الجمـهورية الجزائرية الديمقراطية الشعبيـة
وزارة التعليم العالي والبحث العلمي
جـامعـة سعـيـدة د.مـولاي الطـاهـر
كـليـة التكنولوجيا
قسم: الإعلام الآلي

Mémoire de Master

Spécialité : Sécurité Informatique et Cryptographie

Thème

Détection de Spam Image par les Techniques de Deep Learning.

Présenté par :
HELAL Ahlem

Dirigé par :
Dr. LATRECHE Abdelkrim

Promotion 2021 - 2022

Remerciements

Notre remerciement s'adresse en premier lieu à Allah le tout puissant pour la volonté, la santé et la patience qu'il nous a donnée Durant toutes ces longues années.

Ainsi, nous remercie Allah tout puissant de nous avoir donnés le courage et la patience pour mener à bien ce travail, qu'il soit béni et glorifié.

Nous tenons également à exprimer nos vifs remerciements à Notre encadreur Dr : ***Abdelkrim LATRECHE****, qui nous avoir apporté une aide précieuse.*

Je remercie tout d'abord les membres du jury qui me font honneur en jugeant ce travail.

Enfin, nous tenons à exprimer notre reconnaissance à tous nos Amis et collègues pour le soutien moral et matériel

Dédicace

Je dédie ce travail :

À mes très chers parents pour leur soutient durant toute ma vie

d'études et sans Lesquels je n'aurais jamais devenu

Ce que je suis.

À tous mes sœurs« Koki, Yasma, Samia, Kheira et Touta ».qui ont été toujours là

pour moi et qui m'ont tant soutenu.

À « Bouchafa » qui me donne toujours le courage.

À toutes les professeurs et enseignants que j'ai eu durant tout mon cursus scolaire et qui m'ont permis de réussir Dans mes études.

À tous mes amis qui m'ont aidé tout le long de ce projet.

Et à tous ceux qui ont contribué de près ou de loin pour que ce Projet soit possible, je vous dis merci.

Table de matière

Table des figures

Liste des tableaux

Table des sigles et acronymes

OCR	**: Optical Character Recognition.**
CNN	**: Convolutional Neural Network.**
SVM	**: Support Vector Machine.**
KNN	**: K Nearest Neighbor.**
BPNN	**: BackPropagation Neural Network.**
IA	**: Intelligence Artificielle.**
ML	**: Machine Learning.**
DL	**: Deep Learning.**
ANN	**: Artificial Neural Network.**
DNN	**: Deep Neural Network.**
GD	**: Gradient Descente.**
Tanh	**: Hyperbolic Tangent.**
ReLU	**: Rectified Linear Unit.**
MLP	**: Multi-Layer Perceptron.**
LMS	**: learning management system.**
FC	**: Fully Connected.**
GPU	**: Graphics Processing Unit.**

Introduction Générale

Aujourd'hui, le courrier électronique (e-mail) est devenu l'un des canaux le plus populaires, le plus puissants et plus fréquemment utilisés pour la communication personnelle et professionnelle en ligne (Runbox, 2017). À titre indicatif, En 2015, le nombre d'utilisateurs de courrier électronique était de 2,6 milliards, tandis qu'en 2019, ce nombre passera à environ 2,9 milliards, avec plus d'un tiers de la population mondiale utilisant le courrier électronique pour échanger des messages Le nombre d'e-mails envoyés chaque jour dans le monde est de 293 milliards en 2019 (hors spams). Le succès de l'email est dû en partie à sa rapidité, sa permanence, son faible coût et la facilité de distribution des données.

Malgré ces avantages, le courrier électronique est confronté à un problème de sécurité majeur, à savoir la réception quotidienne par les utilisateurs d'un grand nombre de messages électroniques non sollicités, appelés "spams". Le spam est un courrier texte ou image indésirables et non sollicité reçu par les utilisateurs et souvent envoyé par un expéditeur obscur sans le consentement de l'utilisateur, qui peut souvent contenir des publicités, du contenu pour adultes, des logiciels malveillants, etc. L'utilisation répandue et massive du courrier électronique en fait une cible privilégiée pour les spammeurs. Le spam est devenu un problème majeur pour les réseaux Internet (Ketari, et al, 2012). Selon une étude récente de Symantec, indiquent que 90,4 % des e-mails incluent du contenu spam.

Aujourd'hui, la plupart des systèmes de courrier électronique sont dotés de mécanismes de filtrage de spams qui peuvent bloquer ou mettre en quarantaine les courriers indésirables, et la plupart d'entre eux sont essentiellement basés sur des technologies de filtrage du spam textuel. Dans ce contexte, de nombreux systèmes de classification ont été développés pour détecter et filtrer les courriers indésirables, en fonction d'un certain nombre de caractéristiques, telles que leur en-tête, leur objet et leur contenu. Par exemple, dans (Lai et Tsai, 2004), les auteurs exploitent quatre algorithmes d'apprentissage automatique utilisés pour détecter le spam en utilisant différentes parties du message électronique. Les algorithmes d'apprentissage automatique sont KNN, SVM, Naïve Bayes, etc. Ces classificateurs ont pu classer les spams textuels avec une précision d'environ 95 %. Par conséquent, au fil des années, la détection des spams basés sur le contenu est devenue très facile. Google, Microsoft, Yahoo ont utilisé des techniques qui fonctionnent très précisément pour classer les e-mails authentiques.

Pour contourner ces puissants filtres de détection basés sur le texte, les spammeurs ont réagi en introduisant de nouvelles techniques d'intégration de texte de spam dans des images jointes au courrier électronique, appelées "spam image". Le spam image est une sorte de spam dans lequel le texte du message est incorporé dans une image qui est ensuite jointes à l'e-mail. Les premiers spam images contenaient du texte facilement lisible, comme le montre la figure 1.3. Le texte spam intégré dans une image peut être une méthode efficace pour contourner les systèmes de filtrage textuelle (Gao et al., 2008). Ce type de spam s'est rapidement développé ces dernières années, le défi majeur des nouveaux systèmes de filtrage est donc de trouver des méthodes efficaces pour distinguer une image spam d'une image légitime (ham) contenue dans l'email. Pour atteindre cet objectif, de nombreux travaux ont été réalisés en proposant des techniques pour filtrer ce type d'images contenues dans les e-mails. En général, les techniques de détection d'images spam sont divisées en trois catégories : i) Techniques basées sur l'en-tête de l'e-mail spam qui se compose de nombreux champs qui fournissent une gamme d'informations utiles pour l'analyse et la détection ii) Techniques basées sur l'OCR (Optical Character Recognition)

qui utilisent la technique OCR pour extraire le texte intégré dans l'image (Dredze, et al, 2007). iii) Techniques non basées sur le contenu utilisant l'analyse du contenu de l'image et l'extraction de caractéristiques.

Les techniques basées sur l'OCR utilisent des techniques de reconnaissance optique de caractères pour extraire le texte intégré dans les images spam, puis le soumettre avec le corps du texte de l'e-mail à des techniques de détection basées sur le texte (Nisha et Gaikwad, 2015). Récemment, pour contourner ce type de filtre anti-spam, les spammeurs ont introduit des techniques d'obscurcissement des images spam afin d'empêcher les outils OCR de lire le texte intégré dans les images. Quelques exemples sont présentés dans la figure 1.1. Cela a soulevé la question de l'amélioration de la détection du spam image à l'aide d'autres techniques. En particulier, plusieurs chercheurs ont étudié la possibilité d'utiliser des fonctionnalités d'image génériques de bas niveau pour reconnaître les images spam bruités.

Les techniques non basées sur le contenu sont destinées à étudier et analyser les caractéristiques et le contenu de l'image, tels que la couleur, la texture, le bord, l'ombrage, la surface, etc. sont extraits de l'image et qui sont utilisés pour filtrer les images spam.

Des recherches antérieures sur la détection du spam image ont montré que certains types de spam image peuvent être détectés avec une grande précision. Par exemple, les travaux présentés dans (Annadatha & Stamp, 2018 ; Chavda, and al, 2018), une grande variété de propriétés d'image sont extraites et les images sont classées comme spam ou ham (c'est-à-dire, images non-spam) basée sur des techniques d'apprentissage automatique. Cependant, certains types complexes de spam image sont difficiles à détecter à l'aide de ces techniques.

De nombreux chercheurs proposent des méthodes basées sur Deep Learning (DL) pour diverses applications de cyber sécurité, telles que la détection de logiciels malveillants, la détection d'intrusion, etc. Le succès actuel des réseaux de neurones convolutés (CNN) dans la classification d'images s'est étendu au problème de la détection et filtrages des spam image. Le deep Learning et plus particulièrement les réseaux de neurones convolutionnels (CNN) ont apparu spécialement pour résoudre les problèmes rencontres en machine Learning et d'améliorer la précision de la détection. Les réseaux de neurones convolutifs (CNN) est l'une des structures réseau les plus représentatives de la technologie d'apprentissage en profondeur et a connu un grand succès dans le domaine du traitement et de la reconnaissance d'images. CNN a le potentiel de traiter les entrées de données brutes (par exemple, l'image d'entrée elle-même) en extrayant des caractéristiques importantes (de bas niveau) de manière automatisée (Krizhevsky et al., 2017).

Les techniques DL peuvent également être utilisés pour détecter les spam image qui peut exploiter les performances des méthodes existantes (Shanget al.,2016;Aiwan et al., 2018 ; Fatichah, et al., 2019 ; Kumar, et al., 2018). Cependant, nous avons constaté que la précision de détection du modèle de détection de spam d'image basé sur CNN (Shanget al., 2016) pouvait être considérablement dégradée par rapport aux e-mails basés sur le spam d'image nouveaux et invisibles.

L'objectif de ce mémoire consiste à proposer un système de détection de spam image basé sur le deep Learning (apprentissage en profondeur) et plus particulièrement les réseaux de neurones convolutifs. Ce système est capable de faire la distinction entre les images spam et les images légitimes (ham). Nous proposons deux modèles CNN différents et nous comparons les résultats obtenus au modèle VGG et à d'autres techniques d'apprentissage automatique tel que le SVM. Nous prenons en compte à la fois les spams images du monde réel et les ensembles de données complexes de type spam images. Des tests expérimentaux seront réalisés sur une base de données réelle.

Ce mémoire est constitué en quatre chapitres, et organise comme suit :

- Dans le premier chapitre, nous présentons les concepts de base sur les spams en générale et les spams images en particulier et les différentes techniques utilisées pour détecter ce type de spam.
- Ensuite, le second chapitre, nous le consacrons à la présentation de l'apprentissage profond, ou nous donnerons plus de détails sur les réseaux de neurones convolutifs.
- La conception de notre approche de détection de spam image basée sur les CNNs est présentée dans le troisième chapitre.
- Le dernier chapitre est consacré à la description des différents outils utilisés dans le développement de notre application, ainsi que les différents résultats obtenus.
- Et enfin, nous terminerons ce mémoire par une conclusion générale et quelques perspectives.

Chapitre 1 : Détection et filtrage de spam image

1.1 Introduction

Aujourd'hui, le courrier électronique (e-mail) est devenu l'un des canaux le plus populaires, le plus puissants et plus fréquemment utilisés pour la communication personnelle et professionnelle en ligne. Malgré ces avantages, le courrier électronique est confronté à un problème de sécurité majeur, à savoir la réception quotidienne par les utilisateurs d'un grand nombre de messages électroniques non sollicités, appelés "spams". L'utilisation répandue et massive du courrier électronique en fait une cible privilégiée pour les spammeurs. Le spam est devenu un problème de sécurité majeur pour les réseaux Internet. De nombreuses solutions avaient été suggérées pour résoudre le problème.

Dans ce chapitre, nous présentons les concepts de base sur les spams en générale et les spam images en particulier et les différentes techniques utilisées pour détecter ce type de spam.

1.2 Notions sur le spam

1.2.1 Définition du spam

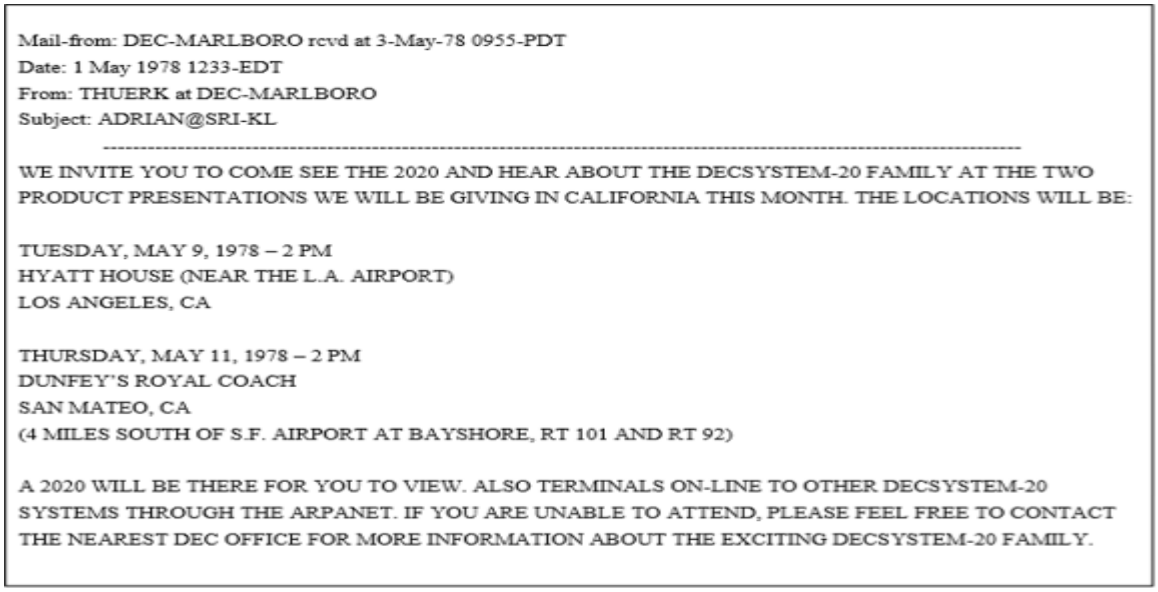
Le spam est un message électronique non sollicité, envoyé massivement à un grand nombre de destinataires, à des fins publicitaires ou malveillantes.

Le terme spam est aussi utilisé pour désigner le même type de message transmis par d'autres moyens de communication électroniques tels que les messageries instantanées, les blogs, les forums, et plus récemment, des réseaux de téléphonie mobile, via les SMS ou MMS. Même si le moyen de communication est différent, les techniques d'envoi et de détection restent relativement similaires.

Le premier spam (Figure 1.1) date du 3 mai 1978. Ce jour-là, sur le réseau ARPANET2, Gary Thuerk, commercial de la société informatique DEC3, invitait par e-mail 393 personnes à découvrir sa nouvelle machine, le 2020.

1.2.2 Types de spam

En plus des spams e-mail, il existe d'autres types de spam applicables à différents moyens de communication. Par exemple, les spams de messagerie sur téléphone mobile, ainsi que les spams des moteurs de recherche Web et les spams des réseaux sociaux (Dhanaraj & Karthikeyani, 2013).



Mail-from: DEC-MARLBORO rcvd at 3-May-78 0955-PDT
Date: 1 May 1978 1233-EDT
From: THUERK at DEC-MARLBORO
Subject: ADRIAN@SRI-KL

WE INVITE YOU TO COME SEE THE 2020 AND HEAR ABOUT THE DECSYSTEM-20 FAMILY AT THE TWO PRODUCT PRESENTATIONS WE WILL BE GIVING IN CALIFORNIA THIS MONTH. THE LOCATIONS WILL BE:

TUESDAY, MAY 9, 1978 – 2 PM
HYATT HOUSE (NEAR THE L.A. AIRPORT)
LOS ANGELES, CA

THURSDAY, MAY 11, 1978 – 2 PM
DUNFEY'S ROYAL COACH
SAN MATEO, CA
(4 MILES SOUTH OF S.F. AIRPORT AT BAYSHORE, RT 101 AND RT 92)

A 2020 WILL BE THERE FOR YOU TO VIEW. ALSO TERMINALS ON-LINE TO OTHER DECSYSTEM-20 SYSTEMS THROUGH THE ARPANET. IF YOU ARE UNABLE TO ATTEND, PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY.

Figure 1. 1 : Premier spam sur le réseau ARPANET2, Gary Thuerk.

Spam d'e-mail est la forme de spam la plus répandue. Dans le spam par e-mail, les messages sont envoyés à un grand nombre d'adresses e-mail. Ces messages de spam peuvent inclure des publicités de produits, des liens vers des sites Web de phishing ou des liens vers des installateurs de logiciels malveillants. Historiquement, le spam par e-mail ne contenait que des messages texte. Au fur et à mesure que les filtres textuels s'amélioraient, les spams basés sur des images sont apparus comme un moyen de contourner ces filtres (Annadatha & Stamp, 2018). Il existe de nombreuses autres formes de spam par e-mail, y compris ce que l'on appelle le spam vierge, qui ne contient aucun message dans l'e-mail et est utilisé pour collecter des adresses e-mail légitimes.

Spam de message de téléphone mobile (SMS) fait référence aux messages indésirables envoyés aux téléphones mobiles. De tels messages sont gênants pour les utilisateurs de téléphones mobiles, mais comme il y a des coûts associés au spam par SMS, il est moins courant que le spam par e-mail (Annadatha & Stamp, 2018).

Spam des moteurs de recherche fait référence aux mesures qui tentent d'affecter la position d'un site Web après une requête. En guise de contre-mesure, lorsqu'un site Web est détecté comme contenant du spam de moteur de recherche, le site est marqué et pénalisé. Une enquête a révélé que 51,3 % des piratages de sites Web étaient liés au spam des moteurs de recherche (Schwartz, 2018).

Spam des réseaux sociaux vise les sites Web de réseaux sociaux tels que Facebook et Twitter. Une technique de spamming social consiste à créer un faux compte dans une application sociale, qui est ensuite utilisé pour pirater des comptes d'utilisateurs valides. Ces faux comptes sont utilisés pour envoyer des messages en masse ou des liens malveillants, avec l'intention de nuire. À mesure que les sites de réseautage social sont devenus plus populaires, les activités de spam social telles que le clickbaiting ou le likejacking sont devenues plus courantes (Tolentino, 2015).

Spam de jeu consiste à envoyer des messages en masse aux joueurs en utilisant une salle de discussion commune ou une zone de discussion publique. Les spammeurs peuvent cibler les utilisateurs qui aiment les jeux afin de vendre des articles de jeu contre de l'argent réel ou de la monnaie du jeu.

1.2.3 Objectifs des spam

Au départ, le spam visait principalement des objectifs publicitaires. Aujourd'hui, il s'est considérablement développé, diversifié et complexifié, pour atteindre de plus en plus souvent des objectifs malveillants. En effet, Le spam s'est non seulement développé en termes de volume, mais également en termes de contenu. Aujourd'hui, les objectifs des spam sont très variés en voici une liste non exhaustive :

- **Hameçonnage (ou phishing)** : L'objectif est de réussir à se faire passer pour un organisme connu par l'utilisateur, dans le but de lui voler des informations à caractère confidentiel. Par exemple, on reçoit un mail provenant "apparemment" de notre banque, ou d'un autre site où l'on dispose d'informations personnelles. Dans ce mail, il est demandé de cliquer sur un lien (pour des motifs divers : Réactualisation, etc.), après avoir cliqué sur ce lien, une page web s'affiche... sur laquelle il est demandé de rentrer ses coordonnées bancaires ou toute autre information personnelle. Parmi les sites Top les plus contrefaits pour les attaques de phishing, on retrouve eBay, Paypal et Bank of America.
- **Publicité** : L'objectif est de vanter les mérites d'un produit quelconque. Il s'agit par exemple de produits pharmaceutiques, de produits de luxe, de logiciels divers et variés, de jeux d'argent. Ils peuvent également soutenir-agate idées politiques, culturelles ou religieuses et / ou organisations.
- **Scam** : Il s'agit d'une attaque basée sur la naïveté des destinataires dans le but de leur soutirer de l'argent. L'exemple le plus courant est le scan nigérien: un dignitaire d'un pays d'Afrique vous demande de servir d'intermédiaire pour une transaction financière importante, en vous promettant un bon pourcentage de la somme. Pour amorcer la transaction, il vous faut donner de l'argent.
- **Canular** : L'objectif est de faire circuler une information semblant très sensible, souvent avec un caractère d'urgence : fausse alerte de virus, fausse alerte de contamination potentielle, chaine de solidarité…. Par exemple : « un nouveau virus très dangereux se propage, il faut faire circuler l'information »; « des sous-vêtements sont infectés par une dangereuse bactérie ».
- **Malware** : Est un logiciel conçu pour infiltrer ou endommager un système informatique. Il est communément pris pour contenir des virus informatiques, vers, chevaux de Troie, spywares et adwares. Ce type de logiciel est souvent envoyé en tant que non suspect d'une pièce jointe. Lorsque l'utilisateur ouvre le fichier, le logiciel malveillant s'installe. L'interdépendance entre les spams et les logiciels malveillants a évolué Spam logiciels malveillants propagation des e-mails, les logiciels malveillants est utilisé pour infecter un hôte de sorte que l'hôte peut être contrôlé à distance et utilisé pour l'envoi de plus de spams. Ces hôtes infectés sont désignés comme des « ordinateurs zombies ». Beaucoup

de gens croient que la plupart des spams sont envoyés par des botnets, qui constituent un réseau de PC zombies.

