République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira -Bejaia-
Faculté Des Sciences Exactes
Département D'Informatique

Mémoire de Fin de Cycle

En vue de l'obtention du diplôme de Master Recherche en Informatique

Option : Réseaux et Sécurité

Réalisé par :

Djafour Lyna
Ziani Thanina
Encadré par : Mme Sabri Salima

Thème

Détection de botnets basée sur une classification Random Forest dans IoT

Soutenue publiquement, le **30/06/2025**, devant le jury composé de :

Présidente :	Mme Bouadem Nassima	Université de bejaia
Examinatrice :	Mme Bouallouche Louiza	Université de bejaia
Examinatrice :	Mme Ouyahia Samira	Université de bejaia
Examinateur :	M. Sadi Mustapha	Université de bejaia

Dédicace

Avant toute chose, je remercie Dieu de m'avoir donné la force, la patience et la persévérance nécessaires à la réalisation de ce travail.

À ma précieuse et bien-aimée mère, *Ton amour est mon refuge, ta voix est mon réconfort. Par tes conseils, ta tendresse et ta sagesse, tu as toujours su apaiser mes doutes et me guider, telle une véritable psychologue de l'ombre. Merci pour ta présence constante, tes prières silencieuses et ton cœur immense.*

À mon cher et courageux père, *Modèle de force, de persévérance et de générosité. Tu as travaillé sans relâche pour nous offrir un avenir meilleur, toujours discret, mais toujours présent. Ton courage face aux difficultés et ta sérénité face aux épreuves m'inspireront toute ma vie.*

Je dédie ce mémoire à mes parents, Papa et Mama, avec un amour infini. Merci pour votre soutien inconditionnel, vos sacrifices quotidiens, et votre confiance en moi. Vous êtes ma plus grande force, mes repères, mes piliers.

À mes sœurs et à mon frère, Yassmine,Houda,Abderrahmane et Imane, *merci pour votre amour, vos sourires et votre présence constante, qui m'ont soutenue à chaque étape.*

Yassmine, *ma grande sœur et mon modèle, ta réussite et ta bienveillance m'inspirent.*

Houda, *posée et lucide, ta logique m'a souvent guidée.*

Abderrahmane, *mon petit frère au regard sage, tu es une source de tendresse et de réconfort.*

Imane, *mon rayon de lumière, ton amour sincère m'a portée plus que tu ne l'imagines.*

À toute ma famille, *Merci pour votre bienveillance et vos encouragements, toujours réconfortants, même à distance.*

À mes amis fidèles, *Pour vos mots d'encouragement, votre écoute, et vos présences sincères, je vous suis profondément reconnaissante.*

Une mention toute spéciale à Thanina, *Ma binôme, mais avant tout mon amie. Merci pour ta patience, ta loyauté, ton humour et ton esprit d'équipe. Ce parcours, je suis fière de l'avoir partagé avec toi.*

*Enfin**, à vous, chers lecteurs,*** *Merci de prendre le temps de découvrir ce travail. J'espère qu'il éveillera votre curiosité, nourrira votre réflexion, et peut-être, vous inspirera.*

Lyna

Je dédie ce travail à toutes les personnes qui, de près ou de loin, m'ont soutenu et encouragé tout au long de mon parcours.

Tout d'abord à Dieu, qui m'a donné le pouvoir, la chance de lire et d'écrire, la force et le courage de franchir chaque étape, et qui a toujours écouté chacune de mes prières.

À mes parents bien-aimés :

À mon cher père, *qui a toujours veillé à ce que je ne manque de rien, présent dans les moments les plus importants pour me soutenir dans toutes mes démarches. Tes sacrifices, ta présence constante et ton soutien sont pour moi une véritable source de motivation.*

À ma chère mère, *si précieuse et si tendre, qui me place toujours au premier plan, s'inquiète pour la moindre chose qui pourrait m'affecter, et qui, portée par son amour inconditionnel, ne laisse jamais passer un jour sans prier pour moi.*

À mon grand frère Massi, *pour sa bienveillance, son soutien moral et sa présence rassurante. Toujours présent dans les moments importants, il sait me réconforter et m'encourager lorsque j'en ai besoin.*

À mes grandes sœurs Nacera et Saliha, *qui sont comme des secondes mères pour moi, toujours présentes pour veiller à ce que je ne manque de rien, qui savent trouver des solutions et sont toujours à l'écoute, contribuant chaque jour à faire de moi la personne que je suis aujourd'hui.*

À mon petit frère Walid, *que je porte toujours dans mon cœur.*

À mes amis, *avec qui j'ai partagé autant de moments de joie que d'épreuves, qui ont été présents durant toutes ces années, toujours là quand j'en avais besoin. Merci pour votre amitié sincère, votre soutien et vos encouragements. Tous ces bons moments restent des souvenirs gravés que je garderai précieusement.*

Et enfin, ***à ma chère binôme,*** *qui a su faire ressortir le meilleur en moi. Plus qu'une simple binôme, tu as été comme une sœur : travailler avec toi a été un véritable plaisir et une belle expérience. Si c'était à refaire, je te choisirais sans hésiter, à chaque fois.*

Merci pour tout.

Thanina

Remerciement

Nous tenons tout d'abord à remercier Dieu de nous avoir donné la force et le courage nécessaires pour réaliser ce modeste travail.

Nos remerciements les plus sincères s'adressent à toutes les personnes qui ont, de près ou de loin, contribué à la réalisation de ce mémoire.

Nous exprimons notre profonde reconnaissance à ***Mme Salima Sabri****, notre encadrante, pour son accompagnement attentif, ses conseils avisés et sa disponibilité constante. Sa méthode de travail méthodique et rigoureuse et son soutien bienveillant ont été déterminants dans l'orientation et la réussite de ce projet.*

Nous remercions chaleureusement nos parents pour leur amour inconditionnel, leur confiance, et leur soutien indéfectible tout au long de notre parcours. Leur présence à nos côtés a été une source d'inspiration et de force.

À nos frères et sœurs, nous adressons toute notre affection et notre reconnaissance pour leur soutien moral et leur encouragement au quotidien. Merci également à toute notre famille pour leur présence et leurs encouragements constants.

Nous souhaitons également remercier nos amis, pour leurs échanges, leur aide et leurs encouragements tout au long de cette aventure.

Un remerciement particulier à nos camarades de la spécialité Réseau et Sécurité, pour leur solidarité, leur esprit d'équipe et les nombreux moments de partage qui ont marqué cette formation.

Nous remercions enfin les membres du jury pour le temps qu'ils consacreront à l'évaluation de notre travail, ainsi que l'ensemble des enseignants et du personnel administratif de l'université pour leur accompagnement et pour la qualité de l'enseignement reçu.

Merci à tous.

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

IoT	Internet des Objets
IIoT	Internet Industriel des Objets
DoS	Déni de service
DDoS	Déni de service distribué
C&C	Commande et Contrôle
P2P	Pair-à-pair
SDN	Réseau défini par logiciel
DNS	Système de noms de domaine
IA	Intelligence Artificielle
ML	Apprentissage automatique
DL	Apprentissage profond
DT	Arbre de décision
RF	Forêt aléatoire
XGB	XGBoost
KNN	K plus proches voisins
LR	Régression logistique
K-Means	Regroupement K-Moyennes
SVM	Machine à vecteurs de support
PCA	Analyse en composantes principales
MLP	Perceptron multicouche
AE	Autoencodeur

RNN	Réseau de neurones récurrent
GRU	Unité récurrente avec portes
EO	Optimiseur d'Équilibre
AEO	Optimiseur d'Équilibre Adaptatif
BRO	Battle Royale Optimizer
TP	Vrai positif
FP	Faux positif
FN	Faux négatif
TN	Vrai négatif
GNB	Classifieur Bayésien naïf gaussien

Introduction Générale

À l'ère de l'Internet des Objets (IoT), les réseaux informatiques sont désormais composés de milliards de dispositifs interconnectés, capables de produire, transmettre et recevoir d'importants volumes de données. Si cette révolution technologique ouvre la voie à des applications variées dans des domaines tels que la santé, l'industrie, la domotique ou encore les transports, elle expose également les infrastructures à de nouvelles formes de menaces.

Parmi celles-ci figure l'exploitation d'objets connectés par des botnets, ces réseaux d'appareils compromis utilisés pour mener des cyberattaques à grande échelle (attaques DDoS, vol de données, espionnage, etc.).

Ce mémoire s'inscrit dans le domaine de la cybersécurité, et plus particulièrement dans la détection des botnets. Dans ce contexte, la détection précoce des botnets devient un enjeu majeur pour garantir la sécurité des réseaux modernes. Toutefois, cette tâche s'avère particulièrement complexe : les botnets évoluent rapidement, adoptent des techniques de dissimulation sophistiquées et exploitent l'hétérogénéité du trafic réseau, rendant leur détection difficile, surtout dans les environnements non étiquetés ou bruités.

Motivation

L'analyse de la littérature scientifique montre que les approches supervisées classiques, telles que **Random Forest** ou **XGBoost**, obtiennent de bons résultats sur des jeux de données bien étiquetés. Cependant, elles montrent rapidement leurs limites lorsqu'elles sont confrontées à des environnements plus réalistes, comme ceux des réseaux IoT, où les données sont souvent partielles, déséquilibrées et bruitées.

À l'inverse, les approches non supervisées, basées sur le clustering ou l'analyse de reconstruction par autoencodeur, sont capables de détecter des comportements anormaux sans connaissance préalable. Toutefois, leur précision reste limitée.

Ces constats ont motivé le développement d'une approche hybride, tirant parti à la fois de la capacité d'adaptation des méthodes non supervisées et de la puissance des modèles supervisés, dans une architecture intégrée, robuste et explicable.

Objectifs

L'objectif principal de ce travail est de concevoir une méthode efficace pour la détection des botnets dans des réseaux IoT complexes, dynamiques et faiblement étiquetés.

Plus précisément, cette approche vise à :

- Détecter efficacement les botnets, qu'ils soient connus ou nouveaux, à partir de flux réseau réels, bruités et volumineux, sans dépendre uniquement des données étiquetées ;
- Adapter une technique de détection de comportements malveillants tout en maintenant une grande précision ;
- Réduire les faux positifs en combinant des techniques non supervisées et supervisées pour mieux différencier le trafic normal, de fond et malveillant ;
- Assurer une détection rapide et en quasi temps réel, adaptée aux contraintes techniques des réseaux IoT ;
- Garantir une classification fiable du trafic en distinguant efficacement les comportements normaux et malveillants, même dans un environnement bruité et déséquilibré.

Ce mémoire est structuré en trois chapitres complémentaires :

- **Chapitre 1 :**Présente les généralités nécessaires à la compréhension de la problématique. Il introduit les concepts fondamentaux liés à l'Internet des Objets (IoT), au trafic réseau, aux botnets, ainsi qu'aux techniques d'intelligence artificielle utilisées pour la détection des botnets, notamment le machine learning (ML) et le deep learning (DL).
- **Chapitre 2 :**Propose une revue de littérature centrée sur les méthodes de détection des botnets basées sur l'intelligence artificielle. Il présente les principaux algorithmes de machine learning, de deep learning ainsi que les approches hybrides appliquées à cette problématique. Les métriques utilisées pour évaluer les performances des modèles sont également décrites. Une synthèse critique met en évidence les limites des approches actuelles et les pistes d'amélioration.
- **Chapitre 3 :**Détaille notre contribution à la détection des botnets dans le trafic réseau IoT. L'architecture proposée, nommée **KRAST**, combine des techniques supervisées et non supervisées dans une approche hybride. L'implémentation optimisée sur TPU est décrite, suivie de l'évaluation expérimentale sur le dataset CTU-13. Les résultats obtenus sont comparés à ceux des approches existantes, et des pistes d'amélioration sont proposées pour renforcer l'efficacité et l'adaptabilité de la solution.

Chapitre 1

Généralités et concepts de base

Introduction

Dans un contexte où les technologies numériques ne cessent de se développer, l'Internet des Objets (IoT) joue un rôle de plus en plus central dans les systèmes connectés, industriels comme personnels. Si ces objets facilitent l'automatisation, l'accessibilité aux données et l'efficacité opérationnelle, ils ouvrent également la porte à de nouvelles menaces, parmi lesquelles les botnets constituent une forme particulièrement redoutable. Ce chapitre vise à poser les bases nécessaires à la compréhension de cette problématique en présentant les notions fondamentales de l'IoT, du trafic réseau et de la cybersécurité associée. Nous aborderons également les caractéristiques et le fonctionnement des botnets, leurs types, ainsi que les techniques de détection et de prévention actuelles. Enfin, une section sera dédiée à l'intelligence artificielle et à l'apprentissage automatique, dont l'usage s'avère prometteur pour faire face à ces menaces émergentes.

1.1 Internet des objets (IoT)

Cette section est consacrée à l'exploration de l'IoT : ses composants, son architecture, ses techniques de communication, ainsi que ses avantages et ses défis.

1.1.1 Définition de l'IoT

L'Internet des Objets (IoT) est un réseau d'objets physiques et virtuels connectés à Internet, capables de communiquer et d'échanger des données en temps réel [1]. Ces objets,

équipés de systèmes d'identification électronique et de technologies, majoritairement sans fil, peuvent être identifiés de manière unique et interagir de façon autonome [2].

L'IoT permet l'automatisation et l'optimisation de nombreux secteurs, notamment la santé, l'industrie, la logistique, la sécurité et les transports intelligents [2]. D'après l'Union Internationale des Télécommunications (UIT) [3], l'IoT représente une infrastructure mondiale conçue pour utiliser la collecte, le traitement et la transmission des données produites par les objets connectés, dans l'objectif d'améliorer les services et d'accroître l'efficience des systèmes [3].

1.1.2 Composants de l'IoT

Un système IoT repose sur plusieurs composants essentiels qui assurent son bon fonctionnement. Parmi ces composants [4],[5],[6] on trouve :

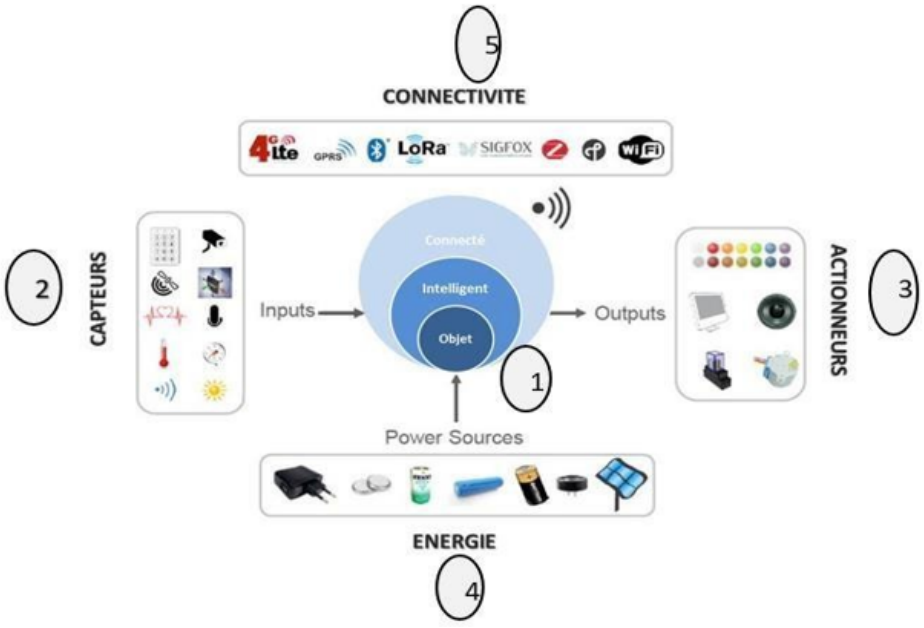

Figure 1.1 – Composants de l'IoT [4].

1. **L'objet connecté (OC) :**Appareil mécanique ou électrique capable d'échanger des informations avec d'autres dispositifs ou internet.

Figure 1.2 – Les objets connectés [5]

2. **Capteurs :**Dispositifs convertissent des mesures physiques (température, mouvement, etc.) en données numériques. Exemples : Capteurs de température, proximité, pression, humidité, fréquence, cardiaque (ECG), lumière ...

Figure 1.3 – Capteurs.

3. **Actionneurs :**Contrairement aux capteurs, les actionneurs convertissent une donnée numérique en phénomène physique en réalisant des actions spécifiques. Exemples : Écran, Haut-parleur, Pompe bleue, vannes ...

Figure 1.4 – Actionneurs.

4. **Source d'énergie :** Il existe quatre types de sources d'énergie que l'objet connecté nécessite (alimentation électrique, batteries ou piles, capteurs d'énergie, ondes électromagnétiques des lecteurs RFID (Radio Frequency Identification) et NFC (Near Field Communication).

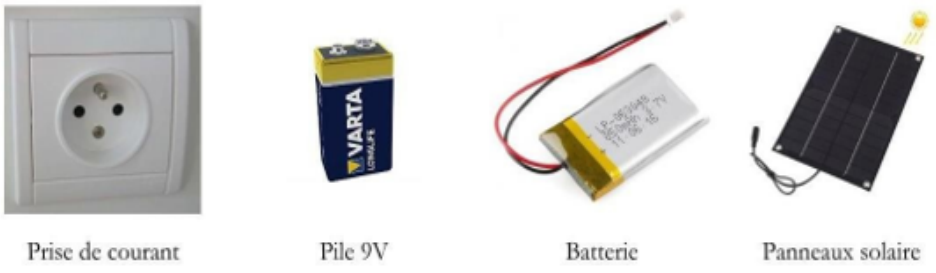

Figure 1.5 – Sources d'énergie.

5. **Connectivité et traitement des données :** La connectivité de l'objet est assurée par une antenne radio fréquence, permettant la communication avec un ou plusieurs réseaux. Les données collectées par les capteurs sont ensuite transmises à un processeur intégré puis stockées dans un serveur, pour être analysées à distance par des algorithmes.

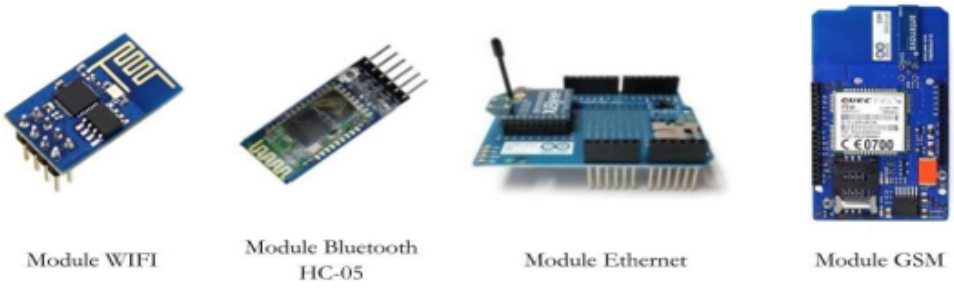

Figure 1.6 – Modules de Connectivité.

1.1.3 Architecture de l'IoT

L'architecture de l'Internet des Objets (IoT) est généralement composée de plusieurs couches. Bien que leur nombre puisse varier selon les modèles, les plus courants en comportent cinq. Voici les principales couches [7],[8],[9] :

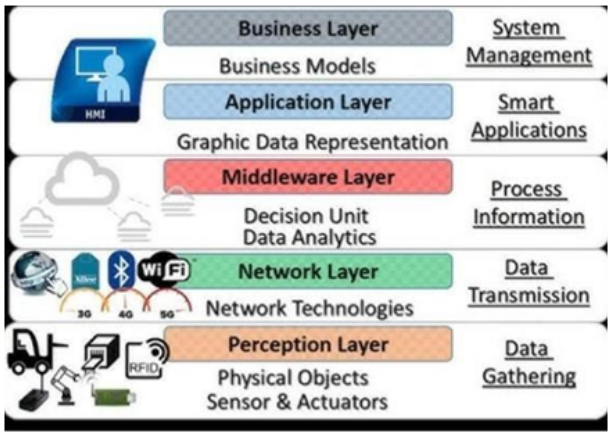

Figure 1.7 – Architecture de l'IoT [10].

1. **Couche de perception :** Également nommée Couche physique, elle englobe tous les objets et appareils qui entrent en interaction directe avec le monde réel. Elle renferme des capteurs et des actionneurs qui récupèrent et transforment les signaux analogiques en informations numériques exploitables.
2. **Couche réseau /Transport de données :** Cette couche est responsable de la transmission des données entre les dispositifs et les autres niveaux du système. Elle relie les capteurs, les actionneurs et autres équipements au cloud ou à d'autres réseaux grâce à des technologies de communication comme le Wi-Fi, le Bluetooth, la 4G, la 5G, le WAN,

entre autres. Les systèmes d'acquisition de données (DAS) et les passerelles Internet sont essentiels pour assurer une liaison stable.

3. **La couche de traitement :** Elle est responsable de la sauvegarde, de l'analyse et du traitement des données qu'elle reçoit. Elle exploite des technologies comme l'intelligence artificielle (IA), l'apprentissage automatique (ML) et différents algorithmes afin d'extraire des informations significatives et de prendre des décisions. Il est possible de traiter les données sur place ou de les transférer vers un centre de données pour une analyse plus détaillée.
4. **Couche d'application :** Cette couche comprend l'ensemble des logiciels et interfaces qui permettent aux utilisateurs d'interagir avec le système IoT. Qu'il s'agisse d'applications mobiles, de tableaux de bord web ou d'interfaces de contrôle, cette couche permet de configurer les dispositifs, de visualiser les données en temps réel et de gérer les flux de travail IoT. Elle permet aussi d'ajuster les paramètres du système pour répondre aux besoins spécifiques des utilisateurs.
5. **Couche métier :** La dernière couche est orientée vers les processus d'affaires. Elle convertit les données traitées en informations exploitables pour la prise de décision stratégique, grâce à l'intégration avec des systèmes d'information, les outils de Business Intelligence ou de visualisation de données, cette couche aide les décideurs à améliorer la productivité, à optimiser les ressources et à orienter les stratégies commerciales.

1.1.4 Technologies de communication de l'IoT

On utilise diverses technologies dans l'exploitation de l'IoT, dont on peut citer :

- **Le wifi :**La technologie Wi-Fi (IEEE 802.11) est essentielle pour l'Internet des Objets (IoT), surtout pour les appareils qui requièrent une large bande passante et une connexion stable, tels que les caméras de surveillance, les assistants vocaux et les dispositifs médicaux connectés. Il propose une portée impressionnante pouvant aller jusqu'à 100 mètres en milieu dégagé (en extérieur sans obstructions), même si cette distance est généralement limitée à 20-30 mètres à l'intérieur en raison des murs et des interférences. Le Wi-Fi facilite également la connexion avec les services cloud. Toutefois, sa grande consommation énergétique le rend inadapté aux capteurs autonomes[11].
- **Le Bluetooth :** basé sur la norme IEEE 802.15.1 c'est une technologie sans fil à courte portée souvent utilisée dans les dispositifs IoT destinés au grand public. La distance

de transmission dépend de la classe du dispositif : environ 1 mètre pour la Classe 3, 10 mètres pour la Classe 2 (la plus courante) et jusqu'à 100 mètres pour la Classe 1. Le Bluetooth Low Energy (BLE), une version optimisée pour réduire la consommation d'énergie, a généralement une portée de 10 à 30 mètres. Le BLE, avec sa consommation réduite, est de plus en plus intégré dans les environnements industriels intelligents. Il facilite la communication directe entre plusieurs dispositifs tout en préservant l'énergie [12].

