Apprentissage automatique pour la détection et la localisation d'erreurs de configuration dans un réseau d'opérateur

El-Heithem Mohammedi

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
l'Université de Toulouse III - Paul Sabatier
Discipline ou spécialité :
Informatique et Télécommunications

Présentée et soutenue le *04/07/2023* **par :**
El-Heithem MOHAMMEDI

Apprentissage automatique pour la détection et la localisation d'erreurs de configuration dans un réseau d'opérateur

---

JURY

Selma Boumerdassi	Maître de Conférences (HDR), Cnam Paris	Rapporteur
Yassine Hadjadj-Aoul	Professeur, Université de Rennes 1	Rapporteur
Emmanuel Chaput	Professeur, Toulouse INP	Examinateur
Guillaume Fleury	Ingénieur, IMS Networks	Invité
Emmanuel Lavinal	Maître de Conférences (HDR), Université Toulouse III	Directeur

---

École doctorale :
Mathématiques, Informatique, Télécommunications de Toulouse (MITT)
Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (IRIT)
Directeur de Thèse :
Emmanuel Lavinal

Résumé

La vérification de configuration dans les réseaux d'opérateurs constitue un enjeu crucial pour garantir la qualité et la continuité des services fournis aux clients, tels que le service de réseaux privés virtuels (VPN). Les méthodes classiques de vérification de configuration présentent plusieurs limites, notamment en termes de complexité et de passage à l'échelle. Elles sont généralement basées sur des règles et des contraintes qui doivent être mises à jour manuellement. La mise à jour manuelle de cette base de règles est très chronophage et ne garantit pas sa complétude par rapport à toutes les erreurs de configuration possibles. De plus, ces méthodes classiques ont des difficultés à s'adapter aux évolutions des réseaux et aux besoins spécifiques des clients. Cette thèse, intitulée « *Apprentissage automatique pour la détection et la localisation d'erreurs de configuration dans un réseau d'opérateur* », explore l'utilisation des méthodes d'apprentissage automatique pour améliorer la vérification de configuration des réseaux d'opérateurs.

Le travail présenté dans cette thèse se divise en deux parties principales. La première partie consiste en l'application de méthodes d'apprentissage supervisé pour détecter et localiser les incidents liés à des erreurs de configuration. Le travail réalisé dans cette partie nous a permis d'étudier la modélisation des données de configuration et de comparer les performances de trois modèles d'apprentissage automatique différents : un modèle classique d'arbres de décision (DT), un modèle de méthodes d'ensemble (RF) et un modèle de réseaux de neurones (MLP). La seconde partie aborde la problématique de vérification de configuration. Nous avons proposé une approche basée sur les réseaux de neurones de graphes (GNN) pour détecter et localiser les erreurs de configuration de routage dans les réseaux VPN BGP/MPLS de niveau 3. Les résultats obtenus démontrent le potentiel des méthodes d'apprentissage automatique, en particulier les réseaux de neurones de graphes, pour améliorer la vérification de configuration dans les réseaux d'opérateurs. Ils soulignent également certaines limites et ouvrent des perspectives d'amélioration pour les travaux futurs.

Mots clés : Réseaux d'opérateurs, Réseaux Virtuels Privés (VPN), Gestion de configuration, Vérification de configuration, Apprentissage automatique, Réseaux de Neurones de Graphes (GNN).

Abstract

Configuration verification in network operators is crucial for ensuring the quality and continuity of services provided to customers, such as virtual private networks (VPN) services. Traditional configuration verification methods have several limitations, particularly in terms of complexity and scalability. They generally rely on rules and constraints that need to be manually updated. These manual updates are time-consuming and do not guarantee completeness with respect to all possible configuration errors. Furthermore, these traditional methods struggle to adapt to network evolutions and specific customer needs. This thesis explores the use of machine learning methods to improve configuration verification in network service providers.

The work presented in this thesis is divided into two main parts. The first part involves the application of supervised learning methods to detect and locate incidents related to configuration errors. The work carried out in this part allowed us to study the modeling of configuration data and to compare the performances of three different machine learning models : a classic decision tree model (DT), an ensemble method model (RF), and a neural network model (MLP). The second part addresses the issue of configuration verification. We proposed a graph neural network (GNN) approach to detect and locate routing configuration errors in BGP/MPLS layer 3 VPN networks. The results obtained demonstrate the potential of machine learning methods, particularly Graph Neural Networks, for improving configuration verification in network operators. They also underline some limitations and open up opportunities for improvement in future work.

Keywords : Network service providers, Virtual Private Networks (VPN), Configuration management, Configuration verification, Machine learning, Graph Neural Networks (GNN).

Remerciements

Je souhaite adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué, d'une manière ou d'une autre, à l'accomplissement de mes travaux de thèse.

Tout d'abord, je tiens à exprimer ma gratitude envers **IMS Networks** et **IRIT** pour avoir mis à ma disposition tous les moyens nécessaires pour mener à bien mes travaux de thèse et pour m'avoir offert un environnement de travail stimulant et enrichissant.

Je tiens à remercier les membres du Jury de thèse, les rapporteurs **Selma Boumerdassi** et **Yassine Hadjadj-Aoul**, ainsi que l'examinateur **Emmanuel Chaput**, pour avoir accepté d'évaluer mes travaux.

Je remercie chaleureusement mon directeur de thèse, **Emmanuel Lavinal**, qui m'a proposé ce projet passionnant et a su me guider tout au long de ces années. Sa disponibilité, son expertise technique, son soutien organisationnel et moral ont été pour moi une source d'inspiration et de motivation inépuisable.

Je suis également reconnaissant envers **Guillaume Fleury**, mon encadrant chez IMS Networks, pour sa patience, sa bienveillance et son expertise. Il a toujours été disponible pour répondre à mes questions et m'aider à surmonter les difficultés rencontrées au cours de cette thèse.

Je souhaite également exprimer ma gratitude à **Nicolas Douvillé**, l'ancien directeur technique de la BU Réseaux chez IMS Networks, qui m'a offert cette opportunité de thèse unique.

Un grand merci à **mes collègues d'IMS Networks** pour leur camaraderie, leur collaboration et les encouragements qu'ils m'ont prodigués lors de nos échanges informels et pauses café. Leur présence a rendu ces années de travail d'autant plus agréables et stimulantes.

Je tiens également à remercier ma famille pour leur soutien infaillible tout au long de ce parcours :

À ma chère maman Ouahiba, je tiens à te témoigner toute ma reconnaissance et ma profonde gratitude pour tout ce que tu as fait pour moi et mes sœurs. Malgré les défis et les difficultés, tu as été un pilier solide dans nos vies, nous offrant un cadre de vie exceptionnel et une éducation de qualité. Tu as assumé le rôle de deux parents avec dévouement et courage depuis le décès de papa, et tu as fait de notre foyer un

endroit chaleureux, aimant et rempli de bonheur. Tu as sacrifié tant de choses pour nous élever et nous donner les meilleures chances de réussite. Je suis conscient que mes accomplissements sont le reflet de ton soutien inébranlable. Je t'aime énormément. Ton amour inconditionnel restera à jamais gravé dans mon cœur. Maman, je suis fier d'être ton enfant.

À ma merveilleuse épouse Rania, je souhaite exprimer toute ma gratitude et mon amour infini envers toi. Tu es la personne qui illumine ma vie chaque jour et qui me soutient inconditionnellement dans tous les aspects de ma vie. Ta présence et ton soutien constant ont été une véritable bénédiction. Tu m'as encouragé à poursuivre mes rêves, tu m'as soutenu dans les moments difficiles et tu as partagé avec moi les joies les plus intenses. Je suis reconnaissant d'avoir une partenaire aussi aimante, attentionnée et dévouée à mes côtés. Merci d'être celle qui rend chaque jour meilleur et d'être le pilier solide de notre famille. Je t'aime profondément.

À ma chère fille Sydra, au moment où tu es venue au monde, ma vie a été remplie d'une joie indescriptible et d'un amour inconditionnel. Ta naissance pendant cette période de ma thèse a apporté une lueur d'espoir et une motivation supplémentaire pour poursuivre mes recherches et accomplir mes objectifs. Tu es une merveilleuse source d'inspiration et chaque jour, je suis émerveillé par ta présence, ton sourire et ta douce personnalité. Je promets de toujours être là pour toi, de te soutenir dans toutes tes aspirations et de t'encourager à suivre tes propres rêves. Que ta vie soit remplie de bonheur, de réussite et de moments précieux. Je t'aime de tout mon cœur.

À mes chères sœurs Zineb et Asma, vous êtes les complices de mes souvenirs d'enfance et les confidentes de mes joies et peines. Vous avez toujours été là pour moi, m'encourageant dans mes études et mes projets, me rappelant mes forces et me soutenant dans les moments difficiles. Je vous aime énormément.

Je n'oublie pas **mon oncle Mustapha**, qui a été comme un deuxième père pour moi, et **ma grand-mère Fetoum Djamila**, dont la sagesse et la bienveillance ont été un refuge constant. Malheureusement, mon oncle nous a quittés, mais son souvenir et son impact restent gravés dans mon cœur. Que Dieu accorde Sa miséricorde à mon oncle et apporte guérison et soulagement à ma grand-mère.

Ma belle-mère, Nassira, tu m'as toujours demandé si j'avais terminé mes travaux. Enfin, je peux te répondre. J'ai fini ! Je te remercie sincèrement pour tout ton soutien.

À mes amis que je considère comme mes frères : **Islam, Mohamed HB, Hichem, Nadji et Aziz**, vous avez été d'un soutien indéfectible. Merci d'avoir été là à chaque étape de ce parcours, de m'avoir soutenu, écouté et encouragé. Notre amitié est un trésor que je chérirai toujours.

Je remercie également **Mohamed Ait Haj Nani** pour ses précieux conseils et son soutien moral qui ont grandement contribué à l'accomplissement de cette thèse.

Enfin, je tiens à remercier toutes les personnes qui ont croisé mon chemin et partagé un moment de ma vie, même brièvement. Chacun d'entre vous a laissé une empreinte dans mon parcours, et je suis profondément reconnaissant pour ces rencontres et ces expériences. Si j'ai involontairement oublié de mentionner quelqu'un, veuillez accepter mes excuses et sachez que votre contribution à cette thèse n'a pas été moins appréciée.

En terminant ces remerciements, je réalise combien je suis chanceux d'avoir été entouré et soutenu par tant de personnes formidables. Cette thèse n'aurait pas été possible sans vous tous, et je vous suis infiniment reconnaissant pour votre soutien, votre confiance et votre amitié.

Table des matières

Table des figures

Liste des tableaux

Liste des acronymes

ANN	Artificial Neural Networks
API	Application Programming Interface
ARC	Abstract Representation for Control planes
AS	Autonomous System
ASN	Autonomous System Number
BGP	Border Gateway Protocol
CE	Cross Entropy
CE	Customer Edge
CNN	Convolutional Neural Networks
CTI	Country-level Transit Influence
DT	Decision Trees
E-LSR	Edge Label Switching Router
eBGP	External BGP
ECC	Edge-Conditioned Convolutional layers
EGP	Exterior Gateway Protocol
ES	Evolution Strategies
FAI	Fournisseur d'Accès Internet
FN	False Negative
FP	False Positive
GAT	Graph Attention Networks
GCN	Graph Convolutional Networks
GGNN	Gated Graph Neural Networks
GNN	Graph Neural Networks
IA	Intelligence Artificielle
iBGP	Internal BGP
IGP	Interior Gateway Protocol
IoV	Internet of Vehicles

IP Internet Protocol
IPv4 Internet Protocol verison 4
IPv6 Internet Protocol verison 6
IS-IS Intermediate System to Intermediate System
ISP Internet Service Provider
IXP Internet eXchange Points
LAN Local Area Network
LDP Label Distribution Protocol
LIB Label Information Base
LSP Label Switched Path
LSR Label Switching Router
LSTM Long Short-Term Memory
ML Machine Learning
MLP Multi-Layer Perceptron
MSE Mean Squared Error
MP-BGP Multiprotocol-BGP
MPLS Multiprotocol Label Switching
NAT Network Address Translation
NOC Network Operation Center
NoSQL Not-only SQL
OSI Open Systems Interconnection
OSPF Open Shortest Path First
P Provider
P2P Peer to Peer
PE Provider Edge
PoP Points of Presence
QoS Quality of Service
RCC The Router Configuration Checker
RD Route Distinguisher
ReLU Rectified Linear Unit
RF Random Forest
RNN Recurrent Neural Networks

RR	Route Reflector
RT	Route Target
SD-WAN	Software Defined Wide Area Network
SDCoR	Software Defined Cognitive Routing
SDN	Software Defined Networking
SQL	Structured Query Language
TN	True Negative
TP	True Positive
TTP	Template Text Parser
VPN	Virtual Private Networks
VRF	Virtual Routing and Forwarding
VRRP	Virtual Router Redundancy Protocol
WDM	Wavelength Division Multiplexing

Introduction générale

Cadre de la thèse

Cette thèse, intitulée « *Apprentissage automatique pour la détection et la localisation d'erreurs de configuration dans un réseau d'opérateur* », est réalisée dans le cadre d'une Convention Industrielle de Formation par la REcherche (CIFRE). La CIFRE est un dispositif français qui vise à renforcer la collaboration entre les institutions académiques et les entreprises en soutenant financièrement la réalisation de thèses doctorales sur des problématiques industrielles.

Cette thèse CIFRE est menée en partenariat avec l'entreprise *IMS Networks*, un opérateur réseaux, et le laboratoire *IRIT (Institut de Recherche en Informatique de Toulouse)*. Cette coopération permet d'aborder la problématique de vérification de configuration dans les réseaux d'opérateurs, tout en s'appuyant sur l'expérience et les besoins concrets de l'entreprise *IMS Networks* en matière de gestion et de maintenance des réseaux.

Contexte et problématique

La vérification de configuration dans le contexte des réseaux d'opérateurs est un enjeu crucial pour assurer une qualité de service optimale et garantir la satisfaction des clients. En particulier, la configuration du service de réseaux VPN BGP/MPLS de niveau 3 est complexe et exigeante. Ce service permet d'interconnecter les différents sites distants des clients, tout en assurant une isolation de flux. Les opérateurs réseaux doivent gérer et maintenir un grand nombre de routeurs et de commutateurs, chacun avec leur propre configuration, pour fournir une connectivité de bout-en-bout fiable et sécurisée à leurs clients. À *IMS Networks*, nous avons observé que les erreurs de configuration représentent une des causes principales qui entraînent des incidents, des dégradations de performances et des indisponibilités de services, impactant ainsi négativement l'expérience des clients.

Les méthodes classiques de vérification de configuration, telles que les vérifications manuelles et les méthodes basées sur des listes de règles et de contraintes, présentent plusieurs limites. Elles peuvent être coûteuses en termes de temps et de ressources,

et leur efficacité dépend souvent de l'expertise des ingénieurs réseaux. De plus, ces méthodes ont des difficultés à s'adapter aux évolutions des réseaux et aux besoins spécifiques des clients.

La problématique principale de cette thèse porte sur la vérification de configuration dans le contexte des réseaux d'opérateurs, en particulier la configuration du service de réseaux VPN BGP/MPLS. L'objectif est de proposer une nouvelle approche basée sur des méthodes d'apprentissage automatique pour améliorer la détection et la localisation d'erreurs de configuration, tout en surmontant les limites des méthodes classiques. En explorant les possibilités offertes par l'apprentissage automatique, cette thèse vise à contribuer à l'amélioration de la gestion des réseaux d'opérateurs et à la fiabilité des services de réseaux de type VPN.

Contributions

Au-delà du travail d'état de l'art effectué pour présenter les opérateurs de réseaux, les configurations nécessaires pour déployer leurs services, ainsi que les méthodes d'apprentissage automatique et leurs applications dans le domaine des réseaux de communication, les contributions de cette thèse ont été élaborées en deux parties distinctes :

— **Détection et localisation d'incidents provenant d'erreurs de configuration :** Dans un premier temps, nous avons formulé le problème d'apprentissage de manière à ce que les modèles d'apprentissage automatique puissent détecter les incidents de connectivité de bout-en-bout provenant d'erreurs de configuration dans une architecture de réseaux VPN BGP/MPLS, et de localiser les sites distants qui seraient indisponibles suite à ces incidents. En se basant sur les configurations des clients d'*IMS Networks*, nous avons réalisé une modélisation de données qui nous a permis de générer plusieurs jeux de données contenant à la fois des configurations valides et des configurations erronées. En utilisant ces jeux de données, nous avons pu entrainer et comparer les performances de trois modèles d'apprentissages différents : un modèle classique d'arbres de décision (DT), un modèle de méthodes d'ensemble (RF) et un modèle de réseaux de neurones (MLP).

— **Détection et localisation d'erreurs de configuration :** Dans la seconde partie de notre travail, nous avons appliqué une approche basée sur les réseaux de neurones de graphes pour détecter et localiser les erreurs de configuration dans une architecture de réseaux VPN BGP/MPLS. Le choix d'utiliser les réseaux de neurones de graphes est basé sur les conclusions de la première partie de nos contributions. Pour ce faire, nous avons proposé deux modèles distincts : un modèle dédié à la vérification de configuration sur la partie de routage inter-sites, et un modèle dédié à la vérification de configuration de routage sites-backbone. La combinai-

son de ces deux modèles nous permet de couvrir un plus grand nombre d'erreurs impactant la connectivité de bout-en-bout.

Organisation générale du manuscrit

Cette section présente la structure du manuscrit qui, après cette introduction, se compose de quatre (4) chapitres et d'une conclusion générale.

— **Chapitre 1 :** Nous présentons dans ce chapitre un état de l'art des réseaux d'opérateurs et des configurations nécessaires pour fournir les services réseaux aux clients. Nous abordons également le besoin de vérification de configuration, en expliquant la complexité des configurations dans le contexte des réseaux d'opérateurs et en présentant quelques travaux de recherche menés sur ce sujet.

— **Chapitre 2 :** Ce chapitre décrit les principes et les techniques de l'apprentissage automatique et de l'apprentissage profond en particulier. Il présente également un état de l'art de l'utilisation des méthodes d'apprentissage automatique pour résoudre des problématiques dans le domaine des réseaux de communication.

— **Chapitre 3 :** Dans ce chapitre, nous développons la première partie de nos contributions dans le cadre de cette thèse. Dans cette première partie, notre objectif consiste en l'application des méthodes d'apprentissage supervisé pour la détection et la localisation d'incidents provenant d'erreurs de configuration dans une architecture de réseaux VPN BGP/MPLS de niveau 3.

— **Chapitre 4 :** Suite aux conclusions du chapitre précèdent, nous proposons d'utiliser des réseaux de neurones de graphes pour détecter et localiser les erreurs de configuration. Ce chapitre présente les modèles développés et les résultats obtenus sur les jeux de données produits en se basant sur les configurations utilisées par *IMS Networks*. De plus, des éléments d'intégration de la méthode à l'environnement de production sont également proposés.

— **Conclusion générale :** Dans ce chapitre final, nous rappelons le contexte et la problématique de la thèse, nous présentons une synthèse de nos contributions, et nous exposons quelques pistes d'amélioration comme perspectives de notre travail.

Chapitre 1

Les configurations dans un réseau d'opérateur

1.1 Introduction

Dans ce chapitre, nous abordons les réseaux d'opérateurs et présentons les configurations des différents protocoles nécessaires au déploiement de leurs infrastructures et à la fourniture de services tels que l'accès Internet et les réseaux virtuels privés. Nous abordons également la complexité de ces configurations ainsi que la nécessité de les vérifier avant chaque nouveau déploiement ou mise à jour. Nous présentons quelques travaux de recherche menés sur ce sujet, puis nous concluons en expliquant la nécessité d'étudier une nouvelle approche.

1.2 Réseaux d'opérateurs

1.2.1 Définition

Un opérateur réseaux, aussi appelé Fournisseur d'Accès Internet (FAI) ou Internet Service Provider (ISP), est une entreprise qui fournit l'accès Internet à ses clients (particuliers ou entreprises) à travers son réseau *backbone*.

Le réseau *backbone* d'un opérateur est un réseau de transit constitué de plusieurs routeurs et de plusieurs commutateurs (*switchs*) hébergés dans des points de présences *« Points of Presence (PoP) »* appartenant à l'opérateur ou à un hébergeur tiers [1]. La connexion entre deux routeurs de deux PoPs différents se fait, généralement, par une liaison pair-à-pair *« Peer to Peer (P2P) »* sur un support de multiplexage en longueur d'onde *« Wavelength Division Multiplexing (WDM) »* ou de fibre optique. Les sites des clients sont raccordés au PoP le plus proche de l'opérateur via un réseau d'accès fibre optique ou autre. Les liens du réseau d'accès peuvent être fournis par l'opérateur ou par un autre fournisseur tiers. La figure 1.1 illustre un exemple d'architecture d'un réseau d'opérateur.

Un opérateur fournit aussi d'autres services liés au domaine des réseaux. Un exemple de ces services est le déploiement et la gestion des réseaux virtuels privés *« Virtual Private Networks (VPN) »* pour interconnecter les différents sites d'un client.

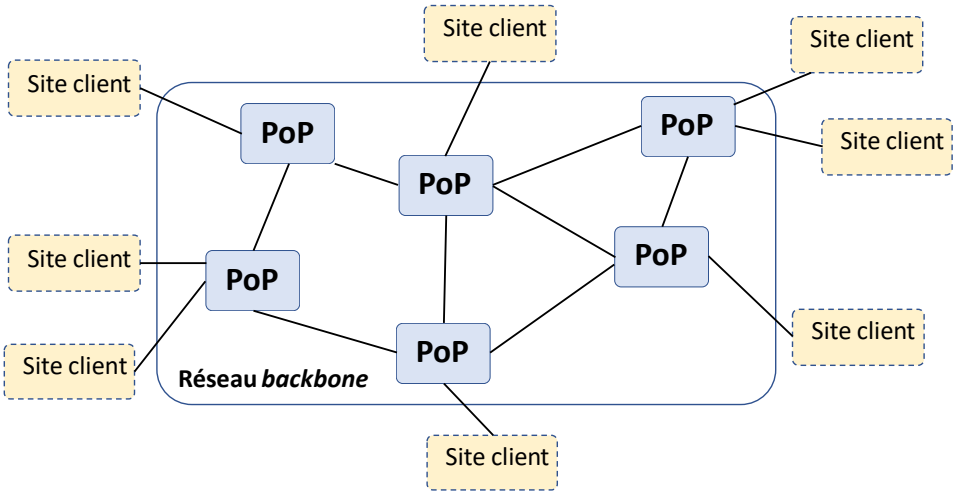


FIGURE 1.1 – Exemple d'architecture simplifiée d'un réseau d'opérateur.