1.2.4 Statistiques sur les spam

À titre indicatif, En 2015, le nombre d'utilisateurs de courrier électronique était de 2,6 milliards, tandis qu'en 2019, ce nombre passera à environ 2,9 milliards, avec plus d'un tiers de la population mondiale utilisant le courrier électronique pour échanger des messages Le nombre d'e-mails envoyés chaque jour dans le monde est de 293 milliards en 2019 (hors spams). Selon le rapport du laboratoire Kaspersky, en 2015, le volume de spams envoyés a été réduit à son plus bas niveau en 12 ans. Le volume de spams est tombé en dessous de 50 % pour la première fois depuis 2003. En juin 2015, le volume de spams est tombé à 49,7 % et en juillet 2015, les chiffres ont encore été réduits à 46,4 % selon le développeur de logiciels antivirus Symantec. Cette baisse a été attribuée à la réduction du nombre de botnets majeurs responsables de l'envoi de spams par milliards. Le volume de spams malveillants a été signalé comme étant constant en 2015. Le nombre de spams détectés par Kaspersky Lab en 2015 se situait entre 3 et 6 millions. À l'inverse, alors que l'année touchait à sa fin, le volume de spams a augmenté. Un autre rapport de Kaspersky Lab a indiqué que les messages spam contenant des pièces jointes pernicieuses telles que des logiciels malveillants, des rançongiciels, des macros malveillantes et JavaScript ont commencé à augmenter en décembre 2015. Cette dérive s'est poursuivie en 2016 et en mars de la même année, le volume de spam avait quadruplé par rapport à celle observée en 2015. En mars 2016, le volume de spams découverts par Kaspersky Lab est de 22 890 956. À ce moment-là, le volume de spams avait grimpé en flèche pour atteindre une moyenne de 56,92 % pour le premier trimestre de 2016. Les dernières statistiques montrent que les spams représentaient 56,87 % du trafic de courrier électronique dans le monde et que les types de spam les plus connus étaient les soins de santé et spam de rencontres. Le spam entraîne une utilisation improductive des ressources sur les serveurs SMTP (Simple Mail Transfer Protocol) car ils doivent traiter un volume important de courriers électroniques non sollicités. Le volume de spams contenant des logiciels malveillants et d'autres codes malveillants entre le quatrième trimestre de 2016 et le premier trimestre de 2018 est illustré dans la figure 2 ci-dessous. Selon une étude récente de Symantec, indiquent que 90,4 % des e-mails incluent du contenu spam.

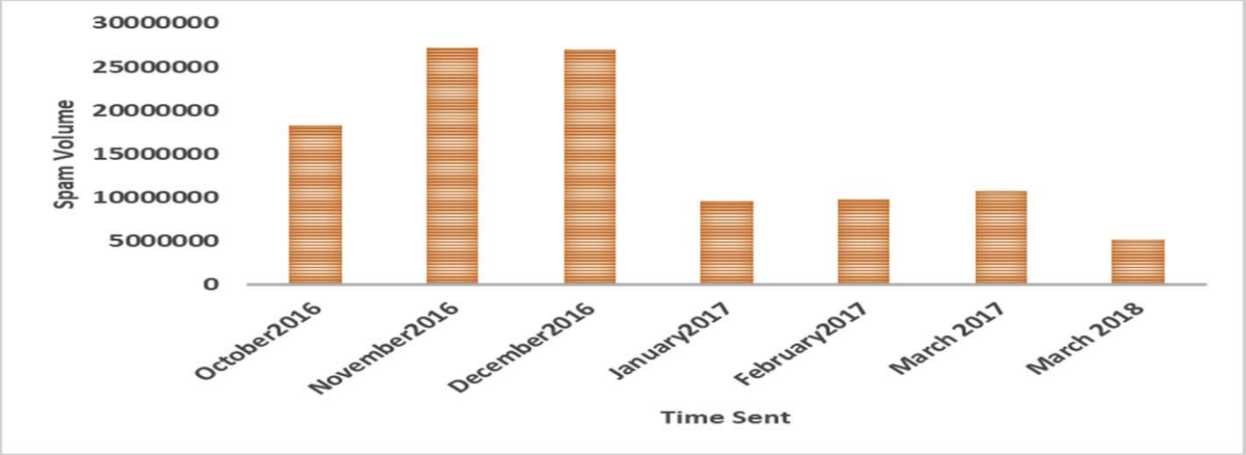


Figure 1. 2 : Le volume de spam emails du 4th quart 2016 to 1st quart 2018.

1.2.5 Impactes du spam sur les utilisateurs et les fournisseurs

Dans cette section, nous présentons les effets du spam, au niveau des utilisateurs, entreprises.

1.2.5.1 Perte de temps

- Encombrement anormal des boîtes aux lettres.
- Suppression des courriels indésirables.
- Configuration et maintenance des filtres.
- Consultation des courriels rejetés pour y détecter les bons à cause du risque de passer à côté d'emails importants mal catalogués par les outils de détection anti-spam.

1.2.5.2 Perte de bande passante et d'espace disque

- Spécialement pour les utilisateurs de modems.
- Les pièces jointes des virus et spam peuvent être grands.

1.2.5.3 Pertes financières non négligeables aux niveaux des entreprises et FAI

- une augmentation des coûts de gestion opérationnelle et support lié à la gestion anti spam.
- perte de productivité des salariés, Selon une étude, le spam aurait coûté environ 712 $ par employé et par an aux entreprises. À ce chiffre, il faut rajouter 113 à 183 $ par employé et par an pour la gestion des emails en quarantaine.

1.3 Notions sur le spam image

1.3.1 Définition

Le spam image est une sous-classe du spam e-mail. Le spam image est une sorte de spam dans lequel le texte du message est incorporé dans une image qui est ensuite jointes à l'e-mail. Comme mentionné ci-dessus, le spam image est apparu comme une technique d'obscurcissement pour échapper aux filtres anti-spam textuels. Le spam image est généralement utilisé pour faire la publicité de produits, tromper les utilisateurs pour obtenir des données personnelles ou pour diffuser des logiciels malveillants (Dhanaraj & Karthikeyani, 2013). Il est plus difficile de détecter le spam image que le spam textuel et les techniques d'offuscation basées sur l'image peuvent être utilisées pour créer un spam image encore plus difficile que celui généralement observé dans la pratique (Annadatha & Stamp, 2018 ; Chavda et al. 2018). Des exemples de spam d'images du monde réel sont donnés dans les figures 1.3, 1.4 et 1.5.

1.3.2 Les types

Les e-mails basés sur le spam image sont vus dans différents e-mails de spam avec différentes formes. Les e-mails basés sur le spam image sont généralement classés en trois branches :

- L'e-mail spam image contient une image qui présente la cible du spammeur et une URL qui adresse l'image du site Web du spammeur. Dans ce cas, l'utilisateur après avoir reçu le courrier doit saisir l'URL dans la barre d'adresse pour visiter le site Web. L'image exploitée doit donc être suffisamment attrayante pour persuader l'utilisateur de le faire.

- Le contenu du spam image est similaire à tout le contenu utilisé dans les spams textuels. On peut dire que l'image utilisée est une capture d'écran du spam textuel habituel. Toutes les cibles sont visibles dans l'image avec tous les détails que le spammeur souhaite partager avec les utilisateurs. Par exemple, si le spammeur souhaite afficher des annonces dans le spam image, il peut contenir le nom du produit, la description du produit, le nom du producteur, l'adresse, le numéro de téléphone, etc.
- Le spam image est un lien hypertexte vers un site Web. L'utilisateur après avoir cliqué sur l'image peut voir le site Web spécial qui contient toute la description de la cible du spammeur avec des détails complets. En raison de la curiosité de l'utilisateur, le site Web hyperlié s'ouvrait généralement par un simple clic sur l'image.

Figure 1. 3 : Exemples de spam image (Yan Gao et al., 2008).

Figure 1. 4 : Exemples de spams images (i) financiers, (ii) produits, (iii) Internet et (iv) loisirs

Figure 1. 5 : Exemple d'image non spam de l'ensemble de données

1.3.3 Types de spam images

Le spam image a évolué au fil du temps et peut prendre plusieurs formes pour contourner les techniques anti-spam classiques. Les images utilisées pour le spam peuvent inclure des images textuelles, des images découpées et des images aléatoires, comme indiqué ci-dessous.

- **Image texte uniquement** : Le spam image de première génération consistait en des images contenant uniquement du texte. De telles images contiennent du texte pur intégré dans une image essentiellement vierge. Ces images ressemblent à des e-mails textuels, mais sont en réalité des images. La reconnaissance optique de caractères (OCR) peut être utilisée pour extraire ce texte, auquel cas des filtres traditionnels basés sur le texte peuvent être appliqués.


Hello,

You have been chosen to participate in an invitation only limited time event!

Are you currently paying over 3% for your mortgaged STOP! We can help you lower that today!

Answer only a few questions and we can give you an approval in under 30 seconds it is that simple!

And stop fighting for lenders, let them fight for you! Make them work for your business by giving you the lowest rates around! Two hundred and thirty thousand dollar loans are available for only three hundred and forty dollars/month!

WE ARE PRACTICALLY GIVING AWAY MONEY!

Think your credit is too bad to get a deal like this? THINK AGAIN! We will have you saving your money in no time!

Are you ready to save your money?
Regards

Figure 1. 6 : Spam image texte uniquement

- **Image randomisée** font référence à la randomisation des pixels de l'image. Pour créer une image aléatoire, les spammeurs modifient chaque pixel de l'image. les spammeurs ajoutent des bandes de couleurs aléatoires, des pixels colorés aléatoires, des nuances de couleurs. Par conséquent, il peut être difficile de distinguer l'image randomisée de l'image d'origine. Les modifications apportées n'affectent généralement pas sensiblement l'apparence de l'image, mais modifient les valeurs de hachage et peuvent même influencer les résultats des techniques de détection basées sur l'OCR.

Figure 1. 7 : Image avec pixel de couleur aléatoire

- **Image obscurcie** : L'obscurcissement est l'une des plus anciennes astuces dans ce domaine, par ex. mots mal orthographiés, rotation légère du texte, flou des contours du texte, ajout d'ombre et ajout de bruit aléatoire pour rendre l'image difficile à lire et empêcher la détection de plusieurs images similaires.

Figure 1. 8 : Exemples d'obscurcissement utilisés dans le spam d'image.

- **Image découpée** se compose de plusieurs images fusionnées à la manière d'un puzzle. Ce type de spam image est difficile à détecter et l'image combinée passe souvent par les filtres anti-spam d'image.

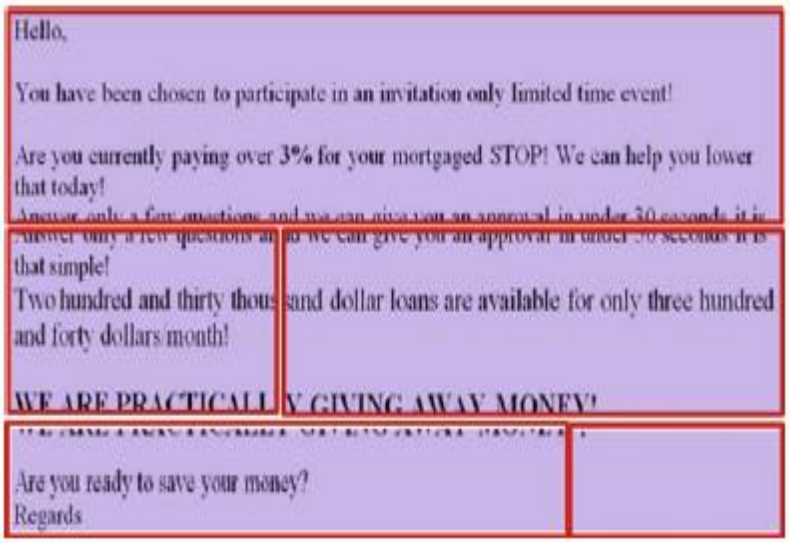


Figure 1. 9 : Image découpé

- **Image avec fond sauvage**: Les spammeurs utilisent des images avec des arrière-plans vagues et étranges. Ils utilisent des figures géométriques et des couleurs variées. Cette astuce cible l'OCR car l'OCR est normalement basé sur la mesure de la géométrie et recherche des formes géométriques similaires à des lettres et les place dans un fichier texte. Toutes les méthodes qui utilisent l'OCR sont la cible de cette astuce.

Figure 1. 10 : Image avec fond sauvage.

- **Images gif animées et en plusieurs parties** : Les images sont divisées en plusieurs parties, certaines contenant le message et d'autres contenant une animation. Les cadres de l'image tournent assez rapidement pour n'afficher que le résultat final à l'utilisateur. Ceux-ci sont créés en combinant plusieurs images gif dans un seul fichier, affichées l'une après l'autre, donnant l'apparence d'un mouvement.

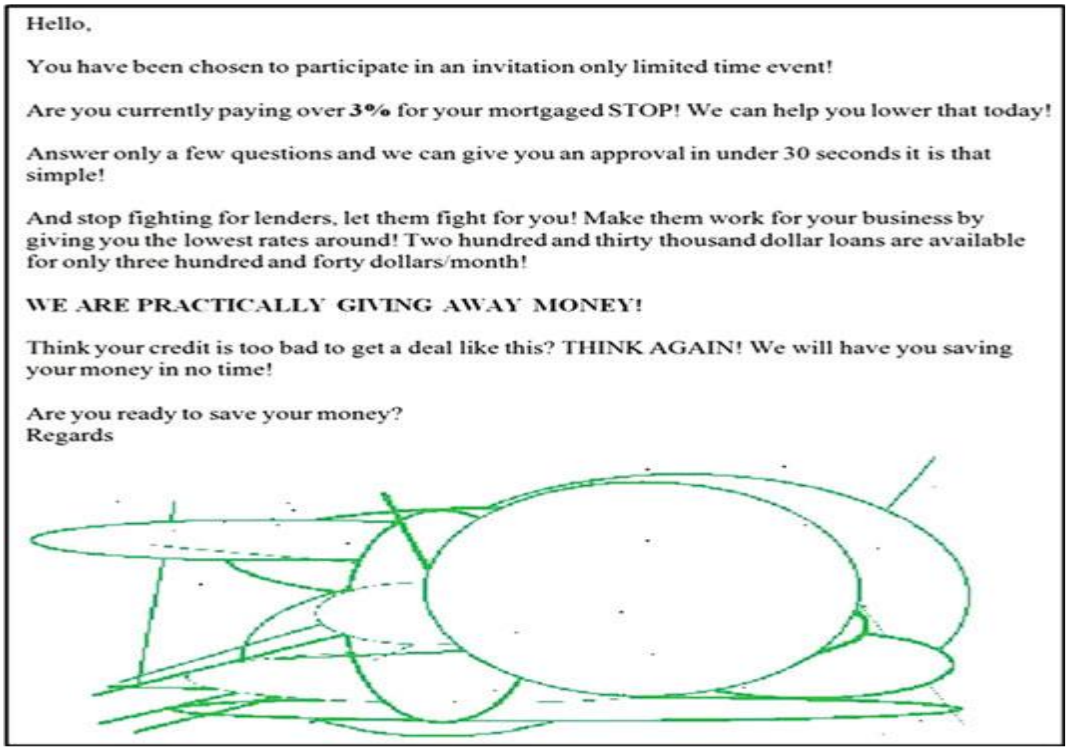
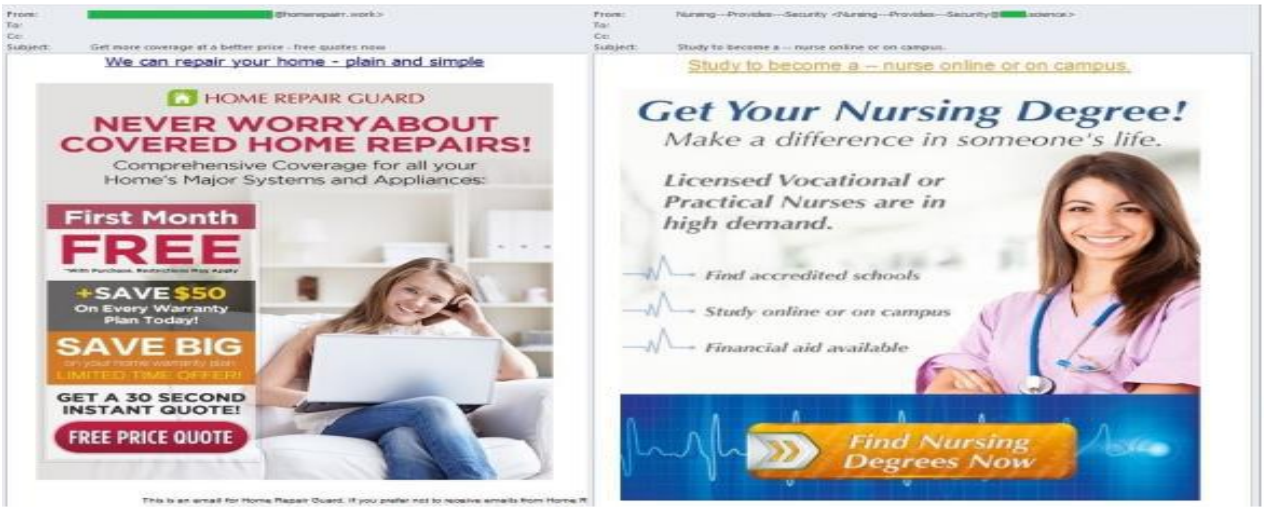


Figure 1. 11 : Exemple de spam image Gifs animés multi-images.

- **Dessin animé** : Cette nouvelle astuce utilise des polices de dessins animés qui sont inhabituelles et étranges pour le filtre anti-spam. Ils utilisent plusieurs couleurs, plusieurs styles et des formes spéciales afin de produire un très bel artefact attrayant et perceptible pour l'homme, mais très difficile à détecter pour la machine.

Figure 1. 12 : Exemple de spam image de dessin animé.

- **Images standard ou naturelle**: Ce sont des images soignées, aucune des astuces ci-dessus n'est utilisée et cela lui donne un aspect authentique. Le message entier est contenu dans l'image et les filtres anti-spam ne peuvent donc pas le détecter. En fait, de nombreuses images qui arrivent aujourd'hui comme spam ont un aspect professionnel, ce qui les fait ressembler à des photographies.



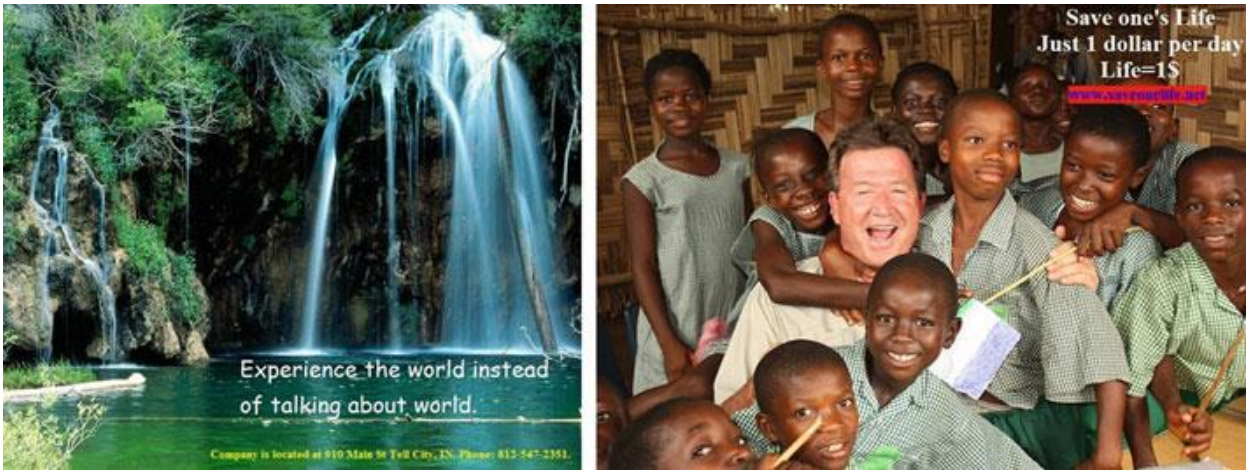


Figure 1. 13 : Exemple de spam images naturelles / standards.

1.3.4 Filtrage de spam image

Dans ce qui suit, nous discutons de trois approches de détection de spam image. Plus précisément, nous considérons les techniques basées sur les en-têtes, les techniques basées sur le contenu et les techniques non basées sur le contenu. Notez que ces techniques ne sont pas mutuellement exclusives.

1.3.4.1 Techniques basées sur l'en-tête

Aujourd'hui, les clients de messagerie modernes masquent souvent les en-têtes de la vue de l'utilisateur, c'est pourquoi de nombreuses personnes n'ont jamais vu d'en-tête de courrier électronique. Cependant, les en-têtes sont toujours livrés avec le contenu du message. La plupart des clients de messagerie offrent une option pour activer ou désactiver l'affichage de l'en-tête de l'e-mail. L'idée de ce groupe est d'analyser uniquement la partie en-tête des e-mails, car les messages électroniques contiennent plus qu'un simple message. L'en-tête de l'e-mail se compose de données sur l'expéditeur et le destinataire, y compris l'adresse e-mail de l'expéditeur, la date, de, à, etc. Les champs de l'en-tête de l'e-mail contiennent des informations précieuses qui peuvent être utiles pour distinguer le spam et le non-spam. Les attributs de l'en-tête de l'e-mail fournissent une gamme d'informations utiles pour l'analyse et la détection et ont été utilisés avec succès pour entraîner des modèles à détecter le spam (Hassanet al., 2017 ; Saraubon and Limthanmaphon, 2009 ; Liu et al. 2010). Ces techniques sont applicables au spam image. Certains des champs d'en-tête mentionnés sont utilisés dans la pratique et sont décrits ci-dessous

1.3.4.2 Techniques basées le contenu

Les filtres basés sur le contenu peuvent, par exemple, rechercher dans un e-mail des mots-clés particuliers qui se trouvent généralement dans le corps du message du spam. En règle générale, le corps d'un e-mail contient les informations réelles à fournir. Pour le spam image, les techniques basées sur l'OCR (Optical Character Recognition) peuvent être utilisé pour extraire le texte intégré dans l'image qui est ensuite transmis à un filtre basé sur le texte (Apache Software Foundation, 2018).Les techniques basées sur l'OCR utilisent des techniques de reconnaissance optique de caractères pour extraire le texte intégré dans les images spam, puis le

soumettre avec le corps du texte de l'e-mail à des techniques de détection basées sur le texte (Nisha et Gaikwad, 2015 ; Dredze, et al, 2007). Récemment, pour contourner ce type de filtre anti-spam, les spammeurs ont introduit des techniques d'obscurcissement des images spam afin d'empêcher les outils OCR de lire le texte intégré dans les images. Cela a soulevé la question de l'amélioration de la détection du spam image à l'aide d'autres techniques.