- **Zigbee** il s'agit d'un protocole de réseau local basé sur la norme IEEE 802.15.4, conçu pour des transmissions à faible débit sur une distance restreinte (qui peut aller jusqu'à 100 m). Cet appareil, qui fonctionne en 2,4 GHz et utilise une topologie maillée, a la capacité de connecter des milliers de nœuds afin d'assurer un transfert efficace des informations entre les capteurs et les équipements [12].
- **Near-Field Communication (NFC) :** Le NFC est une technologie de communication sans fil à très courte portée (généralement entre 1 et 4 cm), simple d'utilisation, qui permet l'échange sécurisé de données entre appareils électroniques en proximité immédiate. Elle offre des taux de transmission variant de 106 à 424 kbit/s grâce à sa fréquence de 13,56 MHz, assurant une liaison ultra-rapide (moins d'une seconde). On utilise souvent cette technologie pour les paiements sans contact (Apple Pay, Google Pay), l'accès sécurisé , le partage immédiat d'informations (cartes de visite numériques) et la connexion à des dispositifs (haut-parleurs Bluetooth, écouteurs). Contrairement au RFID standard, le NFC permet une communication dans les deux sens et adopte des protocoles de sécurité sophistiqués en accord avec les normes ISO/IEC 14443 et 18092, tout en consommant peu d'énergie grâce à ses modes passif et actif [13].
- **Identification par radiofréquence(RFID) :**Cette technologie englobe tous les systèmes qui font appel à des ondes radio pour stocker et consulter des informations à distance, avec une portée pouvant varier de quelques centimètres à 100 mètres selon le type d'étiquette (passif, actif ou semi-passif) et la fréquence utilisée (LF, HF, UHF). Elle utilise des étiquettes RFID, composées d'une antenne et d'une puce électronique, qui interagissent avec les ondes radio produites par un appareil de lecture pour communiquer les informations [14].
- **Technologie cellulaire :**Les technologies de communication mobile, telles que le GSM (2G), la 3G, la 4G et le NB-IoT, permettent la transmission des données des appareils IIoT sur de longues distances, allant de quelques kilomètres à plusieurs dizaines de

kilomètres selon la couverture réseau. Ces technologies sont adaptées aussi bien aux transmissions à fort volume de données qu'aux communications à faible débit et faible consommation. La 5G vient renforcer ces capacités avec des débits beaucoup plus élevés et une latence très faible, favorisant ainsi l'automatisation industrielle, la robotique de précision et l'Internet des objets massif (mIoT). Elle permet de connecter des milliards de dispositifs, même dans des zones éloignées, grâce à des infrastructures réseau étendues et fiables [15].

1.1.5 Avantages de l'IoT

L'IoT offre de nombreux avantages, comme l'illustrent les points suivants [16],[17] :

- L'IoT permet une automatisation intelligente et une optimisation des ressources grâce à l'intégration de capteurs surveillant en continu les équipements industriels, agricoles et urbains, réduisant ainsi les interventions humaines et les gaspillages énergétiques.
- L'union de l'intelligence artificielle et de l'IoT renforce la prévention des pannes et la sûreté, en particulier grâce à des modèles d'analyse prédictive utilisés dans la maintenance industrielle et les appareils médicaux connectés.
- L'IoT réduit la charge de travail humaine et procure des économies de temps en automatisant les tâches répétitives comme la gestion de l'éclairage et le contrôle thermique dans les maisons intelligents.
- Les appareils connectés améliorent la sécurité grâce à des systèmes intelligents de vidéo surveillance, des détecteurs de mouvements et des notifications instantanées pour repérer les incendies ou les intrusions.
- L'IoT favorise la mobilité et l'accès instantané, rendant possible le contrôle à distance des appareils par le biais de smartphones, tablettes ou interfaces cloud .
- Cette technologie facilite une communication fluide entre dispositifs à travers des protocoles standardisés et une structure décentralisée, simplifiant la gestion de plusieurs capteurs.
- Elle participe à l'amélioration de la productivité et à la réduction des coûts, la maintenance prédictive pouvant engendrer jusqu'à 25 % d'économies sur les interventions industrielles.
- Les données collectées optimisent les décisions stratégiques et les processus opérationnels en fournissant des métriques précises sur performances et comportements.

- L'IoT facilite le contrôle et l'administration des infrastructures essentielles, comme les réseaux de transport, les systèmes d'électricité ou les équipements d'irrigation, en optimisant la planification et diminuant les interruptions.

1.1.6 Les Défis de l'IoT

Les principaux défis de l'Internet des Objets (IoT) [18],[19],[20] :

- La cybersécurité est un grand défi car beaucoup d'objets connectés utilisent des mots de passe par défaut, des protocoles non sécurisés ou des logiciels obsolètes, ce qui facilite les attaques comme les botnets ou les DDoS .
- Protéger les données personnelles est difficile, surtout pour des informations sensibles comme la santé ou la localisation, et il faut respecter des lois strictes comme le règlement général sur la protection des données (RGPD) .
- Il n'y a pas de normes communes entre fabricants, ce qui complique la connexion des objets entre eux à cause de protocoles différents et de programmation applicative (API) propriétaires.
- Avec l'augmentation du nombre d'objets, il faut des infrastructures capables de gérer beaucoup de données en temps réel .
- Il est nécessaire d'avoir des solutions efficaces pour stocker, analyser et sécuriser ces données, souvent basées sur le cloud ou l'informatique en périphérie (edge computing) .
- Connecter les objets IoT aux systèmes industriels déjà en place pose des problèmes techniques, notamment dans l'industrie et l'énergie .
- Installer et entretenir les infrastructures IoT coûte cher, ce qui freine les petites entreprises à les adopter .
- L'autonomie des objets fonctionnant sur batterie, surtout dans des endroits isolés comme en agriculture, est un défi pour assurer leur fonctionnement durable .
- L'augmentation du nombre d'objets connectés crée plus de déchets électroniques, ce qui pose un problème pour l'environnement .
- Certains objets collectent des données sans que les utilisateurs le sachent, ce qui peut mener à une surveillance excessive (ex : assistants vocaux, caméras intelligentes) .
- L'usage commercial ou abusif des données collectées par des objets comme les montres connectées ou les voitures pose des questions éthiques et juridiques .

- Beaucoup de fabricants ne mettent pas à jour leurs appareils, laissant des failles de sécurité longtemps après leur vente .
- L'absence de systèmes automatisés pour gérer à distance les failles de sécurité augmente les risques, surtout pour les objets déployés en grand nombre .

1.2 Trafic réseau

Cette section approfondit l'étude du trafic réseau en abordant ses sources et destinations, ses types, les protocoles de communication impliqués, les unités de mesure utilisées, les méthodes d'analyse, les enjeux de sécurité et de surveillance, ainsi que ses applications dans divers domaines.

1.2.1 Réseau informatique

Un réseau informatique est un système qui permet à plusieurs appareils—rdinateurs, téléphones, tablettes, serveurs ou objets connectés — de communiquer et d'échanger des données. Ces connexions peuvent être filaires (via un câble Ethernet) ou sans fil (Wi-Fi, Bluetooth). Le but est de partager des informations ou d'accéder à des services comme naviguer sur Internet, imprimer un document, consulter une base de données ou échanger des messages. Les réseaux peuvent être locaux (LAN), limités à une zone géographique précise, ou étendus (WAN), comme Internet, qui connecte des milliards d'appareils dans le monde [21].

1.2.2 Le trafic réseau

Le trafic réseau désigne l'ensemble des données numériques échangées entre les appareils d'un réseau informatique. Ces échanges se font sous forme de paquets et proviennent de nombreuses activités : navigation web, envoi de messages, téléchargement de fichiers ou utilisation de services en ligne. Le trafic peut être entrant, sortant ou local, et son volume varie selon le type de réseau (domestique, professionnel, industriel). Une bonne gestion et une analyse régulière de ce trafic sont essentielles pour assurer les performances du réseau, garantir la qualité de service et renforcer la cybersécurité [22].

1.2.3 Origine et destination

Chaque communication sur un réseau implique un émetteur (source) et un récepteur (destination). Par exemple, quand un utilisateur consulte une page web, son appareil envoie une requête vers un serveur distant, qui renvoie ensuite les données demandées. Ce modèle d'échange existe aussi bien pour les communications via Internet que pour les échanges internes à un réseau local. Comprendre les flux entre sources et destinations permet de mieux suivre l'activité du réseau, de localiser les erreurs et de repérer des comportements suspects [23].

1.2.4 Types de trafic réseau

Le trafic réseau peut être classé en différentes catégories, selon le sens des échanges [24],[25] :

- **Trafic entrant**

  Le trafic entrant correspond aux données qui arrivent sur un appareil depuis l'extérieur du réseau local, généralement via Internet. Cela inclut la réception d'emails, le streaming de vidéos ou le téléchargement de fichiers. Ce type de trafic est étroitement surveillé car il peut transporter des menaces comme des virus, des logiciels espions ou des tentatives d'intrusion. Une gestion efficace du trafic entrant contribue à la protection du système.

- **Trafic sortant**

  Le trafic sortant désigne les données générées par un appareil local et envoyées vers l'extérieur du réseau. Cela comprend l'envoi d'emails, le partage de fichiers sur le cloud ou la soumission de formulaires en ligne. Ce trafic doit être surveillé pour limiter les risques de fuite de données sensibles ou de comportements suspects liés à un logiciel malveillant. En entreprise, des règles de filtrage spécifiques sont souvent mises en place pour contrôler le trafic sortant.

- **Trafic local**

  Le trafic local regroupe les échanges entre les appareils d'un même réseau local, sans transiter par Internet. Par exemple, un ordinateur qui envoie un document à une imprimante en Wi-Fi. Ces échanges sont plus rapides et sécurisés, car ils restent confinés dans l'environnement local. Une bonne gestion du trafic local permet d'optimiser les performances réseau et d'éviter une surcharge inutile de la connexion Internet.

1.2.5 Les protocoles organisent les échanges de données

Dans un réseau, les échanges de données obéissent à des règles strictes appelées protocoles de communication. Ils déterminent la manière dont les informations sont organisées, transmises, reçues et validées entre les appareils. Sans ces mécanismes, les ordinateurs, smartphones et objets connectés ne pourraient interagir efficacement. Chaque protocole intervient à un niveau spécifique du modèle en couches OSI, bien que, dans la pratique, la suite TCP/IP soit la plus largement utilisée sur Internet. Voici les principaux protocoles [26],[27] :

- **TCP/IP (Transmission Control Protocol / Internet Protocol)**

  Ce protocole est la base d'Internet. IP s'occupe de l'adressage et du routage des paquets, tandis que TCP garantit la fiabilité des échanges, en s'assurant que les paquets arrivent complets et dans le bon ordre.

- **UDP (User Datagram Protocol)**

  Ce protocole est plus rapide que TCP, mais ne vérifie pas la réception des paquets. Il est utilisé pour les flux en temps réel comme les vidéos, les appels audio ou les jeux en ligne, où la vitesse est plus importante que la fiabilité absolue.

- **HTTP / HTTPS (HyperText Transfer Protocol)**

  C'est le protocole utilisé pour accéder aux pages web. HTTPS est la version sécurisée, qui chiffre les échanges pour garantir la confidentialité et l'intégrité des données.

- **DNS (Domain Name System)**

  Il permet de traduire les noms de domaine en adresses IP, pour que les appareils puissent localiser les serveurs correspondants.

- **DHCP (Dynamic Host Configuration Protocol)**

  Ce protocole attribue automatiquement une adresse IP aux appareils qui se connectent au réseau, simplifiant leur configuration.

- **FTP (File Transfer Protocol)**

  FTP permet de transférer des fichiers entre un client et un serveur, notamment dans les environnements professionnels.

1.2.6 Unités de mesure

Pour surveiller un réseau, plusieurs indicateurs sont utilisés [28] :

- **La bande passante**

  Elle représente la capacité maximale du réseau à transporter des données sur une période donnée, exprimée en Mbps ou Gbps. On peut la comparer à la largeur d'une autoroute : plus elle est large, plus de paquets de données peuvent y circuler simultanément.

- **Le volume de données**

  C'est la quantité totale de données échangées sur le réseau, mesurée en Mo, Go ou To. Une augmentation soudaine du volume peut signaler une activité anormale, comme un téléchargement massif ou une attaque.

- **La latence**

  Elle mesure le temps nécessaire pour qu'un paquet de données effectue un aller-retour entre la source et la destination, en millisecondes. Une faible latence est essentielle pour les applications en temps réel, comme les visioconférences ou les jeux en ligne.

Ces indicateurs permettent d'évaluer les performances du réseau et de repérer d'éventuels problèmes.

1.2.7 Analyse du trafic

L'analyse du trafic réseau s'appuie sur des outils spécialisés tels que Wireshark, tcpdump ou NetFlow, qui offrent une visualisation détaillée des données circulant sur un réseau. Grâce à ces solutions, il est possible d'observer [29],[30] :

- **Les adresses IP source et destination :** elles révèlent quels appareils communiquent entre eux.
- **Les ports utilisés :** ils permettent d'identifier quels services ou applications sont actifs (HTTP, HTTPS, SMTP...).

- **Le volume de données échangé :** cela aide à détecter les pics d'activité ou les transferts inhabituels.
- **Les protocoles utilisés :** ils permettent de catégoriser le trafic (web, mail, transfert de fichiers, etc.).
- **Les anomalies éventuelles :** en comparant le trafic observé à un comportement réseau normal, il est possible de détecter des attaques, des intrusions ou des malwares.

1.2.8 Sécurité et surveillance

La sécurité des réseaux repose en grande partie sur l'analyse du trafic, car chaque échange de données peut révéler des comportements anormaux ou des signes d'intrusion. Cette surveillance permet non seulement de détecter les attaques en cours, mais aussi d'anticiper les menaces. Voici les principaux aspects à considérer :

- **Trafic comme indicateur de compromission**

  Le trafic réseau peut refléter aussi bien un fonctionnement normal qu'une activité malveillante. Lorsqu'un appareil est compromis, ses échanges changent : il peut envoyer des paquets vers des adresses inhabituelles, recevoir des commandes à distance ou participer à des attaques. Ces modifications sont souvent les premiers signes visibles d'un incident de sécurité [31].

- **Botnets**

  Une menace fréquente repose sur l'exploitation du réseau par des botnets : des réseaux d'équipements infectés, souvent des objets connectés mal protégés, contrôlés à distance par des cybercriminels. Ces appareils peuvent générer du trafic malveillant, être utilisés pour des attaques par déni de service distribué (DDoS) ou agir comme relais pour masquer l'origine de l'attaquant [32].

- **Attaques DDoS**

  Les attaques DDoS visent à submerger un serveur ou un service avec un volume massif de trafic, jusqu'à l'épuisement de ses ressources. L'analyse du trafic est alors essentielle pour identifier l'origine des requêtes, leur typologie et mettre en place des contre-mesures rapidement [32].

- **Malwares et communications réseau**

  De nombreux malwares utilisent le réseau pour se propager, exfiltrer des données ou recevoir des instructions. Par exemple, un ransomware peut envoyer des fichiers chiffrés

vers un serveur de commande ou attendre un signal de déclenchement. Bien que le trafic généré soit parfois chiffré ou dissimulé, il présente souvent des comportements inhabituels détectables [32].

- **Outils de sécurité réseau**

  Pour se protéger contre les menaces, les administrateurs réseau utilisent divers outils de sécurité. Les systèmes de détection d'intrusion (IDS) jouent un rôle clé en analysant le trafic en temps réel afin d'identifier des comportements suspects ou des signatures d'attaque connues. En parallèle, les pare-feux contrôlent les flux de données entrants et sortants en appliquant des règles strictes, bloquant ainsi les communications non autorisées ou potentiellement dangereuses. Ensemble, ces dispositifs permettent de renforcer la défense du réseau face aux attaques [31].

- **Analyse comportementale**

  Cette approche consiste à établir un modèle du comportement normal du réseau (types de connexions, horaires, volumes habituels...) et à détecter automatiquement tout écart significatif. Elle est particulièrement efficace dans les réseaux IoT, où les objets connectés ont des comportements réguliers [33].

1.2.9 Application dans différents contextes

Le trafic réseau varie selon les contextes d'utilisation, qu'il s'agisse de réseaux domestiques, professionnels, industriels ou encore de secteurs sensibles tels que la santé ou l'administration.

- **Réseaux domestiques**

  Ce type de trafic est lié à des usages personnels : navigation Internet, streaming, jeux en ligne, objets connectés. Bien que moins critique, il reste vulnérable, notamment à cause de la faible sécurité de certains appareils IoT [34].

- **Réseaux d'entreprise**

  Le trafic y est souvent structuré et critique : envoi d'emails, accès à des bases de données, visioconférences, cloud computing, etc. Il nécessite une haute disponibilité, une sécurité renforcée, et une surveillance constante [34].

- **Réseaux IoT**

  Les objets connectés génèrent un trafic fréquent et souvent en petites quantités. La latence doit être faible et la sécurité adaptée aux contraintes de ressources des appareils [35].

- **Réseaux industriels (IIoT)**

  Ils assurent le contrôle de machines automatisées. Le trafic doit être extrêmement fiable et sécurisé, car une défaillance peut entraîner des conséquences physiques ou financières importantes [35].

- **Réseaux dans les secteurs sensibles (santé, éducation, administration)**

  Le trafic y transporte des données personnelles ou confidentielles. Il est essentiel de garantir sa confidentialité, sa disponibilité et son intégrité [34].

1.3 Bontnet

Cette section traite les botnets : leur définition, leurs types, leurs composants, ainsi que les différents Modèles d'architecture. Nous aborderons également le fonctionnement et les types des attaques, les techniques de détection, quelques exemples connus, ainsi que les méthodes de prévention et de protection.

1.3.1 Définition d'un botnet :

Le terme « botnet » est une contraction des mots « robot » et « network » (réseau). Il désigne un ensemble d'ordinateurs ou de dispositifs contrôlés par un cybercriminel, utilisés pour mener des actions malveillantes contre une cible spécifique [36].

La création d'un botnet implique généralement l'infiltration progressive d'un système, permettant des attaques à grande échelle : vols de données, saturation de serveurs, propagation de logiciels malveillants ou attaques par déni de service distribué (DDoS) [37].

À l'origine, ces réseaux de bots (ou « zombies ») servaient à des fins légitimes, comme la modération automatisée des chats en ligne. Toutefois, leur capacité d'exécuter des ordres à distance a entraîné leur exploitation pour des actions illégales, y compris l'espionnage, le piratage de mots de passe ou les cyberattaques de grande envergure [38]. À l'heure actuelle, les botnets sont surtout liés aux attaques DDoS, durant lesquelles des milliers d'appareils compromis inondent un serveur de requêtes au point de le rendre indisponible [39]. Par exemple, ce genre d'attaque a provoqué certaines des plus importantes interruptions de l'internet [38].

1.3.2 Types d'un botnet :

Il est essentiel de connaître les les différents types de botnets, qui se distinguent par leurs caractéristiques et leurs modes de fonctionnement. Ils peuvent être classés selon plusieurs critères [39],[40] :

- **Classification par Fonctionnalité :** les botnets peuvent être servir à des fins variées. Certains sont conçus pour l'espionnage, notamment en collectant les frappes sur le clavier (keylogging) ou en réalisant des captures d'écran. D'autres sont utilisées pour propager massivement des courriers indésirables, dérober des informations sensibles ou déclencher des attaques de type déni de service distribué (DDoS). Ces botnets sont généralement contrôlés à partir d'un serveur de commande et de contrôle (C &C) centralisé, ce qui simplifie la gestion des actions malveillantes.
- **Classification par Architecture :** Les botnets peuvent adopter diverses structures : le modèle client-serveur, le plus répandu, repose sur un serveur central (CC), ce qui en facilite la détection grâce à son point de contrôle unique. Le modèle décentralisé P2P (Pair- à-Pair) permet à chaque nœud de fonctionner en tant que client et serveur, rendant ainsi plus complexe la tâche d'élimination du botnet. Enfin, le modèle hybride combine client- serveur et P2P, renforce la résilience tout en rendant la détection plus complexe.
- **Plates-formes ciblées :** Les botnets sont classés selon les systèmes d'exploitation qu'ils ciblent : ceux qui exploitent Windows tirent parti de sa prévalence et de ses vulnérabilités, tandis que ceux basés sur Linux ciblent principalement les serveurs et les dispositifs connectés. Avec la montée en puissance du mobile, les botnets Android les plus récents représentent une menace grandissante. En définitive, les botnets mobiles sur iOS et BlackBerry sont rares en raison des restrictions strictes imposées par ces systèmes.
- **Mode de contrôle :** Les botnets ont la capacité de fonctionner de façon indépendante, sans besoin d'interaction humaine, en exploitant les ressources des appareils compromis pour orchestrer des attaques subtiles et difficiles à détecter. Ils peuvent aussi être manuels, dans lesquels les pirates supervisent directement les dispositifs compromis, offrant une plus grande souplesse pour accomplir des actions particulières .
- **Origine du botnet :** Certains sont conçus spécifiquement par des groupes de cybercriminalité, généralement dans l'intention d'atteindre un objectif ciblé ou stratégique.

Sur le dark web, divers botnets sont proposés à la vente comme des services, accessibles pour des individus ayant des compétences techniques moins avancées. Cette accessibilité a favorisé l'essor des attaques via des botnets.

- **Type de malware utilisé :** Les botnets peuvent intégrer différents outils malveillants. Certains utilisent des malwares spécialisés dans les attaques DDoS, tandis que d'autres utilisent ceux dédiés à la détection automatique de vulnérabilités sur Internet. Certains mettent en place des portes dérobées pour assurer un accès ultérieur, tandis que d'autres utilisent des enregistreurs de frappe pour recueillir des informations délicates. Certains sont également employés pour l'envoi en masse de courriels de phishing ou de spams.
- **Méthode d'infection de la machine victime :** Les botnets exploitent différents mécanismes pour leur propagation. Les kits d'exploitation sont conçus pour exploiter les failles de sécurité des logiciels afin de pénétrer dans les systèmes. Les campagnes de spam comportent des liens ou des fichiers joints malveillants qui, une fois ouverts, mettent en place le logiciel malveillant. En définitive, les sites web piratés peuvent contaminer l'utilisateur dès sa visite, sans qu'il n'y ait d'interaction directe.

1.3.3 Composant d'un botnet

Un botnet regroupe plusieurs composants clés, parmi lesquelles figurent [26] :

- **Bot master :** Aussi appelé cybercriminel ou opérateur, c'est l'individu ou l'entité responsable de la conception, de la gestion et du fonctionnement du botnet. C'est lui qui conçoit le logiciel malveillant, dirige les attaques et supervise les opérations en utilisant des interfaces de commande. Il agit souvent dans l'ombre, utilisant des techniques pour cacher son identité et sa localisation [41].
- **Bots :** Les bots (ou zombies) sont des dispositifs infectés par un malware, constituant un réseau contrôlé par un bot master. Il peut s'agir d'ordinateurs, de serveurs, de smartphones ou même d'objets connectés (IoT). Ces appareils se distinguent par leur type de cible, leur mode de connexion (Internet, Bluetooth, SMS, etc.) et par un code malveillant souvent fragmenté. Ils exécutent automatiquement ou à distance des tâches prédéfinies, permettant aux cybercriminels d'exploiter des données, souvent à l'insu des utilisateurs [42].
- **Système de commande et de contrôle (C &C) :** Le C &C est l'élément central d'un botnet. Cela rend la communication entre le bot principal et les appareils compromis

(bots) plus simple, lui offrant la possibilité de coordonner leurs actions, de transmettre des instructions et de collecter les informations volées. Selon la structure du botnet, celle-ci peut être centralisée (avec un seul serveur) ou décentralisée (en utilisant un réseau P2P) [43].

- **Interface homme-machine (HMI) :** C'est un outil centralisé que le bot master emploie pour communiquer avec les machines défaillantes. Elle offre une interface élémentaire qui autorise la configuration du programme malveillant, l'élaboration de charges utiles pour diriger de nouvelles attaques, le suivi de l'état du réseau des bots et la mise en œuvre d'instructions. Cette plateforme facilite l'automatisation de différentes tâches, rendant la gestion, le développement et l'utilisation du botnet plus simples, même pour des agresseurs avec moins de compétences[44].

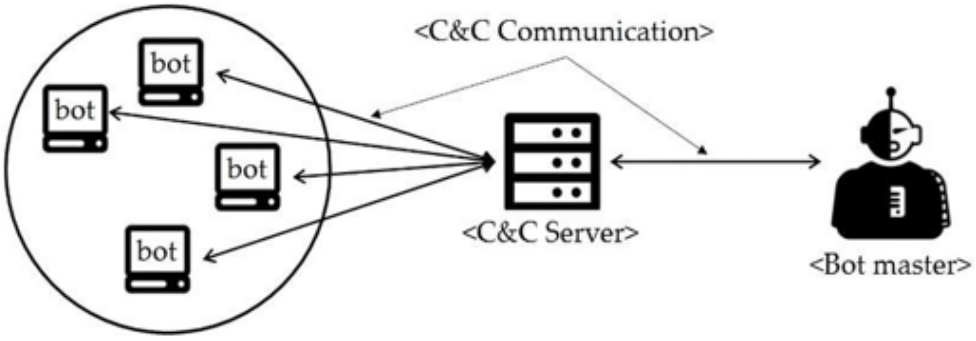

Figure 1.8 – Composant d'un botnet [45].

