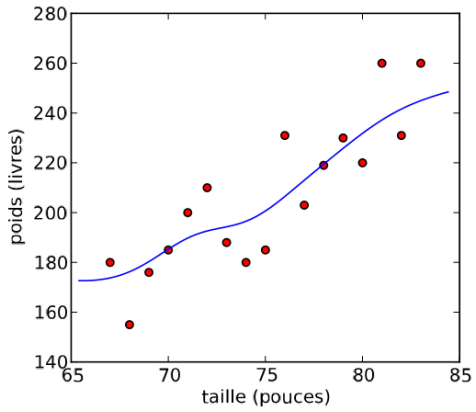

FIGURE 2.10 – Modèle non paramétrique : régression par fenêtres de Parzen. Les données sont les mêmes que dans l'exemple de la régression linéaire (La figure 2.9) [22].

2.6 Algorithmes d'apprentissage

2.6.1 Réseaux de neurones

Comme leur nom l'indique, les réseaux de neurones sont inspirés de l'architecture du cerveau, étant organisés en couches de neurones connectées entre elles. La première couche, appelée la couche d'entrée, est de la même dimension que les entrées x et l'activation de son j-ème neurone (c'est-à-dire la valeur qu'il calcule) est égale à x_{ij} lorsque l'on calcule la prédiction du réseau sur l'exemple x_i.
La dernière couche, appelée la couche de sortie, est de la même dimension que l'étiquette dans le cas d'une tâche supervisée. Par exemple, pour la classification, le j-ème neurone de la couche de sortie va calculer $P(Y = j \mid x_i)$ lorsque x_i est dans la couche d'entrée. Les couches intermédiaires sont appelées les couches cachées. II existe de nombreuses variations dans les architectures de réseaux de neurones.
L'architecture la plus connue est celle où il existe une unique couche intermédiaire h qui

calcule une transformation non linéaire des entrées, de la forme [22] :

$$H(x) = \sigma\left(W^T x + b\right) \tag{2.6}$$

Où W est une matrice de poids, b est un vecteur de biais (de la même taille que le nombre de neurones dans la couche cachée), et σ est une transformation non linéaire élément par élément, dont l'opération sur chaque élément est typiquement la sigmoïde ou la tangente hyperbolique [22] :

$$\text{Sigmoid}(u) = \frac{1}{1 + e^{-u}} \tag{2.7}$$

$$\tanh\tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}} \tag{2.8}$$

La fonction donnant la sortie $\hat{y}$ en fonction de h dépend de la tâche ; en classification, on écrit souvent le $j^{\text{éme}}$ neurone de $\hat{y}$, qui estime $P(Y = j \mid x)$, sous la forme :

$$\frac{e^{V_j^T h + c_j}}{\sum_k e^{V_k^T h + c_k}} \tag{2.9}$$

Avec V_j la j-ème rangée d'une matrice de poids V, et c le biais du j-ème neurone de sortie. La figure 2.11 montre une telle architecture à une couche cachée.

L'apprentissage dans un réseau de neurones consiste à optimiser les poids et biais de façon à minimiser un coût approprié à la tâche. Par exemple, pour la classification, on minimisera la log-vraisemblance négative empirique (éq. 2.3) qui, en notant Θ les paramètres à optimiser, et $f_j\left(\mathrm{x}_i; \Theta\right)$ la valeur du j-éme neurone de sortie Iorsque x_i est en entrée du réseau, se réécrit [22] :

$$\hat{C}(\Theta) = -\frac{1}{n} \sum_{i=1}^{n} \ln\ln f_{y_i}\left(x_i; \Theta\right) \tag{2.10}$$

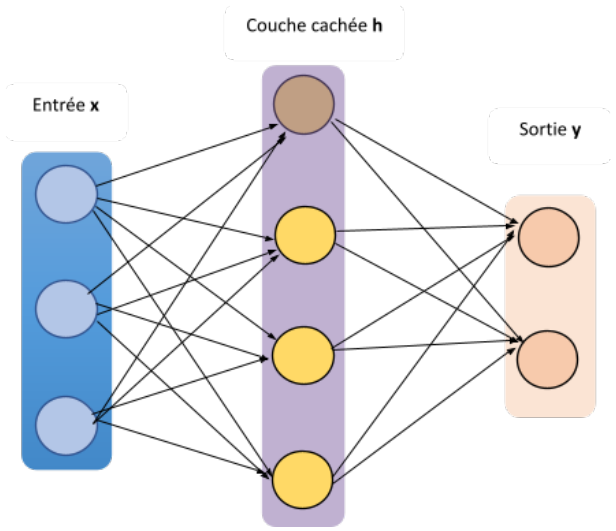

FIGURE 2.11 – Réseau de neurones à une couche cachée. Une flèche du neurone i vers le neurone j indique que l'activation de j dépend directement de celle de i [22].

Un autre exemple, en apprentissage non supervisé, est celui des réseaux de neurones auto-associateurs dont le but est d'extraire dans la couche cachée une représentation qui permet de reconstruire l'entrée x_i (donc l'étiquette y_i est en fait égale à x_i). Dans ce cas, le coût le plus fréquemment utilisé est l'erreur de reconstruction quadratique moyenne [22].

$$\hat{C}(\Theta) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\left(f_i\left(x_i;\Theta\right) - x_{ij}\right)^2 \tag{2.11}$$

Mais lorsque les entrées $x_{ij} \in \{0,1\}$, on peut également minimiser l'entropie croisée

$$\hat{C}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\left(\left(x_{ij}\ln f_i\left(x_i;\Theta\right) + \left(1-x_{ij}\right)\ln\ln\left(1-f_i\left(x_i;\Theta\right)\right.\right.\right. \tag{2.12}$$

Où le terme dans la somme peut s'interpréter comme la log-vraisemblance des données d'entraînement selon la distribution $P_\Theta\left(X_{ij}=1 \mid x_i\right) = f_i\left(x_{ij};\Theta\right)$. L'apprentissage dans un réseau de neurones consiste à optimiser les poids et biais de façon à minimiser un coût approprié à la tâche. Les algorithmes d'optimisation les plus souvent utilisés pour minimiser $\hat{C}(\Theta)$ sont des algorithmes d'optimisation locale basés sur l'idée de la descente de gradient : le gradient du coût $\hat{C}(\Theta)$ par rapport aux paramètres Θ, noté $\frac{\partial\hat{C}(\theta)}{\partial\theta}$ indique la direction dans laquelle le coût augmente le plus lorsque Θ varie. Descendre le gradient signifie déplacer les paramètres Θ dans la direction opposée, de manière à diminuer le coût. Le gradient par rapport à tous les paramètres *du* réseau peut se calculer de manière efficace grâce à l'algorithme de rétro-propagation du gradient [33, 22].
En apprentissage non supervisé, on parle des réseaux de neurones auto-associateurs dont le but est d'extraire dans la couche cachée une représentation qui permet de reconstruire l'entrée x_i (donc l'étiquette y_i est en fait égale à x_i). Dans ce cas, le coût le plus

fréquemment utilisé est l'erreur de reconstruction quadratique moyenne. On trouve donc que les réseaux de neurones représentent l'algorithme le plus approprié pour notre projet et pour l'implémentation [22].

2.6.2 Machines de Boltzmann restreintes

Dans sa version la plus simple, une machine de Boltzmann restreinte (RBM)est un algorithme non supervisé qui modélise la distribution $P(X)$ où $x \in \{0,1\}^d$ comme la marginale d'une distribution jointe [22] :

$$P(x,h) = \frac{e^{-\mathrm{z}(x,h)}}{\mathrm{z}} \tag{2.13}$$

Où Z est la fonction de partition assurant que cette probabilité est bien normalisée :

$$Z = \sum_{x,h} e^{-\varepsilon(x,h)} \tag{2.14}$$

Le vecteur $h \in \{0,1\}^k$ représente la couche cachée, tandis que le vecteur x est appelé la couche visible. Les éléments d'une couche sont généralement appelés unités plutôt que neurones. La quantité $\varepsilon(x,h)$ est l'énergie de la paire (x,h), et l'équation 13 montre que plus l'énergie est faible, plus cette paire est probable. Une forme typique pour ε est [22] :

$$\varepsilon(x,h) = h^T W x - x^T b - h^T c \tag{2.15}$$

Où la matrice $W \in R^{k \times d}$ et les vecteurs $b \in R^d$ et $c \in R^k$ sont les paramètres du modèle. Bien que ce modèle ait originellement été introduit sous un autre nom [34], il s'agit bien d'une forme particulière de machine de Boltzmann [35, 36]. Comparée à une machine de Boltzmann générique, l'énergie d'une RBM à la propriété que les probabilités conditionnelles $P(x \mid h)$ et $P(h \mid x)$ se factorisent, ce qui rend l'entraînement (un peu) plus aisé. Cette propriété se visualise lorsque l'on représente une RBM sous la forme d'un modèle graphique non dirigé [37], comme dans la figure 2.12 : il y a des connexions entre les unités visibles et les unités cachées, mais aucune connexion de visible à visible, ni de cachée à cacher [22].

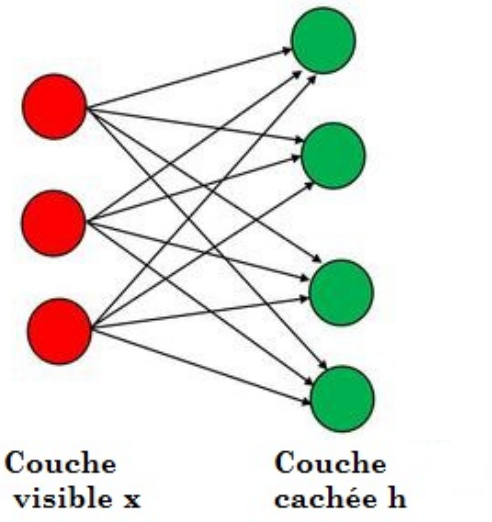

FIGURE 2.12 – Machine de Boltzmann restreinte [38].

