



MEMOIRE

Présenté à

**L'Institut Supérieur des Sciences Appliquées et de Technologie de
Gafsa**

(Département Informatique et télécommunication)
En vue de l'obtention Diplôme en MPSD

MASTERE

Dans la discipline Sciences des données

Par
Bilel Mahmoudi

**REALISATION D'UN MODELE DE SUPPRESSION DE
TATOUAGE DES IMAGES NUMERIQUES TATOUÉES A L'AIDE
D'UNE APPROCHE DE RESEAU DE NEURONES CONVOLUTIVE
BASE SUR L'AUTO-ENCODEUR**

Soutenu devant le jury composé de :

M.	Mohamed Wassim JMAL	<i>Président</i>
Mme	Fatma HRIZI	<i>Rapporteur</i>
M.	Ahmed REKIK	<i>Encadreur</i>
M.	Mounir TELLI	<i>Co-Encadreur</i>

A.U : 2022 – 2023

Remerciements

Je remercie Dieu le tout Puissant qui m'a donné la force et la volonté pour réaliser ce travail.

Je tiens à présenter ma reconnaissance et mes remerciements à mon encadreur « **Mr. Ahmed REKIK** » pour le temps consacré à la lecture et aux réunions qui ont rythmé les différentes étapes de cette mémoire.

Les discussions que nous avons partagées ont permis d'orienter notre travail d'une manière pertinente. Je le remercie aussi pour sa disponibilité à encadrer ce travail à travers ses critiques et ses propositions d'amélioration.

J'adresse aussi mes remerciements à tous mes professeurs de l'ISSAT pour la formation et l'expérience qu'ils m'ont transmise.

Enfin, Je tiens à adresser mes remerciements à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet.

Dédicaces

A mes chers parents

Respect, amour, reconnaissance, sont les moindres sentiments que je puisse vous témoigner, vous avez fait tout pour mon bonheur et ma réussite. Aucune dédicace ne saurait exprimer mon respect, ma considération et ma grande admiration.

Que dieu vous garde.

A mes très Chères sœurs et frères.

Vous m'avez toujours apporté l'affection, l'encouragement et le soutien moral, vous êtes les symboles de ma sûreté et ma sécurité.

A mes amis proches qui m'ont soutenu tout au long de ce travail.

A tous mes professeurs qui m'ont appris

Puisse ce modeste travail vous exprimer ma profonde reconnaissance, mon respect et mon Admiration sans limites à votre égard

Bilel Mahmoudi

Sommaire

Introduction Générale	1
Chapitre 1 : Etat du l'art.....	3
1. Introduction	4
2. Cadre général du projet	4
3. Problématique.....	5
4. Solution proposée et défit.....	5
5. Tatouage numérique	6
5.1. Cryptographie et Stéganographie	6
5.2. Historique de tatouage numérique.....	8
5.3. Type de tatouage.....	8
5.3.1. Tatouage visible	8
5.3.2. Tatouage invisible	8
5.4. Méthodes d'insertion	9
5.4.1. Méthodes additives.....	9
5.4.2. Méthodes substitutives.....	9
5.5. Critères de dissimulation d'information.....	10
5.5.1. Robustesse	10
5.5.2. Capacité.....	10
5.5.3. Imperceptibilité	10
6. Applications visées par le tatouage	11
6.1. Protection des droits d'auteur.....	11
6.2. Vérification de l'intégrité du contenu d'une image.....	11
6.3. Gestion du nombre de copies d'une image	12
6.4. Non répudiation d'accès et suivi de copies	12
7. Etude de quelques outils pour la suppression de tatouage numérique des images ...	13
8. Conclusion.....	14
Chapitre2 :Les réseaux de neurones convolutifs	15
1. Introduction	16
2. Généralités sur les réseaux de neurones	16
2.1. Neurone biologique	16
2.2. Réseau formel.....	17
2.3. Apprentissage des Réseaux de neurones artificiels.....	18
3. Définition CNN	19

4.	L'architecture des réseaux de neurones convolutifs (CNN)	20
4.1.	Couche entièrement connecté (Fully Connected)	21
4.2.	Couches convolutives	21
4.3.	Couches de pooling	21
4.4.	Couches totalement connectées	22
5.	Principe général de CNN.....	23
6.	Auto-encodeur (AE)	24
6.1.	Description des auto-encodeurs.....	24
6.2.	Architecture des auto-encodeurs	25
6.3.	Apprentissage des auto-encodeurs	25
7.	Conclusion.....	26
Chapitre3 :Expérimentations		27
1.	Introduction	28
2.	Les outils de développement	28
2.1.	PyCharm.....	28
2.2.	PyTorch	29
2.3.	Numpy	30
2.4.	Pandas	30
3.	Base d'apprentissage Utilisé	31
3.1.	Collecte des données	31
3.2.	. Préparation des données	34
4.	Modélisation les données	37
4.1.	Importation des bibliothèques	37
4.2.	Création de modèle.....	37
4.3.	Entraînement de modèle	39
4.4.	Résultats.....	40
5.	Conclusion.....	40
Conclusion Générale.....		41
Références bibliographie		42
Résumé.....		43
Abstract		43

Liste des figures

Figure 1 : Architecture de modèle	5
Figure 2 : Exemple d'image tatouée	7
Figure 3 : Exemple de tatouage visible	9
Figure 4 : Interface de GIMP	13
Figure 5 : Interface de Photoshop	14
Figure 6 : Le neurone biologique	16
Figure 7 : Schéma d'un neurone formel	17
Figure 8 : Différents types de fonctions de transfert pour le neurone artificiel	18
Figure 9 : Architecture des réseaux de neurones convolutifs (CNN)	20
Figure 10 : Principe de la couche entièrement connectée	21
Figure 11 : Max pooling	22
Figure 12 : Fonctionnement d'un CNN	23
Figure 13 : Schéma De L'architecture Qui Permettant De Faire La Fusion De Donnees Multi-Sources Dans Un Cnn	23
Figure 14 : Auto_Encodeur	24
Figure 15 : Logo pycharm	28
Figure 16 : Logo Pytorch	29
Figure 17 : Logo Pandas	30
Figure 18 : Site d'Unsplash pour la base de données	31
Figure 19 : La fonction collect_images()	32
Figure 20 : Le fichier "photos.tsv000 "	33
Figure 21 : Exemple des photos de base d'apprentissage	34
Figure 22 : La fonction de création de tatouage	35
Figure 23 : La classe Dataset()	36
Figure 24 : Code d'importation des bibliothèques	37
Figure 25 : Classe Générateur de modèle	38
Figure 26 : Classe discriminateur de modèle	38
Figure 27 : Code de training	39
Figure 28 : Exécution de 20 époques	39
Figure 29 : Résultats obtenus	40

Glossaire

DL : Deep learning

ML : Machine Learning

IA : Intelligence Artificielle

RN : Réseaux de neurones

CNN : Convolutionnel neural network

AE : Auto Enodeur

Introduction Générale

L'intelligence artificielle joue et continuera à jouer un rôle important dans nos vies quotidiennes. L'utilisation de méthodes, dans des systèmes d'intelligence artificielle, leur permettant d'imiter un être humain en termes de perception, d'apprentissage, de raisonnement et de compréhension du langage humaine change la donne dans le monde entier.

Nous constatons aujourd'hui une rupture dans les relations entre l'homme et la machine, rupture permise par l'intelligence artificielle qui met l'humain en position de déléguer certaines actions à ces machines autonomes. Depuis l'avènement des ordinateurs, la tendance dans les interactions entre l'homme et la machine a été de placer le plus gros de la communication sur la machine plutôt que sur l'homme.

Le monde entier vit l'ère du tout numérique, où l'imagerie prend une place primordiale dans la société, parce que devenue incontournable dans les activités socio-économiques et culturelles quotidiennes et notamment dans la communication. Dès lors, les nouveaux défis des chercheurs et ingénieurs du domaine de l'imagerie et de la vidéo numériques, consistent à chercher une amélioration continue de la qualité de service des équipements d'imagerie et de vidéos numériques. De plus, nous avons constaté depuis quelques temps sur les photos des appareils photographiques, les photos contenues sur la toile et les applications de capture, que la majorité de ces photos comporte un titre ou une expression sous forme, soit d'un nom de marque ou un logo, soit une signature ou une adresse d'un site web. Ces motifs sont appelés filigranes ou tatouages. Au départ, ils ont été conçus dans le but de protéger les photos et les images contre les falsifications et les vols. Mais ces tatouages peuvent-être un obstacle pour qui, veut utiliser ou exploiter ces images.

De ce qui précède, notre but est de supprimer le tatouage de manière automatique en utilisant Deep Learning, c'est-à-dire, de façon plus explicite, nous concevons des réseaux de neurones convolutionnels capables de supprimer cette marque et d'effectuer un 'montage' qui soit le plus fidèle possible à la photo originale, sans passer par cette même photo de départ, tout en conservant sa qualité.

Le deep learning ou apprentissage profond est un type d'intelligence artificielle dérivé du machine learning (apprentissage automatique) où l'ordinateur est capable d'apprendre par lui-même, contrairement à la programmation où l'ordinateur se contente d'exécuter des règles déterminées au préalable. Cet apprentissage profond est basé sur un réseau de neurones artificiels. Il est composé de dizaines, parfois de centaines de couches de neurones, chacune recevant les informations de la couche précédente. Le système apprendra par exemple à déterminer s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit, ou de reconnaître les lettres avant de s'attaquer aux mots dans un texte.

Pour cela nous avons partagé notre travail en trois grandes parties organisées comme suit :

- Premier chapitre est consacré de la préparation d'une étude bibliographique sur le tatouage numérique. Nous sommes intéressées dans ce chapitre aux différents types d'insertion et nous avons présenté deux méthodes pour la suppression de filigrane.
- Dans le deuxième chapitre, nous présentons un aperçu général sur le domaine des réseaux de neurones artificiels. Nous y présentons les concepts de bases du perceptron, du perceptron multicouches, la rétro propagation et les différents types de réseaux de neurones et nous abordons l'évolution majeure de ces dernières années : les réseaux de neurones profonds (deep learning). Nous détaillons l'architecture typique d'un réseau de neurones convolutionnel, qui est une partie fondamentale des réseaux profonds, par la présentation des couches qui le constituent.
- Enfin, le dernier chapitre, nous mettrons en pratique toutes les connaissances et les techniques que nous avons apprises afin de créer un modèle de suppression de tatouage numérique en utilisant le Deep Learning et spécifiquement les réseaux de neurones convolutifs.

Chapitre 1

Etat de l'art

1. Introduction

Le tatouage numérique est une sorte de marqueur incorporé secrètement dans un signal tolérant au bruit tel que des données audio, vidéo ou image. Il est généralement utilisé pour identifier la propriété du droit d'auteur d'un tel signal.

Dans ce chapitre nous allons présenter une étude bibliographique sur le tatouage numérique ainsi qu'une étude technologique sur les approches de la suppression de tatouage numérique.

2. Cadre général du projet

Le tatouage numérique est le processus consistant à cacher des informations numériques dans un signal porteur ; les informations cachées devraient, mais ne doivent pas nécessairement, contenir une relation avec le signal de porteuse. Les filigranes numériques peuvent être utilisés pour vérifier l'authenticité ou l'intégrité du signal porteur ou pour montrer l'identité de ses propriétaires. Il est largement utilisé pour localiser les violations de droits d'auteur et pour l'authentification des billets de banque. Comme les filigranes physiques traditionnels, les filigranes ou tatouage numériques ne sont souvent perceptibles que dans certaines conditions, par exemple après avoir utilisé un algorithme. Si un tatouage numérique déforme le signal porteur de manière qu'il devienne facilement perceptible, il peut être considéré comme moins efficace en fonction de son objectif. Les filigranes traditionnels peuvent être appliqués aux supports visibles (comme les images ou la vidéo), tandis que dans le filigrane numérique, le signal peut être audio, images, vidéo, textes ou modèles 3D. Un signal peut porter plusieurs filigranes en même temps [1].

Le tatouage a été conçu dans le but de protéger les photos et les images contre les falsifications et les vols et il est habituellement utilisé pour empêcher des images et des photos d'être réutilisées sans la permission de leurs propriétaires. Ils s'avèrent parfois difficiles à supprimer.

Dans le cadre de notre étude en Mastère Professionnelle en Sciences des Données à l'ISSAT de Gafsa, nous allons développer un projet qui s'intitule : Réalisation d'un modèle de suppression de tatouage numérique basé sur les réseaux de neurones convolutifs. Donc, nous allons par la suite essayer de comprendre les notions fondamentales de l'apprentissage artificielle (IA) pour arriver à créer ce modèle.

3. Problématique

Les filigranes sont habituellement utilisés pour empêcher des images et des photos d'être réutilisées sans la permission de leurs propriétaires. Ils s'avèrent parfois difficiles à supprimer. Et à nos recherches, nous trouvons que l'intelligence artificielle est un phénomène d'actualité d'aujourd'hui qui ne cesse de développer et d'évoluer puisque on trouve des nouveautés dans ce domaine presque chaque jour, c'est pour cela nous trouvons plusieurs entreprises investissent dans le système afin de faciliter leurs activités commerciales et répondre aux demandes de leurs clients.