1.2.2 Raccordement des opérateurs à Internet

Internet est un réseau mondial de réseaux constitué de l'interconnexion des réseaux d'opérateurs. Pour qu'un opérateur soit capable de vendre l'accès Internet, il doit être connecté, au moins, à l'un des opérateurs composant Internet. Pour faire cela, deux méthodes d'interconnexion sont possibles : le Transit et le Peering [2]. Ces deux méthodes sont mises en place en échangeant les routes entre les opérateurs avec le protocole de routage *« Border Gateway Protocol (BGP) »*.

1.2.2.1 La relation de Transit

Le Transit est un service vendu par un opérateur, appelé transitaire, à un autre opérateur. Il permet au trafic de l'opérateur client d'être acheminé vers Internet en passant par le réseau de l'opérateur transitaire [3]. La figure 1.2 montre par un schéma la relation de Transit entre deux opérateurs A et B.

1.2.2.2 La relation de Peering

Le Peering est un accord permettant d'échanger mutuellement les données entre deux opérateurs. En général, la quantité de trafic échangé est quasiment identique pour

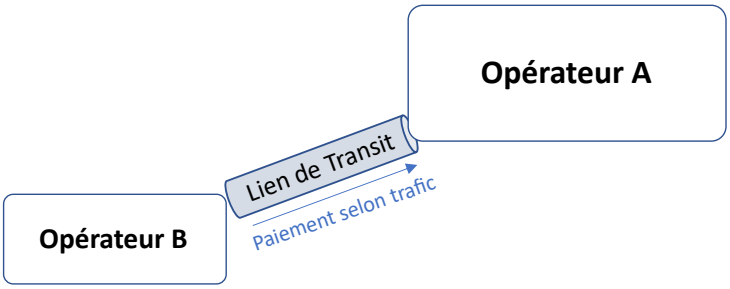


FIGURE 1.2 – La relation de Transit entre deux opérateurs.

les deux opérateurs [4]. Cet accord est gratuit, mais il peut devenir payant pour l'un des opérateurs s'il émet un trafic plus important que celui émis par l'autre opérateur. La figure 1.3 illustre une relation de Peering entre deux opérateurs B et C, qui sont connectés avec le même opérateur A de niveau supérieur pour Transit. Le trafic entre les deux opérateurs B et C passe par le lien de Peering, contrairement au reste de trafic qui doit passer par les liens de Transit.

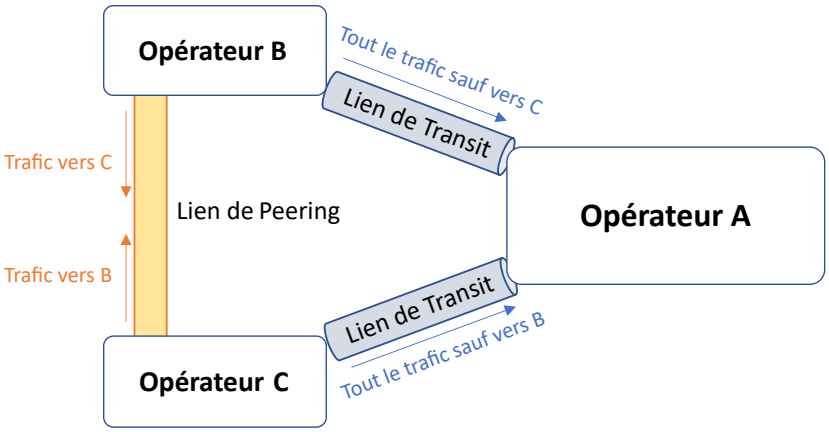


FIGURE 1.3 – La relation de Peering entre deux opérateurs.

1.2.3 Hiérarchie des opérateurs réseaux

Les relations d'interconnexion des réseaux d'opérateurs sont souvent résumées à l'aide d'une hiérarchie informelle, comprenant trois niveaux (ou tiers) d'opérateurs : les opérateurs de tier 1, les opérateurs de tier 2 et les opérateurs de tier 3 [2] [5]. La figure 1.4 représente un exemple des relations de Transit et de Peering entre les opérateurs des trois niveaux.

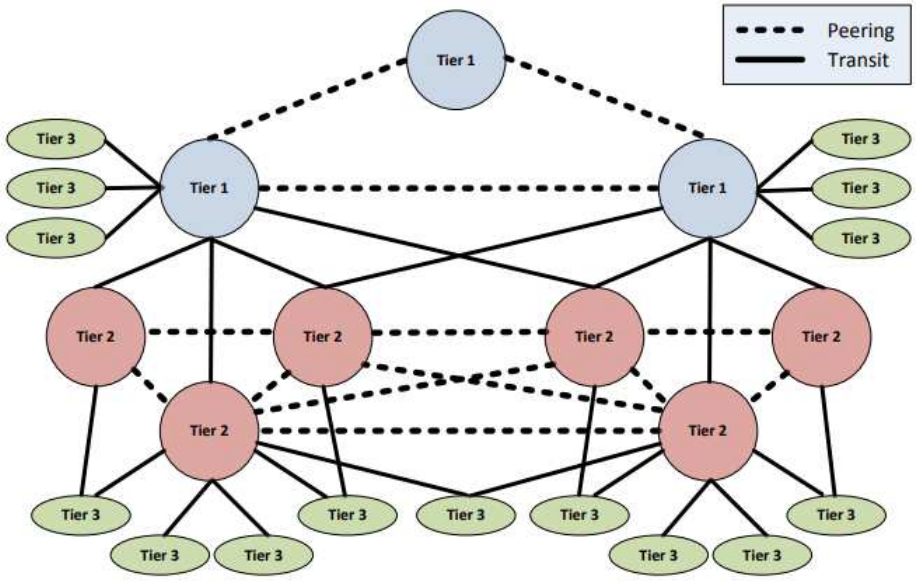


FIGURE 1.4 – Exemple des relations entre les opérateurs de différents niveaux [5].

1.2.3.1 Les opérateurs de tier 1

Ce sont des grands opérateurs internationaux qui n'achètent pas de liens de Transit auprès d'autres opérateurs. Ils ont des relations de Peering avec tous les autres opérateurs de tier 1 pour maintenir l'accessibilité mondiale.

1.2.3.2 Les opérateurs de tier 2

Ce sont des opérateurs ayant des clients de Transit et des liens de Peering avec d'autres opérateurs de même niveau. Ils achètent aussi des liens de Transit auprès des opérateurs de tier 1 pour avoir accès à Internet.

1.2.3.3 Les opérateurs de tier 3

Ce sont des petits opérateurs qui ne fournissent pas des liens de Transit, et en général, n'ont pas des liens de Peering. Pour avoir accès à Internet, ils achètent du Transit auprès des opérateurs des niveaux supérieurs.

1.2.4 Points d'échange Internet

Les points d'échange Internet, ou Internet eXchange Points (IXP), sont des infrastructures réseau permettant aux opérateurs d'établir des liens de Peering. En général, ils opèrent à la couche 2 pour faire de la commutation (*switching*) *Ethernet* [6]. Les opérateurs membres partagent un domaine de diffusion commun, et ils reçoivent une adresse IP unique par routeur à partir d'un bloc IP commun (tel qu'un /24) [5].

L'intérêt pour un opérateur de se connecter à un IXP est de pouvoir se connecter en Peering avec un maximum d'opérateurs à moindre coût et avec une complexité minimum. Un IXP permet une interconnexion logique avec n'importe quel opérateur membre, sans avoir besoin de liens physiques dédiés coûteux en termes d'argent et de temps. Le modèle de coût typique est que le membre paie pour le port et peut ensuite établir autant de sessions de peering qu'il le souhaite [5]. Un IXP devient plus attractif quand il a un nombre important d'opérateurs membres connectés à lui.

En plus des opérateurs réseaux, les grands fournisseurs de contenus (comme *Google*, *Facebook*, *Amazon* et *Netflix* ...) sont aussi présents sur les IXPs. Leur objectif est de se connecter en Peering avec le plus d'opérateurs possibles pour être plus proche des utilisateurs finaux d'Internet.

1.3 Configuration du routage Internet

Le routage de trafic sur Internet signifie le routage des paquets IP depuis leur source vers leur(s) destination(s) en passant par les différents opérateurs réseaux. Ce routage se fait essentiellement en utilisant BGP, qui est un protocole de routage extérieur *« Exterior Gateway Protocol (EGP) »* [7] permettant le routage inter systèmes autonomes *« Autonomous System (AS) »*. Un AS est défini comme un ensemble de réseaux et de routeurs, contrôlés et gérés par une seule entité administrative. Généralement, un AS correspond au réseau *backbone* d'un opérateur et est identifié par un seul numéro unique appelé *« Autonomous System Number (ASN) »*.

Dans cette section, nous définissons brièvement le protocole de routage BGP. Ensuite, nous présentons les configurations requises pour mettre en place un réseau *backbone* opérationnel et pour l'interconnecter avec d'autres ASs gérés par d'autres opérateurs ou qui représentent des sites de clients.

1.3.1 Le protocole de routage BGP

Le fonctionnement du protocole de routage BGP est défini dans le RFC-4271 [8]. Son rôle est d'échanger les informations de routage entre les routeurs voisins, y compris

les informations sur la liste des ASs. Les voisins BGP devraient être configurés manuellement et ne sont pas obligatoirement adjacents. Les informations échangées sont utilisées au niveau des routeurs pour définir les routes pour acheminer les paquets IP. Comme la majorité des protocoles de routages classiques, le protocole BGP ne prend en charge que le routage basé sur la destination, c'est-à-dire, les routeurs acheminent les paquets uniquement en fonction de l'adresse IP de destination.

Une route, dans le contexte de BGP, est définie comme une unité d'information qui associe un préfixe IP de destination aux attributs d'un chemin vers cette destination. Le nombre de ces attributs est variable, il existe, dans le RFC-4271, trois (3) attributs obligatoires et quatre (4) autres optionnels. Les attributs obligatoires sont :

— `ORIGIN` : il définit l'origine de la route. Il peut prendre trois (3) valeurs : la route provient d'un protocole de routage intérieur *« Interior Gateway Protocol (IGP) »*, elle provient d'un autre protocole de routage extérieur EGP ou « incomplète » si la route est apprise par d'autres moyens.

— `AS-PATH` : il contient la liste des ASs traversés par cette route.

— `NEXT-HOP` : il définit l'adresse IP du routeur qui devrait être utilisé comme le prochain saut pour cette route.

Les attributs optionnels sont : `MULTI-EXIT-DISC`, `LOCAL-PREF`, `ATOMIC-AGGRREGATE` et `AGGREGATOR`. Il existe aussi d'autres attributs optionnels qui sont définis dans d'autres RFC (comme par exemple, les attributs de communautés *« standard »* [9] et *« extended »*[10]), ou qui sont ajoutés dans des extensions de BGP (comme par exemple, dans le protocole *« Multiprotocol-BGP (MP-BGP) »* [11]).

La mise en place du routage BGP, pour un opérateur réseaux, se fait en deux étapes : le déploiement à l'intérieur du réseau *backbone* (l'AS), puis l'établissement du peering avec les ASs d'autres opérateurs.

1.3.1.1 Internal BGP (iBGP)

Il s'agit des sessions de peering BGP configurées entre les routeurs appartenant au même AS. Dans un *backbone* opérateur, nous configurons iBGP entre les différents routeurs de bordures, appelés routeurs *« Provider Edge (PE) »*, pour assurer le routage du trafic venant des sites des clients ou des autres ASs à l'intérieur du réseau *backbone*.

Le déploiement de l'iBGP nécessite la configuration préalable d'un protocole IGP (par exemple : *« Open Shortest Path First (OSPF) »* [12]) dans le réseau du *backbone* pour permettre le routage des paquets de contrôle.

1.3.1.2 External BGP (eBGP)

Il s'agit du peering BGP configuré entre deux routeurs appartenant à deux ASs différents. Il est utilisé dans le cas de routage entre deux opérateurs réseaux, ou le routage entre un site client et le réseau *backbone* de l'opérateur.

1.3.1.3 Multiprotocol-BGP (MP-BGP)

MP-BGP est une extension du protocole BGP. Il est décrit dans le *RFC*-4760 [11]. Il prend en charge le routage *unicast*, *multicast* et VPN pour les deux protocoles IPv4 et IPv6. Contrairement au protocole BGP, qui ne prend en charge que le routage *unicast* pour le protocole IPv4.

1.3.2 Configuration d'un *backbone* BGP/MPLS

En général, un opérateur réseaux configure son réseau *backbone* en se basant sur l'architecture *« Multiprotocol Label Switching (MPLS) »* [13][14]. Le principe de cette architecture est de combiner les concepts de routage de niveau 3 avec les concepts de commutation de niveau 2, afin d'améliorer les performances du réseau pour acheminer le trafic plus rapidement.

L'acheminement des paquets dans l'architecture MPLS est basé sur la commutaion de labels. À la réception d'un paquet, le routeur de bordure d'entrée, appelé *« Ingress node »*, consulte sa table de commutation afin d'attribuer un label au paquet selon l'adresse IP de destination, puis il l'achemine vers le prochain routeur dans le *backbone*, appelé *« Label Switching Router (LSR) »*. Losqu'un LSR reçoit un paquet MPLS, il change le label du paquet par un autre, puis il l'achemine vers le prochain routeur. Cela est fait à l'aide de la table *« Label Information Base (LIB) »* du routeur. Cette table est mise à jour en utilisant un protocole de distribution de label (par exemple, Label Distribution Protocol (LDP) [15]) et l'IGP du *backbone*. À la sortie du *backbone*, le routeur de sortie, appelé *« Egress node »*, enlève le label du paquet MPLS, puis il le transmet à la couche réseau pour faire le routage classique.

La figure 1.5 illustre l'acheminement des paquets par commutation de labels dans l'architecture MPLS. Les routeurs *« Ingress node »* et *« Egress node »* sont appelés aussi routeurs LSR de bordures *« Edge Label Switching Router (E-LSR) »*. Le chemin emprunté par le paquet, c'est-à-dire, la suite des routeurs par lesquels le paquet passe, est appelé *« Label Switched Path (LSP) »*.

Dans le *backbone* BGP/MPLS déployé par les opérateurs réseaux, les routeurs PE jouent le rôle des E-LSRs, et les routeurs *« Provider (P) »* se trouvant à l'intérieur

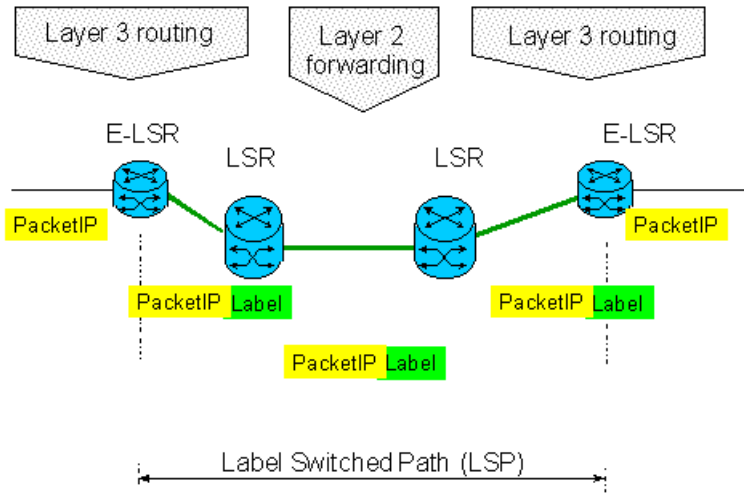


FIGURE 1.5 – La commutation de labels dans l'architecture MPLS [16].

du *backbone* jouent le rôle des LSRs. La figure 1.6, illustre un exemple simple d'un réseau *backbone* contenant deux routeurs PE, un routeur P et un routeur « *Route Reflector (RR)* ».

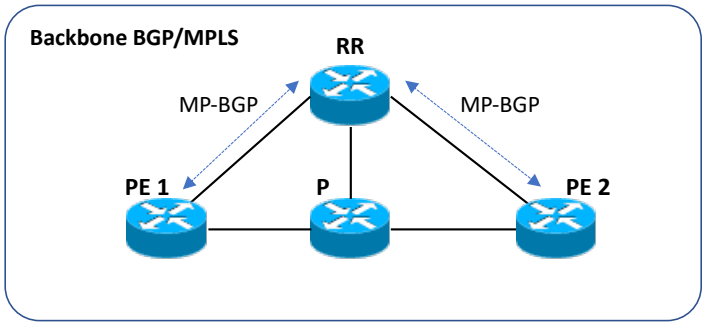


FIGURE 1.6 – Exemple simple d'un *backbone* BGP/MPLS.

Le déploiement d'un routeur RR est un moyen pour minimiser la complexité de configuration et de gestion des sessions de peering iBGP dans le *backbone*. Le principe est de configurer le peering BGP uniquement entre le routeur RR et chaque routeur PE. La configuration des sessions de peering entre les différents routeurs PE devient inutile. Le routeur RR regroupe toutes les informations de routage BGP via les sessions de peering, puis il les diffuse vers tous les routeurs PE. Sans le déploiement d'un routeur RR, la configuration des sessions de peering entre tous les routeurs PE en *full-mesh* est obligatoire pour diffuser toutes les informations de routage BGP dans le *backbone*.

La configuration d'un *backbone* BGP/MPLS opérationnel se fait en plusieurs étapes. Dans la première étape, nous commençons par les configurations initiales des routeurs. Nous configurons les adresses IP et les descriptions des interfaces physiques pour permettre la connectivité directe entre les routeurs. Aussi, une interface logique, appelée interface de *Loopback*, doit être configurée sur chaque routeur. L'adresse IP de cette interface est utilisée comme un identifiant du routeur pour les échanges de contrôle pour les différents protocoles.

Les étapes, qui suivent cette première étape, consistent à configurer les différents protocoles requis pour le bon fonctionnement du *backbone*. Ces étapes sont décrites ci-dessous. Pour cela, nous avons choisi d'illustrer des configurations de routeurs du constructeur « *Cisco* ». Les paramètres de configurations sont souvent similaires pour tous les constructeurs, il n'y a que la syntaxe qui diffère.

1.3.2.1 Configuration de l'IGP

L'objectif de l'IGP est de faire le routage entre les différents routeurs du *backbone* pour permettre la distribution des labels MPLS et l'établissement des sessions de peering BGP. Les protocoles IGP les plus utilisés par les opérateurs réseaux sont OSPF [12] et « *Intermediate System to Intermediate System (IS-IS)* » [17]. Nous illustrons ci-dessous la configuration d'OSPF.

La configuration basique d'OSPF est simple. Tout d'abord, nous devons l'activer en entrant la commande « `router OSPF <ID-PROCESSUS>` ». Puis, nous définissons l'identifiant du routeur et les adresses réseaux qui seront annoncées. Une configuration d'OSPF sur un routeur « *Cisco* » est comme suit :

```
router ospf 1
    router-id 10.0.0.1
    network 10.0.0.1 0.0.0.0 area 0
    network 10.10.12.0 0.0.0.3 area 0
    network 10.10.13.0 0.0.0.3 area 0
```

Le routeur en question a deux interfaces physiques connectées avec d'autres routeurs du *backbone*. Les deux dernières commandes « `network` » permettent d'annoncer les adresses réseaux de ces interfaces, et celle en premier permet d'annoncer l'adresse de l'interface de *Loopback* qui est utilisée aussi comme un identifiant du routeur. Les parametres de la commande « `network` » sont : l'adresse IP du réseau, le Masque Inverse (*Wildcard Mask*) et le numéro de la zone OSPF. Le numéro de la zone doit être identique sur tous les routeurs du *backbone*.

Une autre manière, pour annoncer l'adresse IP réseau d'une interface via OSPF, est de le faire depuis la configuration de l'interface comme suit :

```
interface GE 0/0
    ...
    ip ospf network point-to-point
    ip ospf 1 area 0
```

La commande « `ip ospf network point-to-point` » indique que cette interface est connectée en lien P2P, et la commande « `ip ospf 1 area 0` » permet d'annoncer l'adresse IP du réseau de cette interface via OSPF dans la zone 0.

1.3.2.2 Configuration de MPLS

En ce qui concerne MPLS, nous devons l'activer tout d'abord globalement sur le routeur et indiquer le protocole de distribution de labels utilisé (pour cette illustration, nous choisissons le protocole LDP), puis nous l'activons pour toutes les interfaces du *backbone*. Nous ne devons pas l'activer sur les interfaces qui sont connectées avec des sites clients ou avec d'autres ASs. La configuration requise est :

```
mpls ip
mpls label protocol ldp
!
interface GE 0/0
    ...
    mpls ip
```

1.3.2.3 Configuration d'iBGP

En interne, le protocole de routage BGP doit être configuré uniquement sur les routeurs PE et sur le routeur RR. Des sessions de peering sont configurées entre chaque routeur PE et le routeur RR.

Les configurations sur les routeurs PE sont parfaitement identiques. Nous activons BGP en précisant le numéro de l'AS du *backbone* avec la commande « `router BGP <ASN>` », puis nous configurons le seul routeur voisin qui est le RR en utilisant l'adresse IP de son interface *Loopback*. La configuration BGP des routeurs PE est comme suit :

```
router bgp 100
    router-id 10.0.0.1
    neighbor 10.0.0.10 remote-as 100
    neighbor 10.0.0.10 update-source Loopback0
    neighbor 10.0.0.10 next-hop-self
```

La commande « `neighbor <ADRESSE-IP> remote-as <ASN>` » permet de configurer un voisin BGP en indiquant le numéro de son AS. Dans le cas de peering entre un routeur PE et le routeur RR, les ASNs doivent être identiques et correspondent au numéro de l'AS du *backbone* de l'opérateur.