2.6.3 K-plus proches voisins

L'algorithme des k plus proches voisins est un algorithme non paramétrique utilisé pour la régression et la classification. Etait donné une mesure de distance dans l'espace d'entrée R^d (souvent prise comme la distance Euclidienne$\|\|x_i - x_j\|$) la prédiction du modèle sur un exemple de test $x \in T$ dépend uniquement des k plus proches voisins de x dans l'ensemble d'entraînement D. En notant $i_1(x), \ldots, i_k(x)$ les indices des k exemples de D les plus proches de x selon la distance choisie (ses "voisins"), la prédiction du modèle en régression est la moyenne des étiquettes observées chez ces k voisins [22] :

$$f(x) = \frac{1}{k} \sum_{j=1}^{k} y_{i_j(x)} \tag{2.16}$$

Et en classification il s'agit d'un vote parmi les k voisins :

$$f(x) = \text{argmax}_y \sum_{j=1}^{k} 1_{y=y_{i,(x)}} \tag{2.17}$$

Où en cas d'égalité parmi les votes le modèle choisit aléatoirement l'une des classes majoritaires. La classification par les k plus proches voisins est illustrée en figure 2.13.

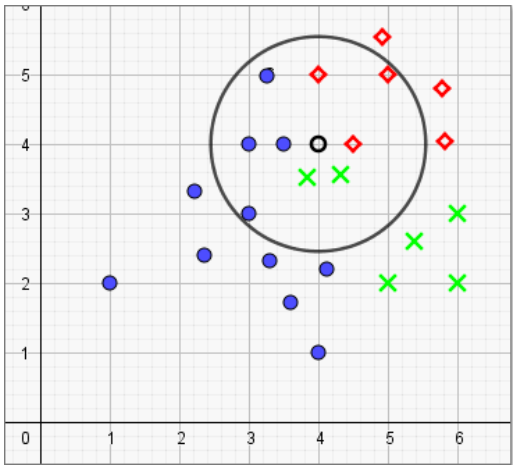

FIGURE 2.13 – K-plus proches voisins (k = 9, tâche de classification) [39].

2.6.4 Fenêtres de Parzen

L'algorithme des fenêtres de Parzen a déjà été présenté dans le contexte de la régression non paramétrique, où on l'appelle parfois la régression à noyau ou la régression de Nadaraya et Watson [31, 32]. On peut également utiliser une approche similaire en apprentissage non supervisée pour l'estimation de densité [40, 41], en estimant la densité de probabilité au point x par [22] :

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(x, x_i\right) \tag{2.18}$$

Ce qui correspond à placer une masse de probabilité "autour" de chaque exemple d'apprentissage x_i, dans un volume défini par le noyau K. Ici, K doit respecter les contraintes [22] :

$$K\left(x, x_i\right) \geq 0 \tag{2.19}$$

$$\int_x K\left(x, x_i\right) dx = 1 \tag{2.20}$$

De manière à ce que f soit une densité de probabilité valide. Le choix le plus répandu pour K est le noyau Gaussien, mais avec la normalisation appropriée [22] :

$$K\left(x, x_i\right) = N\left(x_{i'}; x_{j'} o^2 I\right) = \frac{1}{(2\pi)^{\frac{d}{g\sigma}}} e^{-\frac{\|x_i - x\|^2}{2\sigma}} \tag{2.21}$$

où l'on note $N(\cdot; \mu, \Sigma)$ la densité de probabilité d'une gaussienne de moyenne μ et covariance Σ. Un exemple d'estimation de densité par fenêtres de Parzen avec un noyau Gaussien est montré en figure 2.14 [22].

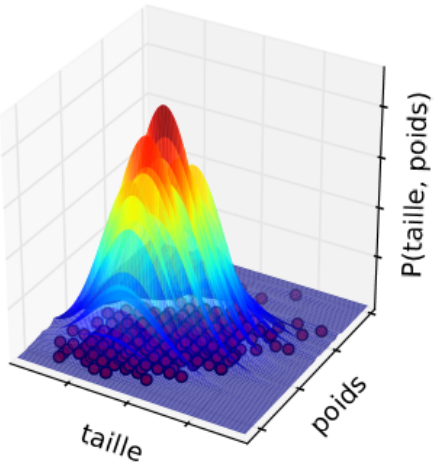

FIGURE 2.14 – Fenêtres de Parzen pour l'estimation de densité [22].

2.6.5 Mélanges de Gaussiennes

Les mélanges de Gaussiennes généralisent les fenêtres de Parzen (avec noyau Gaussien) pour l'estimation de densité, en écrivant la densité comme une somme pondérée de Gaussiennes [42] :

$$f(x) = \sum_{j=1}^{N} \alpha_j N\left(x; \mu_{j'} \Sigma_j\right) \tag{2.22}$$

Où N est le nombre de composantes du mélange, et α_j le poids de la jème composante (les poids sont tels que $\alpha_j \geq 0$ et $\Sigma_j \alpha_j = 1$). L'interprétation dite "générative" de cette équation est que le modèle suppose que chaque exemple observé a été généré de la façon suivante [22] :

1. Une composante j est choisit aléatoirement, avec probabilité α_j.
2. Un exemple est généré par une distribution Gaussienne centrée en μ_j avec covariance Σj.

Si N $= n, \alpha_j = n^{-1}, \mu_j = x_i$ et $\Sigma_j = \sigma^2$ on retrouve les fenêtres de Parzen non paramétriques vues précédemment. Mais on peut également apprendre un modèle paramétrique de mélange en fixant N et en optimisant les poids α_j, les centres μ_j et les covariances Σ_j de chaque Gaussienne. L'algorithme le plus populaire pour l'apprentissage d'un mélange de gaussiennes est l'algorithme Espérance-Maximisation (EM) [43].

2.6.6 Méthodes à noyau

Plusieurs algorithmes mentionnés précédemment utilisent une fonction noyau $\mathrm{K}(\mathrm{x_i}, \mathrm{x_j})$ pour mesurer la similarité entre deux entrées x_i et x_j. Les méthodes dites "à noyau" incluent en particulier une catégorie d'algorithmes basés sur ce que I'on appelle "l'astuce du noyau", qui s'applique à tout algorithme qui peut s'exprimer uniquement à partir de produits scalaires de la forme $x_i^T x_j$. Cette astuce consiste à remplacer $x_i^T x_j$ dans l'algorithme d'origine par une fonction noyau $\mathrm{K}(\mathrm{x_i}, \mathrm{x_j})$, où K doit satisfaire certaines propriétés mathématiques (on dit que le noyau est défini positif c'est le cas par exemple du noyau Gaussien que nous avons déjà utilisé). Cela revient à appliquer l'algorithme sur les données transformées implicitement et de manière non linéaire par une fonction ϕ telle que $\varphi(x_i)^T \varphi(x_j) = K(x_i, x_j)$ (une telle fonction ϕ existe automatiquement si K est défini positif, et en pratique on n'a pas besoin de la calculer explicitement). L'intérêt du noyau est principalement d'effectuer une extraction de caractéristiques non linéaires à partir des entrées, ce qui peut améliorer les performances de l'algorithme [22].
La plus célèbre méthode à noyau exploitant cette astuce est l'algorithme de la machine à vecteurs de support (SVM, pour "Support Vector Machine" en anglais). Il s'agit d'un algorithme de classification basée sur l'idée de marge : pour obtenir une meilleure généralisation, il est préférable de laisser une marge entre les exemples et la surface de décision [44, 45, 46]. Ce concept de marge est illustré par la figure 2.15 : dans le cas linéaire (c'est-à-dire. Sans utiliser l'astuce du noyau) la marge du classifieur est la distance entre l'hyper-plan séparant les deux classes et les exemples les plus proches. Dans le cas non linéaire, le même principe s'applique dans l'espace des exemples projetés implicitement par ϕ, ce qui se traduit dans l'espace des entrées par une séparation non linéaire des exemples de chaque classe. Depuis les SVMs, l'astuce du noyau et l'idée de marge ont inspiré un grand nombre de nouveaux algorithmes, ainsi que la "noyautisation" d'algorithmes existants [47, 22].

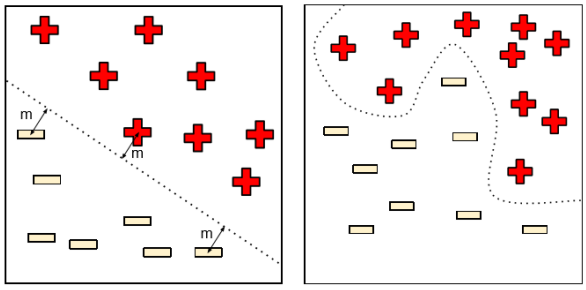

FIGURE 2.15 – Machine à vecteurs de support, dans le cas linéaire (à gauche) et non linéaire (à droite) [22].

2.6.7 Arbres de décision

Les arbres de décision forment une famille d'algorithmes d'apprentissage utilisés pour la classification et la régression [48]. Un arbre de décision est un arbre où chaque nœud interne représente un test sur l'entrée x_i. L'exemple typique d'un arbre de décision est un arbre binaire où le test effectué à chaque nœud k est de la forme $x_{ij} < \theta_k$, c'est-à-dire. Qu'on compare la $j^{\text{éme}}$ coordonnée de x_i à un seuil θ_k : si elle est plus petite, on continue de parcourir l'arbre en suivant la première branche du nœud, sinon on suit la seconde branche. Lorsqu'on atteint finalement une feuille de l'arbre (un nœud sans enfants), on dit que l'exemple x_i appartient à cette feuille. La figure 2.16 montre un tel arbre de décision [22].