Notre question est « comment réaliser un modèle de suppression de tatouage numérique ? » ce système doit être un outil qui permet à un utilisateur d'interagir avec un système informatique. Son but est de répondre aux demandes et il est capable de faire une tâche de suppression.

4. Solution proposée et défi

Pour faire face aux problèmes qui existent, nous avons besoin d'un système qui automatise toutes les tâches de suppression. A ce niveau nous proposons une solution améliorée et fiable qui prend en compte tous les exigences actuelles de la technologie et qui permettra de répondre aux futurs besoins des utilisateurs. Et en accordant une grande attention aux systèmes de suppression automatique qui peuvent devenir une solution puissante pour certains cas comme les images qui sont tatouées par des longues textes.

La figure suivante représente l'architecture générale de notre modèle :

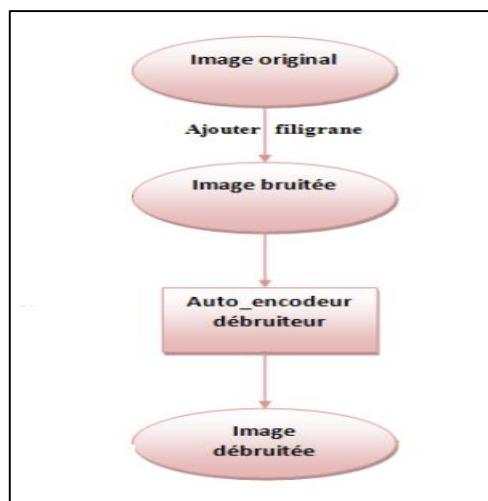


FIGURE 1 : Architecture de modèle

5. Tatouage numérique

5.1. Cryptographie et Stéganographie

La cryptologie existe depuis des siècles. Depuis l'invention de l'écriture, le besoin de sécurité est motivé par les problèmes de confidentialité et d'intégrité : on souhaite éventuellement que l'information écrite ne soit accessible qu'à certaines personnes et qu'elle ne soit pas modifiée volontairement dans un but de mystification. La cryptologie regroupe à la fois la cryptographie, qui désigne l'art de chiffrer le contenu d'un message susceptible d'être intercepté lors de sa transmission, et la cryptanalyse, qui consiste à casser le code protégeant un message chiffré. Depuis son origine, où elle était principalement réservée à un usage militaire et diplomatique, la cryptologie a considérablement évolué, notamment avec l'apparition de l'ordinateur, et s'étend aujourd'hui au domaine civil pour la protection des données circulant sur les réseaux informatiques. Ainsi, la cryptologie moderne est maintenant une discipline de recherche publique de l'informatique théorique utilisant des outils mathématiques sophistiqués [2].

Le « watermarking » (littéralement filigrane) ou tatouage d'image peut être perçu comme une branche de la stéganographie. Le mot stéganographie vient du grec « steganos » (caché ou secret) et « graphy » (écriture ou dessin), et signifie littéralement « écriture cachée ».

La stéganographie consiste à cacher, de manière subliminale, un message secondaire dans un message primaire. Le message primaire reste lisible de tous, tandis que le message secondaire n'est lisible que par une ou plusieurs personnes propriétaires d'une information secrète. Pour la petite histoire, les premières traces de stéganographie remontent à l'Antiquité. Un légataire romain voulant envoyer un message à César, le camoufla dans une amphore qu'il lui envoya en guise de cadeau. Une autre forme de stéganographie, elle aussi très rudimentaire, consistait à raser le crâne d'un esclave. On y tatouait alors le message, et l'esclave était envoyé lorsque ses cheveux avaient repoussé. Le destinataire n'avait plus qu'à le faire raser de nouveau pour faire apparaître le message. Une autre forme de stéganographie très connue est le principe de l'encre invisible. Cette technique était très utilisée au moyen âge pour envoyer des messages secrets. A l'époque, l'encre était fabriquée simplement à base de jus d'oignons et de chlorure d'ammoniac. L'écriture était alors rendue visible en approchant le papier d'une flamme de bougie. La stéganographie se distingue de la cryptographie dans la mesure où l'objectif principal de la cryptographie est de rendre illisible

le message à toute personne ne possédant pas l'information secrète adéquate. De plus, alors que la cryptographie offre une sécurité plutôt a priori (par exemple, contrôle d'accès), la stéganographie offre une sécurité plutôt a posteriori, dans la mesure où le message secondaire est supposé rester accessible après recopies et manipulations du message primaire

Un tatouage physique est une image ou un motif d'identification sur le papier qui apparaît sous la forme de diverses nuances de clarté / d'obscurité lorsqu'il est vu par lumière transmise (ou lorsqu'il est visualisé par la lumière réfléchie, sur un fond sombre), causé par des variations d'épaisseur ou de densité du papier. Des filigranes ont été utilisés sur des timbres-poste, des devises et d'autres documents gouvernementaux pour décourager la contrefaçon. Les tatouages varient considérablement dans leur visibilité ; tandis que certains sont évidents lors d'une inspection occasionnelle, d'autres nécessitent une étude pour être repérés. Un tatouage numérique est très utile dans l'examen du papier car il peut être utilisé pour dater, identifier les formats, les marques de fabrique et les emplacements, et déterminer la qualité d'une feuille de papier [1].



FIGURE 2 : Exemple d'image tatouée

5.2. Historique de tatouage numérique

Le tatouage numérique a été introduits pour la première fois à Fabriano, en Italie, en 1282[17] sur papier, mais Le terme tatouage numérique a été inventé par Andrew Tirkel et Charles Osborne en décembre 1992 [18]. La première intégration et extraction réussies d'un filigrane à spectre étalé stéganographique a été démontrée en 1993 par Andrew Tirkel, Charles Osborne et Gerard Rankin [18]. Les tatouages continuent d'être utilisés aujourd'hui comme marques de fabricant et pour empêcher la falsification.

5.3. Type de tatouage

5.3.1. Tatouage visible

Le tatouage visible est très simple .il altère le signal ou le fichier par exemple ajout d'une image pour en marquer une autre. Il est fréquent que les agences de photo ajoutent un filigrane visible en forme de copyright aux versions de prévisualisation basse résolution de leurs photos. Ceci afin d'éviter que ces versions ne se substituent aux versions hautes résolutions payantes. Le tatouage visible est un sujet à controverse. Il y a une branche de chercheurs qui disent que si le tatouage est visible, alors elle peut être facilement attaquée. Néanmoins, nous trouvons des applications qui demandent que le tatouage soit visible, c'est le cas du logo des sociétés dans les programmes télévisuels.

Dans les techniques de filigrane visible :

- La marque insérée est facilement enlevée par un simple cropping.
- La visibilité de la marque insérée dégrade la qualité visuelle de l'image

5.3.2. Tatouage invisible

Le tatouage invisible est un concept beaucoup plus complexe. Le tatouage invisible modifie le signal d'une manière imperceptible par l'utilisateur final. Par exemple, dans l'agence des photos, les photos hautes résolutions vendues par l'agence possèdent elles au contraire un tatouage invisible, qui ne dégrade donc pas le contenu visuel.

Dans un tatouage invisible :

- il n'est pas facile de faire la distinction entre l'image originale et l'image filigranée.
- il est difficile d'enlever ou détruire la marque insérée sans avoir une dégradation de la qualité visuelle de l'image filigranée de manière significative.

Dans la figure suivante, un exemple de comparaison entre tatouage invisible et visible :

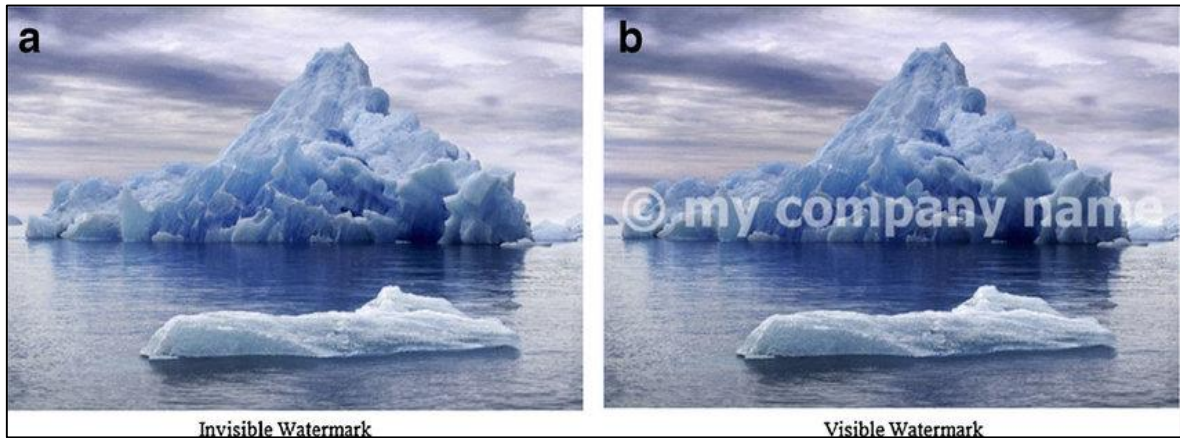


FIGURE 3 : EXEMPLE DE TATOUAGE VISIBLE

5.4. Méthodes d'insertion

5.4.1. Méthodes additives

Lors de l'insertion, le signal représentant la marque est ajouté à certaines composantes du médium. Pour y parvenir, il s'agit d'adapter la marque au médium, afin que le signal qu'elle représente ne soit ni trop faible (risques de non-détectabilité et problèmes de robustesse), ni trop fort (effacement du signal initial, et donc trop grande dégradation de celui-ci).

5.4.2. Méthodes substitutives

Dans les méthodes substitutives, la marque à insérer est substituée à des composantes de l'image originale, ce qui correspond principalement à deux comportements : le premier est un tatouage quantitatif appelé aussi tatouage substitutif avec dictionnaire, le second est un tatouage substitutif avec contraintes. Il consiste à imposer un ensemble de contraintes aux données marquées. Plusieurs techniques substitutives ont été proposées dans la littérature. La première méthode consiste à remplacer les bits les moins significatifs ou les bits de poids faibles des pixels d'une image par les bits de la marque. Cette méthode a été améliorée dans pour les images couleurs. Une technique très utilisée dans le tatouage des images est appelée la quantification par la modulation d'index (QIM) . Le principe de cette technique consiste à quantifier l'image en utilisant un ensemble de quantificateurs indexés par la marque à insérer. Ainsi, chaque élément de la marque est attaché à un quantificateur différent, et le

tatouage s'effectue par quantification de l'image avec le quantificateur correspondant à la marque.

5.5. Critères de dissimulation d'information

Les applications de dissimulation d'information sont triées en fonction de trois critères :

5.5.1. Robustesse

C'est le pouvoir de récupérer la marque insérée même si l'image filigranée a été manipulée par des attaques. Il est nécessaire de distinguer plusieurs types d'attaques selon qu'elles sont considérées comme étant bienveillantes ou malveillantes. Les attaques bienveillantes sont les manipulations effectuées de bonne foi par un utilisateur. On retrouve dans cette catégorie : la compression JPEG, certaines transformations géométriques, le filtrage spatial et fréquentiel, l'ajout de bruit, l'impression et la numérisation, la correction gamma et l'égalisation d'histogramme.

5.5.2. Capacité

Elle représente la quantité d'information que l'on veut insérer dans une image. Cette quantité varie selon l'application. En général, quelques bits suffisent pour la protection du droit d'auteurs à l'aide d'un identifiant, mais pas pour insérer un logo de société. En revanche, il est nécessaire de cacher plusieurs bits d'information pour permettre l'authentification des images.

5.5.3. Imperceptibilité

Le tatouage numérique va certainement introduire des distorsions. Cette contrainte exige que lesdites distorsions soient les plus faibles possibles afin que visuellement l'image filigranée reste fidèle à l'image originale. Pour ce faire, les caractéristiques du système visuel humain (SVH) peuvent être exploitées pour rendre la marque moins perceptible. La qualité de l'image tatouée par rapport à l'image originale peut être évaluée à l'aide d'outils mathématiques tels que le rapport, signal sur bruit de crête (PSNR), la similitude structurale (SSIM), etc. Le critère d'imperceptibilité est une propriété liée uniquement au tatouage numérique invisible.

6. Applications visées par le tatouage

6.1. Protection des droits d'auteur

La protection des droits d'auteur a été une des premières applications étudiées en tatouage d'image. Ce service reste cependant toujours d'actualité et concerne encore la majorité des publications. L'objectif est d'offrir, en cas de litige, la possibilité à l'auteur ou au propriétaire d'une image d'apporter la preuve qu'il est effectivement ce qu'il prétend être, et ce même si l'image concernée a subi des dégradations par rapport à l'original.

La mise en place d'un tel service doit respecter les trois contraintes suivantes : **Préserver la qualité de l'image** : la distorsion liée à l'insertion de la marque dans l'image doit être la plus faible possible, de manière que la qualité visuelle de l'image tatouée soit quasi identique à l'originale. Malheureusement cette notion d'invisibilité est très subjective et difficile à modéliser. D'autant plus qu'elle dépend de nombreux facteurs, tels que : la nature de l'image à tatouer (peinture, image médicale, photo satellite, etc.), la qualité du document original (plus une image est de bonne qualité, plus il est difficile de garantir l'invisibilité de la marque) et des conditions de visualisation (problème bien connu en codage de source avec pertes).