La commande « `neighbor <ADRESSE-IP> update-source Loopback0` » permet d'indiquer que les mises à jour de routage envoyées à ce voisin seront envoyées avec l'interface de *Loopback*. Nous n'utilisons pas l'une des interfaces physiques, car si un incident se produit sur l'interface utilisée, la session de peering sera interrompue.

La commande « `neighbor <ADRESSE-IP> next-hop-self` » permet de modifier l'attribut *« Next-Hop »* des routes envoyées vers ce voisin par son adresse IP utilisée comme router-id, c'est-à-dire, l'adresse IP de l'interface de *Loopback*. Cela signifie que pour aller vers les destinations des routes annoncées par ce routeur, il faut passer par ce routeur lui-même.

En ce qui concerne le routeur RR, nous activons le routage BGP, puis nous configurons tous les routeurs PE comme des voisins BGP en indiquant qu'ils sont des clients *route reflector*. La configuration BGP du routeur RR doit être comme ci-dessous :

```
router bgp 100
    router-id 10.0.0.10
    neighbor 10.0.0.1 remote-as 100
    neighbor 10.0.0.1 update-source Loopback0
    neighbor 10.0.0.1 route-reflector-client
    ...
    neighbor 10.0.0.x remote-as 100
    neighbor 10.0.0.x update-source Loopback0
    neighbor 10.0.0.x route-reflector-client
```

1.3.3 Configuration du peering BGP entre deux ASs

L'objectif du peering BGP entre deux ASs, c'est-à-dire eBGP, est d'interconnecter les réseaux de deux opérateurs ou de connecter un site client au réseau *backbone* d'un opérateur. Dans le deuxième cas, le réseau du site client est considéré comme un AS.

La configuration de l'eBGP est similaire dans les deux cas. Dans le cas d'interconnexion de deux réseaux d'opérateurs, nous devons configurer le voisinage eBGP entre deux routeurs PE des deux opérateurs. Et dans le cas de connexion d'un site client à un opérateur, nous devons configurer le voisinage eBGP entre le routeur de bordure de client, appelé routeur « *Customer Edge (CE)* », et un routeur PE de l'opérateur. Le protocole de routage BGP devrait être déjà activé sur les deux routeurs concernés avec le numéro de l'AS approprié. Nous configurons le voisin sur chaque routeur en indiquant son adresse IP et son numéro d'AS. La différence avec la configuration du peering dans le même AS est que les numéros d'AS des deux voisins sont différents, et l'adresse IP utilisée est celle de l'interface physique d'interconnexion. Nous n'utilisons pas l'adresse IP de l'interface de *Loopback*, car cette dernière n'est pas routée vers d'autres AS. Elle est routée uniquement à l'intérieur du *backbone*. La configuration du peering entre deux routeurs de deux ASs se fait comme ci-dessous, l'adresse de réseau d'interconnexion est 10.100.200.0/30 :

— **Routeur de l'AS 100 :**

```
router bgp 100
    ...
    neighbor 10.100.200.2 remote-as 200
```

— **Routeur de l'AS 200 :**

```
router bgp 200
    ...
    neighbor 10.100.200.1 remote-as 100
```

Généralement, cela est suffisant pour le routage entre deux ASs de deux opérateurs. Cette configuration permet aux routeurs PE de s'échanger toutes les routes, puis de les communiquer à l'intérieur des ASs via l'iBGP. Nous pouvons limiter l'échange de routes ou spécifier exactement quelles sont les routes à échanger en configurant des politiques de routage.

En ce qui concerne la connexion d'un site client au *backbone* de l'opérateur, plusieurs configurations sont possibles selon le service et les besoins de client. Si le client veut avoir toutes les routes d'Internet, la configuration sera similaire à celle du peering entre deux ASs. Souvent, c'est le cas des grands clients qui sont connectés à plusieurs opérateurs. Ils préfèrent avoir toutes les routes pour que leurs routeurs CE acheminent les paquets via l'opérateur qui annonce la meilleur route vers la destination concernée. Un client de ce type peut annoncer sur son routeur CE via BGP les routes vers une ou plusieurs adresses IP publiques. Ces adresses IP publiques peuvent être utilisées pour des mécanismes de

translation d'adresses IP (« *Network Address Translation (NAT)* ») pour pouvoir se connecter à Internet, ou elles peuvent être attribuées à des serveurs se trouvant dans le réseau du site client et accessibles via Internet. L'annonce de routes via BGP se fait par la commande « `network <ADRESSE-IP> <MASQUE>` » comme suit :

```
router bgp 200
    ...
    network 172.160.10.0 255.255.255.0
    network 172.160.11.9 255.255.255.255
```

Si le client veut avoir sur son routeur CE qu'une route par défaut pour aller vers Internet, un routage statique pourrait être plus simple que le peering BGP ente le CE et le PE de l'opérateur. Dans ce cas, nous configurons sur le routeur CE une route statique par défaut vers le routeur PE, et sur le routeur PE une ou plusieurs routes statiques vers le routeur CE avec comme destination les adresses IP que le client veut annoncer. La commande qui permet de configurer une route statique est « `ip route <IP-DEST> <MASQUE-DEST> <IP-PROCHAIN-ROUTEUR>` ». Les configurations ci-dessous sont un simple exemple de ce routage statique. Sur le routeur PE, nous devons ajouter aussi la redistribution des routes statiques dans la configuration de BGP pour que ces routes soient annoncées sur Internet via BGP. Si nous ne voulons pas redistribuer toutes les routes statiques ou redistribuer que des routes spécifiques, nous pouvons le faire en configurant des politiques de routage.

— **Sur le routeur CE du client :**

```
ip route 0.0.0.0 0.0.0.0 10.100.200.1
```

— **Sur le routeur PE de l'opérateur :**

```
router bgp 100
    ...
    redistribute static
...
ip route 172.160.10.0 255.255.255.0 10.100.200.2
ip route 172.160.11.9 255.255.255.255 10.100.200.2
```

1.4 Configuration des réseaux virtuels privés de niveau 3

Le déploiement des réseaux virtuels privés (VPNs) est l'un des services les plus vendus par les opérateurs réseaux. Un VPN permet d'interconnecter plusieurs sites distants d'un client [18]. Lorsque un opérateur fournit ce service, il permet à ses clients d'interconnecter leurs sites à travers son réseau *backbone* tout en assurant l'isolation des flux et la qualité de service *« Quality of Service (QoS) »*.

Les VPNs peuvent être de niveau 2 ou de niveau 3 du modèle *« Open Systems Interconnection (OSI) »* [19]. Les VPNs de niveau 2, ou de la couche liaison, permettent de créer des tunnels de niveau 2 entre les sites distants. D'un point de vu utilisateur, le réseau *backbone* de l'opérateur joue le rôle d'un switch virtuel, et les différents sites distants sont considérés comme étant interconnectés dans le même réseau local *« Local Area Network (LAN) »*. Contrairement aux VPNs de niveau 3, ou de la couche réseau, qui ont pour objectif de créer une interconnectivité IP entre les différents sites distants, où chaque site possède son réseau LAN. La figure 1.7 illustre la différence entre les deux types de VPNs.

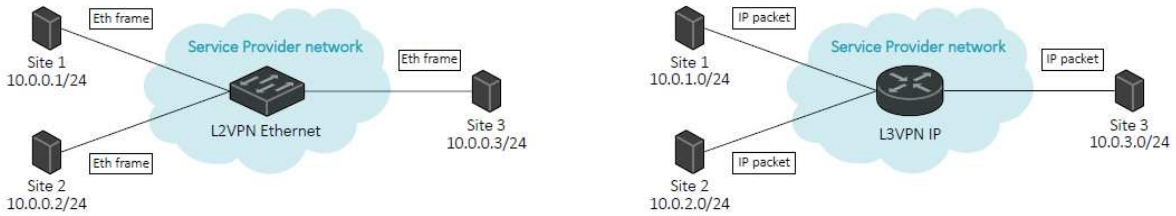


FIGURE 1.7 – La différence entre un VPN de niveau 2 et un VPN de niveau 3 [20].

Dans cette section, nous illustrons et nous expliquons les configurations pour mettre en place des réseaux VPNs de niveau 3 sur un *backbone* BGP/MPLS (toujours en utilisant la syntaxe du constructeur *Cisco*). Nous avons choisi de nous focaliser sur les réseaux VPNs de niveau 3 car ils sont complexes à configurer, et aussi, ils sont les plus présents chez les opérateurs réseaux.

L'architecture des réseaux VPNs de niveau 3 sur un *backbone* BGP/MPLS est définie dans le RFC 4364 [21]. La figure 1.8 montre un exemple simple de déploiement de cette architecture, avec trois (3) clients ayant chacun deux (2) ou trois (3) sites interconnectés à travers un *backbone* contenant trois routeurs PE.

Nous considérons que la configuration du réseau *backbone BGP/MPLS* est déjà réalisée. Pour la mise en oeuvre de ce type de topologie, il faut commencer par activer le routage VPN entre les voisins iBGP dans le *backbone*. C'est-à-dire, nous l'activons entre le routeur RR et chaque routeur PE, si nous utilisons un RR. Sinon, nous l'activons entre tous les routeurs PE. Cela permet aux routeurs PE de s'échanger les routes de

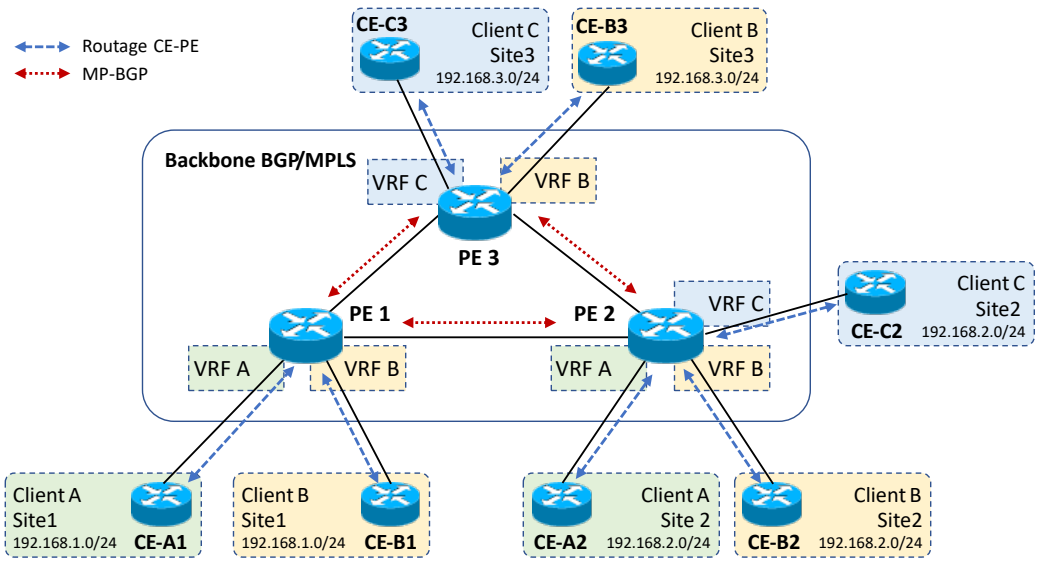


FIGURE 1.8 – Exemple simple d'une architecture VPN BGP/MPLS de niveau 3.

VPNs. Ensuite, nous devons créer une table de routage virtuelle *« Virtual Routing and Forwarding (VRF) »* par client sur chaque routeur PE. Ces tables VRFs sont utilisées pour router le trafic de chaque VPN séparément. À la fin, nous configurons le routage entre les routeurs PE et les routeurs CEs de chaque site. Pour cela, le routage statique est le plus simple à configurer, mais dans le cas où le site du client dispose d'un routeur de redondance, il est préférable de configurer un protocole de routage dynamique (généralement eBGP).

1.4.1 Configuration du routage VPN dans le *backbone*

L'activation du routage VPN via BGP avec un routeur voisin se fait dans le mode « `address-family vpnv4` » de BGP comme suit :

```
router bgp 100
    ...
    address-family vpnv4
        neighbor 10.0.0.10 activate
        neighbor 10.0.0.10 send-community both
```

La commande « `neighbor <ADRESSE-IP> send-community both` » permet de s'échanger les communautés « *standard* » et « *extended* » entre les routeurs voisins. Ces communautés sont des attributs de routes BGP optionnels. Leur utilité dans le cas de routage VPN dans un *backbone* BGP/MPLS est décrite ci-dessous.

1.4.2 Configuration des VRFs

Pour créer une VRF sur un routeur PE, il faut configurer trois paramètres essentiels : i) Le *« Route Distinguisher (RD) »*, ii) un *« Route Target (RT) »* pour l'exportation des routes de la VRF et iii) un RT pour l'importation des routes dans la VRF. Le RD est un identifiant unique pour chaque VRF. Il permet d'identifier le domaine de routage virtuel de chaque VPN dans le réseau *backbone* afin d'assurer l'isolation des flux. En ce qui concerne les RTs, ils sont utilisés comme attributs (communautés) dans les routes de VPNs échangées via BGP entre les routeurs PE dans le réseau *backbone*. Le routeur PE associe toutes les routes, apprises via un site client, au RT utilisé pour l'exportation des routes de la VRF concernée. Ensuite, il les communiquent aux autres routeurs PE du réseau *backbone*. Lorsqu'une route ayant un RT est reçue par un routeur PE, elle est éligible pour être utilisée dans les VRFs de ce routeur ayant ce RT pour l'importation de routes. La configuration ci-dessous est un exemple de configuration d'une VRF sur un routeur PE.

```
ip vrf CLIENT
    rd 100:1
    route-target export 100:1
    route-target import 100:1
```

En général, nous définissons la même valeur pour les trois (3) paramètres. Cette configuration est la plus simple. Elle est utilisée pour les réseaux VPN qui n'ont pas de contraintes de routage entre leurs sites distants, c'est-à-dire, tous les sites peuvent communiquer entre eux sans aucune contrainte. Ce type de réseaux VPN est le plus présent dans les réseaux d'opérateurs et est appelé « *Full-mesh* ».

Cependant, il existe d'autres configurations pour des besoins particuliers des clients. Pour répondre à cela, nous pouvons configurer plusieurs RTs pour l'importation ou l'exportation des routes. Nous pouvons aussi configurer des politiques de routage pour importer ou associer les routes à des RTs d'exportation selon les adresses IP de destination. Un bon exemple à cela est les réseaux VPN de type « *Hub-and-spoke* ». Dans un réseau VPN adoptant une architecture *Hub-and-Spoke*, les sites distants (appelés « *Spokes* ») ne peuvent communiquer qu'avec un seul site appelé « *Hub* ». Pour mettre en place cette architecture, il existe plusieurs solutions. À *IMS Networks*, sur chacun des routeurs PE connectés aux sites *Spokes*, nous attribuons deux politiques de routage à la VRF appartenant au VPN *Hub-and-Spoke*, permettant l'importation et l'exportation de routes VPN respectivement. La politique de routage d'importation est utilisée pour importer uniquement les routes du site *Hub*. Cela est réalisé en vérifiant l'adresse IP de destination et le RT associés aux routes reçues via MP-BGP. Les valeurs de ces deux paramètres doivent correspondre aux valeurs définies dans la politique de routage.

En ce qui concerne la politique de routage d'exportation, elle est utilisée pour exporter les routes des sites *Spokes* depuis la VRF, afin qu'elles soient communiquées via MP-BGP au routeur RR. Enfin, la configuration de la VRF sur le routeur PE connecté au site *Hub* reste identique à celle d'un VPN *Full-mesh* classique. Nous illustrons ci-dessous la configuration des VRFs adoptée par *IMS Networks* pour les réseaux VPN *Hub-and-Spoke*.

— **La configuration sur un routeur PE connecté à un site *Spoke*** :

```
ip prefix-list hub_prefix seq 10 permit 192.168.1.0/24
ip prefix-list spoke_prefix seq 10 permit 192.168.2.0/24
!
route-map import_map permit 10
    match ip address prefix_list hub_prefix
    match extcommunity rt 100:1
!
route-map export_map permit 10
    match ip address prefix_list spoke_prefix
    set extcommunity rt 100:1
!
ip vrf CLIENT_SPOKE
    rd 100:2
    import map import_map
    export map export_map
    route-target import 100:1
```

— **La configuration sur le routeur PE connecté au site *Hub*** :

```
ip vrf CLIENT_HUB
    rd 100:1
    route-target export 100:1
    route-target import 100:1
```

1.4.3 Configuration du routage CE-PE

Coté client, plusieurs architectures peuvent être adoptées pour connecter les sites distants au réseau *backbone* de l'opérateur. Le client peut se contenter d'un lien unique entre un routeur CE de son site et un routeur PE du réseau *backbone*. Cependant, il peut avoir besoin d'une redondance pour ne pas suspendre ses activités en cas d'incident réseau. Plusieurs architectures de redondance sont possibles. Sur la figure 1.9, nous illustrons trois sites clients avec trois architectures réseau différentes : un site d'un client A sans redondance et deux sites de deux clients B et C avec deux solutions de redondance différentes.

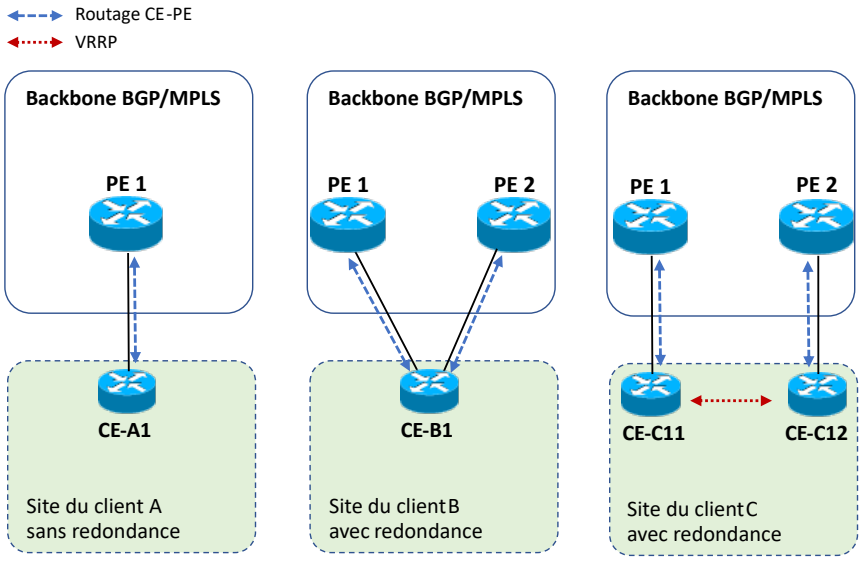


FIGURE 1.9 – Exemple d'architectures de sites clients VPN BGP/MPLS de niveau 3.

Pour configurer le routage pour le site du client A, nous devons configurer le protocole eBGP ou le routage statique entre le routeur CE et le routeur PE comme expliqué dans la sous-section 1.3.3, mais en indiquant sur le routeur PE la VRF du réseau VPN auquel le site appartient. Pour faire cela, il faut d'abord assigner l'interface du routeur PE à la bonne VRF avec la commande « `ip vrf forwarding <NOM-VRF>` ». Ensuite, s'il s'agit du routage eBGP, nous configurons le routeur CE comme voisin BGP dans le mode « `address-family ipv4 vrf <NOM-VRF>` ». Sinon, s'il s'agit du routage statique, nous configurons les routes statiques vers le site du client en précisant le nom de la VRF, puis nous configurons la redistribution de routage statique dans BGP dans le mode « `address-family ipv4 vrf <NOM-VRF>` ». Un exemple de cette configuration est illustré ci-dessous :

— **La configuration sur le routeur PE pour le routage statique :**

```
interface GE 0/1
    ip vrf forwarding client_A
    ip address 10.10.1.1 255.255.255.252
!
ip route vrf client_A 192.168.1.0 255.255.255.0 10.10.1.2
!
router bgp 100
    ...
    address-family ipv4 vrf client_A
        redistribute static
```

— **La configuration sur le routeur PE pour le routage eBGP :**

```
interface GE 0/1
    ip vrf forwarding client_A
    ip address 10.10.1.1 255.255.255.252
!
router bgp 100
    ...
    address-family ipv4 vrf client_A
        neighbor 10.10.1.2 remote as 200
```

Sur le routeur CE, la configuration du routage est exactement comme décrit dans la section 1.3.3.

Dans le cas où le routeur CE du site client est connecté par un second lien à un autre routeur PE du backbone pour des raisons de redondance, comme illustré dans la figure 1.9 pour le site du client *B*, le protocole de routage le plus adapté est le protocole eBGP. La configuration dans ce cas est similaire à la configuration du routage eBGP pour une architecture sans redondance. La seule différence est que nous devons configurer l'attribut de routes BGP *« Local Preference »* sur les routeurs PE. Cet attribut est utilisé pour sélectionner la route à prendre en compte pour acheminer le trafic du réseau *backbone* vers le site du client. Nous l'utilisons dans ce cas, car nous possédons deux routes possibles vers le site du client. La route sélectionnée est celle qui a la valeur de *Local Preference* la plus élevée.

Pour réaliser cette configuration, il faut commencer par configurer sur chacun des routeurs PE une politique de routage qui définit la valeur de la *Local Preference*. Puis, nous attribuons cette politique de routage aux routes apprises par le routeur voisin

eBGP, c'est-à-dire par le routeur CE concerné. Nous illustrons ci-dessous un exemple de configuration du routage sur les routeurs PE dans le cas du site du client B dans la figure 1.9. Nous considérons que le lien avec le routeur PE 1 est le lien primaire.

