Université de BLIDA 1
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'étude

Option : *Sécurité des systèmes d'information*

Un modèle d'apprentissage en profondeur pour détecter les URLs d'hameçonnage

---

Réalisé Par :

Feknous Chalabia Lilia

Della Fatma Zohra Roumaissa

Membres de jurys composés de :

Mme.ABED Hafida	Presidente
Mme.BOUDRAA Sawsen	Examinatrice
Mme. BOUMAHDI Fatima	Promotrice
Mr. REMMIDE Mohamed Abdelkarim	Co-Promoteur

Soutenu le 06/07/2022

Résumé

L'hameçonnage est un type d'attaque informatique qui communique des messages d'ingénierie sociale aux humains via des canaux de communication électroniques afin de les persuader d'effectuer certaines actions au profit de l'attaquant. Ce travail propose une approche d'apprentissage profond basées sur l'utilisation des réseaux neuronaux convolutifs temporels (TCN) pour la détection des URLs de phishing. Le modèle proposé utilise des architectures profondes pour classer les adresses URL comme légitimes ou illégitimes.

Dans ce mémoire, plusieurs jeux de données ont été utilisé et sont disponible sur la plateforme Kaggle. Une précision optimale de $99,96\%$ a été atteinte après un ensemble de diverses séries d'expériences avec l'utilisation de plusieurs méthodes et techniques qui seront examinées dans ce mémoire.

Mots Clés : Hameçonnage,réseaux neuronaux convolutifs temporels(TCN),Uniform Resource Locator(URL), Apprentissage profond.

Abstract

Phishing is a type of computer attack that communicates social engineering messages to humans via electronic communication channels in order to persuade them to perform certain actions to benefit the attacker. This work proposes a Deep Learning approach based on the use of Temporal Convolutional Neural Networks (TCN) for the detection of phishing URLs.

The proposed model uses deep architectures to classify URLs as legitimate or illegitimate. In this research, several datasets were used and are available on the Kaggle platform. An optimal accuracy of 99.96%was achieved after a set of various sets of experiments with the use of several methods and techniques that will be discussed in this thesis.

Keywords : Phishing, Temporal Convolutional Network (TCN), Uniform Resource Locator (URl), Deep Learning

ملخص

التصيد الاحتيالي هو نوع من هجمات الكمبيوتر التي تنقل رسائل الهندسة الاجتماعية إلى البشر من خلال قنوات الاتصال الإلكترونية لإقناعهم بتنفيذ إجراءات معينة لصالح المهاجم.

يقترح هذا العمل نهج التعلم العميق القائم على استخدام الشبكات العصبية التلافيفية الزمنية للكشف عن روابط التصيد.

يستخدم النموذج المقترح معماريات عميقة لتصنيف الروابط على أنها مشروعة أو غير مشروعة.

في هذا البحث تم استخدام العديد من مجموعات البيانات وهي متوفرة على منصة kaggle. وقد تم تحقيق الدقة المثلى بنسبة 99.96٪ بعد مجموعة من التجارب المتتالية باستخدام عدة طرق وتقنيات سيتم دراستها في هاته المذكرة.

كلمات مفتاحية: التصيد الاحتيالي، الشبكات العصبية التلافيفية الزمنية، الروابط ،تعلم عميق.

Remerciments

Nous tenons à exprimer tous nos reconnaissances à nos promoteurs Madame Boumahdi Fatima et Monsieur Remmide Abdelkarim. Nous les remercions de nous avoir encadrés, orientés, aidés et conseillés lors de nos recherches.

Nous remercions nos très chers parents Feknous Abdelkarim, Abada Khadidja, Della Karim et Touazi Karima, qui ont toujours été là pour nous.

Nous remercions aussi notre sœur Feknous Sonia et nos frères Della Abdelmadjid, Della Yasser ,Feknous Ameur Abdellali et Feknous AbdelRahim pour leurs encouragements.

Et en fin nous remercions nos amies Rayane Ait Ali Yahia,Cheifa Ikram et Yahiaoui Asma pour leurs soutien et encouragement qui ont été un grand aide.

Table des matières

Table des figures

Liste des tableaux

Introduction

Avec le développement rapide d'Internet et la popularité croissante du paiement électronique dans les services Web, la fraude sur Internet et la sécurité du Web ont progressivement été la principale préoccupation du public [Huang et al., 2017]. Le phishing Web est un moyen de fraude de ce type, qui utilise une technique d'ingénierie sociale par le biais de messages courts, d'e-mails et de WeChat pour inciter les utilisateurs à visiter de faux sites Web afin d'obtenir des informations sensibles telles que leur compte privé, un jeton de paiement, des informations de carte de crédit.

Problématique

La première attaque de phishing sur AOL (AmericaOnline) remonte au début de 1995 [Rekouche , 1995]. Un hameçonneur a réussi à obtenir les informations personnelles des utilisateurs d'AOL. Cela peut conduire non seulement à l'abus d'informations de carte de crédit, mais également à une attaque contre le système de paiement en ligne tout à fait réalisable. L'activité de phishing au début de 2016 était la plus élevée jamais enregistrée depuis le début de la surveillance en 2004. Le nombre total d'attaques de phishing en 2016 était de 1 220 523 attaques [Activity et Report, 2020].Il s'agit une croissance progressive passant de 162.155 au dernier trimestre de 2019 à 165 772 [Activity et Report, 2020] cas au premier trimestre 2020. Le phishing à causer de graves dommages à de nombreuses organisations et à l'économie mondiale. Soit une augmentation de 5753% sur 12 ans [Activity et Report, 2020] .

Selon le 3ème rapport Microsoft Computing Safer Index publié en février 2014, l'impact mondial annuel du phishing pourrait atteindre 5 milliards de dollars. Avec la prévalence du réseau, le phishing est devenu l'une des menaces de sécurité les plus graves dans la société moderne, faisant ainsi de la détection et de la défense contre le phishing Web une tâche de recherche urgente et essentielle. La détection du phishing Web est cruciale pour les utilisateurs

privés et les entreprises [Wu et al., 2015]. Certaines solutions possibles pour lutter contre le phishing ont été créées, y compris une législation et des technologies spécifiques. D'un point de vue technique, la détection du phishing comprend généralement les catégories suivantes : détection basée sur une liste noire [Dinler et Sahin, 2021] et une liste blanche, détection basée sur les fonctionnalités d'URL (Uniform Resource Locator) [Vaitkevicius et Marcinkevicius, 2020], détection basée sur le contenu web, et la détection basée sur l'apprentissage automatique.

Objéctif

Notre objéctif est de créé un nouveau système de deep learning qui détécte les urls d'hammaçonnages et sécurise l'accès au web .Des chercheurs ont travaillé sur l'amélioration de la précision de la détection de phishing via diverses méthodes basées sur des listes et sur l'apprentissage automatique qui tirer parti des fonctionnalités artisanales des URL. Néanmoins, la détection du phishing reste une course aux armements sans solution définitive en raison de la constante évolution du phishing, qui rend les techniques capables d'extraire fonctionnalités utiles automatiquement et détecter les sites Web de phishing avec précision pour contrecarrer les attaques de phishing continuent d'être une priorité avoir besoin. Pour résoudre ce problème, nous proposons une approche basée sur l'apprentissage profond. Approche qui peut identifier avec précision les sites Web de phishing via des techniques d'incorporation de mots qui sont particulièrement utiles pour traiter les mots rares, un problème généralement observé dans les tâches de détection d'URL malveillantes.

Les attaques de phishing peuvent se propager en intégrer des URL de phishing dans des messages falsifiés. Donc, développer des techniques pour détecter efficacement les URL de phishing peut aider à contrecarrer les attaques de phishing dans une large mesure. La raison pourquoi nous faisons l'extraction des représentations de caractéristiques à partir d'URL uniquement. Tout d'abord, extraire des fonctionnalités à partir d'informations sur l'hôte ou le contenu du site Web prend du temps en raison de la nécessité interaction avec les sites Web. Deuxièmement, le phishing basé sur la détection des URLs a montré des performances prometteuses puisqu'il est non seulement léger, mais a également une précision de détection relativement élevée . En raison de l'accent mis sur l'URL elle-même, notre approche peut détecter les sites Web de phishing plus rapidement que d'autres méthodes qui prendre en compte le contenu du site Web ou les informations sur l'hôte en plus ; en attendant, notre approche peut être appliquée partout où une URL peut être intégrée, par exemple, des e-mails, des messages Twitter, etc.

Plan du mémoire

Le mémoire est organisé en cinq chapitres suivis d'une conclusion générale, chacun représente une partie du travail de ce projet qu'elle soit théorique ou pratique :

- Le premier chapitre clarifie le contexte général de ce travail en présentant l'hameçonnage avec ces différents types, impact des attaques de phishing, avec les techniques de détection et méthodes de protection.

- Pour le 2eme chapitre nous allons présenté l'apprentissage automatique ainsi que ses différents types. Nous avons exploré également les différents modèles d'apprentissage profond pour le traitement de notre problématique.

- Dans le chapitre 3 Nous allons cité l'ensemble des travaux existants, qui tente d'étudier et saisir les avantages et les faiblesses des différentes approches proposées, dans la littérature pour la détection des URLs de phishing.

- Dans le chapitre 4 Nous allons présenter le processus de notre solution avec les concepts utilisés pour la classification des URLs de phishing.

- Et le dernier chapitre sera consacré à la présentation et l'évaluation des algorithmes que nous proposons pour la détection d'hameçonnage avec l'application finale pour le déploiement de ces algorithmes.

Chapitre 1

GÉNÉRALITÉS SUR LE PHISHING

1.1 Introduction

Le World Wide Web est devenu le critère le plus essentiel pour la communication de l'information et la diffusion des connaissances. Il aide à traiter les informations en temps opportun, rapidement et facilement. Le vol et la fraude d'identité sont considérés comme les deux faces de la cybercriminalité dans lesquelles les pirates informatiques et les utilisateurs malveillants obtiennent les données personnelles des utilisateurs légitimes existants pour tenter une fraude ou une tromperie à des fins de gain financier. Ce chapitre présente le phishing et ses divers types avec les principales méthodes de détection d'hameçonnages par URL.

1.2 Définition d'Hameçonnage

L'hameçonnage est un cybercrime qui utilise de faux e-mails, sites Web et SMS pour voler des informations personnelles et d'entreprise sensibles. La victime est amenée à fournir des informations personnelles telles que son adresse, sa date de naissance, son nom et son numéro de sécurité sociale.

1.3 Types d'Hameçonnages

Il existe plusieurs types de phishing qui sont :

1.3.1 Courrier

Il s'agit de la tactique de phishing la plus courante. On envoie des courriers à plusieurs destinataires les exhortant de mettre à jour leurs informations personnelles, de vérifier les détails de leur compte ou de changer leur mot de passe. De façon générale, le courrier est formulé de façon à susciter chez le destinataire un sentiment d'urgence à se protéger contre un crime. En

apparence, il semble provenir d'une source légitime, par exemple le service à la clientèle d'Apple, une banque, Microsoft, PayPal ou une autre entreprise connue [Alkhalil et al., 2021].

1.3.2 Injection De Contenu

Du contenu malveillant est injecté dans un endroit Web commun, par exemple, la page d'accueil d'un compte de mail électronique ou d'une institut bancaire en ligne. Ce contenu prend la forme d'un lien, d'une forme ou d'une fenêtre surgissant qui dirige les gens vers un site Web secondaire où on leur demande de confirmer leurs informations personnelles, mettre à jour les détails de leur carte de crédit et changer leurs mots de passe [Abutair et al., 2019].

1.3.3 Formulation Des Liens

L'e-mail soigneusement conçu contient un lien malveillant vers un site Web bien connu comme Amazon. Cliquer sur le lien redirige les personnes vers un faux site Web identique au site Web d'origine, où elles sont invitées à mettre à jour ou à vérifier les informations de leur compte [Abutair et al., 2019].

1.3.4 Faux Sites Web

Les pirates créent des faux sites Web qui ont littéralement la proche apparence que les vrais. Le crins site Web se distingue par un domaine inconsidérément différent, par exemple yahoo.you.live.com au place de yahoo.live.com. Les monde croient qu'ils sont sur le bon site Web et s'exposent malheureusement au vol d'identité [Abutair et al., 2019].

1.3.5 Maliciel

Les malicieux sont déclenchés par une personne qui clique sur la pièce jointe d'un courrier et qui installe par inadvertance un logiciel qui fouille l'ordinateur et le réseau à la recherche d'information. L'enregistreur de frappe (keylogger) est un type de logiciel qui enregistre la pression des touches sur le clavier et permet de révéler des mots de passe. Le cheval de Troie est un autre type de maliciel utilisé qui incite les gens à saisir leurs informations personnelles. Un ransomware un autre type de maliciel qui s'introduit dans votre système et cherche à accéder à vos fichiers et dossiers. Les ransomwares sont créés par des groupes de cybercriminels qui vont tout simplement vous demander une rançon, en l'échange de la clé qui permet de déchiffrer ces fichiers [Abutair et al., 2019].

1.3.6 L'homme Du Milieu

Dans l'attaque l'homme du milieu, le criminel emmène deux personnes à partager de l'information entre elles. L'hameçonner peut soit envoyer de fausses demandes à chacune des parties ou modifier l'information qui est transmise. Les personnes impliquées ont l'impression qu'elles communiquent entre elles et n'ont aucune idée qu'elles sont manipulées par une tierce partie [Alkhalil et al., 2021]

1.3.7 Vishing

Le phishing vocal ou vishing est la version audio du phishing sur Internet. L'attaquant tente de persuader les victimes par téléphone de fournir des informations personnelles qui peuvent ensuite être utilisées pour le vol d'identité. De nombreux appels automatisés sont des tentatives de vishing [Leonov et al., 2022].

1.3.8 Smishing

Le smishing est la version SMS du phishing. Vous recevrez un SMS vous demandant de cliquer sur un lien ou de télécharger une application. Mais en fait, il télécharge des logiciels malveillants sur votre téléphone. Ce logiciel malveillant détourne vos informations personnelles et les envoie au pirate [Leonov et al., 2022].

1.3.9 Pharming

Le pharming est une attaque visant à rediriger le trafic d'un site web vers un autre site frauduleux. Le pharming interfère avec la résolution du nom de domaine à une adresse IP, de sorte que le nom de domaine du site web authentique est mappé sur l'adresse IP d'un faux site web fictif [Banu et al., 2013].

1.3.10 Whaling

Whaling est une forme de phishing visant la direction d'organisations. L'entreprise peut recevoir un e-mail qui ne semble pas dangereux et qui contient un rappel ou une plainte concernant un emploi, un appel aux forces de l'ordre ou simplement une clarification de toute information. Ce type d'attaque peut également conduire à une menace persistante avancée (APT) interne. Lorsque des liens ou des pièces jointes sont ouverts, cela permet d'accéder à des informations d'identification et à d'autres informations personnelles ou de lancer un logiciel malveillant qui conduira à une APT [Leonov et al., 2022].

1.3.11 Harponnage

Le harponnage est une attaque de phishing qui cible une personne ou un groupe de personnes spécifiques. Pour ce faire, les attaquants effectuent généralement un travail de renseignement approfondi, en faisant des recherches dans les médias sociaux et d'autres sources contenant des informations sur leur cible, ce qui augmente considérablement les chances de succès. L'objectif du harponnage est le même : obliger la victime à se rendre sur un faux site et à obtenir des informations d'identification, ou la forcer à ouvrir un document spécifique en cliquant sur un lien, ce qui installera automatiquement un logiciel malveillant. Grâce à ce logiciel malveillant, les attaquants peuvent manipuler à distance un ordinateur infecté [Leonov et al., 2022]

1.4 Fonctionnement d'Hameçonnage URL

Les pirates créent des sites de phishing pour récolter des données personnelles ou d'autres données précieuses. Ils envoient des messages électroniques à leurs victimes pour tenter de les attirer sur le site de phishing comme montre l'exemple illustré dans la figure 1.1. Ces attaques sont couronnées de succès lorsque la victime suit un lien vers un site Web et fournit toutes les informations demandées. Normalement, ces liens sont déguisés en réinitialisation de mot de passe ou en confirmation d'identité pour des services légitimes le site web est également déguisé pour que la victime ne remarque pas qu'il s'agit d'un faux site web.

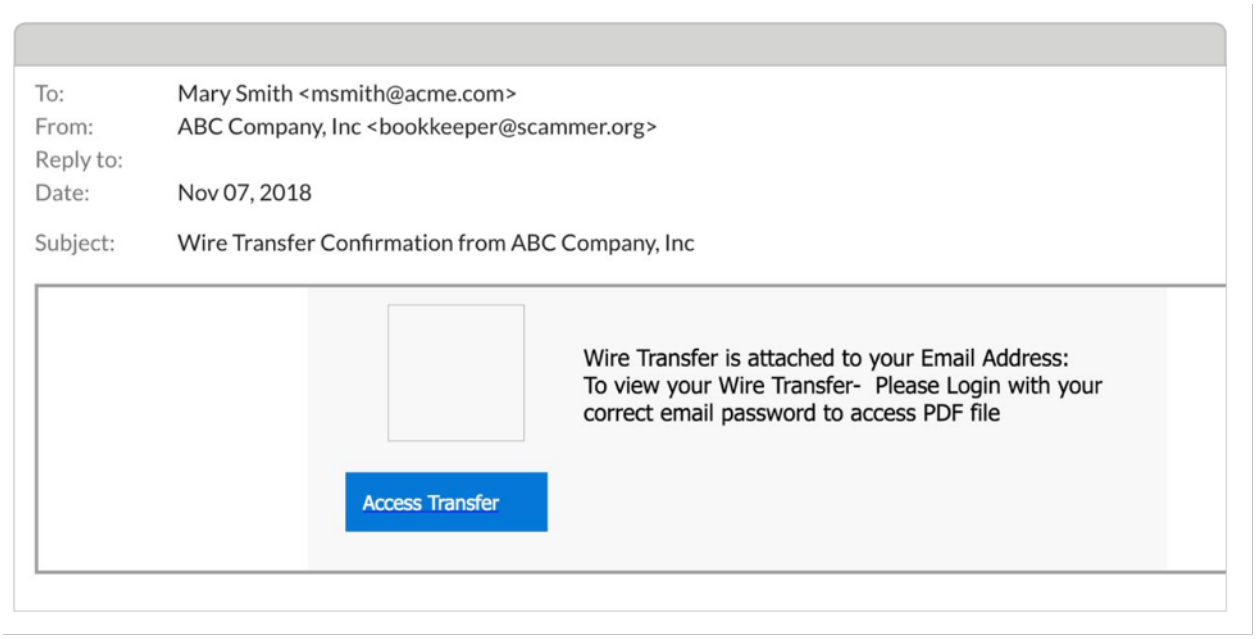

FIGURE 1.1 – Exemple phishing par url.

Ce fonctionnement se fait en cinq phases :

1.4.1 Construction du Faux Site web

Le pirate récupère le code réel et les images légitimes d'un site web pour construire des sites frauduleux. Selon certaines estimations, les cybercriminels créent 1,5 million de sites frauduleux chaque mois. C'est plus facile que jamais, grâce aux outils de scraping web qui sont à la portée de tous. En général, les pirates créent des sites d'imitation à partir de domaines connus et fiables. Et ils sont de plus en plus habiles dans leur art. Même les professionnels de la sécurité bien formés peuvent avoir du mal à distinguer les faux [Charton, 2013] .

1.4.2 Envoi De l'Email De Phishing

Une fois que le pirate a créé le faux site, il lance une campagne de phishing par courrier électronique. Ces courriels contiennent lien vers le site frauduleux. L'email incite la victime à cliquer sur le lien [Charton, 2013].

1.4.3 Réduire La Recherche D'Une Victime

Si les emails n'atteignent pas la cible recherchée, le pirate continue à essayer de trouver les bonnes adresses mail. Cependant, cette étape est un peu un "jeu de devinettes" pour l'attaquant [Charton, 2013].

1.4.4 Mordre à l'Hameçon

Tôt ou tard, si un pirate est persistant, une victime sans méfiance tombe dans le piège. Une attaque non détectée peut faire des milliers de victimes [Charton, 2013].

1.4.5 Collecte Des Données Volées

Les pirates qui volent les données des clients recherchent différentes choses. Parfois, ils veulent voler les actifs financiers de la victime, comme des cartes de crédit, des comptes bancaires ou une déclaration d'impôts. D'autres veulent rassembler autant d'informations d'identification que possible pour les vendre sur le dark web et en tirer un profit considérable. Certains adversaires cherchent à exposer ou à humilier les victimes en révélant des informations confidentielles au public [Charton, 2013].

1.5 Impact Des Attaques d'Hameçonnage

Les attaques de phishing utilisent diverses techniques telles que la manipulation de liens, l'évasion de filtres, la falsification de sites Web, la redirection secrète et l'ingénierie sociale. L'approche la plus courante consiste à créer une page Web d'usurpation d'identité qui imite un site Web légitime. Ces types d'attaques étaient les principales préoccupations des derniers rapports sur la criminalité Internet 2018, publié par le Centre des plaintes contre la criminalité Internet (IC3) du Fédéral Bureau of Investigations des États-Unis. Les statistiques recueillies par le FBI IC3 pour 2018 ont montré que le vol, la fraude et l'exploitation sur Internet restent omniprésents et ont été à l'origine de pertes financières stupéfiantes de 2,7 milliards de dollars en 2018 [Alkhalil et al., 2021]. l'IC3 a reçu 20 373 plaintes[Activity et Report, 2020] contre la compromission des e-mails professionnels et la compromission des comptes de messagerie, avec des pertes de plus de 1,2 milliard de dollars [Activity et Report, 2020]. Le rapport note que le nombre de ces attaques sophistiquées a augmenté de plus en plus ces dernières années. L'Anti-Phishing Working Group (APWG) souligne que les attaques de phishing se sont multipliées ces dernières années ; figure 1.2 illustre le nombre total de sites de phishing détectés par l'APWG au premier trimestre de 2020 et le dernier trimestre de 2019.

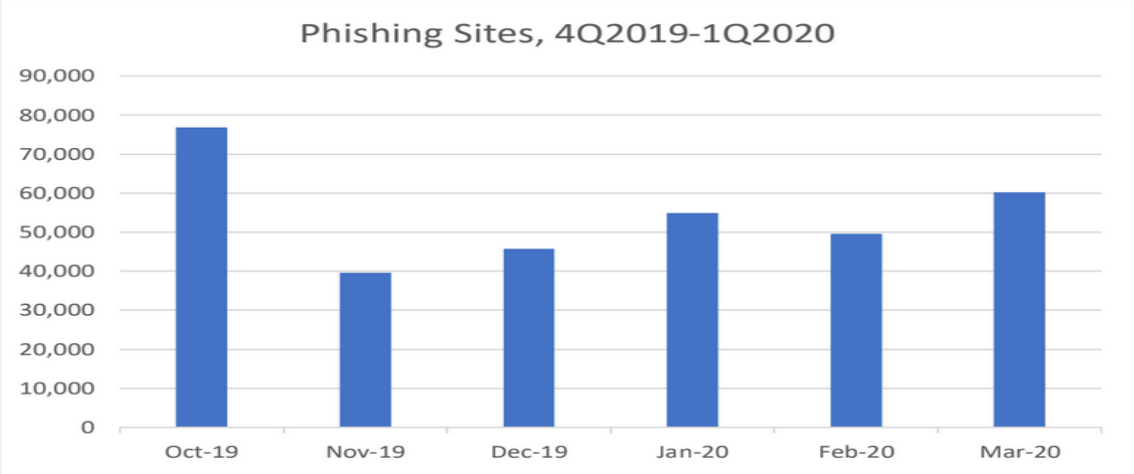

FIGURE 1.2 – Nombre total de sites Web de phishing détectés par l'APWG

Ce nombre a une croissance progressive passant de 162.155 [Activity et Report, 2020] au dernier trimestre de 2019 à 165 772[Activity et Report, 2020] cas au premier trimestre 2020. Le phishing a causé de graves dommages à de nombreuses organisations et à l'économie mondiale. Au quatrième trimestre de 2019, OpSec Security, membre de l'APWG, a constaté que les sites SaaS et de messagerie Web restait les cibles les plus fréquentes des attaques de phishing. Les hameçonneurs continuent de récolter les informations d'identification de ces cibles en exploitant BEC et accèdent ensuite aux comptes SaaS d'entreprise. De nombreuses approches ont été utili-

sées pour filtrer les sites Web de phishing. Chacune de ces méthodes est applicable à différentes étapes du flux d'attaque, par exemple, la protection au niveau du réseau, l'authentification, l'outil côté client, la formation des utilisateurs, les filtres côté serveur et les classificateurs. Bien qu'il existe des caractéristiques uniques dans chaque type d'attaque de phishing, la plupart de ces attaques présentent des similitudes et des modèles traits de phishing communs, par conséquent, reconnaître les sites Web de phishing.

1.6 Approches De Détection d'Hameçonnage

Les approches de détection sont une solution anti-phishing qui vise à identifier ou à classer les attaques de phishing, ce qui inclut les approches de formation des utilisateurs et les approches de classification de logiciels [Khonji et al., 2013].

1.6.1 Approches Défensives

Rendre les campagnes de phishing inutiles pour les attaquants en les perturbant. Ceci est souvent réalisé en inondant les sites Web de phishing avec de fausses informations d'identification afin que l'attaquant aurait du mal à trouver les véritables informations d'identification telles que BogusBiter et Humboldt [Khonji et al., 2013].

1.6.2 Approches De Correction

Dans le cas des attaques de phishing, la correction [Khonji et al., 2013] consiste à supprimer les ressources de phishing. Cela s'étend également à la fermeture des entreprises qui fournissent fréquemment des services aux attaquants par hameçonnage. Par exemple :

- Suppression du contenu d'hameçonnage des sites Web ou suspension des services d'hébergement
- Suspension des comptes de messagerie, des relais SMTP, des services VoIP.
- Retracer et arrêter les botnets. Cela s'étend également à la fermeture des entreprises qui fournissent fréquemment des services aux attaquants par hameçonnage.

Le processus de fermeture peut être initié par des organisations qui fournissent des services de protection de marque à leurs clients, qui peuvent inclure des sociétés bancaires et financières susceptibles d'être victimes d'attaques de phishing. Lorsque des campagnes de phishing sont identifiées, elles peuvent être signalées à leurs fournisseurs de services d'hébergement Internet et d'hébergement Web pour un arrêt immédiat.

1.6.3 Approches De Prévention

Minimiser la possibilité pour les attaquants de lancer des campagnes de phishing via des poursuites et des sanctions contre les attaquants par les agences d'application de la loi (LEA). Une fois les sources des attaques de phishing identifiées, LEA peut alors engager des poursuites judiciaires qui peuvent à leur tour infliger des sanctions telles que : des peines d'emprisonnement, des amendes et la confiscation des équipements utilisés pour véhiculer les attaques [Khonji et al., 2013].

1.7 Protection Contre l'Hameçonnage

Pour luter la menace du phishing, un certain nombre de solutions anti-phishing ont été proposées, Les techniques anti-hameçonnage peuvent en général être divisées en trois catégories. 1. Les filtres anti-spam 2. Barres d'outils anti-hameçonnage et 3. Mécanisme de protection par mot de passe.

1.7.1 Filtres Anti-Spam

Une catégorie de tactiques anti-hameçonnage vise à résoudre le problème de l'hameçonnage au niveau du courrier électronique. L'idée importante est que si un courrier électronique d'hameçonnage n'atteint pas ses victimes, celles-ci ne peuvent pas tomber dans l'escroquerie. Par conséquent, des filtres et des techniques d'évaluation du contenu sont régulièrement utilisés pour tenter de percevoir les courriers électroniques d'hameçonnage avant qu'ils ne soient envoyés aux clients. Il est certain que cette ligne d'études est étroitement liée à la recherche sur le courrier non sollicité[Dumais., 1996]. par le biais d'un entraînement continu des filtres (par exemple, les filtres bayésiens), une grande variété de courriers électroniques d'hameçonnage peuvent être bloqués, Cela est dû au fait que ces courriels contiennent souvent des mots qui peuvent être diagnostiqués comme des jetons suspects qui n'apparaissent pas fréquemment dans les courriels valides (par exemple, login) Le principal inconvénient des techniques anti-spam est que leur succès dépend de la disponibilité de ces filtres et de leur formation adéquate. En d'autres termes, si la personne ne participe plus activement à la formation du filtre, celui-ci ne fonctionne généralement pas comme prévu. De plus, même si les filtres sont bien formés et qu'une personne reçoit rarement des courriers non sollicités ou des courriers de phishing, dès qu'un courrier électronique de phishing contourne le filtre, la croyance du consommateur en la légitimité de ce courrier est renforcée.

1.7.2 Barres D'outils Anti-hameçonnage

Pour découvrir qu'une page est un site d'hameçonnage en ligne, plusieurs stratégies peuvent être utilisées, notamment les listes blanches (listes de sites Web considérés comme sûrs), les listes noires (listes de sites frauduleux reconnus), Un certain nombre de barres d'outils qui peuvent être utilisées pour l'anti-hameçonnage sont [Cranor et al., 2006],

1.7.2.1 Barre d'outils eBay

La barre d'outils eBay [J. Hong et al., 2006] utilise une combinaison d'heuristiques et de listes noires. La barre d'outils donne en outre aux utilisateurs la possibilité de signaler sites Web d'hameçonnage, afin qu'ils soient ensuite testés avant d'être sur une liste noire.

1.7.2.2 Barre d'outils TrustWatch De GeoTrust

Le site Web de GeoTrust [J. Hong et al., 2006]en ligne ne fournit aucune information sur la manière dont TrustWatch détermine si un site est enregistré ou non sur la liste noire. Comment TrustWatch détermine si un site est frauduleux ; cependant on peut supposer que l'entreprise compile une liste noire qui consiste en des sites signalés par les utilisateurs via un bouton fourni sur la barre d'outils.