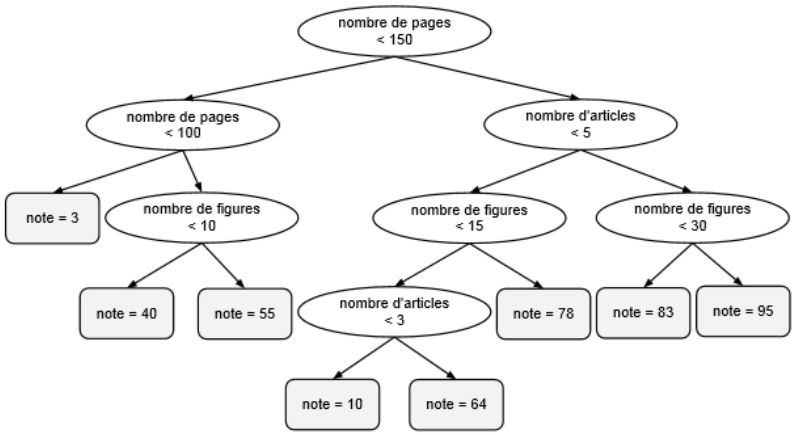

FIGURE 2.16 – Arbre de décision typique [22].

L'apprentissage de ce type d'arbre de décision consiste à choisir les variables testées à chaque nœud, les seuils de comparaison, la profondeur de l'arbre, ainsi que la fonction de décision associée à chaque feuille. II existe plusieurs algorithmes pour cela, mais leur principe de base est de faire en sorte que le résultat du test effectué à chaque nœud donne de l'information supplémentaire sur $P(Y \mid x)$. Idéalement, la distribution $P(Y \mid x)$ pour un nouvel exemple x doit être bien approximée par la distribution empirique des étiquettes des exemples d'entraînement appartenant à la même feuille que [22].

2.6.8 Méthodes Bayésiennes

Les méthodes Bayésiennes tirent leur nom de la célèbre règle de Bayes [22] :

$$P(D) = \frac{P(D)P(\Theta)}{P(D)} \tag{2.23}$$

Qui est ici appliquée avec d'une part les paramètres Θ d'un modèle, et d'autre part les données d'entraînement D observées. La quantité $P(D)$ est appelée la vraisemblance :

c'est la probabilité d'observer les données D en supposant qu'elles ont été générées par notre modèle dont les paramètres sont Θ.$P(\Theta)$ est appelée la distribution a priori : c'est une distribution sur l'espace des paramètres qui reflète notre croyance sur les valeurs possibles des paramètres avant l'observation des données. $P(\Theta \mid D)$ est alors calculé comme étant proportionnelle au produit de la vraisemblance par la distribution a priori : elle est appelée la distribution a posteriori, c'est-à-dire. Qu'elle indique la probabilité des paramètres après avoir observé les données. Le terme $P(D)$ est un terme de normalisation qui peut se calculer, si nécessaire, par $P(D) = \int_\Theta P(D \mid \Theta)P(\Theta)d\Theta$. II suffira de garder à l'esprit qu'il s'agit de méthodes probabilistes, qui ont l'avantage en particulier de prendre en compte l'incertitude de manière naturelle : par exemple, la prédiction d'un modèle Bayésien supervisé sur une nouvelle entrée x peut s'écrire comme [22].

$$P(y \mid x, D) = \int_\Theta P(\Theta)P(\Theta \mid D)d\Theta \tag{2.24}$$

Et la variance de cette distribution peut être interprétée comme l'incertitude sur la prédiction de l'étiquette y. Notons qu'il n'est en général pas trivial de calculer une telle intégrale, et que plusieurs techniques ont été développées spécifiquement dans ce but [49, 22].

2.6.9 L'apprentissage en profondeur (Deep Learning)

Le Deep Learning est un nouveau domaine de recherche du ML, qui a été introduit dans le but de rapprocher le ML de son objectif principal : l'intelligence artificielle. Il concerne les algorithmes inspirés par la structure et le fonctionnement du cerveau. Ils peuvent apprendre plusieurs niveaux de représentation dans le but de modéliser des relations complexes entre les données.

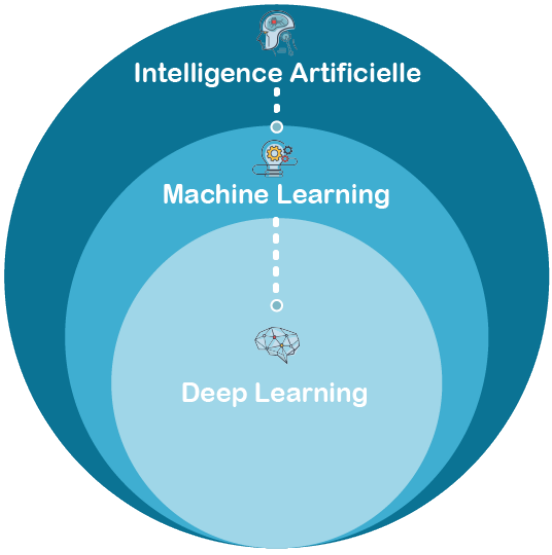

FIGURE 2.17 – La relation entre l'intelligence artificielle, le ML et le deep Learning [50].

L'apprentissage en profondeur est basé sur l'idée des réseaux de neurones artificiels et il est taillé pour gérer de larges quantités de données en ajoutant des couches au réseau. Un modèle de deep learning a la capacité d'extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires et apprendre sur ses caractéristiques petit à petit à travers chaque couche avec une intervention humaine minime [51].

Il existe différentes architectures de Deep Learning. Nous pouvons ainsi citer :

2.6.9.1 Convolutional Neural Networks (CNN)

Les réseaux de neurones convolutifs, où CNN, sont le choix populaire des réseaux de neurones pour différentes tâches de vision par ordinateur tel que la reconnaissance d'image.
Le nom « convolution » est dérivée d'une opération mathématique impliquant la convolution de différentes fonctions. La conception d'un CNN comporte 4 étapes principales :

- **Convolution :** le signal d'entrée est reçu à ce stade.
- **Sous-échantillonnage :** les entrées reçues de la couche de convolution sont lissées pour réduire la sensibilité des filtres au bruit ou à toute autre variation.
- **Activation :** cette couche contrôle la façon dont le signal circule d'une couche à l'autre, semblable aux neurones de notre cerveau.
- **Entièrement connecté :** dans cette étape, toutes les couches du réseau sont connectées avec chaque neurone d'une couche précédente aux neurones de la couche suivante.

Voici un aperçu approfondi de l'architecture CNN et de son fonctionnement, comme l'explique le célèbre chercheur en IA Giancarlo Zaccone [50].

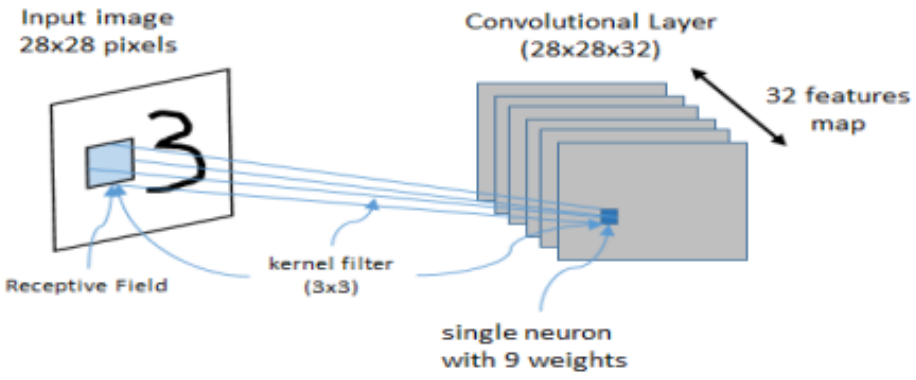

FIGURE 2.18 – Un échantillon de CNN en action [50].

2.6.9.2 Récurrent Neural Networks (RNN)

Les réseaux neuronaux récurrents (RNN) ont été très populaires dans les domaines où la séquence dans laquelle les informations sont présentées est cruciale. En conséquence, ils trouvent de nombreuses applications dans des domaines du monde réel tel que le traitement du langage naturel, la synthèse vocale et la traduction automatique.
Les RNN sont appelés « récurrents » principalement parce qu'une tâche uniforme est exécutée pour chaque élément unique d'une séquence, la sortie dépendant également des calculs précédents. Considérez ces réseaux comme ayant une mémoire, où chaque information calculée est capturée, stockée et utilisée pour calculer le résultat final.
Au fil des ans, plusieurs variétés de RNN ont été recherchées et développées :

- **RNN bidirectionnel :** La sortie de ce type de RNN dépend non seulement du passé mais aussi des résultats futurs
- **RNN profond :** Dans ce type de RNN, il y a plusieurs couches présentes par étapes, permettant un plus grand taux d'apprentissage et plus de précision [50].

2.7 Conclusion

Nous avons présenté au cours de ce chapitre l'apprentissage automatique avec ses différents types (supervisé, non-supervisé, semi supervisé et avec renforcement), les différents types de modèles, ensuite nous avons présenté les algorithmes d'apprentissage tel que l'apprentissage profond et ses différentes méthodes : CNN, RNN.
Dans le chapitre suivant, nous expliquant la conception de notre système de dissimuler multiples images dans une image, ainsi les approches utilisées et on montre les résultats obtenus.

Chapitre 3

Conception et implémentation

3.1 Introduction

Dans ce dernier chapitre nous présentons la conception de notre projet qui permettra d'encoder plusieurs images dans une image de couverture unique. Ensuite, nous allons décrire les différentes parties de notre système et les outils, Logiciels utilisés dans l'implémentation de notre projet, et enfin la mise en place et les résultats.

3.2 Conception générale

Dans notre projet nous proposons une approche de stéganographie pour cacher multiples images dans une seule image.
Pour ce faire, nous utilisons les idées de l'article (Baluja, 2017), Baluja a proposé un modèle de stéganographie d'image fondé sur CNN profond pour intégrer l'image entière dans une autre image de même taille [2]. On utilise un modèle CNN basé sur un encodeur automatique, le système est composé en 2 parties, l'encodeur qui contient le réseau de préparation qui est responsable de la préparation des images secrètes à masquer, et le réseau de masquage qui génère l'image du conteneur en utilisant la sortie du réseau de préparation et l'image de couverture comme entrée. La deuxième partie est le décodeur qui contient un seul réseau, c'est le réseau de détection qui supprime l'image de couverture pour révéler les images secrètes.