Garantir la non-ambiguïté de la preuve : le tatouage doit constituer une preuve irréfutable. Pour cela il convient d'assurer l'unicité (éviter les problèmes de collision) et de l'authenticité de l'identifiant, mais également de dater le dépôt (au cas où l'image aurait été tatouée plusieurs fois avec des marques différentes). Pour cela, il convient de définir des protocoles stricts excluant toute ambiguïté.

Assurer la robustesse des éléments de preuve (tatouage) : l'algorithme utilisé doit être capable d'extraire correctement la marque cachée, même si l'image a été manipulée.

6.2. Vérification de l'intégrité du contenu d'une image

L'idée de base consiste à utiliser les techniques de tatouage d'image afin de cacher dans certaines zones de l'image des informations sur d'autres zones. Ces informations servent à alerter l'utilisateur face à une éventuelle modification ou découpe de l'image par une personne non autorisée et à localiser précisément les régions manipulées, voire éventuellement à les restaurer. Ce service remet partiellement en cause les paramétrages usuellement établis dans le cadre d'un service classique de droits d'auteur, notamment en

termes de quantité et nature des informations à cacher. Mais on peut déjà se demander s'il est préférable d'utiliser un tatouage fragile (ou semi-fragile), un tatouage robuste, ou opter au contraire pour une technique faisant appel à une signature externe.

6.3. Gestion du nombre de copies d'une image

Contrairement aux données de nature analogique pour lesquelles une succession de reproductions entraîne rapidement une perte significative de la qualité, les données numériques peuvent être dupliquées quasiment à l'infini. Dans ce contexte, une personne ayant accès à ce type de données et au matériel adéquat, est potentiellement capable de les reproduire bit à bit à l'identique. Il est évident que cette personne, si elle est malintentionnée, peut ensuite redistribuer illégalement des copies avec une qualité égale au document d'origine. Face à cette nouvelle situation, certaines instances proposent d'utiliser des techniques de tatouage afin de limiter l'ampleur de ce phénomène. C'est le cas des systèmes DVD, où un filigrane numérique indiquera si la vidéo peut être lue et/ou recopiée. Ce procédé n'est bien entendu fiable que si tous les constructeurs de lecteurs et d'enregistreurs de DVD tiennent compte de l'indicateur de copie.

6.4. Non répudiation d'accès et suivi de copies

Ce service est une bonne illustration de la complémentarité entre les techniques de cryptographie et le tatouage d'image. Soit une image diffusée sous forme chiffrée, interdisant qu'une personne puisse avoir accès à son contenu lors de la transmission de celle-ci. Au niveau du destinataire, on procède simultanément au décryptage et au tatouage de l'image de telle sorte que si l'utilisateur remet illégalement en circulation cette image, il sera alors possible de remonter à la source du délit grâce à l'identifiant du destinataire (ou « fingerprint ») caché dans l'image. Ce type d'application ne va pas sans poser de sérieux problème en matière de sécurité, dans la mesure où il existe plusieurs copies d'une même image contenant des tatouages différents. En effet, sous l'hypothèse que l'ensemble des contributions des tatouages soit à moyenne nulle, plusieurs clients malhonnêtes peuvent tenter de reconstruire l'image originale en calculant une image moyenne à partir de leur image respective.

7. Etude de quelques outils pour la suppression de tatouage numérique des images

Il existe plusieurs applications pour supprimer un tatouage sur une image. Nous allons citer quelque application :

🌈 **Gimp** : Le programme GIMP ou GNU (General Public License) Manipulation de l'image, un logiciel open-source gratuit qui peut être téléchargé à partir de gimp.org à beaucoup des mêmes caractéristiques qu'un programme professionnel, propriétaire de retouche d'image, diffusé sous la licence GPLv3 comme un logiciel gratuit et libre. Il en existe des versions pour la plupart des systèmes d'exploitation dont GNU/Linux, macOS et Microsoft Windows. Si un tatouage est créé sur une couche dans une image, vous pouvez supprimer la couche de filigrane en utilisant GIMP. Toutefois, si le filigrane est une partie de la couche d'image elle-même, la seule façon de supprimer le filigrane est de modifier l'image [3].



FIGURE 4 : INTERFACE DE GIMP

🎨 **Photoshop** : Les prémices de Photoshop remontent toutefois à 1987, lorsque Thomas Knoll, doctorant à l'Université du Michigan, inventa le logiciel Display pour ses propres besoins de recherches sur l'affichage sur ordinateurs. C'est dans un deuxième temps qu'il s'est associé à son frère John Knoll pour créer la première version de Photoshop, Photoshop 0.87, livrée en exclusivité avec des scanners. La société fut alors rachetée par Adobe en avril 1989, et l'entreprise lança dans le commerce Photoshop 1.0 en 1990 sur MacOS puis en 1992 sur Windows [4].

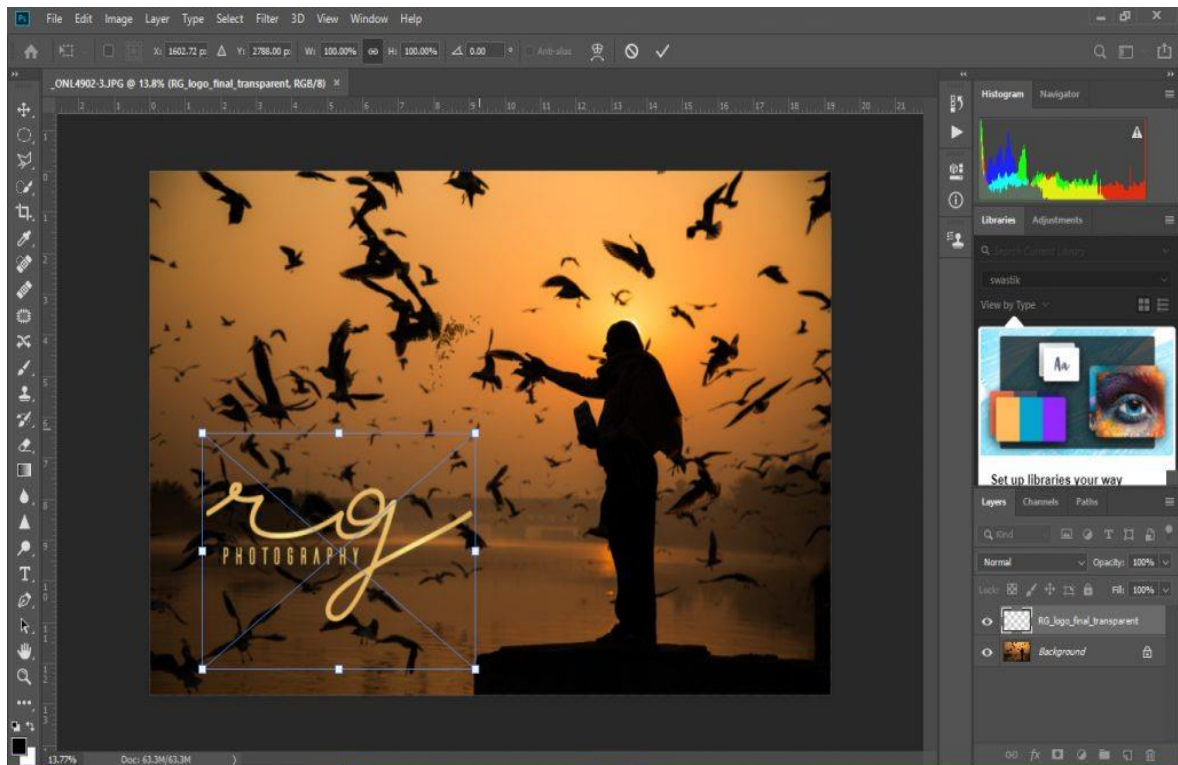
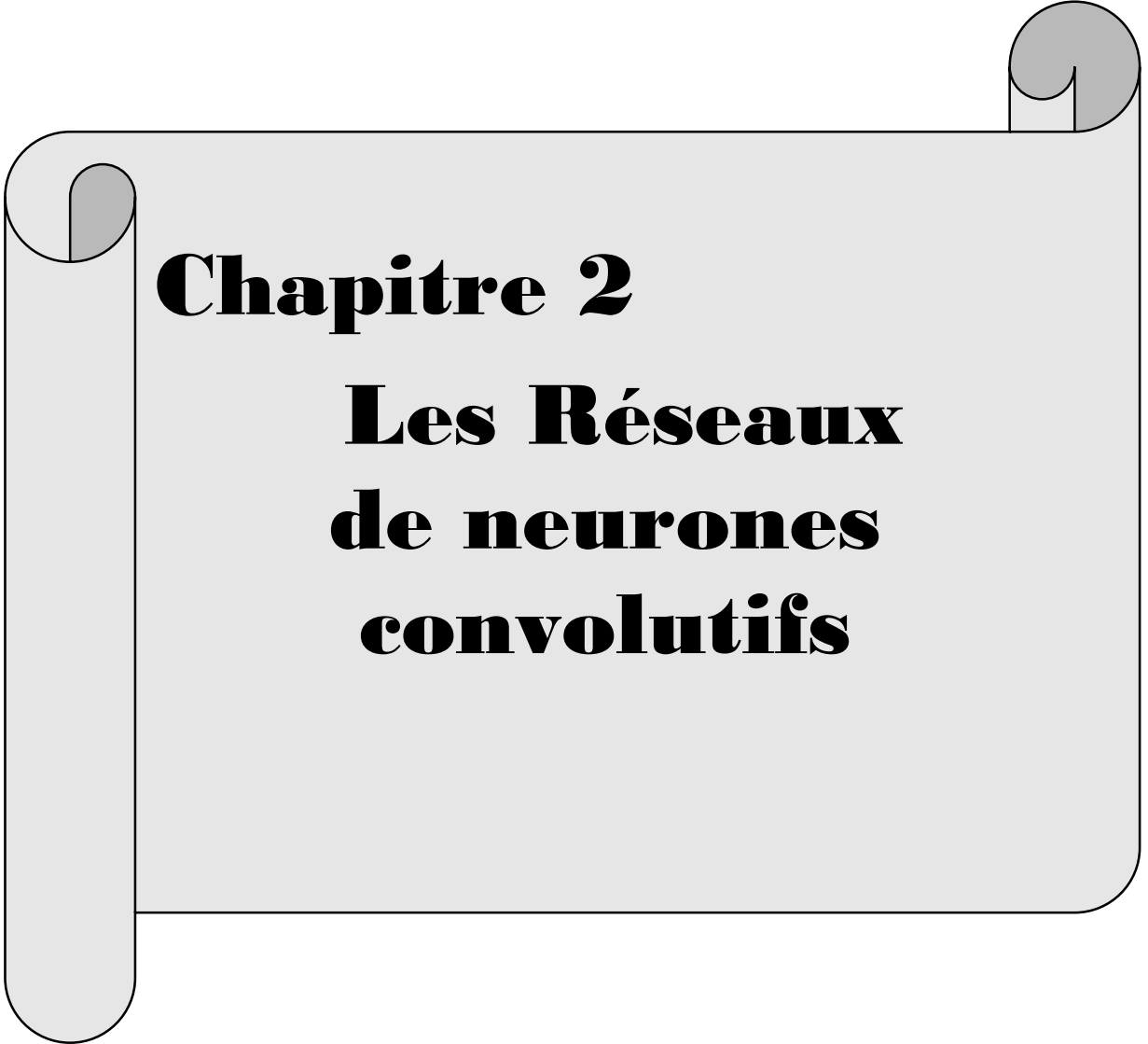


FIGURE 5 : INTERFACE DE PHOTOSHOP

8. Conclusion

Dans ce chapitre, nous avons présenté le cadre dans lequel, notre projet sera mis en place. Ainsi que le tatouage des images numériques. Nous sommes intéressés aux types d'insertion du tatouage numérique. Nous avons abordé les Types de tatouage et présentées des applications de suppression d'un tatouage.



Chapitre 2

Les Réseaux de neurones convolutifs

1. Introduction

Les réseaux de neurones convolutifs ont une méthodologie similaire à celle des méthodes traditionnelles d'apprentissage supervisé : ils reçoivent des images en entrée, détectent les features automatiquement de chacune d'entre elles, puis entraînent un classifieur dessus.

Donc les CNN réalisent eux-mêmes tout le boulot fastidieux d'extraction et description de features. Lors de la phase d'entraînement, l'erreur de classification est minimisée afin d'optimiser les paramètres du classifieur et les features. De plus, l'architecture spécifique du réseau permet d'extraire des features de différentes complexités, des plus simples au plus sophistiquées. L'extraction et la hiérarchisation automatiques des features, qui s'adaptent au problème donné, constituent une des forces des réseaux de neurones convolutifs. Aujourd'hui, les réseaux de neurones convolutifs, aussi appelés CNN ou ConvNet pour Convolutional Neural Network, sont toujours les modèles les plus performants pour la classification d'images. Cette partie leur est donc naturellement consacrée.