1.3.4 Les d'architecture d'un botnet :

Il existe divers modèles d'architecture pour les botnets, qui définissent la manière dont les bots infectés interagissent avec le serveur de contrôle. Voici les principaux modèles :

- **Architecture centralisée (client-serveur) :** Dans ce modèle, une liaison est établie avec un seul serveur qui joue le rôle de bot master. Ce serveur gère la transmission de commandes et d'informations entre les bots, instaurant ainsi un système de commandement et de contrôle (C &C). Ce modèle s'appuie sur un programme spécifique qui permet au bot master de maintenir la supervision de tous les bots qu'il administre. Identifier un botnet qui utilise une structure client-serveur est assez simple, étant donné qu'il dispose d'un seul point de contrôle. C'est pour cette raison que certains cybercriminels optent pour des schémas plus pénibles à détecter et à neutraliser [46].

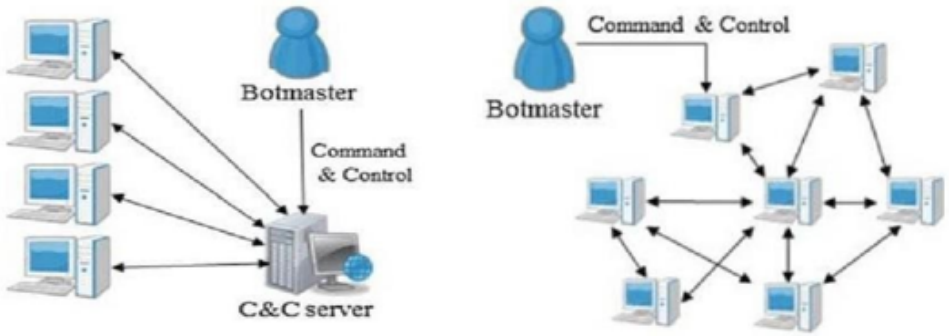

Figure 1.9 – Architecture centralisée [47].

- **Architecture décentralisée (P2P) :** Le modèle P2P (peer-to-peer) distribue les tâches de contrôle entre tous les ordinateurs zombies. Tant que le créateur du botnet peut se connecter à l'un des ordinateurs infectés, il peut transmettre des instructions aux autres. Ce système offre une plus grande discrétion quant à l'identité du cybercriminel. Aujourd'hui, l'architecture P2P, qui présente des avantages indéniables par rapport aux modèles centralisés, est devenue plus répandue et plus difficile à détecter [48].

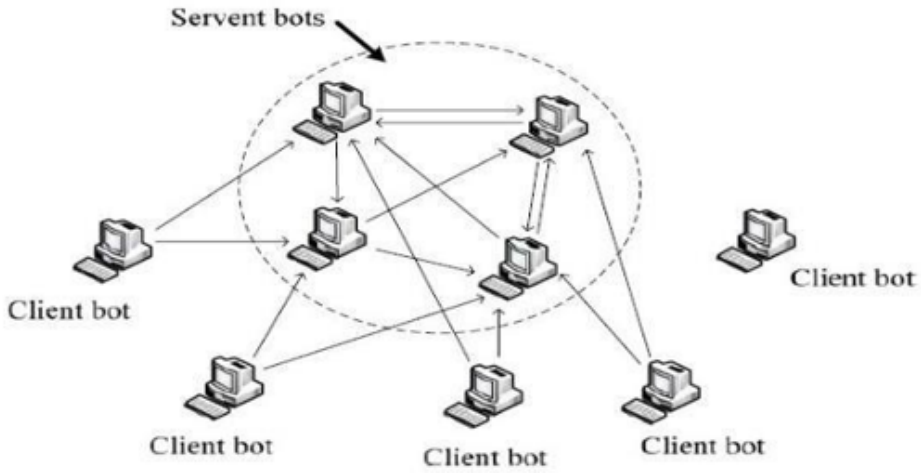

Figure 1.10 – Architecture décentralisée (P2P) [49].

- **Architecture hybride :** Ce modèle combine de manière optimale les avantages des architectures centralisées (stabilité du contrôle) et décentralisées (résilience P2P). Il utilise certains serveurs C &C discrets pour la phase d'initialisation tout en favorisant une communication décentralisée entre les bots.Son architecture binaire intègre des mécanismes de dissimulation avancés et conserve une redondance fonctionnelle constante,

même suite à la neutralisation partielle de ses infrastructures , ce qui explique son adoption massive et son faible taux de détection [50].

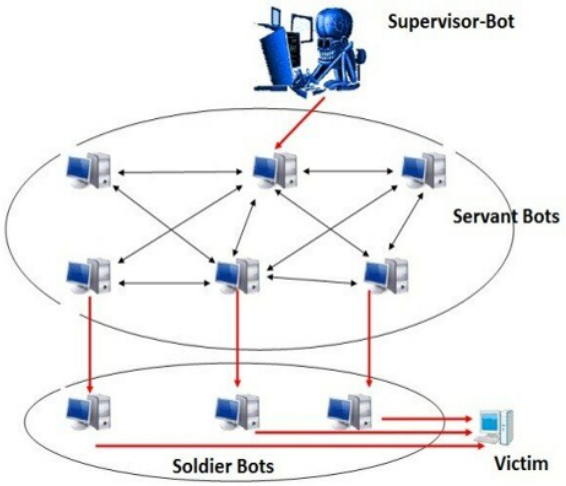

Figure 1.11 – Architecture hybride [51].

1.3.5 Fonctionnement des attaques par botnet

Un botnet suit généralement trois phases de fonctionnement [52],[53] :

— **Phase 1 : Préparation et exposition**

Les cybercriminels utilisent les vulnérabilités des sites internet, des applications ou même les comportements de l'homme pour infecter les utilisateurs. Ils emploient différentes techniques, comme l'exploitation de failles logicielles, l'envoi de faux courriels (phishing) ou encore la compromission de sites web légitimes pour réaliser des téléchargements dissimulés.

— **Phase 2 : Infection**

L'utilisateur est contaminé lorsqu'il installe ou lance un programme malveillant, souvent un virus de type cheval de Troie, ou quand il navigue sur un site piraté. Dans certaines situations, l'infection peut survenir sans action de l'utilisateur, par le biais de vulnérabilités exploitables à distance. Dès qu'il est infecté, l'appareil se transforme en bot capable d'exécuter des commandes néfastes.

— **Phase 3 : Activation et contrôle**

Après la création du réseau de bots, le bot master le dirige à distance grâce à une structure de commandement et de contrôle (CC). Il peut donc le mettre à profit pour diverses

actions nuisibles. Les botnets contemporains peuvent également être commercialisés ou loués à d'autres cybercriminels via des plateformes illicites.

1.3.6 Types d'attaques

On peut utiliser des botnets pour réaliser diverses attaques dans le domaine de l'informatique, comme :

- **Attaques par déni de service distribué (DDoS)**

  On utilise souvent les botnets pour mener des attaques DDoS, qui consistent à saturer un serveur ou un réseau avec une quantité considérable de requêtes nuisibles, ce qui rend le service indisponible. En utilisant un vaste réseau d'appareils compromis, les attaquants intensifient l'effet de ces impacts, visant particulièrement des sociétés, des organisations gouvernementales ou des infrastructures vitales. Ces offensives peuvent causer des dommages financiers majeurs et perturber des services vitaux [54].

- **Spam et phishing**

  Les botnets sont couramment utilisés pour propager des campagnes de phishing et de spam. Des millions de courriels frauduleux contenant des liens malintentionnés ou des pièces jointes contaminées sont envoyés par les cybercriminels. Ces méthodes de manipulation incitent les victimes à révéler des informations confidentielles (comme leurs mots de passe ou leurs coordonnées bancaires) ou à télécharger des programmes nuisibles. Des botnets comme Cutwail et Rustock ont été couramment employés pour diffuser ces dangers à une échelle considérable [55].

- **Fraude au clic (Click Fraud)**

  Il existe des botnets qui se spécialisent dans la fraude au clic, créant de manière artificielle des profits publicitaires en reproduisant des clics sur des publicités en ligne. Les malfaiteurs utilisent les dispositifs compromis pour naviguer sur des sites web ou interagir avec des publicités sans l'accord des utilisateurs, ce qui déforme les données statistiques et les bénéfices des plateformes publicitaires [56].

- **Vol d'informations sensibles**

  Les botnets ont la capacité d'extraire des informations sensibles (identifiants, données financières, informations personnelles) en employant des méthodes comme le keylogging ou la capture d'écran. Parmi les exemples notables, on retrouve Zeus et SDBot, qui

visent à dérober les identifiants de connexion des utilisateurs ou leurs informations de carte bancaire [57].

- **Exploitation des ressources système**

  Des tâches malveillantes telles que l'extraction de cryptomonnaies, la création de faux votes sur internet ou la falsification de données web peuvent être réalisées par des appareils compromis dans un botnet. Ces actions exploitent les ressources des dispositifs compromis sans le savoir de leurs propriétaires, diminuant ainsi leur efficacité et augmentant leurs dépenses énergétiques [58].

- **Cyber-guerre et attaques ciblées**

  Des botnets sont parfois employés dans le contexte de cyber-guerres ou d'espionnage industriel. Des entités gouvernementales ou des collectifs structurés sont susceptibles de mettre en œuvre des botnets pour nuire à des infrastructures adverses, dérober des secrets de fabrication ou orchestrer des attaques persistantes (APT). Ces cyberattaques élaborées constituent un danger prépondérant pour la sécurité économique et nationale [59].

1.3.7 Techniques de détection de botnets

La détection des botnets repose sur diverses techniques, chacune présentant ses avantages et ses limites. Ces approches se différencient selon la nature des données exploitées ou la méthode d'analyse appliquée, telles que l'analyse du trafic réseau, la détection basée sur les signatures, l'analyse comportementale, l'examen des données DNS ou encore l'utilisation de l'apprentissage automatique. Chaque technique est illustrée à travers des exemples issus de la littérature scientifique.

- **Analyse du trafic réseau :** Une méthode de détection des botnets qui consiste à surveiller les communications entre les appareils connectés et le réseau afin d'identifier des comportements inhabituels. En examinant les flux de données, cette technique permet de repérer des signes d'activité malveillante, comme des pics soudains de bande passante, des connexions fréquentes vers des adresses IP suspectes, ou des schémas de communication caractéristiques d'un botnet. Des outils comme Wireshark ou CICFlowMeter sont couramment utilisés pour capturer et analyser ces flux, facilitant ainsi la détection précoce des menaces [60].

  Le travail présentés par les auteurs K. Singh et al.[61] repose sur l'analyse du trafic réseau pour la détection des botnets.

- **Détection basée sur les signatures :** Cette méthode identifie les botnets en recherchant dans le trafic réseau ou les fichiers exécutables des motifs spécifiques de logiciels malveillants, tels que des séquences de code, des adresses IP ou des domaines de commande et contrôle (C &C). Elle repose sur une base de signatures préenregistrées correspondant à des menaces connues. Cette approche est efficace pour détecter des attaques déjà identifiées, mais reste limitée face aux botnets nouveaux, polymorphes ou modifiés, qui ne correspondent pas aux signatures existantes [62].

  Par exemple, Behal (2015) [63] montre que les signatures statiques peuvent détecter des botnets comme Zeus, mais échouent contre des versions obfusquées (ex : Emotet v2)

- **Détection basée sur le comportement :** Cette méthode consiste à identifier les comportements inhabituels ou déviants dans le fonctionnement des systèmes informatiques. Elle repose sur l'observation de signes d'activité anormale tels qu'une utilisation excessive du processeur, des volumes de trafic réseau inhabituels, des connexions sortantes non sollicitées ou encore des changements soudains dans les modèles d'usage. Contrairement aux approches basées sur les signatures, cette technique permet de détecter des menaces encore inconnues, notamment des botnets émergents dont les empreintes ne sont pas encore répertoriées [64].

  L'étude menée par Ndiaye et al. (2023) [65] illustre cette approche en surveillant les contrats intelligents afin d'identifier des transactions suspectes.

- **Analyse de données DNS :** Permet de détecter des activités suspectes en examinant le trafic des requêtes DNS. Cette méthode identifie notamment des noms de domaine générés aléatoirement, des résolutions DNS très fréquentes ou des domaines contenant des mots-clés malveillants, qui sont autant d'indicateurs potentiels d'activités de botnets [66],[67],[68].

  L'approche CASTAFIOR de Morel et al. [25] illustre l'analyse statistique du trafic DNS pour repérer des comportements anormaux liés aux botnets.

- **Détection basée sur l'apprentissage automatique :** Cette technique utilise des algorithmes de machine learning pour analyser de grandes quantités de données et détecter des schémas de comportement anormaux, facilitant ainsi l'identification de botnets, même ceux qui ne sont pas encore connus [25].

  les travaux [61],[68],[69],[70],[71],[72]montrent que l'apprentissage automatique est une approche centrale et efficace dans la détection des botnets.

1.3.8 Exemples des botnets

Botnet	Définition/Description	Type d'attaque/Solution proposée
Neris [73]	Botnet utilisé pour mener des attaques de type DDoS, souvent via le détournement de serveurs web. Il peut également être utilisé pour exfiltrer des données sensibles.	DDoS, vol de données Solutions : détection comportementale, filtrage du trafic,systèmes IDS/IPS
Rbot [74]	Botnet basé sur des vers IRC. Il se propage en exploitant des vulnérabilités Windows, et permet à un attaquant de contrôler les machines infectées.	Propagation par ver,contrôle distant Solutions : mises à jour de sécurité, suppression des canaux IRC non autorisés, antivirus
Sogou [75]	Ce n'est pas un botnet classique, mais un crawler web chinois (lié au moteur de recherche Sogou). Cependant, il peut être abusé dans certains cas pour générer un trafic anormal.	Trafic automatisé (crawler) Solutions : limitation des crawlers, règles de pare-feu
Virut [76]	Botnet polymorphe capable de se propager via des exécutables Windows. Il ouvre des portes dérobées pour des attaques variées, y compris le spam, DDoS et le vol de données.	Propagation virale, contrôle distant Solutions : réinstallation système, antivirus à jour, blocage de C2
Menti [77]	Peu de sources documentées. Apparemment un trafic malveillant généré par un bot ou malware custom (utilisé dans certains datasets comme CTU-13).	Trafic inconnu ou personnalisé Solutions : détection comportementale, analyse de trafic anormal
Nsis [78]	Souvent référencé comme NSIS. Inject, utilisé pour injecter du code malveillant dans des installateurs Windows. Peut servir à propager d'autres malwares ou à contourner des protections.	Injection de malware Solutions : validation des installateurs, filtrage antivirus, contrôle de l'intégrité des logiciels installés
Mirai [79]	Botnet IoT célèbre pour avoir compromis des millions d'objets connectés faiblement sécurisés (caméras IP, routeurs) afin de mener des attaques DDoS massives.	DDoS IoT Solutions : durcissement IoT, changement des mots de passe par défaut, segmentation réseau
Gafgyt [80]	Botnet IoT proche de Mirai, ciblant des routeurs et objets connectés vulnérables. Utilisé pour DDoS et propagation rapide.	DDoS IoT Solutions : patchs firmware, surveillance réseau, détection comportementale

1.3.9 Prévention et protection contre les botnets

Stratégies de préventions contre les Botnets

Les mesures de sécurité pour prévenir les botnets comprennent des actions visant à diminuer le danger d'infection. Voici quelques approches essentielles [81],[82],[83] :

- Il est recommandé d'examiner systématiquement tous les fichiers téléchargés avant de les ouvrir, en utilisant des outils spécialisés capables d'analyser les fichiers pour détecter d'éventuels programmes malveillants.
- Pour assurer une prévention efficace, il convient d'utiliser un antivirus reconnu, capable d'analyser les pièces jointes et les téléchargements, tout en veillant à maintenir ce logiciel régulièrement à jour pour contrer les menaces les plus récentes.
- La mise en place d'un pare-feu configuré rigoureusement aide à filtrer et bloquer les connexions réseau suspectes susceptibles d'être utilisées par des Botnets pour communiquer avec leurs serveurs de commande.
- Lorsqu'un nouvel appareil connecté est ajouté au réseau, il faut impérativement changer ses identifiants par défaut et choisir des mots de passe solides et uniques, ce qui limite considérablement le risque d'accès non autorisé.
- Installer un système de détection d'intrusion (IDS) permet de surveiller en permanence le trafic réseau et de signaler toute activité inhabituelle qui pourrait indiquer une infection ou une attaque en cours.
- L'utilisation d'un réseau privé virtuel (VPN) permet de chiffrer les échanges de données et de masquer l'adresse IP, ce qui réduit la probabilité d'être repéré et ciblé par des Botnets, même si le VPN n'empêche pas directement l'infection.
- Il est crucial de maintenir à jour tous les logiciels et systèmes d'exploitation en appliquant rapidement les correctifs de sécurité fournis par les éditeurs pour combler les failles exploitées par les malwares.
- Former et sensibiliser les utilisateurs à identifier les emails frauduleux, ainsi que les liens ou pièces jointes suspects, contribue à prévenir les attaques de phishing,principale méthode d'entrée des Botnets dans un réseau.
- Segmenter le réseau en créant des zones distinctes, par exemple via des VLANs,limite la capacité d'un Botnet à se propager d'un appareil infecté à d'autres systèmes essentiels.

- Pour réduire les vecteurs d'attaque, il est important de désactiver tous les services et de fermer les ports réseau qui ne sont pas indispensables, ce qui diminue la surface d'exposition aux tentatives d'intrusion.

Protection contre les Botnets

La protection contre les botnets consiste à détecter et empêcher leurs activités après infiltration, afin de réduire les vulnérabilités, limiter les interruptions et protéger les données et systèmes. Pour cela, il faut tenir compte des éléments suivants [84],[85] :

- La protection contre les Botnets consiste à détecter et bloquer leurs activités une fois qu'ils ont déjà infiltré un système ou un réseau. Cette mesure permet de réduire les vulnérabilités, de limiter les interruptions de service et de protéger les données ainsi que les systèmes informatiques contre les attaques.
- Il est crucial de renforcer la collaboration avec les autorités compétentes et les organismes spécialisés en cybersécurité afin de solliciter leur aide et de mener des enquêtes approfondies pour identifier l'origine du Botnet.
- Les fournisseurs d'accès à Internet et les hébergeurs doivent pouvoir surveiller le trafic sur leurs réseaux, prévenir rapidement les utilisateurs infectés, et les aider à nettoyer leurs appareils. Cela permet de détecter vite les activités suspectes, d'informer les personnes concernées, et de supprimer les menaces .
- Pour une protection efficace, il faut disposer d'une méthodologie rigoureuse et d'un plan d'action structuré, dont l'efficacité peut être évaluée scientifiquement. Cela implique également la recherche de solutions pour limiter les efforts nécessaires et réduire les impacts secondaires sur les systèmes et les utilisateurs.

1.4 Intelligence artificielle (IA)

Dans cette section, on a introduit l'intelligence artificielle, défini le machine learning, expliqué ses principaux types d'apprentissage, et présenté quelques algorithmes de ML.

1.4.1 Définition de I'IA

L'intelligence artificielle (IA) désigne l'ensemble des théories, méthodes et technologies permettant de créer des systèmes capables de simuler des fonctions cognitives humaines,

comme le raisonnement, l'apprentissage ou la perception [86]. Parmi ses sous-domaines, on trouve le machine learning (apprentissage automatique), qui permet aux systèmes d'apprendre à partir de données, ainsi que le deep learning (apprentissage profond), une forme avancée de machine learning reposant sur des réseaux de neurones artificiels [87]. Aujourd'hui, l'IA est largement utilisée dans des domaines variés comme la santé, l'industrie les transports, où elle permet de développer des solutions à la fois innovantes et efficaces [88].

1.4.2 Définition Apprentissage automatique (ML)

Apprentissage automatique une branche de l'intelligence artificielle qui regroupe les techniques et algorithmes permettant aux machines d'apprendre à partir de données , sans avoir été explicitement programmées pour chaque tâche .Concrètement, il s'agit d'utiliser des données historiques pour entraîner des modèles mathématiques capables de faire des prédictions ou de prendre des décisions sur de nouvelles données [89].

Apprentissage supervisé

L'apprentissage supervisé est une approche de l'approntissage automatique dans laquelle un modèle est entrainé à partir de données étiquetées, c'est–à–dire pour lesquelles on connait déjà les résultats attendus. L'objectif est de permettre au modèle de faire des prédictions sur de nouvelles données en se basant sur ce qu'il a appris. On distingue principalement deux grands types de problèmes : la régression et la classification[90].

- **Régression :** La régression permet de prédire une valeur numérique continue. Elle est utilisée dans des situations ou la variable cible peut prendre une infinité de valeurs.
- **Classification :** La classification est utilisée pour attribuer une catégorie à chaque observation. Elle est adaptée lorsque la sortie attendue est une variable discrète.

Apprentissage non supervisé

L'apprentissage non supervisé désigne une approche dans laquelle un algorithme exploite un ensemble de données non étiquetées afin d'identifier automatiquement des schémas cachés ou des organisations naturelles dans les données. Cette méthode permet notamment de regrouper des données similaires (clustering) et de réduire la dimensionnalité tout en préservant l'essentiel de l'information [90].

- **Regroupement de données similaires (Clustering) :** Le clustering est une technique centrale de l'apprentissage non supervisé. Il permet de regrouper automatiquement des données ayant des caractéristiques communes sans connaissance préalable des catégories. Ce procédé est utilisé, par exemple, pour détecter des segments clients, analyser des comportements utilisateurs ou classer des documents.
- **Réduction de la dimensionnalité :** Une autre application de l'apprentissage non supervisé consiste à réduire la dimensionnalité d'un jeu de données, c'est-à-dire à simplifier sa structure tout en conservant les informations essentielles. Cela facilite la visualisation, la compression ou le prétraitement des données.

Apprentissage semi-supervisé

L'apprentissage semi-supervisé est une approche qui combine les caractéristiques des apprentissages supervisé et non supervisé. Dans ce cadre, un modèle est formé à partir d'un mélange de données étiquetées et non étiquetées. L'algorithme tente d'apprendre la structure des données à partir de l'ensemble non étiqueté tout en cherchant à prédire les étiquettes des données étiquetées [91].

Apprentissage par renforcement

L'apprentissage par renforcement est une branche de l'intelligence artificielle où un agent apprend à prendre des décisions en interagissant avec son environnement. Il progresse par essais et erreurs dans le but d'obtenir la récompense la plus élevée possible. Lorsqu'on y associe les réseaux neuronaux profonds, on parle d'apprentissage par renforcement profond : dans ce cas, c'est un réseau neuronal qui remplace l'agent pour l'aider à apprendre des situations complexes et prendre des décisions plus efficaces [92].

1.4.3 Définition de l'apprentissage profond (Deep Learning) :

Le deep learning, ou apprentissage profond, est une branche avancée de l'intelligence artificielle, issue du machine learning (apprentissage automatique), qui permet à une machine d'apprendre par elle-même à partir de données, sans se limiter à exécuter des règles prédéfinies [93]. Cette technologie repose sur des réseaux de neurones artificiels comportant de nombreuses couches successives, capables de modéliser des représentations complexes et de reproduire certains mécanismes d'apprentissage du cerveau humain [94].

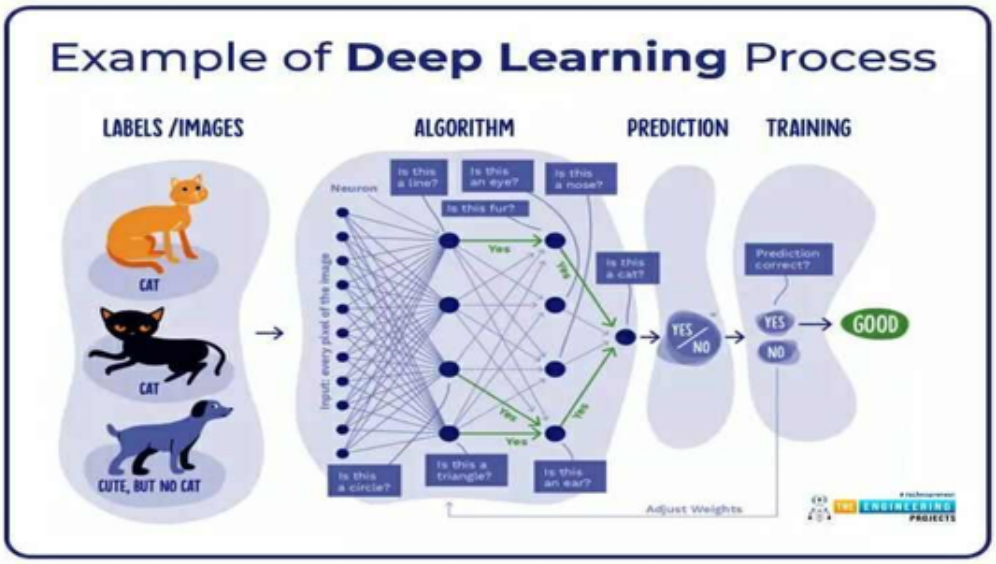

Figure 1.12 – Processus d'apprentissage profond [95].