3.3 Conception détaillé

Dans cette section, nous détaillons notre modèle de stéganographie consistant en une architecture d'encodeur-décodeur basé sur l'apprentissage profond (CNN) pour la stéganographie et la stéganalyse d'images.

Son objectif est de pouvoir encoder des informations sur les images secrètes S1, S2 et S3 dans l'image de couverture C, générant C' qui ressemble beaucoup à C, tout en étant capable de décoder les informations de C' pour générer les images secrètes décodées S1', S2'et S3' qui devraient ressembler le plus possible aux images secrètes.
La figure 3.1 illustré les différentes parties de notre architecture générale.

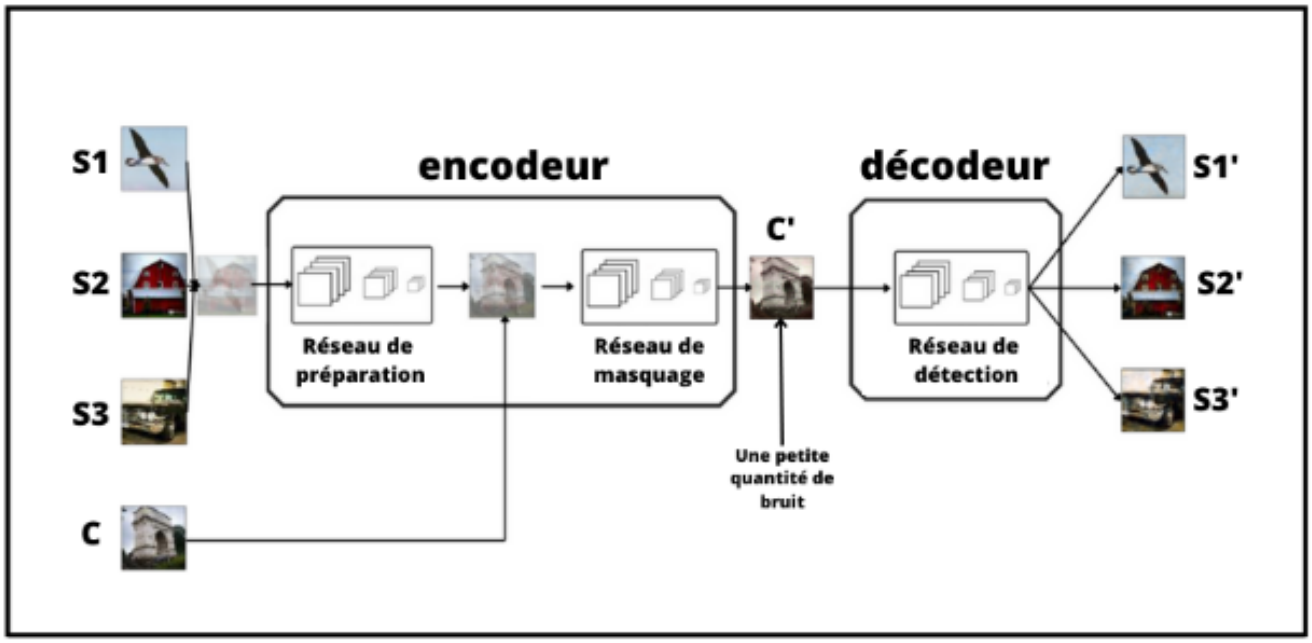

FIGURE 3.1 – L'architecture générale de notre modèle proposé pour la stéganographie et la stéganalyse.

3.3.1 L'architecture du modèle proposé

L'architecture du modèle est telle que décrite dans la figure 3.1 . L'architecture sous-jacente de chacun des sous-réseaux est la suivante :

- **Le réseau de préparation :**
  Après les avoir concaténées, les images (images secrètes) sont remises au réseau dans le but de préparer les images secrètes à cacher, et servent un autre objectif, dans les cas où la taille des images secrètes (M × M) est plus petite que l'image de couverture (N × N), le réseau de préparation augmente progressivement la taille des images secrètes jusqu'à la taille de la couverture, répartissant ainsi les bits des images secrètes sur l'ensemble des N × N pixels (nous n'avons pas besoin de cet objectif dans notre système car toutes nos images dans la base de données ont la même taille (64 × 64)).
  Ainsi, quelle que soit la taille d'origine des images, les images sont redimensionnées par le réseau de préparation, ces images sont traitées par deux couches convolutives (Conv2D), Le nombre de canaux pour chaque couche est 50 avec noyau (kernel) de

taille 3 pour chacun. La longueur de foulée reste constamment 1 le long des deux axes. Un remplissage (Padding) approprié est ajouté à chaque couche Conv2D afin de conserver l'image de sortie dans les mêmes dimensions. Chaque couche Conv2d est suivie d'une activation ReLU.

- **le réseau de masquage :**

  Génère l'image du conteneur (C') en utilisant la sortie du réseau de préparation et l'image de couverture en entrée. un champ de 64*64 pixels contenant des canaux RVB concaténés en profondeur de l'image de couverture et des canaux modifiés des images cachées sert d'entrée du réseau.

  Le réseau de masquage est une agrégation de 5 couches. La structure sous-jacente des couches Conv2D dans ce réseau est similaire aux couches Conv2D dans le réseau de préparation.

  L'architecture de l'encodeur (réseau de préparation + réseau de masquage) est présenté dans la figure 3.2

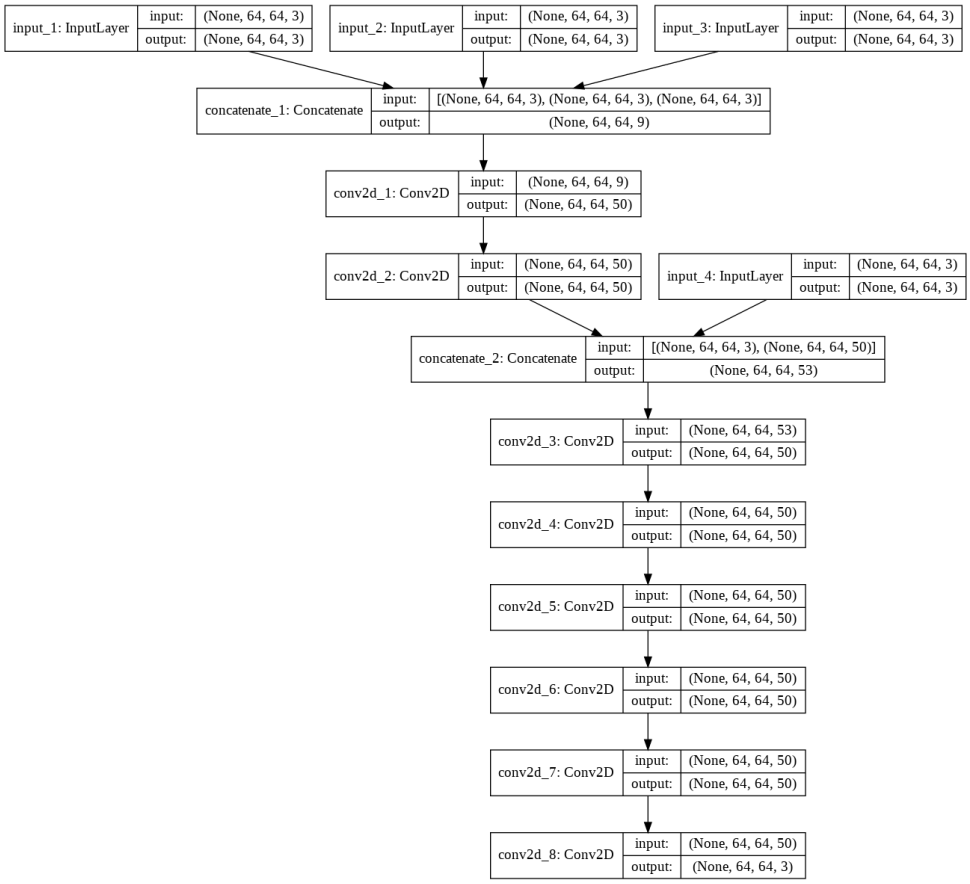

FIGURE 3.2 – L'architecture de l'encodeur.

- **le réseau de détection :**

  C'est le décodeur, il ne reçoit que l'image du conteneur (C') (pas la couverture ni les images secrète). Le réseau du décodeur supprime l'image de couverture pour révéler les images secrètes.