1.7.2.3 Google Safe Browsing

Google fournit le code source de la fonction de navigation sécurisée [Cranor et al., 2006] et affirme qu'il évalue les URL par rapport à une liste noire.

1.7.3 Mécanisme De Protection Par Mot De Passe

Un mot de passe est un mot secret ou une chaîne de caractères qui est utilisée pour l'authentification, afin de prouver l'identité ou d'obtenir l'accès à une ressource. Le mot de passe doit être gardé secret pour ceux qui ne sont pas autorisés à y accéder. Ainsi, la principale préoccupation de tout utilisateur est de protéger son mot de passe. Le mot de passe peut être craqué par des attaques telles que : attaque par force brute, attaque par dictionnaire, attaque par hameçonnage, etc. Un autre problème concernant le mot de passe est le problème du mot de passe unique, où l'utilisateur utilise un seul mot de passe pour les sites vulnérables et les sites financiers.Les pirates peuvent s'introduire dans les sites vulnérables qui stockent simplement le nom d'utilisateur et le mot de passe et appliquer ces combinaisons de nom d'utilisateur et de mot de passe sur des sites de haute sécurité tels que les sites bancaires. Tous ces problèmes peuvent être résolus d'un seul coup en hachant le mot de passe principal en utilisant le nom de

domaine comme clé du côté client. Certaines des applications/outils qui utilisent cette puissante technique sont :

1.7.3.1 Compositeur De mots De Passe

Cette extension [Reddy et al., 2011] met une petite icône rose à gauche de la zone d'accès au mot de passe. Si l'on clique sur cette icône, le mot de passe est superposé à une entrée de remplacement, dans laquelle il est facile de fournir un mot de passe sûr et unique et sécurisé (mot de passe principal).

1.7.3.2 Hassapass

Hassapass [Reddy et al., 2011] génère mécaniquement des mots de passe robustes à partir d'un mot de passe principal et d'un paramètre comme le nom de domaine. La génération du mot de passe est réalisée dans cette même fenêtre du navigateur en JavaScript.

1.7.3.3 Générateur De Mots De Passe

Générateur de mots de passe [Reddy et al., 2011] récupère le nom d'hôte à partir de l'URL de la page Web et le mélange avec le mot de passe personnel à l'aide de la cryptographie MD5. Il obtient toujours le même résultat si le nom d'hôte et le mot de passe sont les mêmes mais n'obtiendra jamais ce résultat si les deux changent.

1.8 Conclusion

Nous avons présenté dans ce chapitre l'hameçonnage commençant par sa définition ensuit nous avons cité ses types, impacts et approches ainsi que les techniques qui ont été réalisés pour sa détection. Nous présenterons dans le chapitre suivant l'apprentissage profond.

Chapitre 2

APPRENTISSAGE PROFOND

2.1 Introduction

L'intelligence artificielle est une discipline scientifique qui étudie les méthodes de résolution de problèmes de complexité logique ou algorithmique élevée. Machine Learning Majeure en Intelligence Artificielle. Par conséquent, l'apprentissage en profondeur est un ensemble de méthodes d'apprentissage automatique qui tentent de modéliser des niveaux élevés d'abstraction de données grâce à des architectures articulées de diverses transformations non linéaires. Dans ce chapitre, nous allons d'abord definer les termes liés au machine learning, puis au deep learning et à ces techniques.

2.2 Apprentissage Automatique

Le « machine Learning », ou littéralement l'apprentissage automatique fait partie de l'une des approches de l'intelligence artificielle. C'est une discipline consacrée à l'analyse des données. Le but de cette discipline est de créer de la connaissance de manière automatique à partir de données brutes (échantillons). Cette connaissance (ou bien modèle) peut alors être exploitée pour prendre des décisions. En fonction de la nature des échantillons (labellisés ou pas), il existe trois types d'apprentissages [Murdoch et al., 2019] :

2.2.1 Apprentissage supervisé

Nous nous intéressons plus particulièrement à l'apprentissage supervisé [Berry, 2020] qu'est simplement une formalisation de l'idée d'apprendre à partir d'exemples. Dans l'apprentissage supervisé, l'apprenant (généralement un programme d'ordinateur) reçoit deux ensembles de données, un ensemble d'apprentissage et un ensemble de tests. L'idée est que l'apprenant « apprenne » à partir d'un ensemble d'exemples labellisés dans l'ensemble d'apprentissage afin qu'il

puisse identifier les exemples non-labellisé dans l'ensemble de test avec la plus grande précision possible. C'est-à-dire que le but de l'apprenant est de développer une règle, un programme ou une procédure qui classifie de nouveaux exemples (dans l'ensemble de test) en analysant des exemples qui lui ont déjà été attribués.

2.2.2 Apprentissage Semi-supervisé

Il existe deux façons d'aborder la résolution d'un problème. L'une consiste à rechercher systématiquement une solution et l'autre à trouver une solution sur place. L'apprentissage semi-supervisé [Salsabil et Amaria, 2017]consiste à utiliser à la fois des données étiquetées et non étiquetées. L'utilisation de données non étiquetées avec des données étiquetées est une meilleure façon de faire un apprentissage non supervisé que d'utiliser uniquement des données étiquetées. Une autre raison pour laquelle l'étiquetage des données est important est qu'il nécessite l'intervention d'un expert humain. Lorsque les ensembles de données deviennent très volumineux, il peut être difficile de les traiter efficacement. Cette situation est propice à un apprentissage semi-supervisé, qui ne nécessite que quelques étiquettes.

2.2.3 Apprentissage non supervisé

L'apprentissage non supervisé est un type d'IA dans lequel les données fournies au système ne sont ni étiquetées ni classées. Les algorithmes du système traitent les données sans formation préalable. La sortie est déterminée par les algorithmes utilisés pour l'encoder. La mise en œuvre d'une approche d'apprentissage non supervisé est une façon d'expérimenter l'intelligence artificielle [Berry, 2020]

2.3 Apprentissage Profond

L'apprentissage en profondeur est un sous-domaine de l'apprentissage automatique qui se concentre sur des algorithmes inspirés de la structure et de la fonction du cerveau, appelés réseaux de neurones artificiels. Le modèle est composé d'un ensemble de blocs simples d'un certain type, certains de ces blocs étant ajustables pour mieux prédire le résultat final. Dans l'apprentissage en profondeur, un modèle informatique peut apprendre 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 très élevée, surpassant parfois les performances humaines. Les modèles sont entraînés à l'aide d'un grand nombre d'ensembles de données étiquetés et d'architectures de réseaux neuronaux à plusieurs couches [Vorontsov et Sci, 2017].

2.4 Réseaux de neurones

Les méthodes d'apprentissage en profondeur utilisent des architectures de réseaux de neurones, c'est pourquoi les modèles d'apprentissage en profondeur sont souvent appelés réseaux de neurones profonds (DNN). Les premiers réseaux de neurones ont été développés au début des années 1940[Hu., 2019]. . L'idée est de créer un modèle de neurone humain qui fonctionne de la même manière qu'un vrai neurone. La fonction d'un neurone formel est de prendre une somme pondérée d'entrées et d'appliquer une fonction d'activation. Les poids sont appelés poids synaptiques et la fonction d'activation utilise un seuil : la sortie étant 1 si la somme pondérée dépasse le seuil et 0 sinon. On remarque que le seuil vaut b :

$$Y = \begin{cases} 1 & si \\ 0 & sinon \end{cases} \sum Wkik > bk \tag{2.1}$$

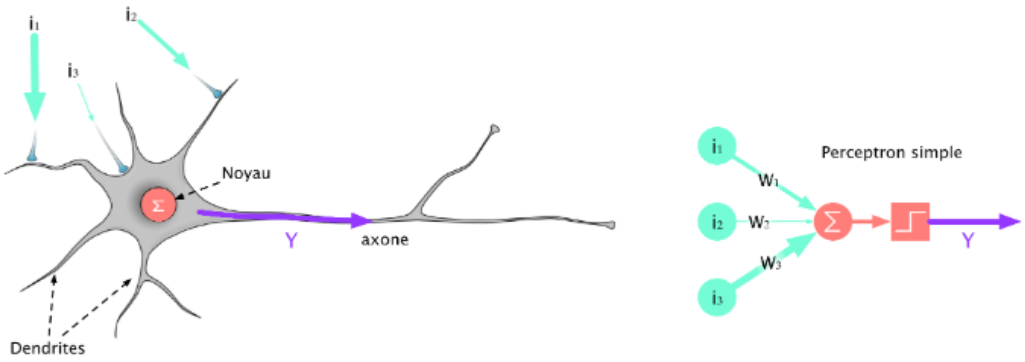

FIGURE 2.1 – Schéma d'un neurone à gauche et représentation d'un neurone formel à droite.

2.4.1 Réseaux de neurones convolutifs CNN

Le CNN [Sami et Hacene, 2020] est un type de modèle d'apprentissage profond pour le traitement des données qui ont un motif de grille, comme les images, qui est inspiré par l'organisation du cortex visuel animal et conçu pour apprendre automatiquement et de manière adaptative des hiérarchies spatiales de caractéristiques, de motifs de bas à haut niveau. Le CNN est une construction mathématique qui se compose généralement de trois types de couches (ou blocs de construction) : les couches de convolution, de mise en commun et les couches entièrement connectées. Les deux premières effectuent l'extraction des caractéristiques, tandis que la troisième transforme les caractéristiques extraites en résultats finaux, tels que la classification.

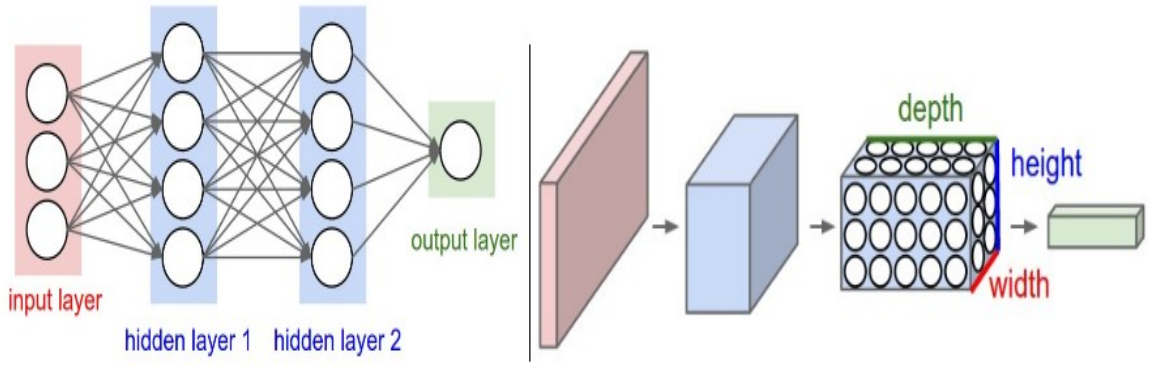

FIGURE 2.2 – À gauche : un réseau neuronal standard à 3 couches. À droite : un ConvNet organise ses neurones en trois dimensions [Sami et Hacene, 2020]

Les CNN utilisent trois principaux types de couches pour créer des architectures ConvNet :

2.4.1.1 Couche de convolution

L'objectif de l'opération de convolution [Sami et Hacene, 2020] est d'extraire les caractéristiques de haut niveau telles que les bords, la couleur, l'orientation du dégradé, etc. Classiquement, la première ConvLayer est responsable de la capture des caractéristiques de bas niveau. Avec des couches ajoutées, l'architecture s'adapte également aux fonctionnalités de haut niveau, nous donnant un réseau qui a une compréhension saine des images dans l'ensemble de données. L'élément impliqué dans la réalisation de l'opération de convolution dans la première partie d'une couche convolutive s'appelle le noyau/filtre, qui est une petite matrice utilisée pour le flou, la netteté, la détection des contours, etc. Une opération de point est effectuée entre une partie de l'image et le noyau.

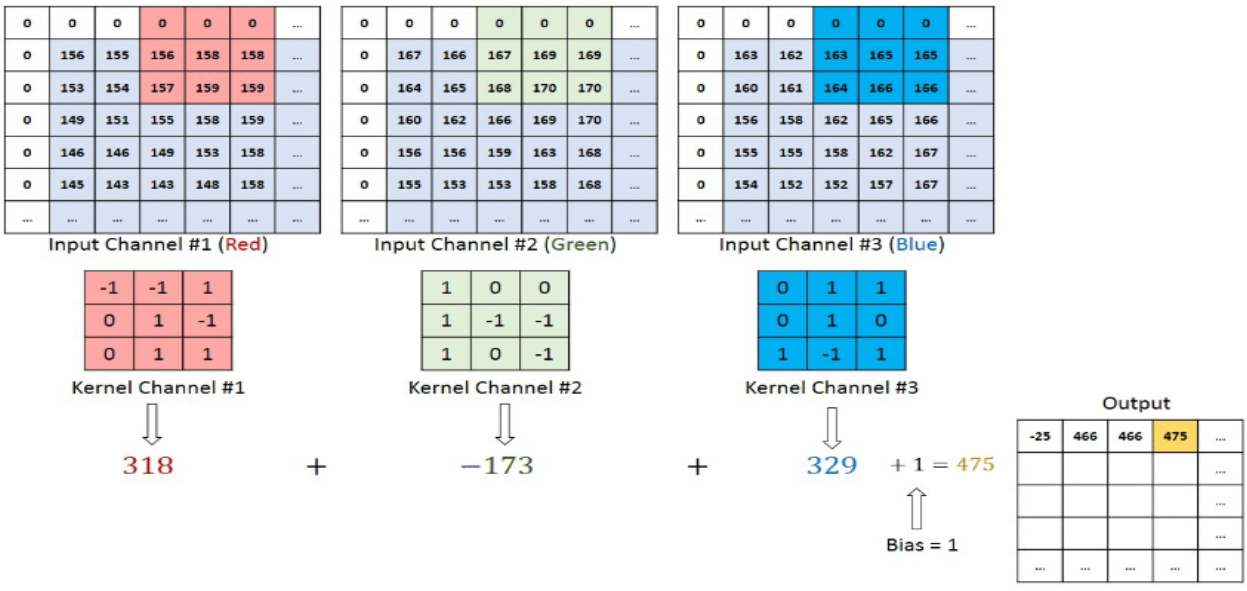

FIGURE 2.3 – Opération de convolution sur une matrice d'images M*N*3 avec un noyau 3x3x3

2.4.1.2 Couche mise en commun

Le but des couches de mise en commun [Scherer et al., 2010] est d'obtenir une invariance spatiale en réduisant la résolution des cartes de caractéristiques. Chaque carte de caractéristiques regroupées correspond à une carte de caractéristiques de la couche précédente. Il existe de nombreux types de regroupement : par exemple, le regroupement maximal et le sous-échantillonnage. Max pooling calcule le maximum dans le voisinage. D'autre part, le sous-échantillonnage renvoie la moyenne de toutes les valeurs de la partie de l'image couverte par le noyau .

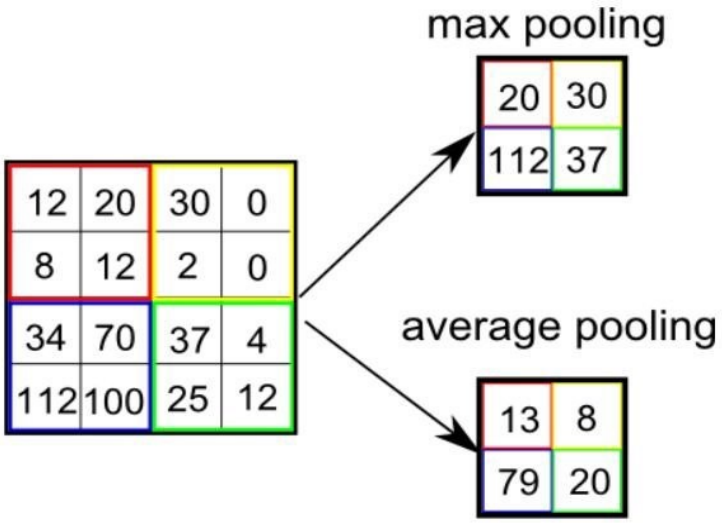

FIGURE 2.4 – Types de mise en commun.

2.4.1.3 Couche entièrement connectée

Exactement comme un réseau neuronal ordinaire, calculera les scores pour chaque classe de sortie.

2.4.2 Réseaux de neurones récurrents RNN

Les réseaux de neurones récurrents constituent l'algorithme de pointe pour les données séquentielles et sont utilisés par Siri d'Apple et la recherche vocale de Google. Il s'agit du premier algorithme qui se souvient de son entrée, grâce à une mémoire interne, ce qui le rend parfaitement adapté aux problèmes d'apprentissage automatique impliquant des données séquentielles. Il s'agit de l'un des algorithmes à l'origine des réalisations étonnantes observées dans le domaine de l'apprentissage profond au cours des dernières années.

Les RNN sont un type de réseau neuronal puissant et robuste, et font partie des algorithmes les plus prometteurs en usage car c'est le seul à disposer d'une mémoire interne.

Comme de nombreux autres algorithmes d'apprentissage profond, les réseaux de neurones récurrents sont relativement anciens. Ils ont été créés dans les années 1980, mais ce n'est qu'au cours des dernières années que nous avons vu leur véritable potentiel. L'augmentation de la puissance de calcul, les quantités massives de données avec lesquelles nous devons désormais travailler et l'invention de la mémoire à long terme (LSTM) dans les années 1990 [Aytug, 2022] ont réellement mis les RNN au premier plan.

Grâce à leur mémoire interne, les RNN peuvent se souvenir d'éléments importants concernant les données qu'ils ont reçues, ce qui leur permet de prédire très précisément ce qui va se passer. C'est pourquoi ils sont l'algorithme préféré pour les données séquentielles telles que les séries temporelles, la parole, le texte, les données financières, l'audio, la vidéo, la météo et bien plus encore. Les réseaux neuronaux récurrents peuvent acquérir une compréhension beaucoup plus profonde d'une séquence et de son contexte que les autres algorithmes [Aytug, 2022].

2.4.2.1 Fonctionnement du RNN

Les réseaux de neurones artificiels [Aytug, 2022] sont constitués de composants de traitement de données interconnectés, conçus de manière modulaire pour fonctionner comme le cerveau humain. Les neurones artificiels d'un réseau de neurones sont constitués de couches de cellules capables de traiter les entrées et d'envoyer les sorties à d'autres cellules du réseau.. Les nœuds sont reliés par des liens ou des poids qui affectent la force d'un signal et la sortie finale du réseau. Certains réseaux de neurones artificiels traitent les informations dans un seul sens, de l'entrée vers la sortie, appelé "feedforward". Ces réseaux de propagation vers l'avant comprennent les réseaux de neurones convolutifs qui sous-tendent les systèmes qui reconnaissent les images. Alors que les réseaux de neurones récurrents sont constitués de couches qui traitent les informations dans les deux sens, les réseaux de neurones récurrents (RNN) sont construits avec des couches qui traitent les informations dans une seule direction.

Comme les réseaux de neurones à propagation directe, les RNN peuvent déplacer des données de l'entrée initiale vers la sortie finale. Contrairement aux autres réseaux, ils utilisent des boucles de rétroaction, telles que la rétropropagation dans le temps (BPTT), tout au long du calcul afin de réinjecter des informations dans le réseau. Étant donné que les données d'entrée sont connectées les unes aux autres, les RNN peuvent traiter des données séquentielles et temporelles [Aytug, 2022].

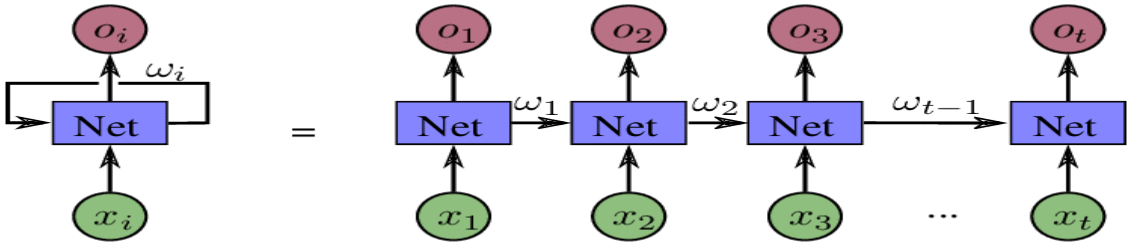

FIGURE 2.5 – Architecture de RNN [Tai et al., 2016].

2.4.2.2 Deux standard problèmes des RNN

Il existe deux obstacles majeurs auxquels les RNN ont dû faire face, mais pour les comprendre, on doit d'abord savoir ce qu'est un gradient. Un gradient est une dérivée partielle par rapport à ses entrées. Il mesure combien la sortie d'une fonction change si vous modifiez un peu les entrées. Le gradient considéré comme la pente d'une fonction. Plus le gradient est élevé, plus la pente est forte et plus le modèle peut apprendre rapidement. Mais si la pente est nulle, le modèle cesse d'apprendre. Un gradient mesure simplement l'évolution de tous les poids par rapport à l'évolution de l'erreur [Aytug, 2022].

2.4.2.2.1 Gradients explosifs On parle de gradients explosifs lorsque l'algorithme, sans grande raison, attribue une importance stupidement élevée aux poids. Ce problème peut être facilement résolu en tronquant ou en écrasant les gradients [Aytug, 2022].

2.4.2.2.2 Gradients évanescents Les gradients évanescents se produisent lorsque les valeurs d'un gradient sont trop faibles et que le modèle cesse d'apprendre ou prend beaucoup de temps en conséquence. Il s'agissait d'un problème majeur dans les années 1990 [Aytug, 2022], beaucoup plus difficile à résoudre que les gradients explosifs.Il a été résolu grâce au concept de LSTM par Sepp Hochreiter et Juergen Schmidhuber .

2.4.3 Réseau convolutif temporel TCN

Les réseaux de neurones artificiels constituent actuellement un domaine scientifique au développement très dynamique. Chaque jour, nous sommes témoins de nouvelles applications des réseaux neuronaux. Le réseau convolutif temporel [Lara-benitez et al., 2020] est un modèle de réseau de neurones qui emploie des convolutions et des dilatations occasionnelles afin qu'il soit adaptatif pour les données séquentielles avec sa temporalité et ses grands champs réceptifs.

L'architecture a été proposée par [Bai et al., 2018] et montre de grandes performances sur des tâches de séquence à séquence.

2.4.3.1 Convolutions causales

Le TCN présente deux caractéristiques distinctives : (1) les convolutions sont causales [Lara-benitez et al., 2020], ce qui signifie que la sortie actuelle se rapporte uniquement aux entrées actuelles et historiques, et non aux entrées futures. (2) l'architecture est capable de prendre une série temporelle de n'importe quelle longueur et de la mettre en correspondance avec une donnée de sortie de même longueur, tout comme un RNN. Pour satisfaire la première caractéristique, la première couche du TCN est un réseau entièrement convolutif unidimensionnel, où chaque couche intermédiaire est de la même taille que la couche d'entrée, et où des paddings de taille nulle sont ajoutés pour que les couches suivantes soient de la même taille que les précédentes. Pour satisfaire la deuxième caractéristique, le TCN ne convolue que l'entrée du temps actuel et du temps précédent. Pour obtenir une longue taille d'historique effective, de grands filtres ou une structure extrêmement profonde sont nécessaires, ce qui entraîne un énorme travail de calcul. Par conséquent, les convolutions dilatées sont utilisées dans le TCN pour permettre à la fois des réseaux très profonds et des historiques effectifs très longs.

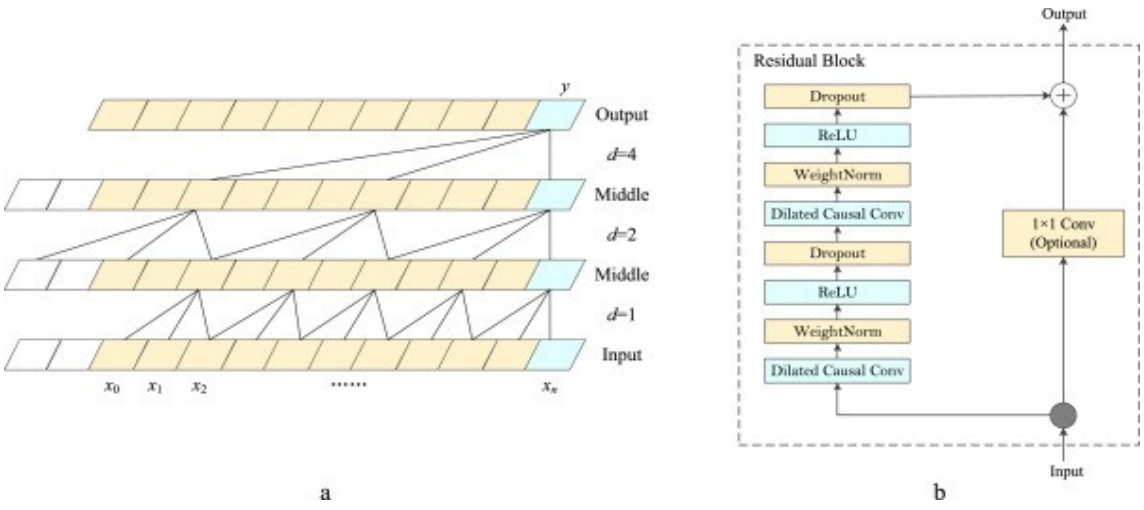

FIGURE 2.6 – (a)Une convolution causale dilatée ;(b)Bloc résiduel TCN.Convolutions dilatées [zhu et al., 2020].

2.4.3.2 Convolutions dilatées

Pour appliquer la convolution causale [zhu et al., 2020] à des séries temporelles ayant un long historique, TCN choisit d'utiliser des convolutions dilatées qui permettent d'obtenir un champ réceptif de taille exponentielle. Comme le montre la figure 2.6(a), la convolution dilatée ajoute certains poids au noyau de convolution et rend les données d'entrée inchangées, de manière à

augmenter la taille des séries temporelles observées par le réseau alors que la quantité de calcul reste fondamentalement inchangée. Formellement, pour une série temporelle unidimensionnelle X et un filtre f, l'opération F de convolution dilatée sur les éléments de la série temporelle est définie comme suit [zhu et al., 2020] :

$$F(s) = (x * f)(s) = \sum k - 1i = 0f(i)xs - di \tag{2.2}$$

Où est l'opération convolutive. K est la taille du filtre et d est le facteur de dilatation. s -di explique la direction du passé. Par conséquent, la dilatation est égale à l'utilisation d'une taille de pas fixe entre chaque deux prises de filtre adjacentes. Lorsque d = 1, la convolution étendue est réduite à une convolution régulière. Avec une expansion plus importante, la sortie de la couche supérieure peut représenter un plus large éventail d'entrées, ce qui élargit efficacement le champ réceptif de TCN. Le TCN dispose de deux façons d'élargir le champ réceptif : en augmentant le facteur de dilatation d et en choisissant des tailles de filtre k plus grandes.

2.4.3.3 Connexions résiduelles

Le réseau résiduel [Lara-benitez et al., 2020] est utilisé pour résoudre le problème de la dégradation des performances du réseau neuronal à convolution lorsque le nombre de couches est très profond, la superposition de convolutions causales et de convolutions dilatées rend le nombre de couches du réseau neuronal progressivement plus profond. Pour Afin d'éviter l'atténuation ou la disparition du gradient au cours du processus de formation, des connexions résiduelles sont introduites dans la couche de sortie du TCN. La couche de sortie du TCN, et l'entrée X est fusionnée dans la sortie du réseau convolutif comme suit :

$$o = Activation(x + F(x)) \tag{2.3}$$

Où F(x) est la sortie de la couche convolutive et Activation (-) est la fonction d'activation. Les connexions résiduelles permettent effectivement à l'apprentissage des couches de modifier le mappage d'identité plutôt que toute la transformation, ce qui a été transformation, ce qui s'est avéré bénéfique à de nombreuses reprises pour les réseaux très profonds. Le bloc résiduel pour TCN est illustré à la Figure 2.6(b). Dans un bloc résiduel, le TCN a deux convolutions causales dilatées et des non-linéarités, de sorte que la fonction d'unité linéaire rectifiée (Relu) est utilisée comme fonction d'activation. La normalisation du poids est utilisée pour normaliser les filtres convolutifs. En outre, une couche d'exclusion est ajoutée après chaque convolution dilatée pour éviter le surajustement.

Les réseaux TCN sont bien adaptés aux tâches de séquence à séquence. En raison de leur construction :

- **Parallélisme :**[Lara-benitez et al., 2020] Contrairement aux RNN où les prédictions pour les étapes temporelles ultérieures doivent attendre que leurs prédécesseurs aient terminé, les convolutions peuvent être calculées en parallèle car le même filtre est utilisé dans chaque couche. Par conséquent, lors de la formation et de l'évaluation, une longue séquence d'entrée peut être traitée dans son ensemble, au lieu d'être traitée séquentiellement comme dans les RNN.
- **Taille flexible du champ réceptif :** [Lara-benitez et al., 2020] La taille du champ réceptif peut être modifiée de plusieurs façons. Par exemple, l'empilement de couches convolutionnelles plus dilatées, l'utilisation de facteurs de dilatation plus importants ou l'augmentation de la taille du filtre sont autant d'options viables. Ainsi, les TCN permettent un meilleur contrôle de la taille de la mémoire du modèle, et sont faciles à adapter à différents domaines.
- **Faible besoin en mémoire pour l'apprentissage :** [Lara-benitez et al., 2020] En particulier dans le cas d'une longue séquence d'entrée, les LSTM et les GRU peuvent facilement utiliser beaucoup de mémoire pour stocker les résultats partiels de leurs multiples portes cellulaires. En revanche, dans les TCN, les filtres sont partagés sur une couche, le chemin de rétropropagation ne dépendant que de la profondeur du réseau.