1.4.4 Fonctionnement du Deep Learning :

Le deep learning repose sur l'utilisation de réseaux de neurones artificiels organisés en couches hiérarchiques. Chaque couche de neurones traite et interprète les informations issues de la couche précédente. Grâce à un entraînement sur de vastes ensembles de données, le modèle ajuste progressivement les connexions entre les neurones, ce qui lui permet de reconnaître des structures complexes. Par exemple, il peut d'abord identifier des formes simples, comme des lettres ou des contours, avant de reconnaître des objets entiers ou des visages [93].

Une fois entraîné, un modèle de deep learning est capable de traiter de nouvelles données et de réaliser des analyses en temps réel, sans intervention humaine. Les processeurs graphiques (GPU), capables de gérer de nombreux calculs parallèles, sont particulièrement adaptés pour accélérer cet entraînement et permettre aux réseaux profonds de converger plus rapidement [96].

1.4.5 Définition d'un réseau neuronal :

Un réseau neuronal est une technique d'intelligence artificielle qui permet à un ordinateur de "raisonner" d'une manière similaire à celle du cerveau humain. Il s'appuie sur un ensemble de nœuds, ou neurones artificiels, organisés en plusieurs couches. Chaque neurone reçoit des informations, les transforme légèrement à l'aide d'une fonction d'activation, puis transmet le

résultat aux couches suivantes. Ce fonctionnement forme un système adaptatif, où le modèle repère ses erreurs, ajuste ses paramètres, et s'améliore à chaque itération. Grâce à cette approche, les réseaux neuronaux sont capables de résoudre des problèmes complexes tels que résumer un article, reconnaître un visage ou analyser une voix avec une précision croissante [97].

Architecture d'un réseau neuronal :

Les réseaux neuronaux artificiels se composent de plusieurs couches interconnectées de neurones artificiels, appelés nœuds, organisés en une architecture hiérarchique .

Un réseau neuronal de base comprend trois types de couches :

- La **couche d'entrée**, qui reçoit les données brutes et les transmet aux couches suivantes.
- Les **couches cachées**, en nombre variable, qui transforment les données de manière progressive en extrayant des caractéristiques de plus en plus complexes.
- La **couche de sortie**, qui génère la prédiction finale, qu'il s'agisse d'une classification binaire, multi-classe ou d'une régression.

Les réseaux neuronaux profonds, ou deep neural networks, se distinguent par la présence de nombreuses couches cachées, pouvant contenir des millions de neurones interconnectés. Les connexions entre les nœuds sont associées à des poids (positifs ou négatifs), qui reflètent l'influence exercée par chaque entrée. Ces architectures permettent de modéliser des relations complexes entre les données, mais nécessitent des volumes importants de données d'entraînement et une puissance de calcul importante pour optimiser correctement les poids [97].

Fonctionnement d'un réseau de neuronal :

Le fonctionnement d'un réseau neuronal repose sur un mécanisme mathématique inspiré de la régression linéaire. Chaque nœud (ou neurone artificiel) reçoit plusieurs entrées pondérées, auxquelles s'ajoute un biais. Le résultat de cette combinaison est ensuite transformé par une fonction d'activation, qui détermine si le neurone "s'active" et transmet l'information aux couches suivantes. Ce mécanisme permet au réseau de sélectionner les informations les plus pertinentes à chaque étape du traitement. Les fonctions d'activation jouent un rôle essentiel en introduisant de la non-linéarité dans le réseau, permettant ainsi de modéliser des relations complexes entre les variables d'entrée et la sortie. Parmi les plus couramment utilisées, on trouve :

- **ReLU (Rectified Linear Unit) :** c'est une fonction d'activation utilisées dans les couches internes .Elle est non-linéaire évite le phénomène du gradient nul, elle renvoie 0 si l'entrée est négative, et l'entrée elle-même si elle est positive. Elle est très utilisée pour sa simplicité et son efficacité dans les réseaux profonds.
- **Activation linéaire :** En apprentissage automatique, l'activation linéaire est une fonction qui renvoie directement la valeur d'entrée sans la modifier. Mathématiquement : f(x)=x.

  Elle ne crée aucune non-linéarité, ce qui limite la capacité du modèle à apprendre des relations complexes.
- **Sigmoïde :** elle transforme l'entrée en une valeur comprise entre 0 et 1, utile pour les problèmes de classification binaire.
- **Tanh (Tangente hyperbolique) :** elle génère une sortie entre -1 et 1, offrant une alternative à la sigmoïde avec une meilleure centration des données.

Les pondérations des connexions entre les nœuds sont ajustées progressivement durant l'apprentissage. Ce processus utilise la descente de gradient pour réduire l'erreur entre la sortie prédite et la sortie réelle, mesurée par une fonction de coût (souvent l'erreur quadratique moyenne, MSE).

Un élément essentiel de cet apprentissage est la rétropropagation, qui propage l'erreur de la couche de sortie vers les couches précédentes, permettant d'ajuster efficacement l'ensemble des paramètres du réseau.

Bien que la majorité des réseaux neuronaux fonctionnent en propagation avant (de l'entrée vers la sortie), c'est grâce à la rétropropagation, lors de la phase d'apprentissage, que le réseau améliore progressivement ses prédictions. Cela lui permet de résoudre des tâches complexes telles que la reconnaissance d'images, la classification de textes ou la prédiction de séquences temporelles [97],[98].

1.4.6 La différence entre le Deep learning et les réseaux neuronaux :

Le deep learning désigne simplement l'utilisation de réseaux neuronaux profonds, comportant un grand nombre de couches cachées. Ces couches permettent d'extraire des caractéristiques de plus en plus abstraites à partir des données (images, sons, texte, etc.). Par rapport à un réseau neuronal "classique", limité à une ou deux couches, cette profondeur accrue permet des classifications plus fines (par exemple, reconnaître non seulement un oiseau,

mais son espèce). Toutefois, cela requiert de vastes ensembles de données d'apprentissage et une puissance de calcul importante, notamment grâce aux GPU optimisés pour les calculs parallèles [96].

1.4.7 Les limites du Deep Learning :

Parmi les limites du deep learning on cite [94],[96] :

- **Besoins massifs en données :** le deep learning nécessite de très grandes quantités de données pour obtenir des résultats pertinents. À l'image de l'apprentissage humain, les modèles doivent être exposés à de nombreux exemples pour améliorer leurs performances.
- **Manque de flexibilité :** : les réseaux de deep learning restent spécialisés sur des tâches précises. Lorsqu'ils sont confrontés à des situations différentes de celles rencontrées durant l'entraînement, ils ont souvent du mal à s'adapter.
- **Manque de transparence :** en raison du nombre élevé de paramètres traités, il est difficile de comprendre comment un réseau neuronal parvient à ses conclusions. Cette opacité ("effet boîte noire") rend l'explication des prédictions complexe et complique la détection de biais.
- **Surapprentissage (overfitting) :** les modèles profonds peuvent facilement surapprendre, c'est-à-dire mémoriser les données d'entraînement sans bien généraliser aux nouvelles données, ce qui limite leur efficacité en production.

Malgré ces obstacles, les progrès dans le domaine (régularisation, dropout, apprentissage non supervisé) permettent de rendre les modèles de deep learning de plus en plus performants, robustes et rapides.

Conclusion

Ce premier chapitre a permis d'introduire les principaux concepts nécessaires à la compréhension des enjeux liés à la détection des botnets dans les environnements IoT. En explorant l'architecture de l'Internet des Objets, ses composants, ses technologies de communication et les défis qu'il pose en matière de sécurité, nous avons mis en évidence la vulnérabilité croissante des dispositifs connectés. L'étude du trafic réseau et des botnets a permis de mieux cerner les vecteurs d'attaque et les mécanismes utilisés par les cybercriminels. Enfin, l'introduction de l'intelligence artificielle et des techniques d'apprentissage automatique a ouvert des perspectives nouvelles pour la détection proactive des menaces. Les éléments présentés ici constituent le socle théorique sur lequel s'appuieront les chapitres suivants pour développer des approches concrètes de détection et de protection contre les botnets IoT.

Chapitre 2

La détection des botnet dans la littératures scientifique

Introduction

La détection des botnets est un enjeu majeur de la cybersécurité moderne, en particulier face à l'évolution constante des techniques d'attaque et à la sophistication croissante des malwares. Ce chapitre propose un état de l'art des approches utilisées dans la littérature scientifique pour détecter ces menaces, en s'appuyant sur des techniques d'apprentissage automatique (machine learning), d'apprentissage profond (deep learning), et des méthodes hybrides. Une attention particulière sera portée à l'algorithme Random Forest, souvent cité pour sa robustesse et sa précision. Après avoir présenté les principaux algorithmes utilisés, nous passerons en revue plusieurs travaux récents qui ont appliqué ces techniques à la détection des botnets dans différents contextes (réseaux IoT, SDN, P2P). Enfin, une synthèse comparative permettra d'identifier les approches les plus efficaces selon les critères de performance.

2.1 Présentation des algorithmes utilisés

Dans cette section, nous allons présenter les algorithmes utilisés dans la littérature pour la détection des botnets.

2.1.1 Algorithmes d'apprentissage automatique (ML)

— **Arbre de décision (DT) [99] :** Un arbre de décision est un algorithme d'apprentissage supervisé utilisé pour des tâches de classification et de régression. C'est un modèle prédictif structuré sous forme d'arborescence, où chaque nœud interne représente un attribut ou une caractéristique des données, chaque branche correspond à une règle de décision, et chaque feuille indique une étiquette de classe ou une valeur de sortie. Le principe de fonctionnement repose sur une série de décisions successives qui permettent de segmenter les données en sous-groupes de plus en plus homogènes. À chaque étape, l'attribut le plus pertinent est sélectionné pour diviser l'ensemble des données, selon un critère de séparation (comme l'entropie ou l'indice de Gini), jusqu'à atteindre une prédiction finale.

Parmi les types d'arbres de décision les plus utilisés, on peut citer :

- **CART (Classification And Regression Trees) :** cet algorithme construit des arbres binaires. It can also be used well for tasks of classification and of regression, dividing the data into two branches at each step.
- **ID3 :** il s'agit d'un algorithme qui construit un arbre en testant les différentes propriétés des objets à classer. Il cherche à déterminer, étape par étape, la meilleure façon de séparer les données selon leurs caractéristiques.
- **C4.5 :** cet algorithme est une amélioration de ID3. Il génère un arbre de décision en divisant récursivement les données selon les attributs qui offrent le meilleur gain d'information parmi tous les tests possibles.

— **Foret aléatoire (RF) [100] :**La forêt aléatoire, ou Random Forest, est un algorithme d'apprentissage supervisé très apprécié, pour les tâches de classification, de régression, de détection d'anomalies ou encore de sélection de variables. Il s'appuyer sur un plusieurs arbres de décision, construit à partir d'échantillons aléatoires des données. Ensuite, il combine les prédictions de tous ces arbres pour produire un résultat final plus fiable et plus robuste.

Développé au début des années 2000 par **Leo Breiman** et **Adele Cutler**, cet algorithme repose sur une technique appelée bagging (bootstrap aggregating). Celle-ci consiste à entraîner plusieurs modèles sur des sous-échantillons aléatoires du jeu de données initial, puis à agréger leurs prédictions. De plus, à chaque nœud de chaque arbre, un sous- ensemble aléatoire de variables est utilisé pour déterminer la meilleure

séparation. Cette double randomisation – au niveau des données et des variables – confère à la forêt aléatoire une forte résistance au surapprentissage (overfitting).

En classification, chaque arbre vote pour une classe, et la classe majoritaire est retenue comme prédiction finale. En régression, on calcule la moyenne des prédictions numériques de l'ensemble des arbres. Ce mécanisme permet de réduire les erreurs individuelles et d'améliorer la stabilité des résultats.

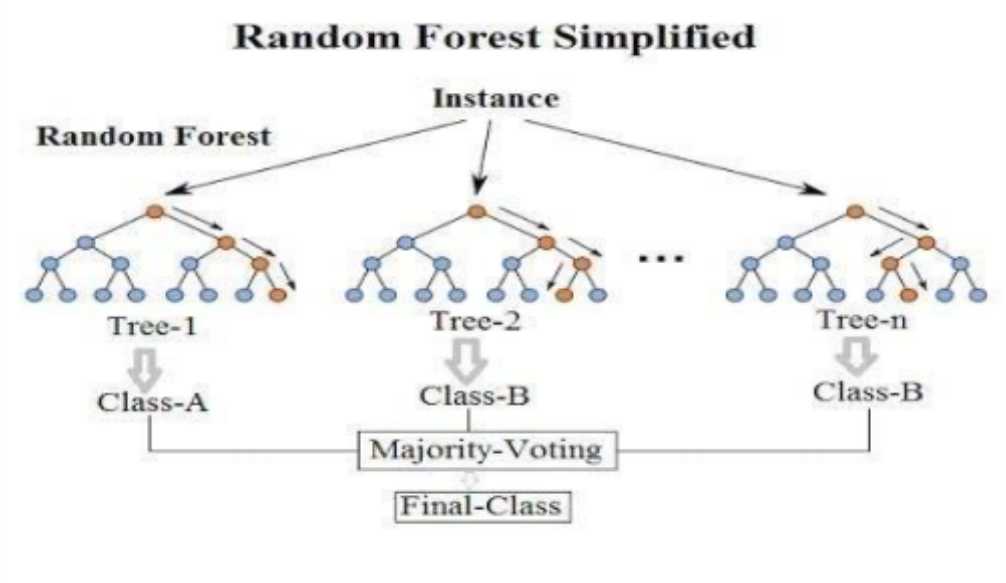

Figure 2.1 – Architecture d'une forêt aléatoire [101].

Ce modèle sera particulièrement exploré dans notre propre démarche, car il a montré d'excellentes performances dans de nombreux travaux antérieurs sur la détection des botnets.

— **Gradient Boosting (XGBoost)[102] :**XGBoost est un algorithme de machine learning performant, issu du gradient boosting, destiné à traiter des problèmes de classification ou de régression. Il fonctionne en élaborant une suite d'arbres décisionnels (apprenants faibles), où chaque arbre rectifie les fautes du précédent en anticipant les résidus. À la différence du Random Forest, il fonctionne de façon séquentielle et incorpore un processus d'élagage destiné à éliminer les arbres de faible performance. Son caractère distinctif provient de son efficacité en matière de rapidité et de performance, grâce à la parallélisation, l'optimisation du cache et le traitement distribué. Ces caractéristiques le rendent approprié pour la gestion de grandes quantités de données

(Big Data). XGBoost s'appuie sur une technique d'assemblage de modèles, fusionnant plusieurs prédicteurs afin d'obtenir une prédiction plus exacte.

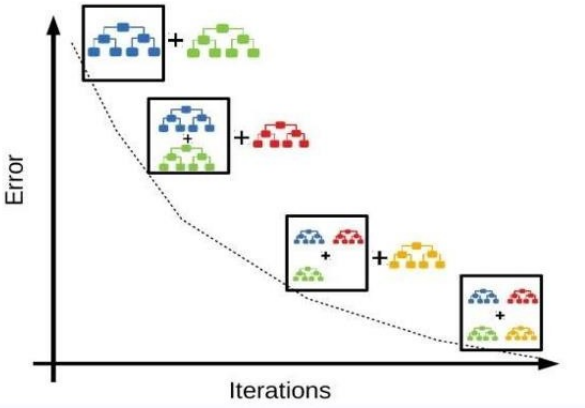

Figure 2.2 – Réduction de l'erreur par boosting [103].

— **K-plus proches voisins (KNN) [104] :**est l'un des algorithmes supervisés aussi, utilisé pour la classification et la régression. Il ne nécessite pas de phase d'entraînement préalable. Pour prédire une nouvelle donnée, il calcule la distance (euclidienne, Manhattan,etc) entre cette dernière et les autres points du jeu d'entraînement, puis sélectionne les k voisins les plus proches. En classification, la classe majoritaire parmi ces voisins est retenue, tandis qu'en régression, c'est la moyenne des valeurs qui est utilisée. Le choix du paramètre k se fait généralement par validation croisée.

— **La régression linéaire (LR) [105] :**est une méthode d'apprentissage supervisé pour modéliser la relation entre une variable dépendante continue et une ou plusieurs variables indépendantes, qu'elles soient continues ou catégorielles. Elle vise à ajuster une droite ou un hyperplan dans le cas de plusieurs variables afin de minimiser l'écart entre les valeurs prédites et les valeurs réelles

— **K-moyen (K-means)[106] :** Le clustering, également connu sous le nom de regroupement, est une technique non supervisée d'analyse de données qui a pour objectif de séparer un ensemble de données en sous-groupes homogènes (clusters). Les composants d'un cluster donné montrent une grande ressemblance entre eux, alors que ceux de clusters distincts montrent des différences perceptibles. K-means se démarque parmi les techniques de regroupement par sa simplicité et son efficacité. Il fonctionne en divisant les données en K groupes, le nombre K étant préétabli.

Chapitre 3

Contribution, Résultats et Discussion.

Introduction

Après l'étude approfondie des travaux existants sur la détection des botnets, présentée dans le chapitre précédent, il est apparu que les modèles supervisés tels que Random Forest (RF) ou XGBoost offrent d'excellentes performances sur des datasets propres et bien étiquetés, mais présentent des limites lorsque le trafic est hétérogène, bruité, ou lorsque les attaques sont inconnues. Inversement, les approches basées sur des auto-encodeurs (AE, VAE, RVAE) montrent une meilleure capacité à généraliser face à des menaces nouvelles, mais leur précision reste souvent en deçà des méthodes supervisées lorsque les labels sont disponibles.

À partir de cette observation et après une analyse comparative des articles récents sur le sujet, nous avons été amenés à réfléchir à une approche hybride combinant les avantages des méthodes supervisées et non supervisées. C'est dans ce cadre qu'est née l'idée de développer une architecture appelée kRAST (KMeans–Random forest–Autoencoder–Stacking Technique), qui intègre de manière cohérente ces deux paradigmes. L'approche kRAST repose sur un auto-encodeur classique permettant de capturer les représentations latentes du trafic réseau, suivi d'un algorithme de clustering K-Means pour identifier et regrouper les comportements types. Enfin, une étape de classification supervisée est assurée par deux modèles performants, Random Forest et XGBoost, dont les prédictions sont agrégées au sein d'une architecture de stacking afin d'améliorer la robustesse et la précision du système de détection.

Notre objectif est donc de développer une méthode efficace et rapide pour détecter les

botnets dans des trafics réseau complexes et variés, en tirant parti de la complémentarité entre apprentissage non supervisé et supervisé.

Notre objectif est donc de développer une méthode efficace et rapide pour détecter les botnets dans des trafics réseau complexes et variés, en tirant parti de la complémentarité entre apprentissage non supervisé et supervisé.

Dans ce chapitre, nous allons détailler l'architecture de l'approche KRAST, son implémentation optimisée sur TPU, évaluer les performances du modèle, le comparer à des études existantes, et proposer des perspectives d'amélioration.

3.1 Origine et motivation de l'approche hybride

3.1.1 Résultats d'application du RF sur différentes datasets :

Dataset	**Accuracy**	**Précision(%)**	Rappel(%)	**F1-score (%)**	**AUC(%)**	**FN**	**FP**
N-BaIot	100	100	100	100	100	0	0
UNSW-NB15	99.89	100	100	100	100	12	1
TON-IoT	99.98	99.99	99.98	99.99	99.99	2	6
Iot-23	99.73	99.99	99.72	99.84	100	1697	174
CICIDS2017	96.93	96.94	96.92	96.93	98.99	21660	21910
CTU-13	96.88	44.23	97.45	60.85	99.48	2002	96779

Tableau 3.1 – Résultats obtenus avec Random Forest sur différents datasets.

Dans ce tableau (Tableau 3.1) Le modèle Random Forest (RF) s'est révélé particulièrement efficace pour détecter les attaques de type botnet, notamment dans des environnements IoT où les données sont bien structurées et correctement étiquetées. Lors de nos expérimentations, nous l'avons appliqué à plusieurs jeux de données de référence, comme N-BaIoT [107] ou TON-IoT [108], et obtenu des résultats remarquables. Dans ces contextes bien définis, où les comportements malveillants sont clairement distincts du trafic normal, RF a atteint des performances proches de la perfection, dépassant les 99 % en précision, rappel, F1-score et AUC – et atteignant même parfois 100 %. Ces résultats témoignent de la capacité du modèle à exploiter des données propres, équilibrées et bien annotées.

Mais ces performances exceptionnelles ne se maintiennent pas dans des environnements

plus complexes. Dès que l'on passe à des datasets comme CICIDS2017 [109] ou surtout CTU-13 [25], les limites du modèle deviennent évidentes. Le dataset CTU-13 [110] en particulier illustre bien cette situation : il contient un trafic réseau riche, varié, bruité et partiellement étiqueté, où se mêlent trafic normal, trafic de fond (background) et multiples attaques. Dans ce contexte, bien que le modèle parvient à maintenir un rappel élevé (97,45 %), sa précision chute fortement (44,23 %), entraînant un grand nombre de faux positifs – un problème majeur pour la fiabilité du système de détection.

Ces constats montrent que si RF est très performant sur des données bien cadrées, il devient rapidement moins efficace dès que l'environnement se complexifie. Pour l'utiliser dans des contextes plus réalistes, il est indispensable de lui adjoindre des étapes de prétraitement avancées (réduction de dimension, équilibrage des classes, etc.) et surtout de le combiner à des approches non supervisées, capables de mieux s'adapter à l'imprévisibilité des données.

C'est précisément pour cette raison que nous avons choisi de concentrer notre étude sur le dataset CTU-13 [110]. Malgré sa complexité, il reflète bien les conditions réelles d'un réseau confronté à des attaques. Il couvre sept types de botnets différents, avec une grande diversité de comportements malveillants, et une coexistence entre trafic normal, background et malveillant. Ces caractéristiques en font un excellent terrain d'expérimentation pour tester des approches plus efficaces, capables de dépasser les limites des méthodes classiques.

En travaillant sur ce dataset, notre objectif est de concevoir une solution plus robuste et plus flexible, capable de maintenir des performances fiables malgré la présence de bruit dans les données, d'un étiquetage partiel et d'un fort déséquilibre des classes — en somme, une approche réellement applicable en conditions réelles.

3.1.2 Limites observées dans le trafic complexe CTU-13

Le dataset CTU-13 [110] présente un certain nombre de limites structurelles qui rendent difficile l'application directe de modèles supervisés comme Random Forest. Voici les principales difficultés rencontrées :

- **Trafic très hétérogène :** le dataset contient trois types de trafic différents : normal, background (trafic de fond), et attaque. Le trafic de fond, non clairement étiqueté.

- **Étiquetage flou ou incomplet :** certains fichiers ne précisent pas clairement la nature des flux. Il arrive que des connexions malveillantes soient mal étiquetées ou mélangées avec du trafic légitime.
- **Multiplicité des attaques :** CTU-13 [110] inclut jusqu'à 7 types de botnets différents, chacun avec des comportements variés. Cette diversité rend la tâche plus complexe pour un modèle supervisé, surtout si certains types de botnet sont sous-représentés.
- **Difficulté à distinguer normal et botnet :** certains flux d'attaque imitent le comportement normal, rendant leur détection très difficile pour un classifieur basé uniquement sur les étiquettes.
- **Grand nombre de faux positifs :** lors de l'application du modèle Random Forest, nous avons observé une forte chute de la précision due à un nombre élevé de faux positifs. Cela montre que RF a du mal à généraliser dans ce contexte réaliste et bruité.

3.1.3 Motivation pour une approche hybride adaptée au trafic complexe

Constat :L'analyse du dataset CTU-13 [110] montre clairement que les approches supervisées, comme Random Forest, atteignent leurs limites face à un trafic complexe, bruité, mal équilibré et partiellement étiqueté. Le grand nombre de faux positifs observés illustre la difficulté du modèle à distinguer les attaques des comportements normaux dans un environnement aussi hétérogène.

Intuition : Cette situation suggère qu'il ne suffit plus d'apprendre à partir des étiquettes, souvent incomplètes ou imprécises. Il devient nécessaire d'adopter une approche plus efficace et flexible, capable de découvrir par elle-même des structures cachées dans les données. Les méthodes non supervisées, comme le clustering ou les auto-encodeurs, apparaissent alors comme des outils essentiels pour détecter les comportements anormaux, sans dépendre exclusivement des étiquettes. Elles permettent de capturer des motifs subtils et de mieux s'adapter à la réalité du trafic réseau.