Pour s'assurer que les réseaux ne se contentent pas d'encoder les images secrètes dans les LSB, une petite quantité de bruit est ajoutée à l'entrée de ce réseau pendant l'entraînement (Bruit gaussien), le bruit a été conçu de telle sorte que le LSB était parfois basculé, cela a permis de s'assurer que le LSB n'était pas le seul conteneur pour les images secrètes.
le réseau de détection partager une architecture sous-jacente similaire avec le réseau de masquage, en utilisant 5 couches Conv2D de forme similaire.(Voir la figure 3.3)

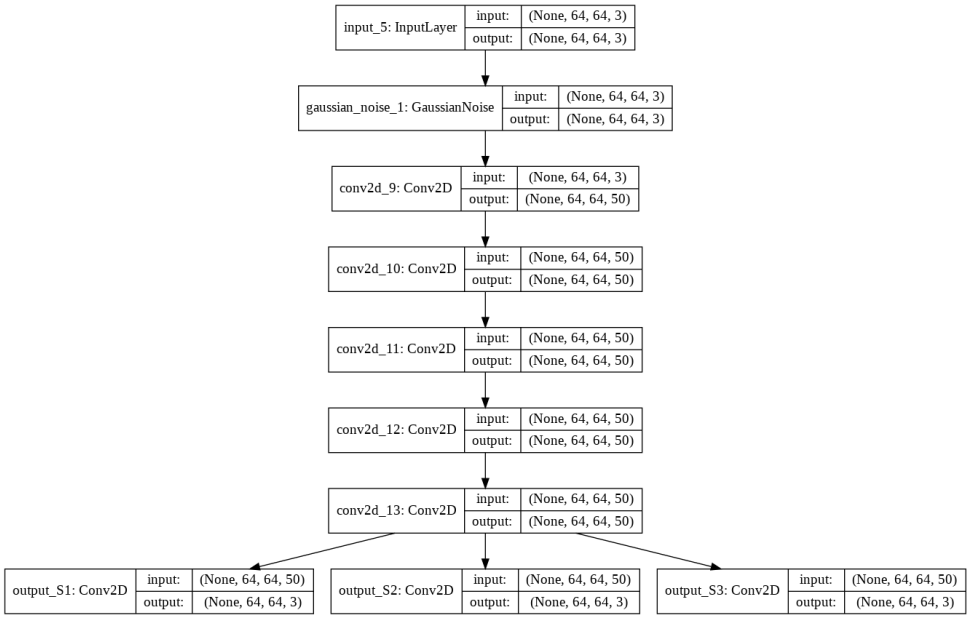

FIGURE 3.3 – L'architecture de décodeur.

3.3.2 Fractionnement des données

La procédure consiste à prendre une base de données et à le diviser en deux sous-ensembles. Le premier sous-ensemble est utilisé pour ajuster le modèle et est appelé base de données d'apprentissage (training dataset en anglais).
Le deuxième sous-ensemble n'est pas utilisé pour former le modèle, à la place, l'élément d'entrée de l'ensemble de données est fourni au modèle, puis des prédictions sont faites et comparées aux valeurs attendues.
Ce deuxième ensemble de données est appelé ensemble de données de validation.

3.3.3 Apprentissage CNN (Cnn learning)

Un aperçu d'un processus de formation de réseau neuronal convolutionnel (CNN). Un CNN est composé d'un empilement de plusieurs blocs de construction : des couches de convolution, des couches de regroupement (pooling layers) et des couches entièrement connectées(fully connected layers).

Les performances d'un modèle sous des noyaux et des poids particuliers sont calculées avec une fonction de perte (loss function) par propagation vers l'avant sur un ensemble de données d'apprentissage, et les paramètres apprenables sont mis à jour en fonction de la valeur de perte par rétropropagation avec un algorithme d'optimisation de descente de gradient [52], comme nous le verrons dans la figure 3.4.

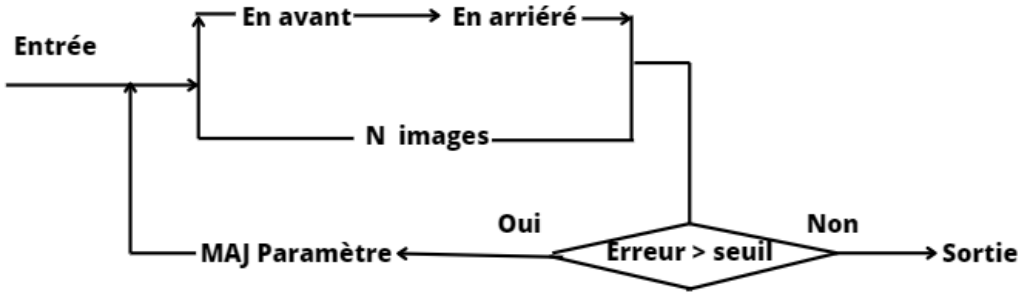

FIGURE 3.4 – L'algorithme sériel d'entraînement.

3.3.3.1 Optimisation de l'apprentissage CNN

Le choix de l'optimiseur dans le modèle d'apprentissage CNN peut améliorer les résultats des modèles et réduire le délai de jours à heures ou minutes.
L'objectif de tous les optimiseurs est d'atteindre le minimum global où la fonction de coût atteint la valeur la plus faible possible. L'optimiseur dans le réseau de neurones est un paramètre qui pourrait faire la différence entre la convergence ou l'explosion d'un algorithme.
Il existe un nombre considérable d'optimiseurs, voici les plus courants : Optimiseur de descente de gradient, Descente de gradient stochastique, Descente de gradient par lots mini, Optimiseur RMSprop et enfin Optimiseur Adam qui nous utilisait dans notre système.

3.3.3.2 Paramètres du modèle

Lors de l'entraînement et de l'ajustement du modèle à un ensemble de données donné, certains paramètres doivent être spécifiés, voici quelques-uns d'entre eux mentionnés brièvement.

1. **Le taux d'apprentissage (The learning rate) :** Est introduit comme une constante (généralement très petite), afin de forcer le poids à se mettre à jour très doucement et lentement (pour éviter de grands pas et un comportement chaotique).
   - Si l'erreur continue de s'aggraver ou oscille énormément, réduisez le taux d'apprentissage.
   - Si l'erreur diminue assez régulièrement mais lentement, augmentez le taux d'apprentissage.

- Baissez le taux d'apprentissage lorsque l'erreur cesse de diminuer.

Dans notre système, le taux d'apprentissage est personnalisé où il reste constant à 0,001 jusqu'aux 200 premières époques, diminuant à 0,0003 de 200 époques à 400 époques, diminuant à 0,0001 de 400 époques à 600, et pour les itérations restantes diminuant encore de 0,00003.

2. **Époque :** Une époque est lorsque l'ensemble de données entier est transmis une seule fois en avant et en arrière à travers le réseau de neurones. Les paramètres d'époque font référence au nombre de fois où les modèles visent à utiliser l'ensemble de données complet pour la formation.
3. **Lot (Batch) :** Pendant l'entraînement, l'ensemble de données est divisé en un certain nombre de lots / parties, grands les mini-lots sont plus efficaces sur le plan informatique.
4. **Taille du lot (Batch size) :** nombre total d'exemples d'entraînement présents dans un seul lot.
   Le modèle a été formé pour 1000 époques avec une taille de lot de 32.
5. **Fonction de perte (loss function) :** La fonction de perte dans ce cas utilisée pour former un auto-encodeur est appelée perte de reconstruction, qui compare l'image d'origine et la reconstruction. Où la robustesse du modèle augmente avec la diminution de la valeur de la perte de reconstruction. Le but du modèle est de minimiser cette perte.
   Dans l'implémentation de notre modèle, on a essayé d'appliquer quatre perte de reconstruction, le carré moyen des erreurs (MSE pour Mean Square Error), il s'agit de la somme du carré de la différence entre la cible prévue et la cible réelle, divisée par le nombre total de pixels dans l'image. Erreur absolue moyenne (MAE pour Mean Absolute Error), la somme des différences absolues entre les valeurs d'entrée et de cible, divisée par le nombre total de pixels dans l'image. La distance euclidienne, qui calculée comme la racine carrée de la somme des différences au carré entre la cible prévue et la cible réelle. Enfin la distance euclidienne sans racine carrée.
   Les quatre fonctions différentes pour calculer la perte de reconstruction sont les suivantes :
   - **Le carré moyen des erreurs (MSE) :**

$$\begin{aligned}\text{Perte } = &\frac{\sum_{i=1}^{N}(C_i - C_i')^2}{N} + \frac{\sum_{i=1}^{N}(S_{1i} - S_{1i}')^2}{N} + \\ &\frac{\sum_{i=1}^{N}(S_{2i} - S_{2i}')^2}{N} + \frac{\sum_{i=1}^{N}(S_{3i} - S_{3i}')^2}{N}\end{aligned} \tag{3.1}$$

- **Erreur absolue moyenne (MAE) :**

$$\begin{aligned}\text{Perte } = \frac{\sum_{i=1}^{N}|C_i - C_i'|}{N} + \frac{\sum_{i=1}^{N}|S_{1i} - S_{1i}'|}{N} + \\ \frac{\sum_{i=1}^{N}|S_{2i} - S_{2i}'|}{N} + \frac{\sum_{i=1}^{N}|S_{3i} - S_{3i}'|}{N}\end{aligned} \tag{3.2}$$

- **La distance euclidienne :**

$$\text{Perte } = \|C - C'\| + \|S_1 - S_1'\| + \|S_2 - S_2'\| + \|S_3 - S_3'\| \tag{3.3}$$

- **La distance euclidienne sans racine carrée :**

$$\text{Perte } = \|C - C'\|^2 + \left\|S_1 - S_1'\right\|^2 + \left\|S_2 - S_2'\right\|^2 + \left\|S_3 - S_3'\right\|^2 \tag{3.4}$$

3.3.4 Prédiction

Après la phase d'apprentissage sur les données, quels que soient les résultats obtenus, nous sauvegardons notre modèle.
Ensuite, nous utilisons le modèle pour prédire les images reconstruites des images secrètes d'entrée.
Avant la prédiction, nous devons considérer que la taille, la forme et les canaux des images sont les mêmes que nos échantillons d'apprentissage ou non, sinon nous devons les normaliser.
Dans notre cas, si la taille des images secrètes est inférieure à l'image de couverture, le réseau de préparation augmente progressivement la taille des images secrètes jusqu'à la taille de la couverture.

3.4 La base de données

Étant donné que notre modèle n'a pas d'exigences spécifiques concernant les classes des images, c'est un avantage du modèle en ce sens qu'il est général et peut-être appliqué à n'importe quel type d'image.
Nous avons utilisé les données de Tiny ImageNet afin d'obtenir la couverture et les images secrètes. Les données sont la collection d'images 64 × 64 × 3, utilisées par la classe Stanford CS231.
Notre base de données d'entraînement est composé d'un sous-ensemble aléatoire d'images provenant des 200 classes. La base de données d'entraînement est créé en prenant 10 images par classe (2000 images au total) et dans la base de données de test, 600 images sont échantillonnées au hasard (3 images par classe).
Les vecteurs d'images sont normalisés sur les valeurs RVB. Nous avons divisé la base de

données (entraînement et test) en quatre moitiés, une pour l'image de couverture et les trois autres moitiés pour trois images secrètes.