2.4.4 Word Embedding

Word Embedding [Li et al., 2018] Sont en fait une catégorie de techniques dans lesquelles les mots individuels sont représentés comme des vecteurs à valeurs réelles dans un espace vectoriel prédéfini. Chaque mot est mis en correspondance avec un vecteur et les valeurs vectorielles sont apprises d'une manière qui ressemble à un réseau neuronal, d'où le fait que la technique est souvent classée dans le domaine de l'apprentissage profond. La clé de cette approche est l'idée d'utiliser une représentation distribuée dense pour chaque mot. Chaque mot est représenté par un vecteur à valeur réelle, souvent de plusieurs dizaines ou centaines de dimensions. Cela contraste avec les milliers ou les millions de dimensions requises pour les représentations de mots éparses, telles qu'one hot encoding.

La représentation distribuée est apprise sur la base de l'utilisation des mots. Cela permet aux mots qui sont utilisés de manière similaire d'avoir des représentations similaires, capturant naturellement leur signification. Cela peut être comparé à la représentation claire mais fragile d'un modèle de sac de mots où, à moins d'être explicitement gérés, différents mots ont des

représentations différentes, indépendamment de la façon dont ils sont utilisés. Nous allons passer en revue deux techniques qui peuvent être utilisées pour apprendre l'intégration de mots à partir des URL [Li et al., 2018].

2.4.4.1 Word2Vec

Word2Vec est une méthode statistique permettant d'apprendre efficacement un word Embedding autonome à partir d'un corpus de textes. Elle a été développée par [Mikolov et al., 2013]. Chez Google en 2013 afin de rendre plus efficace l'apprentissage de l'incorporation basé sur un réseau de neurones. Depuis lors, elle est devenue la norme de facto pour le développement d'une incorporation de mots pré-entraînée.

Deux modèles d'apprentissage différents ont été présentés, qui peuvent être utilisés dans le cadre de l'approche word2vec [Li et al., 2018]pour apprendre l'intégration des mots :

- **Le modèle Continuous Bag-of-Words, ou CBOW.**
- **Le modèle Continuous Skip-Gram.**

Le modèle CBOW apprend l'enchâssement en prédisant le mot actuel sur la base de son contexte. Le modèle de saut de gramme continu apprend en prédisant les mots environnants à partir d'un mot courant. Le modèle de saut de gramme continu apprend en prédisant les mots environnants à partir d'un mot courant.

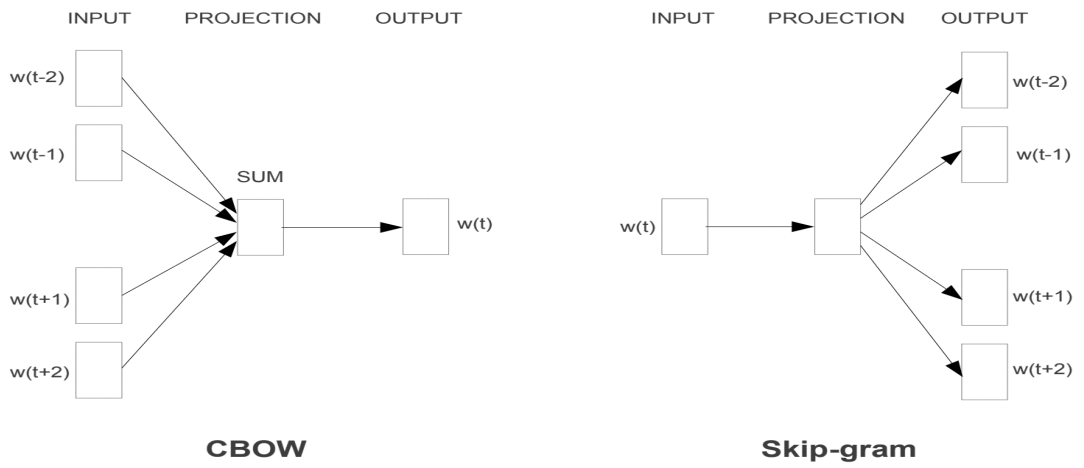

FIGURE 2.7 – Modèles de formation Word2Vec.

Le principal avantage de l'approche est que les incorporations de mots de haute qualité peuvent être apprises efficacement (faible complexité en espace et en temps), ce qui permet d'apprendre des incorporations plus importantes (plus de dimensions) à partir de corpus de textes beaucoup plus grands (des milliards de mots).

2.4.4.2 GLOVE

L'algorithme Global [Li et al., 2018] Vectors for Word Representation, ou GloVe, est une extension de la méthode word2vec pour l'apprentissage efficace des vecteurs de mots, développée par Pennington et al. à Stanford. Les représentations classiques des mots par des modèles d'espace vectoriel ont été développées à l'aide de techniques de factorisation matricielle telles que l'analyse sémantique latente (LSA), qui utilisent bien les statistiques globales du texte mais ne sont pas aussi efficaces que les méthodes d'apprentissage comme word2vec pour capturer le sens et le démontrer dans des tâches telles que le calcul d'analogies. GloVe est une approche qui marie les statistiques globales des techniques de factorisation matricielle comme LSA avec l'apprentissage local basé sur le contexte dans word2vec. Plutôt que d'utiliser une fenêtre pour définir le contexte local, GloVe construit une matrice explicite de cooccurrence de mots ou de contexte de mots en utilisant des statistiques sur l'ensemble du corpus textuel. Le résultat est un modèle d'apprentissage qui peut aboutir à des incorporations de mots généralement meilleures.

2.5 Conclusion

Nous avons présenté dans ce chapitre l'apprentissage automatique ainsi que ses différents types. Nous avons exploré également les différentes techniques d'apprentissage profond afin de les utiliser pour trouver une solution fiable à notre problématique. Nous allons voir dans le chapitre suivant les travaux connexes pour la détection des URLs de phishing.

Chapitre 3

TRAVAUX CONNEXES

3.1 Introduction

Le phishing présente une tendance de développement diversifiée, qui pose de nouveaux défis en matière de détection. Bien que les hameçonneurs soient pernicieux et se cachent, les experts en sécurité et les chercheurs ont consacré de nombreuses techniques à la détection des sites de phishing. Actuellement, les méthodes populaires de détection des URL malveillants comprennent principalement les listes noires, l'apprentissage automatique, similitude visuelle, les heuristiques et les méthodes de détection basées sur l'apprentissage profond. Dans ce chapitre nous allons présenter les différents travaux faits avec ces déférentes techniques pour la détection des URLs de phishing.

3.2 Techniques de détection d'hameçonnage

L'anti-hameçonnage [Patil et Dhage, 2019] désigne la méthode employée pour détecter et prévenir les attaques de phishing. L'anti-hameçonnage protège les utilisateurs contre l'hameçonnage. De nombreux travaux ont été réalisés sur l'anti-hameçonnage en concevant diverses techniques d'anti-hameçonnage. Quelques techniques fonctionnent sur les courriels, quelques-unes sur les attributs des sites Web et quelques-unes sur l'URL des sites Web. Beaucoup de ces techniques permettent aux clients de reconnaître et d'éliminer de nombreux types d'attaques d'hameçonnage. En général, les techniques anti-hameçonnage peuvent être classées dans les catégories suivantes :

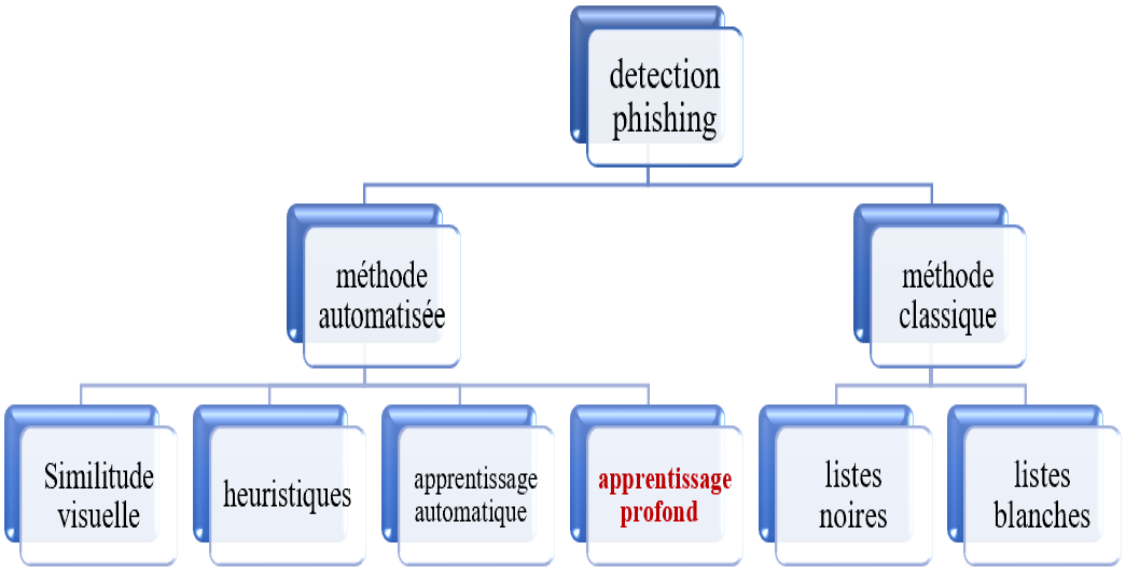

FIGURE 3.1 – Types des Techniques anti-phishing.

3.2.1 Détection du phishing par les listes noires

Une approche de la lutte contre le phishing basée sur une base de données, appelée liste noire, a été développée à partir de plusieurs initiatives de recherche [Sheng et al., 2010]. Cette méthode est totalement basée sur l'utilisation d'une liste prédéfinie contenant des noms de domaine ou des URL de sites web reconnus comme nuisibles. Un site Internet figurant sur une liste noire peut également perdre jusqu'à 95% de son trafic normal, dans le but d'entraver la capacité du site web à générer des revenus et, éventuellement, des revenu [Aburrous et al., 2010]. C'est la raison principale pour laquelle les administrateurs de sites et les administrateurs du réseau accordent une grande attention au problème de la liste noire. Selon [Mohammad et al., 2015] il existe deux types de listes noires en matière de sécurité informatique :

3.2.1.1 Listes noires domaines/URL

Basés sur les domaines/URL[Patil et Dhage, 2019], il s'agit de listes d'URL en temps réel qui intègrent domaines malveillants et qui recherchent généralement des URL malveillant.

3.2.1.2 Liste noire internet protocole

Basée sur le protocole Internet. Il s'agit de listes noires d'URL ou de serveurs de domaines en temps réel qui incluent les adresses IP qui, échangent leur statut, Les sociétés de messagerie, dont Yahoo, vérifient régulièrement les listes noires de serveurs de domaines. Par exemple, pour évaluer si le serveur d'envoi est géré par une personne qui permet à d'autres utilisateurs

d'envoyer des messages à partir de son propre serveur. Permet à d'autres utilisateurs d'envoyer à partir de leur propre source.

Les utilisateurs, les agences ou les organisations de logiciels informatiques peuvent créer des listes noires. Chaque fois qu'un site web est prêt à être consulté, le navigateur évalue l'URL dans la liste noire. Si l'URL figure dans la liste noire, une action certaine est prise pour avertir le consommateur de la possibilité d'une violation de la sécurité. Dans le cas contraire, aucune action n'est entreprise car l'URL du site Internet n'est pas reconnue comme dangereuse. Centaines de listes noires qui sont publiquement disponibles, parmi lesquelles nous mentionnerons la liste noire ATLAS d'Arbor Networks, BLADE Malicious URL analysis, la liste de la DGA, la liste des bogues CYMRU, la liste Scumware.org la liste OpenPhish, la liste noire de Google et la liste noire de Microsoft.

Au motif que tout utilisateur ou toute entreprise, petite ou grande, peut créer des listes noires, les listes noires actuellement disponibles au public ont des niveaux uniques d'efficacité de protection, en particulier en ce qui concerne les deux tiers de la protection :

- Les instances de mise à jour de la liste noire et sa disponibilité régulière.
- Les effets de haute qualité en ce qui concerne la détection correcte du phishing.

Les listes noires sont souvent stockées sur des serveurs, mais peuvent aussi être disponibles localement dans une machine informatique [Mohammad et al., 2014] Ainsi, le processus de vérifie si une URL fait partie de la liste noire est exécuté lorsqu'un site web est sur le point d'être visité par l'utilisateur. Auquel cas le serveur ou la machine locale utilise une méthode de recherche particulière pour vérifier le processus et en déduire une action. La liste noire est généralement mise à jour périodiquement. Par exemple, la liste noire de Microsoft est normalement mise à jour tout es les neuf heures à six jours, tandis que la liste noire de Google est mise à jour toutes les vingt heures à douze jours [Mohammad et al., 2014]. Par conséquent, la fenêtre de temps nécessaire pour modifier la liste noire en incluant de nouvelles malveillantes, ou en excluant d'éventuelles URL faussement positives, pourrait permettre aux hameçonneurs de lancer et de réussir leurs attaques de phishing. En d'autres termes, les hameçonneurs disposent de beaucoup de temps pour lancer une attaque de phishing avant que leurs sites d'hameçonnage ne soient bloqués.(Abdelhamid et al., 2014).Une autre étude de l'APWG a révélé que plus de 75% des domaines de phishing ont véritablement servi des sites web légitimes et, lorsqu'ils sont bloqués, cela implique que plusieurs sites Web dignes de confiance seront ajoutés à la liste noire, ce qui entraînera une réduction drastique des revenus du site web et nuit à sa réputation [Activity et Report, 2015].

3.2.2 Détection du phishing par Liste Blanche

Il y a eu quelques tentatives pour examiner le développement de listes blanches, c'est-à-dire des bases de données d'URL valides, en évaluation des listes noires [Shekokar et al., 2015]. Malheureusement, étant donné que la majorité des sites web nouvellement créés sont d'abord diagnostiqués comme " suspects ", la technique de la liste blanche s'en trouve alourdie. Pour triompher de ce problème, les sites web que l'on prévoit d'être visité par la personne doivent figurer sur la liste blanche. Complexe dans la pratique en raison du grand nombre de sites viables qu'une personne peut parcourir. La méthode de la liste blanche est en réalité peu pratique si l'on considère que le fait de "savoir" à l'avance ce que les clients recherchent surfent pourraient être exclusifs aux sites réellement visités au cours de la navigation. La décision humaine est une procédure dynamique, les utilisateurs échangent leurs idées et se mettent à surfer sur de nouveaux sites web qu'ils n'avaient pas l'intention de visiter au départ.

L'une des premières listes blanches développées a été proposée par [Shekokar et al., 2015], qui s'est transformée en une liste entièrement basée sur la navigation des utilisateurs sur des sites web de confiance. La liste blanche surveille les tentatives de connexion de l'utilisateur et si une connexion répétée a été effectuée avec succès, cette approche invite la personne à d'insérer ce site Internet dans la liste blanche. Un problème évident de l'approche de Chen et Guo est qu'elle suppose que les utilisateurs gèrent des sites Web fiables, ce qui, hélas, n'est pas le cas.

PhishZoo est une approche de liste blanche proposée par [Afroz et Greenstadt,2011]. Cette technique utilise une technique de hachage flou pour créer un profil de site Web, qui utilise un certain nombre de normes pour différencier un site Web de tous les autres. Cela inclut les images, le code source HTML, les URL et les certificats SSL.

3.2.3 Détection de phishing par Similitude visuelle

La méthode de détection basée sur la similarité visuelle doit prendre un instantané de la page Web, nécessite des ressources de calcul et de stockage importantes et détecte principalement le site Web de phishing avec une visualisation de page similaire.

1-[Rao et Ali., 2015] ont proposé une méthode pour juger du type de site Web en comparant la similitude visuelle entre les sites Web de phishing et les sites Web sans phishing. La méthode utilise l'arborescence HTML DOM pour segmenter la page en fonction de « repères visuels », puis utilise trois métriques d'évaluation pour évaluer la similitude visuelle entre le site à tester et le site légitime : similitude au niveau du bloc, similitude de la mise en page et similitude globale du style de pages Web. La méthode peut détecter le phishing avec un faible taux de fausses détections, mais elle prend du temps. De plus, cela dépend en grande partie des résultats

de la segmentation des pages Web. A la différence de cela. (Salve et al., 2015) ont proposé une méthode pour détecter les sites Web de phishing en détectant la méthode de similarité des éléments clés liée aux fichiers CSS.

2-[Shekokar et al., 2015] ont proposé une méthode de détection basée sur la similarité des URL et des pages Web. Ils ont proposé l'algorithme LinkGuard pour déterminer si une URL est suspecte et ont utilisé une approche de correspondance de pages basée sur des images pour obtenir une similitude entre les pages cibles et les pages des sites Web de phishing. Ensuite, un seuil est utilisé pour détecter si la page cible est une page de phishing.

3-[Chiew et al., 2018] ont proposé Phishdentity qui utilise le favicon extrait du site Web et utilise Google comme moteur de recherche d'images pour découvrir les tentatives potentielles de phishing. Il ne nécessite pas d'analyse intensive du contenu textuel ou image, et augmente ainsi la vitesse de détection.

3.2.4 Détection du phishing par heuristique

Les méthodes de détection heuristiques sont basées sur la similarité entre les pages de phishing, les caractéristiques statistiques ou les connaissances préalables des experts. Il extrait plusieurs fonctionnalités des pages de phishing détectées et les généralise en un ensemble de fonctionnalités heuristiques. La détection des attaques de phishing est ensuite mise en œuvre sur la base de ces caractéristiques.

- **CANTINE :** [Xiang et al., 2011] ont proposé une méthode de détection de phishing basée sur l'heuristique nommée CANTINA. Il utilise un moteur de recherche Google pour récupérer des mots-clés et des noms de domaine dans une page Web et détermine si la page est légitime en fonction des résultats renvoyés par la recherche et d'autres fonctionnalités heuristiques.
- **PHISHNET :** [Aljofey et al., 2020] ont proposé PhishNet, qui énumère l'URL simple du site Web de phishing sur la base de cinq règles heuristiques. [Shahriar et Zulkernine, 2012] ont testé la crédibilité de sites Web suspects pour déterminer s'il s'agissait d'un site de phishing.
- **PDA :** [Jain et Gupta, 2016] ont proposé l'algorithme de détection de phishing (PDA) pour déterminer si une URL suspecte est un site Web de phishing. PDA détermine principalement si une URL est légale en calculant le nombre d'hyperliens dans la page Web suspecte.

3.2.5 Détection du phishing par apprentissage automatique

L'un des principaux problèmes rencontrés par la détection heuristique est qu'elle n'est pas assez flexible pour s'adapter aux changements de site de phishing. Même des modifications mineures pourraient entraîner le contournement de cette détection. Par conséquent, le modèle heuristique a donné de la flexibilité en utilisant des techniques d'apprentissage automatique pour s'adapter aux changements. Dans cette technique, les ensembles de données sont préparés pour entraîner le modèle d'apprentissage automatique, et l'ensemble de données représente les valeurs des caractéristiques extraites à l'aide d'une approche heuristique. Certains des algorithmes utilisés sont Support Vector Machine Decision Tree (SVM-DT), Random Forest (RF), Sequential Minimum Optimization (SMO), Principal Component Analysis Random Forest, J48 tree, Multilayer Perceptron, etc. Ces algorithmes peuvent détecter même zéro-day attaques de phishing lorsqu'elles sont entraînées avec des fonctionnalités de modèle heuristique. Si les données d'entraînement sont vastes, ces algorithmes sont encore meilleurs car ils apprennent la plupart des variations possibles des sites de phishing. [Kavya, 2018] ont atteint une précision d'environ $99,5\%$ dans la détection des sites de phishing à l'aide de techniques d'apprentissage automatique. Aussi, selon le récent sondage [Khonji et al., 2013], la détection des sites de phishing avec une précision de plus de 99% pourrait être réalisée à l'aide de techniques d'apprentissage automatique. Les performances de l'algorithme d'apprentissage automatique dépendent de la taille des données d'entraînement, de la qualité des caractéristiques extraites et des valeurs de certains hyperparamètres utilisés pour optimiser la précision.

3.2.6 Détection de phishing par l'apprentissage profond

L'apprentissage en profondeur est une technique d'apprentissage automatique qui permet aux machines d'apprendre des fonctionnalités directement à partir des données. Les données peuvent être n'importe quelle forme d'information. L'apprentissage en profondeur nécessite une grande quantité de données étiquetées et permet à l'unité de traitement graphique (GPU) de former des réseaux profonds en moins de temps. Nouvelle des tendances ont été faites pour exploiter les techniques de réseau neuronal profond (DNN) telles que le réseau feed forward multicouche [Ningxia, 2011], Réseau de neurones convolutifs (CNN) [Le et al., 2018] et le réseau neuronal récurrent (RNN) [Bahnsen et al., 2017] pour détecter et empêcher les attaques de phishing. Ces réseaux sont entraînés à l'aide d'ensembles de données multifonctions obtenus à l'aide de méthodes heuristiques. [Bahnsen et al., 2017] ont entraîné le RNN sur la séquence de caractères URL. Ils ont fait valoir que chaque séquence de caractères a des corrélations, les caractères voisins de l'URL sont susceptibles d'être connectés. Ces modèles séquentiels sont importants

car ils peuvent être utilisés pour améliorer les performances des prédicteurs. [Le et al., 2018] ont utilisé CNN pour apprendre le comportement séquentiel des URL. Ils ont adopté deux techniques qui sont le niveau de caractère CNN et le niveau de mot CNN, qu'identifier des caractères et des mots uniques. Chaque caractère ou mot est représenté sous forme de vecteur et entraîne les vecteurs sur CNN à apprendre le comportement séquentiel de l'URL pour identifier les URL de phishing.

3.3 Travaux Connexes du détection de phishing URL par l'apprentissage profond

La croissance rapide et les progrès de phishing et ses techniques posent un défi majeur pour la sécurité Web [Rofifah, 2020] a proposé une technique basée sur SVM pour détecter les URL de phishing. Les caractéristiques utilisées sont la structure, le mot et les noms de marque présents dans l'URL.

[Wei et al., 2020] ont proposé une méthode d'identification des sites Web de phishing basée uniquement sur le texte de l'adresse URL par un réseau neuronal profond avec des couches convolutives. Ils ont codé les URL en tant que vecteurs de niveau caractère à un coup et les ont présentés comme entrées à un réseau neuronal convolutif. Ils ont montré que l'analyse sans dictionnaire des URL offre une sécurité précise à près de 100% . La détection précise basée sur les URL offre une défense de type (zero day) . Le détecteur de phishing entraîné est petit et rapide et peut être utilisé sans effort sur des appareils mobiles.

[Rasymas et Dovydaitis, 2020] ont proposé une méthode de détection des sites de phishing en utilisant uniquement leurs URLs. Ils ont appliqué une architecture de réseau neuronal profond sur un ensemble de données donné, la classification des URL de phishing et des URL bénignes était effectuée avec une précision de $94,4\%$. Différentes combinaisons de caractéristiques (lexicales, incorporées au niveau des caractères, incorporées au niveau des mots, incorporées au niveau des caractères et des mots, lexicales et incorporées au niveau des caractères et des mots) ont été évaluées. Les résultats ont montré que la meilleure précision est obtenue en utilisant le modèle d'intégration des caractères et des mots.

Afin de détecter efficacement les URL de phishing,[Huang et al., 2019] ont proposé PhishingNet, un cadre basé sur l'apprentissage profond composé d'un module CNN et d'un module RNN hiérarchique basé sur l'attention. Ils ont utilisé deux modules parallèles pour extraire les représentations des caractéristiques des URL, dont l'un est un module CNN au niveau des caractères et l'autre un module RNN hiérarchique basé sur l'attention. Ils ont obtenu une précision de 0,98 et accuracy de 0,97.

[Bahnsen et al., 2017]ont utilisé deux méthodes pour étudier les URL : l'ingénierie des fonctions avec analyse lexicale et statistique et une nouvelle approche utilisant un réseau de neurones à mémoire. ils ont utilisés une base de données de 1 million d'URL légitimes et 1 million d'URL de phishing. Les resultats obtenus sont : le RF a obtenu un score F1 de 0,93 et une précision de $93,5\%$ .Le LSTM a atteint une précision de $98,7\%$ alors que le score F1 était de 0,98.

[Al-Ahmadi et Lasloum, 2020] ont implémenté un classificateur MPL avec deux couches cachées et 100 neurones. Ils ont utilisé deux types de données. Le premier groupe contient 11 055 URL légitimes, tandis que le second groupe contient 2 456 URL légitimes et 1 094 sont des URL de phishing. Trente et une caractéristiques ont été utilisées. Ils ont obtenu une précision de 0,96 pour le premier ensemble de données et de 0,95 pour le deuxième ensemble de données, et ont comparé leurs résultats avec d'autres algorithmes de classification, tels que KNN, SVM.

[Sujithra et al., 2020] ont cherché à détecter les URL malveillants à l'aide de caractéristiques minimales en appliquant des techniques d'apprentissage automatique approfondi. Les URL des pages Web sont introduites dans l'extracteur de caractéristiques. L'extracteur de caractéristiques extrait les caractéristiques requises des sources telles que les URL, les hyperliens et les tiers, puis les transfère à l'algorithme de classement des caractéristiques du gain d'information (IG). L'algorithme IG aide à choisir les caractéristiques les plus performantes. Les meilleures caractéristiques de performance sont à nouveau entraînées sur un réseau neuronal profond (DNN) pour déterminer l'état de la sortie et différencier les URL légitimes des URL de phishing. Ils ont obtenu une accuracy de $99,71\%$ et une accuracy de test de $99,13\%$. Enfin, ils ont obtenu une accuracy de formation de $99,90\%$ sur le réseau neuronal profond.

[Hai et Hwang, 2018] ont proposé une nouvelle approche utilisant la détection par apprentissage automatique pour classer les URL. Cette nouvelle approche dépend des caractéristiques du traitement du langage naturel en utilisant la représentation vectorielle du mot et des modèles qui ont appelé ngram comme les caractéristiques les plus critiques sur le mot de la liste noire. Les caractéristiques extraites des modèles de ngram, de la représentation vectorielle des mots et d'autres propriétés lexicales permettent d'établir une classification et des critères à l'aide de l'algorithme d'apprentissage automatique SVM (Support Vector Machine). Le nombre total de caractéristiques extraites du modèle word2vec est de 100. Cependant, sur 150 397 URL, 107 615 étaient bénignes, tandis que 42 782 étaient malveillantes. Le résultat montre que le SVM atteint un niveau de taux de précision de $97,1\%$ et un score F1 de 0,95 tout en maintenant le temps de classification de 0,01 seconde.

[Le et al., 2018] ont utilisé URLNet, une méthode d'apprentissage profond de bout en bout, pour percevoir les URL de phishing. Ils ont appliqué la méthode CNN aux caractères et aux mots de la chaîne d'URL pour former les URL en les intégrant dans un agenda mutuellement optimisé. Ils ont montré que leur modèle était plus performant que les modèles existants. Le modèle proposé peut échouer si les sites de phishing ont des URL très courtes.

[Sujithra et al., 2020] ont prévu un modèle de détection du phishing plus innovant en appliquant la technique de classification par réseau neuronal. En adoptant le principe de minimisation du risque de conception, le modèle a pu atteindre une très grande précision tout en ayant une excellente capacité de généralisation. Les auteurs ont utilisé l'algorithme de Monte Carlo pour la formation de leur modèle afin d'éviter le sur ajustement. Le modèle de détection qu'ils proposent peut atteindre une très grande précision et un faible FPR, ce qui indique l'excellence du modèle.

[Yi et al., 2018] ont proposé la détection du phishing de sites Web en appliquant un cadre d'apprentissage profond à l'aide d'ensembles de caractéristiques originales et d'interaction. Ils ont extrait les caractéristiques originales de l'analyse des URL et les caractéristiques d'interaction des codes sources des sites Web. Ensuite, ils ont appliqué un réseau de croyance profond (DBN) aux caractéristiques extraites, obtenant une précision satisfaisante.

[Sahingoz et al., 2019] ont appliqué une technique de reconnaissance du phishing basée sur l'apprentissage automatique à partir d'URL. Ils ont utilisé différents types de sept algorithmes de classification ainsi que des caractéristiques créées par le traitement du langage naturel. Les résultats expérimentaux ont démontré que l'algorithme Random-Forest (RF) avec des caractéristiques basées sur le traitement du langage naturel (NLP) est capable d'atteindre une précision très élevée de $97,98\%$ pour détecter les URL de phishing.

[Aljofey et al., 2020] ont appliqué un modèle du réseau neuronal convolutif au niveau des caractères pour détecter le phishing basé sur les URL de sites Web. Le modèle proposé utilise des fonctionnalités de modèle séquentiel pour classer les URL légitimes. Divers ensembles de fonctionnalités, telles que Des fonctionnalités telles que les fonctionnalités conçues, le TFIDF au niveau des caractères, l'intégration des caractères et les vecteurs de comptage au niveau des caractères ont été évalués et comparés à l'aide de méthodes d'apprentissage automatique traditionnelles et d'apprentissage automatique en profondeur. Le modèle a atteint une précision de plus de 95% sur les ensembles de données sélectionnés et de plus de $98,5\%$ sur les ensembles de données de référence.

TABLE 3.1 – Une comparaison entre les travaux connexes de certaines applications d'hameçonnage par url.