2. Généralités sur les réseaux de neurones

2.1. Neurone biologique

Dans le cerveau humain, le neurone est l'élément de base. Il reçoit des signaux en provenance de neurones voisins, les traite, engendre, conduit et transmet l'influx nerveux à d'autres neurones. La figure suivante représente les éléments d'un neurone biologique [1].

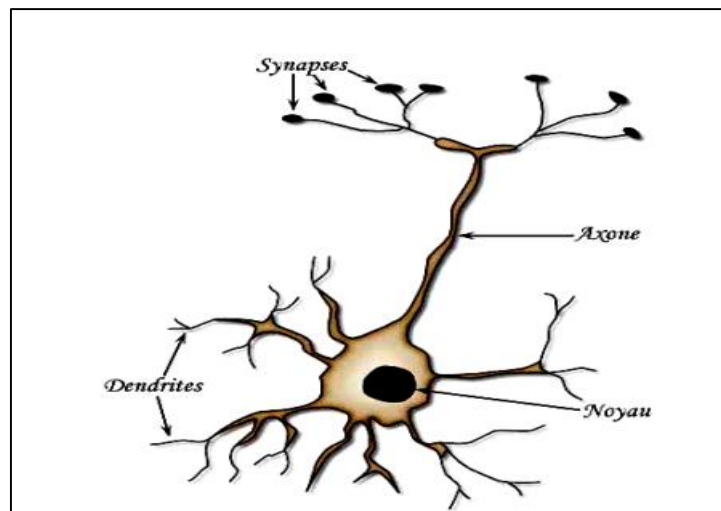


FIGURE 6 : LE NEURONE BIOLOGIQUE

Le neurone est constitué de :

- ✓ Les dendrites : ce sont les récepteurs principaux du neurone pour capter les signaux qui lui parviennent.
- ✓ Le corps cellulaire : Il contient le noyau. C'est un sommateur à seuil. Il effectue une sommation des influx nerveux transmis par ses dendrites. Si la somme est supérieure au seuil, le neurone répond par un influx nerveux ou potentiel d'action qui se propage le long de son axone. Si la somme est inférieure au seuil, il reste inactif.
- ✓ L'axone : Il sert de moyen de transport pour les signaux émis par le neurone. Il se ramifie à son extrémité, là où il communique avec d'autres neurones.
- ✓ Les synapses : Ils permettent aux cellules nerveuses de communiquer entre elles. Les synapses se rencontrent surtout entre les axones et les dendrites.

2.2. Réseau formel

Les réseaux de neurones formels sont à l'origine d'une tentative de modélisation mathématique du cerveau humain. Ils présentent un modèle assez simple pour les neurones et explorent les possibilités de ce modèle. Le neurone représenté par la figure qui suit, se compose d'une cellule possédant plusieurs entrées et une sortie.

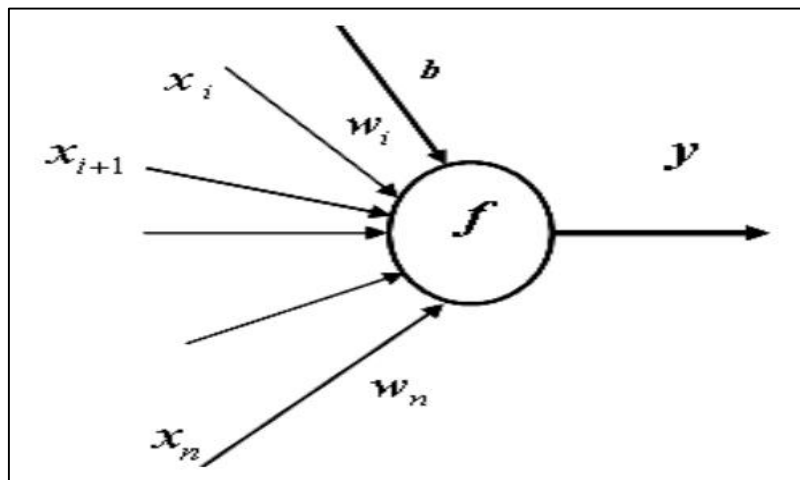


FIGURE 7 : Schéma d'un neurone formel

Avec : $y = f(\sum_{i=1}^n w_i x_i) + b$

- y : La sortie du neurone.
- f : La fonction d'activation (ou de transfert).
- x : Entrée du neurone.
- w : Poids synaptique du neurone.
- b : Biais.

Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont présentées sur la figure 8.

On remarquera qu'à la différence des neurones biologiques dont l'état est binaire, la plupart des fonctions de transfert sont continuës, offrant une infinité de valeurs possibles comprises dans l'intervalle $[0, +1]$ ou $[-1, +1]$.

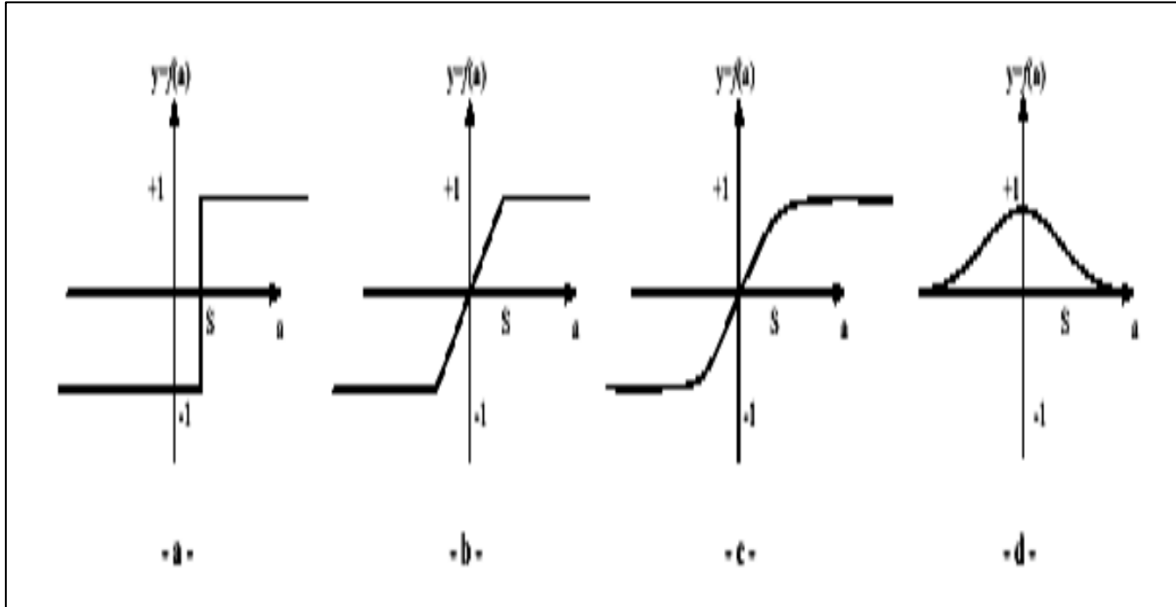


FIGURE 8 : DIFFERENTS TYPES DE FONCTIONS DE TRANSFERT POUR LE NEURONE ARTIFICIEL

2.3. Apprentissage des Réseaux de neurones artificiels

L'apprentissage des Réseaux de neurones artificiels (RNA), est une phase qui permet de déterminer ou de modifier les paramètres du réseau, afin d'adapter un comportement désiré. Plusieurs algorithmes d'apprentissage ont été développés depuis la première règle d'apprentissage de Hebb en 1949 [5].

Les RNA se divisent en deux principales classes, les réseaux à apprentissage supervisés et les réseaux à apprentissage non supervisés.

- Pour les réseaux à apprentissage supervisés, on présente au réseau des entrées, et au même temps les sorties que l'on désirerait pour cette entrée. Le réseau doit alors se reconfigurer, c'est-à-dire calculer ses poids afin que la sortie qu'il donne corresponde bien à la sortie désirée.
- Pour les réseaux à apprentissage non supervisé, on présente une entrée au réseau et on le laisse évoluer librement jusqu'à ce qu'il se stabilise.

L'apprentissage des RNA se fait selon la stratégie suivante [6] :

- ❖ Etape 1 : - Les entrées d'un exemple sont injectées dans les cellules d'entrée du RNA. Le RNA calcule alors une sortie en fonction des états internes des neurones et des poids synaptiques des connexions.
- ❖ Etape 2 : - La sortie obtenue est comparée avec la sortie désirée pour l'exemple. On obtient alors une erreur, particulièrement importante en début d'apprentissage car la réponse du RNA est à ce stade purement aléatoire.
- ❖ Etape 3 : Les poids synaptiques du RNA sont ensuite modifiés afin de réduire l'erreur calculée.
- ❖ Etape 4 : On réitère cette opération un grand nombre de fois, et ce pour chaque exemple de la base, jusqu'à ce que le RNA converge vers une configuration qui lui permette de résoudre le problème à traiter.

3. Définition CNN

Les réseaux de neurones convolutifs désignent une sous-catégorie de réseaux de neurones. Cependant, les CNN sont spécialement conçus pour traiter des images en entrée. Leur architecture est alors plus spécifique : elle est composée de deux blocs principaux.

1- Le premier bloc fait la particularité de ce type de réseaux de neurones, puisqu'il fonctionne comme un extracteur de features. Pour cela, en appliquant des opérations de filtrage par convolution. La première couche filtre l'image avec plusieurs noyaux de convolution, et renvoie des « feature maps », qui sont ensuite normalisées (avec une fonction d'activation) et/ou redimensionnées. Ce procédé peut être réitéré plusieurs fois : on filtre les features maps obtenues avec de nouveaux noyaux, ce qui nous donne de nouvelles features maps à normaliser et redimensionner, et qu'on peut filtrer à nouveau, et ainsi de suite. Finalement, les valeurs des dernières feature maps sont concaténées dans un vecteur. Ce vecteur définit la sortie du premier bloc, et l'entrée du second.

2- Le second bloc : Les valeurs du vecteur en entrée sont transformées (avec plusieurs combinaisons linéaires et fonctions d'activation) pour renvoyer un nouveau vecteur en sortie. Ce dernier vecteur contient autant d'éléments qu'il y a de classes : l'élément i représente la probabilité que l'image appartienne à la classe i .

4. L'architecture des réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs sont de loin le modèle le plus efficace pour classer les images.

Il y a plusieurs couches différentes dans CNN comme le montre la figure 9 :

- Couche d'entrée (Input layer).
- Couche de convolution (Convo layer : Convolution + ReLU).
- Couche de Pooling.
- Couche entièrement connectée (Couche Fullyconnected).
- Couche Softmax/logistique.
- Couche de sortie (Output layer).

Désignés par l'acronyme CNN, ils se composent de deux parties distinctes. En entrée, une image est fournie sous la forme d'une matrice de pixels. Elle a deux dimensions pour une image en niveaux de gris. La couleur est représentée par une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales.

La première partie de CNN est la partie convolutive elle-même (**figure 9**). Elle fonctionne comme un extracteur de caractéristiques des images. Une image est passée à travers une succession de couches, ou noyaux de convolution, créant de nouvelles images appelées cartes de convolutions. Figure représentant les deux parties d'un CNN .

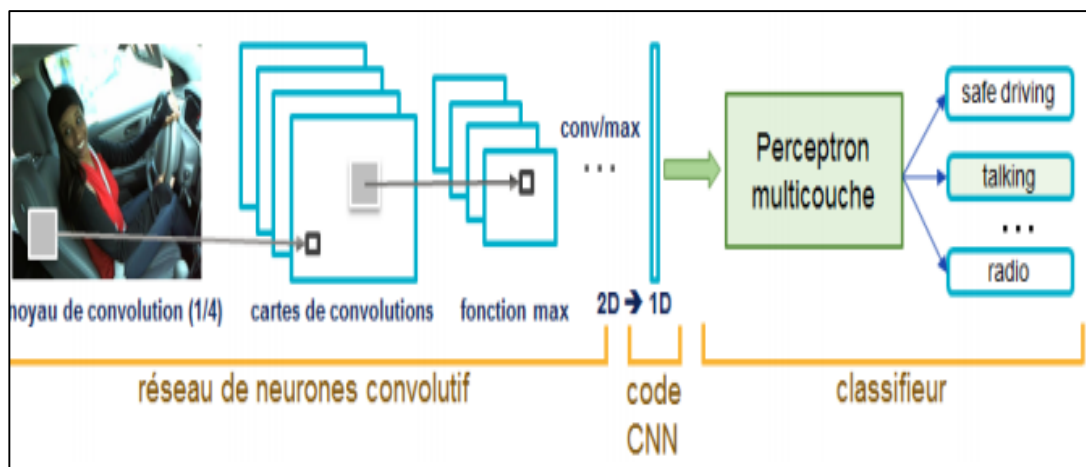


FIGURE 9 : ARCHITECTURE DES RESEAUX DE NEURONES CONVULUTIFS (CNN)

4.1. Couche entièrement connecté (Fully Connected)

Une couche entièrement connectée implique des poids, des biais et des neurones. Il connecte les neurones d'une couche aux neurones d'une autre couche. Il est utilisé pour classer les images entre différentes catégories par formation.