La figure ci-dessous montre exemples d'images dans la base de données d'entrainement.

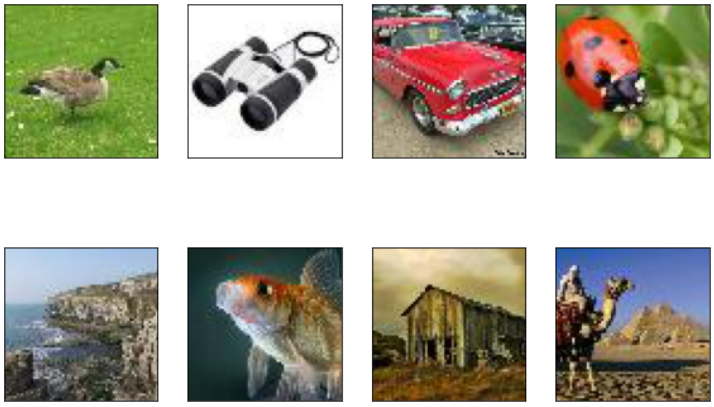

FIGURE 3.5 – Exemples d'images dans la base de données d'entraînement.

3.5 Cadres et outils utilisés dans l'implémentation

Les cadres d'apprentissage en profondeur sont très puissants et facilitent la révolution de l'intelligence artificielle car ils fournissent des outils pré-implémentés sont des modèles pour le prétraitement, la classification et l'évaluation. Sans ces outils, il serait très difficile pour les scientifiques de travailler sur des tâches d'apprentissage en profondeur. Cette section passe en revue les différents frameworks et outils de développement utilisés :

3.5.1 Python

Le langage de programmation le plus utilisé et le plus célèbre en science des données est un langage de programmation de haut niveau, et sa philosophie de conception de base concerne la lisibilité du code et la syntaxe qui permettent aux programmeurs d'exprimer des concepts en quelques lignes de code. Python est développé sous une licence open source approuvée par l'OSI, ce qui le rend librement utilisable et distribuable, même à des fins commerciales. Il est utilisé avec succès dans des milliers d'applications professionnelles réelles à travers le monde, y compris de nombreux systèmes importants et critiques [53].

FIGURE 3.6 – Python logo.

3.5.2 Matplotlib

Matplotlib est une bibliothèque de traçage pour le langage de programmation Python et son extension de mathématiques numériques NumPy. Il fournit une API orientée objet pour intégrer des tracés dans des applications utilisant des kits d'outils GUI à usage général tel que Tkinter, wxPython, Qt ou GTK+. Matplotlib a été écrit à l'origine par John D. Hunter.
Tout dans matplotlib est organisé dans une hiérarchie. Au sommet de la hiérarchie se trouve « l'environnement de machine d'état » matplotlib qui est fourni par le module matplotlib.pyplot. À ce niveau, des fonctions simples permettent d'ajouter des éléments de tracé (lignes, images, texte, etc.) aux axes de la figure [54].

FIGURE 3.7 – Exemples de travaux matplotlib[54].

3.5.3 Tensorflow

TensorFlow est une bibliothèque de logiciels open source pour le calcul numérique haute performance. Son architecture flexible permet un déploiement facile du calcul sur une variété de plates-formes (CPU, GPU, TPU), et des ordinateurs de bureau aux clusters de serveurs en passant par les périphériques mobiles et périphériques.
Développé à l'origine par des chercheurs et des ingénieurs de l'équipe Google Brain au sein de l'organisation IA de Google, il est livré avec un support solide pour l'apprentissage automatique et l'apprentissage en profondeur et le noyau de calcul numérique flexible est utilisé dans de nombreux autres domaines scientifiques [55].

FIGURE 3.8 – Tensorflow logo.

3.5.4 Keras

Keras est une API conçue pour les êtres humains, pas pour les machines. Keras suit les meilleures pratiques pour réduire la charge cognitive : il propose des API simples et cohérentes, il minimise le nombre d'actions de l'utilisateur requises pour les cas d'utilisation courantes et il fournit des commentaires clairs et exploitables en cas d'erreur de l'utilisateur. Keras est également un favori parmi les chercheurs en apprentissage profond, se classant 2émé en matière de mentions dans des articles scientifiques après TensorFlow [56].

FIGURE 3.9 – Keras logo.

3.5.5 NumPy

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux [57].

FIGURE 3.10 – NumPy logo.

3.5.6 Kaggle

Kaggle est une plateforme communautaire en ligne pour les scientifiques des données et les passionnés d'apprentissage automatique. Kaggle permet aux utilisateurs de collaborer avec d'autres utilisateurs, de rechercher et de publier des bases de données, d'utiliser des ordinateurs portables intégrés au GPU et de rivaliser avec d'autres scientifiques des données pour résoudre les défis de la science des données. L'objectif de cette plateforme en ligne (fondée en 2010 par Anthony Goldbloom et Jeremy Howard et acquise par Google en 2017) est d'aider les professionnels et les apprenants à atteindre leurs objectifs dans leur

parcours en science des données grâce aux puissants outils et ressources qu'elle fournit. À ce jour (2021), il y a plus de 8 millions d'utilisateurs enregistrés sur Kaggle [58].
Nous avons utilisé la base de données Tiny ImageNet de Kaggle, cette base de données contient 200 classes. Chaque classe contient 500 images. Toutes les images sont en couleur 64x64.

3.5.7 Google Colab

Google Colab est un service cloud gratuit avec GPU et TPU 2 gratuits avec une limite de 12 heures par session et 12 Go de limites de RAM. L'exécution de modèles profonds avec une architecture compliquée et d'énormes données peut conduire à l'utilisation d'une quantité extrême de matériel, Google Colabs fournit un GPU gratuit, actuellement ils fournissent un GPU avec les spécifications Tesla K80 , ayant 2496 cœurs CUDA, calcul 3.7, 12 Go (11.439 Go utilisable) VRAM GDDR5. et un CPU de CPU : 1 single core hyper threaded c'est-à-dire (1 core, 2 threads) Processeurs Xeon @2.3Ghz (No Turbo Boost), 45MB Cache [59].

FIGURE 3.11 – Google colab logo.

Pourquoi avons-nous besoin de GPU dans le deep learning ?

L'unité de traitement graphique est un processeur capable de gérer des calculs spécialisés. Cela contraste avec une unité centrale de traitement (CPU), qui est un processeur capable de gérer des calculs généraux. Les processeurs sont les processeurs qui alimentent la plupart des calculs typiques de nos appareils électroniques. Un GPU peut être beaucoup plus rapide en calcul qu'un CPU. Par contre, ce n'est pas toujours le cas.
La vitesse d'un GPU par rapport à un CPU dépend du type de calcul effectué. Le type de calcul le plus adapté à un GPU est un calcul qui peut se faire en parallèle. Le calcul parallèle est un type de calcul où, par un calcul particulier, il est divisé en calculs indépendants plus petits qui peuvent être effectués simultanément.
Les calculs résultants sont ensuite recombinés, ou synchronisés, pour former le résultat du calcul original plus grand. Le nombre de tâches dans lesquelles une tâche plus importante peut être divisée dépend du nombre de cœurs contenus sur un élément matériel particulier. Les cœurs sont les unités qui effectuent réellement le calcul dans un processeur donné, et

les processeurs ont généralement quatre, huit ou seize cœurs tandis que les GPU en ont potentiellement des milliers [60].

3.6 Résultats

Durant l'entraînement du notre modèle sur l'ensemble de données pour 1000 époques, la courbe de perte de reconstruction est calculée sur la base des quatre pertes de reconstruction, les résultats sont présentés dans la figure 3.13 ci-dessous.

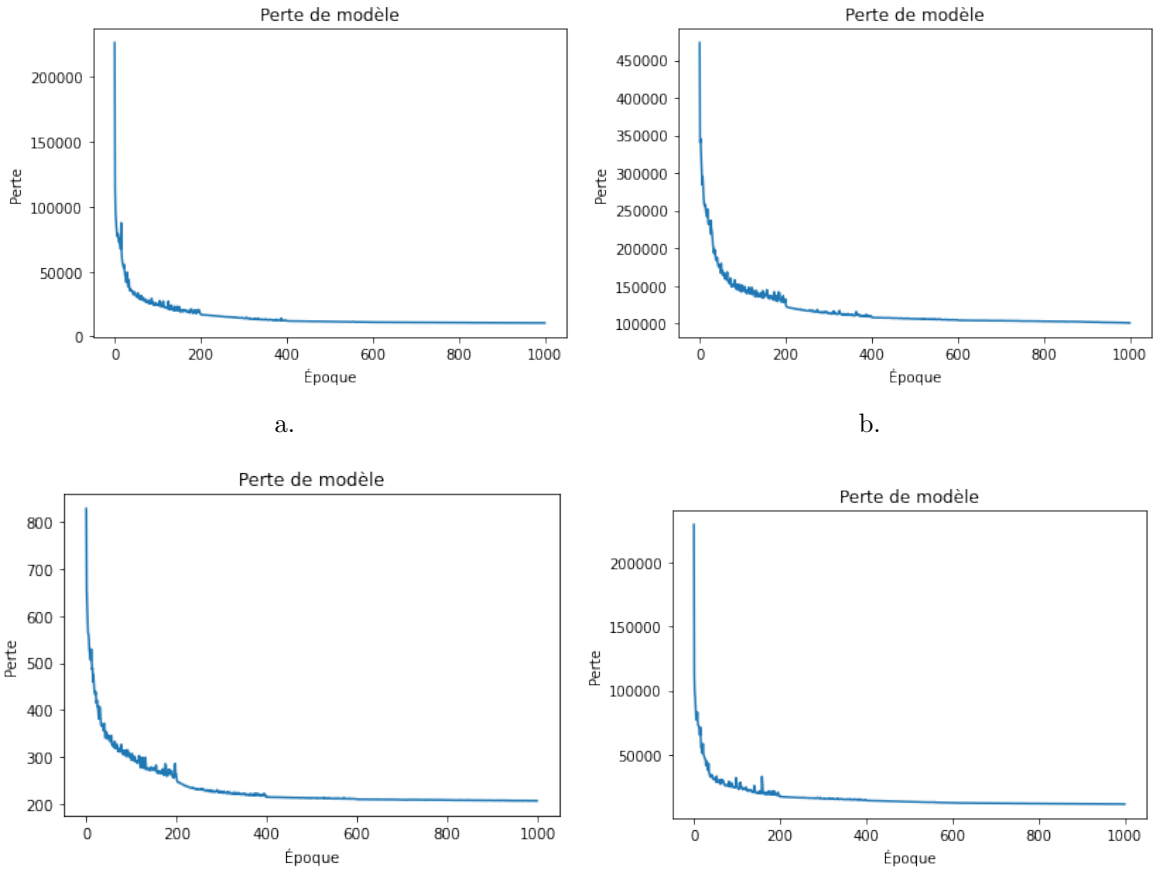

FIGURE 3.12 – (a) MSE (b) MAE (c) La distance euclidienne (d) La distance euclidienne sans racine carrée.