Auteur	**Technique**	**Méthode**	**Base de données**	**Résultats**
Bahnsen et al,2017	Apprentissage profond	LSTM	1 million d'URLS	98, 70% de précision
Hai et Hwang al,2018	Apprentissage automatique	SVM (Support à vecteur de machine)	150 397 URLS (107 615 légitimes,42 782 phishings)	95.00 % de précision
Hassan, A et al,2019	Apprentissage automatique	CRB-PDS	Cas1 :500 (250 phishings et 250 légitimes) Cas2 :750 (375 phishings et 375 légitimes)	95.00% de précision
Sahingoz et al,2019	Apprentissage automatique	KNN (K-Nearest Neighbours)	11 055 URLS	97 .98% de précision
Weiping et al,2019	Apprentissage profond	PDRCNN	500 000 URLS obtenues de PhishStack	97% de précision
Rasymas et Dovydaitis, 2020	Apprentissage profond	Détection des attaques de phishing basées sur l'empoisonnement DNS	387 772 URLS (153 141 phishings ,234 631 légitimes)	94,4% de précision
Huang et al,2020	Apprentissage automatique	SVM	21 208 URLS (50% phishings ,50% légitimes)	95,80% de précision
Wei et al,2020	Apprentissage profond	Réseau de neurones Convolutif (CNN)	21 208 URLS (50% phishing ,50% légitimes)	99.98% de précision
Shah et al,2020	Apprentissage automatique	PhishStack	Cas 1 : 11055 URLS (4898 phishings et 6157 légitimes) Cas 2 : 1353 URLS (548 légitimes et 702 phishings and 103 suspets) Cas 3 ; 10000 (5000 phishings et 5000 légitimes)	Cas1 : 95.61% de précision Cas 2 : 96.72% de précision Cas 3 : 97.63% de précision
Somesha et al,2020	Apprentissage profond	1-DNN 2-LSTM 3-CNN	3526 URLS (2119 phishings,1407 légitimes)	DNN : 99,52 % de précision LSTM : 99,57% de précision CNN : 99,43 % de précision
Mourtaji et al,2021	Apprentissage profond	1-CNN 2-MLP	40 000 URLS (20 000 phishings,20 000 légitime)	CNN :97.94% de précision MLP :93,21

Le tableau 3.1 présenté dans la page précédente présente le résumé des travaux exsistant dans la litérature qui ont atteint des précision comprises entre 94 et 99 depuis 2017 au 2021.

3.4 Conclusion

Nous avons présenté dans ce chapitre les techniques de détection d'hameçonnage avec les travaux connexes qui ont était réalisés dans la littérature pour la détection des URLs de phishing. Dans le chapitre suivant nous allons présenter notre solution proposée ainsi le modèle de conception.

Chapitre 4

SOLUTION PROPOSÉE

4.1 Introduction

L'apprentissage profond utilise des couches de projections non linéaires empilées afin d'apprendre des représentations à plusieurs niveaux d'abstraction. Il a démontré des performances de pointe dans de nombreuses applications (vision par ordinateur, reconnaissance vocale,traitement du langage naturel, etc.). En particulier, les réseaux convolutifs temporels (TCN) ont montré des performances prometteuses pour la classification de textes ces dernières années. Suite à leur succès, nous proposons d'utiliser les réseaux convolutifs temporels pour apprendre une intégration d'URL pour la détection d'URL malveillants.

4.2 Définition du problème

Les URL malveillantes hébergent du contenu non sollicité et sont utilisées pour perpétrer cyber crimes. Il est impératif de les détecter à temps. Traditionnellement, cela se fait par l'utilisation de listes noires, qui ne peut pas être exhaustive et ne peut pas détecter les programmes malveillants nouvellement générés. Pour y remédier, ces dernières années ont vu plusieurs efforts pour effectuer une détection d'URL malveillantes à l'aide de l'apprentissage automatique. Les approches les plus populaires et évolutives utilisent les propriétés lexicales de la chaîne d'URL en extrayant des fonctionnalités de type sac de mots, suivies en appliquant des modèles d'apprentissage automatique tels que les SVM. Il y a également d'autres fonctionnalités conçues par des experts pour améliorer la prédiction performances du modèle. Ces approches souffrent de plusieurs limites :

(i) Incapacité à saisir efficacement le sens sémantique et des modèles séquentiels dans les chaînes d'URL ;

(ii) Nécessitant des ingénieries manuelles des fonctionnalités ;

(iii) Incapacité à gérer l'invisible fonctionnalité et généraliser pour tester les données.

Pour relever ces défis, nous utilisons TCN, un cadre d'apprentissage en profondeur de bout en bout pour apprendre une intégration d'URL non linéaire pour la détection d'URL malveillantes directement à partir de l'URL. Plus précisément, nous appliquons le word embedding aux mots de la chaîne d'URL pour apprendre l'intégration d'URL dans un cadre optimisé. Cette approche permet au modèle de capturer plusieurs types d'informations . Notre objectif est de classer une URL donnée comme malveillante ou légitime le processus est décrit dans le tableau 4.1 et le pipeline présenté dans la figure 4.1 . Pour ce faire, Nous y parvenons en formulant le problème comme une tâche de classification binaire. Considérons un ensemble de T *URLs,((u1, y1), . . ., (uT, yT))* où ut pour $t = (1, \ldots, T)$ représente une URL, et $yt \in (0, 1)$ désigne l'étiquette de l'URL, y = 1 étant une URL malveillante et yt = 0 étant une URL bénigne. La première étape de la procédure de classification consiste à obtenir une représentation de caractéristiques $ut \rightarrow xt$ où $xt \in Rn$ est le vecteur de caractéristiques à n dimensions représentant l'URL ut. L'étape suivante consiste à apprendre une fonction de prédiction $F : Rn \rightarrow R$ Qui est le score prédisant l'affectation de classe pour une instance d'URL x. La prédiction faite par la fonction notée $y\hat{}t = pred(f(xt))$ L'objectif est d'apprendre une fonction qui peut minimiser le nombre total d'erreurs $(y\hat{}t \neq yt)$ dans l'ensemble du jeu de données. Cet objectif est souvent atteint en minimisant une fonction de perte. De nombreux types de fonctions de perte peuvent être utilisés, qui peuvent également inclure un terme de régularisation. Pour l'apprentissage profond, la fonction F est représentée comme un réseau neuronal profond, tel qu'un réseau convolutif temporel. Le TCN pourrait apprendre des informations structurelles utiles dans le texte à partir des valeurs brutes des incorporations de mots ou de caractères. Dans notre modèle, le TCN est utilisé pour apprendre des informations structurelles sur l'URL.

TABLE 4.1 – Étapes de classification des Urls de phishing

Algorithme : Etapes de classification des Urls de phishing
INPUT : array URL [U1,U2,U3...Un]
Output : Classification (0,1)
Begin
1 Rn←array [n] ;
2 $xt \in Rn$;
3 C←[0,1] ;
4 For (i=0,i<n,i++) do
5 Ui ←Xt[i] ;
6 Rn←C ;
7 End for
8 For (i=0,i<n,i++) do
9 Pred(C',xt[i])
10 If($C" \neq C$) Then
11 Update (loss)
12 End if
13 End for
14 End

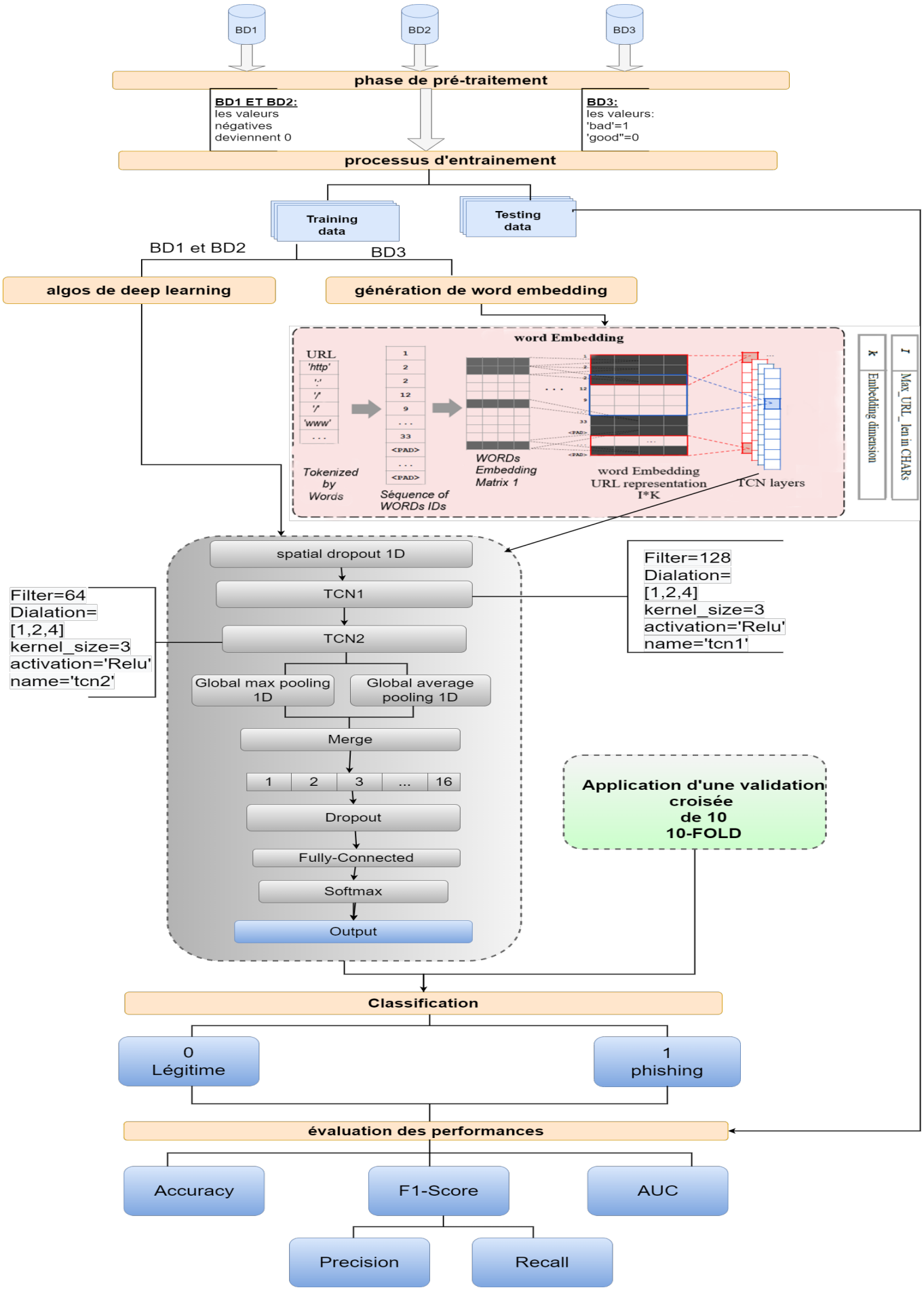

FIGURE 4.1 – Pipeline de notre modèle de détection des URLs de phishing

4.3 Bases de données

Les performances d'un système de détection des URL de phishing dépendent fortement des données utilisées pour le développer. Par conséquent, la collecte de données est une étape importante du processus de développement. Les données collectées doivent comporter un nombre suffisant d'exemples représentatifs de toutes les classes nécessaires pour permettre aux modèles d'apprentissage automatique d'apprendre les paramètres. Nous avons utilisé trois bases de données différentes pour cet examen afin d'étudier les performances des algorithmes d'apprentissage profond ainsi que l'importance des attributs dans les trois ensembles de données.

4.3.1 Bases de Données "Full" et "small"

La première base de données[Vrban et Podgorelec 2020] est divisée en deux parties représentées dans la figure 4.2.

- La première étant les données complètes ou bien dataset-full en anglais. Ces données consistent en une collection d'instances d'URL légitimes et d'URL d'hameçonnage, chaque URL est représentée par un ensemble de 111 caractéristiques (colonnes) qui indiquent si l'URL est légitime ou non. Cet ensemble de données contient 88647 instances au total, 58000 sont légitimes et 30647 frauduleux.
- La seconde est la petite base de données ou bien dataset-small. Elle contient 58645 instances en totale, 27998 d'entre elles sont légitimes et 30647 sont frauduleux.

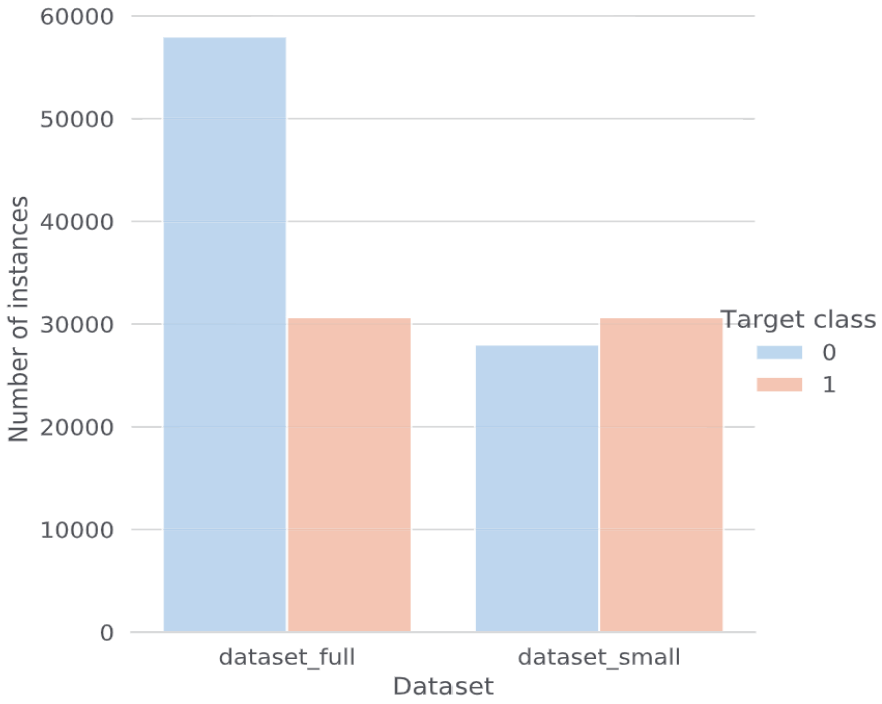

FIGURE 4.2 – Représentation des instances du première base de données.

4.3.2 Deuxième Base de Données

L'ensemble de données 2 présenté dans la figure 4.3 comprend 14 caractéristiques (colonnes) qui identifient de manière unique les URLs de phishing et les URLs légitimes. La sortie est binaire, 1 pour le phishing et 0 pour la légitime comme illustre la figure 4.3. L'ensemble de données est alimenté par des ressources extraordinaires, notamment PhishTank et le moteur de recherche Google. Au total, il contient 11000 instances des URLs, dont 5500 sont des sites légitimes et 5500 des sites de phishing.

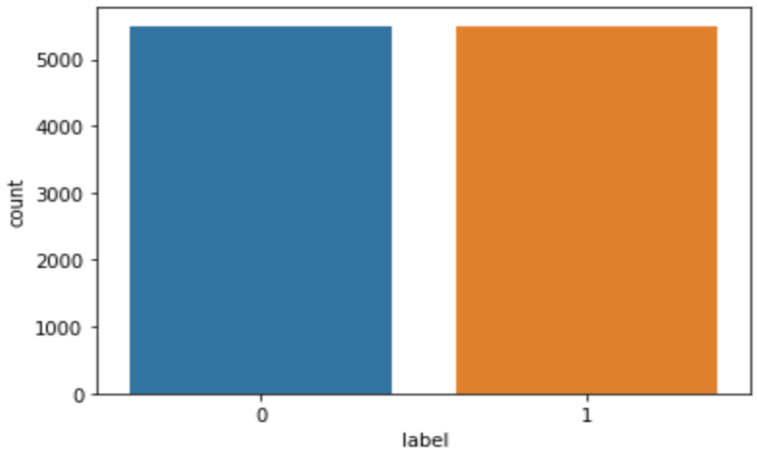

FIGURE 4.3 – Représentation des instances du deuxième de données.

4.3.3 Troisième Base de Données

La troisième base de données provient de Kaggle et contient seulement l'URL et son étiquette (bad pour phishing et good pour légitime)comme illustre la figure4.4. Elle contient les statistiques de 1353 sites Web extraordinaires collectées à partir de ressources spécifiques. Les données collectées comprennent 156422 URLs étiquetées comme mauvaises et 392924 URLs étiquetées comme bonnes.

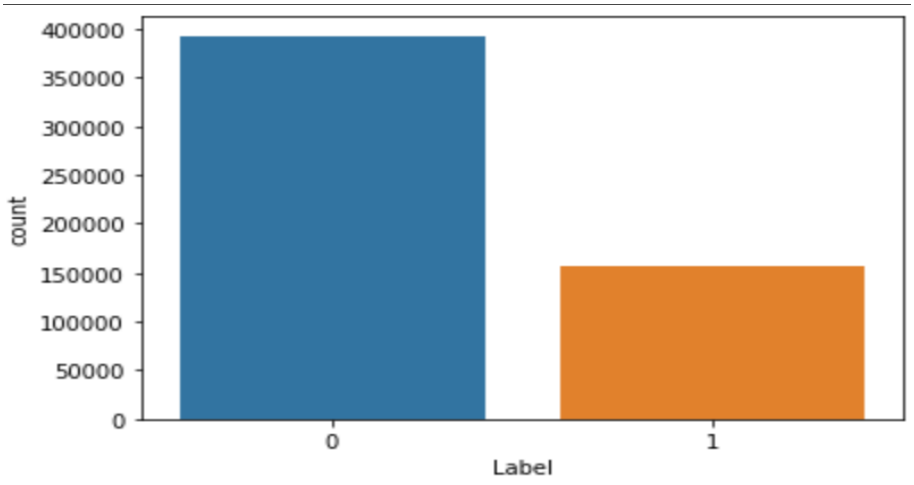

FIGURE 4.4 – Représentation des instances du troisième de données.

4.4 Prétraitement des données

Les données du monde réel sont collectées de différentes manières et ne reflètent pas toujours les intérêts ou les préoccupations spécifiques d'un domaine particulier. Les données incomplètes, non structurées et contenant des erreurs ne sont généralement pas très utiles. Ces données peuvent conduire à des prédictions inexactes si elles sont analysées directement. Dans notre cadre, diverses méthodes sont utilisées pour préparer les données dans la phase de prétraitement.

4.4.1 Codage des caractéristiques catégorielles

Souvent, les caractéristiques ne sont pas données sous forme de valeurs continues mais sous forme de catégories. Par exemple, dans notre cas la 3eme base de données, les URLS sont classées ["good", "bad"]. Ces caractéristiques peuvent être codées efficacement sous forme d'entiers, ["good", "Bad"] pourrait être exprimé par [0, 1] comme illustre la figure4.5 . Pour convertir les caractéristiques catégorielles en de tels codes entiers, nous avons utilisé une fonction. Elle transforme chaque caractéristique catégorielle en une nouvelle caractéristique d'entiers (0 ou 1).

FIGURE 4.5 – exemple sur le codage catégoriel de 'bad'.

4.4.2 Normalisation des Ensembles de Données

La normalisation des ensembles de données est une exigence commune à de nombreux estimateurs d'apprentissage automatique. Ils peuvent se comporter de manière incorrecte si les caractéristiques individuelles ne ressemblent pas plus ou moins à des données standard normalement distribuées : base de données 1 et 2 contient des valeurs négatives. Si une caractéristique a une variance négative, elle peut dominer la fonction objective et rendre l'estimateur incapable d'apprendre d'autres caractéristiques correctement comme prévu. Donc on a appliqué une fonction qui permet de transformer tous les valeurs -1 à 0 comme présente la figure 4.6.

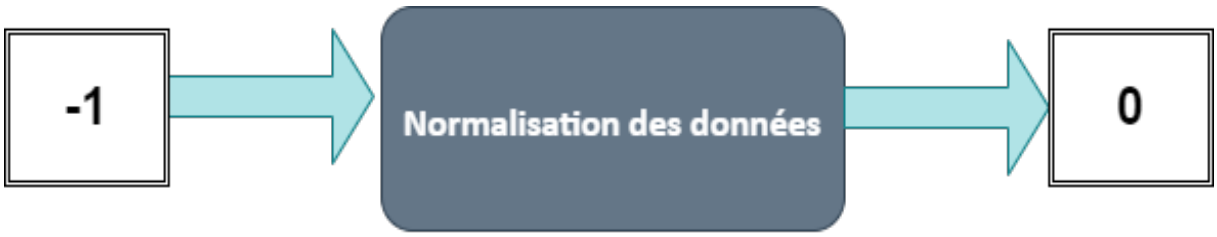

FIGURE 4.6 – exemple sur la normalisation de '-1'.

4.4.3 Rééchantillonnage Des Données

Pour corriger l'apprentissage absorbant de la première base de données de la classe dominante (0), nous avons décidé d'équilibrer l'ensemble de données en utilisant le sous-échantillonnage, qui est une technique utilisée dans l'exploration et l'analyse de données pour modifier des classes de données inégales afin de créer des ensembles de données équilibrés. Également connu sous le nom de rééchantillonnage, nous avons donc réduit le nombre d'échantillons pour chaque classe de l'ensemble de données au nombre d'échantillons dans la classe minoritaire (1) qui contient 30647 échantillons, nous avons extrait 30647 échantillons de l'autre classe (0). Nous avons également suréchantillonné l'ensemble de données en utilisant le suréchantillonnage. En augmentant le nombre d'échantillons pour chaque classe de l'ensemble de données dans le nombre d'échantillons de la classe (0) qui contient 58645 échantillons,nous avons extrait 58645 échantillons de l'autre classe (1).

4.5 Modèle

Notre système de détection du phishing se compose de deux modules. Plus précisément, le module d'apprentissage de l'intégration des mots (word2vec et Glove) réalise la représentation vectorielle des mots dans les URL. Ensuite, le module de détection entraîne des algorithmes d'apprentissage automatique sur les représentations vectorielles des URL afin de les classer entre hameçonnage et légitimes. Étant donné les représentations vectorielles des URLs, n'importe quels algorithmes de classification peuvent être appliqués sans problème pour détecter les URL de phishing. Dans ce travail, TCN est adopté pour son efficacité et son efficience. Dans nos expériences, les performances d'un certain nombre de modèles de classification, notamment réseau neuronal convolutif (CNN), réseau neuronal récurrent (RNN),réseau neuronal convolutif avec mémoire à court terme (CNN+LSTM), sont étudiées à des fins de comparaison. Le modèle TCN proposé est inspiré de Christof Henkel, l'un des grands maîtres de Kaggle. Le modèle se compose de : Deux blocs TCN empilés avec la taille de noyau de 3 et des facteurs de dilatation de 1, 2 et 4. Le premier bloc TCN contient 128 filtres, et le bloc alternatif utilise 64 filtres. Les

caractéristiques d'entrée seront fondées sur l'incorporation de mots, car nous ne pouvons pas transmettre les URL brutes à notre modèle en entrée avant de les avoir converties en chiffres. Le résultat de chaque bloc prendra la forme d'une séquence. La séquence finale est également transmise à deux couches de mise en commun globale différentes. Ensuite, les deux résultats sont concaténés et transmis à une couche dense de 16 neurones, puis transmis à la couche de sortie dense 2 avec une fonction d'activation softmax.

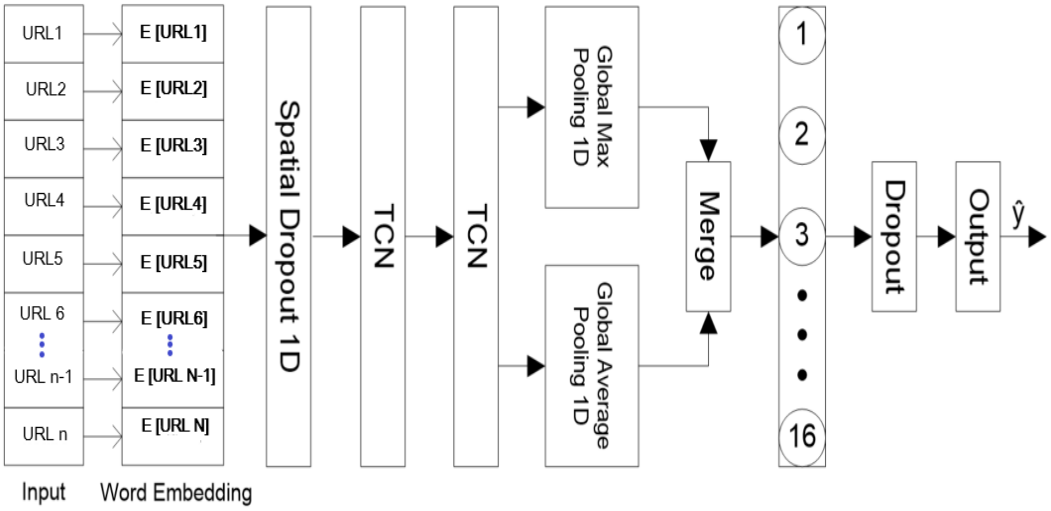

FIGURE 4.7 – Représentation de notre Modèle TCN

4.6 Génération de Word Embedding

Dans le contexte de la modélisation du langage, l'entrée requise est un corpus et un vocabulaire, et la sortie est constituée d'encastrements des mots du vocabulaire. Dans le contexte de la détection du phishing, nous construisons un corpus en collectant un grand nombre d'URL, tandis que le vocabulaire peut être considéré comme les caractères dans les URLs, par exemple, les lettres majuscules, les lettres minuscules, les chiffres, etc. Pour obtenir les enchâssements des caractères dans les URLs, nous adoptons word2vec, alternativement, nous pouvons adopter d'autres stratégies populaires, telles que GloVe des perspectives globales, etc. Nous étudierons les variantes de l'exploitation des différentes stratégies d'apprentissage de l'intégration des caractères, et nous évaluerons leurs influences sur les performances dans nos expériences ultérieures. En fait, les influences du choix de différentes stratégies sont assez faibles, surtout pour les données à grande échelle. C'est cette approche de la représentation des mots et des documents qui peut être considérée comme l'une des principales percées de l'apprentissage profond sur des problèmes difficiles de traitement du langage naturel dans notre cas la représentation des URL à notre modèle TCN.

4.6.1 Application de Word2vec Embedding

Les URL ont été transformées en nombres comme présente la figure 4.8, chaque URL étant représentée par un tableau d'entiers d'une longueur spécifique. Nous avons configuré l'objet Tokenizer, qui a été importé comme un jeton de la bibliothèque Keras. Le tokenizer est ensuite appliqué à l'ensemble de l'URL, chaque URL se voyant attribuer un numéro unique et chaque mot de l'URL étant représenté par un numéro. Ensuite, chaque phrase est convertie en une séquence de nombres qui lui est attribuée. En remplissant les séquences, nous pourrons faire en sorte que toutes les phrases aient la même longueur. Le paramètre maxlen détermine la longueur maximale. Nous fixons la taille du vocabulaire à la quantité de mots dans le vecteur de jetons, plus un, de sorte que si un mot qui n'a jamais été vu auparavant entre dans le modèle, il sera alloué à cet endroit.

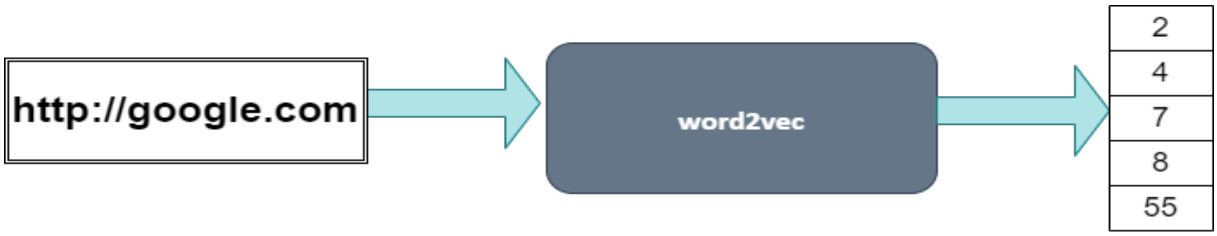

FIGURE 4.8 – exemple sur l'application de word2vec sur un URL.

4.6.2 Application de Glove Embedding

Pour le Glove Embedding, nous avons construit un dictionnaire de vecteurs d'incorporation dont les clés sont définies comme des mots trouvés dans le fichier du Glove Embedding n(840B, 300d), et la valeur de chaque clé étant l'incorporation trouvée dans le fichier. Tous les mots du fichier du Glove Embedding seront inclus dans ce dictionnaire. Nous avons généré une matrice comprenant uniquement les mots de notre vocabulaire ainsi que leurs vecteurs d'incorporation comme présente la figure 4.9.

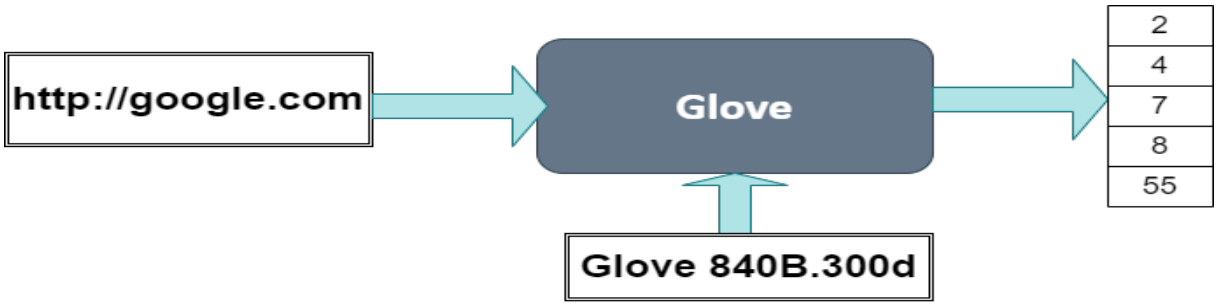

FIGURE 4.9 – Un exemple sur l'application de Glove sur un URL.