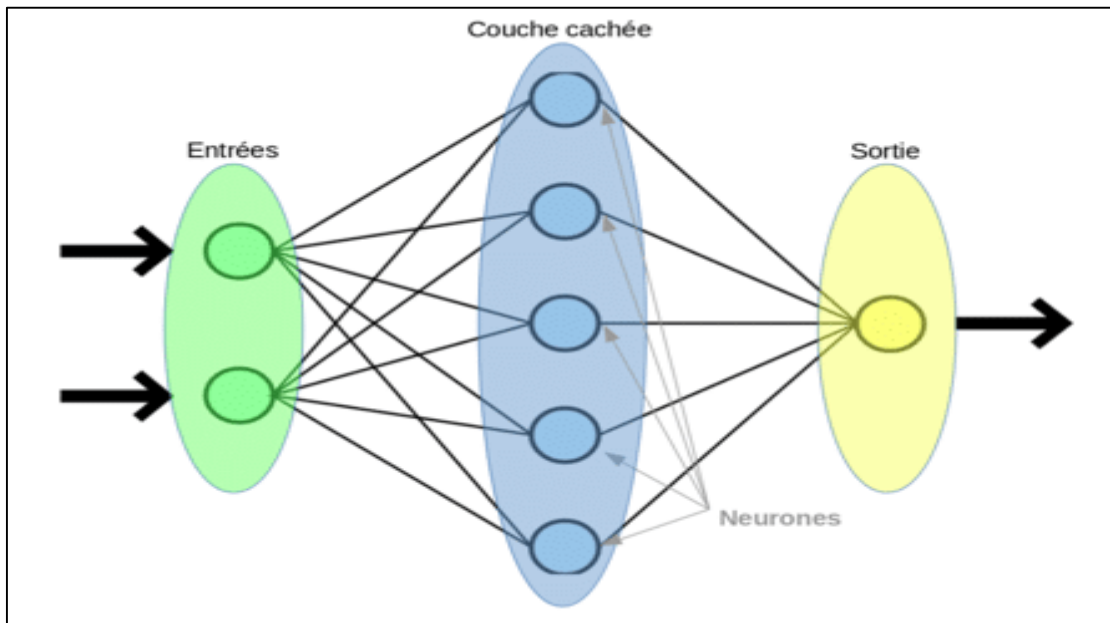


FIGURE 10 : PRINCIPE DE LA COUCHE ENTIEREMENT CONNECTEE

4.2. Couches convolutives

La convolution, d'un point de vue simpliste, est le fait d'appliquer un filtre mathématique à une image. D'un point de vue plus technique, il s'agit de faire glisser une matrice par-dessus une image, et pour chaque pixel, utiliser la somme de la multiplication de ce pixel par la valeur de la matrice. Cette technique nous permet de trouver des parties de l'image qui pourraient nous être intéressantes.

4.3. Couches de pooling

Ce type de couche est souvent placé entre deux couches de convolution : elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling. L'opération de pooling consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. Pour cela, on découpe l'image en cellules régulières, puis on garde au sein de chaque cellule la valeur maximale.

La méthode utilisée consiste à imaginer une fenêtre de 2 ou 3 pixels qui glisse au-dessus d'une image, comme pour la convolution. Mais, cette fois-ci, nous faisons des pas de 2 pour une fenêtre de taille 2, et des pas de 3 pour 3 pixels. La taille de la fenêtre est appelée « kernel size » et les pas s'appellent « strides ». Pour chaque étape, nous prenons la valeur la plus haute parmi celles présentes dans la fenêtre et cette valeur constitue un nouveau pixel dans une nouvelle image. Ceci s'appelle Max Pooling.

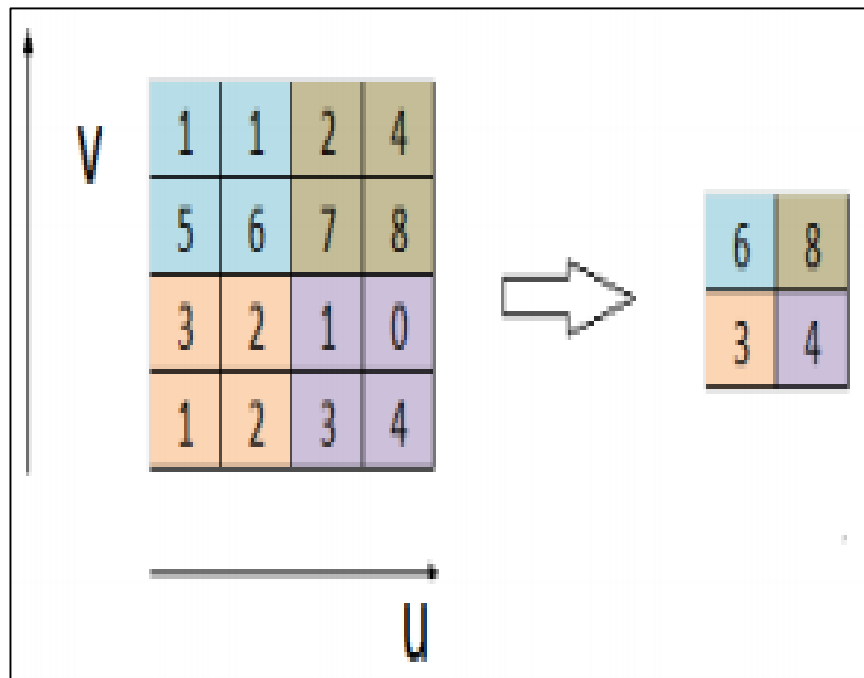


FIGURE 11 : MAX POOLING

4.4. Couches totalement connectées

Enfin, après plusieurs couches de torsion et d'assemblage, Pensée de haut niveau dans un réseau de neurones. Cela se fait par des coutures totalement connectées. Dans les réseaux de neurones convolutifs, chaque couche agit comme un filtre de détection pour la présence de caractéristiques spécifiques ou de motifs présents dans les données d'origine. Les premières couches d'un convolutif détectent des caractéristiques qui peuvent être reconnues et interprétées relativement facilement. Les couches ultérieures détectent de plus en plus des caractéristiques plus abstraites. La dernière couche du réseau convolutif est capable de faire une classification ultra-spécifique en combinant toutes les caractéristiques spécifiques détectées par les couches précédentes dans les données d'entrée.

5. Principe général de CNN

Dans les réseaux de neurones convolutifs (**Figure12**), on retrouve la même base que pour les réseaux de neurones artificiels vue précédemment. La différence notable est due à la présence de couches convolutives et de max-pooling. Celles-ci ont permis grâce à la puissance des ordinateurs actuels de faire apprendre des réseaux très grands et très profonds. Effectivement, sans ce mécanisme convolutif il y aurait eu beaucoup trop de paramètres dans des architectures de telle ampleur

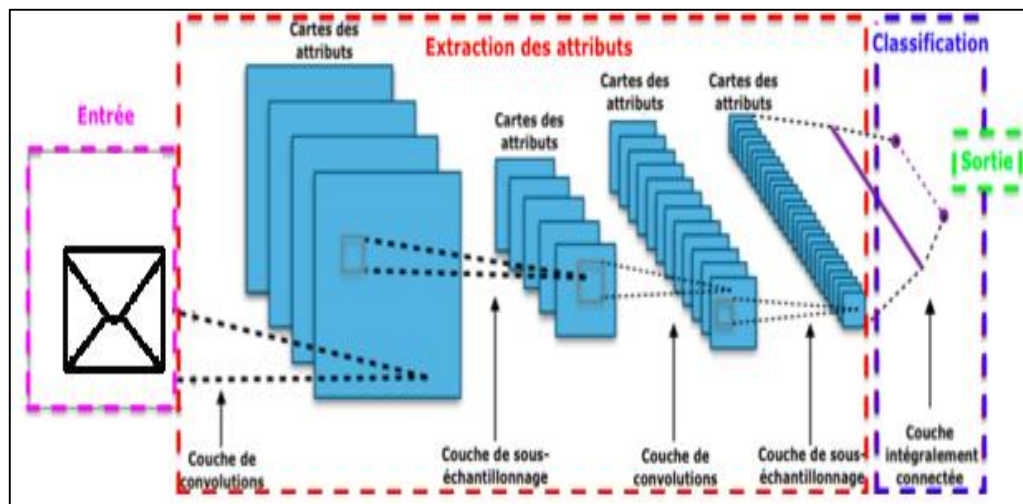


FIGURE 12 : FONCTIONNEMENT D'UN CNN

Le principe de cette architecture est de séparer le traitement des données de type différent dans des branches différentes du réseau et de fusionner les branches avant d'effectuer la classification. Dans notre figure, la branche représentée dans le cadre bleu traite les images avec les canaux bleu, rouge et vert, et la branche dans le cadre rouge traite l'image de profondeur.

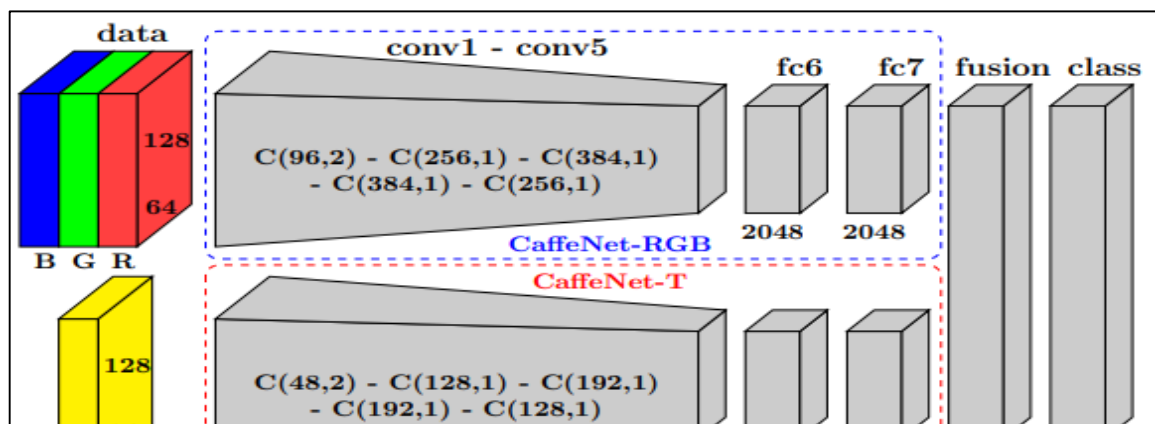


FIGURE 13 : Schéma De L'architecture Qui Permettant De Faire La Fusion De Données Multi-Sources Dans Un Cnn

6. Auto-encodeur (AE)

6.1. Description des auto-encodeurs

Les réseaux de neurones ne se limitent pas forcément à des problématiques d'apprentissage supervisé, même si cela reste leur utilisation principale. En effet, il existe aussi des réseaux de neurones qui permettent de faire de l'apprentissage non supervisé. Les plus utilisés dans ce domaine sont probablement les Auto-Encodeurs. Ces réseaux n'ont pas pour objectif de prédire ce qui se trouve dans une image mais seulement de chercher à trouver une représentation pertinente et compressée de l'image en tirant justement parti du pouvoir de représentation des réseaux de neurones [7].

Les Auto-Encodeurs sont des algorithmes d'apprentissage non supervisé à base des réseaux de neurones artificiels, qui permettent de construire une nouvelle représentation d'une image. Généralement, celle-ci est plus compacte, et présente moins de descripteurs, ce qui permet de réduire la dimensionnalité de l'image. L'architecture d'un auto-encodeur est constituée de deux parties : l'encodeur et le décodeur, ceci est indiqué dans la figure qui suit :