— **La configuration sur le routeur PE 1 :**

```
route-map client_B_LP120 permit 10
    set local-preference 120
!
router bgp 100
    ...
    address-family ipv4 vrf client_B
        neighbor 10.10.31.2 remote as 200
        neighbor 10.10.31.2 route-map client_B_LP120 in
```

— **La configuration sur le routeur PE 2 :**

```
route-map client_B_LP80 permit 10
    set local-preference 80
!
router bgp 100
    ...
    address-family ipv4 vrf client_B
        neighbor 10.10.32.2 remote as 200
        neighbor 10.10.32.2 route-map client_B_LP80 in
```

En ce qui concerne le troisième cas possible (le cas du site du client C) illustré dans la figure 1.9, le client opte pour une architecture de redondance complète. C'est-à-dire, en plus de la redondance de liens vers le backbone avec deux routeurs PE différents, le site du client possède un routeur CE dédie pour chaque lien.

Pour faire fonctionner cela, la configuration du routage reste identique à celle de l'architecture de redondance du site du client B, mais en prenant en compte l'existence de deux routeurs CE. Par contre, un protocole de redondance (par exemple : le « *Virtual Router Redundancy Protocol (VRRP)* » [22]) doit être configuré entre les deux routeurs CE. Le protocole VRRP permet de sélectionner automatiquement, parmi un ensemble de routeurs, le routeur à utiliser comme passerelle par défaut dans le réseau LAN. Cet ensemble de routeurs est considéré comme un routeur virtuel possédant une seule adresse IP. Les paquets à destination de cette adresse IP sont envoyés vers le routeur sélectionné, qui est appelé « *Master* ». Dans le cas où le routeur *Master* n'est pas joignable, un autre

routeur est sélectionné comme *Master*. La configuration du protocole VRRP entre deux routeurs est comme suit :

— **Sur le routeur 1 :**

```
interface GE 0/0
    ip address 192.168.2.1 255.255.255.0
    vrrp 1 ip 192.168.2.254
    vrrp 1 priority 200
```

— **Sur le routeur 2 :**

```
interface GE 0/0
    ip address 192.168.2.2 255.255.255.0
    vrrp 1 ip 192.168.2.254
    vrrp 1 priority 100
```

Dans cet exemple, le routeur *Master* est le routeur 1 et l'adresse du routeur virtuel est 192.168.2.254.

1.5 Complexité des configurations

Un opérateur réseaux gère les configurations de son réseau *backbone* ainsi que les configurations des routeurs CE de tous les sites de ses clients. Ces configurations sont mises à jour quotidiennement, ce qui les rend sujettes aux erreurs. Pour cette raison, un processus de vérification de configurations est nécessaire avant chaque modification pour assurer la disponibilité et la continuité des services fournis aux clients. La complexité de ce processus dépend de la complexité des configurations ainsi que de la fréquence et de la nature des modifications qui leur sont apportées.

Dans cette section, nous illustrons la relation entre la complexité des configurations dans un réseau d'opérateur et la taille et la dynamique du réseau. Puis, nous présentons le besoin de vérification des configurations réseaux, ainsi que certains travaux existants s'appuyant sur des méthodes à base de contraintes.

1.5.1 Taille et dynamique du réseau

La taille d'un réseau d'opérateur est mesurée par le nombre des routeurs composant son réseau *backbone*, le nombre de ses clients et le nombre de sites par client. Cette

taille est plus ou moins importante en fonction de l'opérateur. *IMS Networks*, qui est un opérateur de tier 2, possède dans son infrastructure une vingtaine de routeurs PE répartis dans toute la *France*. *IMS Networks* possède aussi une centaine de clients ayant des nombres de sites différents. Il y a des petits clients qui n'ont que quelques sites, et il y en a d'autres qui sont moyens ou grands et qui ont des dizaines de sites. Ce nombre important de clients et de sites rend les configurations plus complexes. Par exemple, pour configurer le service VPN pour un seul site d'un client, il faut ajouter sur le(s) routeur(s) PE concerné(s) plus d'une dizaine de lignes de configuration, sans compter la configuration de la VRF du VPN client, si elle n'est pas encore réalisée, et la configuration du/des routeur(s) CE du site. Notamment, à *IMS Networks*, un routeur PE, qui est connecté à 115 routeurs *CE*, contient plus de six mille (6000) lignes de configuration. Ces lignes incluent aussi les configurations de routage dans le *backbone*.

En complément de la taille du réseau, la particularité des besoins de chacun des clients complexifie aussi la gestion des configurations. Les opérateurs réseaux utilisent généralement des *templates* génériques pour configurer les nouveaux clients et/ou les nouveaux sites, mais ils rencontrent toujours des difficultés à cause des besoins différents des clients. Par exemple, un client pourrait demander plusieurs réseaux VPNs différents pour plusieurs ensembles de ses sites, mais en ayant besoin aussi d'interconnecter quelques sites appartenant à des VPNs différents. L'opérateur réseau a donc besoin de configurer des politiques de routage spécifiques pour répondre au besoin de ce client.

Un autre facteur important, qui rend la gestion et la vérification de configurations complexes, est la dynamique du réseau. C'est-à-dire, la fréquence et la nature des changements et modifications apportés sur le réseau. Pour ce facteur, nous pouvons distinguer deux parties de configurations différentes. La partie de configurations qui concerne le réseau *backbone*, et la partie de configurations qui concerne les services fournis aux clients. La partie du *backbone* inclut les configurations de routage IGP, de MPLS et d'iBGP. Cette partie est généralement stable. Elle n'est mise à jour que rarement pour des raisons de redimensionnement ou d'ajout de PoPs dans de nouvelles régions géographiques. En ce qui concerne la partie de configurations des services fournis aux clients, c'est la partie la plus dynamique. Car de nouveaux clients et/ou de nouveaux sites sont ajoutés fréquemment. De plus, les clients demandent quotidiennement des modifications pour un ou plusieurs de leurs sites existants. Par exemple, augmenter la bande passante ou annoncer une nouvelle adresse IP dans le réseau VPN sur un site. Aussi, les opérateurs réseaux sont parfois obligés de traiter des demandes de suppression de sites ou de résiliation de contrats. Ces demandes sont motivées généralement par des raisons de déménagement ou de changement d'opérateur. Tous ces ajouts, modifications ou suppressions de façon quotidienne génèrent souvent des erreurs de configurations impactant un ou plusieurs sites d'un ou de plusieurs clients. Ces erreurs de configurations sont parfois très difficiles, voire impossibles, à détecter par les méthodes classiques de

vérification. Un exemple concret de ce type d'erreurs de configurations a été détecté à *IMS Networks* en déployant un nouveau site VPN pour un client. Les ingénieurs du Network Operation Center (NOC) ont été contactés pour un incident réseau sur l'un des autres sites VPN de ce client. Le site n'était plus joignable par les autres sites du VPN. L'incident a duré plus d'une semaine pour être résolu. La raison de cet incident était la duplication de l'adresse IP du routeur CE. Elle a été utilisée par erreur sur le routeur CE du nouveau site. En conséquence, le trafic destiné au site concerné par l'incident était redirigé vers le nouveau site.

1.5.2 Problématique de la vérification des configurations

La vérification des configurations dans le domaine des réseaux permet de limiter les incidents et de garantir le comportement attendu du réseau. Dans le contexte des réseaux d'opérateurs, il s'agit de détecter les erreurs syntaxiques et sémantiques de configuration des différents protocoles, ainsi que les erreurs qui ne génèrent pas d'incidents mais qui ne permettent pas de répondre aux besoins des clients. Nous prenons, comme exemple à ce dernier type d'erreurs, la configuration d'une valeur erronée pour l'attribut *Local Preference* d'une route BGP vers l'un des sites clients. En conséquence, les paquets destinés à ce site seront acheminés mais via un autre lien que celui défini par les besoins du client. Cela pourrait dégrader la qualité de service si le lien emprunté par le trafic n'est pas suffisamment dimensionné.

Détecter les erreurs de configuration représente un défis majeur pour les opérateurs réseaux. L'étude présentée dans [23], illustre les méthodes et les outils de vérification et de test des réseaux. Cette étude comprend les méthodes formelles de vérification et de test du plan de données et du plan de contrôle. Pour notre cas, nous nous intéressons aux méthodes de vérification du plan de contrôle, c'est-à-dire, aux méthodes de vérification de configuration. En général, ces méthodes sont basées sur des règles et sur des contraintes qui doivent être mises à jour régulièrement. La figure 1.10 illustre le processus général utilisé pour vérifier les configurations d'un réseau.

Plusieurs travaux de recherche ont été menés dans ce contexte. Le vérificateur de configuration du routeur, ou *The Router Configuration Checker (RCC)* [24], est le premier outil permettant de détecter automatiquement les erreurs dans les configurations BGP dans les réseaux réels. Le RCC utilise l'analyse statique de configuration. Les outils utilisant cette méthode, y compris le RCC, sont généralement conçus pour des protocoles spécifiques et ne permettent pas de vérifier toute la configuration d'un réseau. *Batfish* [25] répond à cette limite en combinant la vérification du plan de contrôle avec la vérification du plan de données. En premier, *Batfish* génère un modèle logique du plan de contrôle à partir des configurations reçues en entrée. Puis, en utilisant ce modèle logique du plan de contrôle, il obtient le modèle du plan de données qui sera

analysé et vérifié. Cette dernière étape peut aider les opérateurs à comprendre les violations de propriétés et à corriger les erreurs de configuration. *Batfish* présente une limite majeure, le temps de génération du plan de données est très important. La vérification d'un simple fichier de configuration peut prendre jusqu'à deux heures. Aussi, le plan de données généré par *Batfish* ne prend pas en compte la commutation de labels (MPLS) et les modifications de paquets (par exemple, NAT).

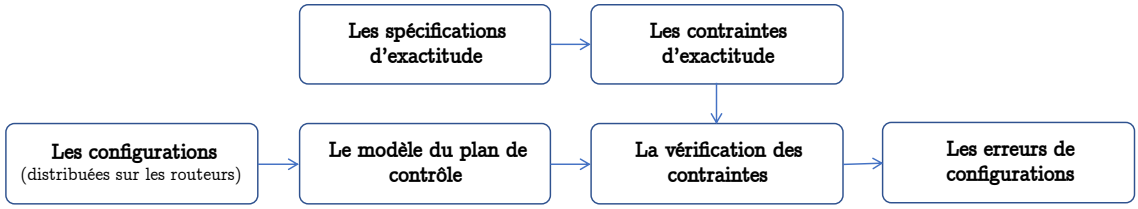


FIGURE 1.10 – Le processus de vérification du plan de contrôle dans les réseaux [23].

Pour éviter de passer par la génération du plan de données et pour optimiser le temps de vérification, dans [26], les auteurs ont proposé une représentation abstraite pour les plans de contrôle, ou *Abstract Representation for Control planes (ARC)*. Cette représentation est de haut-niveau et se concentre sur un ensemble limité de propriétés du protocole de routage. ARC ne peut pas vérifier les réseaux utilisant des propriétés complexes ou utilisant plusieurs protocoles de routage.

Minesweeper [27] fonctionne pour plusieurs protocoles de routage, pour plusieurs fonctionnalités et pour plusieurs topologies. Il traduit les fichiers de configurations en une formule logique qui représente les états stables du réseau après les interactions entre les différents protocoles de routage, tels que OSPF, BGP et le routage statique. Ensuite, il combine cette formule avec des contraintes décrivant la propriété recherchée. Si la formule combinée est satisfaisante, donc il existe un état stable du réseau dans lequel la propriété n'est pas vérifiée, c'est-à-dire, il existe une erreur. Sinon aucun état stable du réseau ne viole la propriété. Les propriétés peuvent être par exemple : la joignabilité des interfaces de management, l'interconnexion des sites clients dans un réseau VPN ou l'absence de trous noirs, etc. *Minesweeper*, comme tous les outils et les méthodes de vérification de configurations réseaux classiques, exige aux utilisateurs d'entrer manuellement des contraintes et des hypothèses sur l'environnement.

Dans la suite de ce manuscrit, nous allons nous intéresser à de la vérification de configurations réseaux, non pas à partir de méthodes formelles ou de tests, mais à partir d'approches orientés sur l'apprentissage automatique.

1.6 Conclusion

Dans ce chapitre, nous avons présenté la structure des réseaux d'opérateurs, leur fonctionnement ainsi que leur rôle. Également, nous avons illustré les configurations des différents protocoles permettant de déployer les infrastructures des opérateurs réseaux, d'interagir entre eux et de répondre aux besoins des clients en leur fournissant les services d'accès Internet et de réseaux virtuels privés (VPNs).

Un opérateur réseaux a besoin de configurer plusieurs protocoles sur les différents routeurs et commutateurs (*switchs*) composant son réseau *backbone*. Parmi ces protocoles, nous trouvons : le protocole de routage intérieur (par exemple : OSPF), le protocole de distribution de labels (LDP), le protocole de commutation par les labels (MPLS) et le protocole de routage extérieur (MP-BGP). Tous ces protocoles sont indispensables au bon fonctionnement du réseau. Une simple erreur sur un seul paramètre de configuration pourrait engendrer des incidents impactant la disponibilité des services fournis aux clients, d'où la nécessité de vérifier toutes les configurations avant chaque modification sur le réseau.

La vérification des configurations est une tâche très difficile à réaliser pour un réseau de taille importante et qui est soumis à des modifications multiples. Un opérateur réseaux doit gérer les configurations de son *backbone* ainsi que les configurations des routeurs CE des sites clients. Les configurations du routage au sein du *backbone* sont stables. Par contre, celles du routage entre les routeurs PE et les routeurs CE ne le sont pas, car de nouveaux sites clients sont ajoutés, modifiés ou supprimés quotidiennement.

En général, les opérateurs réseaux utilisent des méthodes de vérification classiques basées sur des règles et des contraintes qui doivent être mise à jour manuellement. La mise à jour manuelle de cette base de règles est très chronophage et ne garantit pas sa complétude par rapport à toutes les erreurs de configuration possibles. En plus de la difficulté liée à la complexité du réseau à vérifier, ces méthodes classiques présentent une autre limite importante. Les règles et les contraintes utilisées ne peuvent pas être personnalisées pour vérifier si les configurations répondent aux besoins particuliers de chaque client, car elles sont conçues pour vérifier la syntaxe et la structure générale d'une configuration, pas les spécificités propres à chaque client (comme par exemple des politiques de routage qui définissent une topologie de VPN particulière).

Dans ce contexte, l'objectif de notre travail de thèse est d'investiguer une nouvelle approche pour la vérification des configurations des réseaux d'opérateurs, en appliquant des méthodes d'apprentissage automatique. Nous focalisons notre recherche sur le service des réseaux virtuels privés (VPNs), car d'une part, il représente le service le plus répandu à *IMS Networks* et d'autre part, la majorité des incidents qui remontent du centre de supervision réseau (NOC) sont dus à des erreurs de configuration sur le service VPN de niveau 3.

Chapitre 2

Apprentissage automatique dans le domaine des réseaux de communication

2.1 Introduction

L'objectif de ce chapitre est de fournir une base solide pour comprendre les principes et les techniques de l'apprentissage automatique et de l'apprentissage profond en particulier, afin de mieux appréhender leur potentiel d'application dans le contexte des réseaux de communication. Dans la section 2.2, nous introduisons l'apprentissage automatique et ses différentes méthodes. Par la suite, dans la section 2.3, nous présentons l'apprentissage profond en se focalisant sur les réseaux de neurones artificiels et leurs variantes, notamment les réseaux de neurones *feed-forward*, les réseaux de neurones convolutifs et les réseaux de neurones de graphes. Finalement, dans la section 2.4, nous exposons un état de l'art sur l'application des méthodes d'apprentissage automatique dans le domaine des réseaux. Nous concluons le chapitre dans la section 2.5.

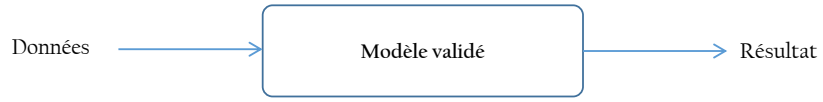
2.2 Aperçu de l'apprentissage automatique

2.2.1 Définition

L'apprentissage automatique, ou le *Machine Learning (ML)*, est une branche de l'Intelligence Artificielle (IA) permettant aux systèmes d'apprendre et de s'améliorer automatiquement par l'expérience [28] [29]. Il existe plusieurs définitions formelles dans la littérature. Dans [30], les auteurs ont cité trois de ces définitions. La première est celle d'*Arthur Samuel* [31]. Il a définit l'apprentissage automatique comme un domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés. La deuxième a été extraite du livre « *Machine Learning* » [32] de *Tom Mitchell*. Il a utilisé un lexique informatique pour expliquer que nous pouvons dire qu'un programme informatique apprend de l'expérience (E) en ce qui concerne une classe de tâches (T) et par rapport à une mesure de performance (P), si sa performance aux

tâches dans T, mesurée par P, s'améliore avec l'expérience E. La troisième définition citée dans [30] est celle de *Ethem Alpaydin* dans son livre « *Introduction to machine learning (3rd edition)* » [33]. Il a présenté l'apprentissage automatique comme le domaine de programmation des ordinateurs pour optimiser un critère de performance en utilisant des données d'exemples ou d'expériences passées. Ces définitions expliquent le même principe. Une tâche réalisée par un modèle d'apprentissage automatique n'est pas littéralement programmée. Le modèle d'apprentissage automatique apprend à réaliser cette tache en identifiant et en exploitant les relations entre les données reçues en entrée durant une première phase, appelée la phase d'entraînement. Après cette phase, le modèle entraîné doit être capable de fournir en sortie un résultat correct correspondant à de nouvelles données. La figure 2.1 illustre ces deux phases, c'est-à-dire la phase d'entraînement et la phase d'utilisation du modèle après sa validation.

(a) Phase d'entrainement



(b) Utilisation après validation

FIGURE 2.1 – Principe du fonctionnement de l'apprentissage automatique.

2.2.2 Méthodes

L'apprentissage automatique englobe une variété de méthodes et d'algorithmes pour la construction de modèles à partir de données. Ces méthodes peuvent être regroupées en trois catégories principales : i) les méthodes d'apprentissage supervisé [34], ii) les méthodes d'apprentissage non-supervisé [35] et iii) les méthodes d'apprentissage par renforcement [36]. Nous décrivons brièvement chacune de ces catégories ci-dessous.

2.2.2.1 Les méthodes d'apprentissage supervisé

Nous utilisons les méthodes d'apprentissage supervisé pour les problèmes de classification ou de prédiction. Un modèle d'apprentissage supervisé est entraîné avec des données étiquetées, c'est-à-dire, chaque donnée d'entraînement x est associée à une valeur de sortie y. Nous prenons un exemple de reconnaissance de fruits. Il s'agit d'un

problème de classification. Nous devons collecter plusieurs images de fruits qui serviront de données d'entraînement. Chaque image collectée doit être étiquetée avec le nom de la classe à laquelle elle appartient (le nom du fruit qu'elle représente). Cela permet au modèle d'apprentissage d'identifier les ressemblances entre les différentes images de chaque fruit et de s'ajuster pour pouvoir prédire correctement les noms de fruits présents sur de nouvelles images reçues après l'entraînement.

Les algorithmes d'apprentissage supervisé sont nombreux. L'algorithme d'arbres de décision, ou de *Decision Trees (DT)* [37], figure parmi les plus utilisés. Pendant l'entraînement, l'algorithme de DT crée un arbre permettant de classifier les données selon leurs propriétés. Chaque nœud dans l'arbre de décision représente une propriété dans une instance de données à classifier, et chaque branche représente une valeur que le nœud peut prendre. Les nœuds terminaux représentent les classes à prédire. Les instances de données sont classifiées à partir du nœud racine et triées en fonction de leurs valeurs de propriétés. La propriété sélectionnée pour le nœud racine est celle qui divise le mieux l'ensemble des données d'entraînement. Il existe différentes méthodes pour sélectionner une propriété qui divise le mieux l'ensemble des données, telles que l'index de Gini ou le gain d'information [38]. Nous illustrons dans la figure 2.2 un exemple d'arbre permettant de classifier différents fruits pour prédire leurs noms.

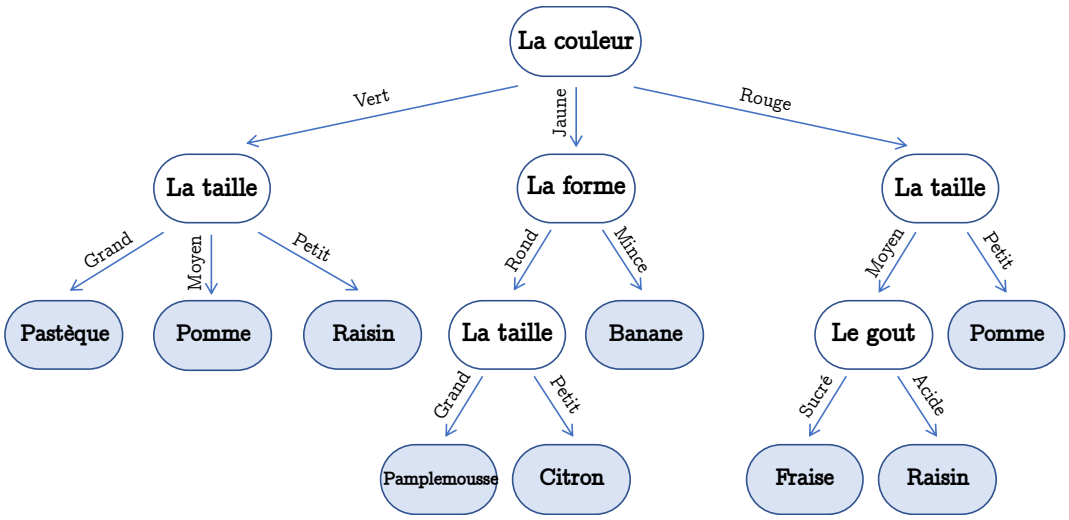


FIGURE 2.2 – Exemple d'un arbre de décision.