4.7 Validation croisée

La validation croisée est une technique utilisée pour évaluer des modèles d'apprentissage automatique sur un petit échantillon de données. L'algorithme de clustering k-means a un seul paramètre, appelé k, qui spécifie le nombre de groupes auxquels un échantillon de données doit être divisé. La validation croisée K-fold est une procédure courante pour améliorer la précision des prédictions en s'appuyant sur un ensemble de données qui a été divisé en k sous-ensembles. Lorsqu'une valeur spécifique pour k est choisie, elle peut être utilisée comme paramètre k dans la référence du modèle, dans notre cas k=10 pour 10 fois la validation croisée. Pour analyser les données, divisez-les en k parties, puis examinez chaque partie séparément. Les plis k-1 sont utilisés pour former le modèle, et le dernier pli est utilisé pour tester le modèle. Ce processus est répété k fois. Chaque fois qu'une mesure du rendement peut être utilisée pour évaluer et noter le rendement, il s'agit d'un outil important pour les gestionnaires.La moyenne des mesures de performance est ensuite enregistrée. Les rapports de classe sont conservées dans StratifiedKFold.

4.8 Conclusion

Dans ce chapitre, nous avons proposé un réseau neuronal profond basé sur le TCN pour la détection d'URL malveillants. Nous avons proposé des techniques d'incorporation de mots qui sont particulièrement utiles pour traiter les mots rares, un problème généralement observé dans les tâches de détection d'URL malveillantes. Cette approche a également permis à notre modèle d'apprendre l'intégration de mots non vus au moment du test et d'exploiter les informations sur les URLs. Dans le chapitre suivant nous allons voir les résultats de notre solution proposée.

Chapitre 5

TEST ET RESULTATS

5.1 Introduction

Les URLs malveillantes sont l'un des principaux mécanismes de perpétration de cybercrimes. Elles hébergent des contenus non sollicités et s'attaquent à des utilisateurs non avertis, les rendant victimes de divers types d'escroqueries. Il est donc devenu impératif de concevoir des techniques robustes pour détecter les URLs malveillantes en temps voulu. Ce chapitre est divisé en deux sections. La première section est consacrée à la présentation et l'évaluation des algorithmes que nous proposons pour la détection des urls de phishing. Dans la deuxième section du chapitre, nous présentons notre application finale pour le déploiement de ces algorithmes.

5.2 Environnement et Outils de Travail

Nous avons mené nos expérimentations en utilisant :

• Python 3.6 : qui est un langage de programmation open source orienté objet [Lutz, 2013].

• Embedding : qui est une bibliothèque de traitement des chaines de caractères en Python, pour effectuer des opérations liées aux mots.

Nous avons développé nos modèles et réalisé les expériences en utilisant les ressources de calcul des GPU et CPU fournies par Google Colaboratory [Bisong , 2019]. Plus communément appelé "Google Colab" ou tout simplement "Colab" qui est un service gratuit hébergé par Google, basé sur l'environnement Jupyter, destiné à l'enseignement et à la recherche sur l'apprentissage automatique. Nos machines informatiques (laptops) :

- **Processeur :** processeur Intel(R) Core(™) i5-7300 à 2,60 GHz 2,71 GHz.
- **RAM :** 8GO.
- **Système :**Système 64 bits, processeur x64

De plus, le processus d'apprentissage automatique a été réalisé à l'aide d'un ensemble de packages Python tels que :

- **Scikit-learn :** [Pedregosa et al., 2011] qui est une bibliothèque d'apprentissage automatique qui affiche une grande diversité d'algorithmes.
- **Keras :** [Ketkar, 2017] qui est l'une des principales API de réseaux de neurones de haut niveau, elle est écrite en Python et prend en charge plusieurs moteurs de calcul de réseau de neurones back-end.
- **Flask :** [Walsh , 2007] qui est un framework web Python petit et léger qui fournit des outils et des fonctionnalités utiles pour faciliter la création d'applications web en Python. Il offre aux développeurs une certaine flexibilité et constitue un cadre plus accessible pour les développeurs, car vous pouvez créer rapidement une application web en utilisant un seul fichier Python.

D'autres bibliothèques ont été invoquées lors de la conduite de nos expériences, notamment Numpy, Pandas, Matplotlib, TensorFlow et imblearn.

5.3 Evaluation Des Performances

Pour évaluer le modéle proposé, nous utilisons le taux de vrais positifs, taux de vrais négatifs, le taux de faux positifs, le taux de faux négatifs, la justesse, le score-f1, la précision, le rappel, la matrice de confusion, le rapport de classification et l'AUC (Area Under Curve).

5.3.1 Matrice de Confusion

Une matrice de confusion [Phone et al., 2020] est un tableau souvent utilisé pour décrire les performances d'un modèle de classification, comme le montre la figure 5.1, sur un ensemble de données de test ou les valeurs réelles sont connues. Dans le cas dans une classification binaire on a :

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

FIGURE 5.1 – Matrice de confusion.

Où :

5.3.1.1 Vrai Positif (VP)

Elément de la classe 1 correctement prédit, dans notre cas un élément qui est légitime prédit correctement comme légitime

5.3.1.2 Vrai Négatif (VN)

Elément de la classe 0 correctement prédit, dans notre cas un élément qui est phishing prédit correctement comme phishing.

5.3.1.3 Faux Positif (FP)

Elément de la classe 1 mal prédit, dans notre cas un élément qui est légitime prédit comme phishing

5.3.1.4 Faux Négatif (FN)

Elément de la classe 0 mal prédit, dans notre cas un élément qui est phishing prédit comme légitime.

5.3.2 Précision

Il s'agit du rapport entre le nombre de sites Web de phishing identifiés avec précision et le nombre total de sites Web identifiés comme phishing [Davis et Goadrich, 2006].

$$PRECISION = \frac{VP}{VP \pm FP} \tag{5.1}$$

5.3.3 Rappel

Le rappel [Davis et Goadrich, 2006] calcule le ratio des sites de phishing correctement identifiés par rapport au total des sites de phishing.

$$RAPPEL = \frac{VP}{VP \pm FN} \tag{5.2}$$

5.3.4 Accuracy

Le taux global de prédictions exactes [Simundic , 2009]est déterminé par l'accuracy.

$$ACCURACY = \frac{VP \pm VN}{VP \pm FP \pm VN \pm FN} \tag{5.3}$$

5.3.5 Score F1

Le score F1 [Lipton et Narayanaswamy, 2014] correspond à la moyenne harmonique de la précision et du rappel.

$$F1 - SCORE = \frac{2PRECISION * RAPPEL}{PRECISION \pm RAPPEL} \tag{5.4}$$

5.3.6 Rapport de Classification

Le rapport de classification[Craig et al., 2017] concerne les mesures clés d'un problème de classification. Vous aurez la précision, le rappel, le f1-score et le support pour chaque classe que vous essayez de trouver.

5.3.7 Courbe ROC

L'illustration de la capacité de diagnostic de la détection de sites de phishing d'un système de classification binaire peut être représentée graphiquement à l'aide d'une courbe ROC [De et Hand, 2010](Receiver Operating Characteristics).

5.3.8 AUC

L'aire bidimensionnelle entière [Simundic , 2009]sous la courbe ROC est mesuré par l'AUC. La courbe ROC est mesurée par l'AUC et fournit une mesure globale de la performance pour tous les seuils de classification possibles.

5.4 Paramètres de compilation

Pour la compilation des modèles plusieurs paramètres ont été utilisé qui sont :

5.4.1 Perte

La perte [Buja et Stuetzle, 2005] est une fonction utilisée pour optimiser les valeurs des paramètres dans un modèle de réseau neuronal. Il existe plusieurs fonctions courantes qui mesurent souvent l'erreur quadratique ou absolue entre la sortie d'un réseau et une sortie cible.

5.4.1.1 Entropie croisée Binaire

Cette perte [Buja et Stuetzle, 2005] est un cas particulier de l'entropie croisée car lorsque nous n'avons que deux classes, elle peut être réduite à une fonction plus simple. Elle est utilisée pour mesurer l'erreur . Cette formule suppose que x et y sont des probabilités, elles sont donc

strictement comprises entre 0 et 1.

$$l(x, y) = L = l1, ..., lNT, ln = -wn[ynlogxn + (1 - yn)log(1 - xn)] \tag{5.5}$$

5.4.2 Fonctions d'Activation

Une fonction d'activation [Nwanpka et al., 2018] décide si un neurone doit être activé ou non. Cela signifie qu'elle décidera si l'entrée du neurone dans le réseau est importante ou non dans le processus de prédiction à l'aide d'opérations mathématiques plus simples.

5.4.2.1 Softmax

Softmax [Nwanpka et al., 2018] est la seule fonction d'activation qu'il est recommandé d'utiliser avec la fonction de perte de crossentropie catégorielle. À proprement parler, la sortie du modèle ne doit être positive que pour que le logarithme de chaque valeur de sortie y^i existe. Cependant, le principal intérêt de cette fonction de perte réside dans la comparaison de deux distributions de probabilité. L'activation softmax redimensionne la sortie du modèle afin qu'elle ait les bonnes propriétés.

5.4.3 Optimiseurs

Les optimiseurs [Pentland et Liu, 1999] sont des algorithmes ou des méthodes utilisés pour modifier les attributs de votre réseau neuronal, tels que les poids et le taux d'apprentissage, afin d'améliorer la qualité du réseau. Afin de réduire les pertes.

5.4.3.1 Adam

Adam [Choi et al., 2019] est l'abréviation d'adaptive moment estimation, et est une autre façon d'utiliser les gradients passés pour calculer les gradients actuels. Cet optimiseur est devenu assez répandu, et son utilisation est pratiquement acceptée pour la formation de réseaux de neurones.

5.4.3.2 SGD

SGD [Zhou et al., 2019] est une variante de la descente de gradient. Au lieu d'effectuer des calculs sur l'ensemble des données, ce qui est redondant et inefficace, SGD ne calcule que sur un petit sous-ensemble ou une sélection aléatoire d'exemples de données produit également la même performance que la descente de gradient régulière lorsque le taux d'apprentissage est faible.

5.4.3.3 Nadam

Nadam [Tato et al., 2018] est utilisé pour les gradients bruyants ou pour les gradients à forte courbure. Le processus d'apprentissage est accéléré en additionnant la décroissance exponentielle des moyennes mobiles pour les gradients précédent et actuel.

5.4.3.4 Adamax

Optimiseur qui implémente l'algorithme Adamax [Llugsi et al., 2021], il s'agit d'une variante d'Adam basée sur la norme de l'infini. Adamax est parfois supérieur à Adam, en particulier dans les modèles avec encastrements.

5.4.3.5 Rmsprop

Rmsprop est une technique d'optimisation basée sur le gradient proposée par Geoffrey Hinton. Moyenne mobile des gradients au carré pour normaliser le gradient lui-même
[Howard et Wang, 2012]

5.4.3.6 Taux d'apprentissage (LR)

Le taux d'apprentissage [Li, 2018] est un paramètre pour les optimiseurs, c'est un hyperparamètre qui contrôle le degré d'ajustement des poids de notre réseau poids par rapport au gradient de perte.

5.5 Expériences

Pour effectuer notre travail nous avons passé par plusieurs expériences qui nous les classent en 5 grandes séries.

- **Expérience 1 :** Dataset-full Vs Dataset-small.
- **Expérience 2 :** Données Equilibrées.
- **Expérience 3 :** WordToVec Vs Glove Embedding.
- **Expérience 4 :** Impact de changement des paramètres du TCN sur les trois datasets.
- **Expérience 5 :** Test Final.

5.5.1 Expérience 1 : Dataset-Full Vs Dataset-Small

Nous avons mené des expériences approfondies pour améliorer la tâche de classification en utilisant différentes méthodes et environnements de travail. Nous avons examiné en détail 3 techniques différentes et étudié leurs performances à l'aide de mesures d'évaluation pour chaque cas afin d'analyser, discuter et de comparer les résultats obtenus. Le premier cas repose sur le test de l'ensemble des données, le seconde implique une combinaison de sous-échantillonnage et d'apprentissage par transfert, et le dernier cas d'étude dans cette série, est une combinaison de sous-échantillonnage, d'apprentissage par transfert et de réglage fin (fine tuning).

5.5.1.1 Cas d'étude 1 : Entrainement sur l'ensemble des données

Dans ce cas, nous avons appliqué TCN sur dataset-full. La figure 5.2 ci-dessous est un graphique circulaire représentatif de la distribution des échantillons d'URLs ce qui indique que notre jeu de données est déséquilibré.

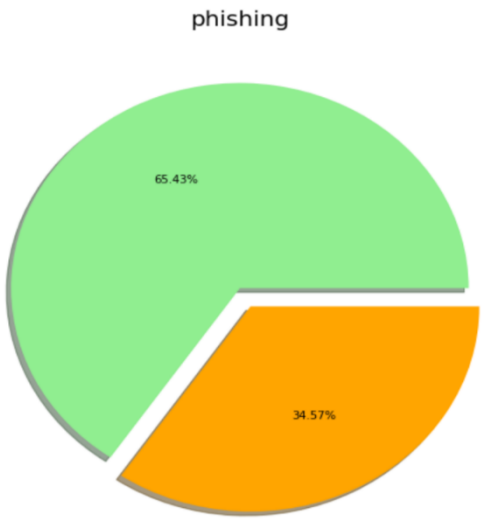

FIGURE 5.2 – Distribution de dataset-full imbalancée.

• Application du Modèle TCN sur dataset-full

Dans ce cas, nous avons obtenu une moyenne valeur d'accuracy de 96, 14%, le tableau 5.1 et la figure 5.3 représentent les métriques d'évaluation : précision, rappel, le f1-score et la matrice de confusion obtenue en appliquant le modèle TCN avec les 3 optimiseurs Adam, RMSprop et Adamax, en utilisant un nombre d'époques=100.

TABLE 5.1 – Mesures d'évaluation de TCN sur dataset-full en utilisant différents optimiseurs

Optimiseur	**Mesures d'évaluation**	**TCN Dataset_full**
ADAM	Précision	0.96
	Rappel	0.96
	F1-score	0.96
RMSprop	Précision	0,93
	Rappel	0,92
	F1-score	0,93
Adamax	Précision	0,92
	Rappel	0,91
	F1-score	0,91

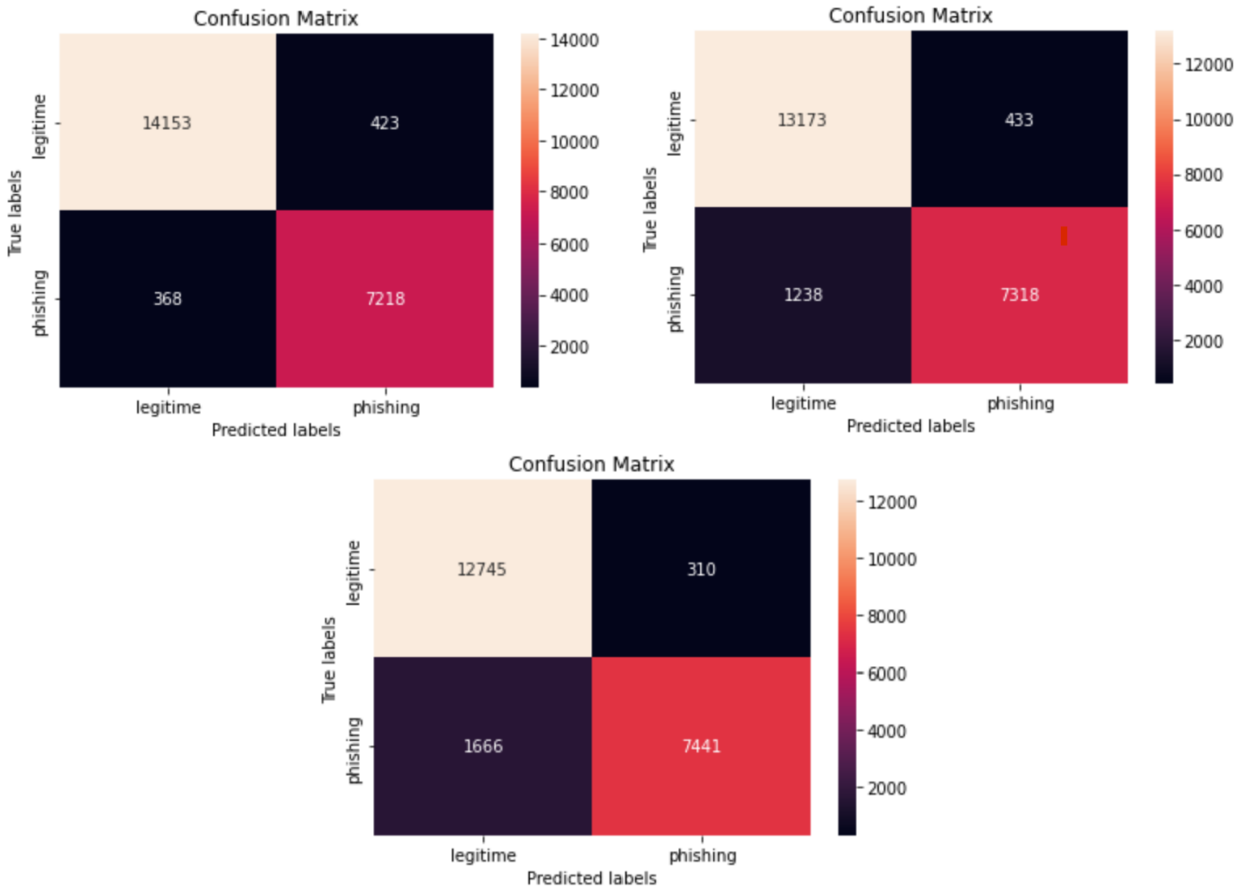

FIGURE 5.3 – Adam-RMSprop-Adamax : Matrices de confusion sur dataset-full.

• Application du Modèle TCN sur dataset-small

Dans ce cas, nous avons obtenu une faible valeur d'accuracy de 93, 84% par raport au dataset-full, le tableau 5.2 et la figure 5.4 représentent les métriques d'évaluation et la matrice de confusion obtenue avec les 3 optimiseurs Adam, RMSprop et Adamax, en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques=100.

TABLE 5.2 – mesures d'évaluation de TCN sur dataset-small en utilisant différents optimiseurs

Optimiseur	**Mesures d'évaluation**	**TCN Dataset_small**
ADAM	Précision	0.94
	Rappel	0.94
	F1-score	0.94
RMSprop	Précision	0,91
	Rappel	0,91
	F1-score	0,91
Adamax	Précision	0,90
	Rappel	0,90
	F1-score	0,90

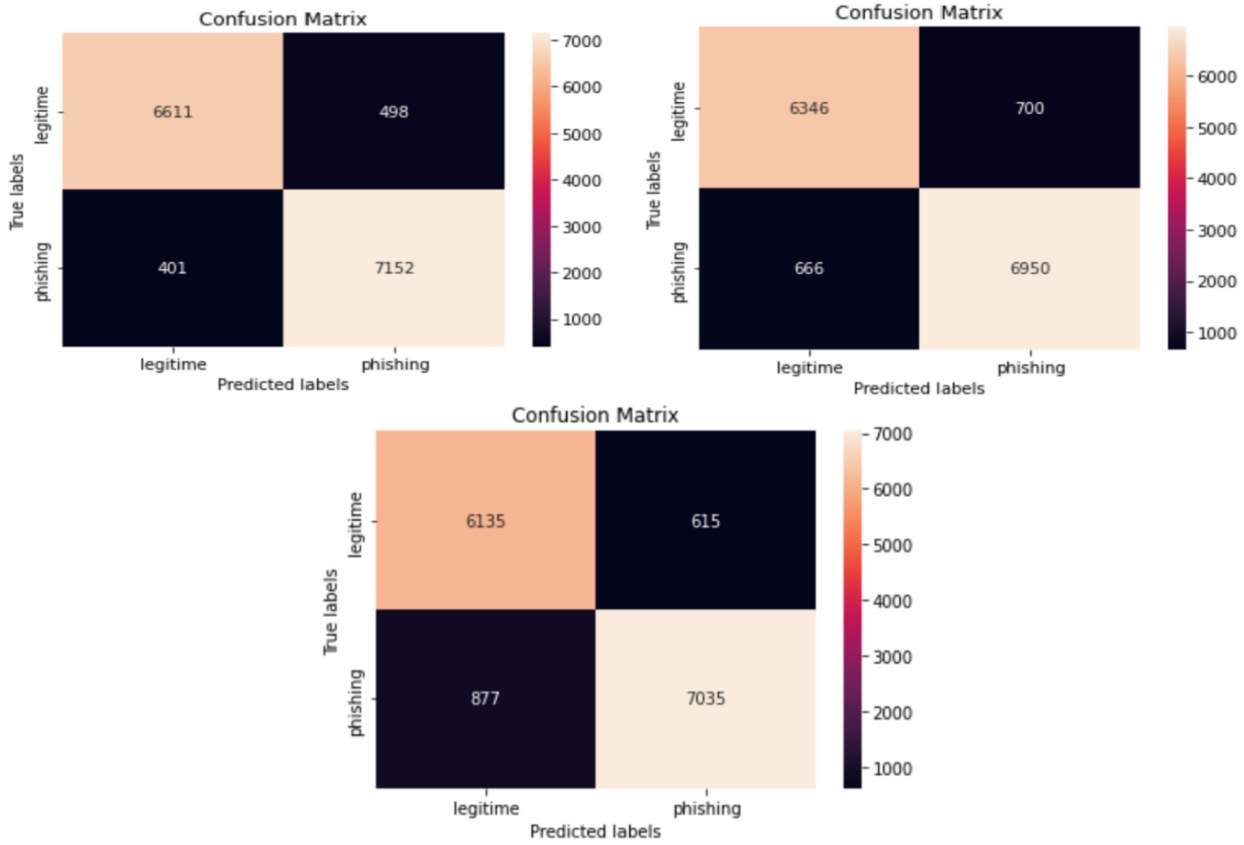

FIGURE 5.4 – Adam-RMSPROP-Adamax : Matrices de confusion sur dataset-small.

5.5.1.2 Cas d'étude 2 : Echantillonnage

Dans ce cas nous avons appliqué deux méthodes d'échantillonnage sur la base de données en comparant leurs résultats.

• Application d'un sous-échantillonnage (undersampling)

Dans cette étude de cas, nous avons réduit le nombre d'échantillons comme illustre la figure 5.5, en utilisant la fonction « NearMiss » pour chaque classe de l'ensemble de données dans le nombre d'échantillons dans la classe minoritaire (1) qui contient 30647 échantillons, nous avons extrait 30647 échantillons de l'autre classe (0).

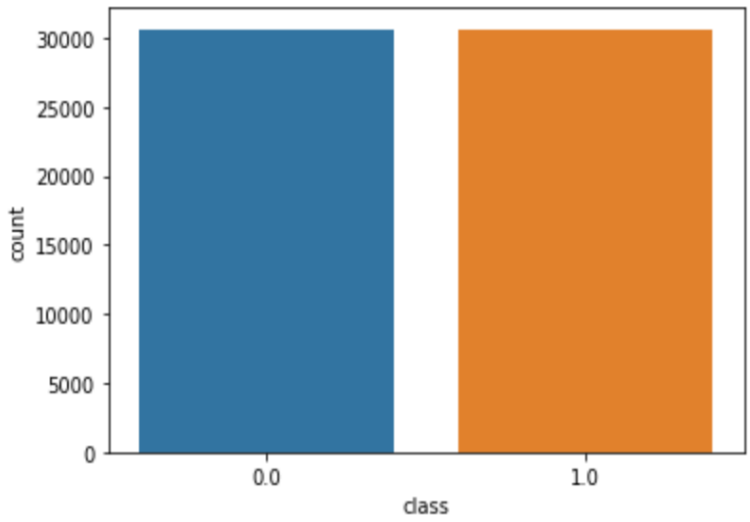

FIGURE 5.5 – Distribution de dataset-full balancée avec sous-échantillonnage

Nous avons divisé nos données en ensembles de train et de validation, puis nous avons utilisé différents optimiseurs sur le TCN. Le tableau 5.3 et les matrices de confusion résument les résultats : Dans ce cas d'étude, TCN a obtenu une faible valeur d'accuracy qui est $95,47\%$ La figure 5.6 ci-après représente les matrices de confusion obtenues en appliquant le modèle TCN avec les 4 optimiseurs Adam, RMSprop, Nadam et Adamax en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques =100.

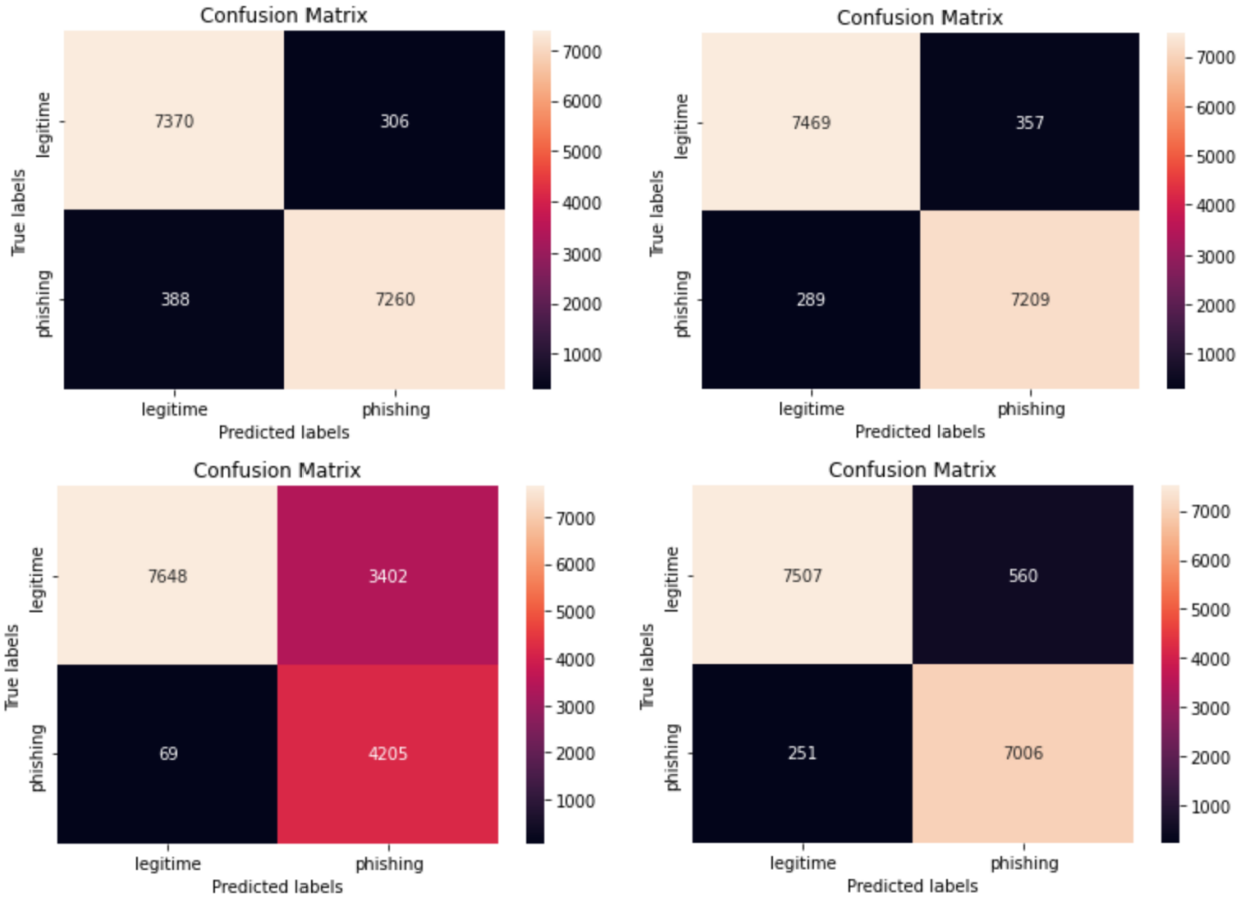

FIGURE 5.6 – Adam- ADAMAX- NAdam – RMSprop : matrices de confusion sur dataset-full balancée avec sous-échantillonnage

• Application d'un suréchantillonnage (oversampling)

Nous avons suréchantillonné l'ensemble de données comme illustre la figure 5.7, nous avons donc augmenté le nombre d'échantillons en utilisant la fonction « SMOTE » pour chaque classe de l'ensemble de données dans le nombre d'échantillons de la classe (0) qui contient 58645 URLs, nous avons extrait 58645 URLs de l'autre classe (1).

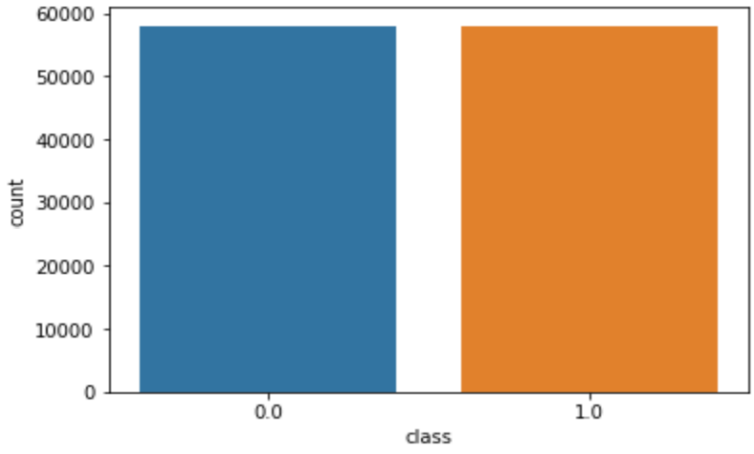

FIGURE 5.7 – Distribution de dataset-full balancée avec sur-échantillonnage.

Nous avons divisé également nos données en ensembles de train et de validation, puis nous avons utilisé différents optimiseurs sur le TCN. Le tableau 5.3 et les matrices 5.8 de confusion ci-dessous résument les résultats : Dans ce cas d'étude, TCN a obtenu une meilleure valeur d'accuracy de $97,17\%$

Les figures 5.8 ci-après représentent les matrices de confusion obtenues en appliquant le modèle TCN avec les 4 optimiseurs Adam, RMSprop, Nadam et Adamax en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques =100.