1.3.4.3 Technique non basée sur le contenu

Les techniques non basées sur le contenu sont destinées à étudier et analyser les caractéristiques et le contenu de l'image, tels que la couleur, la texture, le bord, l'ombrage, la surface, etc. L'objectif est d'utiliser ces caractéristiques de l'image pour filtrer les spam image. Des recherches sur la détection de spam image ont montré que certains types de spam image peuvent être détectés avec une grande précision. Par exemple, les travaux présentés dans (Annadatha & Stamp, 2018 ; Chavda, and al, 2018), une grande variété de propriétés d'image sont extraites et les images sont classées comme spam ou ham (c'est-à-dire, images non-spam) basée sur des techniques d'apprentissage automatique.

1.3.5 Travaux connexes

Depuis l'apparition des spam e-mails, la combinaison de techniques de traitement d'images et de techniques de classification a contribué au développement de systèmes de détection de spam image puissants. Certains travaux se concentrent sur la localisation de régions de texte dans une image spam, et d'autres utilisent différentes techniques permettant de distinguer les images légitimes (ham) des images spam. Ci-dessous, nous passerons en revue les recherches pertinentes et récentes liées à notre travail.

Dhabi et al. (2020) ont proposé une méthode de filtrage de spam image basée sur l'algorithme de détection de région de texte. La méthode proposée est mise en œuvre en plusieurs phases. Premièrement, les transformées en ondelettes basées sur le niveau unique doivent trouver l'image d'entrée pour détecter les régions de texte candidates. Deuxièmement, un ensemble de caractéristiques fiables indiquant le spam sont extraites des régions de texte détectées.

Sharmin et al., (2020), ont proposé d'appliquer les réseaux de neurones convolutifs (CNN) au problème de détection du spam image et d'utiliser des CNN basés sur un nouvel ensemble de fonctionnalités qui est une combinaison de l'image brute et Canny edges.

Annadatha et al. (Annadatha & Stamp, 2018), utilise l'algorithme de classification SVM qui a été appliqué à un ensemble de 21 caractéristiques d'image. En utilisant une sélection de fonctionnalités basée sur des poids SVM linéaires, les auteurs sont en mesure d'atteindre un taux de précision de 0,97 avec un ensemble relativement petit de caractéristiques. Les auteurs fournissent un ensemble de données qui pourrait servir de spam image, mais qui est beaucoup plus difficile à détecter, par rapport au spam image du monde réel.

Chavda et al. (Chavda et al., 2018) mènent deux séries d'expériences avec SVM et traitement d'image. Les auteurs utilisent un ensemble complet de 41 caractéristiques d'image et ils obtiennent une précision de 0,97 et 0,98 sur deux ensembles de données accessibles au public. Ces auteurs fournissent également un ensemble de données, qui s'avère encore plus difficile à détecter que celui développé dans (Annadatha & Stamp, 2018).

Aiwan et al. (Aiwan & Zhaofeng, 2018) proposent une méthode de filtrage de spam image basée sur le réseau de neurones convolutifs (CNN). Le système proposé utilise l'augmentation des données et réalise une amélioration de la précision, par rapport à des exemples sélectionnés de travaux antérieurs.

Kumar et al. (Kumar, R, & KP, 2018) appliquent des techniques d'apprentissage en profondeur au problème du spam d'image. Ils obtiennent une précision d'environ 91%, ce qui est comparable à d'autres recherches dans le domaine.

Chang (2017) a proposé un système de filtrage de spam image à trois couches. Le filtrage est effectué en analysant à la fois l'en-tête de l'image et l'image elle-même. La structure du modèle explique clairement l'idée de conception et les technologies liées au modèle. Les résultats expérimentaux montrent que le taux de classification est d'environ 93,0 %.

Hosseini et. al (2015) a suggéré une méthode qui utilise les caractéristiques de texture d'image pour classer l'image de spam. Pour chaque image, la matrice de cooccurrence de niveaux de gris a été utilisée pour obtenir 22 caractéristiques, qui sont utilisées par le bayésien naïf et le k plus proche voisin (KNN).

Kumaresan et. al (2015) a proposé un schéma pour extraire les caractéristiques, en particulier les métadonnées et les caractéristiques d'histogramme des images. Sur la base de ces caractéristiques extraites, un classificateur SVM avec fonction noyau est utilisé pour détecter spam image. La complexité temporelle reste un problème pour cette méthode, mais la précision est de 90 %.

Chowdhury et al (2015) proposent une méthode qui extrait les métadonnées et les caractéristiques visuelles et les transmet au BPNN pour classification. Ils ont comparé trois algorithmes d'apprentissage automatique qui sont : SVM, Naïve Bayes et BPNN avec le même ensemble de fonctionnalités et sur le même ensemble de données.

Nisha D. Chopra et. al (2015) a proposé un système qui utilise deux méthodes pour classer les images spam. La première méthode utilise un outil OCR pour extraire le texte de l'image, et la seconde méthode utilise un classificateur bayésien pour détecter les spams.

Kumaresan et al. (Kumaresan, Sanjushree et Palanisamy, 2014) proposent une technique de détection de spam image basée sur les caractéristiques de couleur et utilisant l'algorithme *k*-plus proche voisin (K-NN). Plus précisément, les auteurs s'appuient sur les histogrammes RVB et HSV comme caractéristiques. Dans cette recherche, un classificateur simple K-NN donne une précision de 0,945.

Anand et. al (2012) présentent deux méthodes pour filtrer les spam image. La première méthode extrait les caractéristiques de bas niveau tandis que la seconde méthode extrait les caractéristiques de métadonnées de l'image. Les deux méthodes sont appliquées aux images contenant uniquement des zones de texte. Dans les deux procédés, les régions de texte sont extraites pour être entrées dans le système OCR.

Gao et al. (Yan Gao et al. 2008) proposent un schéma de détection de spam image qui s'appuie sur un algorithme d'arbre de boosting probabiliste. L'ingénierie des caractéristiques basée sur la couleur et l'histogramme du gradient orienté (HOG) est utilisée pour générer des vecteurs de

caractéristiques pour cet algorithme d'apprentissage. Ces auteurs obtiennent une précision de 0,8944.

En plus des recherches qui traitent exclusivement les spam image, il existe de nombreux articles sur la détection de spam qui couvrent les aspects du problème de spam image. Par exemple, (Dada et al, 2019) réalise un et de l'art sur les articles de recherche traitant sur la détection de spam.

1.4 Conclusion

Dans ce chapitre, nous avons présenté quelques concepts de base sur les spams en générale et les spams images en particulier avec leurs définitions, objectifs et impacts et sur les différentes techniques utilisées pour détecter ce type de spam.

Chapitre 2 : Apprentissage profond (Deep Learning)

2.1 Introduction

La traduction automatique moderne, les moteurs de recherche et les assistants informatiques sont tous alimentés par l'apprentissage en profondeur. Cette tendance ne fera que se poursuivre à mesure que l'apprentissage en profondeur étendra ses réseaux à la robotique, à la pharmacie, à l'énergie et à tous les autres domaines technologiques contemporains. Les modèles d'apprentissage en profondeur tentent d'imiter, aussi fidèlement que possible, les modèles de traitement de l'information et de communication dans les systèmes nerveux biologiques, tels que le codage neuronal, qui tentent de définir et de décrire la relation entre plusieurs stimuli et les réponses liées aux neurones dans le cerveau.

Le Deep Learning est un nouveau domaine de recherche du Machine Learning, qui a été introduit dans le but de rapprocher le ML de son objectif principal : L'intelligence artificielle. Il concerne les algorithmes inspires 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.

Dans ce deuxième chapitre, nous présentons quelques notions concernant le deep learning.

2.2 Apprentissage En Profondeur (Deep Learning)

2.2.1 Définition

Selon les fondateurs Yann LeCun, YoshuaBengio Geoffrey Hinton "l'apprentissage profond permet aux modèles informatiques composé plusieurs couches de traitement d'apprendre des représentations de données avec plusieurs niveaux d'abstraction.

L'apprentissage profond est un sous-ensemble de l'apprentissage automatique, qui est un ensemble d'algorithmes inspirés de la structure et de la fonction du cerveau humain (neurone biologique) appelé réseaux de neurone artificiel(ANN), ce dernier se combine afin de construire les réseaux de neurones profonds (DNN).

Dans DL, un modèle informatique apprend à effectuer des tâches de classification directement à partir d'images, de texte ou de son. Les modèles d'apprentissage en profondeur peuvent atteindre une précision de pointe, dépassant parfois les performances humaines, qui sont entraînés à l'aide d'un grand nombre de données étiquetées et d'architectures de réseaux neuronaux qui contiennent de nombreuses couches.

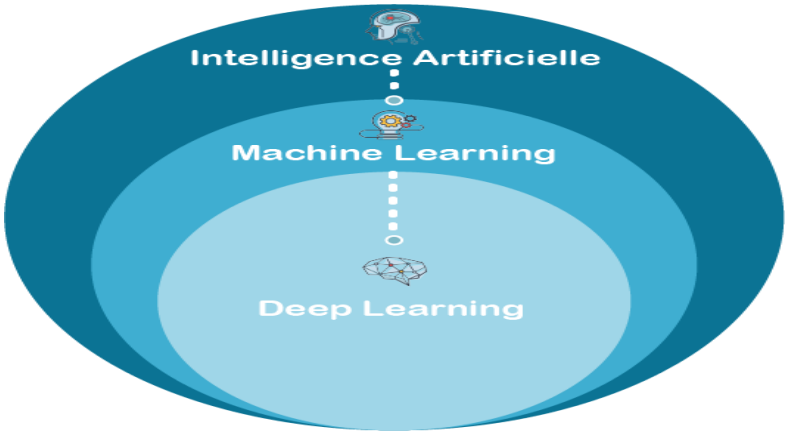


Figure 2. 1 : La relation entre IA, ML et DL.

2.2.2 Pour quoi le choix Deep Learning

Les algorithmes de ML fonctionnent bien pour une grande variété de problèmes. Cependant ils sont échoues a résoudre quelques problèmes majeurs de l'IA telle que la reconnaissance vocale et la reconnaissance d'objets. Tout d'abord les différents algorithmes du deep Learning ne sont apparus qu'a l'échec de l'apprentissage automatique tentant de résoudre une grande variété de problèmes de l'intelligence artificielle (l'IA) :

- Afin d'améliorer le développement des algorithmes traditionnels dans de telles taches de l'IA.
- De développer une grande quantité de données telle que les big data.
- De s'adapter à n'importe quel type de problème.
- D'extraire les caractéristiques de façon automatique.

2.2.3 Réseaux de neurones

Un réseau de neurones est un ensemble de neurones formels fortement interconnectés permettant de recevoir des signaux et de les modéliser afin de produire une seule sortie. Le neurone formel a été inventé par McCulloch and Pitts (1943) en proposant une première modélisation simplifiée d'un neurone biologique et prouvant qu'un neurone formel est capable de réaliser des fonctions logiques et arithmétiques. Cela a permis au psychologue Rosenblatt (1958) de proposer le modèle du perceptron appliqué pour la tâche de reconnaissance de forme.

2.2.3.1 Perceptron

L'origine des réseaux de neurones modernes se trouve dans le perceptron. Cet algorithme de classification, présenté en figure 2.2, peut se modéliser sous la forme d'une fonction de décision f qui prends en entrée un vecteur, ou tenseur, $x = [x_1 \ldots . x_n]^T$ et dont la sortie sera un scalaire $\hat{y}$ dont la valeur servira à déterminer la classe d'appartenance de l'entrée x.

Cette fonction de décision f peut se décomposer en deux étapes :

• Une pré-activation a qui correspond à la combinaison linéaire de x avec un vecteur de poids $w = [w_1, \dots, w_n]^T$, tel que $a(x) = w^T x$.

• Une activation h qui prends en entrée la sortie de la pré-activation en lui ajoutant un terme de biais b et dont la sortie correspondra à $\hat{y}$ tel que :

$$\hat{y} = h\big(a(x)\big) = h(w^T x + b) \qquad (2.1)$$

Pour simplifier la notation nous désignerons par θ le vecteur des paramètres : $\theta = [b, w_1, \dots, w_n]^T$ de manière a pouvoir écrire :

$$\hat{y} = f(x, \theta) = h\big(a(x)\big) = h(w^T x + b) \qquad (2.2)$$

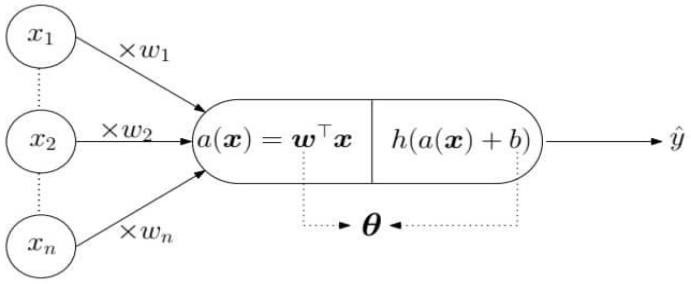


Figure 2. 2 : Percepton

Dans la forme originelle du perceptron, h était la fonction de Heavyside, présentée dans la figure suivante :

$$heavyside(x) = \begin{cases} -1 & si\ x \leq 0 \\ 1 & si\ x > 0 \end{cases} \qquad (2.3)$$

L'utilisation du perceptron est limitée à des problèmes de classifications binaires où l'une des classes correspondra à yˆ = 0 et l'autre à yˆ = 1. De plus, le perceptron ne peut être utilisé que sur des problèmes de classifications ou les variables d'entrées x sont linéairement séparables (MANZANERA & SAUX, 2020).

2.2.3.2 Descente de gradient

Une des façons de minimiser *J* couramment utilisée pour les réseaux de neurones est d'effectuer ce qu'on appelle une rétro-propagation de l'erreur. Un algorithme d'optimisation couramment utilisé pour la rétro propagation appliquée aux réseaux de neurones est la descente de gradient ou GD. Le but de la GD est de mettre à jour de façon itérative les poids du réseau en utilisant le gradient de la fonction de coût J à partir des données d'entrainement x_{train}. Quand *J* est évaluée sur des éléments de x_{train} nous l'appellerons J_{train}. Le déroulé de la GD est décrit dans l'algorithme suivant et illustré dans la figure 2.3 :

Algorithme Descente de gradient

Require: x_{train} , y_{train}
Require: $n_{itérations}$ Détermine le nombre d'itération de la GD
Require: ϵ Détermine la vitesse de la GD Initialiser θ
For $n_{itérations}$ **do**
$\nabla_\theta J_{train} \leftarrow \frac{\delta}{\delta\theta}\frac{1}{card(x_{train})}\sum_{t=1}^{n_{itérations}} E(f(x^{(t)},\theta)y^{(t)})$ Calcul du gradient de la fonction de coût E
$\theta \leftarrow \theta - \epsilon\nabla_\theta J_{train}$ Mise à jour des poids du réseau
End for

Le calcul du gradient est effectué pour l'ensemble des couples $x^{(t)}, y^{(t)}$ disponibles à l'entrainement. Un passage complet sur les données d'entrainement sera appelé époque. En pratique, le nombre d'itérations de l'algorithme de la GD sera souvent appelé nombre d'époques. Il s'agit d'un hyper paramètre.
Le coefficient ϵ est appelé taux d'apprentissage. Il détermine l'amplitude des étapes de la GD et influe donc sur la vitesse de l'apprentissage.
Pour l'instant nous considérons simplement que chaque poids du vecteur θ est initialisé aléatoirement. Notons simplement qu'initialiser θ à 0 ne fonctionne pas car la valeur de $\nabla_\theta J_{train}$ serait aussi nulle et l'algorithme de GD ne pourrais pas démarrer. Nous pouvons voir pendant l'apprentissage que l'algorithme de la GD re-propage le résultat de fonction de coût pour mettre à jour les poids du réseau. En soustrayant à chaque poids du Intuitivement, et comme illustré dans la figure suivante, cela revient à se déplacer sur la surface que définit la fonction d'erreur dans l'espace de θ vers son minimum. Comme on utilise la valeur négative du gradient, ou $-\nabla_\theta J_{train}$, calculé sur l'ensemble des exemples d'entrainement, la direction du déplacement correspondra à la pente la plus forte sur la surface de J_{train}. Le cas présenté ici est un cas simple avec $\theta = \{\theta_1\theta_2\}$ et ou la fonction d'erreur ne présente qu'un minimum global matérialisé par θ_{ideal} dans la figure 2.3 :

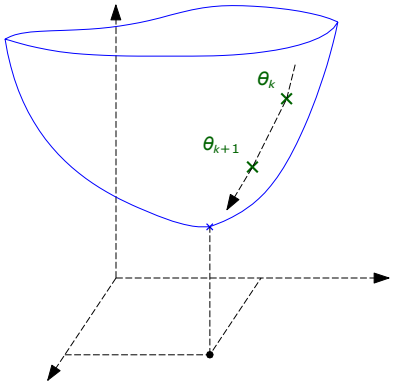


Figure 2. 3 : Représentation simplifiée de la descente de gradient.

Néanmoins, la surface J_{train} est rarement parfaitement convexe. Elle peut présenter des zones particulières, présentées dans la figure 2.4, telles que les minimums locaux, évoqués précédemment, mais aussi des points cols, ou des plateaux. Comme la GD progresse vers la pente la plus forte l'apprentissage peut se retrouver entrainé vers une zone de J_{train} qui est loin

du minimum global et dont il est difficile voir impossible de s'extraire même en jouant sur le taux d'apprentissage ϵ.

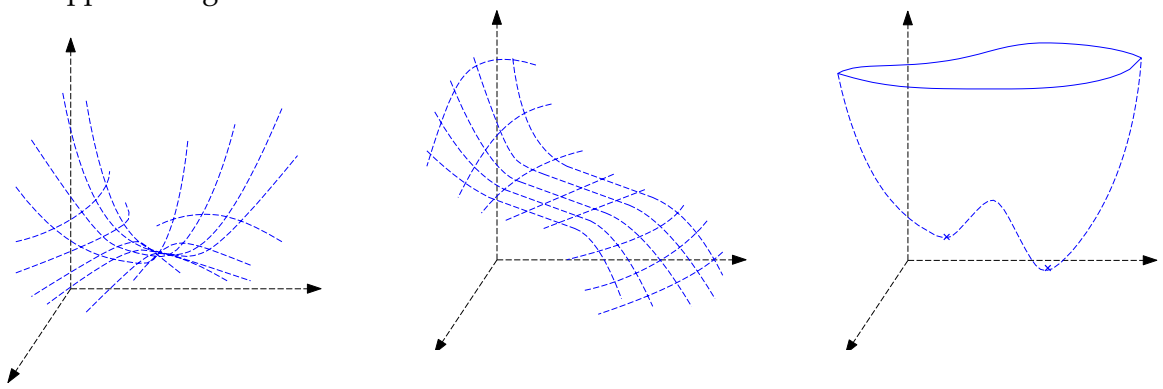


Figure 2. 4 : Illustration des imperfections de la surface représentée par Jtrain simplifiée en J(θ) ici

De plus, Le cout calculatoire associé à la grandeur J_{train} dans l'algorithme de la GD est directement proportionnel à n_{train} Pour un dataset X_{train}de grande taille, ce coût peut devenir très important et impacter l'apprentissage (MANZANERA & SAUX, 2020).

2.2.3.3 Fonctions d'activation

Les fonctions d'activation sont des fonctions utilisées dans les réseaux de neurones pour décider si un neurone doit être activé en calculant une somme pondérée et en ajoutant un biais. Le but de la fonction d'activation est d'introduire une non-linéarité dans la sortie du neurone. Il manipule les données présentées à travers un traitement de gradient généralement gradient descent (GD) et produit ensuite une sortie pour le réseau de neurones, qui contient les paramètres dans les données. Les principales fonctions d'activations sont :

Sigmoid : est une fonction populaire depuis des décennies de nature non linéaire, La fonction sigmoid est donnée par la relation :

$$f(x) = \frac{1}{1+exp^{-x}} \qquad (2.4)$$

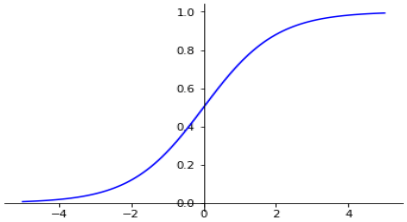


Figure 2. 5 : Représentation graphique des fonctions d'activations sigmoid.

La fonction sigmoid généralement utilisé dans la couche de sortie d'une classification binaire, où le résultat est soit 0, soit 1, car la valeur de la fonction sigmoïde est comprise entre 0 et 1, le résultat peut être facilement prédit comme 1 si la valeur est supérieure à 0,5 et 0 sinon.

Hyperbolic Tangent (Tanh) : est une fonction de type linéaire connue sous le nom de fonction Tanh C'est en fait une version mathématiquement décalée de la fonction sigmoïde. Les deux sont similaires et peuvent être dérivés l'un de l'autre, cette fonction est représentée par la relation :

$$f(x) = \frac{exp^{x} - exp^{-x}}{exp^{x} + exp^{-x}} \qquad (2.5)$$

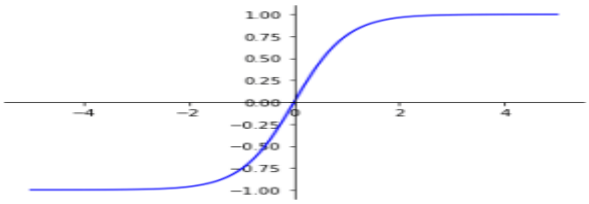


Figure 2. 6 : Représentation graphique des fonctions d'activations Tanh.

La fonction Tanh Généralement utilisé dans les couches cachées d'un réseau de neurones car ses valeurs sont comprises entre -1 et 1, donc la moyenne de la couche cachée est de 0 ou très proche de celle-ci, ce qui aide à centrer les données en rapprochant la moyenne de 0 Cela rend l'apprentissage pour la couche suivante beaucoup plus facile. Ainsi elle donne de meilleures performances d'apprentissage pour les réseaux de neurones multicouches par rapport à la fonction sigmoid.

Rectified Linear Unit (ReLU) : est un autre type de fonction d'activation a été proposée par Nair et Hinton 2010, de type non linéaire, ce qui signifie que nous pouvons facilement rétro propager les erreurs et activer plusieurs couches de neurones. La fonction d'activation ReLU effectue une opération de seuil pour chaque élément d'entrée où les valeurs inférieures à zéro sont mises à zéro, ainsi ReLU est donné par la relation :

$$f(x) = max(0, x) = \begin{cases} x_i, & if \ x_i \geq 0 \\ 0\,, & x_i < 0 \end{cases} \qquad (2.6)$$

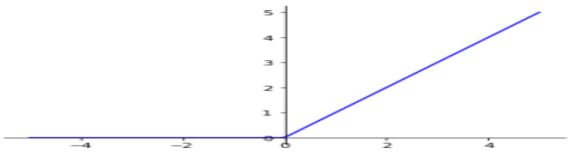


Figure 2. 7 : Représentation graphique des fonctions d'activations ReLu.