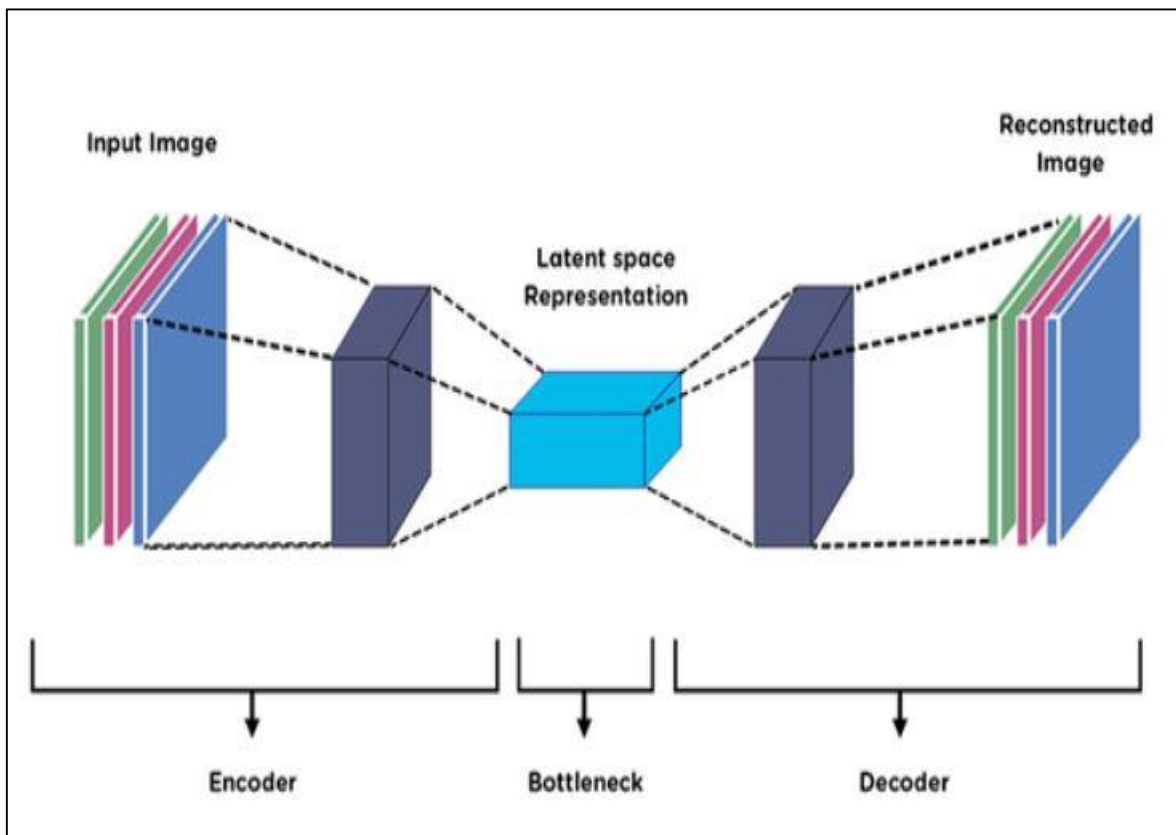


FIGURE 14 : Auto_Encodeur

6.2. Architecture des auto-encodeurs

Cette architecture peut être décomposée en deux parties : L'encodeur : est constitué par un ensemble de couches de neurones, qui traitent les données afin de construire de nouvelles représentations dites encodées. À leur tour, les couches de neurones du décodeur, reçoivent ces représentations et les traitent afin d'essayer de reconstruire les données de départ. Les différences entre les données reconstruites et les données initiales permettent de mesurer l'erreur commise par l'auto-encodeur. L'entraînement consiste à modifier les paramètres de l'auto-encodeur afin de réduire l'erreur de reconstruction mesurée sur les différents exemples du jeu de données. Le décodeur : C'est la dernière couche qui contient uniquement la reconstruction des données initiales, mais plutôt à la nouvelle représentation créée par l'encodeur. L'architecture la plus simple d'un auto-encodeur est semblable à un perceptron multicouche. Cependant, en fonction des données traitées, nous pouvons utiliser différentes topologies de réseaux de neurones. Par exemple, des couches convolutives afin d'analyser des images ou des couches de neurones récurrentes pour traiter des séries temporelles ou des séquences [7].

6.3. Apprentissage des auto-encodeurs

L'idée est simple mais ingénieuse : on entraînera un réseau de neurones à une couche cachée h à prédire en sa sortie z en son entrée x , Une tâche que nous nommons reconstruction. Le critère sera donc la minimisation d'une erreur de reconstruction $L(x, z)$. En principe, la couche cachée devra donc contenir de l'information pertinente à la reconstruction. Par exemple, si la couche cachée contient moins d'unités qu'il y a d'entrées, alors logiquement pour bien reconstruire elle doit apprendre à résumer l'entrée, puisque l'information pour la reconstruction est entièrement contenue dans la couche cachée. Ainsi des caractéristiques pertinentes à la reconstruction devront être extraites. Il faut noter que les paramètres pour passer de h à z , $W0$, ont une forme matricielle qui est la transposée de la forme des paramètres W . Si les poids sont liés (tied) alors il s'agit effectivement de la transposée et durant l'entraînement le contenu de la matrice sera contraint de rester le même pour les deux directions. Autrement, s'ils ne sont pas liés, ils sont libres de prendre des valeurs différentes. Il est aussi possible d'effectuer la reconstruction en faisant passer l'information dans plusieurs couches cachées h_1, \dots, h_n . Nous obtenons ainsi un auto-encodeur profond. Comme ce type de réseau contient plusieurs couches, il sera en général préférable de pré-entraîner les couches comme expliqué dans la section sur les architectures profondes, et de les faire

correspondre par paires : la première et la dernière, la seconde et l'avant dernière, par les correspondances entre les noms des paramètres de chacune des couches. Cependant, dans l'auto-encodeur simple, si le nombre d'unités dans la couche cachée est plus grand ou égal au nombre d'entrées, le modèle peut alors apprendre de façon triviale la fonction identité. C'est facile en copiant une à une les valeurs de l'entrée dans la couche cachée.

7. Conclusion

Dans ce chapitre, nous avons présenté les réseaux de neurones convolutifs. Ces réseaux peuvent extraire les caractéristiques de l'image présentée en entrée et classer ces caractéristiques. En fait, si l'on veut implémenter un modèle CNN pour suppression de tatouage numérique, il faut considérer le nombre de couches, le nombre de neurones dans chaque couche, et même les différentes connexions entre les couches. Ce sont tous des éléments vitaux. Réussir une série de tests / calculs d'erreurs (coût élevé en temps). Dans le chapitre suivant nous présenterons notre modèle, enfin on va expliquer les résultats obtenus lors des phases d'apprentissage et de test et les discuterons et les critiquerons.

Chapitre 3

Expérimentations

1. Introduction

Le but de ce chapitre est de présenter notre système réalisé, alors nous commençons par les différents outils du Deep Learning utilisés. Et par la suite, nous présenterons des différentes interfaces qui donnent la fonction d'application.

2. Les outils de développement

Avant de parler de l'implémentation de notre application, nous allons tout d'abord spécifier les outils utilisés que nous pouvons considérer un bon choix vu les avantages qu'ils offrent.

2.1. PyCharm

PyCharm est un environnement de développement intégré utilisé pour programmer en Python. Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django.



FIGURE 15 : LOGO PYCHARM

2.2. PyTorch

Il s'agit d'une bibliothèque open source conçue avec Python en tête et conçue pour les projets d'apprentissage automatique. Il est spécialisé dans la différenciation automatique, les calculs de tension et l'accélération GPU. Cela le rend particulièrement adapté aux applications d'apprentissage automatique de pointe telles que l'apprentissage en profondeur.

PyTorch est particulièrement populaire parmi les chercheurs en raison de la personnalisation de Python. La création de couches de données personnalisées et d'architectures réseau est particulièrement simple grâce à Python.

PyTorch est basé sur Torch, un cadre précoce pour l'apprentissage approfondi. PyTorch prend simplement le potentiel d'apprentissage approfondi de Torch et le place dans l'environnement Python.

Python est devenu l'un des langages de programmation les plus populaires pour les applications liées au Web, à côté d'autres langages de programmation modernes tels que R. Naturellement, les data scientists, les programmeurs et les développeurs voudraient intégrer les réseaux neuronaux et l'apprentissage approfondi dans leurs projets Python.

Pour ces raisons, nous allons utiliser cet outil dans notre présent travail.



FIGURE 16 : LOGO PYTORCH

2.3. Numpy

Numpy est la bibliothèque de base pour l'informatique scientifique en Python. Il contient des tableaux multidimensionnels et des structures de données matricielles. Elle peut être utilisée pour effectuer un certain nombre d'opérations mathématiques sur des tableaux tels que des routines trigonométriques, statistiques et algébriques. Par conséquent, la bibliothèque contient un grand nombre de fonctions mathématiques, algébriques et de transformation.

Pour réaliser notre application, nous avons besoin d'installer NumPy comme outil pour stocker et traiter de grandes matrices, ce qui est beaucoup plus efficace que la propre structure de liste imbriquée de Python. On dit que NumPy transforme Python en un Matlab system gratuit et plus puissant.

2.4. Pandas

Pandas est une librairie python qui permet de manipuler facilement des données à analyser :

- Manipuler des tableaux de données avec des étiquettes de variables (colonnes) et d'individus (lignes).
- Ces tableaux sont appelés Data Frames, similaires aux data frames sous R.
- On peut facilement lire et écrire ces data frames à partir ou vers un fichier tabulé.
- On peut facilement tracer des graphes à partir de ces Data Frames grâce à matplotlib.



FIGURE 17 : LOGO PANDAS

Pandas est une puissante bibliothèque polyvalente qui va permettre de réaliser facilement des analyses complexes de données.

Pandas est une bibliothèque open-source permettant la manipulation et l'analyse de données de manière simple et intuitive en Python. Elle a été développée par *Wes McKinney* en 2008 alors qu'il travaillait chez AQR Capital Management. À la fin de l'année 2009, elle a été

mise en open source et est aujourd'hui activement utilisée dans le domaine de la Big data et de la data science car celle-ci offre des performances et une productivité élevée à ces utilisateurs.

L'une des forces de Panda est qu'il se base sur la très populaire bibliothèque **NumPy**. Elle fournit diverses structures de données et opérations pour le traitement de données numériques et de séries chronologiques. En plus de cela, les données produites par Panda sont souvent utilisées comme données en entrée pour les fonctions de **plotting** de **Matplotlib**, l'analyse statistique en **SciPy**, les algorithmes de **machine learning** en **Scikit-learn**. Les data scientists l'utilisent pour le chargement, le traitement et l'analyse des données tabulaires (données stockées sous format .csv, .tsv ou .xlsx) à l'aide de requêtes de type SQL.

3. Base d'apprentissage Utilisé

3.1. Collecte des données

Nous avons utilisé une base de données gratuit pour réaliser notre projet, la figure ci-dessous représente le site de ma base d'apprentissage :

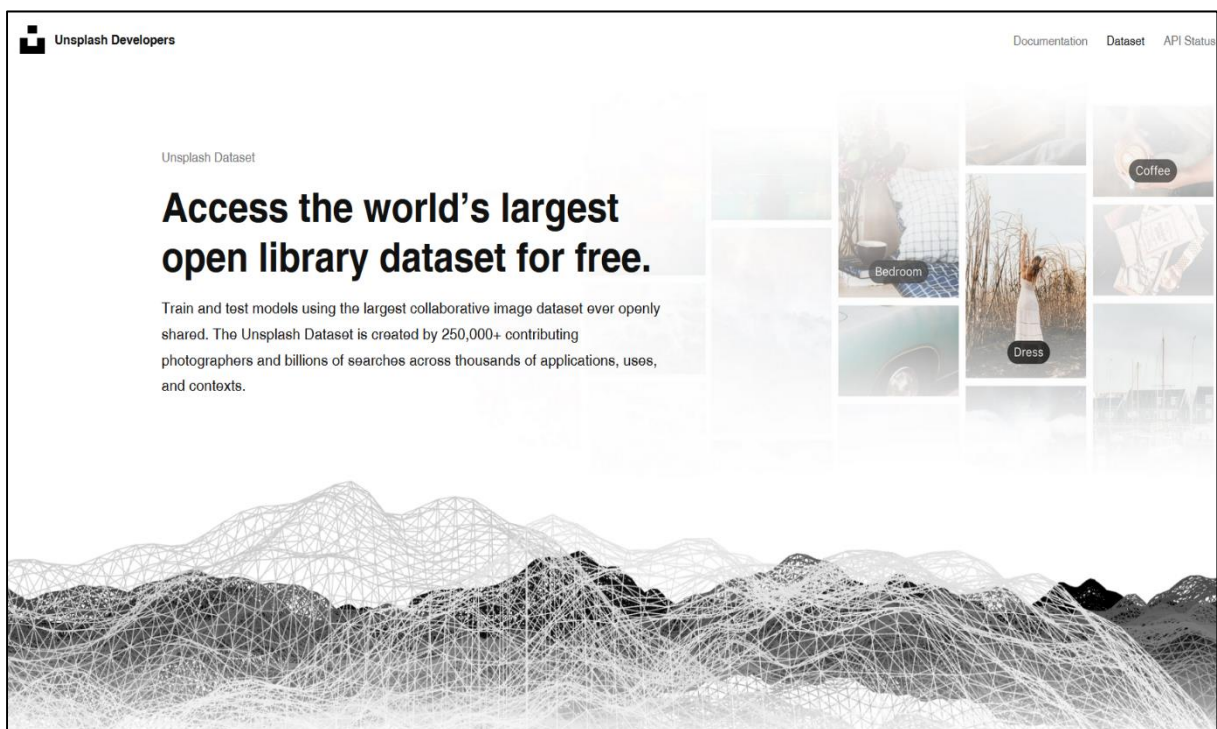


FIGURE 18 : SITE D'UNSPLASH POUR LA BASE DE DONNEES

La base de données utilisée pour l'entraînement du programme est constituée de centaines d'images.

Notre base d'apprentissage a été créée à l'aide de la base des données Lite d'Unsplash ; qui ne contient aucune image tatouée [8].

Lorsque nous travaillons sur ce projet, la première chose que nous avons fait était de collecter un tas d'images qui ne contenaient aucun filigrane. Delà « Unsplash » était une base de données d'images pleine d'images libres de droits ; donc pas de filigranes. Pour cette raison, on a décidé d'utiliser « Unsplash » pour collecter les images.