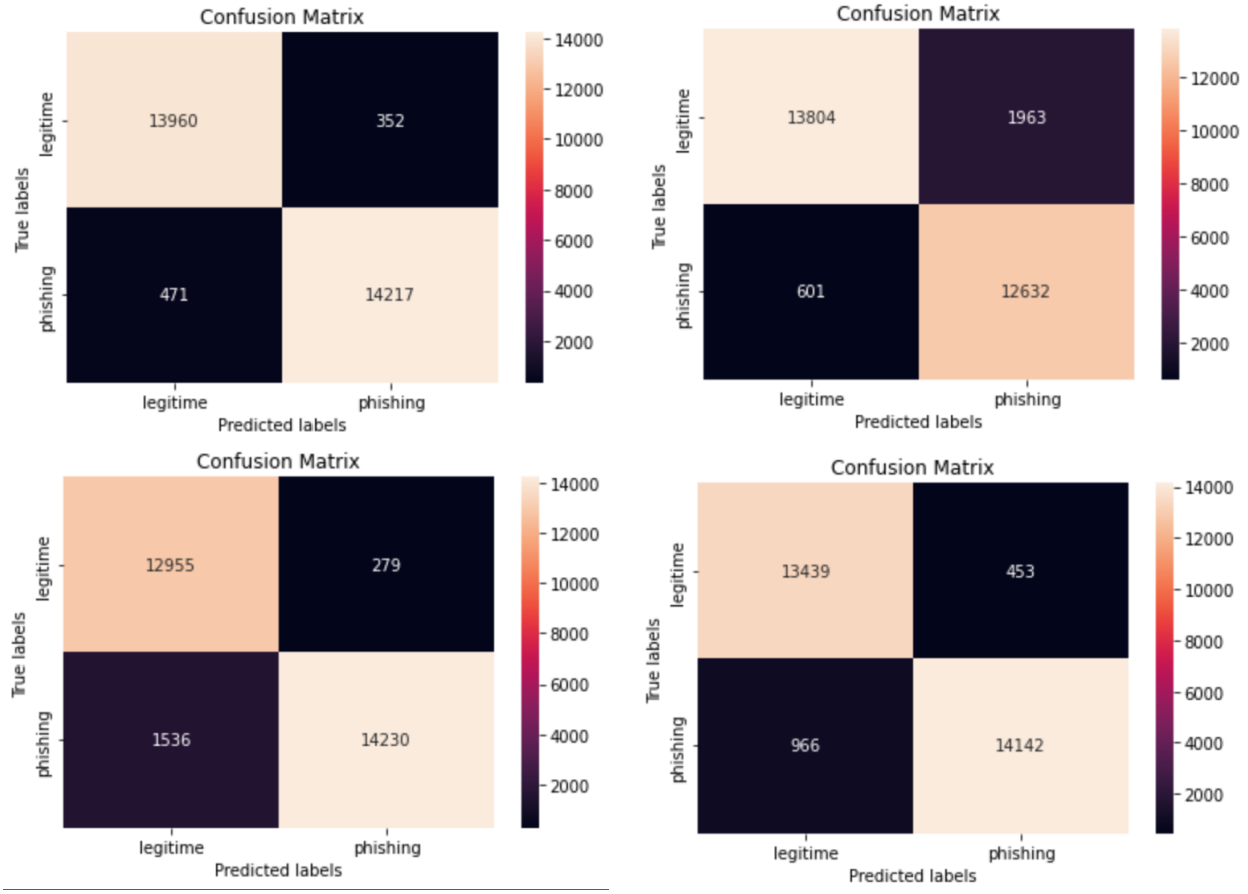

FIGURE 5.8 – Adam- ADAMAX - NAdam – RMSprop : matrices de confusion sur dataset-full balancée avec suréchantillonnage

Le tableau 5.3 présente les mesures d'évaluation : précision, rappel et f1-score obtenus en appliquant le modèle TCN sur les deux méthodes d'échantillonnages avec les 4 optimiseurs Adam, RMSprop, Nadam et Adamax en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques =100. On remarque dans ce cas que le oversampling a donné des meilleurs résultats par rapport au undersampling.

TABLE 5.3 – mesures d'évaluation de TCN sur les données balancées en utilisant différents optimiseurs

Optimiseur	**Mesures d'évaluation**	**TCN Dataset_full Undersampling**	**TCN Dataset_full Oversampling**
ADAM	Précision	0.95	0,97
	Rappel	0.95	0,97
	F1-score	0.95	0,97
RMSPROP	Précision	0,95	0,95
	Rappel	0,95	0,95
	F1-score	0,95	0,95
ADAMAX	Précision	0,96	0,92
	Rappel	0,96	0,91
	F1-score	0,96	0,91
NADAM	Précision	0,84	0,94
	Rappel	0,77	0,94
	F1-score	0,76	0,94

5.5.1.3 Cas d'étude 3 : Fine Tuning

Après avoir testé tous les cas nous avons utilisé un réglage fin (fine tuning) qui consiste à trouver les meilleurs paramètres finaux pour le modèle car cela peut potentiellement apporter des améliorations significatives. Nous avons donc testé cette idée sur notre modèle qui est le TCN avec la même méthode que dans le cas précédent qui est la fonction « SMOTE » mais avec deux types de « sampling-strategy », le premier est réglé sur 'auto' et le second est réglé sur 'all' qui consiste à rééchantillonner toutes les classes, nous avons utilisé différents optimiseurs ,les différents résultats sont représenté dans les tableaux 5.4 et 5.5 :

5.5.1.3.1 Sampling-strategy= 'auto' Comme résultats, nous avons obtenu une accuracy élevée de $96,90\%$Le tableau 5.4 et la figure 5.9 présentent les mesures d'évaluation : précision, rappel et f1-score et la matrice de confusion obtenus en appliquant les 3 optimiseurs Adam, RMSprop et Adamax en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques =100.

TABLE 5.4 – mesures d'évaluation avec sampling-strategy= 'auto' et en utilisant différents optimiseurs

Optimiseur	**Mesures D'évaluation**	**TCN Sampling_strategy= 'auto'**
ADAM	Précision	0.97
	Rappel	0.97
	F1-score	0.97
RMSPROP	Précision	0,96
	Rappel	0,96
	F1-score	0,96
ADAMAX	Précision	0,91
	Rappel	0,91
	F1-score	0,91

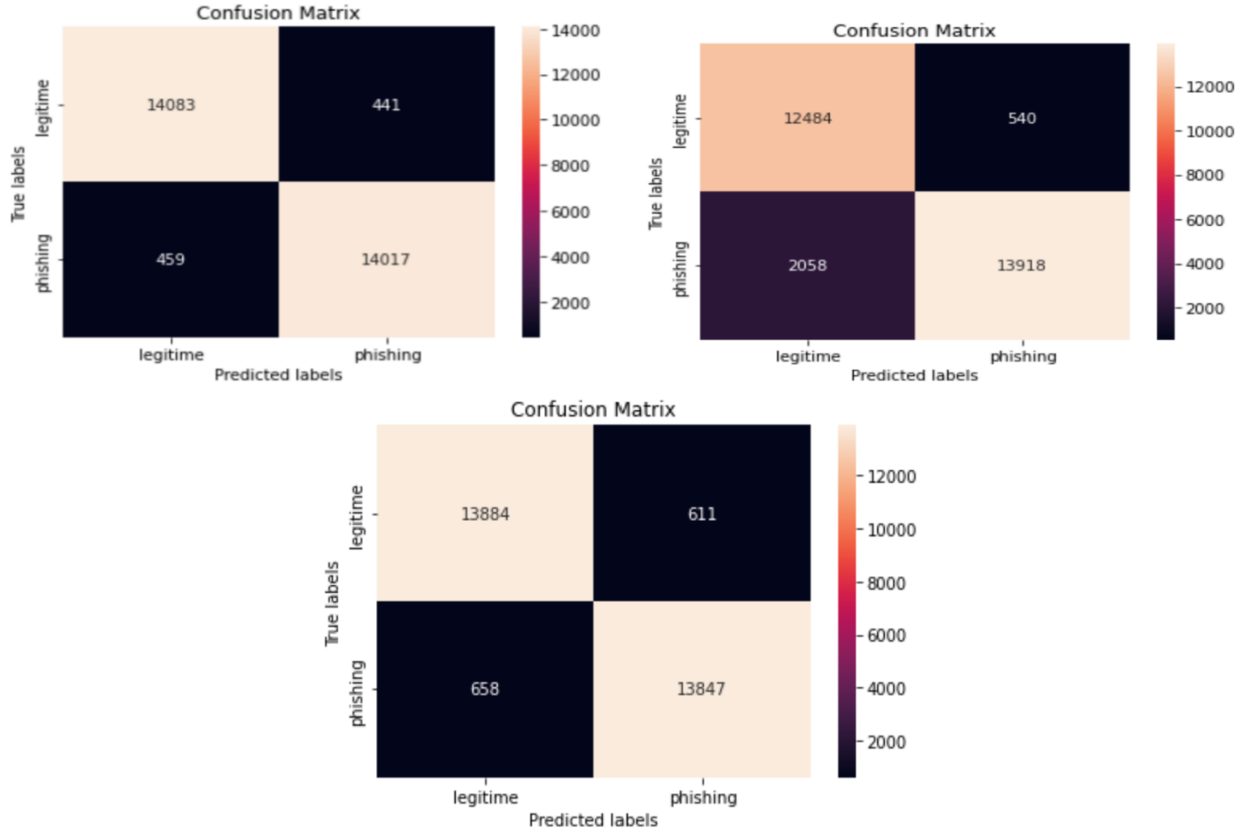

FIGURE 5.9 – Adam – Adamax – RMSprop : matrices de confusion avec sampling-strategy= 'auto'.

5.5.1.3.2 Sampling-strategy= 'all' Comme résultats, nous avons obtenu accuracy de 96, 31%. Le tableau 5.5 et la figure 5.10 présentent les mesures d'évaluation : précision, rappel et f1-score et la matrice de confusion obtenus en appliquant les 3 optimiseurs Adam, RMSprop et Adamax en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques =100.

TABLE 5.5 – mesures d'évaluation avec sampling-strategy= 'all' et en utilisant différents optimiseurs

Optimiseur	**Mesures D'évaluation**	**TCN Sampling_strategy= 'all'**
ADAM	Précision	0.96
	Rappel	0.96
	F1-score	0.96
RMSPROP	Précision	0,94
	Rappel	0,94
	F1-score	0,94
ADAMAX	Précision	0,93
	Rappel	0,93
	F1-score	0,93

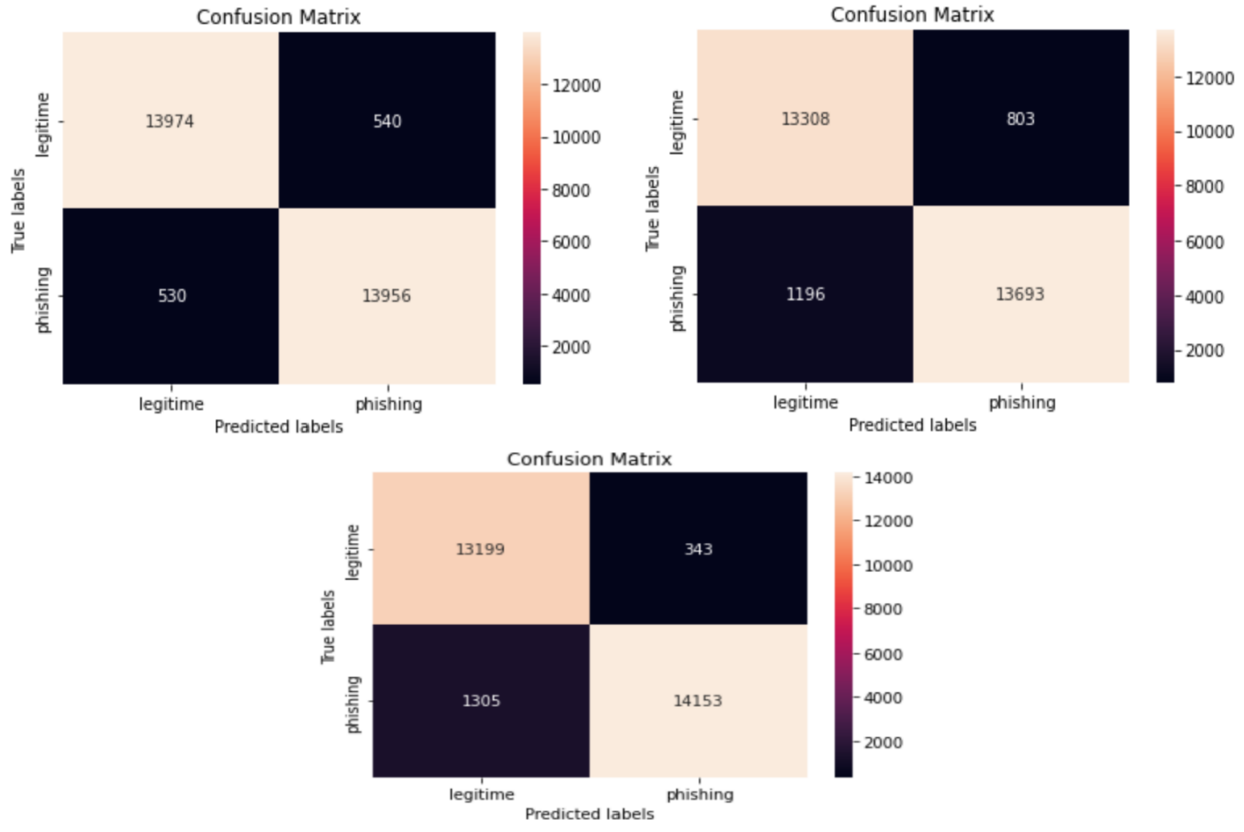

FIGURE 5.10 – Adam – Adamax – RMSprop : matrices de confusion avec sampling-strategy= 'all'.

5.5.2 Expérience 2 : Données équilibrées

En raison du problème de l'apprentissage insuffisant de la classe 1 qui s'est produit dans les dernières expériences, nous avons décidé d'essayer une nouvelle base de données « deuxième base de données » qui est équilibrée. Donc cette expérience repose sur deux cas d'études qui sont le premier un entrainement et test du modèle TCN et le second une application d'une validation croisée sur le même modèle et finalement nous allons étudier leurs performances à l'aide de mesures d'évaluation.

5.5.2.1 Cas d'étude 1 : Entrainement et Test

Cette étude est basée sur deux phases qui sont la phase d'entrainement de modèle et la phase de test en utilisant les mesures d'évaluation.

5.5.2.1.1 Phase d'Entrainement Après avoir défini le modèle, nous l'avons compilé et lancé sur l'ensemble d'entraînement. Pour la compilation, nous avons utilisé l'optimiseur populaire "Adam", avec une entropie croisée binaire comme fonction de perte, qui est largement utilisée dans les problèmes de classification. Nous avons également utilisé l'accuracy du modèle comme métrique. La figure 5.11 montre l'accuracy du modèle en fonction du nombre de fois que l'ensemble de données est complètement traité (époques), où l'on peut voir que l'accuracy de validation converge vers 99% et l'accuracy de l'entrainement converge vers 99%.

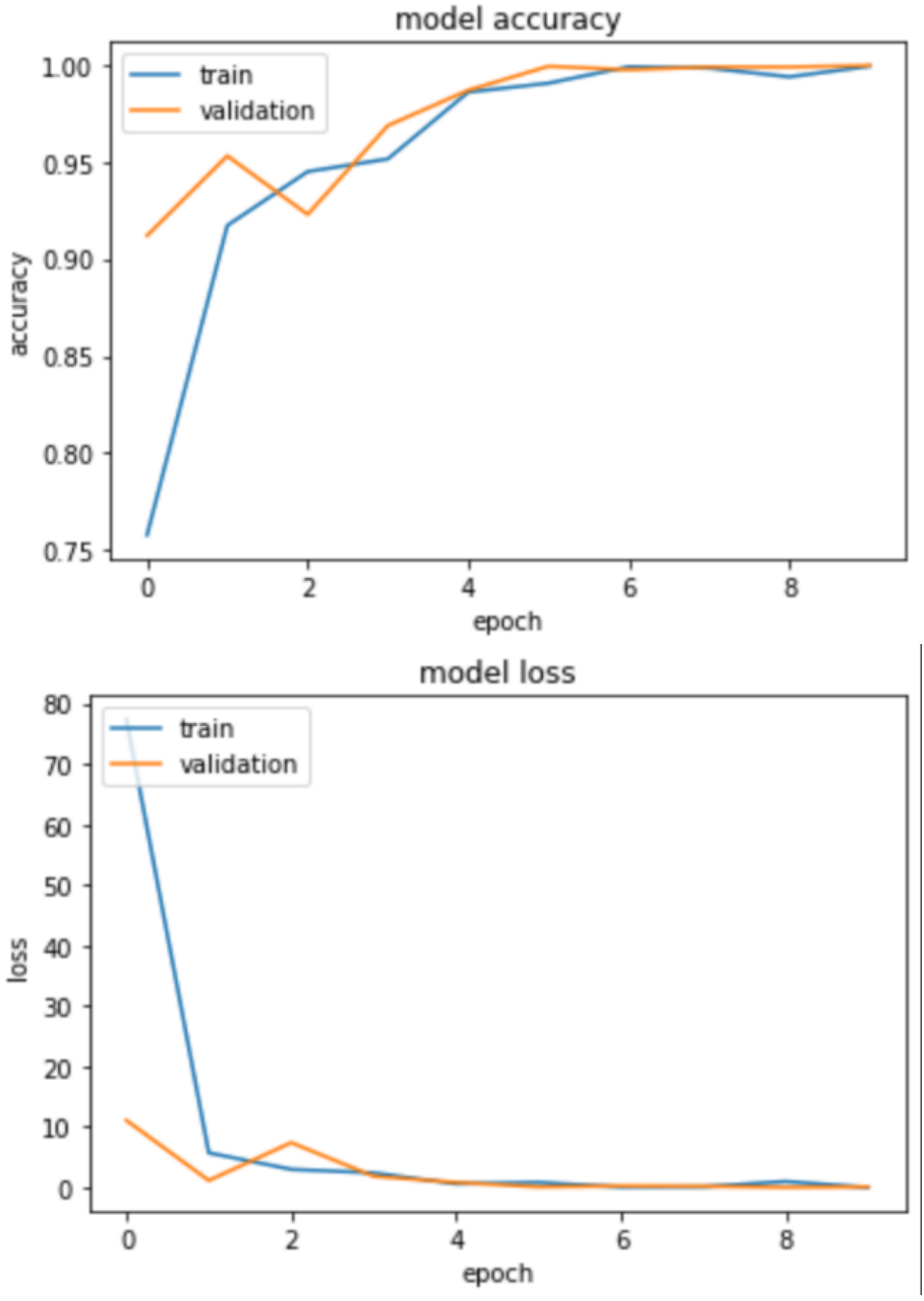

FIGURE 5.11 – Changements d'accuracy et de perte à travers différentes époques en utilisant TCN.

5.5.2.1.2 Phase de Test La Figure 5.12 représente la matrice de confusion obtenu par l'application de modèle TCN avec l'optimiseur Adam et la figure 5.13 représente le rapport de classification qui résulte cette expérience.

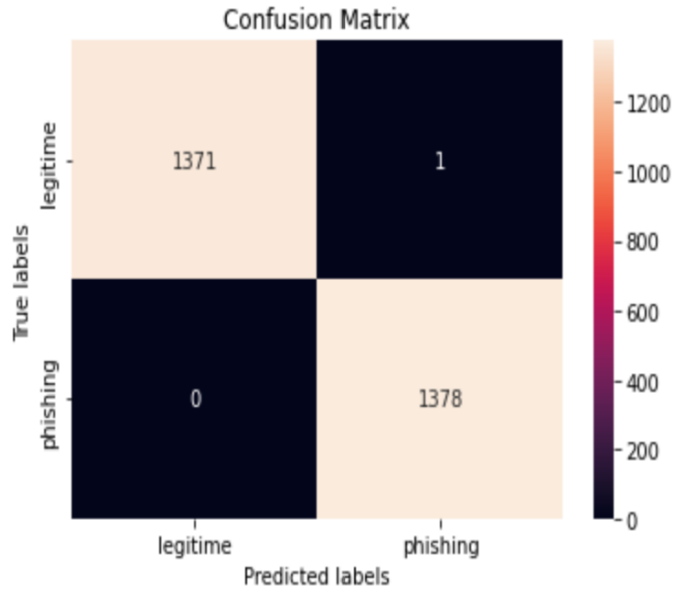

FIGURE 5.12 – Matrice de confusion du test avec accuracy de 99%

	precision	recall	f1-score	support
legitime	0.98	0.99	0.99	1341
phishing	0.99	0.98	0.99	1409
micro avg	0.99	0.99	0.99	2750
macro avg	0.99	0.99	0.99	2750
weighted avg	0.99	0.99	0.99	2750
samples avg	0.99	0.99	0.99	2750

FIGURE 5.13 – Rapport de classification du test.

Le classifieur a fait un total de 2750 prédictions pour les 2 classes : 0 et 1 respectivement, VP avec (1378), VN (1371), FP (1) et FN (0) qui est l'ensemble important de prédictions dans la classification de sécurité, plus les valeurs FN sont faibles donne le meilleur classifieur.

5.5.2.2 Cas d'Etude 2 : Validation croisée

Pour valider le modèle précèdent nous avons utilisé une validation croisée avec un nombre de FOLD (tour)=10. Nous avons obtenu $99,96\%$ d'accuracy, les figures 5.14, 5.15, 5.16 et 5.17 représentent la courbe AUC et la courbe ROC,Changements d'accuracy à travers 10-Fold , la matrice de confusion avec son rapport de classification.

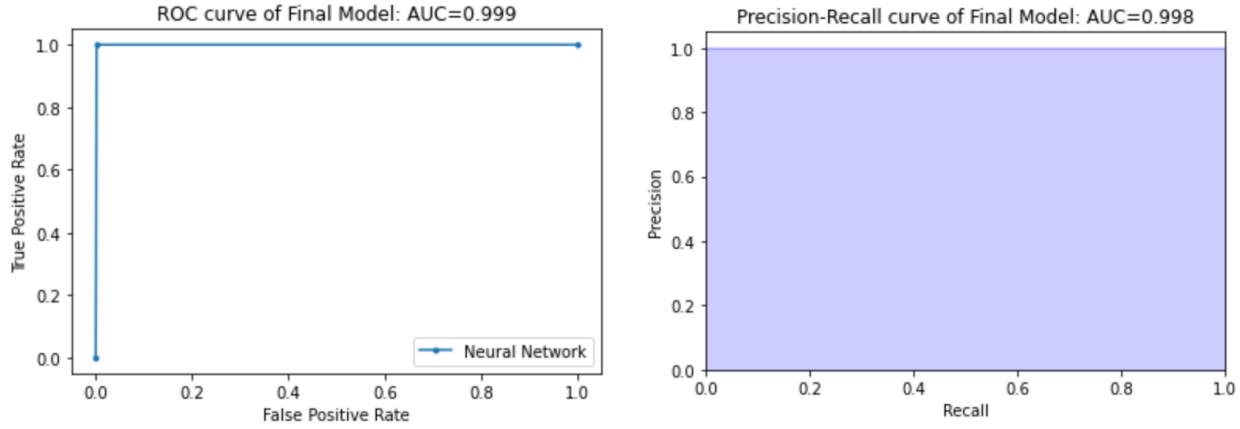

FIGURE 5.14 – Representation du courbe ROC (à gauche) et courbe AUC (à droite) pour 10-FOLD.

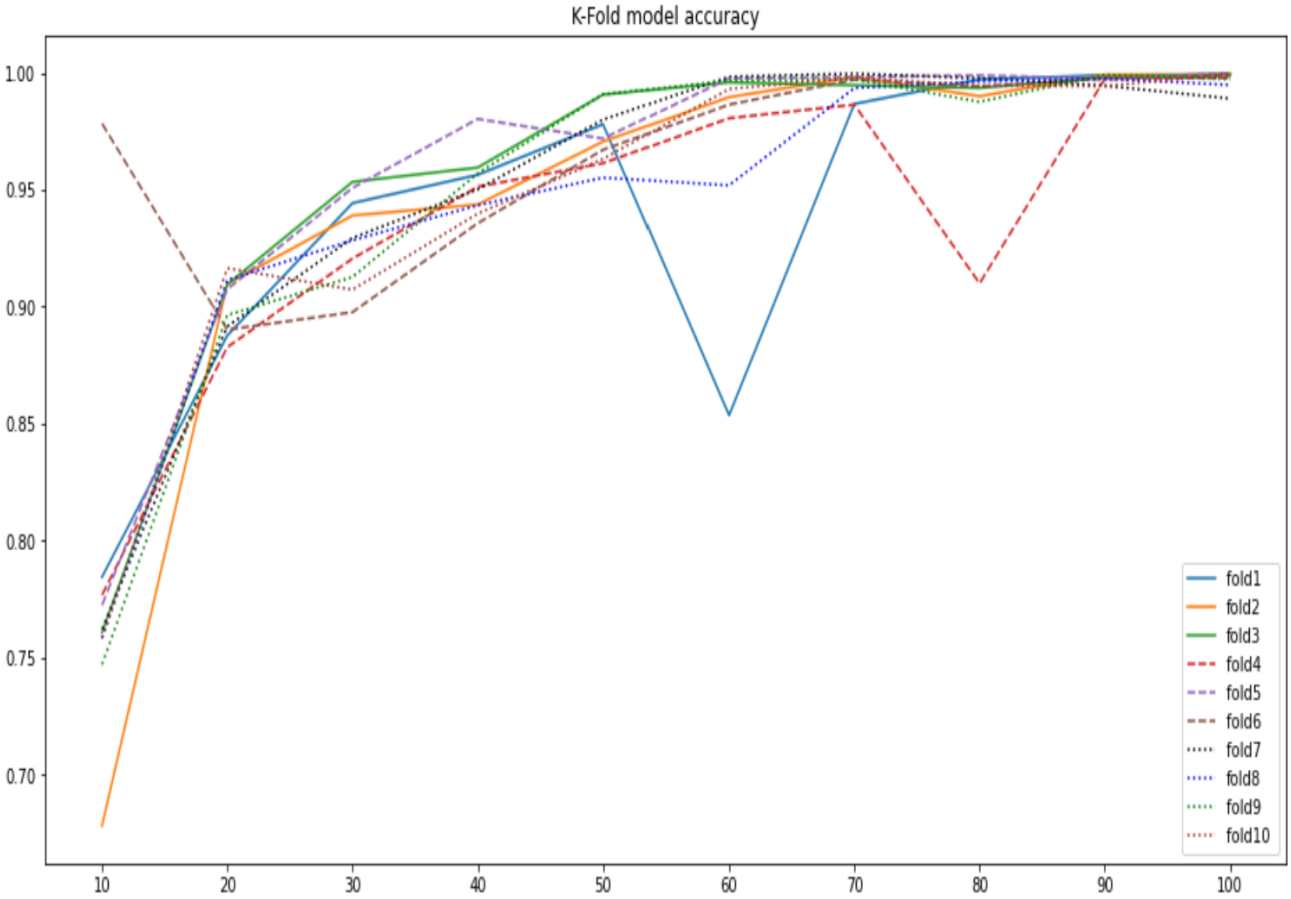

FIGURE 5.15 – Changements d'accuracy à travers 10-Fold en utilisant TCN.

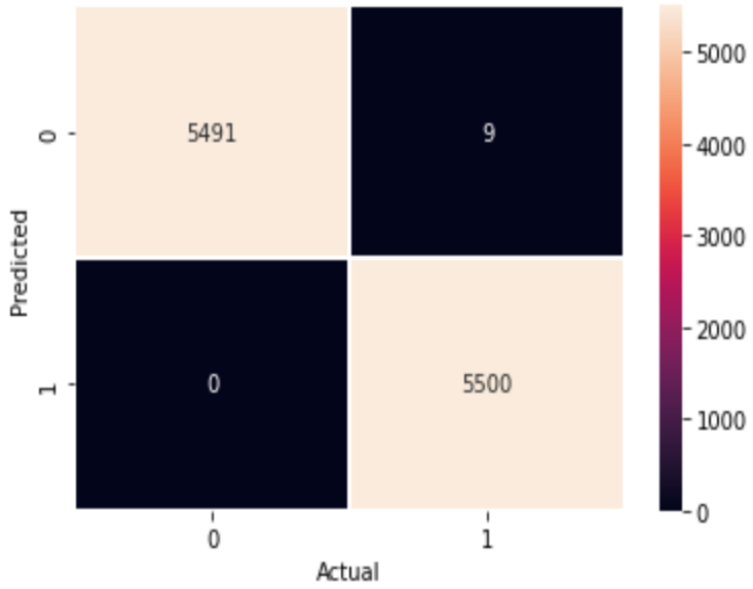

FIGURE 5.16 – Matrice de confusion du test avec une accuracy de 99% en utilisant 10-FOLD.

	precision	recall	f1-score	support
legitime	0.99	1.00	0.99	5500
phishing	1.00	0.99	0.99	5500
accuracy			0.99	11000
macro avg	0.99	0.99	0.99	11000
weighted avg	0.99	0.99	0.99	11000

FIGURE 5.17 – Rapport de classification du test en utilisant 10-FOLD.

5.5.3 Expérience 3 : Word2Vec Vs Glove Embedding

Nous avons décidé d'essayer une nouvelle base de données « Troisième base de données » qui est déséquilibrée mais contient juste l'URL avec son étiquette. Cette expérience repose sur deux cas d'études que nous allons étudier leurs performances à l'aide de mesures d'évaluation.