ReLu est moins coûteux en calcul que Tanh et sigmoïde car il implique des opérations mathématiques plus simples. À la fois, seuls quelques neurones sont activés, ce qui rend le réseau clairsemé, ce qui le rend efficace et facile à calculer. En termes simples, RELU apprend beaucoup plus rapidement que la fonction sigmoïde et Tanh.

Softmax : est également un type de fonction sigmoïde et de nature non linéaire mais elle est pratique lorsque nous essayons de gérer des problèmes de classification. Cette fonction est utilisé lorsque vous essayez de gérer multi classes. La fonction Softmax compresserait les sorties

pour chaque classe entre 0 et 1 et diviserait également par la somme des sorties. C'est calculé en utilisant la relation :

$$f(x) = \frac{exp^{x_i}}{\sum_j exp^{x_j}} \qquad (2.7)$$

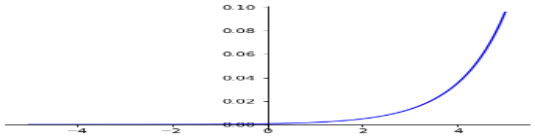


Figure 2. 8: Représentation graphique des fonctions d'activations Softmax.

La fonction Softmax est idéalement utilisée dans la couche de sortie du classificateur où elle renvoie les probabilités de chaque classe, la classe cible ayant la probabilité la plus élevée.

2.2.3.4 Perceptron multi couches (M.MINSKY AND S.PAPERT 1969)

En reliant plusieurs perceptrons ou neurones artificiels, il est possible de former un perceptron multi couches ou MLP pour Multi-Layer Perceptron. Ces modèles permettent de traiter des problèmes plus complexes comme la classification multi-classes. Les neurones du MLP sont regroupés en une série couches successives reliées entre elles pour former un graphe orienté visible dans la figure suivant :

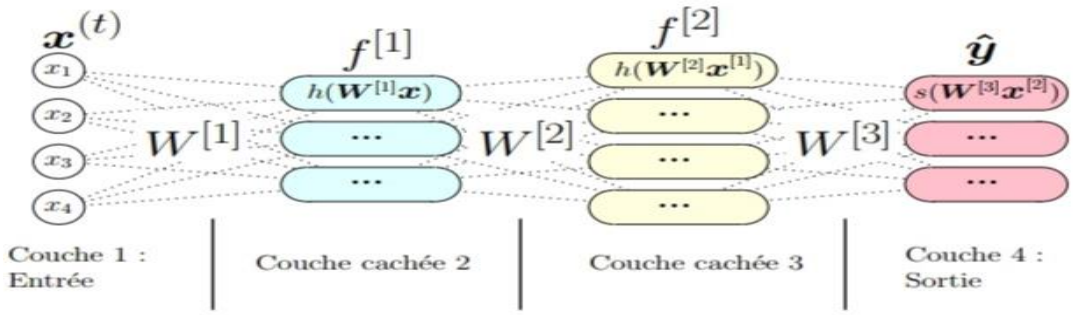


Figure 2. 9 : Perceptron multi-couches.

Un MLP peut aussi être appelé réseau de neurones. Comme pour les perceptrons simples, les MLP prennent en entrée un vecteur $x^{(t)}$qui est transformé par les neurones des couches intermédiaire, appelées couches cachées, jusqu'à la dernière couche dite couche de sortie. C'est cette dernière couche qui servira à déterminer la classe de sortie de $x^{(t)}$. Une couche cachée k ne sera directement reliée qu'à la couche qui la précède et à la couche qui la suit dans le réseau. Les sorties de la couche $k-1$ seront les entrées de la couche k et les sorties de k seront les entrées de $k-1$. Enfin, dans chaque couche, les neurones fonctionnent comme le perceptron décrit dans la figure 2.9. Chaque neurone d'une couche est donc connecté à l'ensemble des neurones de la couche précédente par des liaisons pondérées par son vecteur de poids et un terme de biais b. Ce dernier est absent sur la figure 2.9 pour alléger la notation.

Ainsi pour un neurone quelconque dans une couche k du MLP nous pouvons exprimer sa pré-activation comme étant :

$$a^{[k]}\left(x^{(t)}\right) = w^{[k]}h\left(a^{[k-1]}\left(x^{(t)}\right)\right) + b \qquad (2.\ 8)$$

Et son activation totale :

$$f^{[k]}\left(x^{(t)};\theta^{[k]}\right) = h^{[k]}\left(a^{[k]}\left(x^{(t)}\right)\right) \qquad (2.\ 9)$$

Le terme $w^{[k]}$ est une matrice $n*m$ avec m le nombre de neurones de la couche $k-1$ et n le nombre de neurones de la couche k où chaque ligne correspond à un vecteur w des poids de chaque neurone de la couche. Le terme $b^{[k]}$ correspond au vecteur regroupant l'ensemble des biais des neurones de la couche$k$.

À l'image de la notation utilisée pour le perceptron nous désignerons par θ l'ensemble des paramètres et biais utilisés dans le MLP i.e. $\theta = w^{[1]}, b^{[1]}, \cdots, w^{[k]}, b^{[k]}, \cdots, w^{[l]}b^{[l]}$Avec l le nombre de couches présentes dans le MLP.

Chaque neurone utilise aussi une fonction d'activation h comme pour le perceptron simple. Cette fonction d'activation est généralement la même pour l'ensemble des neurones des couches cachées.

Nous pouvons voir que la valeur de $f(x,\theta)$ est le résultat de la propagation vers l'avant de l'information issue du vecteur$x$, transformé par les différents neurones du réseau. Il n'y a ni cycle, ni boucle dans le chemin de circulation de l'information qui "avance" de x vers la couche de sortie. En raison de ce comportement, ces réseaux sont appelés "Réseaux de neurones à propagation vers l'avant" ou feedforward neural networks (MANZANERA & SAUX, 2020).

2.2.3.5 Backpropagation

Appliquant les idées de Bryson et Kelley aux perceptrons multicouches, Paul Werbos a proposé une extension de la règle d'apprentissage μ − LMS, appelée algorithme de Backpropagation. Considérons un perceptron multicouche avec K couches de neurones $y_0 = x$ vecteur d'entrée de taille 1*N, et les k couches suivantes $y_k = \emptyset(y_{k-1} \dots w_k), \forall\, k = 1, \dots, k$ avec les fonctions d'activation $\emptyset_k(z)$, où y_k est le vecteur de sortie de la taille 1*M. Si les fonctions $\emptyset_k(z)$ sont différenciables, nous pouvons calculer les gradients pour la couche y_{k-1}en utilisant les gradients pour la couche y_k en utilisant la règle :

$$\frac{\partial l}{\partial y_{k-1}^{i}} = \sum_{j=1}^{M} \frac{\partial l}{\partial y_k^j} \cdot \frac{\partial y_k^j}{\partial y_{k-1}^{i}} \sum_{j=1}^{M} \frac{\partial l}{\partial y_k^j} \cdot \frac{\partial \emptyset_k(z)}{\partial z} \qquad (2.\ 10)$$

Ici w j k est le vecteur de poids, reliant la couche k − 1 au neurone j sur la couche k. Comme nous pouvons le voir, les gradients de la couche précédente$\frac{\partial l}{\partial y_{k-1}}$dépendent uniquement des gradients de la couche actuelle$\frac{\partial l}{\partial y_k}$, de la dérivée de la fonction d'activation $\nabla_z \emptyset_z(z)$ et des poids de couche w_k. Par conséquent, il est possible de calculer de manière itérative les gradients pour

toutes les couches de $k-1$ à 0, en utilisant $\frac{\partial l}{\partial y_k}$ lors de la première itération. Les gradients $\frac{\partial l}{\partial w_k}$ peuvent également être calculés à l'aide de la règle :

$$\frac{\partial l}{\partial w_k^{ij}} = \frac{\partial l}{\partial y_k^j} \cdot \frac{\partial y_k^j}{\partial w_k^{ij}} = \frac{\partial l}{\partial y_k^j} \cdot \frac{\partial \phi_k(z)}{\partial z} \qquad (2.\ 11)$$

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

L'algorithme peut être divisé en trois parties : forward étape, backward étape et calcul du gradient de poids. Sur le forward étape, nous initialisons le vecteur d'entrée y_0 avec un objet d'apprentissage x et calculons de manière itérative les valeurs de couche suivantes y_i de 1 à K. Sur la passe arrière, nous initialisons les gradients de prédiction $\frac{\partial l}{\partial y_k}$avec l'équation, et les propageons à travers le réseau pour les couches de $k-1$ à 0, obtenant les gradients $\frac{\partial l}{\partial y_i}$peuvent ensuite être calculés en parallèle ou après le backward étape, en utilisant les valeurs de couche yi et leurs gradients$\frac{\partial l}{\partial y_i}$. Les gradients de poids sont ensuite utilisés l'équation (2.12) pour mettre à jour les poids utilisant l'équation (2.13). Lorsque la taille d'un batch est supérieure à 1, la moyenne des gradients de poids est utilisée. Habituellement, l'algorithme de backpropagation traite plusieurs fois le même ensemble d'apprentissage, jusqu'à ce que l'erreur de test cesse de diminuer (DEMYANOV, 2015).

$$\frac{\partial l}{\partial \hat{y}^i} = \hat{y}^i - y^i \Longrightarrow \frac{\partial l}{\partial \hat{y}} = \hat{y} - y \qquad (2.\ 12)$$

$$w \leftarrow w - \alpha \frac{\partial l(w)}{\partial w} \qquad (2.\ 13)$$

2.2.3.6 Types d'architecture des réseaux de neurones

- **Feedforward Neural Networks (FNN)**

Un réseau à action directe est un réseau neuronal simple composé d'une couche d'entrée, d'une couche de sortie et d'une ou plusieurs couches de neurones. En évaluant sa sortie en examinant son entrée, la puissance du réseau peut être remarquée sur la base du comportement de groupe des neurones connectés et la sortie est décidée. Le principal avantage de ce réseau est qu'il apprend à évaluer et à reconnaître les modèles d'entrée.

- **AutoEncoder (AE) :**

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

- **Restricted Boltzmann Machines (RBM)**

RBM est un ANN stochastique génératif qui peut apprendre la distribution de probabilité sur ses données d'entrée. Ils peuvent être apprendre de façon supervisés ou non supervisés selon la

tâche avec une restriction où la paire de nœuds des unités visibles et cachées peut avoir une connexion symétrique entre eux et il n'y a pas de connexion entre les nœuds au sein d'un groupe.

- **Deep Belief Networks (DBN)**

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

- **Convolution Neural Networks (CNN)**

Les CNNs sont un type spécifique de réseau neurones conçu pour la reconnaissance d'images tout comme les ANNs ont des nœuds (neurones), les CNN utilisent des filtres(Kernels), qui sont essentiellement des grilles de neurones qui peuvent apprendre des motifs dans les images. Les filtres des couches initiales de CNN apprennent des features de base comme les lignes horizontales et verticales. Alors que nous avançons plus profondément dans les couches, le filtre commence à reconnaître les caractéristiques complexes comme les oreilles, le nez, la bouche, etc., et dans la couche finale, le réseau peut identifier l'image entière basée sur ce réseau CNN.

- **Recurrent Neural Networks (RNN)**

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

- **Generative Adversarial Networks (GAN)**

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

2.2.4 Architecture profond

L'architecture profond fait référence au nombre de niveaux de composition d'opérations non linéaires dans la fonction apprise. Inspirés par la profondeur architecturale du cerveau, les chercheurs en réseaux de neurones souhaitaient depuis des décennies apprennent des réseaux de neurones multicouches profonds, mais aucune tentative réussie n'a été signalée avant 2006 (Sauf pour les CNNs) : les chercheurs ont rapporté des résultats expérimentaux positifs avec généralement deux ou trois couches (c.-à-d une ou deux couches cachées), mais l'apprentissage des réseaux plus profonds ont donné des résultats toujours plus faibles(BENGIO & al., 2007).

2.2.5 Techniques d'optimisations dans DL

L'optimiseur est un hyper paramètre qui a une grande influence sur la performance et la convergence d'un MLP.

2.2.5.1 Descente de gradient stochastique (SGD)

Dans la méthode GD, les gradients sont calculés en se basant sur toute la base d'apprentissage, cela limite son utilisation sur les grands volumes de données à cause de l'espace limité de la mémoire et le temps d'apprentissage très élevé. Afin de résoudre ces problèmes, la méthode de descente de gradient stochastique (SGD) est exploitée.

En SGD, le calcule des gradients et la mise à jour des poids est appliquée à chaque instance dans la base d'apprentissage. Ces instances sont choisies aléatoirement durant le processus d'optimisation. La sélection aléatoire et l'exploitation d'une seule instance permettent d'accélérer le temps d'apprentissage et d'améliorer la généralisation. Malgré ces avantages, les mises à jour fréquentes des poids w_{ij} à chaque itération peuvent causer une grande variance et une convergence instable. Afin de minimiser le nombre des mises à jour, la méthode de descente de gradient stochastique à mini-lot est proposée (Nassima, 2020).

2.2.5.2 Momentum

Afin de contrôler l'oscillation de SGD, l'idée d'utiliser momentum est introduite. Inspirée par la première loi du mouvement de Newton, cette technique obtient une convergence plus rapide et une momentum appropriée qui peuvent améliorer les résultats d'optimisation de SGD. D'autre part, plusieurs techniques sont proposées pour déterminer le taux d'apprentissage approprié. Principalement, la décroissance du poids et la décroissance du taux d'apprentissage sont introduites pour ajuster le taux d'apprentissage et accélérer la convergence. Une décroissance de poids fonctionne comme un coefficient de pénalité dans la fonction de coût pour éviter le sur-ajustement (overfitting), et une décroissance du taux d'apprentissage peut réduire le taux d'apprentissage de manière dynamique pour améliorer les performances.

De plus, l'adaptation du taux d'apprentissage par rapport au gradient des étapes précédentes est utile pour éviter la fluctuation (Nassima, 2020).

2.2.5.3 Adagrad

En SGD, le taux d'apprentissage η est fixe et appliqué d'une façon équitable à tous les poids. En effet, les caractéristiques moins fréquentes nécessitent des mises à jour plus importantes. Pour répondre à cette problématique, la méthode Adagrad propose d'adapter le taux d'apprentissage η à chaque paramètre. Cela rend cette technique plus convenable aux bases d'apprentissage creuses en DL. L'équation (2.14) illustre la procédure de calcul des paramètres $\eta(t+1)$ (Nassima, 2020):

$$\begin{cases} \theta^{(t+1)} = \theta^{(t)} - \eta G_{(t)}^{-1} \nabla C(\theta^{(t)}) \\ G(t) = \begin{bmatrix} \sqrt{\sum_{\tau=1}^{t} \theta_1^{(\tau)^2} + \varepsilon} & \cdots & \cdots \\ \cdots & \sqrt{\sum_{\tau=1}^{t} \theta_i^{(\tau)^2} + \varepsilon} & \cdots \\ \cdots & \cdots & \sqrt{\sum_{\tau=1}^{t} \theta_n^{(\tau)^2} + \varepsilon} \end{bmatrix} \end{cases} \quad (2.\ 14)$$

2.2.5.4 AdaDelta

Dans la méthode Adagrad, le taux d'apprentissage $\eta G((t-)1)$ diminue d'une manière continue à cause de la somme accumulée q$\sum t\ \tau$=1 θ1 (τ)2 + e des itérations précédentes. Cela engendre une convergence très lente en raison des petites valeurs du taux d'apprentissage. La méthode AdaDelta est une version optimisée d'Adagrad et qui résout le problème de dégradation du taux d'apprentissage en considérant la moyenne mobile exponentiellement des gradients carrés. L'équation 2.15 illustre la procédure de calcul des paramètres $\theta(t$+1), où γ est la constante de décroissement exponentielle, η le taux d'apprentissage et $g_{ij(t)}$ est la racine carrée moyenne des gradients (Nassima, 2020):

$$\begin{cases} g_{ij}^{(t)} = \gamma g_{ij}^{(t-1)} + (1-\gamma)(C(\theta^{(t)}))^2 \\ \theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{\sqrt{g_{ij}^{(t)} + \varepsilon}} \nabla\, C(\theta^{(t)}) \end{cases} \quad (2.\ 15)$$

2.2.5.5 Adam

Adam appartient à la catégorie des méthodes qui proposent un taux d'apprentissage variant comme Adagrad et AdaDelta. Dans la mise à jour du taux d'apprentissage, cette méthode utilise la moyenne mobile exponentiellement des gradients carrés passés $v_t$et des gradients passés m_t. Les variables $v_t$et $m_t$présentent le premier et le deuxième moment du gradient équation (2.16) où $\beta_1$et β_2 sont les taux de décroissance. Pour éviter la convergence des moments$v_t$et $m_t$vers 0, les moments corrigés du biais m d $ij(t)$ et vc ij (t) sont utilisés équation (2.17). Enfin, les poids wij (t+1) sont mis à jour selon l'équation(2.18)(Nassima, 2020) :

$$\begin{cases} m_{ij}^{(t)} = \beta_1 m_{ij}^{(t-1)} + (1-\beta_1)(\frac{\partial C^{(t)}}{\partial w_{ij}}) \\ v_{ij}^{(t)} = \beta_2 v_{ij}^{(t-1)} + (1-\beta_2)(\frac{\partial C^{(t)}}{\partial w_{ij}})^2 \end{cases} \quad (2.\ 16)$$

$$\begin{cases} \widehat{m_{ij}^{(t)}} = \frac{m_{ij}^{(t)}}{(1-\beta_1^t)} \\ \widehat{v_{ij}^{(t)}} = \frac{v_{ij}^{(t)}}{(1-\beta_2^t)} \end{cases} \quad (2.\ 17)$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \frac{\eta}{\sqrt{\widehat{v_{ij}^{(t)}} + \varepsilon}} \widehat{m_{ij}^{(t)}} \quad (2.\ 18)$$

2.2.6 Les techniques de régularisation

En apprentissage automatique, un modèle efficace est caractérisé par une bonne performance sur les données d'apprentissage et de test. Tandis que la conception d'un bon modèle en généralisation est l'un des grands défis en apprentissage automatique à cause des problèmes de sous-apprentissage et sur-apprentissage. Le sous-apprentissage présente les catégories des modèles qui souffrent d'une faible performance sur les données d'apprentissage et test. En revanche, le sur-apprentissage est défini par les modèles performants sur les données d'apprentissage et mal adaptés aux données tests, et cela engendre une grande variance entre les deux et une mauvaise généralisation.

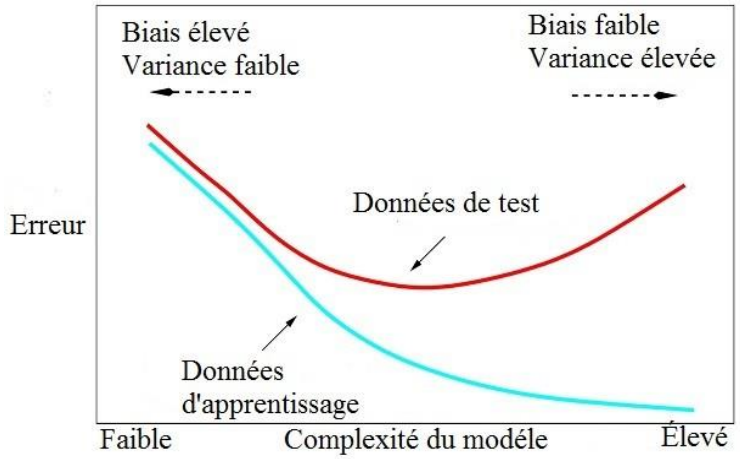


Figure 2. 10 : Modélisation du problème de sur-apprentissage.

En apprentissage profond, les réseaux sont caractérisés par une variance élevée et des biais faibles. La figure 2.10 illustre le problème de sur-apprentissage qui est lié principalement à la complexité du modèle, où les modèles complexes comme les DL ont plus de risque de sur-apprentissage. Afin de réduire ce risque, il existe deux stratégies : la stabilisation structurelle et la régularisation (al. & Bishopet, 1995.) La stabilisation structurelle contrôle la complexité du réseau en fonction de la variation du nombre des paramètres. La méthode de régularisation consiste à ajouter un terme qui pénalise la fonction du coût. Ce terme est utilisé pour pénaliser la fonction du coût en cas de paramètres à grande magnitude. Il existe plusieurs méthodes de régularisations :

2.2.6.1 Les régularisations L1 et L2

Les régularisations L1 et L2 sont parmi les techniques connues en régularisation. Ces méthodes mettent à jour la fonction de coût en ajoutant un terme de régularisation. Leur but est de diminuer les valeurs des poids par l'ajout du terme de régularisation afin de générer des modèles simples et d'éviter les problèmes de sur-apprentissage. Les équations 2.19 et 2.20 présentent les termes de régularisation L1 et L2 respectivement, où λ est le paramètre de régularisation, $\| \theta \|$ et $\| \theta \|^2$ sont les normes l1 et l2 du vecteur des paramètres θ. La technique

de régularisation L2 est aussi connue par la méthode de dégradation des poids (weight decay)(Nassima, 2020).

$$L_1 = \lambda \sum \| \theta \| \qquad (2.19)$$

$$L_2 = \lambda \sum \| \theta \|^2 \qquad (2.20)$$

2.2.6.2 La régularisation par abandon (Dropout)

La régularisation par abandon est une méthode de régularisation qui consiste à négliger l'étape de l'apprentissage au niveau d'un sous ensemble de neurones. Ces neurones sont sélectionnés aléatoirement selon un taux d'abandon et appartiennent aux couches d'entrée ou cachées. Cette technique permet de réduire le nombre important de liens entre les neurones et la spécialisation de certains au détriment d'autres, ce qui améliorera la généralisation et évitera le sur-apprentissage. Le processus d'abandon génère différentes architectures dans chaque itération et la performance totale est définie par la moyenne des performances de ces architectures. Cela permet de réduire la variance entre ces modèles et d'améliorer la généralisation (Nassima, 2020).