L'algorithme de DT est utilisé pour la classification de données dans différents domaines de calcul. Cela est dû à la flexibilité des arbres de décision à s'appliquer à un large éventail de problèmes. L'algorithme de DT est utilisé aussi pour des problèmes de prédiction des valeurs numériques (par exemple, prédire la valeur d'un bien immobilier). En apprentissage automatique, ce type de problèmes est appelé problème de régression.

Les méthodes basées sur les arbres de décision sont hautement évolutives. Dans [39],

P. Geurts et al. ont expliqué qu'en combinant l'algorithme de DT avec les méthodes d'ensemble (*Ensemble Methods*) [40], de meilleurs résultats de prédiction peuvent être obtenus. Les méthodes d'ensemble sont des techniques permettant de combiner plusieurs modèles d'apprentissage automatique avec pour objectif d'améliorer les performances de prédiction. *Random Forest (RF)* [41] fait partie des algorithmes de la famille des méthodes d'ensemble. Il combine plusieurs arbres de décision aléatoires puis agrège leurs prédictions en faisant la moyenne. RF a montré d'excellentes performances, nettement meilleures que l'algorithme de DT.

En plus des algorithmes classiques d'apprentissage supervisé tels que DT et RF, les réseaux de neurones peuvent également être utilisés pour réaliser l'apprentissage supervisé. Nous présentons plus de détails sur les réseaux de neurones dans la section 2.3.

2.2.2.2 Les méthodes d'apprentissage non-supervisé

Contrairement à l'apprentissage supervisé, les données d'entraînement pour l'apprentissage non-supervisé ne sont pas étiquetées. L'objectif de ce type d'apprentissage n'est pas de prédire une classe ou un résultat quantitatif, mais de regrouper les données reçues en entrée. En d'autres termes, un modèle d'apprentissage non-supervisé compare et évalue les données, puis forme des groupes (*clusters*) d'éléments similaires les uns aux autres.

L'algorithme de regroupement (de *clustering*) *K-means* [42] est considéré comme l'un des algorithmes d'apprentissage non-supervisé les plus performants. Les éléments de données, ou les points de données, à regrouper par cet algorithme doivent être représentés chacun par un vecteur de propriétés numériques. Un groupe (*cluster*) est représenté dans *K-means* par son centroïde, qui est calculé en faisant la moyenne pondérée de tous les éléments du groupe [43]. Au début de l'entraînement, k centroïdes sont selectionnés aléatoirement. Ensuite, les points de données sont assignés au centroïde le plus proche pour former k groupes différents. Aprés cela, les k centroïdes sont recalculés de nouveau. Les deux dernières opérations se répètent jusqu'à ce que les centroïdes deviennent stables et ne changent pas. La figure 2.3 illustre un exemple de regroupement de données en utilisant l'algorithme *K-means* [44].

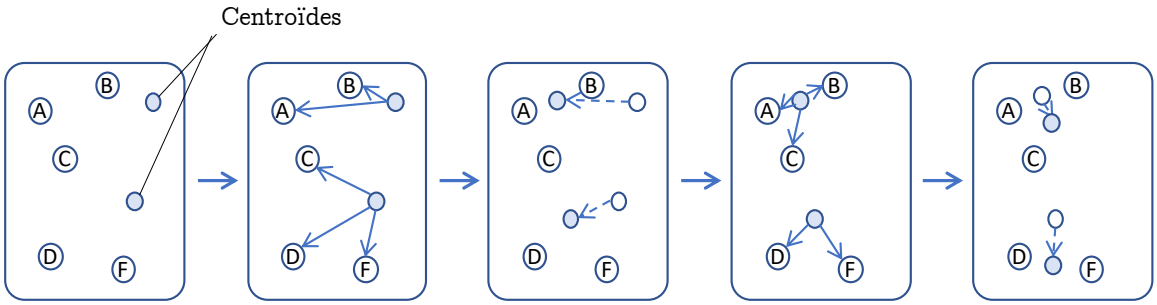


FIGURE 2.3 – Exemple de regroupement de données avec *K-means*.

2.2.2.3 Les méthodes d'apprentissage par renforcement

L'apprentissage par renforcement est différent des deux types d'apprentissage illustrés précédemment (l'apprentissage supervisé et l'apprentissage non-supervisé). Il consiste à apprendre automatiquement par interaction avec l'environnement, et non pas en réalisant un entraînement avec des données historiques. Ce type d'apprentissage est utilisé généralement lorsqu'il s'agit d'un problème de prise de décision. L'objectif est qu'un agent d'apprentissage apprend automatiquement à exécuter les meilleures actions. Après l'exécution de chaque action, l'agent d'apprentissage reçoit de l'environnement un retour lui permettant d'améliorer ses prochaines actions. Ce retour peut être positif (une récompense), comme il peut être négatif (une pénalité). La figure 2.4 illustre l'interaction entre l'agent d'apprentissage et l'environnement [36].

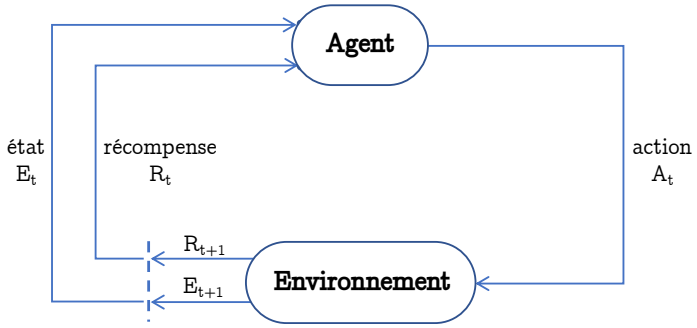


FIGURE 2.4 – L'interaction entre l'agent et l'environnement en apprentissage par renforcement.

2.3 Aperçu de l'apprentissage profond

2.3.1 Définition

L'apprentissage profond, ou en anglais *deep learning*, est un sous-domaine de l'apprentissage automatique qui se concentre sur l'utilisation de réseaux de neurones artificiels à plusieurs couches pour modéliser des structures de données complexes et hiérarchiques [45]. Il se distingue de l'apprentissage automatique classique en permettant d'apprendre directement les représentations des données à partir des données brutes, en utilisant des architectures de réseaux de neurones avec un grand nombre de couches cachées et une grande quantité de neurones par couche. Les réseaux de neurones profonds sont capables d'apprendre des représentations distribuées des données et de capturer des relations complexes et non linéaires entre les caractéristiques et les étiquettes (*labels*) des données [46]. Ils sont particulièrement adaptés pour traiter des données de grande dimension et permettent le transfert de connaissances entre différentes tâches ou domaines [47], [48].

2.3.2 Fonctionnement des réseaux de neurones artificiels

Les réseaux de neurones artificiels, ou en anglais *Artificial Neural Networks (ANN)*, sont des modèles computationnels inspirés du fonctionnement des neurones biologiques et des réseaux neuronaux du cerveau [45]. Ils sont composés de neurones artificiels, également appelés unités de traitement, organisés en différentes couches : une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie [46]. Les neurones d'une couche sont interconnectés avec ceux de la couche suivante par des poids, qui représentent la force des connexions synaptiques. La figure 2.5 illustre la représentation mathématique d'un neurone biologique.

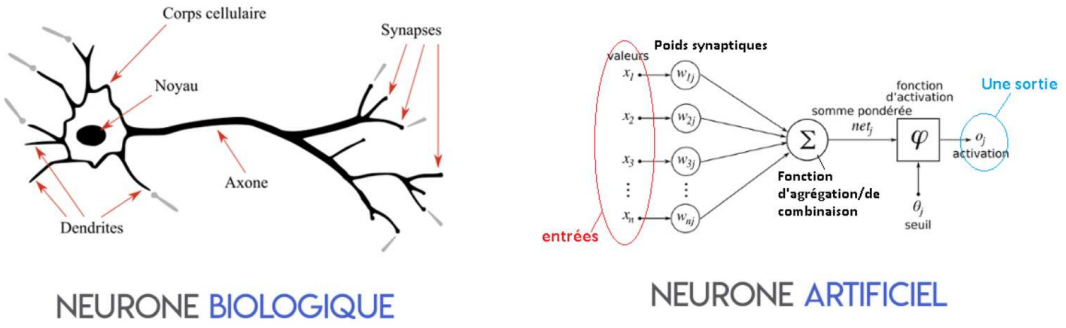


FIGURE 2.5 – Représentation mathématique/informatique d'un neurone biologique [49].

Le fonctionnement d'un ANN se déroule en deux étapes : la propagation vers l'avant (*feed-forward*) et la rétropropagation (*backpropagation*) [50]. Pendant la propagation vers l'avant, les données d'entrée sont traitées par les neurones des couches successives. Chaque neurone calcule une somme pondérée de ses entrées et applique une fonction d'activation non linéaire [51]. Les fonctions d'activation couramment utilisées incluent la tangente hyperbolique (tanh), la fonction sigmoïde et la fonction d'activation rectifiée (en angais, *Rectified Linear Unit (ReLU)*). Les fonctions d'activation introduisent des non-linéarités dans le réseau, permettant ainsi de modéliser des relations complexes entre les entrées et les sorties.

La rétropropagation est l'algorithme d'apprentissage utilisé pour ajuster les poids du réseau en minimisant l'erreur entre les sorties prédites et les sorties réelles. L'erreur est calculée à l'aide d'une fonction de coût, telle que l'erreur quadratique moyenne (en angais, *Mean Squared Error (MSE)*) pour les problèmes de régression ou l'entropie croisée (en angais, *Cross Entropy (CE)*) pour les problèmes de classification [52]. L'erreur est ensuite rétropropagée à travers le réseau pour mettre à jour les poids en utilisant la descente de gradient [53].

Pour améliorer la performance et la robustesse des ANN, diverses techniques de régularisation et d'ajustement des paramètres peuvent être utilisées. Parmi ces techniques, nous trouvons le *dropout* et la normalisation des lots (en angais, *batch normalization*) [54], [55]. Le dropout consiste à désactiver aléatoirement certains neurones pendant l'entraînement, ce qui réduit la dépendance du modèle à des neurones spécifiques et prévient le sur-apprentissage (en angais, *overfitting*). La normalisation des lots, quant à elle, consiste à normaliser les activations des neurones au sein d'un même lot, ce qui accélère l'entraînement et améliore la stabilité du réseau.

Un exemple de réseau de neurones est le perceptron multicouche (en angais, *Multi-Layer Perceptron (MLP)*), qui est un réseau de neurones *feed-forward* composé d'une couche d'entrée, d'une ou plusieurs couches cachées et d'une couche de sortie [56], [57]. Le MLP est utilisé pour résoudre des problèmes de classification et de régression. Chaque neurone d'une couche est connecté à tous les neurones de la couche précédente et de la couche suivante. L'entraînement du MLP se fait généralement par rétropropagation du gradient en combinaison avec la descente de gradient stochastique ou d'autres algorithmes d'optimisation [58]. Le MLP est capable de modéliser des relations non linéaires complexes entre les variables d'entrée et de sortie, grâce aux fonctions d'activation non linéaires appliquées aux neurones de chaque couche. La figure 2.6 illustre un exemple d'un réseau de neurones MLP simple contenant une couche d'entrée, une couche cachée et une couche de sortie.

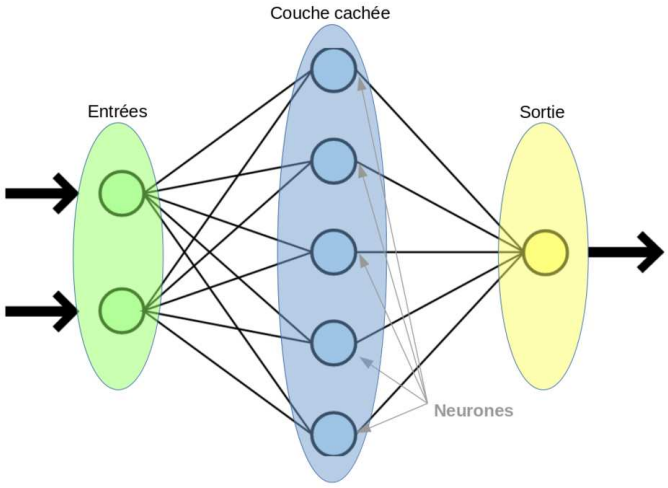


FIGURE 2.6 – Exemple d'un réseau de neurones MLP [59].

Bien que les MLP soient puissants et polyvalents, ils présentent certaines limitations. Par exemple, ils ne sont pas naturellement adaptés pour traiter des données structurées telles que des images ou des graphes. Pour ces types de données, d'autres architectures de réseaux de neurones, comme les réseaux de neurones convolutifs (en angais, *Convolutional Neural Networks (CNN)*) [60] pour les images et les réseaux de neurones de graphes (en angais, *Graph Neural Networks (GNN)*) [61] pour les graphes, ont été développées et ont montré des performances supérieures.

2.3.3 Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs, ou en anglais *Convolutional Neural Networks (CNN)*, sont une classe spécifique de réseaux de neurones profonds conçus pour traiter efficacement des données structurées, en particulier les images [62]. Contrairement aux réseaux de neurones classiques tels que les perceptrons multicouches (MLP), qui modélisent les données sous forme de vecteurs, les CNN conservent la structure spatiale des données en les représentant sous forme de matrices ou de tenseurs. Cette représentation permet aux CNN de tirer parti de la structure locale et de la composition hiérarchique des données, ce qui les rend particulièrement performants pour les tâches de reconnaissance d'images et de classification [63]. L'architecture d'un CNN est composée de plusieurs couches, dont les couches convolutives, les couches de pooling et les couches entièrement connectées (*fully-connected*) [60]. La figure 2.7 illustre un exemple d'un CNN.

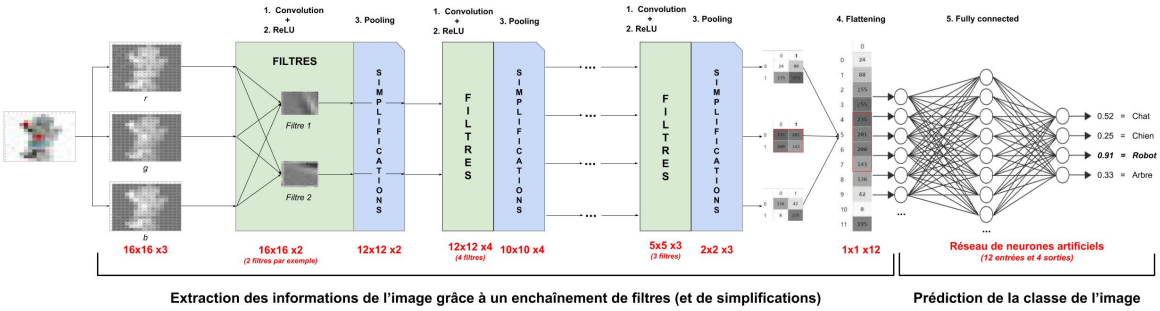


FIGURE 2.7 – Exemple d'un réseau de neurones convolutif (CNN) [64].

Les couches convolutives sont responsables de l'extraction des propriétés locales des données d'entrée, telles que les bords, les coins et les textures. Elles utilisent des filtres (ou noyaux) qui sont appliqués sur des régions locales de l'entrée, permettant ainsi de réduire le nombre de paramètres par rapport aux réseaux de neurones traditionnels [65]. Les filtres sont appris par le réseau au cours de l'entraînement pour détecter les propriétés les plus pertinentes pour la tâche en cours. Les couches convolutives peuvent être empilées pour former des modèles profonds, permettant ainsi d'apprendre des propriétés de plus en plus complexes à mesure que l'on se déplace dans le réseau.

Les couches de pooling, généralement placées entre les couches convolutives, ont pour objectif de réduire la dimension spatiale des représentations intermédiaires, ce qui permet de diminuer la complexité du modèle et d'éviter le sur-apprentissage [54]. Les méthodes de pooling les plus courantes sont le « *max pooling* », qui sélectionne la valeur maximale d'une région de l'entrée, et le « *average pooling* », qui calcule la moyenne des valeurs d'une région. Le pooling réduit également la sensibilité du modèle aux petites variations dans la position des propriétés, ce qui améliore la robustesse de la représentation.

Enfin, les couches entièrement connectées sont utilisées pour combiner les propriétés extraites par les couches convolutives et de pooling, et produire une prédiction pour la tâche en cours, comme la classification ou la régression. Généralement, une fonction d'activation *Softmax* est appliquée à la dernière couche entièrement connectée pour obtenir une distribution de probabilité sur les classes [52]. Dans certains cas, la couche entièrement connectée peut être remplacée par une couche globale de pooling « *global average pooling* », qui calcule la moyenne des activations sur chaque carte de propriétés avant de les passer à la couche de sortie.

Les CNN sont entraînés en utilisant la rétropropagation et des algorithmes d'optimisation similaires à ceux utilisés pour les réseaux de neurones traditionnels, tels que la descente de gradient stochastique ou l'optimisation basée sur la méthode du moment adaptatif (*Adam*) [58].

2.3.4 Réseaux de neurones de graphes

Les réseaux de neurones de graphes, ou en anglais (Graph Neural Networks (GNN)), sont une classe de modèles d'apprentissage profond conçus pour traiter des données représentées sous forme de graphes [61]. Les graphes sont des structures de données flexibles et expressives qui peuvent capturer des relations complexes entre les entités. Les GNN sont particulièrement utiles pour analyser des données qui présentent des dépendances structurelles et des relations non euclidiennes, telles que les réseaux de communication, réseaux sociaux, les molécules, les systèmes de transport et les systèmes de recommandation [66].

Les GNN sont basés sur l'idée de la propagation de messages, où chaque nœud du graphe communique avec ses voisins pour mettre à jour ses propriétés. Cette communication est réalisée par des couches de propagation, également appelées couches d'agrégation, qui sont empilées pour former un réseau profond [67].

Les couches d'agrégation sont définies par une fonction de mise à jour des nœuds, qui combine les propriétés des voisins et les propriétés du nœud courant pour produire une nouvelle représentation du nœud. La fonction d'agrégation est une opération sans état qui agit sur l'ensemble des voisins d'un nœud. Les fonctions d'agrégation courantes incluent la somme, la moyenne, le maximum, ou une combinaison linéaire pondérée des propriétés des voisins [51]. Après l'agrégation, les propriétés agrégées sont généralement passées à travers une fonction d'activation non linéaire, telle que la tangente hyperbolique (tanh) ou la fonction ReLU [68].

En termes d'équations, l'agrégation des propriétés d'un nœud v_i et de ses voisins $N(v_i)$ à une couche k est définie par la formule suivante :

$$h_{v_i}^{(k)} = AGGREGATE_k(\{h_u^{(k-1)} : u \in N(v_i)\})$$

où $h_{v_i}^{(k)}$ est la représentation du nœud v_i à la couche k, et $AGGREGATE_k$ est la fonction d'agrégation à la couche k.

Après l'agrégation, les propriétés sont combinées avec celles du nœud courant et passées à travers une fonction d'activation, comme suit :

$$h_{v_i}^{(k)} = ACTIVATE(COMBINE_k(h_{v_i}^{(k-1)}, h_{v_i}^{(k)}))$$

où $ACTIVATE$ est une fonction d'activation non linéaire et $COMBINE_k$ est une fonction qui combine les propriétés du nœud courant et de ses voisins.

Les fonctions de mise à jour sont apprises pendant l'entraînement pour capturer les propriétés pertinentes pour la tâche en cours. Il existe plusieurs variantes d'architectures GNN, parmi lesquelles on peut citer les réseaux de neurones convolutifs de graphes (en anglais, *Graph Convolutional Networks (GCN)*) [67], les réseaux de neurones de graphes de diffusion (en anglais, *Gated Graph Neural Networks (GGNN)*) [69] et les réseaux d'attention de graphes (en anglais, *Graph Attention Networks (GAT)*) [70]. Les GCN sont une généralisation des réseaux de neurones convolutifs pour les graphes, qui utilisent des opérations de convolution locales pour extraire des propriétés des voisins. Les GGNN ajoutent des mécanismes d'oubli et de mise à jour similaires aux cellules « *Long Short-Term Memory (LSTM)* » [71] pour mieux contrôler la propagation des informations dans le réseau. Les GAT utilisent des mécanismes d'attention pour pondérer l'importance des voisins lors de la mise à jour des propriétés du nœud.

Les GNN peuvent être utilisés pour diverses tâches, telles que la classification de nœuds, la classification de graphes, la prédiction de liens et la génération de graphes [72]. Ils peuvent également être combinés avec d'autres types de réseaux de neurones, tels que les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récurrents (en anglais, *Recurrent Neural Networks (RNN)*) [73], pour traiter des données multimodales ou séquentielles sur des graphes. La Figure 2.8 illustre un schéma d'architecture d'un GNN, dont la tâche d'apprentissage pourrait être la classification de graphes ou la classification de nœuds.

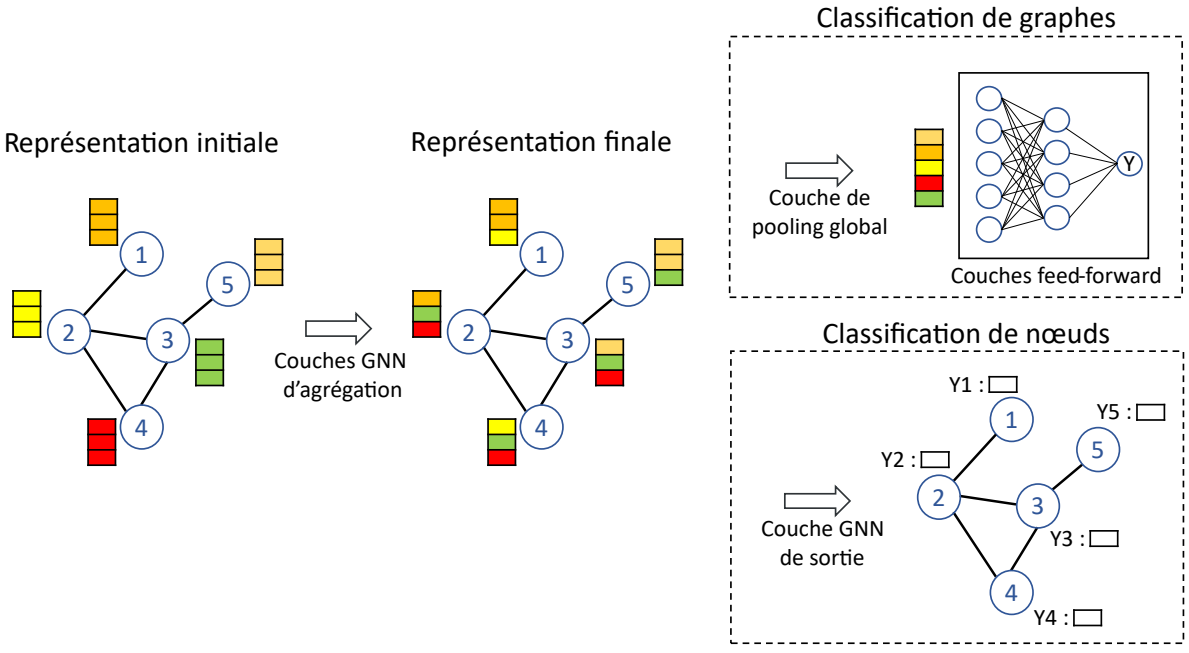


FIGURE 2.8 – Architecture d'un réseau de neurones de graphes (GNN).