5.5.3.1 Cas d'Etude 1 : word2vec Embedding

Vu que le TCN prend en entré des valeurs entières nous avons appliqué un changement sur les URLs en utilisant le word2vec Embedding qui convertit les chaines de caractères en numéros. Nous avons obtenu $97,35\%$ d'accuracy. La figure 5.18 montre l'accuracy et la perte du modèle en fonction d'époques. Le tableau 5.6 et la figure 5.19 présentent les mesures d'évaluation : précision, rappel et f1-score et la matrice de confusion obtenus en appliquant les 3 optimiseurs

Adam, RMSprop, SGD et en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques =100.

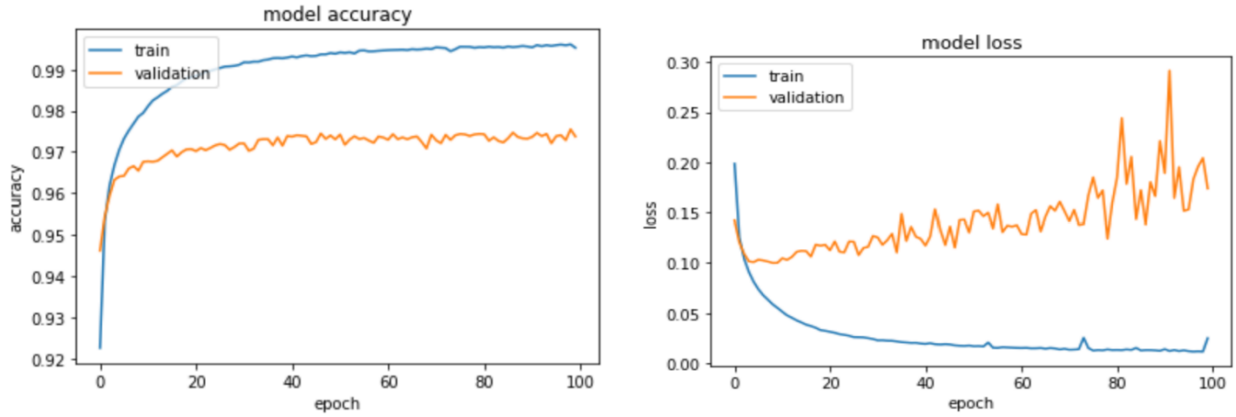

FIGURE 5.18 – Changements d'accuracy et de perte à travers des différentes époques en utilisant TCN avec word2vec Embedding.

TABLE 5.6 – mesures d'évaluation de TCN avec word2vec Embedding et en utilisant différents optimiseurs

Optimiseur	**Mesures D'évaluation**	**TCN Word2vec**
ADAM	Précision	0.97
	Rappel	0.97
	F1-score	0.97
RMSPROP	Précision	0,96
	Rappel	0,96
	F1-score	0,96
SGD	Précision	0,96
	Rappel	0,93
	F1-score	0,96

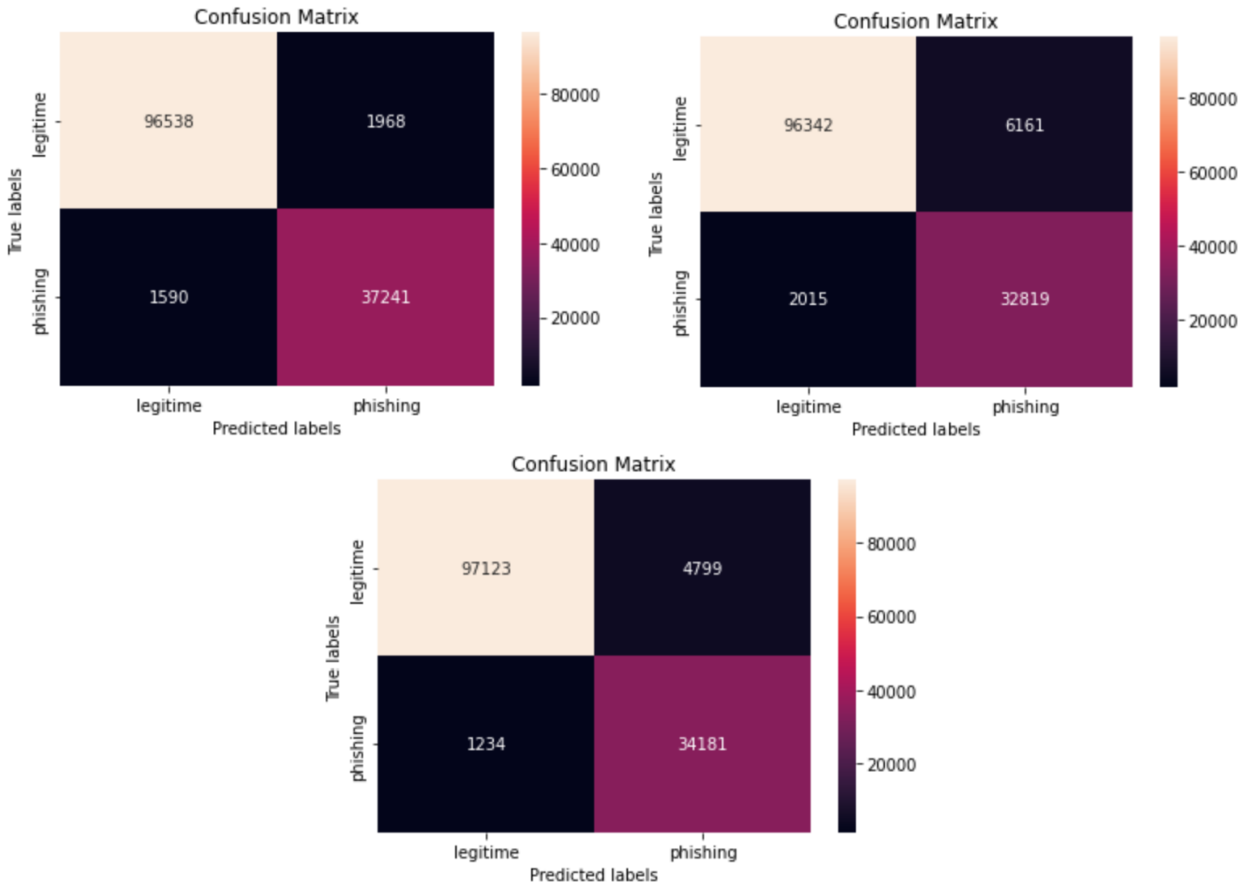

FIGURE 5.19 – Adam – SGD – RMSprop : matrices de confusion avec word2vec Embedding.

5.5.3.2 Cas d'Etude 2 : Glove Embedding

Dans ce cas nous avons obtenu 98, 03% d'accuracy. La figure 5.20 montre l'accuracy et la perte du modèle en fonction d'époques. Le tableau 5.7 et la figure 5.21 présentent les mesures d'évaluation : précision, rappel et f1-score et la matrice de confusion obtenus en appliquant les 3 optimiseurs Adam, RMSprop, SGD et en utilisant la fonction de perte d'entropie croisée binaire et un nombre d'époques =80.

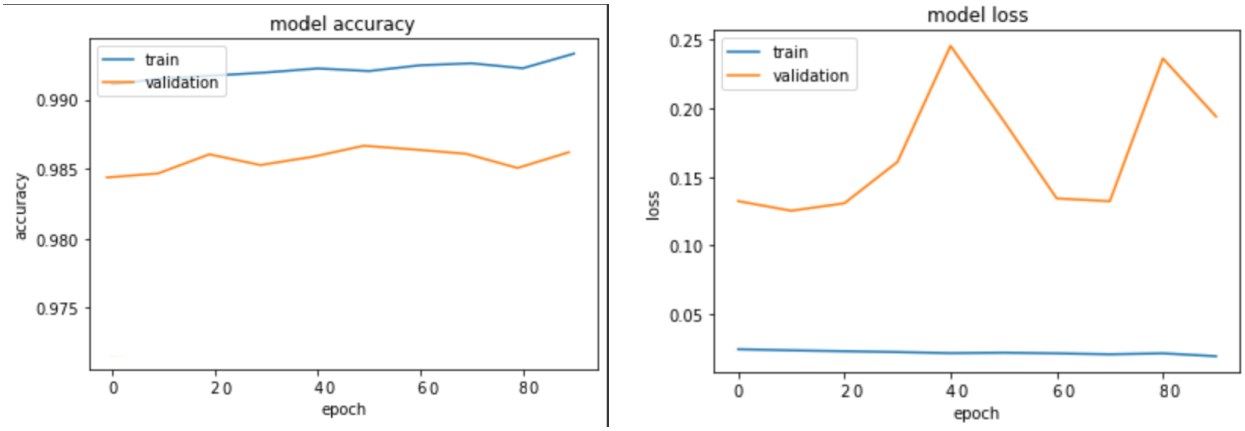

FIGURE 5.20 – Changements d'accuracy (à gauche) et de perte (à droite) à travers des différentes époques en utilisant TCN avec Glove Embedding.

TABLE 5.7 – mesures d'évaluation de TCN avec Glove Embedding et en utilisant différents optimiseurs

Optimiseur	**Mesures D'évaluation**	**TCN Glove**
ADAM	Précision	0.98
	Rappel	0.98
	F1-score	0.98
RMSPROP	Précision	0,97
	Rappel	0,96
	F1-score	0,96
SGD	Précision	0,95
	Rappel	0,94
	F1-score	0,94

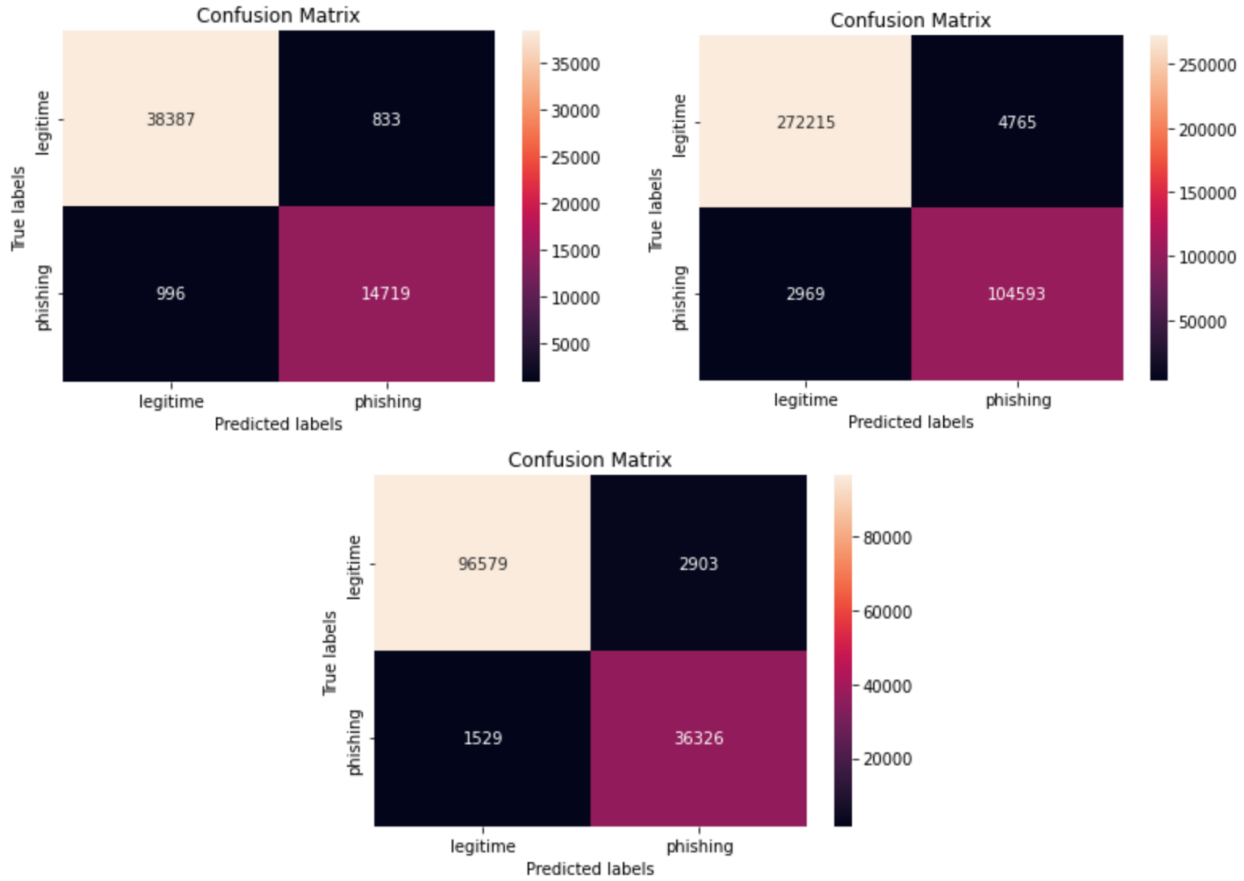

FIGURE 5.21 – RMSprop – SGD – Adam : matrices de confusion avec Glove Embedding.

5.5.4 Expérience 4 : Impact de Changement des paramètres du TCN sur les Trois Datasets

Dans ce cas nous avons changé plusieurs paramètres de modèle TCN afin de voir leurs impacts sur les résultats de prédiction. Donc cette expérience est composée de 5 cas d'études qui sont le changement de kernel-size, batch-size, dialations, learning-rate et finalement le dropout-rate.

5.5.4.1 Cas d'Etude 1 : Effet de kernel-size

Dans cette partie nous avons changé les valeurs de kernel-size qui était initialisé au début par 3 en 5 ensuit 8. Et nous avons appliqué le modèle TCN sur les trois bases de données en utilisant l'optimiseur Adam, la fonction de perte entropie croisée binaire et un nombre d'époques =100. La figure 5.22 représente le graphique à barres qui montre la différence entre les résultats obtenus.

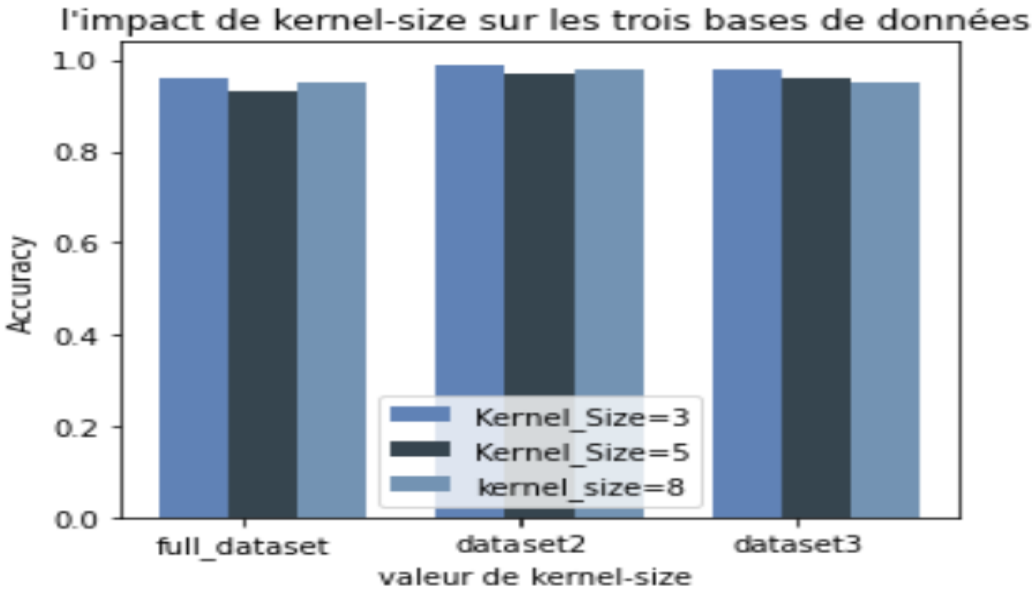

FIGURE 5.22 – histogramme d'impact du changement de kernel-size sur les trois datasets.

Nous remarquons que dans les trois datasets le kernel-size=3 a donné des meilleurs résultats.

5.5.4.2 Cas d'Etude 2 : Effet de batch-size

Dans ce cas nous avons changé les valeurs de batch-size par 64 ensuit 128. La figure 5.23 représente le graphique à barres en appliquant le modèle TCN sur les trois bases de données en utilisant l'optimiseur Adam, la fonction de perte entropie croisée binaire et un nombre d'époques=100.

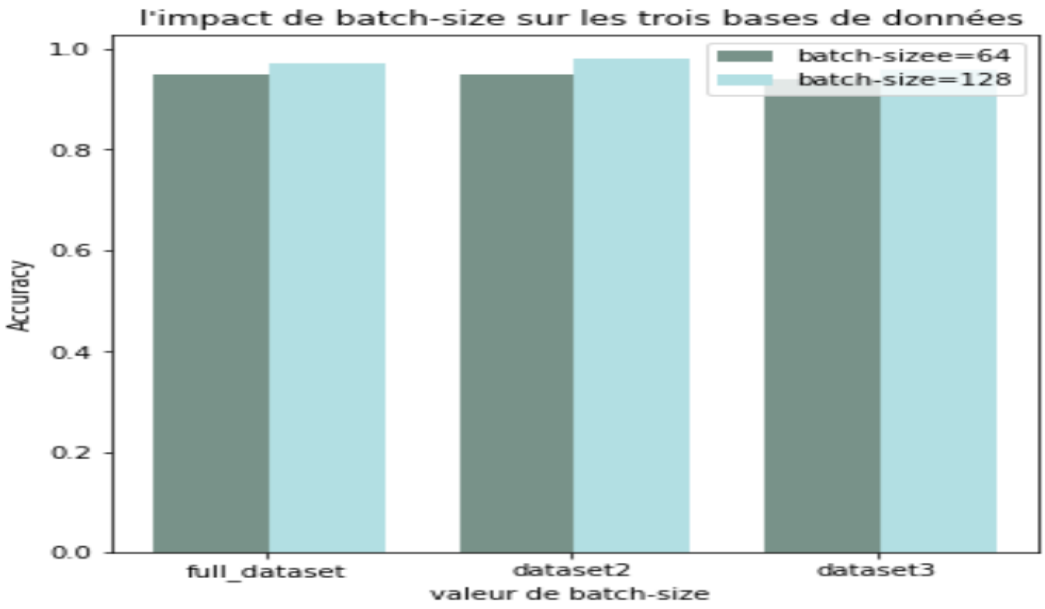

FIGURE 5.23 – histogramme d'impact du changement de batch-size sur les trois datasets.

Nous remarquons que batche-size=128 a donné des bons résultats par rapport au 64 donc à chaque fois qu'on augmente le nombre de ce paramètre on aura une meilleure précision.

5.5.4.3 Cas d'Etude 3 : Effet de dialations

Dans ce cas nous avons changé le vecteur de valeur de dialations qui était [1,2,4] par [8,16,32]. La figure 5.24 représente le graphique à barres en appliquant le modèle TCN sur les trois bases de données en utilisant l'optimiseur Adam, la fonction de perte entropie croisée binaire et un nombre d'époques =100,nous observons que le dialations= [8,16,32] a donné des faibles résultats.

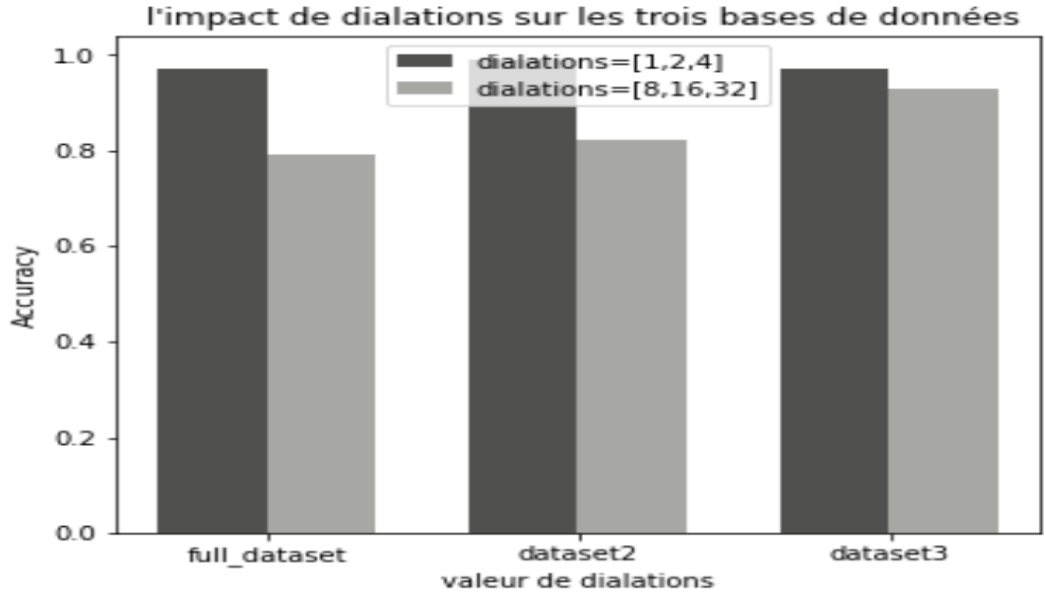

FIGURE 5.24 – histogramme d'impact du changement de dialations sur les trois datasets.

5.5.4.4 Cas d'Etude 4 : Effet de learning-rate

Dans cette partie nous avons changé les valeurs de learning-rate qui par une fois 0,001 et l'autre fois 0,0005. La figure 5.25 représente le graphique à barres en appliquant le modèle TCN sur les trois bases de données en utilisant l'optimiseur Adam avec les valeurs de learning-rate, la fonction de perte entropie croisée binaire et un nombre d'époques =100.

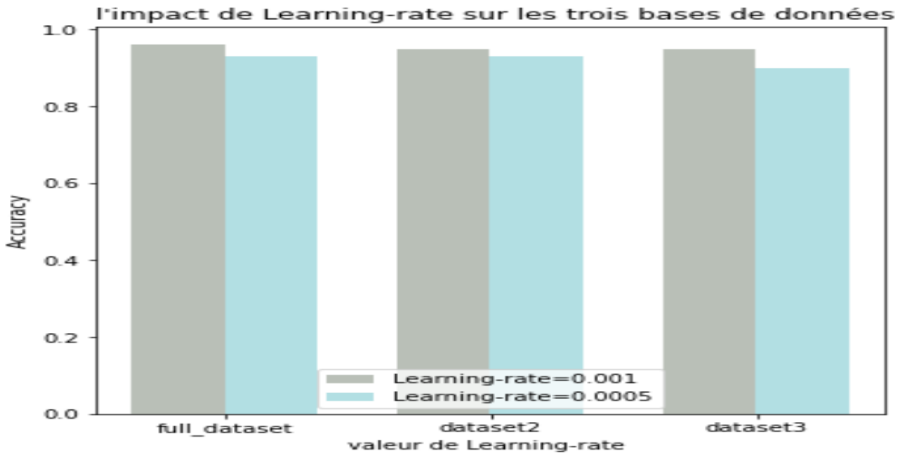

FIGURE 5.25 – histogramme d'impact du changement de leraning-rate sur les trois datasets.

Nous remarquons qu'à chaque fois le learning-rate baisse les résultats seront faible.

5.5.4.5 Cas d'Etude 5 : Effet de dropout-rate

Dans ce cas nous avons changé les valeurs de dropout-rate qui était 0,1 par la plage de valeurs [0.2 ,0.3, 0,4] les résultats obtenus sont représentés dans la figure 5.26 qui est un graphique à barres en appliquant le modèle TCN sur les trois bases de données en utilisant l'optimiseur Adam, la fonction de perte entropie croisée binaire et un nombre d'époques =100,nous remarquons que le 1er dropout a donné des bons résultats pour les trois datasets et à chaque fois qu'on augmente la valeur de dropout les résultats devient faibles.

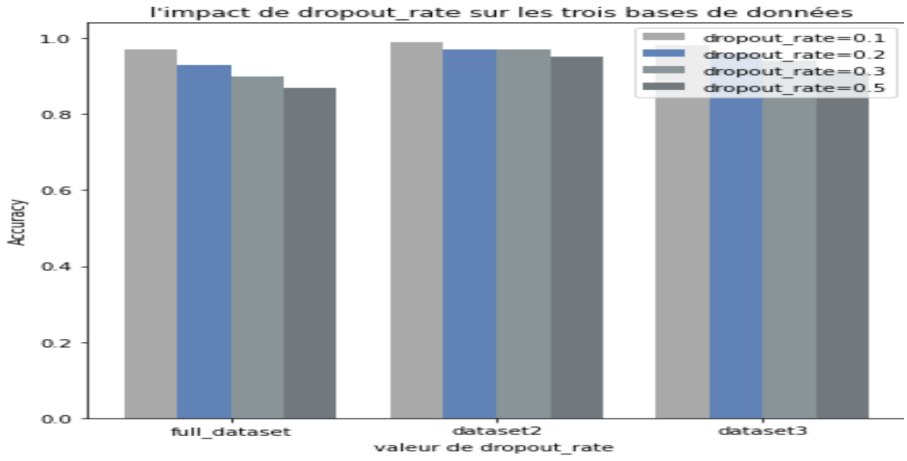

FIGURE 5.26 – histogramme d'impact du changement de dropout-rate sur les trois datasets.

5.5.5 Expérience 5 : Test Final

Pour tester la validation des résultats obtenus dans les expériences précédentes nous avons mené un test sur 100 (51 légitimes et 49 phishings) nouvelles URLs récupéré à partir du site Phistank. Nous avons fait une prédiction sur ces URLs, la précision obtenus est 99% les figures 5.27 et 5.28 représentent la matrice de confusion et le rapport de classification obtenus

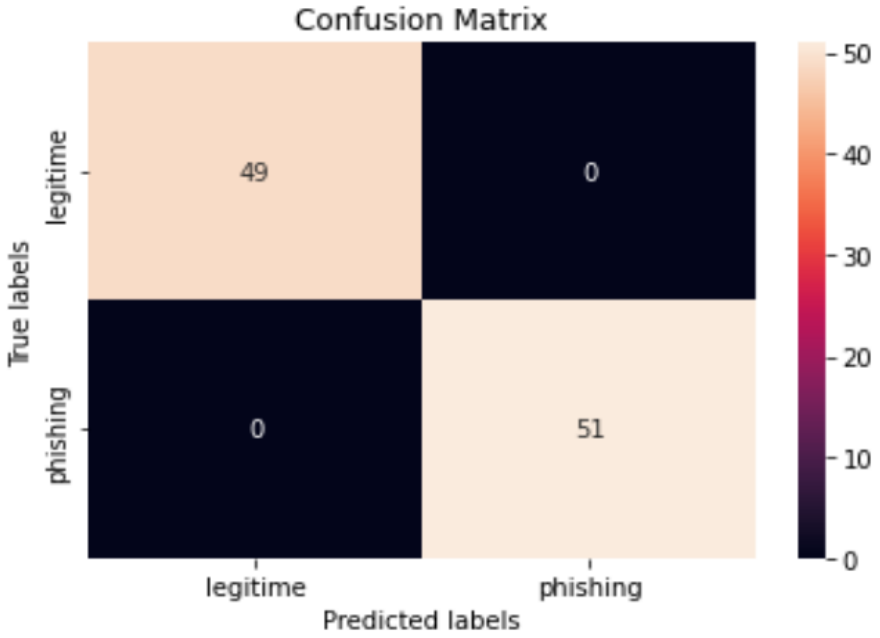

FIGURE 5.27 – matrice de confusion des résultats du test final.

	precision	recall	f1-score	support
legitime	0.99	1.00	1.00	49
phishing	1.00	0.99	1.00	51
accuracy			1.00	100
macro avg	1.00	1.00	1.00	100
weighted avg	1.00	1.00	1.00	100

FIGURE 5.28 – Rapport de classification des résultats du test final.

Pour bien évalué les performances de notre modèle certain nombre de modèles de classification, notamment réseau neuronal convolutif(CNN), réseau neuronal récurrent(RNN),mémoire à court terme(LSTM),réseau neuronal convolutif avec mémoire à court terme(CNN+LSTM),

sont appliqués sur dataset-full à des fins de comparaison, leurs résultats présenté dans la table 5.8.

TABLE 5.8 – Analyse comparative entre notre modèle et d'autres architectures d'apprentissage profond

Algos Sur Dataset-full	**Mesures d'évaluation**		
	Précision	**Rappel**	**F1-Score**
CNN	0,91	0.90	0.90
RNN	0,92	0,92	0,92
LSTM	0,85	0,84	0,84
CNN+LSTM	0,93	0,92	0,92
TCN	0,97	0,97	0,97

5.6 Discussion des Résultats

L'objectif principal de ce projet était de trouver un meilleur modèle TCN pour la détection des URLs de phishing. À cette fin, nous avons réalisé cinq séries d'expériences pour analyser leur comportement en utilisant les bases de données dataset-Full, dataset-small, dataset2 et dataset3. Nous avons examiné différentes techniques en faisant varier leurs paramètres. Nous avons choisi la précision, le rappel, le F-score et l'accuracy comme mesures d'évaluation des performances et nous avons fondé notre analyse et notre discussion sur de nombreux tests statistiques. De cette études expérimentales, nous pouvons tirer les conclusions suivantes :

- La base de données dataset2 donne de meilleurs résultats que les autres bases car elle est équilibrée en termes d'instance.
- Le oversampling sur les données dataset-full a donné des bons résultats qu'undersampling sur la même dataset.
- Les paramètres par défaut du modèle TCN ont donné des résultats élevés par rapport à leurs changements sur les trois datasets .
- La fonction " SMOTE " lorsqu'elle règle son paramètre " sampling-strategy " sur " auto " a donné un meilleur résultat par rapport au " all ".
- Le modèle TCN avec le GLOVE a donné des résultats remarquables la précision a atteint 98% tandis qu'avec le Word2vec a donné 97% sur le même ensemble de données.

5.7 Déploiement du modèle

Dans cette partie nous allons présenter notre système, du côté interface graphique, La figure montre l'interface au début de lancement du site web.