3.2 Présentation de l'approche hybride proposée

3.2.1 Comparaison entre apprentissage supervisé et non supervisé

Dans ce tableau (Tableau 3.2) , nous allons présenter une comparaison entre les approches d'apprentissage supervisé et non supervisé, en mettant en évidence leurs principales caractéristiques, avantages et limites, dans le cadre spécifique de l'analyse de trafic réseau complexe.

Aspect	**Non supervisé**	**Supervisé**
Étiquetage	Non	Oui
Adaptation des données	Excellent(CTU-13)	Faible
Précision	Moyenne	Elevée
Capacité à structuré le Dataset	Forte	Faible
Complémentarité	Prépare et structure les données sans étiquette	Exploite la structure pour apprendre et prédire
Robustesse aux données Bruitées	Bonne (tolère le bruit structurel)	Faible (sensibilité au bruit et aux erreurs d'étiquetage)
Besoins en données	Fonctionne même avec peu ou pas d'étiquettes	Nécessite un jeu labellisé suffisant

Tableau 3.2 – Comparaison entre apprentissage supervisé et non supervisé

3.2.2 Étape 1 : Application de l'auto-encodeur (non supervisé)

Face à la richesse et à la variabilité du trafic réseau, il est essentiel d'extraire des représentations significatives à partir de données brutes, sans dépendre systématiquement des étiquettes.

Dans cette optique, nous avons intégré un auto-encodeur, un réseau de neurones non supervisé capable d'apprendre les caractéristiques du trafic normal en le reconstruisant, afin de mieux repérer les comportements anormaux.

Qu'est-ce qu'un auto-encodeur ?

Un auto-encodeur est un type de réseau de neurones non supervisé conçu pour apprendre une représentation simplifiée des données. Il est constitué de deux parties principales :

- **L'encodeur :**compresse les données d'entrée en une forme plus réduite appelée représentation latente ((bottleneck).
- **Le décodeur :**reconstruit les données originales à partir de cette représentation. L'objectif est que la sortie soit aussi proche que possible de l'entrée. La différence entre les deux, appelée erreur de reconstruction, permet de détecter les données inhabituelles ou anormales.

Pourquoi utiliser un auto-encodeur ?

L'intégration de l'auto-encodeur dans notre approche repose sur quatre objectifs principaux

- **Détection non supervisée des flux ambigus (inconnu) :**

  En étant entraîné uniquement sur des flux normaux, l'auto-encodeur apprend leur structure et permet ensuite d'identifier les flux anormaux (attaques ou background flou) via l'erreur de reconstruction, sans étiquette. Cela répond directement à la difficulté de traiter les flux « background » dans CTU-13, dont la nature exacte n'est pas toujours connue.

  L'erreur de reconstruction (calculée par la MSE) est un indicateur fiable de comportement anormal. Plus l'erreur est élevée, plus le flux est considéré comme suspect ou inconnu.

- **Réduction de la dimensionnalité :** :

  La représentation latente (composée de 10 neurones) extraite par l'auto-encodeur est une version simplifiée des données, qui conserve les informations essentielles. Cela facilite l'analyse tout en limitant la complexité des modèles.

- **Enrichissement du clustering :**

  L'erreur de reconstruction est exploitée comme nouvelle feature dans l'étape de clustering avec K-Means, ce qui aide à mieux regrouper les flux par comportement et à détecter plus finement les anomalies.

- **Renforcement de la classification supervisée (stacking) :**

  Les features générées par l'auto-encodeur à la fois l'erreur de reconstruction et les représentations latentes sont intégrées directement dans le jeu de données utilisé pour entraîner les classifieurs supervisés (Random Forest, XGBoost, LR). Elles enrichissent donc la base d'apprentissage du modèle final et contribuent à améliorer la précision du stacking.

Fonctionnement de l'auto-encodeur dans notre approche

Dans notre approche, nous avons utilisé un auto-encodeur dense, également appelé (undercomplete) auto-encoder. Il s'agit d'un réseau de neurones entièrement connecté, composé uniquement de couches denses.

Ce type d'auto-encodeur est dit simple car sa couche latente (bottleneck) est plus petite que l'entrée. Il est conçu pour forcer une compression d'information, ce qui rend les écarts de reconstruction visibles pour les flux anormaux.

— **Prétraitement des données**

- Les colonnes label et famille sont supprimées afin d'éviter toute fuite de données lors de l'apprentissage non supervisé.
- Ensuite, les variables numériques restantes sont normalisées à l'aide de la méthode MinMaxScaler, afin de ramener chaque valeur dans l'intervalle [0, 1].Cette étape est cruciale pour assurer une convergence stable et cohérente du modèle. Elle est de la forme :

$$Min\_Max = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

— **Entrainement de l'auto-encodeur sur le flux normal**

La première étape consiste à appliquer l'auto-encodeur uniquement sur le trafic normal, c'est-à-dire les instances pour lesquelles la colonne famille est égale à "normal".

Nous avons conçu le schéma ci-dessous afin d'illustrer la structure détaillée de l'auto-encodeur dense utilisé dans notre approche.

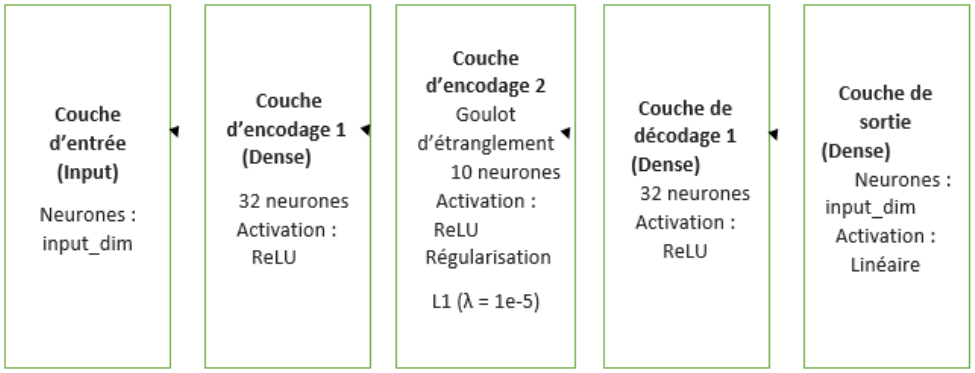

Figure 3.1 – Architecture des couches du auto-encodeur

- **Couche d'entrée :** nombre de neurones égal au nombre de features(10). Chaque neurone de cette couche correspond directement à une variable de trafic réseau . C'est à partir de ces données normalisées que le réseau commence son traitement.
- **Encodage 1 :** 32 neurones avec activation ReLU. Cette première transformation réduit légèrement la dimensionnalité et introduit une première abstraction du trafic. L'activation ReLU permet de ne conserver que les activations positives, apportant de la non-linéarité sans saturer les gradients.
- **couche latente (Bottleneck) :** 10 neurones avec activation ReLU + régularisation L1. Cette couche centrale joue un rôle critique : elle force une compression maximale de l'information utile. Grâce à la régularisation L1, seuls les neurones les plus pertinents sont activés (sparsity), ce qui aide à éliminer le bruit et à révéler les patterns fondamentaux du trafic normal.
- **Décodage 1 :** 32 neurones, activation ReLU. Cette couche commence la phase de reconstruction. Elle essaie d' expanser les données latentes vers un format approchant celui de l'entrée originale.
- **Sortie :** même nombre de neurones que l'entrée (38 ou 45), activation linéaire. Cette couche tente de reconstituer les données d'origine sans contrainte de bornes (grâce à l'activation linéaire), ce qui est essentiel pour évaluer précisément l'erreur de reconstruction.

— **Application du modèle sur l'ensemble des données**

Une fois l'auto-encodeur entraîné uniquement sur les données normales, il est appliqué à l'ensemble du dataset, comprenant :

- Le trafic normal (déjà vu par le modèle)
- Le trafic malveillant (attaques)
- Le trafic de fond (background) souvent ambigu

Pour chaque ligne du dataset, le modèle tente de reconstruire les valeurs originales à partir de la compression apprise. En comparant les valeurs d'entrée et de sortie, on calcule l'erreur de reconstruction pour chaque flux.

Cette erreur est déterminée par la formule suivante : MSE entre les valeurs d'entrée et de sortie. Elle traduit l'incapacité du modèle à comprendre certains flux. Ainsi :

- **Une faible erreur :** signifie que le flux est similaire aux données normales (comportement attendu).
- **Une erreur élevée :** indique un comportement atypique, potentiellement malveillant (botnet) ou ambigu (inconnu) .

Cette valeur d'erreur est ajoutée comme nouvelle colonne (reconstruction_error) dans le jeu de données, et devient une feature essentielle pour les étapes suivantes de clustering et de classification.

— **Paramètre d'apprentissage :**

- **Optimiseur :** Adam

  Est une version avancée de la descente de gradient stochastique. Il combine le principe de momentum et une adaptation dynamique du taux d'apprentissage pour chaque paramètre, ce qui améliore la convergence, en particulier sur les jeux complexes comme le trafic réseau.

- **Epochs :** 10

  Une epoch est un passage complet sur l'ensemble des données. En choisissant 10 epochs, on équilibre entre sous-apprentissage et surapprentissage. Plus d'epochs peuvent affiner le modèle mais risquent aussi de l'adapter trop aux données d'entraînement.

- **Batch size :** 256

  Le batch size désigne combien d'exemples sont utilisés simultanément avant de mettre à jour les poids du modèle. Un batch de 256 permet de lisser les

gradients, d'accélérer l'apprentissage tout en restant raisonnable pour les capacités mémoire.

- **Validation split :** 0.2

  Cela signifie que 20 % du trafic normal est conservé pour valider le modèle après chaque epoch. Cela permet de détecter un surapprentissage si la performance sur les données de validation commence à se dégrader alors que celle sur l'entraînement augmente.

- **Fonction de perte :** Mean Squared Error (MSE)

  La MSE (Erreur quadratique moyenne) mesure la moyenne des carrés des différences entre les valeurs d'entrée et les valeurs reconstruites par l'autoencodeur. Elle est définie comme :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2 \tag{3.2}$$

  Avec :

  - x_i : valeur réelle de la i-ème feature
  - $\hat{x_i}$: valeur reconstruite par le modèle
  - n : nombre total de features

3.2.3 Étape 02 : regroupement des flux avec K-Means (non supervisé)

Après avoir extrait des informations pertinentes à l'aide de l'auto-encodeur en particulier l'erreur de reconstruction et les représentations latentes compressées une étape de clustering non supervisé est appliquée à l'ensemble du dataset. L'algorithme utilisé est K-Means, choisi pour sa simplicité et son efficacité à regrouper les données selon des similarités structurelles.

Qu'est-ce que K-Means ?

K-Means est un algorithme de regroupement basé sur la distance. Il fonctionne en divisant les données en K clusters, en minimisant la distance moyenne entre chaque

point et le centre de son groupe. De manière itérative, l'algorithme ajuste les centres de clusters (centroïdes) jusqu'à ce que les regroupements soient stables.

Chaque flux est donc affecté à un groupe, en fonction de ses caractéristiques numériques.

Pourquoi intégrer un clustering dans notre approche ?

Dans notre cas, l'intérêt de K-Means est multiple :

- Il permet d'organiser les flux sans supervision, en exploitant directement les données issues de l'auto-encodeur ;
- Il est particulièrement utile pour traiter les flux (background) du dataset CTU-13, qui ne sont pas toujours bien définis ni clairement étiquetés ;
- Il offre un moyen de distinguer automatiquement des groupes de comportements similaires, même si leur nature (normale, botnet, ambiguë) n'est pas connue a priori.

En d'autres termes, K-Means agit ici comme un outil de structuration complémentaire, qui enrichit encore la compréhension du dataset avant la phase supervisée.

Application concrète de K-Means dans notre approche

— **Isolement du trafic background**

On isole les lignes du dataset dont la colonne famille est étiquetée comme "background". Ces flux n'ont pas été utilisés pour entraîner l'auto-encodeur, mais peuvent contenir des informations déterminantes.

— **Utilisation de l'erreur de reconstruction**

Chaque flux est associé à une erreur de reconstruction issue de l'auto-encodeur :

- Si l'erreur est faible, le flux est proche du trafic normal (bénin).
- Si l'erreur est élevée, le flux est étrange voire potentiellement malveillant.

Cette métrique permet une mesure fine de la normalité sans supervision.

— **Clustering par K-Means**

L'algorithme K-Means est utilisé pour regrouper les flux background en 3 clusters distincts sur la base de leur erreur de reconstruction et de leurs représentations compressées.

Il est appliqué par rapport à son efficacité pour des données numériques continues, en particulier quand on combine une mesure discriminante (erreur) avec des vecteurs compacts issus de l'apprentissage profond.

- **Distance utilisée :** K-Means utilise la distance euclidienne multidimensionnelle pour regrouper les points.

Dans notre contexte, cette distance est calculée comme suit :

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.3}$$

Elle est appliquée à des vecteurs constitués de :

➤ l'erreur de reconstruction (scalaire)

➤ les valeurs issues de la couche bottleneck (vecteur compressé)

Cette approche permet de regrouper ensemble les flux présentant une proximité à la fois en termes de comportement reconstruit et de structure latente.

— **Interprétation comportementale des clusters**

Pour chaque cluster, on calcule la moyenne de l'erreur de reconstruction afin de valider son interprétation. Mais contrairement à une analyse basée uniquement sur cette erreur, ici l'appartenance au cluster dépend également des features compressées (vecteurs latents). Cette combinaison permet de classer en 3 cluster :

- Le cluster dont les points ont à la fois une erreur faible et des représentations proches du normal → "background_normal"
- Le cluster avec les représentations les plus éloignées du comportement normal, couplé à une forte erreur → "background_botnet"
- Le cluster intermédiaire, avec des représentations atypiques et une erreur modérée → "background_inconnu"

Ce processus permet de traduire les groupes numériques en catégories comportementales interprétables, en s'appuyant à la fois sur les reconstructions et sur les patterns latents extraits automatiquement.

— **Ajout de la feature sub_famille**

On ajoute une nouvelle colonne sub_famille aux lignes concernées, permettant de distinguer :

- “background_normal”
- “background_botnet”
- “background_inconnu”

Cette nouvelle information affine la segmentation du dataset.

— **Intégration dans le dataset principal**

Les valeurs de **famille** et **sub_famille** sont mises à jour dans le dataset principal pour les lignes de type background. Cela permet :

- D’exclure “background_inconnu” et “background_botnet” de l’entraînement supervisé,
- De fournir une base plus nette pour l’apprentissage de classifieurs supervisés.

— **Visualisation et validation**

Une analyse graphique montre une séparation nette entre les trois sous-familles, non seulement selon leur erreur de reconstruction, mais aussi dans l’espace défini par les features compressées. Cette validation visuelle confirme la cohérence du regroupement effectué par K-Means sur la base des informations issues de l’auto-encodeur.

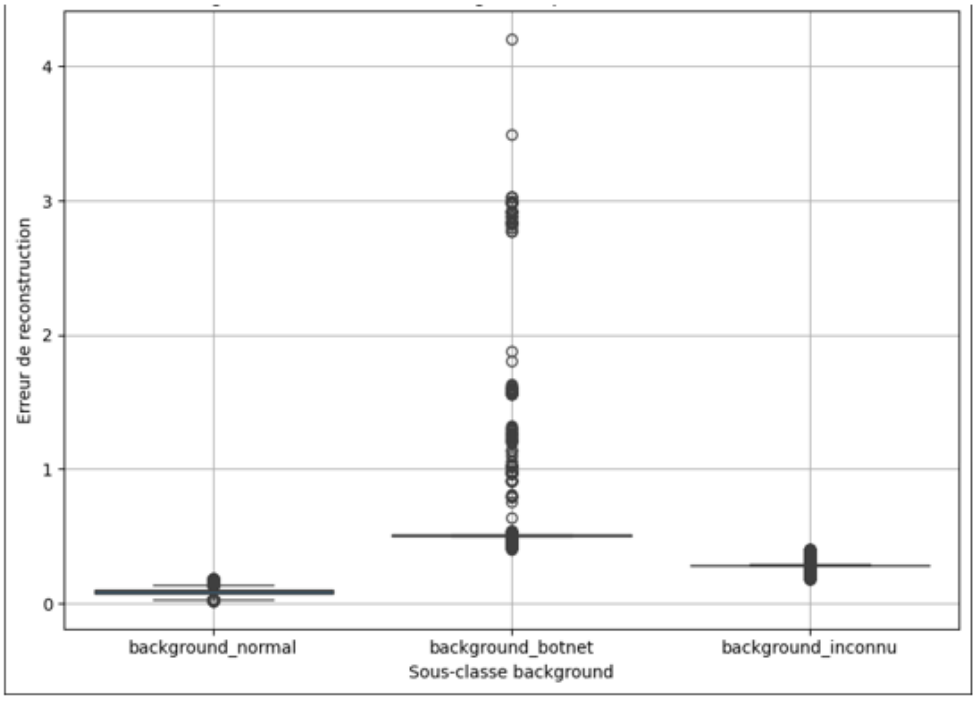

Figure 3.2 – Séparation nette entre les trois sous-familles de background

Distribution observée après clustering :

- "background_normal" : 118 92 flux
- "background_inconnu" : 53 63 flux
- "background_botnet" : 18 55 flux

3.2.4 Étape 03 : classification par stacking (supervisé)

Définition de Stacking

Le stacking (stacked generalization) est une technique d'apprentissage ensembliste qui consiste à combiner plusieurs modèles différents afin d'améliorer la performance globale de la classification.

Au lieu de s'appuyer sur un seul algorithme, le stacking utilise une architecture à deux niveaux :

- **Des modèles de base (base learners) :** comme Random Forest, XGBoost ou régression logistique, qui apprennent chacun sur les mêmes données enrichies ;
- **Un modèle final (meta-learner) :** apprend à pondérer et combiner les prédictions des modèles de base afin de produire une décision plus robuste.

L'objectif est de tirer parti des points forts de chaque modèle tout en compensant leurs faiblesses respectives. En combinant leurs sorties, le stacking génère généralement des prédictions plus précises, plus équilibrées et offre de meilleures capacités de généralisation.

Pourquoi utiliser Random Forest et XGBoost dans notre approche ?

Dans notre approche, les modèles Random Forest (RF) et XGBoost (XGB) sont utilisés pour la classification supervisée (botnet vs normal), car :

- **RF** est robuste, facile à interpréter, peu sensible au bruit et efficace même avec des variables corrélées. Il capture très bien les structures globales.
- **XGBoost** est un modèle de boosting particulièrement performant, souvent capable de capturer des interactions complexes et des patterns subtils dans les données.

En les combinant via un modèle de stacking supervisé avec une régression logistique, on tire profit de leurs complémentarités. Ce méta-apprentissage permet de :

- Pondérer intelligemment leurs prédictions,
- Réduire les faux négatifs (attaques ratées) et faux positifs,
- Améliorer la robustesse globale sur les cas ambigus ou partiellement étiquetés (cas du trafic background).

Ainsi, RF et XGB s'appuient sur les données enrichies par l'auto-encodeur (features compressées + reconstruction error+Cluster) et renforcent les performances finales du classifieur binaire.

Pourquoi utiliser le stacking et comment fonctionne-t-il ?

Nous avons établi le schéma suivant pour illustrer le fonctionnement de la méthode de stacking mise en œuvre dans notre approche hybride. Ce diagramme met en évidence les différentes étapes du processus, depuis la préparation des données enrichies jusqu'à la combinaison des prédictions des modèles de base au sein du méta-modèle.

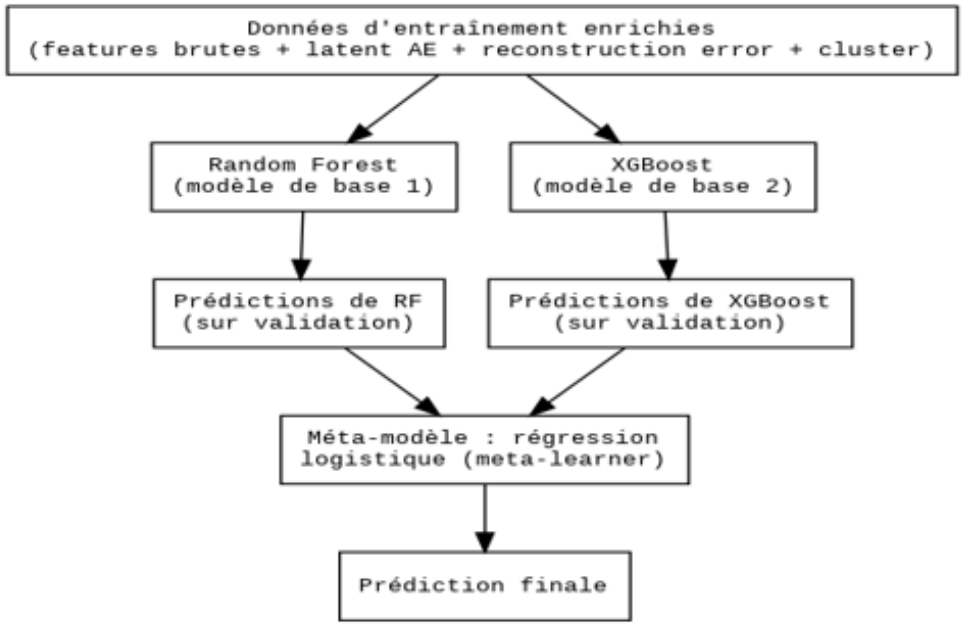

Figure 3.3 – L'architecture de la méthode Stacking dans notre approche hybride.

Dans notre approche, nous avons utilisé la technique du stacking afin de combiner plusieurs classifieurs et ainsi renforcer la qualité des prédictions.

— **Données d'entrée :** Nous avons enrichi les données d'origine avec :

- Les features brutes du trafic réseau ;
- Les représentations latentes issues de l'auto-encodeur (couche comprimée) ;
- L'erreur de reconstruction, calculée par l'auto-encodeur (MSE) ;
- Le cluster K-Means, indiquant l'appartenance comportementale de chaque flux ;

Une feature supplémentaire appelée (famille binaire), conçue pour faciliter la classification binaire (botnet vs normal).

Cette dernière feature synthétise les informations issues de toutes les étapes précédentes en ignorant juste le bakground inconnu et garder le background normal et botnet. Elle permet de concentrer l'apprentissage des modèles sur la détection des flux associés aux attaques de type botnet.

— **Modèles de base (base learners) :** Nous avons ensuite entraîné en parallèle deux modèles supervisés :

- Un Random Forest.
- Un XGBoost.

Ces deux modèles génèrent des prédictions (probabilités de classe) à partir de la même base enrichie.

— **Meta-learner :** Les prédictions des deux modèles de base (sur les données de validation) sont ensuite utilisées comme nouvelles entrées pour un modèle de plus haut niveau :

- Une régression logistique.

— **Prédiction finale :** Le meta-learner apprend à pondérer les prédictions des deux modèles de base et produit la prédiction finale, qui bénéficie de la complémentarité entre Random Forest et XGBoost.

Avantages du Random Forest dans le stacking

- Diversité de Random Forest, en tant que modèle "ensembliste", apporte une approche différente par rapport à XGBoost (boosting).
- Modèle stable et peu sensible au bruit, ce qui permet d'apporter des prédictions robustes au stacking.
- Fournit des prédictions de probabilité cohérentes, utiles pour le méta-learner.

- Capacité à gérer efficacement les features enrichies (erreur de reconstruction, représentations latentes, cluster, famille binaire).
- Apporte une vision complémentaire par rapport à XGBoost (basé sur boosting), ce qui améliore la capacité du stacking à capturer des comportements variés.

Avantage du XGBoost dans le stacking

- Excellente performance sur des datasets complexes et bruités, comme CTU-13 [110].
- Capacité à gérer les interactions non linéaires entre les features (surtout avec les features enrichies).
- Très bon contrôle du sur-apprentissage grâce à la régularisation intégrée (L1/L2).
- Modèle complémentaire de Random Forest : XGBoost est basé sur le boosting (séquentiel), alors que RF est un bagging (parallèle).
- Fournit des prédictions probabilistes précises, utiles pour le méta-learner.
- Apporte au stacking une capacité fine à capturer des patterns rares ou subtils dans les données.

Avantage du régression logistique dans le stacking

- Modèle simple et rapide à entraîner.
- Très efficace pour pondérer les prédictions des modèles de base.
- Moins sujet au sur-apprentissage que des modèles plus complexes.
- Fournit une sortie probabiliste claire pour la classification binaire (botnet vs normal).
- Permet une interprétation facile des poids associés à chaque modèle de base.
- S'adapte bien aux prédictions hétérogènes venant de Random Forest et XGBoost.