De la même manière que les MLP et les CNN, les GNN sont généralement entraînés en utilisant la rétropropagation et des algorithmes d'optimisation, tels que la descente de gradient stochastique ou l'optimisation basée sur la méthode du moment adaptatif (*Adam*) [58].

2.4 Application des méthodes d'apprentissage automatique pour les réseaux

2.4.1 Processus général

Comme toute application de l'apprentissage automatique, l'application des méthodes d'apprentissage automatique pour résoudre des problèmes ou réaliser des tâches dans le domaine des réseaux de communication nécessite de suivre un processus composé de plusieurs étapes. Ces étapes sont indispensables et nécessaires pour garantir une performance optimale du modèle d'apprentissage. La figure 2.9 illustre le processus général de l'apprentissage automatique dans le domaine des réseaux [74]. Nous décrivons brièvement chacune des étapes de ce processus ci-dessous.

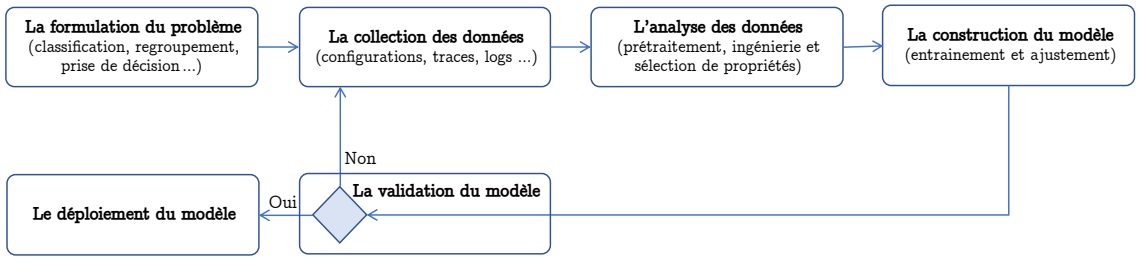


FIGURE 2.9 – Le processus général de l'apprentissage automatique pour les réseaux.

2.4.1.1 La formulation du problème

La formulation du problème est la première étape du processus général. Elle consiste à définir les besoins puis les formuler en un problème d'apprentissage automatique (prédiction de classes ou de valeurs numérique, regroupement de données ou prise de décisions). Cette étape est importante, car elle permet de définir le type de données à collecter et aussi la méthode et le modèle d'apprentissage à choisir.

2.4.1.2 La collection des données

Cette étape consiste à collecter le plus d'informations et de données possible sur le réseau. Ces données doivent être représentatives, car elles sont utilisées dans la prochaine étape pour créer les jeux de données d'entraînement et de tests. Nous identifions les données à collecter selon la formulation du problème. Elles peuvent être des données de configurations, des traces de trafic ou des logs, etc.

2.4.1.3 L'analyse des données

Au début de cette étape, les données collectées sont formatées en vecteurs de propriétés numériques. Pour cela, il est souvent nécessaire de réaliser un prétraitement sur une ou plusieurs propriétés. Par exemple, la normalisation ou l'encodage des données non-numériques. Ensuite, l'objectif est d'extraire les propriétés nécessaires et effectives de ces données pour répondre au problème d'apprentissage formulé. Cela est fait en analysant toutes les propriétés. En apprentissage automatique, cette opération est considérée comme un processus d'ingénierie de propriétés (*feature engineering*).

À la fin de cette étape, le jeu de données obtenu est divisé en deux. En général, 70% des données sont utilisées pour l'entraînement et les 30% restantes sont utilisées pour les tests.

2.4.1.4 La construction du modèle

L'étape de construction du modèle d'apprentissage est très importante. L'objectif est de choisir le modèle d'apprentissage (l'algorithme) approprié, puis de l'entraîner et de l'ajuster (par exemple, ajuster la probabilité de *Dropout* ou le nombre de neurones dans chaque couche pour les réseaux de neurones). Le choix du modèle est réalisé en fonction du problème formulé et en fonction du type et de la taille des données.

2.4.1.5 La validation du modèle

La validation est une étape indispensable dans le processus général de l'application de l'apprentissage automatique. Elle consiste à mesurer et à évaluer les performances du modèle entraîné en réalisant des tests sur le jeu de données dédié à cela. Les métriques d'évaluation varient en fonction de la catégorie du problème d'apprentissage. En fonction des résultats obtenus, nous jugeons si le modèle est suffisamment performant et s'il généralise bien, c'est-à-dire qu'il est capable d'obtenir de bons résultats sur de nouvelles données. Dans le cas contraire, nous devons reprendre les étapes précédentes pour essayer d'améliorer les performances du modèle d'apprentissage.

2.4.1.6 Le déploiement du modèle

Il s'agit de la dernière étape du processus. Elle consiste à déployer le modèle validé dans l'environnement cible, c'est-à-dire, le déployer afin qu'il soit prêt à être utilisé en production.

2.4.2 Domaines d'application

Les domaines d'application de l'apprentissage automatique dans le contexte des réseaux de communication sont nombreux. Raouf Boutaba et al [75] ont réalisé une étude détaillant ces domaines d'application ainsi que l'évolution et les opportunités de recherche dans ce contexte. Dans cette sous-section, nous présentons brièvement les domaines d'application concernant : la prédiction et la classification du trafic, le contrôle de congestion, la gestion d'incidents, la gestion du routage et la sécurité des réseaux. Le tableau 2.1 regroupe quelques exemples d'applications dans ces domaines proposées à travers des travaux de recherche.

Tableau 2.1 – Exemples d'application de l'apprentissage automatique pour les réseaux

Domaine d'application	**Objectif d'apprentissage**
Prédiction et classification du trafic	Prédiction de volume du trafic [76], [77]
	Classification du trafic [78]
Contrôle de congestion	Classification de la perte des paquets [79]
	Prédiction des délais de transmission [80]
Gestion d'incidents	Prédiction d'incidents [81]
	Détection d'incidents [82]
Gestion du routage	Calcul de routes [83], [84]
	Génération de configuration [85]
	Détection d'erreurs de configuration [86]
Sécurité des réseaux	Détection d'intrusions et d'anomalies [87]

2.4.2.1 La prédiction et la classification du trafic

La prédiction et la classification du trafic font partie des premières applications d'apprentissage automatique pour les réseaux de communication. La prédiction du trafic possède un rôle important dans l'administration de réseaux. Il s'agit d'une problématique de recherche fondamentale, car l'estimation précise du volume de trafic est utilisée pour résoudre plusieurs autres problèmes liés aux réseaux de communication. Par exemple, elle peut être utilisée pour le contrôle de congestion, pour l'allocation de ressources, pour l'optimisation de routage et même pour les application de haut-niveau

de diffusion en direct (*live streaming)* ou de jeux vidéos [74]. En ce qui concerne la classification du trafic, elle consiste à identifier le nom des applications et/ou des protocoles correspondant à chaque flux de trafic. L'identification de flux permet de réaliser beaucoup de tâches fondamentales, telles que la détection d'intrusions, la gestion de la qualité de service, le routage applicatif et d'autres [75]. Le chiffrement et l'encapsulation du trafic, ont rendu cette opération très complexe à réaliser par les stratégies et les techniques classiques. Ce qui a motivé les chercheurs à s'intéresser à l'apprentissage automatique en tant que nouvelle direction dans ce domaine, d'autant plus qu'il a montré des signes de réussite, tels que l'extraction de connaissances à partir du trafic chiffré et une gestion plus précise de la qualité de service [88].

2.4.2.2 Le contrôle de congestion

Le contrôle de congestion est une opération indispensable qui consiste à limiter les paquets acheminés dans les réseaux. Il s'agit d'un mécanisme très important dans les réseaux qui garantit un partage efficace et équitable des ressources entre les utilisateurs. Il permet aussi de garantir une qualité optimale des services fournis par les opérateurs réseaux. Récemment, différents travaux de recherche ont été menés avec pour objectif de démontrer l'utilité de concevoir de nouvelles approches basées sur l'apprentissage automatique pour le contrôle de congestion [89], [90]. Les problématiques les plus étudiées sont : la classification des pertes de paquets, la prédiction des délais de transmission, la prédiction de congestion et la mise à jour de la fenêtre de congestion.

2.4.2.3 La gestion d'incidents

La gestion d'incidents implique la prédiction et la détection de comportements anormaux de réseaux, leur isolation et la localisation de leur cause d'origine pour les corriger [75], [91]. Ce processus représente une tâche critique et indispensable pour les opérateurs réseaux, afin de maintenir la disponibilité de leurs services. De plus, cela est complexe à réaliser, car il nécessite des ressources considérables et des connaissances approfondies de l'architecture du réseau, ainsi que des équipements, des technologies et des services déployés. L'application des méthodes d'apprentissage automatique est de plus en plus nécessaire pour améliorer et optimiser ce processus dans les réseaux complexes [91].

2.4.2.4 La gestion du routage

Le processus de routage est fondamental pour les réseaux de communication. Son rôle est de sélectionner les chemins à utiliser pour acheminer le trafic de sa source vers sa/ses destination(s). Plusieurs problématiques dans ce domaine peuvent être traitées

à l'aide des approches d'apprentissage automatique, tels que le calcul des routes, la détection et la localisation d'erreurs de configuration ou la génération de configuration pour les protocoles de routages existants. L'application de l'apprentissage automatique dans ce domaine est abordée dans la section 2.4.3.

2.4.2.5 La sécurité des réseaux

La sécurité des réseaux consiste à protéger les réseaux des menaces pouvant compromettre leur disponibilité, ou entraîner des accès non autorisés ou des mauvaises utilisations des ressources. La majorité des travaux d'application de l'apprentissage automatique dans ce domaine sont focalisés sur la détection d'intrusions [75]. Cela implique la détection de toute forme de cyber-attaque (par exemple : l'hameçonnage et le *DDoS*). En général, la détection d'intrusions est formulée en un problème de classification.

2.4.3 Travaux dans le domaine du routage

Dans notre travail de thèse, nous nous intéressons à l'application de l'apprentissage automatique pour la détection et la localisation d'erreurs de configuration au sein des réseaux d'opérateurs. En particulier, la détection et la localisation d'erreurs de configuration de routage de bout en bout dans les réseaux VPNs de niveau 3. Récemment, différents travaux de recherche ont été menés avec pour objectif de proposer des approches basées sur de l'apprentissage automatique dans le domaine du routage. La prise de décision en utilisant l'apprentissage par renforcement est l'approche la plus dominante et la plus adoptée dans ces travaux. Cela est motivé essentiellement par le coût faible de calcul et de communication des algorithmes traditionnels d'apprentissage par renforcement, ainsi que par leur capacité à trouver une solution optimale et de s'adapter rapidement aux changements dans l'environnement [75]. Une classification de protocoles de routage basés sur l'apprentissage par renforcement a été proposée dans [92]. Cette classification est élaborée selon un ensemble de critères regroupés en trois catégories : i) Les critères liés au contexte d'utilisation, qui décrivent les applications ciblées et leurs caractéristiques et pré-requis ; ii) Les critères liés aux caractéristiques de conception, qui résument comment les auteurs ont conçu leurs protocoles pour les rendre efficaces et différents des autres protocoles ; et iii) Les critères liés aux performances, qui permettent d'évaluer qualitativement les protocoles. La figure 2.10 illustre la liste détaillée de ces critères.

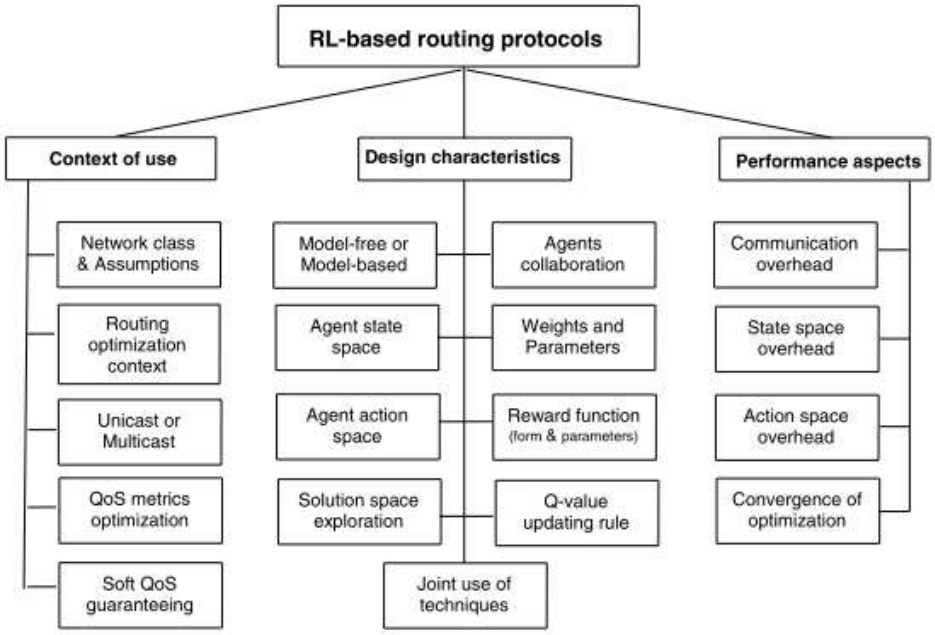


FIGURE 2.10 – Les critères de classification de protocoles de routage basés sur l'apprentissage par renforcement [92].

Parmi les travaux proposant des protocoles de routages basés sur l'apprentissage par renforcement, nous pouvons citer *Software Defined Cognitive Routing (SDCoR)* [83]. Il s'agit d'une solution intégrant la technologie de réseaux définis par le logiciel « *Software Defined Networking (SDN)* » [93] et l'apprentissage par renforcement, pour améliorer le processus de routage dans l'Internet de Véhicules « *Internet of Vehicles (IoV)* » [94]. Dans cette solution, le contrôleur SDN collecte les informations à propos des nœuds (les véhicules), puis adapte le routage par rapport aux conditions du réseau (c'est-à-dire, la mobilité et la charge du trafic). Contrairement aux autres protocoles de routage basés sur l'apprentissage par renforcement, l'agent d'apprentissage dans SDCoR (installé dans le contrôleur SDN) n'apprend pas à calculer les routes optimales, mais il apprend à sélectionner le protocole de routage le plus approprié parmi une liste de candidats. Par la suite, le contrôleur SDN calcule les tables de routages en se basant sur le protocole sélectionné, puis il les envoie aux commutateurs (*switchs*) SDN. Les commutateurs SDN utilisent ces tables de routage, pour router le trafic entre les véhicules, jusqu'à ce qu'elles soient à nouveau mises à jour par le contrôleur.

Les problématiques d'optimisation de routage ou de calcule de routes ne représentent pas les seules problématiques adressées dans la thématique de routage. D'autres problématiques, comme celles liées à la gestion de configurations, intéressent de plus en plus les chercheurs. Ces problématiques concernent plus particulièrement les opérateurs

réseaux. Cela est dû au fait qu'ils gèrent des configurations de routage très complexes et critiques. *DeepBGP* [85] est un exemple de ces travaux qui ciblent des problématiques de configuration de routage dans les réseaux d'opérateurs. Son objectif est de générer les configurations BGP à partir des besoins et de l'intention du haut niveau de l'opérateur réseaux. La solution proposée repose sur un modèle de réseaux de neurones de graphes. Les auteurs ont argumenté ce choix par le fait que les topologies réseaux sont mieux représentées par des graphes. La sortie de ce modèle de réseaux de neurones de graphes est utilisée dans une autre étape par un autre modèle de réseaux de neurones, nommé *read-out layer*, pour générer une matrice représentative des annonces de routes entre les différents routeurs. Cette matrice est transmise par la suite vers une unité de validation, nommée le validateur de configuration, qui calcule une valeur de retour (récompense) basée sur des mécanismes de validation déterministes. La valeur de retour, ou la récompense, est utilisé par un optimiseur de stratégies d'évolution « *Evolution Strategies (ES)* » [95] pour ajuster les paramètres des réseaux de neurones. L'utilisation de l'optimiseur d'ES permet de réduire significativement le temps de calcul, car il ne nécessite pas de faire la rétropropagation (*backpropagation*). Aussi, les ESs permettent d'obtenir un taux de convergence plus rapide et une parallélisation élevée par rapport à l'apprentissage par renforcement [96], [97]. Aprés le validateur de configuration, la dernière étape proposée dans *DeepBGP* est le rédacteur de configuration. Il convertit la matrice contenant les annonces de routes entre chaque paire de voisins BGP en une configuration valide spécifique au constructeur (*Cisco* ou *Huawei)*. La figure 2.11 illustre l'architecture complète de *DeepBGP*.

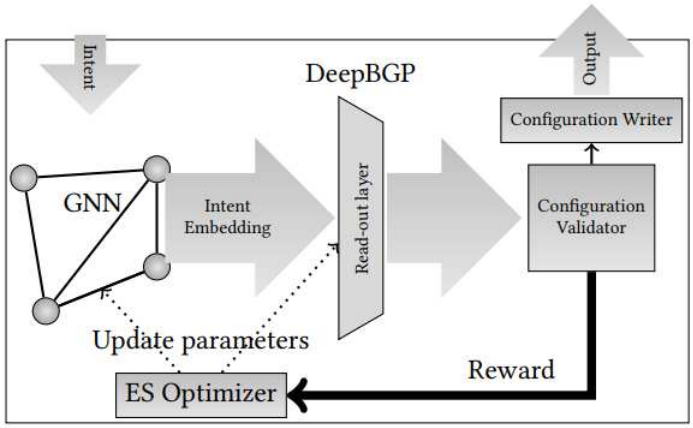


FIGURE 2.11 – L'architecture de *DeepBGP* [85].

En ce qui concerne notre problématique spécifique, c'est-à-dire, la détection et la localisation d'erreurs de configuration de routage, au meilleur de notre connaissance, il n'y a pas eu encore de travaux proposant d'appliquer les méthodes d'apprentissage automatique. Toutefois, nous pouvons citer *DeepMPLS* [86], qui traite une problématique similaire. Les auteurs proposent une nouvelle approche basée sur l'apprentissage

automatique pour accélérer l'analyse des propriétés MPLS, ainsi que pour suggérer les modifications à apporter dans le cas où une propriété n'est pas valide. En d'autres termes, l'objectif de *DeepMPLS* est de prédire la validité d'une propriété MPLS, puis de proposer des modifications de configuration en cas d'erreur. Cette solution est basée sur une architecture de réseaux de neurones de graphes. La prédiction de la validité des propriétés MPLS est formulée en un problème de classification de nœuds, et la génération de nouvelles configurations est formulée en un problème de prédiction de liens.

La modélisation des données de réseaux de communication sous forme de graphes a été largement adoptée ces dernière années dans les travaux de recherche concernant l'application de l'apprentissage automatique pour les réseaux [98]. Cela a motivé les auteurs de [99] et de [100] de proposer des *frameworks* facilitant cela. *IGNNITION* [99] est un *framework open source* proposé avec pour objectif de permettre le prototypage rapide des réseaux de neurones de graphes dans le domaine des réseaux. Il est basé sur une abstraction intuitive de haut niveau cachant la complexité des réseaux de neurones de graphes, mais tout en gardant la possibilité de créer de manière flexible des modèles personnalisés. Afin d'évaluer les performances de ce *framework*, les auteurs l'ont utilisé pour implémenter deux modèles de réseaux de neurones de graphes appliqués aux différents cas d'utilisation dans le domaine de routage. Les deux modèles implémentés sont *RouteNet* [101] et *GQNN* [102]. Les résultats obtenus ont montré que les modèles créés par *IGNNITION* sont équivalents en termes de précision et de performance à leurs implémentations natives. *IGNNITION* est destiné essentiellement aux ingénieurs réseaux, qui souvent n'ont pas l'expertise requise pour manipuler des algorithmes d'apprentissage automatique. Contrairement à cela, le *framework* proposé dans [100] est destiné au monde de la recherche. Il s'agit d'un cadre de référence facilitant l'utilisation des réseaux de neurones de graphes pour la recherche dans le domaine de routage Internet. Le pipeline méthodologique global adopté par les auteurs est illustré dans la figure 2.12. La contribution principale de ce travail consiste en la création d'un jeu de données de référence, en utilisant des données de routage Internet issues de différents sources (*CAIDA AS-rank* [103], *CAIDA AS-relationships* [104], *PeeringDB* [105], *AS hegemony* [106], *Country-level Transit Influence (CTI)* [107] et *ASdb* [108]). Ces sources de données sont des bases de données publiques qui n'incluent aucune donnée sensible ou privée (comme les données de configuration par exemple). Le jeu de données proposé pourrait représenter un gain de temps considérable pour les chercheurs, car il leur permet de se focaliser sur la conception de modèles de réseaux de neurones de graphes plutôt que sur la collecte et le traitement de données, ce qui est une tâche chronophage. Également, ce jeu de données pourrait servir comme un jeu de données commun permettant la comparaison de différents travaux. Cela représente une clé majeure pour faire progresser la recherche dans le domaine du routage Internet en appliquant des méthodes d'apprentissage automatique. Afin de démontrer l'efficacité des réseaux de

neurones de graphes avec le jeu de données proposé, les auteurs ont réalisé une série de tests avec : i) des modèles de réseaux de neurones de graphes (*GraphSAGE* [109], *GCN* [110] et *GAT* [111]) ; ii) d'autres modèles d'apprentissage automatique dédiés aux données en graphes (Node2vec [112] et BGP2vec [113]) ; et iii) un modèle d'apprentissage automatique traditionnel (RF [41]). Les tests réalisés traitent des problématiques de prédiction de liens et de classification de nœuds.