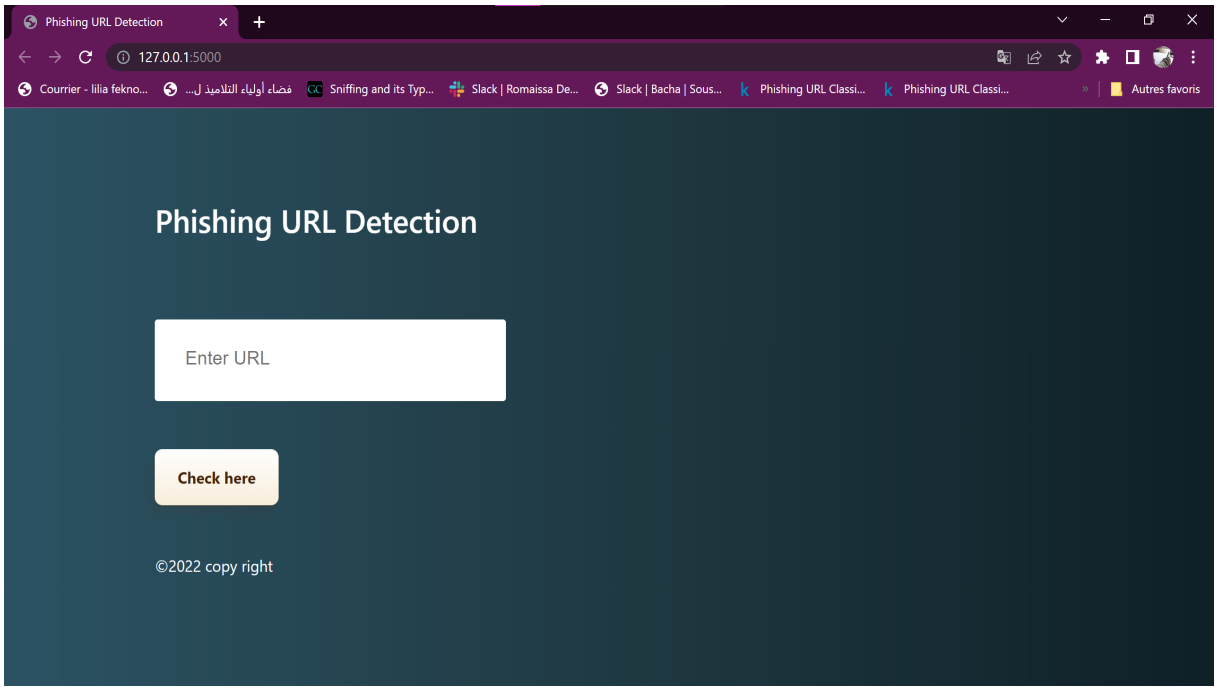

FIGURE 5.29 – interface graphique du Premier lancement de l'application.

A partir de cette interface l'utilisateur peut choisir de tester son url on le collant dans le champ « Entrer url » pour faire la prédiction il doit cliquer sur le bouton « check here » ensuite les résultats seront affichés comme montrent les figures ci-dessous.

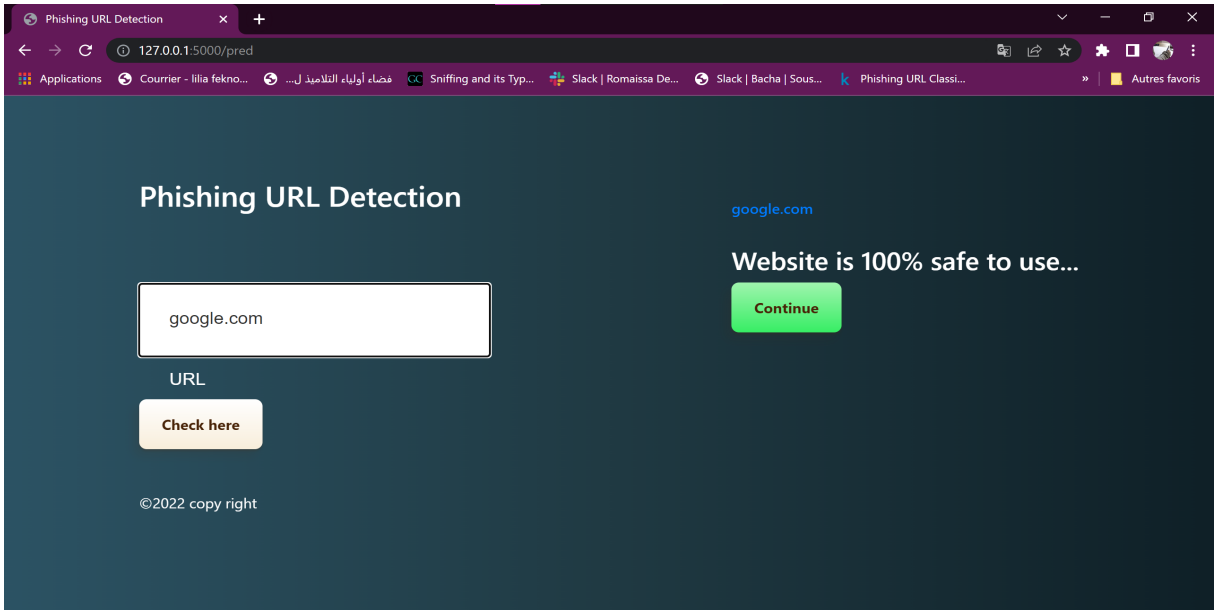

FIGURE 5.30 – interface graphique du test d'URL légitime.

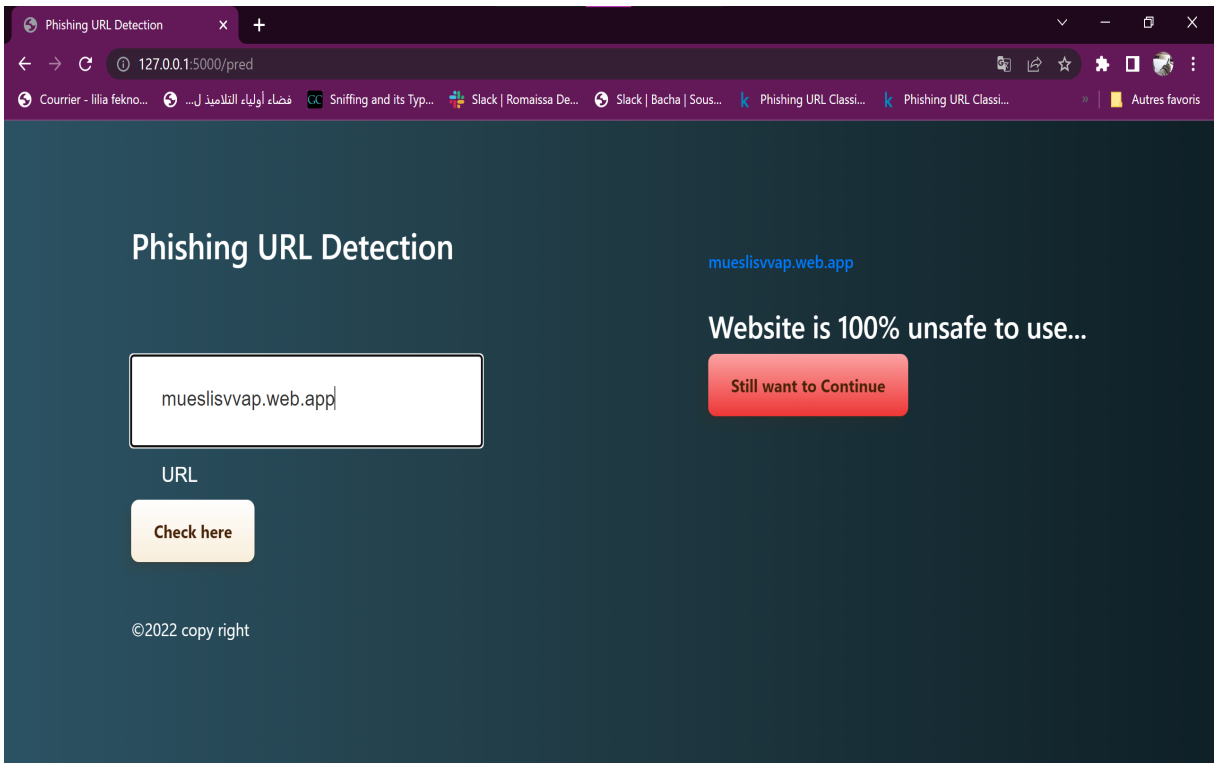

FIGURE 5.31 – interface graphique du test d'URL de phishing.

5.8 Conclusion

Dans ce chapitre Nous avons montré que les réseaux convolutifs temporels sont capables d'apprendre automatiquement les motifs inhérents aux données séquentielles et que les TCN peuvent donc être utilisés pour la détection des anomalies. Notre approche basée sur les réseaux convolutionnels temporels fonctionne bien sur trois ensembles de données ouvertes du monde réel.

Conclusion

Les sites Web d'hameçonnage constituent depuis longtemps une menace sérieuse pour la cybersécurité. Depuis des décennies, de nombreux chercheurs se consacrent à développer de nouvelles techniques pour détecter automatiquement les sites Web de phishing. Alors que des solutions de pointe peuvent atteindre des performances supérieures, ils nécessitent une ingénierie manuelle substantielle des fonctionnalités et ne sont pas aptes à détecter les nouvelles tentatives d'hameçonnage. Par conséquent, développer des techniques capables de détecter le phishing automatiquement et gérer les attaques de phishing de type zero-day rapidement reste un défi ouvert dans ce domaine. Dans ce travail, nous avons proposé TCN, une approche basée sur l'apprentissage profond pour détection des URL d'hameçonnage.

Il est bien connu qu'une bonne détection des URLs de phishing devrait avoir de bonnes performances en temps réel tout en assurant une bonne précision et un faible taux de faux positifs. Notre approche du TCN proposée est conforme à cette idée.

Notre étude a exploré la possibilité de distinguer les légitimes URL à partir d'URL malveillantes en combinant les deux technologies d'incorporation de mots et TCN. Pour évaluer la proposition trois ensembles de données différents ont été utilisés, y compris plus de 192 500 URL de la liste de domaines malveillants et plus de 456 400 URL légitimes.

La méthode proposée a donné une grande précision de classification de $99,96\%$. Par conséquent, la technologie d'intégration de mots peut être utilisée pour intégrer suffisamment de connaissances sémantiques dans les URL malveillantes en vecteurs distribués, puis utilisez des modèles de réseaux de neurones (TCN) pour la classification. Il ne nécessite qu'un traitement simple de l'URL d'origine et ne repose pas sur toute autre fonctionnalité complexe ou experte. Comparé à d'autres méthodes, notre méthode a un meilleur effet de détection.

Bien que le modèle proposé fonctionne bien, de plus des améliorations sont encore nécessaires ainsi que d'autres études pour améliorer l'ensemble du système Comme :

- Modifier notre modèle structure basée sur d'autres classes malveillantes pour effectuer plusieurs classifications.
- Optimiser notre modèle par rapport au temps de réponse pour l'intégrer dans les appareils mobiles et systèmes distribués.
- Dévlopper un plugin à notre application .

Bibliographie

[Abdelhamid et al., 2014] Abdelhamid, N., Ayesh, A., Thabtah, F. (2014). Expert Systems with Applications Phishing detection based Associative Classification data mining. Expert Systems With Applications, 41(13), 5948-5959. https ://doi.org/10.1016/j.eswa.2014.03.019

[Aburrous et al., 2010] Aburrous, M., Hossain, M. A., Thabtah, F. (2010). Predicting Phishing Websites using Classification Mining Techniques with Experimental Case Studies.Seventh International Conference on Information Technology : New Generations, 2010, pp. 176-181, doi : 10.1109/ITNG.2010.117.

[Abutair et al., 2019] Abutair, H., Belghith, A., AlAhmadi, S. (2019). CBR-PDS : a case-based reasoning phishing detection system. Journal of Ambient Intelligence and Humanized Computing, 10(7), 2593-2606. https ://doi.org/10.1007/s12652-018-0736-0

[Activity et Report, 2015] Activity, P., Report, T. (2015). Phishing Activity Trends Report 4 Quarter. December 2014.

[Activity et Report, 2020] Activity, P., Report, T. (2020). Phishing Activity Trends Report 4 Quarter. February 2021.

[Afroz et Greenstadt, 2011] Afroz, S., Greenstadt, R. (2011). PhishZoo : Detecting Phishing Websites By Looking at Them.In 2011 IEEE fifth international conference on semantic computing (pp. 368-375). IEEE. https ://doi.org/10.1109/ICSC.2011.52

[Al-Ahmadi et Lasloum, 2020] Al-Ahmadi, S., Lasloum, T. (2020). PDMLP : Phishing Detection using Multilayer Perceptron. International Journal of Network Security Its Applications, 12(3), 59-72. https ://doi.org/10.5121/ijnsa.2020.12304

[Aljofey et al., 2020] Aljofey, A., Jiang, Q., Qu, Q., Huang, M., Niyigena, J. P. (2020). An effective phishing detection model based on character level convolutional neural network from URL. Electronics (Switzerland), 9(9), 1-24. https ://doi.org/10.3390/electronics9091514

[Alkhalil et al., 2021] Alkhalil, Z., Hewage, C., Nawaf, L., Khan, I. (2021). Phishing Attacks : A Recent Comprehensive Study and a New Anatomy. 3(March), 1-23. https ://doi.org/10.3389/fcomp.2021.563060

[Aytug, 2022] Aytug, O. (2022). Bidirectional convolutional recurrent neural network architecture with group-wise enhancement mechanism for text sentiment classification.Journal of King Saud University-Computer and Information Sciences, 34(5), 2098-2117 https ://doi.org/10.1016/j.jksuci.2022.02.025

[Afroz et Greenstadt,2011] S. Afroz and R. Greenstadt, "PhishZoo : Detecting Phishing Websites by Looking at Them," 2011 IEEE Fifth International Conference on Semantic Computing, 2011, pp. 368-375, doi : 10.1109/ICSC.2011.52.

[Bahnsen et al., 2017] Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J., Gonzalez, F. A. (2017). Classifying phishing URLs using recurrent neural networks.In 2017 APWG symposium on electronic crime research (eCrime) (pp. 1-8). IEEE. https ://doi.org/10.1109/ECRIME.2017.7945048

[Banu et al., 2013] Banu, M. N., Engineering, M. A. M. C., Mohamed, J., Autonomous, C. (2013). A Comprehensive Study of Phishing Attacks.International Journal of Computer Science and Information Technologies, 4(6), 783-786.

[Babu et al., 2020] Babu, D. V., Karthikeyan, C., Kumar, A. (2020, December). Performance analysis of cost and accuracy for whale swarm and rmsprop optimizer. In IOP Conference Series : Materials Science and Engineering (Vol. 993, No. 1, p. 012080). IOP Publishing.

[Bai et al., 2018] Bai, S., Kolter, J. Z., Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.arXiv preprint arXiv :1803.01271.

[Bisong , 2019] Bisong, E. (2019). Building Machine Learning and Deep Learning Models on Google Cloud Platform.(pp. 59-64). Berkeley, CA : Apress.

[Buja et Stuetzle, 2005] Buja, A., Stuetzle, W. (2005). Loss Functions for Binary Class Probability Estimation and Classification : Structure and Applications.Working draft, November, 3, 13.

[Chiew et al., 2018] Chiew, K. L., Choo, J. S. F., Sze, S. N., Yong, K. S. C. (2018). Leverage Website Favicon to Detect Phishing Websites. Security and Communication Networks, 2018. https ://doi.org/10.1155/2018/7251750

[Choi et al., 2019] Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., Dahl, G. E. (2019). On Empirical Comparisons of Optimizers for Deep Learning.arXiv preprint arXiv :1910.05446.

[Cranor et al., 2006] Cranor, L., Egelman, S., Hong, J., Zhang, Y., Cranor, L., Egelman, S., Hong, J., Zhang, Y. (2006). Phinding Phish : An Evaluation of Anti-Phishing Toolbars.In NDSS (pp. 1-19).

[Davis et Goadrich, 2006] Davis, J., Goadrich, M. (2006). The Relationship Between Precision-Recall and ROC Curves.In Proceedings of the 23rd international conference on Machine learning (pp. 233-240).

[De et Hand, 2010] De, L., Hand, D. J. (2010). L'incohérence de l'aire sous la courbe ROC, que faire à ce propos.Revue Modulad, 74(42).

[Dinler et Sahin, 2021] Dinler, Ö. B., Şahin, C. B. (2021). Prediction of Phishing Web Sites with Deep Learning Using WEKA Environment WEKA Ortamını Kullanarak Derin Öğrenme ile Kimlik Hırsızı Web Sitelerinin Tahmini. 24, 35-41. https ://doi.org/10.31590/ejosat.901465

[Dumais., 1996] Dumais, S. (1996). A Bayesian Approach to Filtering Junk E-Mail.In Learning for Text Categorization : Papers from the 1998 workshop (Vol. 62, pp. 98-105).

[Hai et Hwang, 2018] Hai, Q. T., Hwang, S. O. (2018). Detection of malicious URLs based on word vector representation and ngram. Journal of Intelligent and Fuzzy Systems, 35(6), 5889-5900. https ://doi.org/10.3233/JIFS-169831

[Hong, 2012] Hong, B. J. (2012). The State of Phishing Attacks.Communications of the ACM, 55(1), 74-81. https ://doi.org/10.1145/2063176.2063197

[J. Hong et al., 2006] Hong, J., Hong, J., Hong, J. (2006). Phinding Phish : An Evaluation of Anti-Phishing Toolbars Related papers.Phinding Phish : An Evaluation of Anti-Phishing Toolbars. In NDSS (pp. 1-19).

[Howard et Wang, 2012] Howard, A. G., Wang, W. (2012).Mobilenets : Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv :1704.04861.

[Hu., 2019] Hu, daikui shouren. (1996). an object neural network language. In [Proceedings] 1991 IEEE International Joint Conference on Neural Networks (pp. 1606-1611). IEEE.

[Huang et al., 2017] Huang, H.-C., Zhang, Z.-K., Cheng, H.-W., Shieh, S. W. (2017). Web Application Security : Threats, Countermeasures, and Pitfalls. Computer, 50(6), 81-85. https ://doi.org/10.1109/MC.2017.183

[Huang et al., 2019] Huang, Y., Yang, Q., Qin, J., Wen, W. (2019). Phishing URL detection via CNN and attention-based hierarchical RNN. Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019, 112-119. https ://doi.org/10.1109/TrustCom/BigDataSE.2019.00024

[Craig et al., 2017] Craig, J. P., Nichols, K. K., Akpek, E. K., Caffery, B., Dua, H. S., Joo, C.-K., Liu, Z., Nelson, J. D., Nichols, J. J., Tsubota, K., Stapleton, F. (2017). TFOS DEWS II Definition and Classification Report. The Ocular Surface, 15(3), 276-283. https ://doi.org/https ://doi.org/10.1016/j.jtos.2017.05.008

[Jain et Gupta, 2016] Jain, A. K., Gupta, B. B. (2016). A novel approach to protect against phishing attacks at client side using auto-updated white-list. Eurasip Journal on Information Security, 2016(1). https ://doi.org/10.1186/s13635-016-0034-3

[Kavya, 2018] Kavya, P. (2018). An Efficient Machine Learning based Algorithm for Preventing Phishing Websites. 5(12), 10-13. K

[Ketkar, 2017] Ketkar, N. (2017). Introduction to Keras BT - Deep Learning with Python : A Hands-on Introduction (N. Ketkar (éd.) ; p. 97-111). Apress. https ://doi.org/10.1007/978-1-4842-2766-47

[Khonji et al., 2013] Khonji, M., Iraqi, Y., Jones, . Phishing detection : A literature survey. IEEE Communications Surveys and Tutorials, 15(4), 2091-2121. https ://doi.org/10.1109/SURV.2013.032213.00009

[Lara-benitez et al., 2020] Lara-benítez, P., Carranza-garcía, M., Luna-romera, J. M. (2020). applied sciences Temporal Convolutional Networks Applied to Energy-Related Time Series Forecasting. 1-17. https ://doi.org/10.3390/app10072322

[Le et al., 2018] Le, H., Pham, Q., Sahoo, D., Hoi, S. C. H. (2018). URLNet : Learning a URL Representation with Deep Learning for Malicious URL Detection. arXiv preprint arXiv :1802.03162. http ://arxiv.org/abs/1802.03162

[Leonov et al., 2022] Leonov, P. Y., Vorobyev, A. V, Ezhova, A. A. (2022). The Main Social Engineering Techniques Aimed at Hacking Information Systems. In 2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT) (pp. 0471-0473). IEEE.

[Llugsi et al., 2021] Llugsi, R., El Yacoubi, S., Fontaine, A., Lupera, P. (2021, October). Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Fore-

cast based on Neural Networks for the Andean city of Quito. In 2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM) (pp. 1-6). IEEE.

[Li, 2018] Li, Z. (2018).An exponential learning rate schedule for deep learning. arXiv preprint arXiv :1910.07454.

[Lipton et Narayanaswamy, 2014] Lipton, Z. C., Elkan, C., Narayanaswamy, B. (2014). Thresholding classifiers to maximize F1 score. arXiv preprint arXiv :1402.1892.

[Lutz, 2013] Lutz, M. (2013). Learning Python : Powerful object-oriented programming. " O'Reilly Media, Inc.".

[Mourtaji et al., 2021] Mourtaji, Y., Bouhorma, M., Alghazzawi, D., Aldabbagh, G., Alghamdi, A. (2021). Hybrid Rule-Based Solution for Phishing URL Detection Using Convolutional Neural Network.Wireless Communications and Mobile Computing, 2021.

[Li et al., 2018] Li, Y., Yang, T. (2018). Word embedding for understanding natural language : a survey. In Guide to big data applications (pp. 83-104). Springer, Cham. https ://doi.org/10.1007/978-3-319-53817-4

[Mohammad et al., 2014] Mohammad, R. M., Thabtah, F., Mccluskey, L. (2014). Intelligent rule-based phishing websites classification. 8(July 2013), 153-160. https ://doi.org/10.1049/iet-ifs.2013.0202

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality.Advances in neural information processing systems, 26.

[Mohammad et al., 2015] Mohammad, R. M., Thabtah, F., Mccluskey, L. (2015). ScienceDirect Tutorial and critical analysis of phishing websites methods. Computer Science Review, 17, 1-24. https ://doi.org/10.1016/j.cosrev.2015.04.001

[Murdoch et al., 2019] Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-asl, R., Yu, B. (2019). Definitions , methods , and applications in interpretable machine learning.Proceedings of the National Academy of Sciences, 116(44), 22071-22080. https ://doi.org/10.1073/pnas.1900654116

[Ningxia, 2011] Ningxia Zhang, Y. Y. (2011). Phishing Detection Using Neural Network. 95. http ://cs229.stanford.edu/proj2012/ZhangYuan-PhishingDetectionUsingNeuralNetwork.pdf

[Nwanpka et al., 2018] Nwankpa, C. E., Ijomah, W., Gachagan, A., Marshall, S. (2018). Activation Functions :Comparison of Trends in Practice and Research for Deep Learning.arXiv preprint arXiv :1811.03378.

[Pedregosa et al., 2011] Pedregosa, F., Weiss, R., Brucher, M. (2011). Scikit-learn : Machine Learning in Python. 12, 2825-2830.

[Pentland et Liu, 1999] Pentland, A., Liu, A. (1999). Modeling and Prediction of Human Behavior, 11(1), 229-242.

[Patil et Dhage, 2019] Patil, S., Dhage, S. (2019). A Methodical Overview on Phishing Detection along with an Organized Way to Construct an Anti-Phishing Framework. In 2019 5th International Conference on Advanced Computing Communication Systems (ICACCS) (pp. 588-593).IEEE.

[Rao et Ali., 2015] Rao, R. S., Ali, S. T. (2015). PhishShield : A Desktop Application to Detect Phishing Webpages through Heuristic Approach. Procedia Computer Science, 54, 147-156. https ://doi.org/10.1016/j.procs.2015.06.017

[Rasymas et Dovydaitis, 2020] Rasymas, T., Dovydaitis, L. (2020). Detection of phishing URLs by using deep learning approach and multiple features combinations. Baltic Journal of Modern Computing, 8(3), 471-483. https ://doi.org/10.22364/BJMC.2020.8.3.06

[Reddy et al., 2011] Reddy, V. P., Radha, V., Jindal, M. (2011). Client Side protection from Phishing attack. International Journal of Advanced Engineering Sciences and Technologies, 3(1), 39-45.

[Rekouche , 1995] Rekouche, K. (1995). Early Phishing.arXiv preprint arXiv :1106.4692.

[Rofifah, 2020] Rofifah, D. (2020). Paper Knowledge . Toward a Media History of Documents. Paper Knowledge . Toward a Media History of Documents, 12-26.

[Sahingoz et al., 2019] Sahingoz, O. K., Buber, E., Demir, O., Diri, B. (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications, 117(January 2019), 345-357. https ://doi.org/10.1016/j.eswa.2018.09.029

[Salsabil et Amaria, 2017] Salsabil, L., Amaria, S. (2017). Étude Des Techniques D' Apprentissage Semi- Supervisé Par Regroupement.

[Salve et al., 2015] Salve, A., Salgar, M., Sarode, A., Sardal, T., Said, P. A. (2015). BAIT Alarm : Anti-Phishing Using Visual Similarities. 66-69.

[Sami et Hacene, 2020] Sami, A., Hacene, I. (2020). Extraction of areas of interest in a document (p. 27).

[Simundic , 2009] Šimundić A. M. (2009). Measures of Diagnostic Accuracy : Basic Definitions. EJIFCC, 19(4), 203–211.

[Scherer et al., 2010] Scherer, D., Müller, A., Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. Lecture Notes in Computer

Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6354 LNCS(PART 3), 92-101. https ://doi.org/10.1007/978-3-642-15825-410

[Shahriar et Zulkernine, 2012] Shahriar, H., Zulkernine, M. (2012). Trustworthiness testing of phishing websites : A behavior model-based approach. Future Generation Computer Systems, 28(8), 1258-1271. https ://doi.org/10.1016/j.future.2011.02.001

[Shekokar et al., 2015] Shekokar, N. M., Shah, C., Mahajan, M., Rachh, S. (2015). An ideal approach for detection and prevention of phishing attacks. Procedia Computer Science, 49(1), 82-91. https ://doi.org/10.1016/j.procs.2015.04.230

[Sheng et al., 2010] Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L., Downs, J. (2010). Who Falls for Phish ? A Demographic Analysis of Phishing Susceptibility and Effectiveness of Interventions.

[Sujithra et al., 2020] Sujithra, T., Dwivedi, N., Utakarsha, A. (2020). Detection of phishing websites using deep learning and machine learning. Journal of Critical Reviews, 7(8), 1027-1032. https ://doi.org/10.31838/jcr.07.08.215

[Tai et al., 2016] Tai, L., Liu, M. (2016). Deep-learning in Mobile Robotics - from Perception to Control Systems : A Survey on Why and Why not. December.

[Tato et al., 2018] Tato, A., Nkambou, R. (2018). Improving adam optimizer.

[Vaitkevicius et Marcinkevicius, 2020] Vaitkevicius, P., Marcinkevicius, V. (2020). Comparison of Classification Algorithms for Detection of Phishing Websites. 31(1), 143-160.

[Vorontsov et Sci, 2017] Vorontsov, E., Sci, B. E. (2017). Deep Learning : A Primer for radiologists. Radiographics, 37(7), 2113-2131.

[Vrban et Podgorelec 2020] Vrban, G Podgorelec, V. (2020). Datasets for phishing websites detection i c. Data in Brief, 33, 106438. https ://doi.org/10.1016/j.dib.2020.106438

[Walsh , 2007] Walsh, E. F. (2007). Application of the Flask Architecture to the X Window System Server.

[Wei et al., 2020] Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., Woźniak, M. (2020). Accurate and fast URL phishing detector : A convolutional neural network approach. Computer Networks, 178(April). https ://doi.org/10.1016/j.comnet.2020.107275

[Wu et al., 2015] Wu, L., Du, X., Wu, J. (2015). Effective Defense Schemes for Phishing Attacks on Mobile Computing Platforms. January. https ://doi.org/10.1109/TVT.2015.2472993

[Xiang et al., 2011] Xiang, G., Hong, J., Rose, C. P., Cranor, L. (2011). Cantina+. ACM Transactions on Information and System Security, 14(2), 1-28. https ://doi.org/10.1145/2019599.2019606

[Yi et al., 2018] Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W., Zhu, T. (2018). Web phishing detection using a deep learning framework. Wireless Communications and Mobile Computing, 2018. https ://doi.org/10.1155/2018/4678746

[zhu et al., 2020] Zhu, R., Liao, W., Wang, Y. (2020). ScienceDirect Short-term prediction for wind power based on temporal convolutional network. Energy Reports, 6, 424-429. https ://doi.org/10.1016/j.egyr.2020.11.219

[Zhou et al., 2019] Zhou, Y., Yang, J., Zhang, H., Liang, Y., Tarokh, V. (2019). Sgd converges to global minimum in deep learning via star-convex path. arXiv preprint arXiv :1901.00451.

[Phone et al., 2020] Phone, S., Chen, L., Approaches, L., Madhavan, M. V., Pande, S., Sinaga, A. S., Haris, M., Sitio, A. S. (2020). Performance Comparison of Anti-Spam Technology Using Confusion Matrix Classification Performance Comparison of Anti-Spam Technology Using Confusion Matrix Classification. https ://doi.org/10.1088/1757-899X/879/1/012076

[Berry, 2020] Berry, M. W. (2020). Supervised and Unsupervised Learning for Data Science. January. https ://doi.org/10.1007/978-3-030-22475-2

[Charton, 2013] Charton, É. (2013). Hacker's guide. Pearson Education France.