2.2.6.3 L'augmentation des données

Contrairement aux méthodes d'apprentissage automatique classique, les méthodes d'apprentissage profond exigent un grand volume de données afin d'éviter le problème de sur-apprentissage. La collecte d'une grande quantité de données et leur annotation présentent un défi dans certains domaines comme le domaine médical. Afin de résoudre ces limites, les méthodes d'augmentation de données sont proposées. Cette augmentation peut être effectuée soit avant l'étape de l'apprentissage (hors ligne) ou durant l'apprentissage sur les mini-lots (en ligne). Il existe plusieurs méthodes d'augmentation de données. Parmi les techniques les plus utilisées, nous avons : la rotation, la translation, l'écaillage, le bruit gaussien, division en patch, la normalisation et l'amélioration des couleurs, et la génération de nouvelles instances par les réseaux contradictoires génératifs (GAN)(Nassima, 2020).

2.2.6.4 L'arrêt prématuré

L'arrêt prématuré est une méthode de régularisation implicite (Chiyuan Zhang, 2016). Dans cette méthode, le jeu de données est divisé en base d'apprentissage et de validation. Ensuite, la performance du modèle est évaluée durant l'apprentissage sur les deux bases en se basant sur un indicateur (l'erreur). Généralement, l'apprentissage est arrêté lorsque la valeur de l'erreur sur la base de validation commence à s'incrémenter à cause du problème de sur-apprentissage (Bishop, 2006). Enfin, le modèle qui minimise le taux d'erreur est stocké pour la phase de prédiction

2.2.7 Les différentes Architectures du Deep Learning

Bien qu'il existe un grand nombre de variantes d'architectures profondes. Il n'est pas toujours possible de comparer les performances de toutes les architectures, car elles ne sont pas toutes évaluées sur les mêmes ensembles de données. Le Deep Learning est un domaine à croissance rapide, et de nouvelles architectures, variantes ou algorithmes apparaissent toutes les semaines.

2.2.7.1 Les Réseaux de Neurones Convolutifs

Convolutional Neural Network (CNN) (réseaux de neurones convolutifs) sont un type de réseau de neurones spécialisés pour le traitement de données ayant une topologie semblable a une grille. Qui se sont avères très efficaces dans des domaines tels que la reconnaissance et la classification d'images et vidéos. CNN a réussi a identifier les visages, les objets, panneaux de circulation et auto-conduite des voitures. Récemment, les CNN ont été efficaces dans plusieurs taches de traitement du langage naturel (telles que la classification des phrases).Dans le ML, un réseau convolutif est un type de réseau de neurones feed-forward, il a été inspire par des processus biologiques. Il existe quatre (4) principales opérations illustrées dans le CNN à savoir:

- La couche convolution
- La couche Rectified Linear Unit
- La couche Pooling
- La couche entièrement connectée

2.2.7.2 Réseau de Neurones Récurrents

L'idée derrière les RNN est d'utiliser des informations séquentielles. Dans un réseau neuronal traditionnel, nous supposons que toutes les entrées (et les sorties) sont indépendantes les unes des autres. Mais pour de nombreuses taches, c'est une très mauvaise idée. Si on veut prédire le prochain mot dans une phrase, il faut connaitre les mots qui sont venus avant. Les RNN sont appelés récurrents, car ils exécutent la même tache pour chaque élément d'une séquence, la sortie étant dépendante des calculs précédents. Une autre façon de penser les RNN est qu'ils ont une « mémoire » qui capture l'information sur ce qui a été calculé jusqu'ici. En théorie, les RNN peuvent utiliser des informations dans des séquences arbitrairement longues, mais dans la pratique, on les limite à regarder seulement quelques étapes en arrière. Il est utilisé pour:

- La modélisation du langage et génération de texte
- La traduction automatique
- La reconnaissance vocale
- Et la description des images

2.2.7.3 Modèle Génératif

Si les modèles discriminatifs comme (CNN, RNN) sont utilisés pour prédire les données du label et de l'entrée, tant dis que le modèle génératif décrit comment générer les données, il apprend et fait des prédictions en utilisant la loi de Bayes. Cependant les modèles génératifs sont capables de bien plus que la simple classification comme par exemple générer de nouvelles observations. Voici quelques exemples de modèle génératif :

- Boltzmann Machines
- Restricted Boltzmann Machines
- Deep Belief Networks
- Deep Boltzmann Machines
- Generative Adversarial Networks
- Generative Stochastic Networks

– Adversarial auto encoders

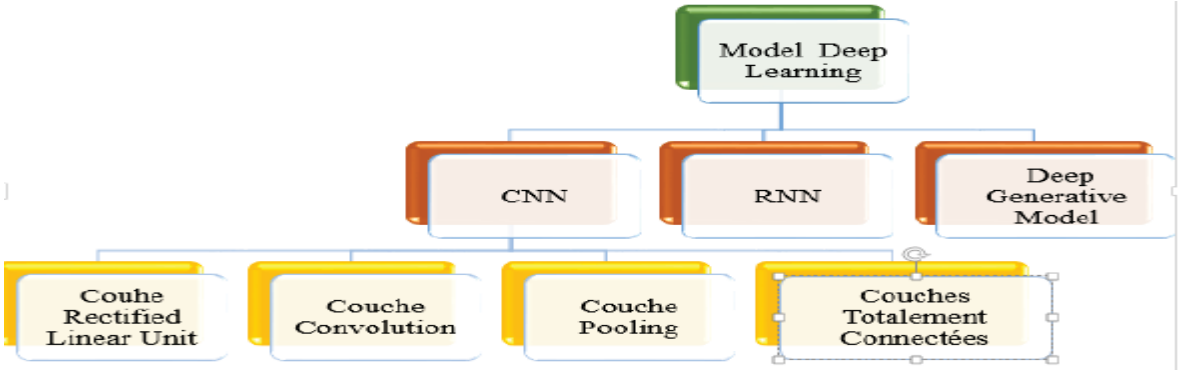


Figure2. 11 : Les different models du Deep Learning

2.2.8 Exemples d'Application de Deep Learning

Les applications du Deep Learning sont utilisées dans divers secteurs et divers domaines, tels que :

- Classification
- Reconnaissance faciale
- Véhicule autonome
- Recherche vocale
- Traduction automatique
- Deep fakes
- Prévision météo
- Reconstruction 3D
- Système multi-agents (jeux)
- Robotique

2.3 Les réseaux de neurone convolutif (CNN)

Les réseaux de neurone convolutif sont des réseaux d'apprentissage profond inspirés du cortex visuel (Wiesel, 1962.). Ces réseaux ont été utilisés dans les systèmes de recommandation (Rex Ying, 2018), en traitement du langage naturel (Kim, 2014), et en vision par ordinateur (Krizhevsky, Sutskever, & Hinton., 2012). Leur exploitation en vision par ordinateur a connu un grand succès grâce à leurs caractéristiques inspirées des systèmes visuels naturels. En vision par ordinateur, le processus de classification par les méthodes d'apprentissage classiques est basé sur deux étapes principales : l'extraction des caractéristiques et l'apprentissage. Ces caractéristiques sont considérées comme des handcrafted features à cause de l'effort manuel et nécessaire dans l'étude des attributs discriminants. Les méthodes utilisées extraient ces caractéristiques d'une manière non supervisée. Cette séparation entre les modules d'extraction et de classification peut nuire la tâche de classification si certains attributs discriminants ont été négligés dans la phase de l'extraction. Contrairement aux méthodes d'apprentissage classiques, les CNN réalisent implicitement le processus d'extraction des caractéristiques à travers les

couches de convolution, où les premières couches représentent les caractéristiques simples. Ensuite, ces caractéristiques sont combinées pour former d'autres qui sont plus complexes dans les couches profondes. Cette spécificité a rendu les CNN un bon outil pour la classification des données non structurées comme les images et les textes.

2.3.1 Les couches dans un CNN

Un réseau de neurone convolutif est composé de trois types de couches : couche de convolution, couche de pooling et couche entièrement connectée.

2.3.1.1 Couche de convolution

La couche de convolution est définie par le bloc de construction principal d'un CNN. Le but de cette couche est d'extraire implicitement les caractéristiques pertinentes des images en entrée durant l'apprentissage. Cette couche effectue une opération de convolution entre deux matrices, la première représente une sous partie des données en entrée (champ réceptif) et la deuxième représente un filtre qui contient les paramètres d'apprentissage. Une opération de convolution génère une troisième matrice référencée par la carte des caractéristiques. La figure 2.13 illustre une opération d'une convolution qui est réalisée par un produit scalaire entre le filtre et un champ réceptif. Ensuite, les résultats du produit sont additionnés pour produire un seul résultat présenté sous forme d'une case dans la carte des caractéristiques. Enfin, le filtre des poids est glissé par un pas S sur le reste des champs réceptifs de la matrice en entrée, et cette opération est répétée pour tous les autres champs.

Dans une convolution, la taille de la nouvelle carte des caractéristiques $N^{(t+1)}$ est calculée en fonction de 4 hyper-paramètres : la taille de l'ancienne carte des caractéristiques ou la matrice en entrée $N^{(t)}$, la taille du filtre F, la valeur du pas S et la valeur de la marge P.

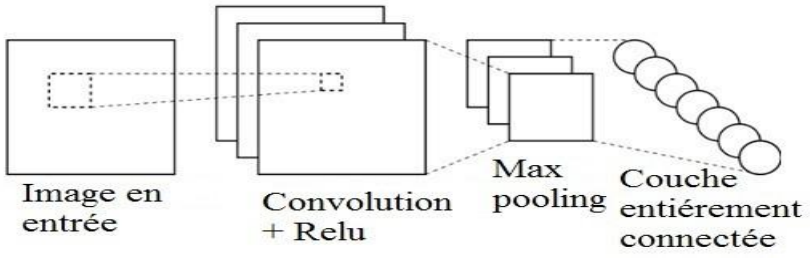


Figure 2. 12 : l'architecture générale d'un réseau de neurones convolutif.

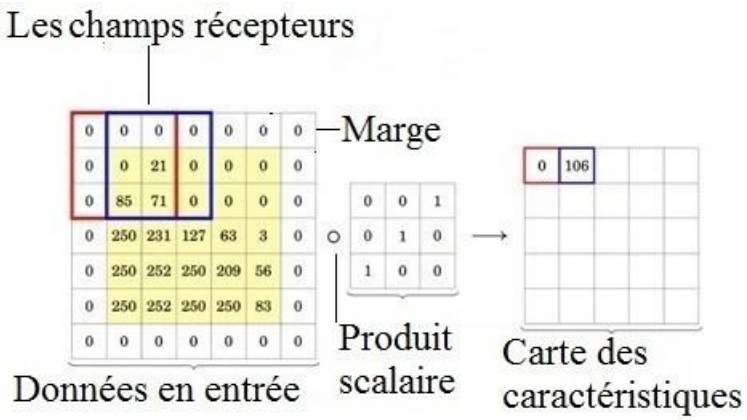


Figure 2. 13 : Une opération de convolution.

La marge représente des valeurs nulles qui entourent la matrice en entrée. Cette marge empêche le filtre de dépasser le cadre de cette matrice. L'application de $N_c$filtres sur les données en entrées résulte une carte des caractéristiques d'une taille de $N^{(t+1)}\times\ N^{(t+1)}\times N_c$, où $N_c$est sa profondeur. La concaténation des cartes des caractéristiques forme une couche de convolution.

$$N^{(t+1)} = \frac{N^t - F + 2P}{S} - 1 \qquad (2.21)$$

Le processus d'une convolution illustre la stratégie de réduction de dimensionnalité d'un CNN, où chaque case (neurone) de la carte de caractéristique courante est connectée seulement à un sous ensemble des neurones en entrées (champs récepteurs). En plus, l'application du même filtre sur toute la carte des caractéristiques lui permet de découvrir les attributs précédemment détectés dans différentes zones de l'image. À la fin de chaque opération de convolution, la fonction d'activation ReLu est appliquée sur la couche de convolution résultante afin d'améliorer la généralisation (Nassima, 2020).

2.3.1.2 Couche de pooling

Le rôle de la couche de pooling est de réduire la dimensionnalité des couches de convolution résultantes. Le but de cette réduction est d'améliorer la précision par la sélection des attributs dominants. En plus, l'optimisation du nombre des paramètres permet de réduire la taille du modèle et d'optimiser la complexité temporelle. La taille de la matrice résultante de l'opération de pooling est calculée par l'équation 2.21 avec $P = 0$. Il existe deux types d'opérations pooling : Max-pooling et Avg-pooling. L'opération de Max-pooling renvoie la valeur maximale du champ réceptif tandis que l'opération Avg-pooling renvoie la moyenne des valeurs. Max-pooling est la forme la plus utilisée dans la majorité des architectures de type CNN (Nassima, 2020).

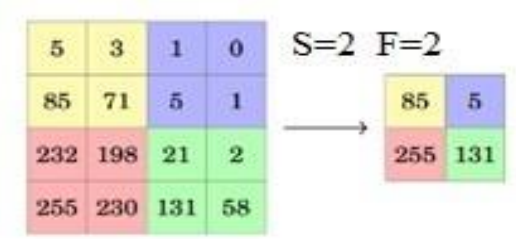


Figure 2. 14 : Une opération de Max-pooling.

2.3.1.3 Couche entièrement connectée

Dans un CNN, les couches entièrement connectées (FC) ont la même structure qu'un MLP. Le but de ces couches est d'apprendre les combinaisons non linéaires entre les caractéristiques extraites par les couches de convolution. Le résultat de la dernière couche de convolution [N, N, Nc] est aplatit dans un vecteur de taille [$N \times N \times Nc$]. Ce vecteur présente la couche d'entrée à l'ensemble des couches entièrement connectées. En classification supervisée, la dernière couche est utilisée pour la prédiction en se basant sur la fonction d'activation Softmax (Nassima, 2020).

2.3.1.4 Les Fonctions d'Activation

La fonction d'activation est une fonction mathématique appliquée a un signal en sortie d'un neurone artificiel. Le terme de "fonction d'activation" vient de l'équivalent biologique "potentiel d'activation", seuil de stimulation qui, une fois atteint entraine une réponse du neurone. La fonction d'activation est souvent une fonction non-linéaire. Leur but est de permettre aux réseaux de neurones d'apprendre des fonctions plus complexes qu'une simple régression linéaire car le fait de multiplier les poids d'une couche cachée est juste une transformation linéaire. Exemple de fonction d'activation Le ReLu (Rectified Linear Units) : Elle est utilisée après chaque opération de convolution, ou toutes les valeurs de pixels négatifs sont mises à zéro.

2.3.2 L'entrainement d'un réseau de neurone convolutionnels

L'entraînement d'un CNN consiste à déterminer et à calculer empiriquement la valeur de chacun de ses poids. Le principe est le suivant : le CNN traite une image (de la base de données d'entraînement) et en sortie il fait une prédiction, c'est-à-dire qu'il dit à quelle classe il pense que cette image appartient. Sachant qu'on connaît préalablement la classe de chacune des images d'entraînement, on peut vérifier si ce résultat est correct. En fonction de la véracité de ce résultat, on met à jour tous les poids du CNN selon un algorithme qui s'appelle la rétro propagation du gradient de l'erreur. Lors de la phase d'entraînement du modèle, le processus expliqué ci-dessus est répété plusieurs fois et avec la totalité des images de la base de données d'entraînement. Le but étant que le modèle classifie au mieux ces données. Lorsque le modèle a fini de mettre à jour ses poids, on évalue le modèle en lui présentant la base de données de validation.

Il classe toutes ces images (qui sont des images que le modèle n'a jamais vues) et on calcule son taux de bonne classification, c'est ce qu'on appelle la précision du modèle(les réseaux de neurones.).

2.3.3 Sur-apprentissage et sous-apprentissage

A la fin du processus d'apprentissage trois cas de figure peuvent se présenter :

- Le modèle est aussi performant sur les données d'entraînement (images sur lesquelles il s'entraîne) que sur les données de validation (images qu'il n'a jamais vues), cela signifie que le modèle a très bien fait son travail et qu'il reconnaît aussi bien les images qu'il connaît que celles qu'il n'a jamais vues.
- Le modèle reconnaît très bien les images d'entraînement et moins bien celles de validation. Le modèle aura une faible capacité prédictive, il n'arrive pas à généraliser. On parle alors de surentraînement. Dans ce cas-là on peut ajouter davantage d'images pour pallier ce problème.
- Le modèle ne reconnaît pas très bien les images d'entraînements et pas très bien non plus les images de validation. On parle alors de sous-apprentissage. Dans ce cas-là ajouter plus d'images ne servira à rien, c'est généralement le modèle choisie qui ne convient pas, il faudrait utiliser un modèle plus complexe (entrainement-dun-rseau-de-neurones-convolutif, 2018).

2.3.4 Indicateurs de performance d'un classifieur

- **Matrice de confusion**

  Pour mesurer les performances de ce classifieur, il est d'usage de distinguer 4 types d'éléments classés pour la classe voulue :

  1. Vrai positif VP, Elément de la classe 1 correctement prédit
  2. Vrai négatif VN, Elément de la classe 0 correctement prédit
  3. Faux positif FP, Elément de la classe 1 mal prédit
  4. Faux négatif FN, Elément de la classe 0 mal prédit

Ces informations peuvent être rassemblés et visualisés sous forme de tableau dans une matrice de confusion.

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

Figure 2. 15 : La matrice de confusion d'un classifieur.

- **Le graphe de précision**

Un autre indicateur important pour visualiser la performance d'un CNN est le graphe de précision, le graphe de précision est une représentation de la précision de modèle en fonction de nombre d'itération utilisé pour l'apprentissage et le test, ce dernier comporte deux courbe, une courbe pour les données d'entraiment et une autre pour les données de test. En particulier, si les deux courbes atteignent une précision = 100% la courbe, le classifieur est parfait.

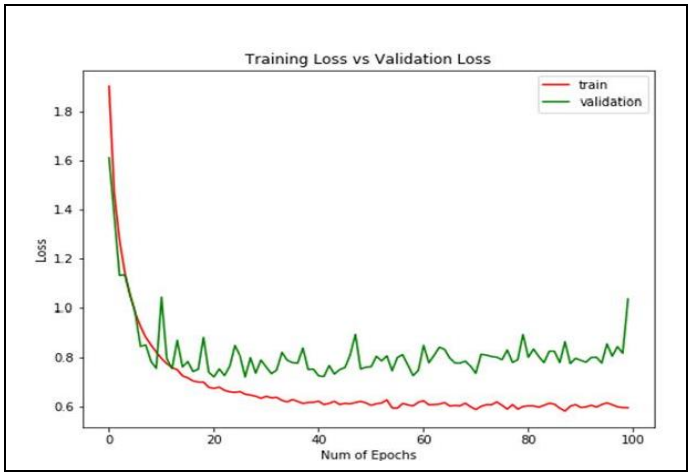


Figure 2. 16 : Le graphe de précision d'un réseau de neurone convolutifs.

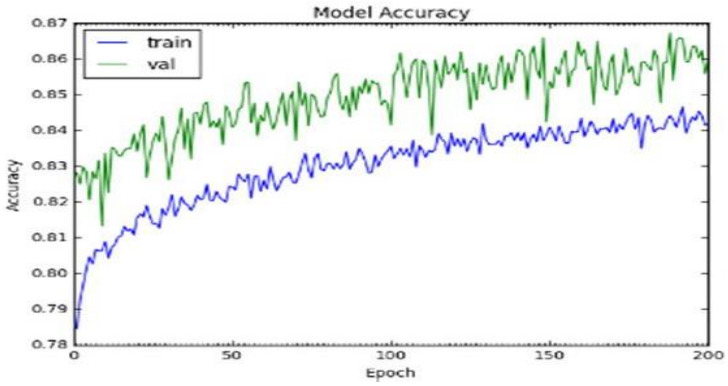


Figure 2. 17 : Le graphe d'erreur d'un réseau de neurone convolutif.

- **Le graphe d'erreur**

Le graphe d'erreur est une représentation de taux de dégradation d'erreur du modèle en fonction de nombré d'itération utilisé pour l'apprentissage et le test, ce dernier comporte deux courbe, une courbe pour les données d'entraiment et une autre pour les données de test. En particulier, si l'erreur de deux courbes diminue, le classifieur est parfait.

- **Indicateurs de base :**

Il est possible de calculer plusieurs indicateurs résumant la matrice de confusion. Par exemple si nous souhaitons rendre compte de la qualité de la prédiction sur laclasse1, on définit :

Précision : Proportion d'éléments bien classés pour une classe donnée:

$$Precision_{\text{de la classe 1}} = \frac{VP}{VP + FP}$$

Rappel : Proportion d'éléments bien classés par rapport au nombre d'éléments de la classe à prédire:

$$Rappel_{de\ la\ classe\ 1} = \frac{VP}{VP + FN}$$

F-mesure : Mesure de compromis entre précision et rappel:

$$F - mesure_{de\ la\ classe\ 1} = \frac{2 * (Précision * Rappel)}{Précision + Rappel}$$

2.3.5 Architectures existantes

Récemment, les réseaux de neurones convolutionnels ont été largement utilisés en vision par ordinateur grâce à leurs stratégies de réduction des paramètres et de la disponibilité de grandes quantités de données. De plus, le développement de la mémoire et de la puissance de calcul (GPU) a encouragé la communauté de la vision par ordinateur à proposer d'autres architectures plus profondes de type CNN. Ces architectures optimisent les couches convolutionnelle traditionnelles. L'objectif principal de ce changement est de réduire le nombre de paramètres et d'ajouter des couches supplémentaires pour améliorer la non-linéarité. Cette non-linéarité est assurée par des fonctions d'activation qui améliorent la capacité du réseau à résoudre des problèmes complexes.

LeNet : LeNet(Colin Lea, 2016) est la première architecture de type CNN proposée pour la classification supervisée. La figure 2.18 illustre la structure du réseau LeNet. Ce réseau est composé de 7couches au total: 3 couches de convolution (C_x), 2 couches d'Avg-pooling (S_x) et 2 couches entièrement connectées(F_x).La base d'apprentissage MNIST a été conçue pour la classification des chiffres manuscrits et contient 60 000 instances pour l'apprentissage et 10 000 instances pour le test. Ces instances sont des images en noir et blanc normalisées et centrées.