Avantage d'utiliser le stacking dans notre approche

- Pour combiner les forces de plusieurs modèles (Random Forest et XGBoost) et compenser leurs faiblesses respectives ;

- Pour obtenir une meilleure capacité de généralisation face aux données bruitées et hétérogènes (CTU-13) ;
- Pour permettre au méta-learner (régression logistique) d'ajuster finement la pondération entre les modèles de base ;
- Pour bénéficier de la complémentarité entre Random Forest (bagging) et XGBoost (boosting), deux algorithmes très différents ;
- Pour produire des prédictions plus robustes, notamment pour la détection binaire (botnet vs normal) ;
- Pour améliorer les performances globales par rapport à l'utilisation d'un modèle seul.

Ainsi, l'utilisation de cette stratégie de stacking nous a permis de tirer parti des différentes forces des modèles de base et de renforcer la capacité globale de détection. En combinant les prédictions de Random Forest et XGBoost au travers d'une régression logistique, nous obtenons un modèle final plus robuste, capable de mieux généraliser face à la diversité des comportements présents dans le trafic réseau.

Dans le cadre de ce travail, nous avons conçu une approche hybride associant des techniques non supervisées (auto-encodeur et clustering K-Means) à des modèles supervisés performants (Random Forest, XGBoost). Pour plus de clarté et afin d'assurer la cohérence de la rédaction, cette approche sera désignée par l'appellation « **approche KRAST** » tout au long de ce mémoire.

3.2.5 Présentation de notre contribution KRAST

Les étapes de l'approche KRAST

— **Importation des bibliothèques**

- **pandas, numpy, os, time :** utilisées pour la manipulation et le traitement des données.
- **matplotlib, seaborn :** utilisées pour la visualisation des données et des résultats.
- **scikit-learn (sklearn) :** utilisée pour le prétraitement des données et la mise en œuvre des modèles de machine learning.

- **tensorflow.keras :** utilisée pour la conception et l'entraînement des réseaux de neurones (auto-encodeur).
- **XGBoost et stacking :** utilisés pour la construction des modèles de base et du méta-modèle de stacking.

— **Prétraitement des données**

- Charger le dataset brut dans un DataFrame.
- Réduire l'empreinte mémoire via conversion optimisée des types de colonnes.
- Créer une nouvelle colonne "**famille**" en regroupant les étiquettes en : normal, botnet, background.
- Reformer les noms de colonnes (espaces, majuscules).
- Supprimer les colonnes non pertinentes (starttime, srcaddr, sport, dstaddr, dport).
- Supprimer les lignes contenant des valeurs manquantes.
- Supprimer les doublons pour éviter la redondance.
- Encoder numériquement les colonnes catégorielles (proto, dir, state) via LabelEncoder.
- Standardiser les variables numériques via Z-score (centrage-réduction).
- Séparer le dataset en trois sous-ensembles selon la colonne famille : normal, botnet, background.
- Rééquilibrer les classes par sous-échantillonnage aléatoire selon la plus petite classe.
- Créer une colonne famille _binaire : 1 pour botnet et background suspect, 0 pour normal.
- Supprimer les colonnes label, famille, sub _famille pour éviter le surapprentissage.
- Séparer les variables explicatives X et la cible y = famille _binaire.
- Diviser X et y en jeux d'entraînement et de test (70/30, stratifié).

— **Apprentissage non supervisé via Auto-encodeur**

- Entraîner un auto-encodeur uniquement sur les flux de trafic normal.

- Construire l'architecture : couche d'entrée, couches cachées, couche latente compressée, décodage.
- Optimiser avec l'algorithme Adam et la fonction de perte MSE.
- Appliquer l'auto-encodeur entraîné sur tout le dataset.
- Calculer l'erreur de reconstruction (MSE) pour chaque ligne du dataset.
- Extraire les représentations compressées (vecteurs latents) du bottleneck pour chaque échantillon.
- Ajouter deux nouvelles familles de features à X : reconstruction _error et features compressés.

— **Clustering des flux background via KMeans**

- Isoler les flux ayant famille = background.
- Appliquer KMeans (k=3) sur les données enrichies (reconstruction _error + vecteurs latents).
- Identifier les sous-types : background _normal, background _botnet, background _inconnu.
- Créer une nouvelle colonne sub _famille avec ces sous-catégories.
- Mettre à jour les colonnes famille et famille _binaire selon les résultats du clustering.

— **Classification supervisée par Stacking**

- Fusionner les features d'entrée : données d'origine, features compressées, reconstruction error, labels mis à jour.
- Réinitialiser la variable X avec l'ensemble enrichi.
- Séparer à nouveau les données X et la cible famille _binaire.
- Diviser à nouveau en train et test (70/30, stratifié).
- Définir les modèles de base : Random Forest et XGBoost.
- Définir le modèle méta-apprenant : Régression logistique.
- Construire un StackingClassifier avec RF, XGB comme base learners, LR comme meta learner.
- Entraîner le modèle de stacking sur l'ensemble X _train.

— **Évaluation du modèle**

- Évaluer le modèle sur X _test à l'aide des métriques : accuracy, precision, recall, F1-score, AUC, matrice de confusion, courbe ROC, rapport de classification.
- calculer le temps de détection des botnets.

Présentation de l'approche KRAST

Notre méthode, baptisée KRAST , a été conçue pour la détection de botnets dans le trafic réseau général, en particulier sur des données massives, bruitées et souvent partiellement étiquetées comme celles issues du dataset CTU-13 [73].

Fondement de l'approche

Le dataset CTU-13 [110] présente un trafic réseau hétérogène où se mêlent du trafic normal, des flux de fond (background) et des communications malveillantes issues de botnets actifs.

Dans ce contexte, les approches supervisées classiques sont vite limitées par l'incomplétude des étiquettes et la complexité du trafic. C'est pourquoi KRAST combine des techniques non supervisées pour détecter les anomalies sans avoir besoin de labels, et des classificateurs supervisés puissants pour affiner la détection dans une perspective binaire (attaque vs non-attaque).

Architecture de KRAST

L'architecture KRAST repose sur cinq composants complémentaires :

K : K-Means

- Pour segmenter le trafic background, souvent non étiqueté, en sous-comportements latents (background normal, suspect, inconnu). Cela permet d'exploiter une portion importante du dataset sans dépendre des labels.

R : Random Forest

- Pour effectuer une classification dans le stacking robuste et performante sur le trafic explicitement étiqueté. Random Forest est particulièrement adapté aux données bruitées du CTU-13.

A : Auto-Encoder

- Entraîné sur le trafic normal, il permet de calculer l'erreur de reconstruction, utilisée ensuite pour détecter des flux anormaux parmi les autres types de trafic.

S : Stacking

- Combine plusieurs modèles de classification (Random Forest, XGBoost) à l'aide d'un méta-modèle de type régression logistique, pour obtenir une décision plus fiable.

T : Traficx

- KRAST se base entièrement sur l'analyse comportementale du trafic réseau, en exploitant à la fois sa structure (via clustering et auto-encodeur) et son étiquetage partiel (via classification supervisée).

Originalité et avantages

- Détection non supervisée des anomalies via l'auto-encodeur, sans dépendre des étiquettes.
- Clustering du trafic background, souvent ignoré par les approches classiques.
- Fusion des connaissances extraites par auto-encodeur (features compressées, erreur de reconstruction et clustering).
- Modèle robuste et explicable via Random Forest.
- Amélioration des performances via stacking adaptatif (Random Forest + XGBoost + Logistic Regression).

- Approche modulaire, scalable et généralisable à d'autres types de trafic réseau, même mal étiquetés.

KRAST : une réponse hybride aux limites du dataset CTU-13

L'approche KRAST propose une stratégie hybride originale, capable de traiter les défis spécifiques du dataset CTU-13 :

- Trafic réel et bruité,
- Présence importante de flux non étiquetés (background),
- Hétérogénéité des attaques,
- Déséquilibre des classes.

En combinant des techniques non supervisées (auto-encodeur et clustering K-Means) à des modèles supervisés puissants (Random Forest, XGBoost) intégrés dans un mécanisme de stacking adaptatif, KRAST permet une détection binaire précise (attaque vs non-attaque), même dans des environnements partiellement observés et peu balisés.

KRAST est donc une approche robuste, flexible et applicable à grande échelle, particulièrement adaptée à des datasets réels comme CTU-13, où les méthodes classiques sont souvent inefficaces.

3.2.6 Prétraitement des données de l'architecture globale

Voici chaque étape de prétraitement présentée dans notre approche :

- **Réduction mémoire des données :** L'une des premières étapes consiste à réduire l'empreinte mémoire du dataset afin de faciliter son traitement, notamment lors de l'entraînement de modèles complexes comme les auto-encodeurs ou les classificateurs en ensemble. Cette réduction avec la fonction df-shrink passe par la conversion des colonnes de type object en category, ce qui transforme les chaînes de caractères répétitives en identifiants numériques économes en mémoire. De plus, les entiers (int) sont convertis en uint32 ou int32 selon leur signe, et les flottants (float64) en float32, ce qui permet de conserver la précision nécessaire sans encombrer inutilement la mémoire. Cette étape repose sur une règle simple d'optimisation des types : ne conserver que la précision suffisante pour représenter les données de manière fiable.

- **Regroupement des étiquettes en familles comportementales :** Le dataset initial contient une colonne label comportant des étiquettes très variées, souvent spécifiques à des attaques précises ou des artefacts de capture. Pour simplifier l'analyse, une nouvelle colonne famille est créée en classant les flux en trois grandes catégories : normal, botnet et background. Cette classification repose sur une analyse textuelle des étiquettes : si le nom du label contient des termes caractéristiques de botnets connus (neris, rbot, sogou, virut, menti, murlo, nsis), il est classé comme botnet. Les labels contenant le mot "background" sont assignés à cette classe spécifique, tandis que le reste est considéré comme normal. Cette simplification permet une meilleure structuration du problème et facilite les approches supervisées ou non supervisées.
- **Nettoyage syntaxique et sélection des colonnes pertinentes :** Afin d'éviter les erreurs liées à la casse ou aux espaces, les noms de colonnes sont systématiquement convertis en minuscules et nettoyés. Ensuite, plusieurs colonnes sont supprimées : starttime, srcaddr, sport, dstaddr, dport, car elles représentent des informations trop spécifiques au contexte de capture ou susceptibles d'introduire un surapprentissage (overfitting). Leur suppression est justifiée par la volonté de construire un modèle généralisable, et non dépendant de l'infrastructure réseau utilisée dans les captures.
- **Traitement des valeurs manquantes et des doublons :** Il est indispensable de s'assurer que le dataset soit propre et cohérent avant d'initier l'apprentissage. Les lignes contenant des valeurs manquantes sont supprimées afin d'éviter toute instabilité dans les algorithmes. De même, les doublons sont éliminés pour éviter que certaines observations ne soient surreprésentées, ce qui pourrait biaiser le modèle et réduire sa capacité de généralisation.
- **Encodage des variables catégorielles :** Certaines colonnes comme proto (protocole), dir (direction) ou state (état de la connexion) contiennent des données non numériques. Pour permettre leur exploitation par les algorithmes d'apprentissage, elles sont encodées numériquement à l'aide d'un label encoder. Ce dernier attribue un identifiant unique à chaque modalité. Cette méthode est adaptée ici car le nombre de modalités est raisonnable et l'objectif est simplement de transformer les données en valeurs numériques sans introduire de hiérarchie entre les catégories.

- **Standardisation des données numériques :** Une fois les variables numériques extraites, elles sont standardisées à l'aide de la méthode Z-score. Cela signifie qu'on applique à chaque colonne une transformation appelée centrage-réduction, qui consiste à soustraire la moyenne (μ) et à diviser par l'écart-type (σ) :

$$z = \frac{x - \mu}{\sigma} \tag{3.4}$$

  Cette standardisation Z-score permet de ramener toutes les variables sur une échelle comparable, avec une moyenne nulle et un écart-type égal à un. Elle est particulièrement cruciale pour les algorithmes sensibles à l'échelle des données, comme les réseaux de neurones et les méthodes de clustering fondées sur des distances, notamment K-Means, où des variables non normalisées pourraient dominer le calcul des distances.

- **Séparation des données par comportement réseau** Le dataset est ensuite scindé en trois sous-ensembles en fonction de la colonne famille : le trafic normal, le trafic background, et le trafic botnet. Cette séparation permet d'appliquer des traitements spécifiques à chaque type, comme l'entraînement d'un auto-encodeur uniquement sur le trafic normal, ou la détection d'anomalies dans le trafic background. C'est une étape essentielle dans les approches semi-supervisées ou hybrides, où chaque type de trafic est traité de manière distincte.

- **Rééquilibrage du dataset par sous-échantillonnage :** Les classes dans les données ne sont généralement pas équilibrées : certaines, comme le trafic normal ou le background, sont souvent surreprésentées par rapport aux attaques. Pour éviter un biais d'apprentissage, un sous-échantillonnage est effectué de manière à obtenir un nombre égal d'échantillons pour chaque classe. On détermine la taille minimale parmi les trois classes, puis on en extrait un échantillon aléatoire sans remise de cette taille pour chaque groupe. Le dataset équilibré obtenu garantit une performance équitable du modèle, en particulier en termes de rappel et de F1-score.

- **Création d'un label binaire pour la classification supervisée :** Dans une optique de détection de botnet, le problème est reformulé en classification binaire. Une nouvelle colonne famille _binaire est créée, regroupant les étiquettes botnet (ou background suspecté) sous la classe 1 (attaque), et les autres (normal) sous la classe 0 (non-attaque). Cette simplification est cruciale pour les algorithmes

de détection d'anomalies et les classificateurs binaires comme Random Forest, XGBoost.

- **Séparation des variables explicatives et de la cible :** Pour l'entraînement supervisé, les variables d'entrée (features) sont séparées de la variable cible (le label). La cible, ici famille _binaire, est encodée numériquement en 0 ou 1. Les colonnes non informatives ou redondantes (label, famille, sub _famille) sont retirées des variables explicatives afin d'éviter les fuites d'information (data leakage), ce qui rend l'apprentissage plus rigoureux et réaliste.
- **Division du dataset en jeu d'entraînement et de test :** Enfin, le dataset est divisé en deux ensembles : un pour l'entraînement (70 %) et un pour le test (30 %). Cette séparation est stratifiée selon la variable cible, ce qui garantit que la proportion des classes (attaque vs. non-attaque) reste la même dans les deux sous-ensembles. Cela permet une évaluation fidèle et équitable du modèle sur des données jamais vues pendant l'apprentissage.

3.2.7 Implémentation technique

Langage : Python

Python est un langage de programmation interprété, reconnu pour sa simplicité, sa lisibilité et sa polyvalence. Il est largement utilisé dans des domaines variés tels que la science des données, l'intelligence artificielle, le développement web ou encore l'automatisation, grâce à une syntaxe claire et un vaste écosystème de bibliothèques performantes.

Environnement :

Nous avons utilisé Google Colab comme environnement de développement,

- **Google Colab : Google Colab** est une plateforme de développement en ligne largement utilisée dans le domaine de la machine learning et de la science des données. Elle repose sur les notebooks Jupyter, ce qui permet d'écrire et d'exécuter du code Python de manière interactive, directement dans le navigateur. L'un de ses grands avantages est la possibilité d'accéder gratuitement à des GPU et TPU (Tensor Processing Units), ce qui est particulièrement utile pour l'entraînement

de modèles complexes. Dans le cadre de notre approche, l'exécution a été réalisée en tirant parti du TPU pour accélérer les phases de calcul intensif. Colab s'intègre aussi très bien avec l'écosystème Google, notamment Google Drive, ce qui facilite la gestion, le stockage et le partage des données. Grâce à sa simplicité d'utilisation et à ses ressources intégrées, Google Colab représente un excellent choix pour expérimenter, tester et développer des solutions basées sur l'intelligence artificielle ou l'analyse de données [111].

Librairies :

Voici les principales librairies utilisées

- **pandas :** utilisée pour la manipulation et l'analyse des données tabulaires sous forme de DataFrame, ce qui facilite le nettoyage et la structuration des données.
- **numpy :** sert à effectuer des opérations mathématiques et des manipulations efficaces sur des tableaux multidimensionnels.
- **os :** permet d'interagir avec le système de fichiers, notamment pour accéder, lire ou écrire des fichiers et dossiers.
- **time :** utilisée pour mesurer le temps d'exécution de certaines parties du code ou pour des opérations de temporisation.
- **sklearn.model _selection.train _test _split :** divise les données en ensembles d'entraînement et de test pour l'évaluation des modèles.
- **sklearn.preprocessing.StandardScaler :** normalise les données en centrant et réduisant les features pour qu'elles aient une moyenne nulle et un écart-type de 1.
- **sklearn.preprocessing.LabelEncoder :** convertit les étiquettes catégorielles en valeurs numériques afin qu'elles soient compatibles avec les algorithmes de machine learning.
- **sklearn.ensemble.RandomForestClassifier :** implémente un classifieur basé sur un ensemble d'arbres de décision, utilisé ici pour détecter les attaques botnet.
- **sklearn.metrics (classification _report, confusion _matrix, ConfusionMatrixDisplay, roc _auc _score, roc _curve) :** fournit des outils pour évaluer les performances des modèles de classification.
- **sklearn.linear _model.LogisticRegression :** fournit une méthode de classification linéaire utilisée comme modèle final dans le stacking.

- **sklearn.cluster.KMeans :** applique une méthode de clustering non supervisée pour segmenter les données (par exemple, le trafic background).
- **sklearn.decomposition.PCA :** effectue une réduction de dimension pour visualiser ou simplifier les données tout en conservant l'essentiel de la variance.
- **xgboost.XGBClassifier :** propose un classifieur par gradient boosting très performant, souvent utilisé pour capturer des relations complexes entre les variables.
- **sklearn.ensemble.StackingClassifier :** permet de combiner plusieurs modèles (ici RF et XGBoost) via un méta-apprentissage afin d'optimiser la précision globale.
- **matplotlib.pyplot :** utilisée pour créer des graphiques et visualisations, notamment pour explorer les données et valider les clusters ou les performances.
- **seaborn :** bibliothèque de visualisation basée sur Matplotlib, offrant des graphiques statistiques plus élégants et plus faciles à interpréter.
- **tensorflow et tensorflow.keras :** utilisés pour construire, entraîner et manipuler des modèles de deep learning, notamment l'auto-encodeur dans l'approche KRAST.
- **tensorflow.keras.models.Model, Input, Dense, regularizers :** composants spécifiques de Keras utilisés pour construire l'architecture de l'auto-encodeur, incluant les couches d'entrée, cachées, de sortie et la régularisation pour éviter surapprentissage.

L'ensemble de données CTU-13 :

Le dataset CTU-13 [73] est une référence majeure dans la recherche en cybersécurité, notamment dans le domaine de la détection de botnets. Il a été construit par le laboratoire Stratosphere de l'Université Technique Tchèque de Prague (CTU) pour offrir une base réaliste, complète et ouverte à la communauté scientifique. Contrairement à de simples simulations, ce dataset capture des scénarios réels de compromission, intégrant à la fois du trafic normal, du trafic malveillant (botnet), et du trafic de fond difficilement classifiable, appelé background.

— **Structure et contenu du dataset :**

- Le dataset CTU-13 se compose de 13 fichiers CSV, chacun correspondant à un scénario réseau unique capturant du trafic en conditions réelles. Chaque

scénario contient un mélange de flux légitimes, flux malveillants et de trafic en arrière-plan, reflétant ainsi la complexité d'un environnement réseau réel.

- **Trafic normal :**

  Il s'agit des communications générées par des utilisateurs ou des services authentiques comme la navigation web, les échanges de courriels, ou les requêtes DNS. Ce trafic représente le comportement attendu d'un réseau sain.

- **Trafic botnet :**

  Ce type de trafic est émis par des machines infectées par des botnets. Ces machines peuvent réaliser divers comportements malveillants : balayage de ports, envoi de spam, propagation de logiciels malveillants ou communication avec des serveurs de commande et de contrôle (C &C).

- **Trafic background :**

  C'est la partie la plus ambiguë du dataset. Le background ne désigne pas un type de trafic précis, mais plutôt tout ce qui n'a pas été formellement étiqueté comme normal ou botnet Il s'agit d'un mélange hétérogène de données, difficile à interpréter mais pourtant crucial à analyser.

- **Trafic background :**

  C'est la partie la plus ambiguë du dataset. Le background ne désigne pas un type de trafic précis, mais plutôt tout ce qui n'a pas été formellement étiqueté comme normal ou botnet Il s'agit d'un mélange hétérogène de données, difficile à interpréter mais pourtant crucial à analyser.

— **Importance de la prise en compte du background**

Dans la littérature, nous avons constaté que de nombreuses approches choisissent souvent de simplifier le traitement de la classe background, en éliminant ou en la négligeant lors de la phase d'apprentissage, afin de faciliter la classification. Or, le trafic background peut inclure à la fois des flux normaux non étiquetés, des attaques de type botnet en phase latente, ou encore des données parasites peu informatives. Dans notre approche KRAST, nous avons fait le choix de traiter cette classe avec le même niveau d'attention que les classes « normal » et « botnet », au lieu de l'ignorer. Ce défi est au cœur de notre problématique : l'auto-encodeur et le clustering K-Means jouent ici un rôle clé, en nous permettant d'analyser en profondeur ce trafic ambigu, de révéler des schémas cachés et de mieux différencier les comportements légitimes des activités malveillantes.

— **Scénarios du dataset CTU-13 et types de botnets impliqués**

Scénario	Fichier CSV	Type de Botnet impliqué
1	1-Neris-20110810.binetflow.csv	Neris
2	2-Neris-20110811.binetflow.csv	Neris
3	3-Rbot-20110812.binetflow.csv	Rbot
4	4-Rbot-20110815.binetflow.csv	Rbot
5	5-Virut-20110815-2.binetflow.csv	Virut
6	6-Menti-20110816.binetflow.csv	Menti
7	7-Sogou-20110816-2.binetflow.csv	Sogou
8	8-Murlo-20110816-3.binetflow.csv	Murlo
9	9-Neris-20110817.binetflow.csv	Neris
10	10-Rbot-20110818.binetflow.csv	Rbot
11	11-Rbot-20110818-2.binetflow.csv	Rbot
12	12-NsisAy-20110819.binetflow.csv	NsisAy
13	13-Virut-20110815-3.binetflow.csv	Virut

Tableau 3.3 – Scénarios et type des botnets du dataset CTU-13.

Comme montré au tableau(Tableau 3.3), le dataset CTU-13 offre une diversité riche de cas d'usage, à travers 7 familles de botnets (Neris, Rbot, Sogou, Virut, Menti, Murlo, NsisAy), ce qui en fait une référence incontournable pour tester des approches robustes de détection, comme notre stratégie KRAST.

— **Les attributs du dataset CTU-13**

Le dataset CTU-13 contient 15 attributs (features) principaux décrivant les flux réseau capturés, présentés dans le tableau suivant (Tableau 3.4).

N°	Colonne	Type	Type optimisé pour la mémoire	Description synthétique
0	StartTime	object	Temporel	Date et heure de début de la session réseau
1	Dur	float64	Numérique réel	Durée de la session (en secondes)
2	Proto	object	Catégoriel	Protocole utilisé (TCP, UDP, ICMP...)
3	SrcAddr	object	Adresse IP	Adresse IP source
4	Sport	object	Port	Port source de la communication
5	Dir	object	Catégoriel	Direction du flux (->, <-, etc.)
6	DstAddr	object	Adresse IP	Adresse IP de destination
7	Dport	object	Port	Port de destination
8	State	object	Catégoriel	État de la connexion (ex : SF, S0, REJ)
9	sTos	float64	Numérique réel	Type of Service (ToS) du paquet envoyé
10	dTos	float64	Numérique réel	Type of Service (ToS) du paquet reçu
11	TotPkts	int64	Numérique entier	Nombre total de paquets échangés
12	TotBytes	int64	Numérique entier	Nombre total d'octets échangés
13	SrcBytes	int64	Numérique entier	Nombre d'octets envoyés par la source
14	Label	object	Catégoriel	Étiquette brute issue du fichier (botnet spécifique, normal, background)

Tableau 3.4 – Les attributs de CTU-13.