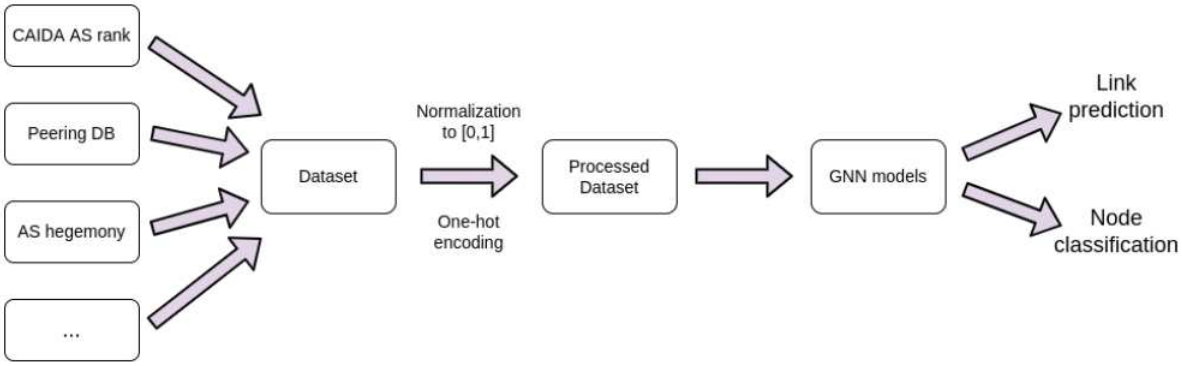


FIGURE 2.12 – Le pipeline méthodologique global adopté dans [100].

2.5 Conclusion

Dans ce chapitre, nous avons présenté un aperçu de l'apprentissage automatique et de son application dans le domaine des réseaux de communication. Nous avons commencé par introduire les concepts fondamentaux de l'apprentissage automatique, en abordant ses différentes méthodes, y compris les méthodes d'apprentissage supervisé, d'apprentissage non-supervisé et d'apprentissage par renforcement. Par la suite, nous avons défini et présenté l'apprentissage profond, en mettant l'accent sur les réseaux de neurones artificiels et leurs variantes, notamment les réseaux de neurones *feed-forward* (MLP), les réseaux de neurones convolutifs (CNN) et les réseaux de neurones de graphes (GNN).

Dans la troisième partie du chapitre, nous avons aborder l'application de l'apprentissage automatique dans le domaine des réseaux, en décrivant le processus général pour mettre en œuvre ces méthodes afin de résoudre les problématiques associées aux réseaux. Nous avons également mis en évidence des domaines d'application spécifiques, tels que la prédiction et la classification du trafic, le contrôle de congestion, la gestion d'incidents, la gestion du routage et la sécurité des réseaux. Enfin, nous avons présenté quelques travaux de recherche récents dans le domaine du routage.

Cet état de l'art nous a permis de souligner l'absence de travaux sur l'application de l'apprentissage automatique pour effectuer de la vérification de configuration du routage, et en particulier la vérification de configuration du routage avec de l'interconnexion de VPN BGP/MPLS. Néanmoins, la synthèse de ces informations nous permet de mieux comprendre les principales méthodes d'apprentissage automatique et la manière dont elles peuvent être utilisées pour résoudre d'autres problématiques liés aux réseaux de communication.

Chapitre 3

Détection et localisation d'incidents provenant d'erreurs de configuration

3.1 Introduction

Nous présentons dans ce chapitre la première partie de notre travail de thèse, dont l'objectif est de détecter et de localiser les incidents provenant d'erreurs de configuration dans une architecture de réseaux VPN BGP/MPLS de niveau 3 avec des méthodes d'apprentissage automatique supervisées. Pour atteindre cet objectif, nous avons établi une modélisation de données et comparé les performances de différents modèles d'apprentissage automatique pour déterminer la meilleure approche à adopter. L'objectif d'apprentissage est de détecter les incidents liés aux erreurs de configuration et de localiser les sites distants des clients VPN impactés par ces incidents. Nous avons réalisé la modélisation des données de configuration en se basant sur les configurations des clients d'*IMS Networks*. Cette modélisation a permis de générer des jeux de données équilibrés contenant à la fois des configurations valides et erronées, qui ont été étiquetées pour permettre l'apprentissage supervisé. Nous avons utilisé ces jeux de données pour entraîner et évaluer trois modèles d'apprentissage différents : un modèle classique d'arbres de décision (DT), un modèle de méthodes d'ensemble (RF) et un modèle de réseaux de neurones (MLP). Le travail réalisé dans cette première partie de la thèse a été concrétisé par un article de recherche « *Configuration faults detection in IP Virtual Private Networks based on machine learning* » [114], publié dans la conférence internationale « *3rd International Conference on Machine Learning for Networking (MLN'2020)* ».

Ce chapitre est structuré de la manière suivante. La section 3.2 expose la formulation du problème, en détaillant l'objectif d'apprentissage, la méthode d'apprentissage et la méthodologie de travail. La section 3.3 présente la modélisation et la génération des données. Ensuite, les résultats obtenus sont présentés dans la section 3.4, suivis de la conclusion dans la section 3.5.

3.2 Formulation du problème

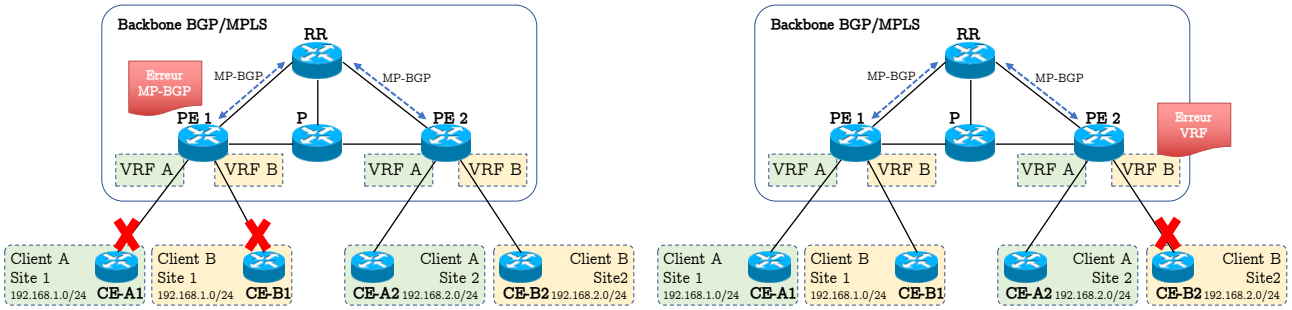
La formulation du problème représente une étape clé dans le processus général de l'apprentissage automatique. En définissant les besoins et les objectifs d'apprentissage, nous pouvons déterminer la méthode d'apprentissage adéquate et identifier les sources et la nature des données à utiliser pour l'entraînement. Dans cette section, nous présentons notre formulation du problème ainsi que la méthodologie de travail que nous avons adoptée.

3.2.1 Objectif et méthode d'apprentissage

La problématique traitée dans cette première partie de notre travail consiste à détecter et localiser les incidents provenant d'erreurs de configuration dans les réseaux VPN BGP/MPLS de niveau 3. Nous définissons un incident comme un comportement anormal du réseau rendant un ou plusieurs sites clients indisponibles. Un site est considéré indisponible dans le cas où il n'est pas joignable par les autres sites du même VPN.

La figure 3.1 illustre deux différents scénarios d'erreurs de configuration générant des incidents dans une architecture VPN BGP/MPLS de niveau 3. Les deux scénarios sont présentés sur une topologie contenant deux clients, dont chacun possède deux sites distants. Chacun des sites dispose d'un seul routeur CE raccordé au réseau *backbone*. Le réseau *backbone* contient deux routeurs PE, un routeur RR et un routeur P. Dans le scénario *a*, la session de peering MP-BGP entre le routeur `PE 1` et le routeur `RR` n'est pas configurée correctement. En conséquence, les informations de routage VPN ne peuvent pas être échangées entre ces deux routeurs. Cela impacte la disponibilité de tous les sites raccordés au routeur `PE 1` indépendamment de la configuration des VRFs. En d'autres termes, cette erreur de configuration rend les deux routeurs `CE-A1` et `CE-B1` injoignables par les autres routeurs des autres sites. En ce qui concerne le scénario *b*, l'erreur de configuration est commise sur l'un des paramètres de la `VRF B` sur le routeur `PE 2`. Cela empêche d'associer les routes émanant du routeur `CE-B2` au bon VPN, c'est-à-dire, au VPN du client `B`. De ce fait, les routeurs des autres sites du client `B` ne peuvent pas communiquer avec ce routeur.

Pour détecter ce type d'incidents, nous avons défini la vérification de la validité de la configuration comme objectif d'apprentissage. Après l'entraînement, le modèle d'apprentissage doit être capable de prédire si une configuration est correcte, c'est-à-dire, si aucun des sites VPN distants ne sera indisponible après le déploiement de cette configuration dans l'environnement de production. Dans le cas contraire, si la configuration contient des erreurs, le modèle d'apprentissage doit localiser les sites qui seront impactés si nous la déployons. Nous formulons cela en un problème de classification multi-label [115].



(a) Labels de sortie : [1, 1, 0, 0]

(b) Labels de sortie : [0, 0, 0, 1]

FIGURE 3.1 – Exemples d'incidents dans une topologie VPN BGP/MPLS de niveau 3.

La différence entre la classification multi-label et la classification classique est principalement dans le format de la sortie attendue du modèle d'apprentissage entraîné. Dans la classification classique, le modèle d'apprentissage renvoie une seule valeur qui représente la classe de la donnée à classifier. Contrairement à cela, dans la classification multi-label, le modèle d'apprentissage renvoie un vecteur contenant deux valeurs ou plus, d'où la notion de « multi-label ». La classification multi-label est utilisée pour classifier les données qui peuvent appartenir à plusieurs classes à la fois. Chaque valeur, ou chaque label, dans le vecteur de sortie représente une classe possible. Si la donnée à classifier appartient à une classe, le label représentant cette classe doit être égale à 1, sinon il doit être égale à 0.

Dans notre cas, nous utilisons la classification multi-label, car une erreur de configuration pourrait impacter plusieurs sites distants à la fois (le cas du scénario *a* de la figure 3.1 par exemple). Le vecteur de sortie contient un label par site. Si un site est indisponible, le label représentant ce site doit être égale à 1, sinon il doit être égale à 0. Dans le cas des deux scénarios illustrés dans la figure 3.1, les vecteurs de sortie doivent être composés de 4 labels : `[CE-A1, CE-B1, CE-A2, CE-B2]`. Le vecteur du scénario *a* doit être égale à `[1, 1, 0, 0]`, et celui du scénario *b* doit être égale à `[0, 0, 0, 1]`. Comme pour tout problème de classification, l'apprentissage doit être supervisé. Cela signifie que nous devons entraîner les modèles d'apprentissage avec des jeux de données contenant différentes configurations étiquetées (labellisées).

3.2.2 Méthodologie de travail

La figure 3.2 illustre les étapes que nous avons suivies afin d'appliquer de l'apprentissage supervisé pour détecter et localiser les incidents provenant des erreurs de configuration dans les réseaux VPN BGP/MPLS de niveau 3. Nous commençons par collecter les configurations du réseau de production, puis nous les analysons afin d'ex-

traire les propriétés nécessaires et effectives pour répondre à notre problème. Ensuite, nous générons les jeux de données en multipliant les configurations en ajoutant différentes erreurs. Ces jeux de données sont utilisés par la suite pour entraîner, tester et valider le modèle d'apprentissage. Nous répétons ces étapes en ajustant les propriétés et les paramètres du modèle, jusqu'à l'obtention de performances satisfaisantes. Une fois que le modèle d'apprentissage est validé, il peut être déployé dans un environnement de production.

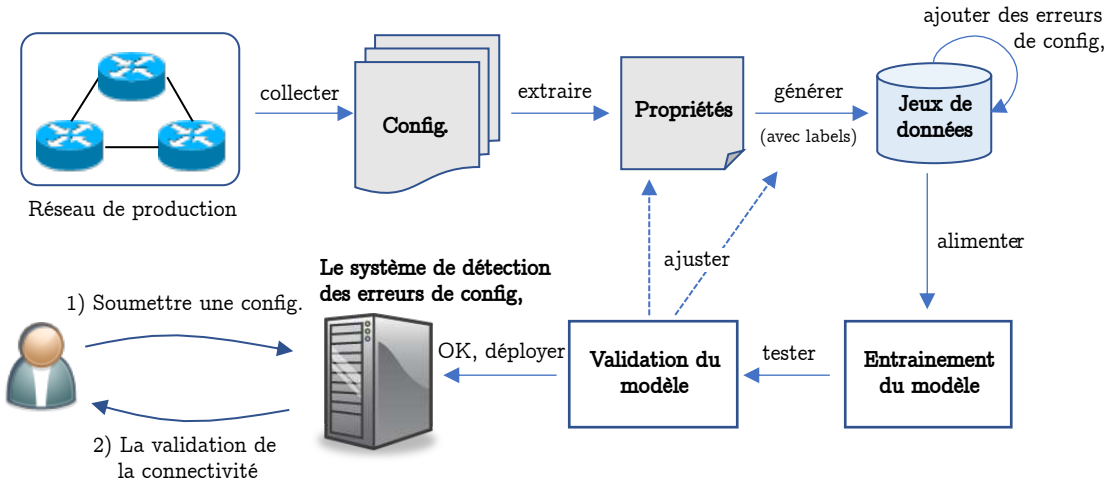


FIGURE 3.2 – La méthodologie générale adoptée dans notre travail.

3.3 Collection, analyse et génération des données

Dans cette section, nous présentons la modélisation ainsi que la génération des données de configuration que nous avons effectuées pour répondre à notre problème d'apprentissage. Ces deux étapes du processus sont importantes, car elles impactent considérablement et de manière directe les résultats que nous obtiendrons après l'entraînement. L'efficacité du processus d'apprentissage est directement liée à la quantité et à la qualité des données. Dans notre contexte, l'étape de modélisation de données consiste à collecter les configurations du réseau de production, à les convertir de leur format texte en un ensemble de propriétés binaires ou numériques, puis à analyser ces propriétés pour identifier celles qui sont pertinentes pour notre problème en particulier, c'est-à-dire celles qui seraient affectées en cas d'erreur de configuration. Cela permet de simplifier les données et de se concentrer uniquement sur les informations importantes pour la détection des incidents issus d'erreurs de configuration.

Afin d'identifier les propriétés les plus pertinentes, nous avons commencé par lister les erreurs de configuration les plus fréquentes observées à *IMS Networks*. Nous avons

ensuite analysé les configurations collectées pour extraire et sélectionner les propriétés en relation avec ces erreurs de configuration. À la fin, nous avons utilisé cette modélisation de propriétés pour générer les jeux de données d'entraînement.

3.3.1 Liste des erreurs de configuration

À *IMS Networks*, nous avons observé que la majorité des incidents provenant d'erreurs de configuration sont causés par des erreurs de routage sur les routeurs CE et sur les routeurs PE. En revanche, nous avons observé qu'il y a très peu d'incidents causés par des erreurs sur la configuration du *backbone* (IGP, MPLS ou LDP). Cela est justifié par le fait que la configuration du *backbone* est stable et change rarement. Contrairement à la configuration du routage VPN, qui est mise à jour quotidiennement. Pour ces raisons, et après avoir collecté et analysé une configuration complète de réseaux VPN *BGP/MPLS* de niveau 3, nous avons décidé de nous focaliser sur trois types d'erreurs : les erreurs de configuration du routage CE-PE, les erreurs de configuration des VRFs et les erreurs de configuration MP-BGP. Nous regroupons dans le tableau 3.1 quelques exemples de ces erreurs de configuration ainsi que leur impact sur la disponibilité des sites clients.

Tableau 3.1 – Les erreurs de configuration dans une architecture VPN BGP/MPLS de niveau 3 et leur impact

Erreurs de configuration		**Sites impactés**
a) Les erreurs de configuration du routage CE-PE	Une erreur de routage entre un CE i et le PE	CE i
	Le préfix annoncé par un CE i est erroné	CE i
b) Les erreurs de configuration des VRFs	Une VRF j n'est pas configurée sur un PE i	Tous les CEs du client j connectés au PE i
	Le RD et/ou le RT mal configurés pour une VRF j sur un PE i	Tous les CEs du client j connectés au PE i
c) Les erreurs de configuration MP-BGP	Le routage BGP n'est pas configuré pour une VRF j sur un PE i	Tous les CEs du client j connectés au PE i
	Une erreur de peering MP-BGP entre un PE i et le RR	Tous les CEs qui sont connectés au PE i

3.3.1.1 Les erreurs de configuration du routage CE-PE

Il s'agit des erreurs de configuration empêchant l'acheminement du trafic entre un routeur CE et le routeur PE auquel il est connecté. Par exemple, une erreur sur le préfixe du réseau LAN annoncé par le routeur CE, ou une mauvaise configuration du routage

statique ou du peering eBGP entre les deux routeurs CE et PE. Ce type d'erreurs rend le routeur CE injoignable de tous les autres routeurs CE, même s'ils sont connectés au même routeur PE.

3.3.1.2 Les erreurs de configuration des VRFs

Un ingénieur réseaux peut oublier de configurer une VRF sur un routeur PE, ou il peut configurer une valeur erronée pour un paramètre spécifique (par exemple, le RD ou le RT). Ce type d'erreurs impacte la disponibilité de tous les routeurs CE appartenant au VPN concerné et qui sont connectés au routeur PE en question.

3.3.1.3 Les erreurs de configuration MP-BGP

Une erreur dans la configuration du peering MP-BGP entre un routeur PE et le routeur RR empêche le routeur PE d'envoyer ou de recevoir les informations de routage VPN. De ce fait, tous les routeurs CE connectés à ce routeur PE seront indisponibles. Aussi, si nous oublions de configurer le routage BGP ou nous le configurons mal pour une VRF, le routeur PE ne communiquera pas les routes du client concerné au routeur RR. Cela rend tous les routeurs CE qui sont connectés à ce routeur PE et qui appartiennent au client concerné injoignables.

3.3.2 Analyse et sélection des propriétés

Nous avons réalisé une analyse des configurations collectées des réseaux VPN de niveau 3 des clients d'*IMS Networks*, ce qui nous a permis de diviser ces configurations en quatre parties distinctes. La première partie concerne la topologie du réseau, et les trois autres concernent les configurations établies sur les différents routeurs déployés, notamment les configurations des VRFs, de MP-BGP et de routage CE-*PE*. Nous décrivons ci-dessous les propriétés sélectionnées pour chacune de ces quatre parties. Le nombre total des propriétés varie selon la taille du réseau, c'est-à-dire selon le nombre de routeurs dans le backbone, de routeurs CE et de réseaux VPN de clients. La combinaison des propriétés des quatre parties forme à la fin un seul vecteur de propriétés représentant une configuration d'un réseau d'opérateur offrant des services de VPN de niveau 3 à ses clients.

3.3.2.1 Les paramètres de topologie

La définition de la topologie fait partie des configurations du réseau. Nous considérons qu'il est indispensable d'inclure des propriétés indiquant chacune l'identifiant du

routeur PE auquel chaque routeur CE est connecté. Cela permet aux modèles d'apprentissage de comprendre et de faire la relation entre les configurations de chaque routeur CE et les configurations du routeur PE correspondant. Nous avons définis ces propriétés sous la forme de nombres entiers uniques. Nous avons défini également dans les paramètres de topologie deux propriétés contenant le nombre des routeurs PE et le nombre des routeurs CE dans le réseau, respectivement.

Nous avons choisi de ne pas inclure d'autres propriétés décrivant les liens au sein du réseau *backbone*, car nous estimons que cela n'est pas nécessaire pour répondre à notre problème d'apprentissage. Cela a été motivé par deux raisons. La première est le fait que tous les routeurs du réseau backbone (les routeurs P, les routeurs PE et le routeur RR) sont interconnectés entre eux (physiquement ou logiquement). Cette information est stable et ne change jamais, d'où l'inutilité de l'ajouter dans les données d'entrainement. Au contraire, ajouter cette information pourrait engendrer de mauvais résultats d'apprentissage, car cela consiste à créer beaucoup de propriétés (le nombre varie selon la taille du réseau) qui sont invariantes et qui diminuent la qualité des données. La deuxième raison pour ne pas inclure ces propriétés est que la détection des erreurs de configuration ciblées dans notre travail ne requière pas de comparaisons entre les configurations de deux routeurs spécifiques du *backbone*, selon leurs types ou selon leurs liens, à l'exception de la détection des erreurs sur la configuration du peering MP-BGP entre chaque routeur PE et le routeur RR. Cependant, étant donné que la configuration du peering MP-BGP est la même pour tous les routeurs PE, une comparaison entre les configurations des routeurs PE est suffisante pour détecter une erreur. Il n'est donc pas nécessaire de définir des propriétés indiquant les liens entre chaque routeur PE et le routeur RR.