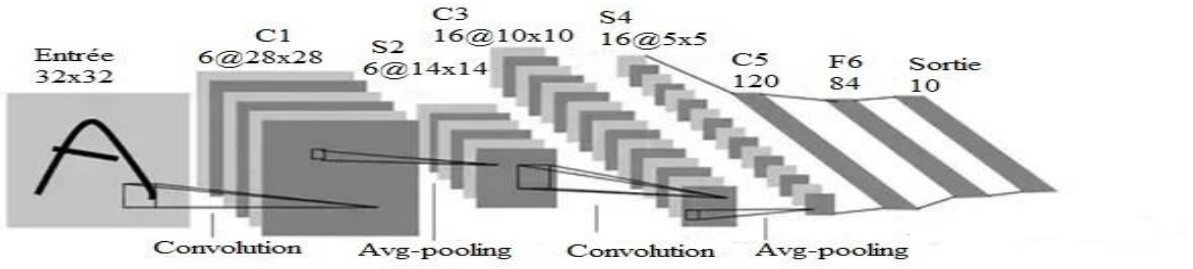


Figure 2. 18 : La structure du réseau LeNet [LeCunetal.1998].

AlexNet: En 2012, le réseau AlexNet (Krizhevsky, Sutskever, & Hinton., 2012) a été proposé dans la compétition ImageNet de reconnaissance visuelle à grande échelle (ILS-VRC). Cette compétition utilise un sous ensemble de la base d'apprentissage ImageNet composé de 1000

catégories, 1.2 million instances d'apprentissage, 50 000 instances de validation, et 150 000 instances de test. Dans cette compétition, le réseau AlexNet a atteint le meilleur taux d'erreur par rapport aux autres méthodes d'apprentissage automatique classiques. Ce résultat était un point important dans l'historique des réseaux d'apprentissage profond, car l'intérêt s'est attiré vers les méthodes DL plus que les méthodes ML classiques en vision par ordinateur. Le réseau Alex Net a une architecture similaire et plus profonde par rapport à LeNet (figure 2.18). Ce réseau est composé de 5 couches de convolution et 3 couches entièrement connectées. Selon la figure 2.19, la première couche de convolution utilise 96 filtres de taille 11 x11, tandis que les autres couches convolutives sont basées sur des filtres de taille 5x5 et 3x3. La première, la deuxième et la cinquième couche de convolution sont suivies par des couches de Max-pooling et les deux premières couches sont suivies par une opération de normalisation (local réponse normalisation(LRN)).La méthode LRN permet d'améliorer la généralisation et la non-linéarité du réseau. Cette opération est appliquée sur les résultats de la

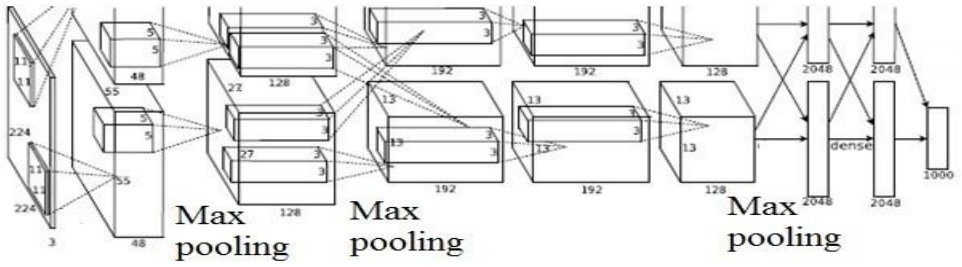


Figure 2. 19 : La structure du réseau AlexNet (Krizhevsky et al.2012).

fonction d'activation ReLu.

VGGNet : VGGN (Simonyan & Zisserman, 2014) est un réseau de neurones convolutif (figure 2.20) qui est basé sur les mêmes principes du réseau AlexNet (Krizhevsky, Sutskever, & Hinton., 2012). Le but de cette version est de proposer des configurations profondes (16 à 19 couches) en se basant sur la technique de stabilisation structurelle. Cette technique permet de contrôler le nombre des paramètres dans les réseaux profonds afin de diminuer les risques de sur-apprentissage. Pour réduire le nombre des paramètres, le réseau VGGNet propose de diminuer la taille des filtres de 7x7 et 5x5 à 3x3. Ce changement permet d'ajouter plus de couches intermédiaires sans risquer d'une augmentation exponentielle dans le nombre des paramètres. L'étude comparative entre le nombre des paramètres dans trois couches de convolution empilées associées à des filtres de taille 3x3 et une seule couche de convolution associée à un filtre de taille 7x7 a montré que les petits filtres réduisent le nombre des paramètres. En résumé, l'empilement des couches de convolution associées à des petits champs réceptifs permet de réduire le nombre des paramètres et d'améliorer la non-linéarité du réseau à travers les fonctions d'activation (ReLu) supplémentaires.

A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2. 20 : Les configurations du réseau VGGNET (Simonyan & Zisserman, 2014)

Inception : Le réseau Inception (Simonyan & Zisserman, 2014)est un réseau de neurone convolutif qui propose l'exploitation des modules d'Inception. Ces modules présentent des variantes optimisées des couches de convolution classiques. Les modules d'Inception introduisent des connexions partielles à l'intérieur d'une couche de convolution afin de réduire sa dimensionnalité. En résumé, les modules d'Inception augmentent la profondeur du réseau en contrôlant en parallèle la complexité de calcul par les techniques de réduction de dimensionnalité. En plus, la variation dans la taille des filtres permet de traiter l'information en entrée dans différentes échelles.

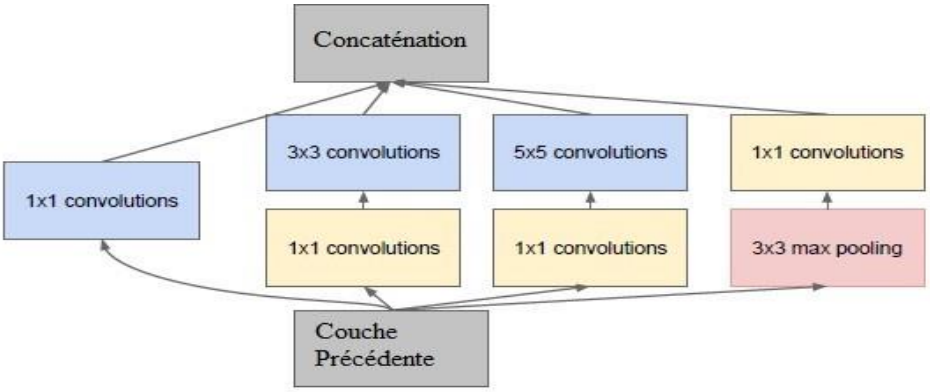


Figure 2. 21 : La structure d'un module d'Inception (Christian Szegedy, 2015).

ResNet : ResNet (Wei Hu, 2015) est un réseau de neurones convolutif basé sur des blocs résiduels. Le but principal de cette architecture est de résoudre le problème de dégradation de gradient. Ce problème apparait dans les réseaux très profonds, où la précision commence à être saturée ensuite dégradera pidement à cause de la diminution des valeurs des gradients. Afin de résoudre ce problème, les blocs résiduels ont été introduits. Les blocs résiduels (figure 2.22) représentent des connexions résiduelles entre la sortie de la couche précédente et la sortie de la couche courante. Ils sont utilisé deux réseaux inspirés de VGG Netet composés de 18 et 34 couches (figure 2.23) au total. L'étude comparative montre que dans la version classique des réseaux CNN, les réseaux moins profonds (18 couches) sont les plus performants. Tandis que dans les réseaux ResNet, les réseaux plus profonds sont les plus performants.

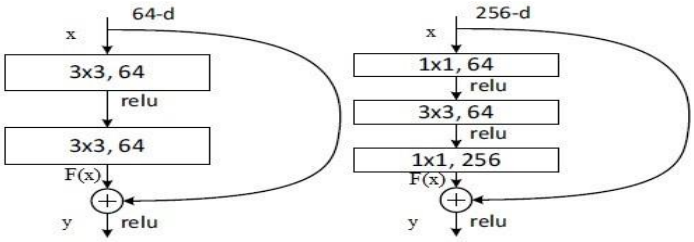


Figure 2. 22 : La structure d'un module d'Inception (Christian Szegedy, 2015).

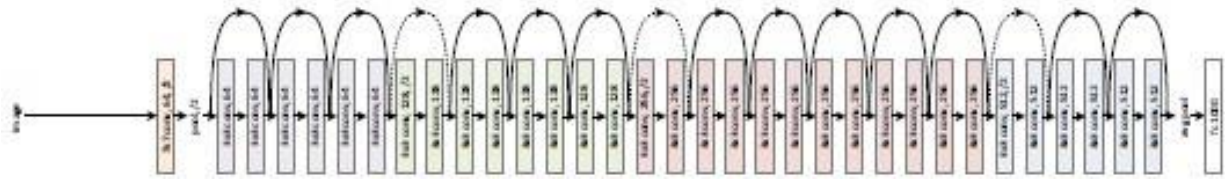


Figure 2. 23 : La structure du réseau ResNet (34couches)(Wei Hu, 2015)

Xception : Xception (Chollet, 2017) est un réseau de neurones convolutif basé sur les blocs extrêmes d'Inception. Ces blocs sont semblables aux convolutions séparables en profondeur (Depthwise séparable convolutions (DSC)) que nous détaillerons par la suite dans le réseau MobileNet. La figure 2.24 illustre la structure d'un bloc extrême d'Inception qui est caractérisé par des convolutions associées à des filtres de taille 1x1. Les convolutions s'associées à des

filtres de taille 3x3 sont appliquées par la suite sur des segments non superposés des cartes des caractéristiques résultantes. Contrairement à DSC, dans un bloc externe d'Inception, les convolutions associées à des filtres de taille 1x 1 sont appliquées en premier. En plus, ces blocs sont caractérisés par des fonctions d'activation ReLu. La figure 2.24 la structure du réseau Xception qui est représenté par une concaténation des blocs extrêmes d'Inception liés par des connexions résiduelles. Ce réseau est composé de 36 couches de convolution pour l'extraction des caractéristiques et une couche de régression logistique pour la classification.

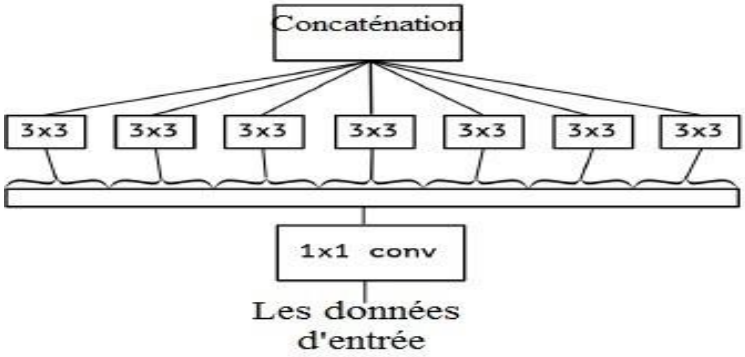


Figure 2. 24 : La structure d'un bloc extrême d'Inception (Chollet, 2017).

2.4 Conclusion

Dans ce chapitre, nous avons présenté les notions important qui sont en relation avec Deep Learning (DL) (définitions, historique etc.), nous avons aussi éclairci les réseaux de neurones artificielles, leurs différentes architectures, les domaines d'applications, et finalement nous avons introduit les notions sur les réseaux de neurones convolutifs(CNN) et leur structures, et ses différentes couches. Et à la fin nous avons présenté quelques exemples d'architectures, parmi eux Xception et VGGNet.

Chapitre 3 : Conception

3.1 Introduction

Une méthode de réseau de neurones convolutif basée sur l'apprentissage profond est utilisée pour la détection de spam image. La détection du spam d'image est un problème de classification binaire (spam et ham). Les réseaux neuronaux convolutifs sont utilisés pour les applications de traitement d'images car ils peuvent traiter efficacement les informations spatiales en capturant les informations liées aux pixels à l'aide de la convolution sur l'image. La principale force de l'utilisation d'architectures d'apprentissage profond est la capacité de reconnaître la signification des données lorsqu'elles sont dans d'énormes volumes et d'ajuster automatiquement la signification résultante avec de nouvelles données sans avoir besoin de l'information d'un expert du domaine. L'approche d'apprentissage profond peut donner une meilleure précision par rapport à l'apprentissage automatique et évite également la tâche d'extraction manuelle de caractéristiques en reconnaissant automatiquement les caractéristiques, ce qui réduit le temps et les efforts.

Dans ce chapitre nous décrivons en détaille la conception du modèle proposé en donnant les détails de chaque module de la conception.

3.2 Architecture générale du système

Nos recherches sur la détection de spam image nous permettent de constater que toutes les solutions proposées pour résoudre ce problème sont construites selon les mêmes architectures globales, fonctionnant en deux modules principaux indépendants :

- Le pre-processing et la normalisation
- La classification

Et ce sont les étapes qui constituent "système de detection de spam image" :

- Acquisition d'image
- Prétraiter et normaliser l'entrée
- Décider si c'est un spam ou non

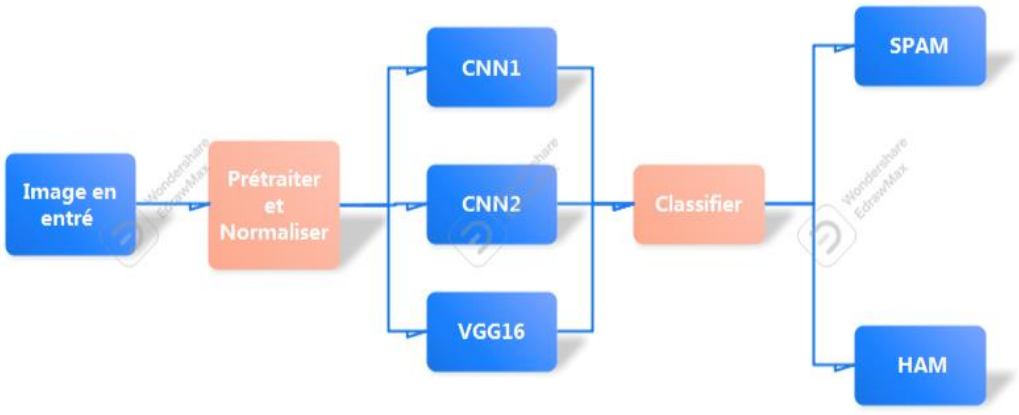


Figure 3. 1 : Architecture générale du système de detection de spam image.

3.2.1 Prétraitement (Pre-processing)

Chaque fois les données sont traitées avant d'être envoyé à l'algorithme d'apprentissage profond, le prétraitement est défini comme l'ensemble des transformations des données brutes avant que l'algorithme n'y accède. Il est probable que le regroupement de réseaux de neurones convolutif sur des images brutes entraîne de mauvaises performances de classification, Pour pallier à ces problèmes, il existe plusieurs méthodes de traitement et d'amélioration des images, telle que : la normalisation, l'égalisation de l'histogramme, etc.

L'ensemble de données utilisées dans ce travail contient de nombreux fichiers indésirables et des images corrompues qui ne peuvent pas être extraites. Il existe de nombreux formats d'images différents, près de 60% à 70% des images sont au format GIF. Ces images GIF ont été traitées pour extraire la première image puis enregistrées au format png. Cela permet d'augmenter dataset. Pour y parvenir, les étapes suivantes ont été suivies :

- Tous les formats indésirables tels que .txt ont été supprimés.
- Tous les fichiers corrompus ont été supprimés.

Enfin, toutes les images sont normalisées et redimensionnées. Les ensembles de données utilisés dans ce travail sont divisés en 70% pour les ensembles de formation et 30% de test.

3.2.2 Les modèles de réseaux neurones convolutifs profonds (CNN1, CNN2)

Les CNN désignent une sous-catégorie de réseaux de neurones et sont à ce jour un des modèles de classification d'images réputés être les plus performants. L'architecture d'un réseau de neurone convolutif (CNN) dispose en amont d'une partie convolutive et comporte par conséquent deux parties bien distinctes :

- **Une partie convolutive**

Son objectif final est d'extraire des caractéristiques propres à chaque image en les compressant de façon à réduire leur taille initiale. En résumé, l'image fournie en entrée passe à travers une succession de filtres, créant par la même occasion de nouvelles images appelées cartes de convolutions. Enfin, les cartes de convolutions obtenues sont concaténées dans un vecteur de caractéristiques appelé code CNN.

- **Une partie classification**

Le code CNN obtenu en sortie de la partie convolutive est fourni en entrée dans une deuxième partie, constituée de couches entièrement connectées appelées perceptron multicouche (MLP pour Multi Layers Perceptron). Le rôle de cette partie est de combiner les caractéristiques du code CNN afin de classer l'image.

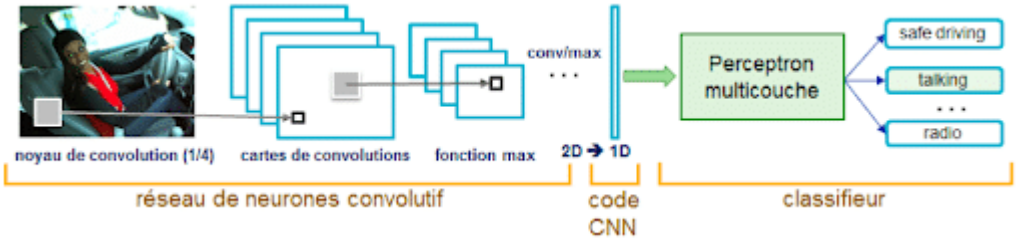


Figure 3. 2 : Schéma représentant l'architecture d'un CNN.

Ce projet traite de deux architectures CNN que nous avons créés et nous les avons nommées CNN1 et CNN2. Nous avons aussi utilisé le modèle VGG16. Les trois modèles ont été entraînés pour 50 itérations avec une taille de lot de 64 images. L'ensemble de données qui contient 1585 images spam et 659 images ham (images légitimes) pour l'apprentissage et 377 images spam et 151 images ham (images légitimes) pour le test.

Dans ce qui suit nous détaillerons ces différents modèles.

Modèle CNN1 : L'architecture CNN1 telle que définie ci-dessous a été utilisée pour obtenir des résultats. Les images ont d'abord été redimensionnées à 224*224*3, puis introduites dans le réseau. CNN1 possède 5 couches de convolution de taille de filtre 32, 64, 128 et 256. Chaque couche convolutionnelle est immédiatement suivie d'activation ReLU et d'une couche de max pooling de taille 2. Après les couches de convolution, une régularisation de type dropout est utilisée. La sortie est aplatie et transmise à une couche Dense qui contient 128 neurones. Cette couche est suivie d'une activation ReLU et d'une régularisation de type dropout. Enfin, une couche dense d'un seul neurone est utilisée avec une fonction d'activation sigmoid.

Figure 3. 3 : Architecture du modèle CNN1

Layer (type)	**Output Shape**	**#Param**
Conv2D	(None, 224, 224, 32)	896
Conv2D	(None, 224, 224, 32)	9248
MaxPooling 2D	(None, 112, 112, 32)	0
Dropout	(None, 112, 112, 32)	0
Conv2D	(None, 112, 112, 64)	18496
MaxPooling2D	(None, 56, 56, 64)	0
Dropout	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 128)	73856
MaxPooling 2D	(None, 28, 28, 128)	0
Dropout	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 256)	295168
MaxPooling 2D	(None, 14, 14, 256)	0
Dropout	(None, 14, 14, 256)	0
Flatten	(None, 50176)	0
Dense	(None, 128)	6422656
Dropout	(None, 128)	0
Dense	(None, 1)	129
Total params: 6,820,449		

Tableau 3. 1 : configuration du modèle CNN1.

Modèle CNN2 : Le modèle CNN telle que définie ci-dessous a été utilisée pour obtenir des résultats. Les images ont d'abord été redimensionnées à 224 * 224 * 3, puis introduites dans le réseau. CNN3 possède également 3 couches de convolution de taille de filtre 32,64 et 128. Chaque couche de convolution est immédiatement suivie d'activation ReLU et d'un max pooling de taille 2. Après les couches de convolution, et la sortie est aplanie et transmis à une couche dense qui contient 128 neurones. Cette couche est suivie d'une activation ReLU et d'une régularisation Dropout. Enfin, une couche dense d'un seul neurone est utilisée avec une fonction d'activation sigmoid.

Figure 3. 4 : configuration du modèle CNN2.

Layer (type)	Output Shape	Param #
Conv2D	(None, 224, 224, 32)	896
MaxPooling2D	(None, 112, 112, 32)	0
Conv2D	(None, 112, 112, 64)	18496
MaxPooling 2D	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 128)	73856
MaxPooling2D)	(None, 28, 28, 128)	0
Flatten	(None, 100352)	0
Dropout	(None, 128)	0
Dense	(None, 1)	129
Total params: 12,938,561		

Tableau 3. 2: configuration du modèle CNN2.

Modèle VGG16 : VGG16 est un modèle de réseau neuronal convolutionnel proposé par K. Simonyan et A.Zisserman de l'Université d'Oxford. Le modèle atteint une précision de 92,7% sur la base de données ImageNet qui contient plus de 14 millions d'images appartenant à 1000 classes. Le modèle VGG16 telle que définie ci-dessus a été utilisée pour obtenir de meilleurs résultats. Les images ont d'abord été redimensionnées à 224 * 224 * 3, puis introduites dans le réseau. VGG16 possède également 13 couches de convolution (5 blocks) de taille de filtre block1 de 64, block2 de 128, block3 de 256, block4 et block5 de 512. Chaque couche de convolution est immédiatement suivie d'activation ReLU et d'un max pooling de taille 2. Après les couches de convolution, et la sortie est aplanie et transmis à deux couches dense qui contient 4096 neurones chaqu'un. Cette couche est suivie d'une activation ReLU. Enfin, une couche dense d'un seul neurone est utilisée avec une fonction d'activation sigmoid.