— **Répartition des types de trafic**

Après regroupement des étiquettes en trois grandes catégories (Normal, Botnet, Background), on observe un déséquilibre très fort, comme le montre la répartition suivante :

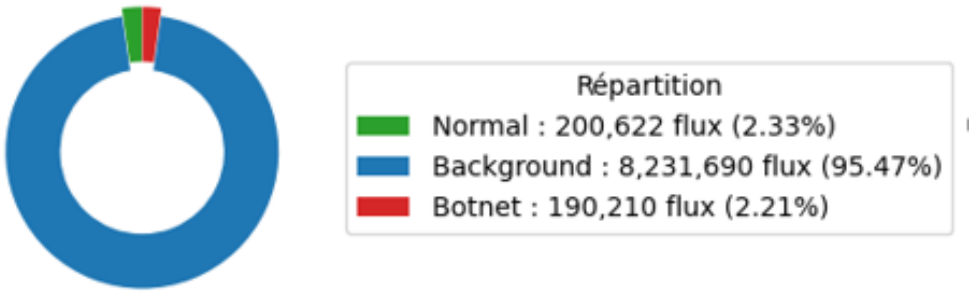

Figure 3.4 – Répartition du trafic CTU-13.

3.2.8 Résultats

Les métriques d'évaluations

Dans notre approche, nous avons évalué les performances du modèle à l'aide de plusieurs métriques adaptées à une tâche de classification binaire (botnet vs normal). Les notions de **vrai positif (TP), vrai négatif (TN), faux positif (FP) et faux négatif (FN)** constituent la base de ces évaluations :

- **Vrai Positif (TP) :** botnet correctement détecté
- **Vrai Négatif (TN) :** trafic normal correctement identifié
- **Faux Positif (FP) :** trafic normal classé à tort comme botnet (fausse alerte)
- **Faux Négatif (FN) :** botnet non détecté

Les métriques calculées à partir de ces valeurs sont les suivantes :

- **Accuracy :**Proportion de prédictions correctes (positives et négatives) sur l'ensemble des exemples.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.5}$$

- **Précision :** Proportion de prédictions positives qui sont réellement des botnets.

$$Prcision = \frac{TP}{TP + FP} \tag{3.6}$$

- **Rappel :** Proportion de botnets réellement détectés par le modèle.

$$Rappel = \frac{TP}{TP + FN} \tag{3.7}$$

- **F1-Score :** moyenne harmonique entre Précision et Rappel.

$$F1 - Score = \frac{2 \cdot (Rappel \cdot Prcision)}{Rappel + Prcision} \tag{3.8}$$

- **AUC :** capacité du modèle à séparer les classes, indépendamment du seuil de décision (valeur entre 0.5 et 1).
- **La courbe ROC :** illustre la capacité du modèle à séparer botnets et trafic normal sur l'ensemble des seuils.
- **Matrice de confusion :** tableau qui résume les performances du modèle sous forme de 4 cases : TP (vrais positifs), TN (vrais négatifs), FP (faux positifs), FN (faux négatifs). Permet de visualiser facilement les erreurs du modèle et comprendre son comportement global.

	Classe prédite : Normal 0	**Classe prédite : Botnet 1**
Classe réelle : Normal 0	Vrais négatifs (TN)	Faux positifs (FP)
Classe réelle : Botnet 1	Faux négatifs (FN)	Vrais positifs (TP)

Tableau 3.5 – Matrice de confusion.

— **Rapport de classification par classes du modèle KRAST**

	Précision	Rappel	F1-score	Support
0– normal	0.95	0.97	0.96	92610
1 - botnet	0.98	0.95	0.96	62479
accuracy			0.97	155089
Macro avg	0.97	0.97	0.97	155089
Weighted avg	0.97	0.97	0.97	155089

Tableau 3.6 – Le rapport de classification binaire KRAST.

Le rapport de classification du modèle KRAST (Tableau 3.6) met en évidence des performances équilibrées sur les deux classes « normal » et « botnet ».

On observe des valeurs de précision et de rappel élevées pour les deux catégories : 0,95/0,97 pour la classe normale, et 0,98/0,95 pour la classe botnet. Le F1-score, qui résume l'équilibre entre précision et rappel, atteint 0,96 sur chaque classe.

Ces résultats confirment que le modèle parvient à distinguer efficacement les flux normaux des flux malveillants, tout en limitant le risque de faux positifs ou de faux négatifs. Le score global de 97 % en accuracy valide la fiabilité générale de l'approche KRAST.

— **Évaluation binaire de Krast (Normal vs Botnet)**

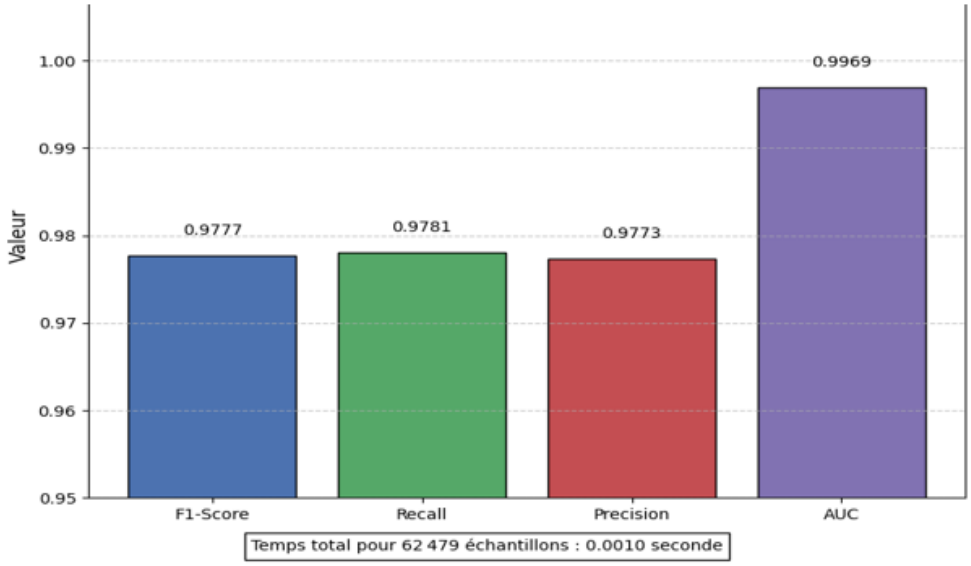

Figure 3.5 – Evaluation binaire de KRAST et le temps de détection des botnet.

Le graphique d'évaluation binaire (Figure 3.5) confirme la capacité du modèle à maintenir un haut niveau de performance, avec un excellent compromis entre précision et rappel.

Le temps de détection des botnets est un autre élément différenciant : KRAST affiche un temps de réponse de l'ordre de 1 ms par flux, ce qui le rend adapté aux environnements à forte contrainte temporelle, tels que les réseaux IoT.

Ce faible temps de détection, combiné à la qualité des prédictions, démontre l'efficacité de l'architecture hybride KRAST.

— **Matrice de confusion de KRAST**

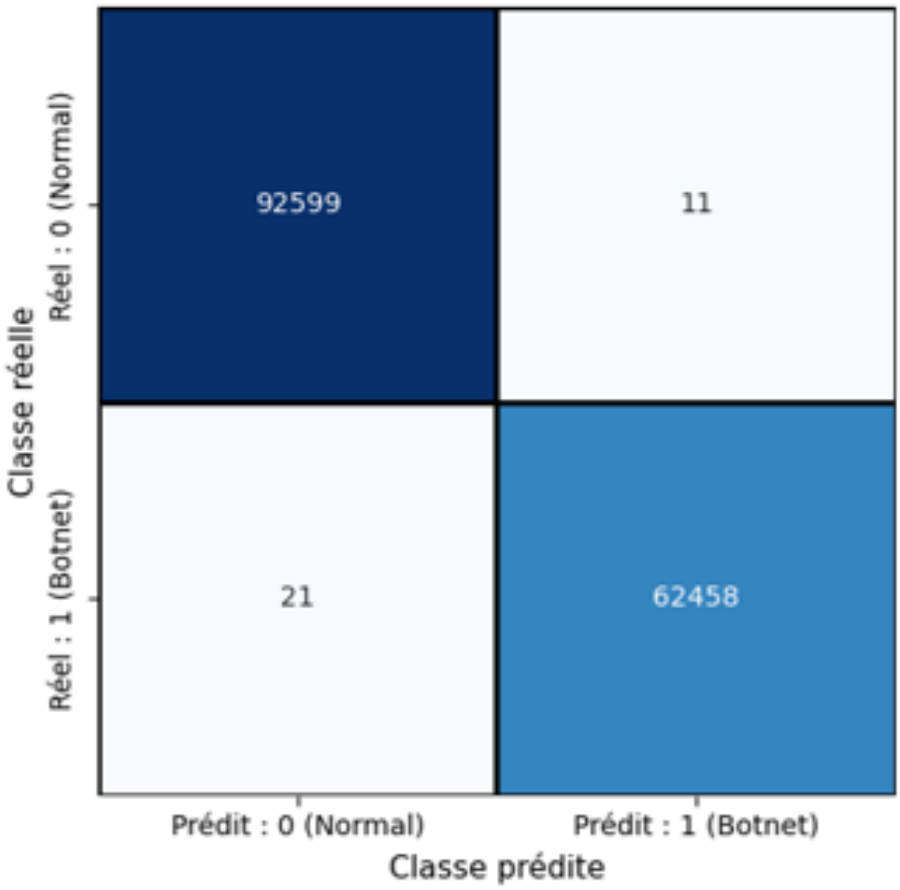

Figure 3.6 – Matrice de confusion du KRAST.

La matrice de confusion (Figure 3.6) met en évidence un très faible taux d'erreurs, avec un nombre limité de faux positifs (classifications à tort de flux normaux en botnets) et de faux négatifs (botnets non détectés).

Les diagonales principales montrent que le modèle distingue clairement les deux classes, et que les erreurs résiduelles restent très marginales au vu du volume global de données traité.

Cela témoigne d'une bonne stabilité du modèle et d'une capacité à généraliser face à la diversité du trafic.

— **La courbe ROC**

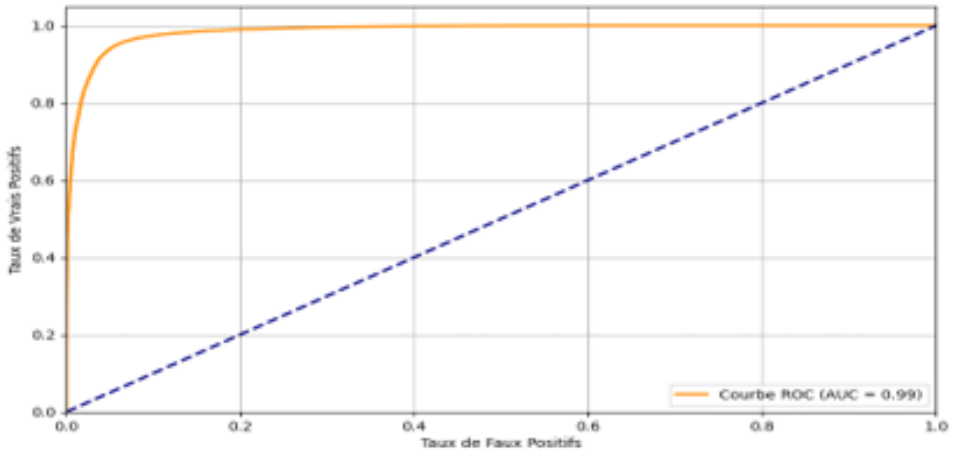

Figure 3.7 – Courbe ROC du KRAST.

La courbe ROC (Figure 3.7) illustre la performance du classifieur pour différents seuils de décision.

La courbe se rapproche fortement du coin supérieur gauche, traduisant une excellente capacité de discrimination entre les flux normaux et les flux botnets.

L'AUC (area under curve) atteint 99,69 %, ce qui confirme que KRAST parvient à détecter avec une très grande fiabilité les activités malveillantes, même en présence d'un trafic de fond complexe.

Comparaison avec d'autres modèles

	RF	**XGBoost**	**AE**	**RF+K-Means**	**KRAST**
Accuracy	96.88 %	97.42 %	85.30 %	86.95 %	97 %
Precision	44.23 %	61.05 %	75.12 %	86.96 %	97.73 %
F1-score	60.85 %	74.10 %	79.80 %	86.92 %	97.77 %
Recall	97.45 %	93.54 %	85.16 %	86.95 %	97.81 %
AUC	99.48 %	98.85 %	91.72 %	96.64 %	99.69 %

Tableau 3.7 – Comparaison du KRAST avec d'autres modèles.

Les résultats présentés dans le tableau (Tableau 3.7) ci-dessus permettent de comparer les performances de notre approche KRAST avec plusieurs autres modèles : Random Forest (RF), XGBoost, Auto-encodeur seul (AE), et la combinaison RF + K-Means. On observe que KRAST surpasse clairement les autres approches sur l'ensemble des métriques clés. En particulier, pour le critère de précision, qui reflète la capacité à éviter les faux positifs, KRAST atteint un score remarquable de 97,73 %, bien supérieur aux autres modèles (61,05 % pour XGBoost, 86,96 % pour RF+K-Means et seulement 44,23 % pour RF seul). On note également que KRAST affiche les meilleures performances en F1-score (97,77 %), en rappel (97,81 %) et en AUC (99,69 %), démontrant ainsi une excellente capacité à détecter efficacement les botnets, y compris dans des scénarios complexes. Ces résultats confirment que l'approche hybride KRAST constitue une solution particulièrement performante et robuste face aux autres méthodes testées.

Discussion des résultats obtenus

L'analyse des résultats du modèle KRAST (combinant Auto-encoder, K-Means et Stacking) met en évidence des performances nettement supérieures à celles des modèles utilisés individuellement. Le rapport de classification (Table 3.2.6.1) montre des scores élevés et équilibrés entre précision, rappel et F1-score pour les deux classes (normal et botnet), avec une accuracy globale de 97 %. Ce résultat confirme la fiabilité du modèle, même dans un environnement complexe et potentiellement bruité comme celui du dataset CTU-13.

Un point particulièrement remarquable est le F1-score de 0.96 pour la classe botnet, qui montre que le modèle détecte très bien les attaques tout en réduisant le nombre de fausses alertes. Cela est d'autant plus confirmé par la matrice de confusion (Figure 3.2.6.3), qui indique seulement 21 faux négatifs (attaques non détectées) et 11 faux positifs (faux soupçons). En cybersécurité, minimiser ces erreurs est essentiel, car un seul botnet non détecté peut compromettre tout un système.

La courbe ROC (Figure 3.7), avec une AUC de 99.69 %, montre que KRAST distingue parfaitement le trafic normal du trafic malveillant, même si les classes sont déséquilibrées. C'est un signe clair de la robustesse du modèle face à la variabilité du trafic réseau.

Comparé aux autres modèles (Tableau 3.7), KRAST prend l'avantage sur toutes les métriques :

- Il surpasse Random Forest et XGBoost, notamment en précision (97.73 % contre seulement 44.23 % pour RF).
- Il atteint le meilleur F1-score (97.77 %), ce qui montre une performance globale très homogène.
- Il reste meilleur que l'approche combinée RF + KMeans, qui pourtant améliore déjà les résultats.
- En plus de cela, KRAST est extrêmement rapide, avec un temps de détection de seulement 0.0010 seconde pour 62 479 échantillons, ce qui en fait un modèle adapté à une utilisation en temps réel.

Ce qui rend KRAST vraiment puissant, c'est justement l'idée de combiner des modèles supervisés et non supervisés. D'un côté, les modèles non supervisés comme l'Auto-encoder et K-Means permettent de mieux comprendre la structure cachée des données, même sans étiquette. Ils sont capables d'identifier des comportements anormaux subtils ou inconnus. De l'autre côté, les modèles supervisés comme Random Forest et le Stacking assurent une généralisation solide et une capacité d'apprentissage ciblée, en s'appuyant sur les connaissances connues (étiquettes).

En combinant les deux types d'approches, on tire parti des forces de chacun : la flexibilité des méthodes non supervisées, et la puissance prédictive des méthodes supervisées. Cette complémentarité est au cœur de l'efficacité de KRAST.

En conclusion, les résultats obtenus montrent que KRAST est une solution à la fois efficace, robuste et rapide pour la détection de botnets, en particulier dans des contextes IoT ou réseau très hétérogène, où les menaces évoluent rapidement et sont parfois difficiles à détecter par des méthodes classiques.

3.2.9 Comparative avec des études existantes

L'approche RVAE (Recurrent Variational Auto-encoder), proposée par Jeeyung Kim et ses collègues, repose sur une méthode entièrement non supervisée, spécifiquement conçue pour modéliser le comportement du trafic réseau normal et détecter les anomalies à partir de l'erreur de reconstruction. Son principal atout réside dans sa capacité à détecter des botnets inconnus, grâce à l'apprentissage de motifs de trafic jamais vus auparavant. Cette propriété est essentielle dans des environnements réseau où les menaces évoluent constamment. De plus, RVAE se distingue par un temps de détection très ra-

pide (10 millisecondes), ce qui en fait une solution intéressante pour des applications en temps réel.

Elle est également adaptée à des situations où les données annotées sont rares, comme c'est souvent le cas dans les réseaux IoT. Néanmoins, ses performances sur des attaques connues sont plus limitées (F1-score : 0.90, précision : 0.91, rappel : 0.89), et sa mise en œuvre est techniquement complexe, nécessitant des ajustements fins de plusieurs modules (VAE, GRU bidirectionnel, régularisation KL).

À l'opposé, l'approche KDR, développée par A. Arshad et al., adopte une stratégie supervisée classique, en combinant KNN, Decision Tree et Random Forest par vote majoritaire. Cette méthode atteint des scores quasi parfaits sur les données connues (F1-score : 100 %, précision : 99,7 %, rappel : 100 %), ce qui montre son efficacité dans des contextes où les attaques sont bien identifiées. Toutefois, cette haute précision s'accompagne d'un temps de traitement très élevé (˜13 secondes), ce qui limite fortement son usage en temps réel. De plus, KDR repose uniquement sur des données annotées, ce qui le rend peu efficace pour la détection de nouvelles attaques ou de botnets inconnus.

Face à ces deux extrêmes, l'approche KRAST propose une solution hybride alliant méthodes non supervisées et supervisées, pensée spécifiquement pour l'analyse du trafic réseau IoT. Son architecture repose sur trois étapes : (1) un auto-encodeur pour la réduction de dimension et l'apprentissage du trafic normal, (2) un clustering KMeans pour isoler le trafic de fond, et (3) une classification par stacking, combinant Random Forest et XGBoost.

Les résultats expérimentaux montrent que KRAST atteint une précision, un rappel et un F1-score de 97 %, avec un temps de détection ultra-rapide (˜1 milliseconde). Cela lui permet non seulement de reconnaître efficacement les attaques connues, mais aussi de détecter des anomalies inconnues grâce à la structure latente extraite par l'auto-encodeur et filtrée par le clustering.

Ce qui fait la force de KRAST, c'est sa capacité à s'adapter à la complexité du trafic réseau IoT, souvent massif, non équilibré et dynamique. Grâce à la combinaison du clustering non supervisé et des classifieurs supervisés puissants, le modèle parvient à généraliser sur des attaques inédites tout en maintenant une haute précision sur les motifs connus. Le stacking permet de renforcer la robustesse des décisions des attaques

.

— **Comparaison des performances**

Modèle	Précision	Rappel	F1-score	Temps de détection
RVAE	0.91	0.89	0.90	˜10 ms
KDR	0.997	1.00	1.00	˜13 s
KRAST	0.97	0.97	0.97	˜1 ms

Tableau 3.8 – Comparaison des performances

En tenant compte de l'ensemble des critères essentiels — précision, rappel, temps de réponse, capacité à détecter des botnets inconnus, robustesse face aux données déséquilibrées et originalité de l'approche — KRAST se démarque comme une solution à la fois équilibrée et performante. Elle se distingue notamment par un temps de traitement largement inférieur à celui de l'approche KDR, avec un temps de détection d'environ 1 ms, contre un délai estimé à près de 13 secondes pour KDR lors de la détection sur un dataset équivalent. KRAST offre également des performances globales supérieures à celles du modèle RVAE, et démontre une meilleure capacité de généralisation face aux attaques inconnues, comparée à KDR. Elle constitue ainsi une réponse concrète et efficace aux défis actuels de la détection de botnets dans des environnements réseau modernes, en particulier au sein des infrastructures IoT hétérogènes et critiques.

Conclusion

Ce chapitre présente en détail l'approche hybride KRAST, conçue pour répondre aux limites rencontrées par les méthodes classiques face à un trafic réseau hétérogène et bruité. En combinant un auto-encodeur non supervisé, un clustering K-Means, et des classificateurs supervisés (Random Forest et XGBoost), intégrés dans une architecture de stacking, KRAST permet d'atteindre des performances élevées, tant en précision qu'en capacité de généralisation.

Les résultats obtenus sur le dataset CTU-13 sont très encourageants, avec un F1-score de 97.77 %, une AUC de 99.69 % et un temps de détection de l'ordre de la milliseconde, rendu possible par l'optimisation sur TPU. Cette approche démontre ainsi sa pertinence

pour des scénarios réels de détection de botnets, notamment dans des environnements IoT à fort volume de données et à trafic non balisé.

KRAST constitue ainsi une solution robuste, évolutive et rapide, qui pourrait être adaptée et déployée dans de nombreux contextes de cybersécurité réseau.

Conclusion Générale

Tout au long de ce mémoire, nous avons exploré les enjeux liés à la détection des botnets dans le trafic réseau IoT. Face à la multiplication des objets connectés et à l'évolution constante des cyberattaques, il devient impératif de concevoir des solutions capables de s'adapter aux nouvelles menaces et à la diversité du trafic réseau.

L'analyse des approches existantes a mis en évidence les limites des méthodes purement supervisées ou non supervisées face à des scénarios complexes. Pour répondre à ces défis, nous avons proposé une approche hybride pour la détection des botnets appelé **KRAST**, qui exploite la complémentarité entre apprentissage non supervisé (autoencodeur + clustering K-Means) et apprentissage supervisé (classifieurs en stacking).

Notre approche KRAST parvient à concilier précision, rapidité et capacité de généralisation, même dans des contextes de trafic fortement déséquilibré ou partiellement inconnu.

Perspectives

Les résultats obtenus avec l'approche KRAST démontrent l'intérêt de combiner des méthodes non supervisées et supervisées pour améliorer la détection des botnets dans un trafic réseau complexe comme celui du dataset CTU-13.

Plusieurs pistes d'amélioration et de développement futur peuvent être envisagées pour renforcer l'efficacité et l'adaptabilité de l'approche KRAST :

- **Validation sur d'autres datasets :**Étendre les expérimentations à des jeux de données plus récents, hétérogènes ou réalistes pour tester la capacité de généralisation de KRAST dans divers environnements.
- **Utilisation d'auto-encodeurs avancés :**Explorer des variantes plus complexes, comme les Variational Auto-encoders (VAE) ou les auto-encodeurs récurrents (RNN- AE), afin de mieux modéliser la dimension temporelle des flux réseau.
- **Techniques de data augmentation :**Introduire des méthodes d'augmentation de données spécifiques au domaine du trafic réseau pour enrichir l'ensemble d'entraînement et améliorer la robustesse du modèle.
- **Calibration des probabilités dans le stacking :**Intégrer des mécanismes de calibration pour affiner les scores de confiance des prédictions, et ainsi réduire l'incertitude dans les décisions finales.
- **Optimisation temps réel sur TPU :**Tirer parti des unités de traitement TensorFlow (TPU) pour améliorer la vitesse d'inférence et rendre KRAST exploitable en contexte de détection en temps quasi réel.
- **Déploiement dans des systèmes distribués :**Envisager une intégration de KRAST dans des architectures de sécurité distribuées (IDS/IPS), en vue d'un déploiement industriel à large échelle.

Bibliographie

[1] J. T. H. Win et al., « "Securing the Internet of Things : Strategies for a Resilient Cyber-Physical Ecosystem," » *Preprints,*, jan. 2025, [Consulté le : 4 mars 2025]. doi : 10.20944/preprints202501.0610.v1..

[2] B. Kaissa et B. Ghada, « Les Objets connectés Internet of Everything «IoE», » Doctoral dissertation, Univ. Mouloud Mammeri, 2017.

[3] U. I. des Télécommunications (UIT)., Internet des objets – Rapport de synthèse.Genève., (2015).

[4] A. I. L. Atzori et G. Morabito, « "The Internet of Things : A survey" », *Computer Networks*, t. vol.54 , n° no.15, pp. 2787-2805, oct. 2010, [Consulté le : 16 mars 2025]. doi : 10.1016/j.comnet.2010.05.010..

[5] «. I. des objets », *Futura-Sciences*. https://www.futura-sciences.com/tech/definitions/internet-internet-objets-15158/?source=post_page----------., [Consulté le : 16 mars 2025].

[6] «. A. de l'Internet des objets », *Algerie IoT*. http://algerieiot.free.fr/pages/architecture.html., [Consulté le : 14 mai 2025].