Méthode de collecte initiale :

La méthode initiale de collecte d'images pour notre base de données consistait à utiliser l'API d'Unsplash. Cette API permet à l'utilisateur d'envoyer un terme de recherche (par exemple "Tatouage") et de recevoir jusqu'à 30 résultats. Dans chaque résultat renvoyé, il y avait des informations sur l'image, y compris l'URL de cette image. À l'aide de cette API, nous avons créé une fonction appelée *collect_images()* dans *data_collection.py* qui prend un terme de recherche et collecte les URL des 30 résultats renvoyés. Ensuite, pour chaque URL, s'enregistre l'image sur notre ordinateur.

```
def collect_images(query: str) -> None:
    """THIS FUNCTION IS NO LONGER NEEDED.

    Takes a search term and downloads the first 30 images for from the
    unsplash API search results. The images are saved in the directory assigned
    to the 'ORIGINAL_IMAGE_DIRECTORY' variable found in the 'settings.py' file.

    Args:
        query (str): A keyword used to search images on unsplash, e.g. san diego.
    """
    # Cleans up query string to get it ready for url link.
    query = query.strip().replace(" ", "-")

    # The url needed to send searches queries using the unsplash API.
    api_url = f'https://api.unsplash.com/search/photos/?per_page=30&query={query}&client_id={secret.CLIENT_ID}'

    # Getting the json data for the returned search results.
    r = requests.get(api_url)
    json_data = r.json()

    # Finding the the image urls in the json and giving them a name to save.
    for index, image in enumerate(json_data['results']):
        image_url = image['urls']['raw']
        image_name = f"{query}_original_{index}.png"

        # Saving the image.
        try:
            with open(settings.ORIGINAL_IMAGE_DIRECTORY + "/" + image_name, 'wb') as f_obj:
                r_image = requests.get(image_url)
                f_obj.write(r_image.content)
        except:
            print("Could not save image.")
```

FIGURE 19 : LA FONCTION COLLECT_IMAGES()

Cette méthode fonctionnait mais était extrêmement lente puisque nous avons téléchargé les images brutes. Cela signifiait également qu'en ce qui concerne l'apprentissage de notre modèle, cela prendrait un temps extrêmement long, en raison de la taille des images. Étant donné que nous pouvons recevoir que 30 résultats par terme de recherche, cela signifiait que nous devons proposer une longue liste de termes de recherche afin d'obtenir un ensemble de données important. Après avoir découvert combien de temps il fallait pour télécharger des images et réaliser que donc nécessité d'une quantité ridicule de termes de recherche, nous avons donc décidé d'essayer de trouver un meilleur moyen d'obtenir suffisamment d'images pour les données.

Méthode de collecte finale :

Après avoir essayé de trouver un meilleur moyen de collecter des images pour notre base de données, on a découvert qu'Unsplash avait ses propres bases de données. Une base de données était la base de données Unsplash Lite et la base de données Unsplash Full. L'ensemble de données Lite contient 25 000 images qui sont libres d'utilisation, c'est donc l'ensemble de données que on a décidé d'utiliser.

Cette base de données est fournie avec plusieurs fichiers. *tsv* ; Variables séparées par des tabulations. L'un de ces fichiers *tsv* est **photos.tsv000** qui contient des informations sur les images de la base de données. Dans ce fichier, chaque image est sur sa propre ligne et chaque ligne contient diverses informations sur l'image. L'une de ces colonnes est un lien dynamique vers l'image. Pour cette raison, cela signifiait que nous avons un tas de liens directement vers des images. Cela signifiait que nous pouvons non seulement télécharger des images plus rapidement, mais utiliser le lien dynamique pour réduire l'image de sa taille brute à une image de 256x256 pixels. La figure suivante montre l'entête et quelques ligne de fichier « photos.tsv000 » :

photo_id	photo_url	photo_image_url	photo_submitted_at	photo_featured	photo_width	photo_height	photo_aspect_ratio	photo_description	photographer_username
XMyPniM9LF0	https://unsplash.com/photos/XMyPniM9LF0	https://images.unsplash.com/uploads/14119492946973137ce46/f1f2ebf3	2014-09-29 00:08:38.594364	t	4272	28			
rDLBArZU1lc	https://unsplash.com/photos/rDLBArZU1lc	https://images.unsplash.com/photo-1416339411116-62e1226aacd8	2014-11-18 19:36:57.08945	t	3000	4000			
cNDGZ2sQ3Bo	https://unsplash.com/photos/cNDGZ2sQ3Bo	https://images.unsplash.com/photo-1420142515034-86cc8c508475	2015-01-01 20:02:02.097036	t	2564	1710			
iuZ_Dleog9k	https://unsplash.com/photos/iuZ_Dleog9k	https://images.unsplash.com/photo-1414872809883-7620d2ae7566	2014-11-01 20:15:13.410073	t	2912	4368			
BeD3vjQ8SI0	https://unsplash.com/photos/BeD3vjQ8SI0	https://images.unsplash.com/photo-1417007594043-344e0104c3ce	2014-11-26 13:13:50.134383	t	4896	3264			
d00KS_QGnzY	https://unsplash.com/photos/d00KS_QGnzY	https://images.unsplash.com/uploads/1411476843343e89a8f76/bda95c47	2014-09-23 12:56:00.965116	t	2816	21			
ocwmW1NAWGs	https://unsplash.com/photos/ocwmW1NAWGs	https://images.unsplash.com/reserve/m6r74MYFQ7CT8j9m2AEC_JakeGivens%20-%20Sunset%20in%20the%20Park.JPG	2014-0						
cGe1PV_yZso	https://unsplash.com/photos/cGe1PV_yZso	https://images.unsplash.com/photo-1428550670225-15f007f6f1ba	2015-04-09 03:37:53.01747	t	2000	1333			
Wkt6okFYtg4	https://unsplash.com/photos/Wkt6okFYtg4	https://images.unsplash.com/photo-1429270958905-78f90e2dca75	2015-04-17 11:42:41.31631	f	5184	3456			

FIGURE 20 : LE FICHIER "PHOTOS.TSV000 "

Après la finalisation de l'étape de collecte de données, on a obtenu 10000 images dans notre base d'apprentissage. Cette figure montre quelqu'une avec la taille 256x256 pixels.

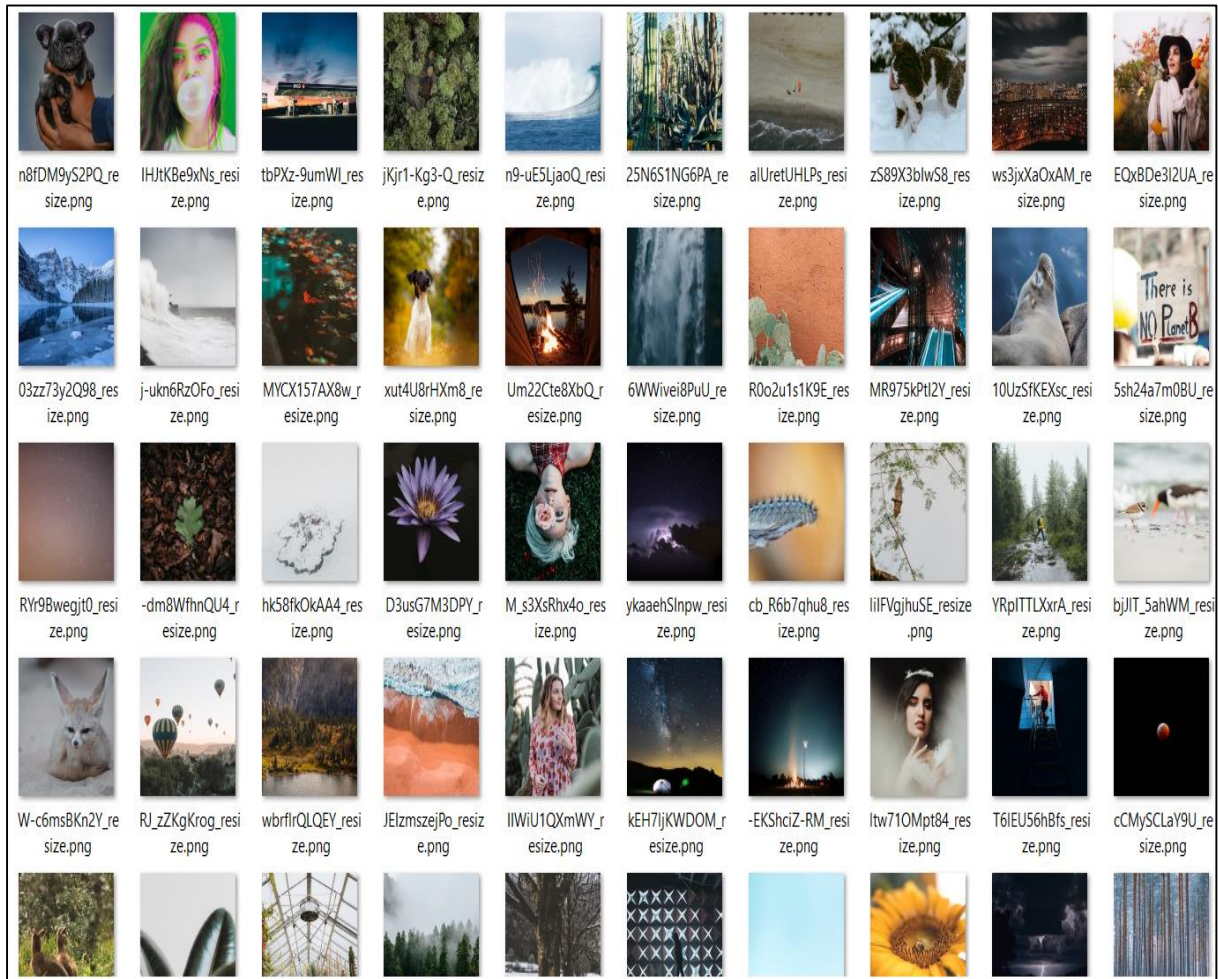


FIGURE 21 : EXEMPLE DES PHOTOS DE BASE D'APPRENTISSAGE

3.2. . Préparation des données

Après avoir collecté toutes les images dont nous avons besoin (environ 2000 images), nous avons besoin de créer de nouvelles images qui seraient utilisées pour l'entraînement. Pour ce faire, on a créé une copie de chaque image de notre version initiale, y ajoute un filigrane et crée une nouvelle image avec les versions originales et tatoué côte à côte.

Nous avons créé ces images de training pour chaque image originale que nous avons téléchargée. Chaque image a également son propre tatouage avec différents textes et emplacements aléatoires. Ceci est à l'aide de la classe `create_watermarked_image()` dans le fichier `data_collection.py`.

```
def create_watermarked_image(filename: str, text: str, font_size=100)-> None:
    """THIS FUNCTION IS NO LONGER NEEDED.

    Takes a non-watermarked image and creates a copy off it with a
    watermark. The new image is saved in the directory assigned to the
    'WATERMARKED_IMAGE_DIRECTORY' variable found in the 'settings.py' file.

    Args:
        filename (str): The filename of the non-watermarked image.
        text (str): The text that will be in the watermark, e.g. shutterstock.
        font_size (int, optional): The font size of the text in the watermark. Defaults to 100.
    """
    # Creating the output filename.
    watermarked_filename = filename.replace("original", "watermarked")

    # Getting the original image.
    try:
        image = Image.open(settings.ORIGINAL_IMAGE_DIRECTORY + "/" + filename).convert('RGBA')
        text_obj = Image.new('RGBA', image.size, (255, 255, 255, 0))
        font = ImageFont.truetype('arial.ttf', font_size)

        # Create watermark layer.
        draw_obj = ImageDraw.Draw(text_obj)

        # Adding watermarks to the watermark layer.
        width, height = image.size

        y = 200
        for i in range(10):
            x = random.randint(0, width)
            y += random.randrange(0, int(height / 8), 19) + random.randint(0, 100)
            draw_obj.text((x, y), text, fill=(255, 255, 255, 175), font=font)

        # Combining the original image layer with the watermark layer.
        watermarked_image = Image.alpha_composite(image, text_obj)
        watermarked_image.save(settings.WATERMARKED_IMAGE_DIRECTORY + "/" + watermarked_filename)
    except:
        pass
```

FIGURE 22 : LA FONCTION DE CREATION DE TATOUAGE

Après cela, on a retiré au hasard 20 % de ces images d'entraînement et les enregistrées pour les utiliser comme images de validation.