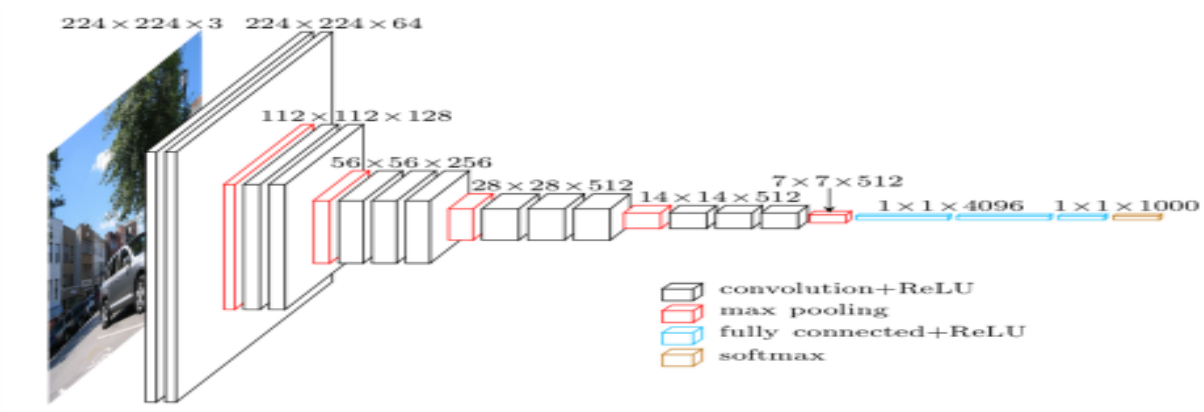


Figure 3. 5 : architecture du CNN VGG

Layer (type)	Output Shape	Param #
Input Layer	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3211392
dense_1 (Dense)	(None, 1)	129
Total params: 17,926,209		

Tableau 3. 3 : Configuration du CNN VGG16

3.3 Apprentissage par transfert (Transfer Learning)

Il existe des modèles pré-entraînés qui sont mis à disposition par certaines communautés. Ces modèles sont entraînés sur des milliards d'images comme la base de données ImageNet. L'apprentissage par transfert est utilisé pour réduire le temps de calcul nécessaire à l'entraînement de votre propre réseau et pour vous permettre d'utiliser ces réseaux pré-entraînés. Nous pouvons geler certaines couches en fonction de nos besoins et n'entraîner qu'un sous-ensemble de ces couches sur notre propre dataset. Il existe deux modèles pré-entraînés de ce type disponibles en open source : VGG16 et VGG19. Cependant, ce projet ne traite que du modèle VGG16.

3.4 Conclusion

Dans ce chapitre nous avons détaillé l'architecture générale de notre système de détection de spam image, puis nous avons introduit les trois modèles CNN utilisés par notre système de détection des spam image (architecture de chaque modèle).

Chapitre 4 : Implémentation et expérimentation

4.1 Introduction

L'objectif de ce chapitre est de présenter les étapes de l'implémentation de l'approche proposée dans le cadre d'un système de détection de spam image et les différentes étapes de son réalisation. Nous nous sommes intéresses a l'utilisation de réseau neuronal convolutif. On va appliquer les trois modèles créé sur les bases d'images choisies. Et pour cela, on va travailler avec les bibliothèques Tensorflow et Keras pour l'apprentissage et la classification et afin d'améliorer les performances des modèles on va utiliser quelques techniques simple et efficaces comme data augmentation et dropout.
Nous commençons tout d'abord par la présentation des ressources, du langage et de l'environnement de développement que nous avons utilisé. Puis les étapes de la réalisation du modèle.

Nous poursuivons ce chapitre par la présentation des différents résultats expérimentaux obtenus et discussion, quelques captures d'écrans de notre application.

4.2 Outils d'implémentation

Pour implémenter et optimiser notre model, nous avons maintenant des Frameworks de DL open source faciles à utiliser qui a l'objectif de Simplifier la mise en œuvre des modèles complexes et à grande échelle.

- **Python :** Le langage de programmation scientifique multi-paradigme Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991. Ce langage de programmation présente de nombreuses caractéristiques intéressantes :

- Il est gratuit et multiplateforme. : Windows, Mac OS X, Linux, Android, iso, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.
- C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.
- C'est un langage dynamique, extensible Il favorise la programmation structurée fonctionnelle et orientée objet. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d'interactions. Il est relativement simple à prendre en main (Poulain, 2019). La syntaxe de Python est très simple et, combinée à des types de données évolués (listes, dictionnaires,...), conduit à des programmes à la fois très compacts et très lisibles. Il gère ses ressources (mémoire, descripteurs de fichiers...) sans intervention du programmeur (python).
- Enfin, il est très utilisé en bioinformatique et l'intelligence artificielle et plus généralement en analyse de données. Toutes ces caractéristiques font que Python est outils idéal pour implémenter notre application.

- **TensorFlow :** TensorFlow est un framework de programmation pour le calcul numérique qui a été rend Open Source par Google en Novembre 2015. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multidimensionnelles, appelées Tenseurs (Tensor). Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix (Zakaria, 2017).

- **Keras :** Keras est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow ou Theano. Il a été développé en mettant l'accent sur l'expérimentation rapide. Être capable d'aller de l'idée à un résultat avec le moins de délai possible est la clé pour faire de bonnes recherches. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un ingénieur Google. En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de backend.

- **OpenCV :** OpenCV (Open Source Computer Vision Library) est une bibliothèque qui aide à la vision par ordinateur Depuis son lancement officiel en 1999 par l'équipe d'Intel, un certain nombre de programmeurs ont contribué aux derniers développements de la bibliothèque. Le dernier changement majeur intervenu en 2009 (OpenCV 2) qui inclut les principales modifications apportées à l'interface C. OpenCv est spécialisé en :
  - Manipulation des données d'images et vidéo, les matrices et les vecteurs.
  - Différentes structures de données dynamiques (listes, files d'attente, ensembles, arbres, graphiques).
  - Analyse du mouvement (flux optique, segmentation du mouvement, suivi), Reconnaissance d'objets.
  - Interface graphique de base (image / vidéo à afficher, gestion du clavier et de la souris, barres de défilement…)(Agam, 2006).

- **NumPy :** NumPy est le paquet fondamental du calcul scientifique avec Python. Il contient entre autres des choses :
  - un puissant objet tableau à N dimensions
  - fonctions sophistiquées (diffusion)
  - des outils pour l'intégration du code C / C ++ et Fortran
  - capacités utiles d'algèbre linéaire, de transformée de Fourier et de nombres aléatoires

- **Matplotlib :** Est une bibliothèque de traçage pour le langage de programmation Python et son extension mathématique numérique NumPy. Il fournit une API orientée objet permettant d'incorporer des graphiques dans des applications à l'aide de kits d'outils d'interface graphique à usage général tels que Tkinter, wxPython, Qt ou GTK +.

- **Scikit-learn :** Scikit-learn est une bibliothèque libre Python dédiée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support.

4.3 Environnement d'implémentation

- Processeur : Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz 1.70 GHz
- RAM : 4,00 Go
- Système d'exploitation : Windows 10, 64 bits

4.4 Ensemble de données

Ce projet utilise la combinaison de trois ensembles de données différents :

- **Dredze Image Spam Dataset :**(Dredze et al. 2007) ont créé un ensemble de données contient 3 ensembles d'images. Ham personnel (PHam) contient 2 021 images dont 1 517 sont uniques. Personal Spam (PSpam) contient 3 298 images, dont 1 274 sont uniques. Enfin, les archives de spam (SpamArch) contiennent 16 028 fichiers de différents formats (JPEG, PNG, GIF, etc.), dont 3 039 images uniques. En outre, l'archive de spam contenait beaucoup de fichiers non traités dans différents formats tels que gif, txt, jpg, etc

- **Image Spam Hunter (ISH)** (Gao et al. 2008) : Ce dataset contient à la fois des images spam et ham au format JPEG qui sont collectées à partir des e-mails originaux. Il y a 810 images ham et 929 images spam au total. Le nombre d'images uniques de spam et de ham est de 879 et 810 respectivement.

- **Improved Dataset** (Annadatha, 2019) : Ce dataset a été créé par Aneri et al, en effectuant une transformation sur les images ham pour les faire ressembler à des spam. Les images ham ont été redimensionnées à la taille des images spam pour aligner leurs caractéristiques de métadonnées. Du bruit a été introduite dans les images spam pour rendre difficile la détection des bords, puisque les images spam ont généralement moins de bruit que les images ham. Le nombre d'images uniques dans ce jeu de données est de 975. Il est disponible à l'adresse.

Nous avons également prétraité et combiné ces ensembles de données pour en construire un seul ensemble de données qui contient 1585 images spam et 659 image ham pour l'apprentissage et 377 images spam et 151 images ham pour le test. Ensuite, les expériences ont été réalisées sur cet ensemble de données.

4.5 Implémentation et Réalisation :

Cette section décrit les différents modules du notre système de détection de spam image.

- Prétraitement : Les images données sont divisées en deux ensembles différents qui sont l'ensemble d'apprentissage et ensembles de test qui sont ensuite redimensionner au format 224/224/3 puis normalisé.
- Construire les modèles CNN (CNN1, CNN2, VGG) :

```
# ------------------------------------------------------------------------
 create model structure "ModelCNN2.h5"


def CNN3(input_shape, num_classes):
    model = Sequential()

    model.add(Conv2D(32, (3, 3), activation='relu',
              padding='same', input_shape=input_shape))
    model.add(MaxPooling2D((2, 2)))

    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))

    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))

    model.add(Flatten())

    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))

    model.add(Dense(num_classes, activation='sigmoid'))

    return model
```

Figure 4. 1 : un aperçu sur le code source pour l'architecture CNN2.

- Formation des modèles :

```
model.compile(loss='binary_crossentropy', optimizer=Adam(
    learning_rate=0.0001, decay=1e-6), metrics=['accuracy'])
model.summary()
# Train the neural network/model
checkpoint = ModelCheckpoint(
    "best_weights.h5", monitor='val_acc', verbose=1, save_best_only=True, mode='max')

early_stopping = EarlyStopping(monitor='val_loss',
                               min_delta=0,
                               patience=3,
                               verbose=1,
                               restore_best_weights=True
                               )

reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                                        factor=0.2,
                                        patience=3,
                                        verbose=1,
                                        min_delta=0.0001)

callbacks_list = [early_stopping, checkpoint, reduce_learningrate]
#callbacks_list = [checkpoint]

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.n//train_generator.batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_generator.n//validation_generator.batch_size,
    callbacks=callbacks_list
)
```

Figure 4. 2 : un aperçu sur le code source pour l'entrainement des modèles.

- Tester les modèles

```
def classifierImage(img_path):
    # loading models
    spam_classifier = load_model(spam_model_path, compile=False)
    # spam_classifier.summary()
    img_width = 224
    img_height = 224
    img = load_img(img_path, grayscale=False, target_size=(img_width, img_height))
    orig_frame = cv2.imread(img_path)
    # convert to numpy array
    img_array = np.array([img_to_array(img)/255.])
    # image classification on processed image
    pred = spam_classifier.predict(img_array)
    pred = pred[0][0]
    result = 'SPAM' if pred > 0.5 else 'HAM'
    cpred = pred if pred > 0.5 else 1-pred
    orig_frame = cv2.resize(orig_frame, (400,400))
    label=result+"  ("+str(round(100*cpred,2))+" %)"
    # Afficher resultat
    font = cv2.FONT_HERSHEY_SIMPLEX
    textsize = cv2.getTextSize(result, font, 3, 3)[0]
    textX = round((orig_frame.shape[1] - textsize[0])/2 )
    textY = round((orig_frame.shape[0] + textsize[1])/2)
    cv2.putText(img=orig_frame, text=result, org=(textX, textY), fontFace=font, fontScale=3, color=(0, 0, 255),thickness=
    cv2.imshow('Spam detection : '+label, orig_frame)
    if (cv2.waitKey(0) & 0xFF == ord('q')):
        sys.exit("Thanks")

    cv2.destroyAllWindows()
```


Figure 4. 3 : un aperçu sur le code source pour le teste des modèles.

4.6 Résultats obtenus et discussions

Les résultats sont présentés et commentés et une synthèse est faite à la fin.

4.6.1 Matrices de confusion

Pour montrer les résultats obtenus par les trois modèles, nous illustrons ci-dessous les résultats des matrices de confusion pour chacun des trois modèles.

- **Le premier modèle (CNN1)**

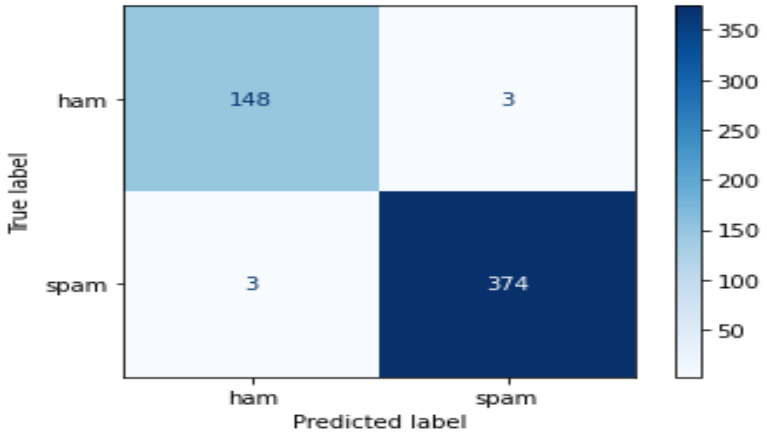


Figure 4.4 : la matrice de confusion du premier modèle.

La matrice de confusion de la figure 4.4 résume les prédictions des images de test sur les deux classes. Pour les ham : 148 images classées dans ham, 3 images classées dans spam.
Pour spam : 374 image classées dans spam, 3 images classées dans ham.

Nous remarquons que la totalité des images sont bien classées ce qui donne une précision de 98.86 %.

- **Le deuxième modèle (CNN2)**

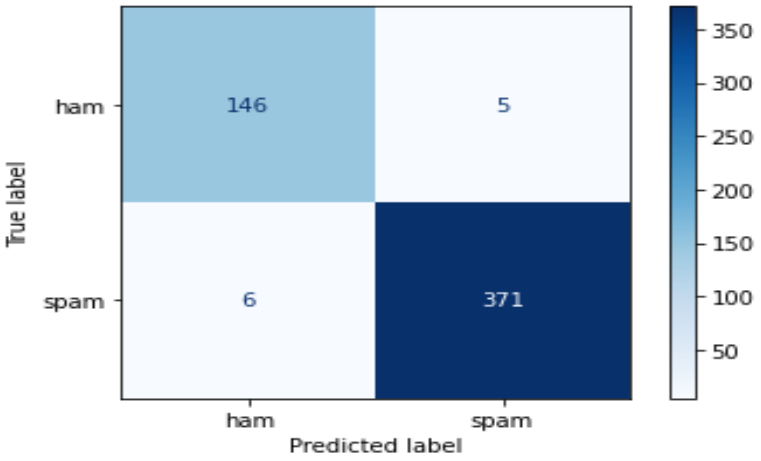


Figure 4. 5 : la matrice de confusion du deuxième modèle.

La matrice de confusion de la figure 4.5 résume les prédictions des images de test sur les deux classes.
Pour les ham : 146 images classées dans ham, 5 images classées dans spam.
Pour spam : 371 image classées dans spam, 6 images classées dans ham.
Nous remarquons que presque la totalité des images sont bien classées ce qui donne une précision de 97.92%.

- **Le troisième modèle (VGG16)**

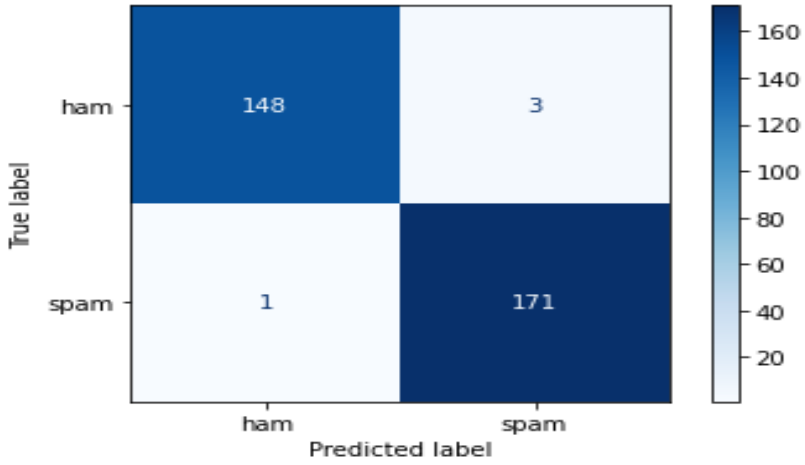


Figure 4. 6 : la matrice de confusion du troisième modèle.

La matrice de confusion de la figure 4.6 résume les prédictions des images de test sur les deux classes.

Pour les ham : 148 images classées dans ham, 3 images classées dans spam.
Pour spam : 371 image classées dans spam, 1 images classées dans ham.
Nous remarquons que presque la totalité des images sont bien classées ce qui donne une précision de 98.76%.

4.6.2 Expérimentations et discutions

Du point de vue résultats, nous obtenons après l'apprentissage du premier modèle (CNN1) une précision de test de 98.86%, pour le deuxième modèle (CNN2) un taux de précision estimé à 97.92% est observé et enfin pour le troisième modèle (VGG16) un taux de précision estimé à presque 98.76% est observé.

- **Résultats obtenus pour lemodèle1 (CNN1)**

La figure 4.5 évalue la précision atteinte pour le premier modèle (CNN1). D'après la figure la précision de l'apprentissage et de test augmente avec le nombre d'epochs ceci reflète qu'à chaque epoch le modèle apprenne plus d'informations, si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'epochs. On constate que le pic de la précision atteint a été de 99% pour la formation et de 98 % pour la validation.

La figure 4.7 évalue la perte observée. De même l'erreur d'apprentissage et de la validation diminue avec nombre d'epochs. On remarque que la perte pendant la formation est inférieure à celle au cours de la validation. La figure 4.8 évalue la perte observée. On remarque que la perte pendant la formation est presque égale à celle au cours de la validation.

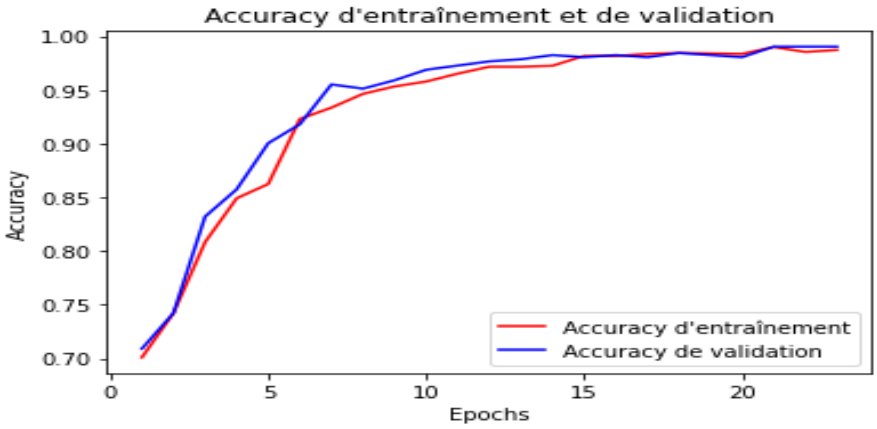


Figure 4. 7 : la courbe de précision du premier modèle (CNN1).

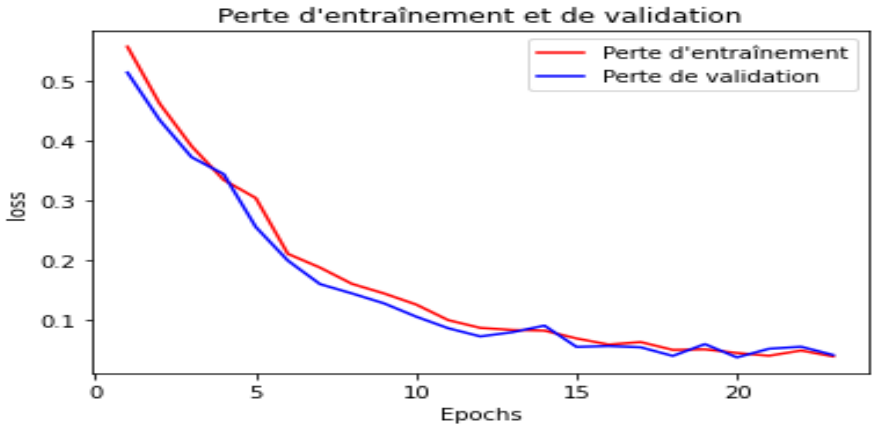


Figure 4. 8 : la courbe d'erreur du premier modèle (CNN1).

Le tableau 4.1 résume le rapport de classification du modèle CNN1.

	Précision	rappel	F1-mesure	support
Spam	0.98	0.98	0.98	377
Ham	0.99	0.99	0.99	151

Tableau 4. 1 : rapport de classification du premier modèle.

- **Résultats obtenus pour le modèle2 (CNN2)**

La figure 4.9 évalue la précision atteinte pour le premier modèle (CNN2).D'après la figure on constate que le pic de la précision atteint a été de 99% pour la formation et de 99 % pour la validation. La figure 4.10 évalue la perte observée. On remarque que la perte pendant la formation est presque égale à celle au cours de la validation.

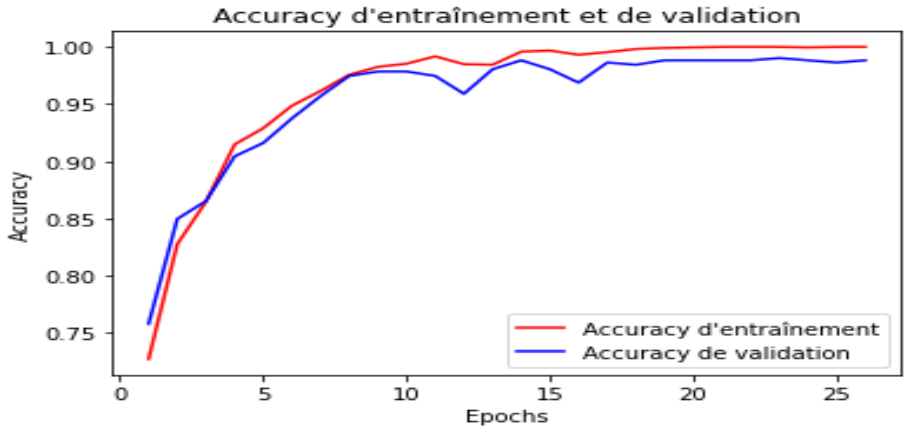


Figure 4. 9 : la courbe de précision du deuxième modèle (CNN2).