Après l'entraînement du modèle, diverses méthodes sont utilisées pour évaluer le modèle. Certaines des méthodes bien connues sont l'erreur quadratique moyenne (MSE), le rapport signale sur bruit de crête (PSNR), la mesure de l'indice de similarité structurée (SSIM), la capacité de charge utile.
Pour évaluer notre modèle, nous avons utilisant la méthode d'erreur par pixel, MSE et SSIM, et les résultats sont comme suit :
Quand le modèle utilise MSE comme perte de reconstruction, nous avons trouvé les résultats suivants : pour la méthode d'erreur par pixel, les valeurs sont les suivantes :

19,5075, 18,7921, 21,6515 et 30,2061 pour S1, S2, S3 et C, respectivement, et 0.0058, 0.0054, 0.0072 et 0.0140 avec MSE, et avec SSIM nous avons trouvé ces valeurs 0.9494, 0.9546, 0.9265 et 0.8741. Où une valeur de 0 pour MSE indique une similitude parfaite, par opposition au SSIM où des valeurs plus petites indiquent moins de similitude.
Quand on a utilisé MAE comme perte de reconstruction, les valeurs de performance sont : pour la méthode d'erreur par pixel : 23.5829, 16.2439, 30.2992 et 32.0534 pour S1, S2, S3 et C, respectivement, la méthode MSE : 0.0085, 0.0040, 0.0141 et 0.0158, et avec SSIM nous avons trouvé ces valeurs 0.9313, 0.9688, 0.8702 et 0.8648.
L'utilisation de la distance euclidienne comme perte de reconstruction, nous donnait ces les valeurs : pour la méthode d'erreur par pixel : 19.8646, 18.2634, 23.6769 et 30.7240 pour S1, S2, S3 et C, respectivement, la méthode MSE : 0.0060, 0.0051, 0.0086 et 0.0145, et avec SSIM nous avons trouvé ces valeurs 0.9482, 0.9569, 0.9299 et 0.8981.
Et enfin avec l'utilisation de la distance euclidienne sans racine carrée comme perte de reconstruction, valeurs de performance calculées sont : pour la méthode d'erreur par pixel : 16.3549, 14.7368, 29.4467 et 30.3939 pour S1, S2, S3 et C, respectivement, la méthode MSE : 0.0041, 0.0033, 0.0133 et 0.0142, et avec SSIM nous avons trouvé ces valeurs 0.9659, 0.9717, 0.8912 et 0.9000.

Tous ces résultats sont résumés dans les tableaux ci-dessous.

TABLE 3.1 – Les valeurs de performance (la perte de reconstruction : MSE.)

	Erreur par pixel [0, 255]	MSE	SSIM
S1	19.5075	0.0058	0.9494
S2	18.7921	0.0054	0.9546
S3	21.6515	0.0072	0.9265
C	30.2061	0.0140	0.8741

TABLE 3.2 – Les valeurs de performance (la perte de reconstruction : MAE.)

	Erreur par pixel $[0, 255]$	MSE	SSIM
S1	23.5829	0.0085	0.9313
S2	16.2439	0.0040	0.9688
S3	30.2992	0.0141	0.8702
C	32.0534	0.0158	0.8684

TABLE 3.3 – Les valeurs de performance (la perte de reconstruction : la distance euclidienne.)

	Erreur par pixel $[0, 255]$	MSE	SSIM
S1	19.8646	0.0060	0.9482
S2	18.2634	0.0051	0.9569
S3	23.6769	0.0086	0.9299
C	30.7240	0.0145	0.8981

TABLE 3.4 – Les valeurs de performance (la perte de reconstruction : la distance euclidienne sans racine carrée.)

	Erreur par pixel [0, 255]	MSE	SSIM
S1	16.3549	0.0041	0.9659
S2	14.7368	0.0033	0.9717
S3	29.4467	0.0133	0.8912
C	30.3939	0.0142	0.9000

Dans ce qui suit, nous montrant les différents résultats selon la fonction de perte utilisée.
Certains résultats qualitatifs, en utilisant MSE, sont illustrés dans la figure 3.14.

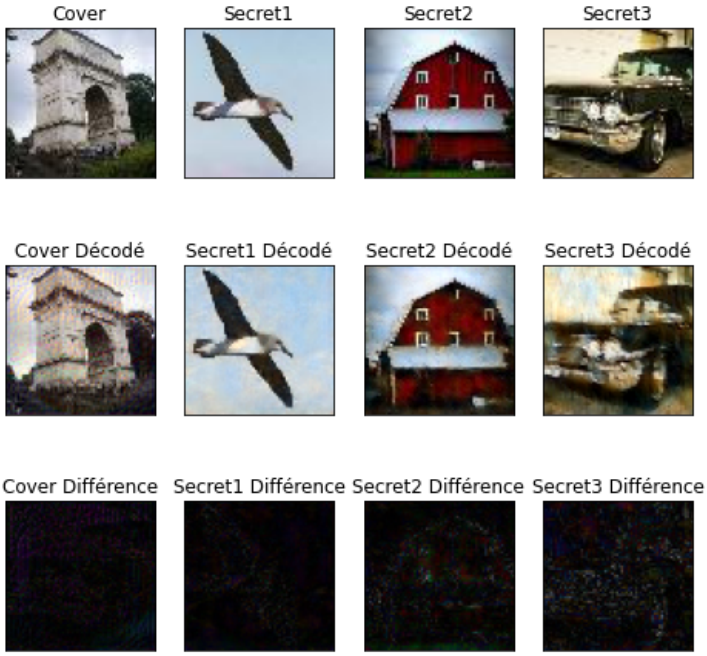

FIGURE 3.13 – Le résultat en utilisant MSE comme la perte de reconstruction.

Certains résultats qualitatifs, en utilisant MAE, sont illustrés dans la figure 3.15.

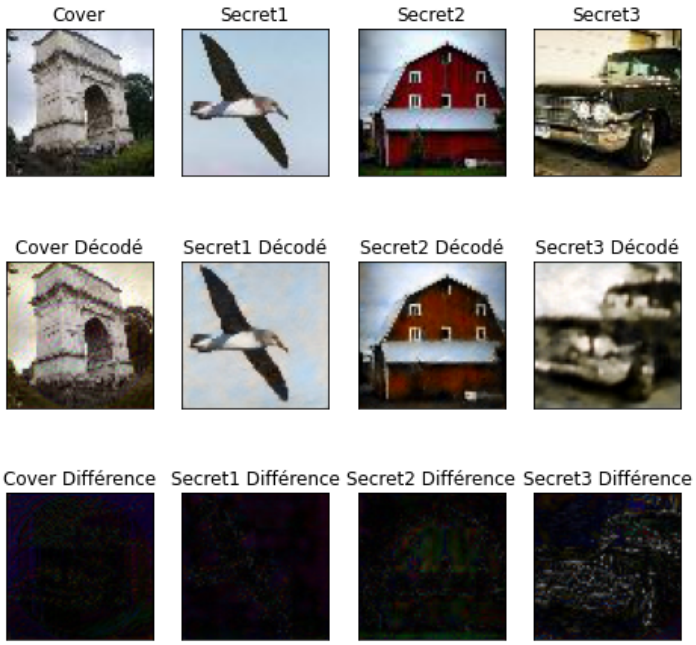

FIGURE 3.14 – Le résultat en utilisant MAE comme la perte de reconstruction.

Certains résultats qualitatifs, en utilisant la distance euclidienne, sont illustrés dans la figure 3.16.

FIGURE 3.15 – Le résultat en utilisant la distance euclidienne comme la perte de reconstruction.

Certains résultats qualitatifs, en utilisant la distance euclidienne sans racine carrée, sont illustrés dans la figure 3.17.

FIGURE 3.16 – Le résultat en utilisant la distance euclidienne sans racine carrée comme la perte de reconstruction.

"Différence" qui indiquait dans les figures ce-dissus, c'est les différences absolues entre la cible prévue et la cible réell.
D'après les résultats obtenus et les tableaux qui contiennent les valeurs de performance, nous avons remarqué que l'image de couverture et la troisième image secrète sont les plus nuisibles dans tous les cas de la fonction de perte de reconstruction.

3.7 Conclusion

Dans ce dernier chapitre nous avons illustré toutes les étapes de notre système de stéganographie, ainsi qu'un résumé des paramètres du modèle et des outils utilisés pour la mise en œuvre, nous avons aussi illustré les résultats obtenus.