Ensuite, nous avons créé une classe **Dataset** dans le fichier *dataset.py*. Cette classe est la façon dont on charge les données dans notre modèle. La classe prend également chaque image, y ajoute quelques augmentations et la divise au milieu en deux images : l'image d'entrée et l'image cible. L'image d'entrée contient le tatouage et l'image cible ne contient pas de tatouage.

```
class WatermarkDataset(Dataset):
    def __init__(self, root_directory) -> None:
        """Inits WatermarkDataset with a directory."""
        self.root_directory = root_directory
        self.list_files = os.listdir(self.root_directory)

    def __len__(self) -> int:
        """Returns the number of files in the dataset."""
        return len(self.list_files)

    def __getitem__(self, index: int) -> tuple:
        """Takes and image and splits it into two. One is the input image,
        the other is the target image.

        Args:
            index (int): The index of the current image from the image list.

        Returns:
            tuple: Two arrays in a tuple. The first is the input image and the
                second is target image.
        """
        image_file = self.list_files[index]
        image_path = os.path.join(self.root_directory, image_file)
        image_object = Image.open(image_path).convert('RGB')
        # image_width = image_object.width
        image = np.array(image_object)

        # The center point width-wise to split the image in two.
        # center_width = int(image_width / 2)
        center_width = int(image.shape[1] / 2)

        # Splitting image in two. Input image and Target image.
        input_image = image[:, center_width:, :]
        target_image = image[:, :center_width, :]

        # Adding Augmentations to the images.
        augmentations = settings.both_transform(image=input_image, image0=target_image)
        input_image = augmentations["image"]
        target_image = augmentations["image0"]

        input_image = settings.transform_only_input(image=input_image)["image"]
        target_image = settings.transform_only_mask(image=target_image)["image"]

        return input_image, target_image
```

FIGURE 23 : LA CLASSE DATASET()

4. Modélisation les données

4.1. Importation des bibliothèques

Nous importons tous les modules nécessaires pour entraîner le modèle, dans la figure ci-dessous représente l'importation de différentes bibliothèques :

```
from datetime import datetime

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader

from tqdm import tqdm

import secret
import settings
import utilities
from dataset import WatermarkDataset
from generator_model import Generator
from discriminator_model import Discriminator
```

FIGURE 24 : CODE D'IMPORTATION DES BIBLIOTHEQUES

4.2. Création de modèle

Nous allons créer le réseau de Convolution. Ce réseau est basé sur plusieurs couche qui répètent le motif de Couche de Convolution, Normalisation, MaxPooling et UpSampling.

Le choix de la structure du réseau de Convolution est primordial pour l'optimisation des performances. Il est important de procéder à des tests pour valider le modèle de réseau qui permet d'obtenir les meilleures performances.

Notre modèle est basé sur l'auto-encodeur, il contient deux parties : un générateur et un discriminateur. Le générateur était lourd et basé sur un modèle U-NET où nous sous-échantillonons puis sur-échantillonner les données. Quant au discriminateur, il ne s'agit que de quelques couches CNN.

Au fur et à mesure que le modèle est formé, des points de contrôle sont enregistrés pour le discriminateur et le générateur du modèle toutes les 5 époques. De plus, un échantillon d'image est enregistré pour chaque époque en montrant l'image cible, l'image d'entrée et l'image générée.

Voici le code de la partie générateur :

```
class Generator(nn.Module):
    def __init__(self, in_channels=3, features=64):
        super().__init__()
        self.initial_down = nn.Sequential(
            nn.Conv2d(in_channels, features, 4, 2, 1, padding_mode="reflect"),
            nn.LeakyReLU(0.2),
        ) # 128

        self.down1 = Block(features, features*2, down=True, act="leaky", use_dropout=False) # 64
        self.down2 = Block(features*2, features*4, down=True, act="leaky", use_dropout=False) # 32
        self.down3 = Block(features*4, features*8, down=True, act="leaky", use_dropout=False) # 16
        self.down4 = Block(features*8, features*8, down=True, act="leaky", use_dropout=False) # 8
        self.down5 = Block(features*8, features*8, down=True, act="leaky", use_dropout=False) # 4
        self.down6 = Block(features*8, features*8, down=True, act="leaky", use_dropout=False) # 2
        self.bottleneck = nn.Sequential(
            nn.Conv2d(features*8, features*8, 4, 2, 1, padding_mode="reflect"), nn.ReLU(), # 1x1
        )
        self.up1 = Block(features*8, features*8, down=False, act="relu", use_dropout=True)
        self.up2 = Block(features*8*2, features*8, down=False, act="relu", use_dropout=True)
        self.up3 = Block(features*8*2, features*8, down=False, act="relu", use_dropout=True)
        self.up4 = Block(features*8*2, features*8, down=False, act="relu", use_dropout=False)
        self.up5 = Block(features*8*2, features*4, down=False, act="relu", use_dropout=False)
        self.up6 = Block(features*4*2, features*2, down=False, act="relu", use_dropout=False)
        self.up7 = Block(features*2*2, features, down=False, act="relu", use_dropout=False)
        self.final_up = nn.Sequential(
            nn.ConvTranspose2d(features*2, in_channels, kernel_size=4, stride=2, padding=1),
            nn.Tanh(), # We want all pixel values to be between -1 and 1
        )
```

FIGURE 25 : CLASSE GENERATEUR DE MODELE

Et ceci le code de la partie discriminateur :

```
class Discriminator(nn.Module):
    def __init__(self, in_channels=3, features=[64, 128, 256]):
        super().__init__()
        self.initial = nn.Sequential(
            nn.Conv2d(
                in_channels * 2,
                features[0],
                kernel_size=4,
                stride=2,
                padding=1,
                padding_mode="reflect",
            ),
            nn.LeakyReLU(0.2),
        )

        layers = []
        in_channels = features[0]
        for feature in features[1:]:
            layers.append(
                CNMBlock(in_channels, feature, stride=1 if feature == features[-1] else 2),
            )
            in_channels = feature

        layers.append(
            nn.Conv2d(
                in_channels, 1, kernel_size=4, stride=1, padding=1, padding_mode="reflect"
            ),
        )

        self.model = nn.Sequential(*layers)
```

FIGURE 26 : Classe discriminateur de modèle

4.3. Entraînement de modèle

Il est enfin temps d'entraîner le modèle pour 20 époques comme suit :

```
# Training Loop.
for epoch in range(settings.NUM_EPOCHS):
    train_function(discriminator,
                  generator,
                  train_loader,
                  discriminator_optimizer,
                  generator_optimizer,
                  L1_LOSS,
                  BCE,
                  generator_scaler,
                  discriminator_scaler)

    if settings.SAVE_MODEL and epoch % 5 == 0:
        utilities.save_checkpoint(generator, generator_optimizer, filename=settings.CHECKPOINT_GEN)
        utilities.save_checkpoint(discriminator, discriminator_optimizer, filename=settings.CHECKPOINT_DISC)
```

FIGURE 27 : CODE DE TRAINING

Nous allons obtenir cette démarche d'entraînement, ce qui nous aidera à analyser visuellement les performances de notre modèle.

```
warnings.warn('User provided device_type of \'cuda\', but CUDA is not available. Disabling')
100%|██████████| 375/375 [19:08<00:00, 3.06s/it, disc_generated_image=0.346, disc_input_image=0.623]
=> Saving Checkpoint
=> Saving Checkpoint
100%|██████████| 375/375 [18:43<00:00, 3.00s/it, disc_generated_image=0.261, disc_input_image=0.772]
100%|██████████| 375/375 [18:33<00:00, 2.97s/it, disc_generated_image=0.259, disc_input_image=0.631]
100%|██████████| 375/375 [18:23<00:00, 2.94s/it, disc_generated_image=0.184, disc_input_image=0.898]
100%|██████████| 375/375 [18:17<00:00, 2.93s/it, disc_generated_image=0.365, disc_input_image=0.882]
100%|██████████| 375/375 [18:15<00:00, 2.92s/it, disc_generated_image=0.0863, disc_input_image=0.22]
=> Saving Checkpoint
=> Saving Checkpoint
100%|██████████| 375/375 [18:54<00:00, 3.02s/it, disc_generated_image=0.295, disc_input_image=0.899]
100%|██████████| 375/375 [18:09<00:00, 2.90s/it, disc_generated_image=0.167, disc_input_image=0.454]
100%|██████████| 375/375 [18:38<00:00, 2.98s/it, disc_generated_image=0.185, disc_input_image=0.789]
100%|██████████| 375/375 [18:12<00:00, 2.91s/it, disc_generated_image=0.113, disc_input_image=0.901]
100%|██████████| 375/375 [18:14<00:00, 2.92s/it, disc_generated_image=0.117, disc_input_image=0.583]
=> Saving Checkpoint
=> Saving Checkpoint
100%|██████████| 375/375 [18:48<00:00, 3.01s/it, disc_generated_image=0.159, disc_input_image=0.735]
100%|██████████| 375/375 [18:50<00:00, 3.01s/it, disc_generated_image=0.26, disc_input_image=0.912]
100%|██████████| 375/375 [18:11<00:00, 2.91s/it, disc_generated_image=0.288, disc_input_image=0.729]
100%|██████████| 375/375 [18:01<00:00, 2.88s/it, disc_generated_image=0.265, disc_input_image=0.544]
100%|██████████| 375/375 [18:40<00:00, 2.99s/it, disc_generated_image=0.297, disc_input_image=0.409]
=> Saving Checkpoint
=> Saving Checkpoint
100%|██████████| 375/375 [18:21<00:00, 2.94s/it, disc_generated_image=0.102, disc_input_image=0.518]
100%|██████████| 375/375 [18:13<00:00, 2.92s/it, disc_generated_image=0.527, disc_input_image=0.981]
100%|██████████| 375/375 [18:13<00:00, 2.92s/it, disc_generated_image=0.108, disc_input_image=0.707]
100%|██████████| 375/375 [18:18<00:00, 2.93s/it, disc_generated_image=0.134, disc_input_image=0.829]
```

FIGURE 28 : EXECUTION DE 20 EPOQUES

4.4. Résultats

Nous allons dans cette partie évaluer l'apprentissage du réseau. Ci-dessous, nous montrons un graphique représentant une comparaison visuelle entre une image d'entrée et une image de sortie capturée après un apprentissage de 20 itérations dont chacune a duré presque 20 minutes.



FIGURE 29 : RESULTATS OBTENUS

5. Conclusion

Ce chapitre a été consacré en premier lieu à la présentation des outils utilisés pour la configuration de notre travail, nous a permis d'acquérir une bonne base de connaissance sur python. Ensuite, nous avons décrit les différentes technologies utilisées. Enfin, nous avons détaillé les fonctions de ce système et analysé les résultats.

Conclusion Générale

Ce présent travail est élaboré dans le cadre d'élaboration de mémoire de fin d'études pour l'obtention du diplôme de Mastère Professionnelle en Sciences en Données.

Dans le cadre de ce projet pédagogique, nous nous sommes intéressés à la réalisation d'un modèle de suppression de tatouage numériques des images.

Pour conclure sous forme d'un résumé mettant en évidence les points importants de notre travail, nous pouvons dire que nous avons montré la grande efficacité des réseaux de neurone convolutifs aux algorithmes de tatouage numérique.

Au terme de ce rapport, nous pouvons conclure que ce projet de fin d'études nous a donné une occasion opportune nous permettant de confronter l'acquis théorique à l'environnement pratique.

En effet, notre nous a permis de prendre certaines responsabilités, par la suite de consolider de plus en plus nos connaissances théoriques et pratiques. C'est là que réside la valeur d'un tel projet de fin d'études qui combine les exigences de la vie professionnelle aux côtés bénéfiques de l'enseignement pratique que nous avons eu à ISSAT.

L'avancement de l'Intelligence Artificielle est en corrélation positive avec la voluminosité et la variété des données accumulées sur son utilisation.

Cependant, un aspect de la technologie qui restera le même est le principe de simplicité de conception : la technologie doit toujours rester accessible à son public cible, tandis que les tâches plus techniques sont cachées.

Références bibliographiques

- [1] T. Nadjia, «Sécurisation de la transmission de l'information par les techniques du tatouage robuste et applications,» 2019.
- [2] D. Ulyanov, «Deep Image Prior».
- [3] The GIMP Team, «<https://www.gimp.org/>,» [En ligne]. [Accès le 2023].
- [4] «<https://www.adobe.com/fr/products/photoshop.html>,» [En ligne]. [Accès le 2023].
- [5] WikiSat, «Réseaux de neurones».
- [6] F. HOCINE, «Application des réseaux de neurones artificiels au diagnostic des défauts des machines tournantes,» 2020.
- [7] «Les Autoencoders – modèles d'apprentissage non supervisé,» [En ligne]. Available: <https://datascientest.com/les-autoencoders-modeles-dapprentissage-non-supervise>. [Accès le 2023].
- [8] «unsplash,» [En ligne]. Available: <https://unsplash.com/data>. [Accès le 2023].
- [9] l. c. d. c. l. c. d. p. l. c. d. c. R. e. l. c. f.-c. les CNN sont spécialement conçus pour traiter des images en entrée: elle est composée de quatre types de couches pour un réseau de neurones.
- [10] <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn>.
- [11] <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn>.

Résumé

Dans ce projet nous avons utilisé la méthode d'apprentissage profond et les réseaux de neurones convolutifs. Nous avons expérimenté ces réseaux pour supprimer le tatouage à partir l'image originale.

Pour se faire, nous avons appliqué la combinaison auto-encodeur sur un ensemble d'images tatoués. La base des images utilisée est Unsplash dont nous avons testées 2000 images (1500 images pour l'entraînement, 500 images pour la validation), nous avons déformé ces images en ajoutant un text (filigrane).

Mots clés : Google Colab, python, CNN, Tatouage numérique, Intelligence artificielle.

Abstract

In this project we used the deep learning method and convolutional neural networks. We experimented with these networks to remove the watermark from the original image.

To do this, we applied the auto-encoder combination to a set of watermarked images. The base of the images used is Unsplash, of which we tested 2000 images (1500 images for training, 500 images for validation), we distorted these images by adding a text (watermark).

Keywords: Google Colab, python, CNN, Watermarking, Artificial Intelligence.