[7] F. Skandrani. (fév. 2022). Architecture de l'Internet des objets. [Consulté le : 5 mars 2025], adresse : https://iotindustriel.com/iot-iiot/architecture-iot-lessentiel-a-savoir/.

[8] B. Nath. (mai 2024). IoT Architecture. [Consulté le : 5 mars 2025], adresse : https://geekflare.com/fr/iot-architecture/.

[9] S. J. D. Navani et S. Nehra, « L'Internet des objets (IoT) : une étude des éléments architecturaux », in *13e Conférence internationale sur les technologies signal-image et les systèmes basés sur Internet (SITIS)*, IEEE, déc. 2017, p. 473-478.

[10] No Author. (n.d.). Five-Layer IoT Architecture. [Consulté le : 16 mars 2025], adresse : https://www.researchgate.net/publication/324797771/figure/fig1/AS:633748894785536@1528108919467/Five-Layer-IoT-Architecture.png.

[11] W. J. R. Liu et T. He, « Xfi : Collecte de données IoT inter-technologies via le Wi-Fi standard », in *28e Conférence internationale IEEE sur les protocoles réseau (ICNP)*, IEEE, oct. 2020, p. 1-11.

[12] J. C. A. M. L. P. Cope et et al., « Security vulnerabilities in Bluetooth technology as used in IoT », *Journal of Sensor and Actuator Networks*, t. 7, n° 3, p. 28, 2018.

[13] K. O. V. Coskun et B. Ozdenizci, *Communication en champ proche (NFC) : de la théorie à la pratique*. John Wiley & Sons, 2011.

[14] K. A. et M. Harrag, « Surveillance des zones critiques et des accès non autorisés en utilisant la technologie RFID », 2022.

[15] J.-M. S. G. R. Stanica, Y. Mouline et et al., « Thèse de doctorat », thèse de doct., INSA Lyon ; SPIE ICS, 2020.

[16] LEMAGIT. (2022). Comment l'IoT automatise les tâches et augmente la productivité. [Consulté le : 15 mai 2025], adresse : https://www.lemagit.fr.

[17] PCINDUS. (2021). IoT et sécurité urbaine : surveillance intelligente. [Consulté le : 16 mai 2025], adresse : https://www.pcindustries.com.

[18] FORTINET. (2023). Vulnérabilités des appareils connectés et solutions. [Consulté le : 16 mai 2025], adresse : https://www.fortinet.com.

[19] C. Corse. (2024). Analyse des menaces et réglementations IoT. [Consulté le : 17 mai 2025], adresse : https://www.clusir-corse.fr.

[20] Efani. (2025). 9 défis majeurs de l'IoT en 2025. [Consulté le : 16 mai 2025], adresse : https://www.efani.com.

[21] A. S. Tanenbaum et D. J. Wetherall, *Computer Networks, 5e éd.* Pearson, 2011.

[22] A. V. Dastjerdi et R. Buyya, « Fog computing : Helping the Internet of Things realize its potential », *Computer*, t. 49, n° 8, p. 112-116, 2016.

[23] Y. Xiang, W. Zhou, M. Guo et L. T. Yang, « Understanding network behavior with flow analysis », *IEEE Network*, t. 25, n° 4, p. 6-11, juil. 2011.

[24] P. K. Sharma et J. H. Park, « Blockchain based hybrid network architecture for the smart city », *Future Generation Computer Systems*, t. 86, p. 650-655, sept. 2018.

[25] TECHTARGET. (n.d.). Inbound vs outbound firewall rules. [Consulté le : 18 mai 2025], adresse : https://www.techtarget.com/searchsecurity/definition/inbound-and-outbound-traffic.

[26] J. Postel. (1981). Transmission Control Protocol. [Consulté le : 12 mai 2025], adresse : https://www.rfc-editor.org/info/rfc793.

[27] ——, (1981). Transmission Control Protocol. [Consulté le : 18 mai 2025], adresse : https://www.rfc-editor.org/info/rfc793.

[28] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng et J. Zhang, « Botnet : Classification, attacks, detection, tracing, and preventive measures », *EURASIP Journal on Wireless Communications and Networking*, t. 2009, n° 1, p. 1-11, 2009.

[29] G. Combs. (n.d.). Wireshark : The world's foremost network protocol analyzer. [Consulté le : 12 mai 2025], adresse : https://www.wireshark.org.

[30] R. Sommer et V. Paxson, « Outside the Closed World : On Using Machine Learning for Network Intrusion Detection », in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 2010, p. 305-316. doi : 10.1109/SP.2010.25.

[31] Anonymous, « Outside the Closed World : On Using Machine Learning for Network Intrusion Detection », in *Proc. IEEE Symposium on Security and Privacy (SP)*, 2010, p. 305-316.

[32] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng et J. Zhang, « Botnet : Classification, Attacks, Detection, Tracing, and Preventive Measures », *EURASIP Journal on Wireless Communications and Networking*, t. 2009, n° 1, p. 1-11, 2009.

[33] M. Zolanvari, A. Teixeira, L. Gupta, K. M. Khan et R. Jain, « Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things », *IEEE Internet of Things Journal*, t. 6, n° 4, p. 6822-6834, août 2019.

[34] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari et M. Ayyash, « Internet of Things : A Survey on Enabling Technologies, Protocols, and Applications », *IEEE Communications Surveys & Tutorials*, t. 17, n° 4, p. 2347-2376, 2015, Fourth Quarter.

[35] M. Zolanvari, A. Teixeira, L. Gupta, K. M. Khan et R. Jain, « Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things », *IEEE Internet of Things Journal*, t. 6, n° 4, p. 6822-6834, août 2019.

[36] Proofpoint. (n.d.). Botnet – Threat Reference. [Consulté le : 2 mars 2025], adresse : https://www.proofpoint.com/us/threat-reference/botnet.

[37] Kaspersky. (n.d.). Botnet Attacks. [Consulté le : 2 mars 2025], adresse : https://www.kaspersky.fr/resource-center/threats/botnet-attacks.

[38] Fortinet. (n.d.). What is a Botnet. [Consulté le : 2 mars 2025], adresse : https://www.fortinet.com/fr/resources/cyberglossary/what-is-botnet.

[39] Cloudflare. (2025). What is a DDoS Botnet ? [Consulté le : 2 mars 2025], adresse : https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/.

[40] Netacea. (n.d.). Types of Botnets. [Consulté le : 3 mars 2025], adresse : https://netacea.com/learn/types-of-botnets/.

[41] E. Freyssinet, « Lutte contre les botnets : analyse et stratégie », [Consulté le : 4 mars 2025], thèse de doct., Université Pierre et Marie Curie - Paris VI, 2015.

[42] A. Karami et M. Guerrero-Zapata, « A fuzzy rule-based system for detecting DDoS attacks in cloud computing environments », *International Journal of Information Security*, t. 14, n° 3, p. 261-273, 2015.

[43] S. R. Chowdhury, S. Hoque et M. Zaber, « Botnet detection and mitigation for home routers : A survey », *Computer Networks*, t. 194, p. 108 132, 2021.

[44] S. M. Zargar, J. Joshi et D. Tipper, « A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks », *IEEE Communications Surveys & Tutorials*, t. 15, n° 4, p. 2046-2069, 2013.

[45] No Author. (n.d.). Basic entities of botnet (bots, command and control, C&C server, and bot master). [Consulté le : 19 mars 2025], adresse : https://www.researchgate.net/publication/335316113/figure/fig1/AS:794509667491840@1566437274809/Basic-entities-of-botnet-bots-command-and-control-C-C-server-and-bot-master.ppm.

[46] D. Dagon et et al., « A Taxonomy of Botnet Structures », in *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC)*, [Consulté le : 7 mars 2025], 2007, p. 325-339. doi : 10.1109/ACSAC.2007.21.

[47] No Author. (n.d.). Structures of the botnet : a) centralized structure, b) decentralized structure. [Consulté le : 19 mars 2025], adresse : https://www.researchgate.net/publication/299350086/figure/fig2/AS:667890764238855@1536248975216/Structures-of-the-botnet-a-centralized-structure-b-decentralized-structure.png.

[48] S. Karuppayah, T. Herawan, A. M. N. Omar et et al., « Monitoring Peer-to-Peer Botnets : Requirements, Challenges, and Future Works », *Computers, Materials & Continua*, t. 75, n° 2, p. 3375-3398, 2023, [Consulté le : 7 mars 2025]. doi : 10.32604/cmc.2023.038388.

[49] No Author. (n.d.). Architecture of P2P Botnet. [Consulté le : 19 mars 2025], adresse : https://www.researchgate.net/profile/Wan-Ahmad-Ramzi-Wan-Yusuf/publication/322114602/figure/fig1/AS:578243705409536@1514875450216/Architecture-of-P2P-Botnet-5_W640.jpg.

[50] R. Villamarín-Salomón et et al., « Hybrid Peer-to-Peer Botnets », *IEEE Transactions on Dependable and Secure Computing*, t. 18, n° 3, p. 1124-1139, 2021. doi : 10.1109/TDSC.2019.2913366.

[51] *Hybrid Architecture*, https://www.researchgate.net/publication/261154715/figure/fig3/AS:360575755407363@1462979367392/Hybrid-Architecture.png, [Consulté le : 19 mars 2025].

[52] K. Thomas et et al., « Watering Hole Attacks Analysis », in *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, 2022.

[53] R. Villamarín-Salomón, « Botnet C&C Architectures », *IEEE Access*, 2023.

Type your text

[54] A. Aleroud, K. Zohdy et A. Al-Dhelaan, « Context-Aware Detection of Cyber Attacks in IoT Systems », *IEEE Internet of Things Journal*, t. 10, n° 3, p. 2112-2124, 2023. DOI : 10.1109/JIOT.2023.123456.

[55] M. Antonakakis, T. April, M. Bailey et et al., « Understanding the Mirai Botnet », in *Proc. of the 26th USENIX Security Symposium*, [Consulté le : 19 mai 2025], 2017, p. 1093-1110. adresse : https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf.

[56] T. Moore et B. Edelman, « Click Fraud in Online Advertising : A Case Study », *IEEE Security & Privacy*, t. 18, n° 3, p. 53-60, 2020. DOI : 10.1109/MSEC.2020.1234567.

[57] B. Stone-Gross, R. Abman, R. Kemmerer, C. Kruegel et G. Vigna, « The Underground Economy of Spyware », in *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS)*, [Consulté le : 19 mai 2025], 2009, p. 285-296. adresse : https://www.cs.ucsb.edu/~vigna/publications/2009_stonegross_spyware.pdf.

[58] C. Canali, A. Lanzi et D. Balzarotti, « Mining Your Ps and Qs : Detection of Widespread Weak Keys in Network Devices », in *Proc. of the 21st USENIX Security Symposium*, [Consulté le : 19 mai 2025], 2012. adresse : https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final50.pdf.

[59] N. R. Mead, D. Stehney et R. E. Linger, « Security Threats and Risks in Cyber Physical Systems : A Case Study Approach », *IEEE Security & Privacy*, t. 10, n° 4, p. 58-63, juil. 2012, [Consulté le : 19 mai 2025]. DOI : 10.1109/MSP.2012.58.

[60] A. Haddadi et A. N. Zincir-Heywood, « Botnet Detection Using Network Traffic Analysis », *Journal of Computer Virology and Hacking Techniques*, t. 12, n° 2, p. 59-72, mai 2016. DOI : 10.1007/s11416-015-0252-0.

[61] K. Singh, S. C. Guntuku, A. Thakur et C. Hota, « Big Data Analytics Framework for Peer-to-Peer Botnet Detection Using Random Forests », *Procedia Computer Science*, t. 78, p. 443-451, 2016.

[62] A. Patcha et J. M. Park, « An Overview of Anomaly Detection Techniques : Existing Solutions and Latest Technological Trends », *Computer Networks*, t. 51, n° 12, p. 3448-3470, août 2007. doi : 10.1016/j.comnet.2007.05.005.

[63] S. Behal, « Signature-based Botnet Detection and Prevention », *International Journal of Engineering and Technology*, t. 7, n° 5, p. 220-225, 2015.

[64] Y. Mehmood, I. Awan, M. M. Younas et S. A. Madani, « Behavior-Based Analysis of Malicious Bots in Internet of Things (IoT) Devices », *Journal of Network and Computer Applications*, t. 174, p. 102 865, fév. 2021. doi : 10.1016/j.jnca.2020.102865.

[65] M. Ndiaye, T. A. Diallo et K. Konaté, « ADEFGuard : Anomaly Detection Framework Based on Ethereum Smart Contracts Behaviours », *Blockchain : Research and Applications*, t. 4, n° 3, p. 100 148, 2023.

[66] M. Morel, F. Allard, R. Dubois et J.-F. Gompel, « CASTAFIOR : Détection automatique de tunnels illégitimes par analyse statistique du trafic DNS », *Revue des Sciences et Technologies de l'Information - Série RIA*, t. 35, n° 2, p. 123-140, 2021.

[67] A. Osia, A. H. Jahangir et A. Dehghantanha, « Botnet Detection Using Machine Learning Techniques : A Survey », *Computers & Security*, t. 105, p. 102 297, mar. 2021. doi : 10.1016/j.cose.2021.102297.

[68] J. Kim, A. Sim, J. Kim et K. Wu, « Botnet Detection Using Recurrent Variational Autoencoder », in *Proc. 19th IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, Miami, FL, USA, 2020, p. 785-790. doi : 10.1109/ICMLA51294.2020.00132.

[69] L. Mhamdi et M. M. Isa, « Securing SDN : Hybrid Autoencoder–Random Forest for Intrusion Detection and Attack Mitigation », *Journal of Network and Computer Applications*, t. 225, p. 103 868, 2024.

[70] T. Zhukabayeva, L. Zholshiyeva, K. Ven-Tsen, A. Adamova, Y. Mardenov et N. Karabayev, « Enhancing IoT Security : Effective Botnet Attack Detection Through Machine Learning », in *Procedia Computer Science*, t. 241, 2024, p. 421-426.

[71] A. Arshad, M. Jabeen, S. Ubaid, A. Raza, L. Abualigah, K. Aldiabat et H. Jia, « A Novel Ensemble Method for Enhancing Internet of Things Device Security Against Botnet Attacks », *Decision Analytics Journal*, t. 8, p. 100 307, 2023.

[72] Q. B. Baker et A. Samarneh, « Feature Selection for IoT Botnet Detection Using Equilibrium and Battle Royale Optimization », *Computers & Security*, t. 147, p. 104 060, 2024.

[73] CTU-13 Dataset, https://www.kaggle.com/datasets/sherinclaudio/ctu13-dataset, [Consulté le : 21 mai 2025].

[74] CICIDS2017 Dataset, https://www.kaggle.com/datasets/dhanushnarayananr/cicids2017, [Consulté le : 21 mai 2025].

[75] N-BaIoT Dataset, https://www.kaggle.com/datasets/boltzmannbrain/n-baiot-dataset, [Consulté le : 21 mai 2025].

[76] S. García, M. Grill, J. Stiborek et A. Zunino, "An empirical comparison of botnet detection methods", Computers & Security, t. 45, p. 100-123, 2014.

[77] L. Bilge, E. Kirda, C. Kruegel et M. Balduzzi, "EXPOSURE : Finding Malicious Domains Using Passive DNS Analysis", in NDSS, 2011.

[78] Y. Zhao,Y.Xie,F. Yu,Q.Ke,Y.Yu,Y.ChenetE.Gillum,"BotGraph: Large Scale Spamming Botnet Detection", in NSDI, 2009.

[79] M. F. et al., Symantec Internet Security Threat Report, 2009.

[80]S. Garcia, M. Grill, J. Stiborek et A. Zunino, « CTU-13 Dataset: Synthetic "Menti" Traffic Generated for Experiments », 2014.

[81] M. E. et al., "A Survey on Automated Dynamic Malware-Analysis Techniques and Tools", ACM Computing Surveys, t. 44, no 2, 2012.

[82] Symantec Threat Report, "Trojan.NSIS.Inject" analysis, 2018.

[83] A. A. et al., "Understanding the Mirai Botnet", in USENIX Security, 2017.

[84] Y. M. et al., "Detection of Unauthorized IoT Devices Using Machine Learning Techniques", arXiv preprint arXiv :1709.04647, 2017.

[85] Google, VirusTotal– Online Malware Scanner, [Consulté : 21-mai-2025], 2023.

[86] AV-Test, Antivirus Comparative Results, [Consulté : 21-mai-2025], 2023.

[87] Microsoft, Windows Defender Firewall Documentation, [Consulté : 21 mai-2025], 2023.

[88] APNIC, Botnet Mitigation Toolkit, 2023.

[89] MITRE, ATT&CK Framework for Cyber Threat Intelligence, 2023.

[90] D. Pastre, L'intelligence artificielle : Définition– Généralités– Historique– Domaines. Université Paris, 2000.

[91] S. N. et al., "Vers un usage des méthodes d'intelligence artificielle en génie des procédés", in Proc. SFGP, Nantes, France, 2019.

[92] G. Briganti, "Intelligence artificielle : une introduction pour les cliniciens", Rev. Mal. Respir., t. 40, no 4, p. 308-313, 2023. doi : 10.1016/j.rmr.2023.02.005.

[93] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville : Deep Learning : The MIT Press, 2016, 800 pp, ISBN : 0262035618", Genetic Programming and Evolvable Machines, t. 19, no 1, p. 305-307, 2018.

[94] T. Hastie, R. Tibshirani et J. H. Friedman, Les éléments de l'apprentissage statistique : Exploration de données, inférence et prédiction. New York : Springer, 2009, t. 2.

[95] O. Chapelle, B. Schölkopf et A. Zien, "Discussion sur l'apprentissage semi-supervisé et la transduction", in Apprentissage semi-supervisé, MIT Press, 2006, p. 473-478.

[96] M. Zimmer, "Apprentissage par renforcement développemental", thèse de doct., Université de Lorraine, 2018.

[97] Deep learning, https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/, [Consulté le : 09 mars 2025].

[98] Deep Learning : Définition, https://datascientest.com/deep-learning definition, [Consulté le : 09 mars 2025].

[99] Image Deep Learning, https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSi378fZmrS-zYeW928FQHP5W9t1QDZqE59Hg&s, [Consulté le : 09 mars 2025].

[100] Qu'est-ce que le Deep Learning?, https://www.oracle.com/africa fr/artificial- intelligence/machine- learning/what- is- deep learning/, [Consulté le : 09 mars 2025].

[101] Qu'est-ce qu'un réseau de neurones?, https://aws.amazon.com/fr/what is/neural-network/, [Consulté le : 09 mars 2025].

[102] Neural Networks, https://www.ibm.com/fr-fr/think/topics/neural networks, [Consulté le : 09 mars 2025].

[103] J. R. Quinlan, "Induction of decision trees", Machine Learning, t. 1, no 1, p. 81-106, 1986. doi : 10.1007/BF00116251.

[104] L. Breiman, "Random forests", Machine Learning, t. 45, no 1, p. 5-32, 2001. doi : 10.1023/A:1010933404324.

[105] Image Random Forest, https://miro.medium.com/v2/resize:fit: 592/1*i0o8mjFfCn-uD79-F1Cqkw.png, [Consulté le : 09 mars 2025].

[106] J. H. Friedman, "Greedy function approximation : A gradient boosting machine", Annals of Statistics, t. 29, no 5, p. 1189-1232, 2001. doi : 10. 1214/aos/1013203451.

[107] Image XGBoost Training, https://www.researchgate.net/publication/ 382460289/figure/fig1/AS:11431281262732010@1721705457338/Error Reduction-through-Progressive-Training-Using-Extreme-Gradient Boosting.png, [Consulté le : 22 mars 2025].

[108] T. CoveretP.Hart,"Nearestneighbor pattern classification", IEEE Trans. Information Theory, t. 13, no 1, p. 21-27, 1967. doi : 10.1109/TIT.1967. 1053964.

[109] D. W. Hosmer, S. Lemeshow et R. X. Sturdivant, Applied Logistic Re gression, 3rd. Wiley, 2013.

[110] J. MacQueen, "Some methods for classification and analysis of multiva riate observations", in Proc. 5th Berkeley Symp. Mathematical Statistics and Probability, 1967, p. 281-297.

[111] C. Cortes et V. Vapnik, "Support-vector networks", Machine Learning, t. 20, no 3, p. 273-297, 1995. doi : 10.1007/BF00994018.

[112] K. Pearson, "On lines and planes of closest fit to systems of points in space", Philosophical Magazine, t. 2, no 11, p. 559-572, 1901. doi : 10.1080/ 14786440109462720.

[113] Z. Zhou, Ensemble Methods : Foundations and Algorithms. Chapman et Hall/CRC, 2012.

[114] T. G. Dietterich, "Ensemble methods in machine learning", in Proc. 1st Int. Workshop on Multiple Classifier Systems, 2000, p. 1-15. doi : 10.1007/ 3-540-45014-9_1.

[115] F. Rosenblatt, "The perceptron : A probabilistic model for information storage and organization in the brain", Psychological Review, t. 65, no 6, p. 386-408, 1958. doi : 10.1037/h0042519.

[116] D. E. Rumelhart, G. E. Hinton et R. J. Williams, "Learning internal representations by error propagation", in Parallel Distributed Processing : Explorations in the Microstructure of Cognition, vol. 1 : Foundations, MIT Press, 1986, p. 318-362.

[117] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures", in Proc. ICML Workshop on Unsupervised and Transfer Learning, 2012, p. 37 50.

[118] Image Autoencoder, https://miro.medium.com/max/1200/1*ViBG49eTCKqqO2UVRL9mEw.png, [Consulté le : 30 mars 2025].

[119] J. L. Elman, "Finding structure in time", Cognitive Science, t. 14, no 2, p. 179-211, 1990. doi : 10.1207/s15516709cog1402_1.

[120] M. H. Hossain, TONIoTDatasets, https://www.kaggle.com/datasets/mhmdhossain/toniot-datasets.

[121] Google, Google Colaboratory, https://colab.research.google.com.

Résumé

Avec l'expansion rapide de l'Internet des Objets (IoT), les réseaux modernes sont de plus en plus exposés à des menaces sophistiquées, notamment sous la forme de botnets. Ces réseaux de dispositifs compromis sont capables de mener des attaques massives et souvent indétectables, en exploitant la faible sécurité de nombreux objets connectés.
La détection efficace de ces botnets dans des environnements complexes, où le trafic est hétérogène et évolutif, représente aujourd'hui un défi majeur. Ce mémoire propose une approche hybride, baptisée **KRAST**, combinant des techniques d'apprentissage non supervisé (autoencodeur + clustering K-Means) avec des méthodes supervisées (Random Forest, XGBoost) intégrées dans une architecture de stacking.
Après une revue critique des approches existantes, l'architecture KRAST a été testée sur le dataset CTU-13, reconnu pour sa complexité. Les résultats démontrent des performances robustes, avec une bonne capacité de généralisation face à des attaques connues et inconnues, et un bon équilibre entre précision, vitesse et adaptabilité.
Ce travail ouvre des perspectives pour le développement de solutions plus efficaces de détection de botnets, adaptées aux spécificités des réseaux IoT modernes. L'approche proposée pourrait être enrichie par des optimisations en temps réel et par l'intégration de techniques plus avancées de deep learning.

Mots Clés
Botnet — Internet des Objets (IoT) — Détection d'anomalies — Apprentissage supervisé — Apprentissage non supervisé — Deep learning — Autoencodeur — Clustering K-Means — Stacking — Random Forest — XGBoost — Sécurité des réseaux.

Abstract

With the rapid expansion of the Internet of Things (IoT), modern networks are increasingly exposed to sophisticated threats, particularly in the form of botnets. These networks of compromised devices are capable of launching large-scale and often undetectable attacks by exploiting the weak security of many connected objects.

Effectively detecting these botnets in complex environments, where network traffic is heterogeneous and constantly evolving, remains a major challenge. This thesis proposes a hybrid approach, called **KRAST**, combining unsupervised learning techniques (autoencoder + K-Means clustering) with supervised methods (Random Forest, XGBoost) integrated within a stacking architecture.

Following a critical review of existing approaches, the KRAST architecture was evaluated on the CTU-13 dataset, known for its complexity. The results demonstrate robust performance, with good generalization capabilities for both known and unknown attacks, achieving a strong balance between accuracy, speed, and adaptability.

This work opens new perspectives for developing more effective botnet detection solutions, specifically adapted to the unique challenges of modern IoT networks. The proposed approach could be further enhanced with real-time optimizations and the integration of more advanced deep learning techniques.

Keywords

Botnet — Internet of Things (IoT) — Anomaly Detection — Supervised Learning — Unsupervised Learning — Deep Learning — Autoencoder — K-Means Clustering — Stacking — Random Forest — XGBoost — Network Security.