Conclusion

Ce travail est une extension du modèle de masquage à une image, où il consiste à implémenter une architecture encodeur-décodeur pour la stéganographie et la stéganalyse du multiples images dans une image de couverture unique basée sur les réseaux de neurones à convolution (CNN).
Après avoir entraîné notre modèle sur quatre fonctions de perte de reconstruction différentes (MSE, MAE, la distance euclidienne et la distance euclidienne sans racine carrée) et évaluait ce modèle avec trois méthodes (Erreur par pixel, MSE, SSIM) ainsi que la perception visuelle, nous avons remarqué que l'image de couverture et la troisième image secrète sont les plus nuisibles, en particulier la troisième image secrète est la plus importante que la l'image de couverture, et qui doit ressembler le plus possible à l'image originale.
En ce qui concerne les perspectives de travaux futurs, notre idée et objectif est de prévoir d'affiner le modèle pour récupérer des images encodées avec une perte minimale, en utilisant d'autres fonctions de perte et d'autres architectures (avec GAN par exemple).

Bibliographie

[1] Adda Ali Pacha et al. “Stéganographie : Sécurité par Dissimulation”. In : (jan. 2006).

[2] Shumeet Baluja. “Hiding images in plain sight : Deep steganography”. In : *Advances in neural information processing systems* 30 (2017).

[3] Zhangjie Fu, Fan Wang et Xu Cheng. “The secure steganography for hiding images via GAN”. In : *EURASIP Journal on Image and Video Processing* 2020.1 (2020), p. 1-18.

[4] Ammouri Ahmed et Gada Hassina. “Réalisation d’un système de lecture des plaques d’immatriculation Algérienne”. Thèse de doct. Université Mouloud Mammeri, 2016.

[5] *Vision Par ordinateur*. Jan. 2022. URL : `https://fr.wikipedia.org/wiki/Vision_par_ordinateur`.

[6] HADJADJ OUALIE EDDINE. “Une approche pour la détection du texte dans les images”. In : (2021).

[7] André Marion. *Introduction aux techniques de traitement d’images*. Eyrolles, 1987.

[8] *mapschool*. URL : `https://mapschool.io/`.

[9] *http ://di.univ-blida.dz :8080/jspui/bitstream/123456789/11363/1/pfe.ouachemi-ouchfoun.classif* URL : `http://di.univ-blida.dz:8080/jspui/bitstream/123456789/11363/1/pfe.ouachemi-ouchfoun.classification.pdf`.

[10] Jonathan Weber. “Segmentation morphologique interactive pour la fouille de séquences vidéo”. Thèse de doct. Université de Strasbourg, 2011.

[11] Hadjar Yacine et Medrar Silia. “Véhicule intelligent pour la détection des plaques d’immatriculation suspectes.” Thèse de doct. Université Mouloud Mammeri, 2019.

[12] Sabah Delenda. “Méthodes pour la dissimulation d’information dans une image”. Thèse de doct. Université de Batna 2, 2019.

[13] Lotfi BERREZOUG et Ali BENYAGOUB. “Utilisatition des Algorithmes de L’Intéligence Artificielle appliquer à l’Aide à la Conduite”. Thèse de doct. Directeur : M. ABDELLAOUI Ghouti, 2021.

[14] Mohamadi Redha Badr Eddine Laati Taha et Mohamed Ayyoub Laksi Aymen. “Détection et lecteur de Qr code avec OpenCV”. Thèse de doct. Faculté des Sciences et Technologies, 2021.

[15] *La détection des objets dans des images*. URL : http://e-biblio.univ-mosta.dz/bitstream/handle/123456789/13141.

[16] MEKHALDI Nadia. “ANALYSE DES IMAGES MEDICALES PAR DES TECHNIQUES HYBRIDES”. Thèse de doct. Universite mohamed boudiaf’des sciences et de la technologie d’oran, 2014.

[17] *la stegéganologie - elearning-facsci.univ-annaba.dz*. URL : https://elearning-facsci.univ-annaba.dz/pluginfile.php/9429/mod_resource/content/1/Steganographie%20IATI.pdf.

[18] BENDJERIOU RAMZI et ARAR ABDELHAKIM. “Une étude comparative entre la stéganographie JPEG et la stéganographie tcp/ip pour une meilleure technique de dissimulation des données”. Thèse de doct. thèse master academique, Université Kasdi Merbah Ouergla, p (8, 9), 2016.

[19] Berkouk Nassima et Briouche Saida. “Prédiction de flux de trafic routier à l’aide de l’apprentissage profond.” Thèse de doct. Université Mouloud Mammeri, 2019.

[20] Zaimeche Mourad. “Apprentissage Automatique et Optimisation pour la Résolution de Problèmes.” Thèse de doct. Abdelhafid boussouf university Centre mila, 2015.

[21] Joseph Weizenbaum. “ELIZA—a computer program for the study of natural language communication between man and machine”. In : *Communications of the ACM* 9.1 (1966), p. 36-45.

[22] Olivier Delalleau. “Apprentissage machine efficace : théorie et pratique”. In : (2012).

[23] URL : http://biblio.univ-antananarivo.mg/pdfs/AndriamiandrisoaI_ESPA_MAST_2019.pdf.

[24] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[25] Geoffrey S Watson. “Smooth regression analysis”. In : *Sankhyā : The Indian Journal of Statistics, Series A* (1964), p. 359-372.

[26] Anselm Blumer et al. “Occam’s razor”. In : *Information processing letters* 24.6 (1987), p. 377-380.

[27] Ray J Solomonoff. “A formal theory of inductive inference. Part I”. In : *Information and control* 7.1 (1964), p. 1-22.

[28] Andrei Nikolaevic KOLMOGOROV. "Three approaches to the quantitative definition of information". In : *International journal of computer mathematics* 2.1-4 (1968), p. 157-168.

[29] Ming LI, Paul VITÁNYI et al. *An introduction to Kolmogorov complexity and its applications*. T. 3. Springer, 2008.

[30] Tilke JUDD et al. "Learning to predict where humans look". In : *2009 IEEE 12th international conference on computer vision*. IEEE. 2009, p. 2106-2113.

[31] Elizbar A NADARAYA. "On estimating regression". In : *Theory of Probability & Its Applications* 9.1 (1964), p. 141-142.

[32] Joseph WEIZENBAUM. "ELIZA—a computer program for the study of natural language communication between man and machine". In : *Communications of the ACM* 9.1 (1966), p. 36-45.

[33] Bogdan M WILAMOWSKI, Yixin CHEN et Aleksander MALINOWSKI. "Efficient algorithm for training neural networks with one hidden layer". In : *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. T. 3. IEEE. 1999, p. 1725-1728.

[34] Sara SABOUR, Nicholas FROSST et Geoffrey E HINTON. "Dynamic routing between capsules". In : *Advances in neural information processing systems* 30 (2017).

[35] Geoffrey E HINTON, Terrence J SEJNOWSKI et David H ACKLEY. *Boltzmann machines : Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.

[36] Geoffrey E HINTON, Terrence J SEJNOWSKI et al. "Learning and relearning in Boltzmann machines". In : *Parallel distributed processing : Explorations in the microstructure of cognition* 1.282-317 (1986), p. 2.

[37] Martin J WAINWRIGHT, Michael I JORDAN et al. "Graphical models, exponential families, and variational inference". In : *Foundations and Trends® in Machine Learning* 1.1–2 (2008), p. 1-305.

[38] Acervo LIMA. URL : `https://fr.acervolima.com/machine-boltzmann-restreinte/`.

[39] *QCM Didactique NSI*. URL : `http://multisiteeric.free.fr/QCM_NSI_algo/bases.html`.

[40] M ROSENBLATT. "Remarks on some nonparametric estimates of a density function. Annals of Mathematical Statistics". In : (1956).

[41] Emanuel PARZEN. "On estimation of a probability density function and mode". In : *The annals of mathematical statistics* 33.3 (1962), p. 1065-1076.

[42] *Tensorflow-GPU*. URL : `https://pypi.org/project/tensorflow-gpu/`..

[43] A. P. Dempster, N. M. Laird et D. B. Rubin. "Maximum-likelihood from incomplete data via the EM algorithm". In : *Journal of the Royal Statistical Society : Series B (Methodological)* 39.1 (1977), p. 1-22. doi : 10.1111/j.2517-6161.1977.tb01600.x.

[44] Bernhard E Boser, Isabelle M Guyon et Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In : *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, p. 144-152.

[45] Corinna Cortes et Vladimir Vapnik. "Support-vector networks". In : *Machine learning* 20.3 (1995), p. 273-297.

[46] Vladimir N Vapnik. *Lecture Notes in Economics and Mathematical Systems*. 1998.

[47] Bernhard Scholkopf, Alexander J Smola, Francis Bach et al. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[48] Leo Breimann et al. "Classification and regression trees". In : *Pacific Grove, Wadsworth* (1984).

[49] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[50] Lynda GASMI. "Deep Learning for face Recognition". Thèse de doct. FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE DEPARTEMENT D'INFORMATIQUE ..., 2020.

[51] Djaloul Youcef Moualek. "Deep Learning pour la classification des images". In : *Master's thesis* (2017).

[52] Rikiya Yamashita et al. "Convolutional neural networks : an overview and application in radiology". In : *Insights into imaging* 9.4 (2018), p. 611-629.

[53] *Python core team. url.* URL : https://www.python.org/.

[54] *matplotlib documentation. url.* URL : https://matplotlib.org/faq/usage_faq.html#coding-styles..

[55] *TensorFlow. url.* URL : https://pypi.org/project/tensorflow-gpu/..

[56] *Keras Team. keras. url.* URL : https://keras.io/.

[57] *NumPy. url.* URL : https://fr.wikipedia.org/wiki/NumPy.

[58] *what is Kaggle. url.* URL : https://www.datacamp.com/blog/what-is-kaggle.

[59] *Google colab. url.* URL : https://colab.research.google.com/.

[60] *CUDA Explained - Why Deep Learning uses GPUs. url.* URL : https://deeplizard.com/learn/video/6stDhEA0wFQ.