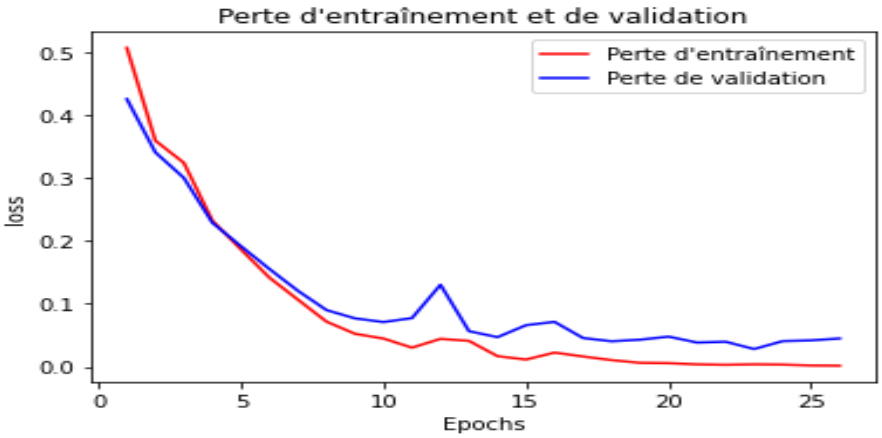


Figure 4. 10 : la courbe de précision du deuxième modèle (CNN2).

Le tableau 4.2 résume le rapport de classification du modèle CNN.

	précision	rappel	F1-mesure	support
Spam	0.99	0.98	0.99	377
Ham	0.96	0.97	0.96	151

Tableau 4. 2 : rapport de classification du deuxième modèle (CNN2).

- **Résultats obtenus pour le modèle3 (VGG16)**

La figure 4.11 évalue la précision atteinte pour le troisième modèle (VGG).D'après la figure on constate que le pic de la précision atteint a été de 100% pour la formation et de et pour la validation. La figure 4.12 évalue la perte observée. On remarque que la perte pendant la formation est presque égale à celle au cours de la validation.

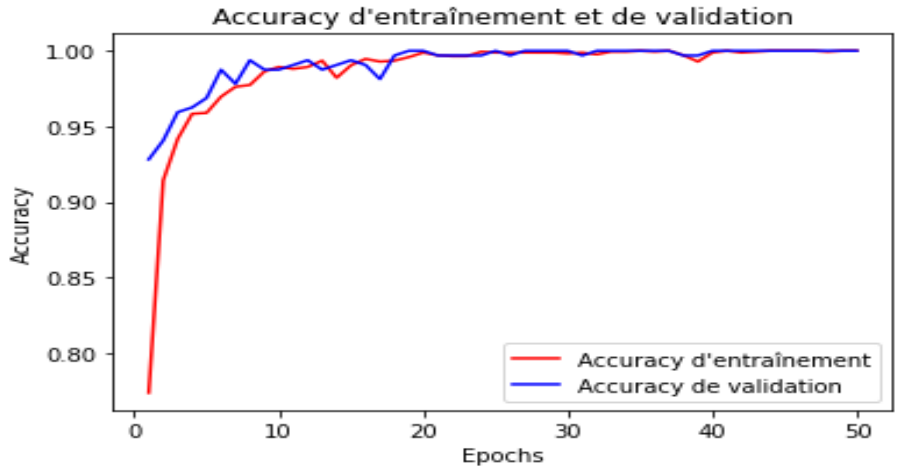


Figure 4. 11 : la courbe de précision du troisième modèle.

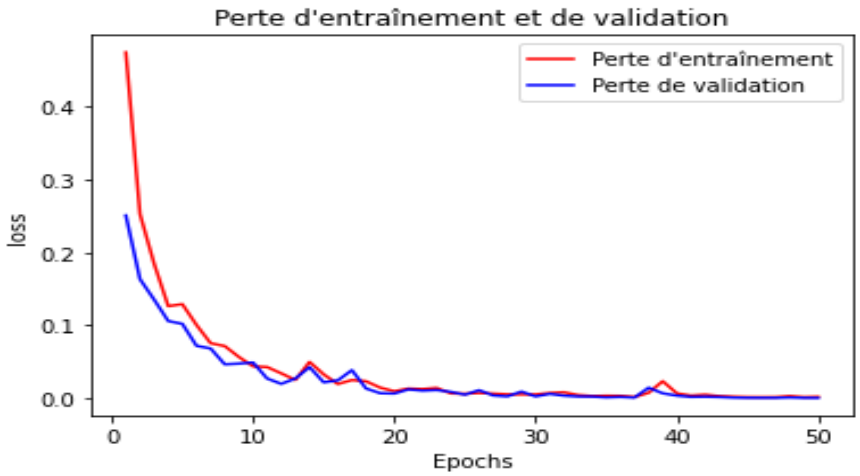


Figure 4. 12 : la courbe d'erreur du troisième modèle.

Le tableau 4.3 résume le rapport de classification du modèle CNN2.

	précision	Rappel	F1-mesure	support
Spam	0.98	0.99	0.99	377
Ham	0.99	0.99	0.99	151

Tableau 4. 3 : rapport de classification du troisième modèle (VGG).

Le tableau 4.4 résume les résultats obtenus pour les trois modèles crées. Chaque modèle est représenté par son architecture ainsi que le nombre d'itération utilisée et la précision lors de l'entrainement et de test. D'après les résultats décrits dans ce tableau, nous constatons que les résultats obtenus par le premier est meilleur que les résultats obtenus par le deuxième et le troisième.

D'une manière générale, le nombre de couche de convolution et le nombre d'itérations sont des facteurs primordiaux pour améliorer la performance d'un réseau de neurone convolutif en termes de précision. Les techniques simples et efficaces comme data augmentation et dropout peuvent améliorer d'avantage les performances des modèles.

Modèle #param	**Architecture utilisé**				**Nombre époque**	**Précision obtenu sur la base de apprentissage**	**Précision obtenu sur la base de validation**
	Couche de Conv.	**Couche de Pooling**	**Dropout**	**Fully connected**			
CNN1 6,820,449	**05**	**04**	**05**	**02**	**50**	**99.00**	**98.86**
CNN2 12,938,561	**03**	**03**	**00**	**02**	**50**	**99.00**	**97.91**
VGG16 17,926,209	**13**	**05**	**00**	**02**	**50**	**99.00**	**98.76**

Tableau 4. 4 : Tableau de comparaison de résultat.

D'après les résultats de la figure 4.13, nous pouvons conclure que notre approche les CNNs comme classifier surpasse clairement les classificateurs classiques d'apprentissage automatiques tels que SVM en termes de précision.

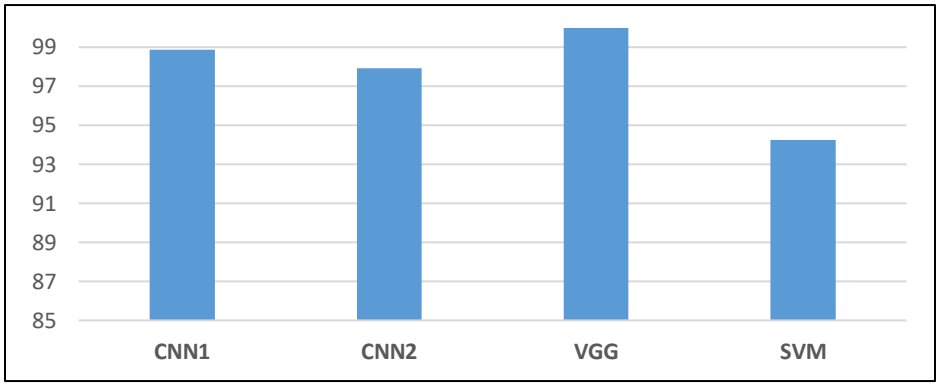


Figure 4. 13 : comparaison entre les résultats des modèles CNN et SVM.

4.7 L'interface de l'application

La figure 4.14 ci-dessous montre l'interface principale de notre application de détection de spams image où nous testons nos modèles. Le système nous permet de choisir l'image à classer puis le modèle CNN à utiliser pour la classification. Une fois le choix est fais le système nous affiche une fenêtre qui contient l'mage entrée et le résultat de la classification (Spam ou Ham) avec la probabilité du résultat obtenu (Voir figure 4.15).

Figure 4. 14 : interface principale du système de détection de spam image.

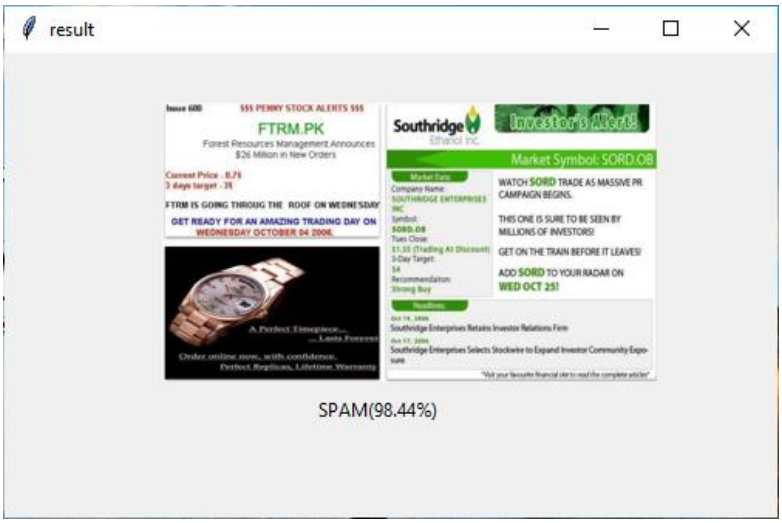


Figure 4. 15 : Un exemple de résultat de la classification.

4.8 Conclusion

Dans ce chapitre, nous avons présenté notre application de détection de spam image basée sur les réseaux de neurones de types CNN, nous avons utilisé trois architectures différentes la première et la deuxième proposés par nous-mêmes et le troisième est inspirée de l'architecture VGG16, nous avons présenté aussi les différents résultats obtenus pour chaque architecture. Notre système a été testé sur la base de données réelle. Pour conclure, les expérimentations ont montré que le modèle de VGG est le plus efficace que les autres en terme de taux de précision. Par conséquence, le nombre d'epochs, la taille de la base et la profondeur de réseaux, sont des facteurs importants pour l'obtention de meilleurs résultats.

Conclusion générale

La distinction entre les images spam et les images de ham est un problème difficile en soi. A travers ce projet, nous avons fait une généralité sur la détection des spam image à travers les réseaux de neurone convolutif qui exploite les convolutions séparables en profondeur. En parcourant les chapitres, nous avons décrit et clarifié la définition des spam image, l'architecture du système, les objectifs atteints, l'exposition des résultats obtenus et leur discussion lors de différents tests réalisés. Notre système utilise un classifieur de spam image créé à l'aide de Deep Learning avec trois architectures CNN (CNN1, CNN2 et VGG16) en utilisant 3 datasets différents. Certains des modèles proposés ont obtenu de meilleurs résultats par rapport au classifieur traditionnel comme SVM. On peut en déduire qu'afin de construire un meilleur classificateur de spam d'image, des informations supplémentaires, telles que les métadonnées, devraient également être évaluées lors de la formation du modèle.

Comme perspectives, nous souhaitons :

- Tester notre modèle sur d'autres bases des données plus volumineuse.
- Utiliser l'architecture VGG avec d'autres architectures en appliquant une hybridation comme l'architecture Xception ou LeNet.

Bibliographie

(Annadatha & Stamp, 2018): Annadatha, A., & Stamp, M. (2018). Image spam analysis and detection. Journal of Computer Virology and Hacking Techniques, 14(1), 39–52. doi:10.1007/s11416-016-0287-x.

(Agam, 2006): Agam, G. (2006). Introduction to programming with OpenCV. Department of .al., & Bishopet, C. M. (1995.). Neural networks for pattern recognition. Oxford university press.

(Aiwan & Zhaofeng, 2018): Aiwan, F., & Zhaofeng, Y. (2018). Image spam filtering using convolutional neural networks. Personal Ubiquitous Computing, 22(5–6), 1029–1037. doi:10.1007/s00779-018-1168-8.

(BENGIO & al., 2007): BENGIO, Y., & al., Y. L. (2007). Scaling learning algorithms towards AI. Large-scale kernel machines 34.5.

(Chiyuan Zhang, 2016): Chiyuan Zhang, S. B. (2016). Understanding deep learning requires rethin-king generalization.

(Chowdhury et al, 2015): Chowdhury, M., Gao, J. & Chowdhury, M. (2015). Image Spam Classification Using Neural Network. Australia: Springer International Publishing, 2015, pp. 622–632.

(Chavda, and al, 2018): Chavda, A., Potika, K., Di Troia, F., & Stamp, M. (2018). Support vector machines for image spam analysis. Proceedings of the 15th international joint conference on e-business and telecommunications (pp. 597–607), Porto Portugal.

(Chollet, 2017): Chollet, F. (2017). Xception : Deep learning with depthwise separable convolutions.

(Christian Szegedy, 2015): Christian Szegedy, W. L.-m. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pat- tern recognition.

(Colin Lea, 2016): Colin Lea, R. V. (2016). Temporal convolutional networks . A unified approach to action segmen- tation. In European Conference on Computer Vision.

(Dada et al, 2019): Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwae, O. E. (2019). Machine learning for email spam filtering: Review, approaches and open research problems.Heliyon, 5 (6), e01802.

(Dhabi et al. 2020): Dhahi E.H., Ali S.A., and Naser M.A. (2020) Text Region Extraction for Noisy Spam Image. In: Mallick P., Balas V., Bhoi A., Chae GS. (eds) Cognitive Informatics and Soft Computing. Advances in Intelligent Systems and Computing, vol 1040. Springer, Singapore.

(Dredze, et al, 2007): Dredze, M, Gevaryahu, R. & Elias-Bachrach, A., (2007). Learning fast classifiers for image spam. in CEAS, 2007, pp. 2007–487.

(Dhanaraj & Karthikeyani, 2013): Dhanaraj, S., & Karthikeyani, V. (2013). A study on e-mail image spam filtering techniques. In 2013 international conference on pattern recognition, informatics and mobile engineering (pp. 49–55).

(DEMYANOV, 2015): DEMYANOV, S. (2015). Regularization methods for neural networks and related models. Thèse de doct.

(entrainement-dun-rseau-de-neurones-convolutif, 2018): entrainement-dun-rseau-neurones-convolutif. (2018, MAY 17). Récupéré sur natural-solutions:https://www.natural-solutions.eu/blog/entrainement-dun-rseau-de-neurones-convolutif.

(Gao et al., 2008): Gao, Y., Yang, M., Zhao, X., Pardo, B. Wu, Y. , Pappas, T. N. & Choudhary, A. (2008). Image spam hunter, IEEE International Conference on Acoustics, Speech and Signal Processing., pp. 1765–1768.

Bibliographie

(Hassanet al., 2017): Hassan, M., Mirza, W., & Hussain, M. (2017, October). Header based spam filtering using machine learning approach. International Journal of Emerging Technologies in Engineering Research, 5 (10), 133–140.

(Hosseini et. al 2015): Hosseini, M. S. & Rahmati, M.. (2015). A Method for Image Spam Detection Using Texture Features, International Academic Journal of Science and Engineering, vol. 2, pp. 51-58, 2015.

(Ketari, et al, 2012): A Study of Image Spam Filtering Techniques. In Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference, pp. 245-250, 2012.

(Kim, 2014): Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Me- thods in Natural Language Processing, Doha, Qatar.

(Krizhevsky, Sutskever, & Hinton., 2012): Krizhevsky, A., Sutskever, I., & Hinton., G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems.

(Kumar, R, & KP, 2018): Kumar, A. D., R, V., & KP, S. (2018). Deep image spam: Deep learning based image spam detection. Retrieved from https://arxiv.org/abs/1810.03977.

(Kumaresan et. al 2015): Kumaresan, T., Sanjushree, S., Suhasini, K. & Palanisamy, C. (2015). Image spam filtering using support vector machine and particle swarm optimization. IJCA Proceedings on National Conference on Information Processing and Remote Computing.

(les réseaux de neurone): (s.d.). Récupéré sur Statistica.

(Liu et al. 2010): Liu Q, Zhang F, Qin Z, Wang C, Chen S, Ma Q (2010) Feature selection for image spam classification. In: International conference on communications, circuits and systems (ICCCAS), China.

(MANZANERA & SAUX, 2020): MANZANERA, A., & SAUX, B. L. (2020). Réseaux de neurones profonds pour la classification d'objets en imagerie infrarouge : apports de l'apprentissage à partir de données. thèse de doctorat, L'ENSTA BRETAGNE ÉCOLE DOCTORALE NO 601 Mathématiques et Sciences et Technologies de l'Information et de la Communication, Paris.

(Nassima, 2020): Nassima, D. (2020). L'apprentissage profond pour letraitement d'images. thèse doctorat, UniversitéDjillaliLiabès Faculté des sSciencesExactes Département d'informatique Laboratoire EEDIS, SidiBelAbbès.

(Nisha et Gaikwad, 2015): Nisha D. Chopra, K. P. Gaikwad, (2015). Image and Text Spam Mail Filtering, International Journal of Computer Technology and Electronics Engineering (IJCTEE), vol 5, pp.15-18, June 2015.

numpy: (s.d.). Récupéré sur https://www.numpy.org/

(Poulain, 2019): Poulain, P. F.-P. (2019). livre Cours de Python. Université Sorbonne paris.

(python): python.(s.d.).Récupéré sur http://www.linux center.org/articles/9812/python.html.

(Rex Ying, 2018): Rex Ying, R. H.-c. (2018). Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery.

(Runbox, 2017): How email works. (2017). https://blog.runbox.com/articles/how-email-works/.

(Shanget al., 2016): Shanget, E.X., Zhang, H.G.: Image spam classification based on convolutional neural network. In: Proceedings of the 15th International Conference on Machine Learning and Cybernetics, pp. 398–403 (2016).

(Simonyan & Zisserman, 2014): Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Bibliographie

(Tolentino, 2015): Tolentino, J. (2015). 5 types of social spam (and how to prevent them). TNW.https://thenextweb.com/future-of-communications/2015/04/06/5-types-of-social-spam-and-how-to-prevent-them/.

(Wei Hu, 2015): Wei Hu, Y. H. (2015). Deep convolutional neural networks for hyperspectral image classification.

(Wiesel, 1962.): Wiesel, D. H. (1962.). binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology .Xiang Li, X. H. (2019). Improving semantic feature learning in convolutional networks.

(Zakaria, 2017): Zakaria, M. M. (2017). Classification des images avec les réseaux de neurones convolutionnels. Université Abou Bakr Belkaid, Tlemcen.

ملخص

جراء تقدم الذكاء الاصطناعي الحديث ظهرت تقنيات جديدة مكنتنا من التصدي لانتهاكات الشبكة العنكبوتية و هذا عن طريق التعلم العميق.

سابقا تجلت تقنيات اقتحام البريد الالكتروني في الرسائل الغير مرغوبة(spam) لكن تصدى لها التقنيون و مع التطور التكنولوجي ظهرت الصور العشوائية ظاهرها تحسين وضع او تسويق منتج و باطنها سلب خصوصية المستخدم.

بفضل الجهود المتواصلة في مجال الامن المعلوماتي استخدم نظام الاعصاب اللفافية لحماية المستخدم من الصور العشوائية(spam image) و هذا ما تطرقنا اليه في مشروعنا هذا مع مقارنة النتائج المتحصل عليها مع نتائج تقنيات اخرى .

Abstract

Electronic mail (e-mail) has become one of the most popular and frequently used channels for personal and business communication online. Despite its advantages, e-mail faces a major security problem, which is the daily receipt of a large number of unsolicited e-mail messages, called "spam". Today, most e-mail systems have simple spam filtering mechanisms based on text-based spam filtering technologies. To circumvent these powerful text-based detection filters, spammers have responded by introducing new techniques for embedding spam text in email attachments, called "image spam". Image spam is a type of spam in which the message text is embedded in an image that is then attached to the e-mail. To deal with it, researchers have several machine learning approaches that use various features such as metadata, color, shape, texture features, etc. With the emergence of the concept of deep learning and large databases, a new research direction is developed. Our project consists in proposing an image spam detection system based on deep learning and more particularly on convolutional neural networks. This system is able to distinguish between spam images and legitimate images (ham). We propose two different CNN models and compare the results obtained with the VGG model and other machine learning techniques such as SVM. Experimental tests have been performed on a real database, and we obtained satisfactory results.

Résumé

Le courrier électronique (e-mail) est devenu l'un des canaux les plus populaires et les plus fréquemment utilisés pour la communication personnelle et professionnelle en ligne. Malgré ses avantages, le courrier électronique est confronté à un problème de sécurité majeur, qui est la réception quotidienne d'un grand nombre de messages électroniques non sollicités, appelés "spams". Aujourd'hui, la plupart des systèmes de messagerie électronique disposent de mécanismes simples de filtrage des spams basés sur des technologies de filtrage des spams textuels. Pour contourner ces puissants filtres de détection basés sur le texte, les spammeurs ont réagi en introduisant de nouvelles techniques d'intégration de texte de spam dans des images jointes au courrier électronique, appelées "spam image". Le spam image est une sorte de spam dans lequel le texte du message est incorporé dans une image qui est ensuite jointes à l'e-mail. Pour y faire face, les chercheurs disposent de plusieurs approches d'apprentissage automatique qui utilisent diverses fonctionnalités telles que les métadonnées, la couleur, la forme, les caractéristiques de texture, etc. Avec l'apparition du concept de deep Learning (apprentissage en profondeur) et les bases de données volumineuses, un nouvel axe de recherche est développé. Notre projet consiste à proposer un système de détection de spam image basé sur le deep Learning (apprentissage en profondeur) et plus particulièrement les réseaux de neurones convolutifs. Ce système est capable de faire la distinction entre les images spam et les images légitimes (ham). Nous proposons deux modèles CNN différents et nous comparons les résultats obtenus au modèle VGG et à d'autres techniques d'apprentissage automatique tel que le SVM. Des tests expérimentaux ont été réalisés sur une base de données réelle, et nous avons obtenus des résultats satisfaisants.