3.3.2.2 Les paramètres des VRFs

La configuration des VRFs permet de garantir le routage entre les sites distants des clients de manière à ce que l'isolation du trafic entre les différents réseaux VPN soit assurée. Sur les routeurs PE, l'importation et l'exportation de routes VPN dans et depuis les tables VRFs sont principalement basées sur les valeurs des paramètres de configuration de ces dernières, tels que le RD, le RT d'importation et le RT d'exportation (voir la section 1.4 pour plus de détails). C'est pourquoi, nous avons choisi d'inclure, dans notre vecteur de données, deux propriétés pour chaque VRF sur chaque routeur PE, représentant les valeurs des paramètres RD et RT de cette VRF sur ce routeur PE. Nous avons choisi de définir une seule propriété pour le RT au lieu de deux, avec une pour le RT d'importation et une pour le RT d'exportation, pour ne pas dupliquer la même information. En général, les opérateurs réseaux configurent la même valeur pour les deux paramètres (c'est le cas d'*IMS Networks*). Les deux propriétés sélectionnées ont un format numérique, avec des valeurs allant de 1 jusqu'au nombre maximum possible

des VRFs dans le réseau d'opérateur. Si une VRF n'est pas configurée sur un routeur PE, les propriétés associées à ses paramètres seront égales à 0.

Dans une configuration VPN BGP/MPLS de niveau 3, nous assignons les interfaces des routeurs PE connectées avec des routeurs CE à la VRF du client auquel le routeur CE appartient. Pour simplifier au maximum notre vecteur de données, nous avons choisi de définir une propriété pour chaque routeur CE afin de l'associé directement avec la VRF. La valeur de cette propriété est égale à la valeur du paramètre RD de la VRF. Nous utilisons la valeur du RD comme un identifiant unique des VRF sur les routeurs PE.

3.3.2.3 Les paramètres MP-BGP

Les configurations du protocole MP-BGP nécessaires sur les routeurs PE sont : le peering avec le routeur RR, l'activation de l'échange de routes VPN avec le routeur RR et le routage BGP pour les VRFs présentes sur le routeur PE. Pour le peering avec le routeur RR, nous avons défini deux propriétés par routeur PE. Une de ces deux propriétés indique si le routeur RR est bien configuré comme un voisin BGP, et la seconde indique si la commande « `next-hop-self` » a été activée pour ce voisin. Pour l'échange de routes VPN avec le routeur RR, nous avons défini une troisième propriété pour chacun des routeurs PE, permettant d'indiquer son activation. Dans une architecture VPN BGP/MPLS de niveau 3, pour configurer le peering et activer l'échange de routes *VPN* entre un routeur PE et le routeur RR, il est nécessaire de réaliser également des configurations sur le routeur RR (voir les sections 1.3.2 et 1.4.1). Dans notre modélisation de données, nous avons choisi de ne pas inclure de propriétés concernant cela, car la configuration du routeur RR est stable. Elle ne change que dans le cas d'ajout ou de suppression d'un routeur PE, ce qui arrive rarement dans un réseau d'opérateur.

En ce qui concerne la configuration du routage BGP pour les différentes VRFs présentes sur les routeurs PE, nous avons décidé d'inclure une propriété pour chaque PE et pour chaque VRF indiquant si la configuration a été bien réalisée.

Les différentes propriétés sélectionnées dans les paramètres MP-BGP ont été modélisées sous format binaire. Elles sont égales à 1 si la configuration correspondante est établie, et à 0 sinon.

3.3.2.4 Les paramètres de routage CE-PE

Le processus de routage, entre les routeurs CE et les routeurs PE auxquels il sont connectés, peut être garanti en configurant le routage statique ou en utilisant l'un des protocoles de routage dynamique. Dans le cas de notre étude, nous supposons que seul

le protocole de routage eBGP peut être utilisé. Pour le déployer, il est nécessaire de configurer le peering eBGP entre le routeur CE et le routeur PE, puis d'annoncer les adresses IP et les masques des réseaux LAN du site client via BGP. Dans notre cas, nous considérons qu'il n'est possible d'annoncer qu'un seul réseau LAN par routeur CE. Nous avons résumé ces configurations, pour chacune des connections CE-PE, en trois propriétés : une propriété binaire qui indique si la configuration du peering eBGP entre les deux routeurs CE et PE est bien réalisée, et deux propriétés numériques indiquant l'adresse IP et le masque du réseau LAN annoncé par le routeur CE.

3.3.3 Génération des données

Il existe de multiples données réseaux disponibles sur Internet, telles que les traces de trafic, les métriques de performances, les alertes de sécurités et les journaux d'évènements. En général, ces données peuvent contenir des informations sensibles et privées sur les opérateurs réseaux ainsi que sur leurs clients. Il peut s'agir d'informations telles que les adresses IP, les identifiants de connexion, les informations de trafic, et d'autres données similaires. Pour protéger la confidentialité de ces données, il est courant de les anonymiser avant de les publier. Cela les rend moins spécifiques à un opérateur réseaux ou à une entreprise particulière. Dans le cas de routage dans les réseaux d'opérateurs, nous trouvons de nombreuses bases de données contenant des informations publiques sur les différents ASs des opérateurs, telles que : *CAIDA AS-rank* [103], *CAIDA AS-relationships* [104], *PeeringDB* [105], *AS hegemony* [106], CTI [107] et *ASdb* [108]. Dans [100], les auteurs ont utilisé des données issues de ces bases de données pour proposer un jeu de données de référence permettant l'application des GNNs dans le domaine de routage Internet. Le jeu de données proposé contient toutes les informations de routage relatives aux ASs, à l'exception des données de configuration des routeurs car elles sont privées et ne peuvent pas être rendues publiques. Elles permettent d'obtenir les politiques de routage, les informations de topologie de réseau et l'adressage IP. Les données de configuration sont spécifiques à l'opérateur réseaux et sont donc difficiles à anonymiser efficacement. De plus, obtenir un grand nombre d'erreurs de configuration dans un réseau de production est très difficile. Également, il est irréaliste d'injecter des erreurs dans le réseau uniquement pour créer des données d'entraînement pour les modèles d'apprentissage automatique. Cela pourrait causer des perturbations significatives dans le fonctionnement normal du réseau et impacter la disponibilité et la sécurité des services fournis aux clients. Pour ces différentes raisons, nous avons opté pour la génération des données de configuration en se basant sur des configurations existantes déployées à *IMS Networks*. Cela nous a permis d'avoir des jeux de données suffisamment grands et diversifiés pour contenir à la fois des configurations correctes et des configurations incorrectes.

Afin de générer des configurations réalistes, nous avons défini trois topologies de

réseaux de différentes tailles que nous avons utilisées pour créer différents jeux de données : i) une topologie T1 de petite taille composée de 5 routeurs PE et de 20 routeurs CE appartenant à 3 clients différents, ii) une topologie T2 de taille moyenne composée de 8 routeurs PE et de 50 routeurs CE appartenant à 5 clients différents, et iii) une topologie T3 de plus grande taille composée de 10 routeurs PE et de 100 routeurs CE appartenant à 10 clients différents. Dans les trois topologies, les routeurs CE sont connectés aux routeurs PE et assignés aux clients de manière aléatoire. Ensuite, selon cette répartition, nous initialisons les propriétés en utilisant des valeurs numériques basées sur l'identifiant du routeur ou du client (par exemple, les adresses IP et les RDs), ou en utilisant des valeurs binaires indiquant si un paramètre de configuration est présent ou non (par exemple, présence ou absence d'une VRF sur un routeur PE). Cela nous permet d'obtenir une configuration globale valide pour chacune des trois topologies.

La prochaine étape dans le processus de génération des données consiste à multiplier la configuration de chacune des trois topologies en introduisant des erreurs et en attribuant les labels correspondants. Cela doit aboutir à des jeux de données équilibrés, contenant suffisamment de configurations pour entraîner et tester les modèles d'apprentissage automatique. L'attribution des labels est une opération très importante en apprentissage supervisé. Dans notre contexte, nous attribuons autant de labels que de routeurs CE à chaque configuration. Chaque label doit représenter l'état de disponibilité d'un routeur CE. Si le label d'un routeur CE est égale à 0, cela indique que ce routeur CE sera disponible après le déploiement de la configuration. Sinon, s'il est égale à 1, le routeur CE sera indisponible. Au départ, comme la configuration globale générée est valide, nous initialisons tous les labels à 0. Puis, à chaque fois que nous reproduisons cette configuration en générant des erreurs, nous mettons à jour les labels des routeurs CE impactés pour les rendre égales à 1. Pour la génération des erreurs de configuration, nous considérons 10 types d'erreurs que nous avons regroupés en trois catégories (voir section 3.3.1) : i) les erreurs de routage CE-PE (la configuration d'une adresse IP erronée ou d'un masque erroné pour un réseau LAN d'un site client, une erreur sur la configuration du peering eBGP), ii) les erreurs sur les VRFs (une VRF non configurée, la configuration d'une valeur erronée pour le RD ou le RT) et iii) les erreurs MP-BGP (le protocole BGP non configuré, une erreur sur le peering avec le routeur RR, l'échange de routes VPN avec le routeur RR non activé, le routage BGP non configuré pour une VRF). Nous avons crée, en utilisant ces types d'erreurs, 4 jeux de données pour chacune des topologies (la topologie T1, la topologie T2 et la topologie T3), dont un jeu de données contenant tous les types d'erreurs et trois autres contenant chacun une catégorie d'erreurs. Cela nous a permis de comparer les performances des modèles d'apprentissage avec des erreurs de configuration différentes et de comprendre l'impact de chaque catégorie d'erreurs sur les résultats. Chaque jeu de données créé contient 150000 configurations différentes.

Nous avons créé tous les jeux de données en introduisant les erreurs de configuration

de manière aléatoire. Nous avons sélectionné aléatoirement le type d'erreur à chaque fois, puis l'avons introduit dans la configuration en sélectionnant aléatoirement le(s) routeur(s) et/ou la VRF concernée. Nous avons réalisé cela tout en s'assurant que les jeux de données soient équilibrés en termes de répartition des erreurs, c'est-à-dire que chaque routeur CE devait être impacté de manière équitable par toutes les erreurs dans environ 50% des configurations. Cela est indispensable pour avoir de meilleures performances après l'entraînent des modèles d'apprentissage automatique [116].

3.4 Entraînement et évaluation des modèles d'apprentissage automatique

3.4.1 Modèles d'apprentissage automatique entraînés

Nous avons entraîné et testé dans cette première partie de notre travail trois modèles d'apprentissage automatique : un modèle de Decision Trees (DT) [37], un modèle de Random Forest (RF) [41] et un modèle de Multi-Layer Perceptron (MLP) [57].

3.4.1.1 Le modèle de Decision Trees (DT)

Le modèle de DT [37] est l'un des modèles d'apprentissage supervisé les plus simples. Lors de l'entraînement, il crée un arbre de décision représentant une fonction qui prend en entrée un vecteur de propriétés et renvoie en sortie une décision (une classe). Dans notre cas, nous l'avons implémenté en utilisant *Scikit-learn* [117], un module *Python* qui permet d'implémenter différents algorithmes d'apprentissage automatique. Avec *Scikit-learn*, nous avons la possibilité d'ajuster plusieurs paramètres pour l'entraînement du modèle. Comme par exemple : la profondeur maximale et la profondeur minimale de l'arbre de décision, la stratégie utilisée pour diviser les données (*best* ou *random*) et la fonction qui mesure la qualité de division des données (*gini*, *entroy* ou *log_loss*). Nous pouvons consulter la liste détaillée de ces paramètres dans la documentation de *Scikit-learn* [118]. En ce qui concerne notre cas, nous avons réalisé plusieurs entraînements en ajustant les paramètres du modèle à chaque fois. Les meilleurs résultats ont été obtenus avec les paramètres par défaut de *Scikit-learn*.

3.4.1.2 Le modèle de Random Forest (RF)

Le modèle de RF [41] est une extension du modèle de DT. Il est composé de plusieurs arbres de décision qui opèrent ensemble pour prédire la classe d'un vecteur de propriétés reçu en entrée. La sortie d'un modèle de RF est la classe prédite par la majorité des

arbres de décision constituant ce modèle. Nous avons implémenté, pour notre problème d'apprentissage, le modèle de RF en utilisant *Scikit-learn*. La liste des paramètres du modèle de RF comprend les paramètres du modèle de DT ainsi que d'autres paramètres supplémentaires. Le nombre d'arbres de décision utilisés pour créer le modèle de RF est l'un des paramètres supplémentaires. La liste détaillée des paramètres du modèle de RF est consultable dans la documentation de *Scikit-learn* [119]. Dans notre cas, nous avons choisi d'utiliser les paramètres par défaut de *Scikit-learn*. Le nombre d'arbres de décision par défaut dans le modèle est égale à 100 arbres.

3.4.1.3 Le modèle de Multi-Layer Perceptron (MLP)

Le modèle de MLP [57] consiste en un réseau de neurones artificiels. Il contient au minimum trois couches de neurones : une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. La figure 3.3 illustre l'architecture du modèle MLP utilisé dans notre travail pour répondre à notre problème d'apprentissage. Le modèle utilisé est composé de 3 couches de neurones. En plus des couches d'entrée et de sortie, il contient une couche cachée composée de $((N_{propriétés} - N_{CEs})/2 + N_{CEs}) \times 1.25$ neurones, tel que $N_{propriétés}$ est le nombre de propriétés dans le vecteur de données d'entrée et N_{CEs} est le nombre de routeurs CE (c'est-à-dire, le nombre de *labels* de sortie). Nous avons défini également pour la couche cachée une probabilité de *Dropout* de 0.25. Cela signifie qu'à chaque itération de l'entraînement, chaque neurone dans la couche cachée a une probabilité de 0.25 d'être désactivé [120]. Cette technique permet de réduire le sur-apprentissage (*overfitting*) [121] et d'améliorer les performances générales du modèle. La fonction d'activation utilisée pour la couche cachée est ReLU, et celle utilisée pour la couche de sortie est sigmoïde. Nous avons utilisé la fonction sigmoïde pour la couche de sortie car elle donne une valeur entre 0 et 1 (une probabilité) pour chacun des neurones de sortie. Si la probabilité calculée pour un neurone de sortie est proche de 1, le modèle considère que la classe du label représenté par ce neurone est positive (le routeur CE concerné est indisponible). Sinon, il considère qu'elle est négative (le routeur CE concerné est disponible).

Pour implémenter ce modèle de MLP, nous avons utilisé *Keras* [122], une API *Python* de haut-niveau basée sur *Tensorflow* [123] et qui facilite la création, l'entraînement et l'évaluation de modèles de réseaux de neurones.

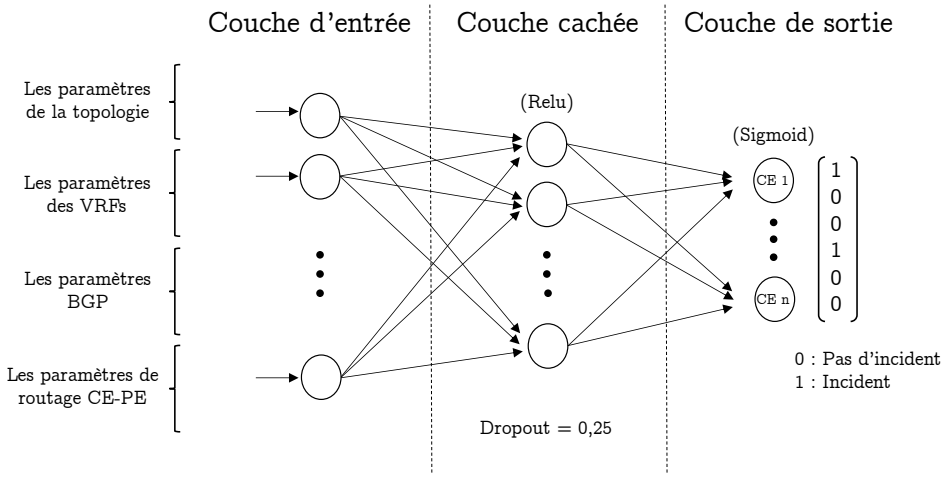


FIGURE 3.3 – L'architecture du modèle MLP pour détecter et localiser les incidents provenant d'erreurs de configuration.

3.4.2 Métriques d'évaluation

Nous avons évalué les performances des trois modèles entraînés en s'appuyant sur des métriques conventionnelles utilisées dans les problèmes de classification, notamment l'*Accuracy*, la *Precision*, le *Recall* et le *F1-score*. Pour comprendre ces métriques dans le contexte de notre étude, il est nécessaire de définir les catégories de prédictions : vrais positifs (True Positive (TP)), vrais négatifs (True Negative (TN)), faux positifs (False Positive (FP)) et faux négatifs (False Negative (FN)). Les prédictions TP et TN représentent les prédictions correctes pour les routeurs CE qui sont respectivement indisponibles et disponibles. En ce qui concerne les prédictions FP et FN, ils représentent les prédictions incorrectes, c'est-à-dire prédire l'indisponibilité des routeurs CE qui sont en réalité disponibles et prédire la disponibilité des routeurs CE qui sont en réalité indisponibles. Nous précisons que les routeurs CE impactés par des erreurs de configuration sont considérés comme des routeurs CE positifs (indisponibles), tandis que les routeurs CE non-impactés par des erreurs de configuration sont considérés comme des routeurs CE négatifs (disponibles). Nous décrivons les métriques d'évaluation et illustrons leurs formules mathématiques ci-dessous.

— L'*Accuracy* représente le taux des prédictions correctes (le nombre de routeurs CE dont l'état de leur disponibilité est bien prédite) par rapport au nombre total des prédictions (le nombre total des routeurs CE).

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

— La *Precision* est le taux du nombre de routeurs CE positifs correctement prédits par rapport au nombre total des routeurs CE prédits comme positifs. Plus la précision est élevée, moins il y a de faux positifs dans les prédictions du modèle.

$$Precision = \frac{TP}{TP + FP}$$

— Le *Recall* est le taux du nombre de routeurs CE positifs correctement prédits par rapport au nombre total de routeurs CE réellement positifs (indisponibles). Plus ce taux est élevé, plus le modèle est capable de détecter les routeurs CE positifs. Cependant, un *Recall* élevé peut également entraîner un nombre élevé de faux positifs. Il est donc important de considérer la métrique du *Recall* en combinaison avec d'autres métriques, comme la *Precision*, pour évaluer les performances globales du modèle.

$$Recall = \frac{TP}{TP + FN}$$

— Le *F1-score* est la moyenne harmonique de la *Precision* et du *Recall*. Elle permet d'évaluer les performances du modèle en prenant en compte les deux métriques.

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

3.4.3 Résultats et discussions

Les expérimentations ont été menées sur les jeux de données présentés dans 3.3.3 : trois topologies de réseaux de différentes tailles (T1, T2 et T3) et différents types d'erreurs de configuration pour chaque taille de topologie. La topologie T1 est composée de 5 routeurs PE et de 20 routeurs CE appartenant à 3 clients différents. Quant à la topologie T2, elle est composée de 8 routeurs PE et de 50 routeurs CE appartenant à 5 clients différents. Enfin, la topologie T3 est composée de 10 routeurs PE et de 100 routeurs CE appartenant à 10 clients différents.

Chacun des jeux de données contient 150000 configurations et est divisé en deux : un pour l'entraînement (contenant 70% des données) et un pour les tests (contenant 30% des données).

3.4.3.1 Le temps d'entraînement

Dans le tableau 3.2, nous présentons les temps d'entraînement de chaque modèle d'apprentissage. Pour mesurer ces temps, nous avons réalisé une série d'entraînements avec différents jeux de données. Pour chaque jeu de données, nous avons fait varier

la taille du réseau (topologie T1, topologie T2 et topologie T3) et les types d'erreurs (d'abord, les erreurs sur les tables VRFs seulement ; ensuite, nous avons ajouté les erreurs sur le routage MP-BGP ; et en dernier, nous avons ajouté les erreurs sur le routage CE-PE). Nous avons réalisé les différents entraînements sur une machine virtuelle équipée de 4 processeurs virtuels ($vCPU$) et de 48 Go de mémoire vive (RAM).

Nous constatons que le nombre et le type d'erreurs de configuration n'ont pas d'influence sur le temps d'entraînement. Nous observons que ce temps reste assez constant pour les trois modèles, quel que soit le nombre de catégories d'erreurs (une, deux ou trois). En revanche, les résultats montrent que le modèle de réseaux de neurones (MLP) prend plus de temps pour l'entraînement que les autres modèles (DT et RF). Il prend environ deux à trois fois plus de temps que le modèle RF, pour les topologies T3 et T2 respectivement. Toutefois, il prend au total moins de 22 minutes pour la topologie T3 avec les trois catégories d'erreurs, ce qui est très raisonnable pour un entraînement hors ligne.

Tableau 3.2 – Comparaison des temps d'entraînement

	Topologie T1			**Topologie T2**			**Topologie T3**		
Erreurs sur :	**DT**	**RF**	**MLP**	**DT**	**RF**	**MLP**	**DT**	**RF**	**MLP**
les VRFs	0m 8s	1m 42s	8m 49s	0m 46s	4m 47s	15m 32s	2m 31s	10m 2s	21m 20s
+ le MP-BGP	0m 10s	1m 53s	8m 45s	0m 42s	4m 52s	15m 57s	1m 56s	9m 3s	21m 37s
+ le routage CE-PE	0m 16s	2m 31s	8m 29	0m 68s	6m 18s	14m 40s	3m 13s	11m 39s	21m 34s

3.4.3.2 Les performances générales

Les figures 3.4 et 3.5 illustrent les résultats détaillés obtenus avec les jeux de données de tests des topologies de réseaux T1 et T3 respectivement. Ces jeux de données contiennent toutes les erreurs de configuration décrites dans la section 3.3.3 (10 types d'erreurs).

Dans l'ensemble, nous observons que le modèle de DT est moins performant que les modèles de RF et de MLP, plus particulièrement lorsque la taille du réseau devient plus grande (la valeur du *F1-score* du modèle DT est égale à 60% pour la topologie T3). Par ailleurs, les résultats sur les deux graphiques (avec un peu plus sur le deuxième) montrent que le modèle MLP a de meilleures performances que le modèle RF. Entre le modèle RF et le modèle MLP, la valeur du *F1-score* augmente de 6% pour la topologie T1 et de 17% pour la topologie T3.

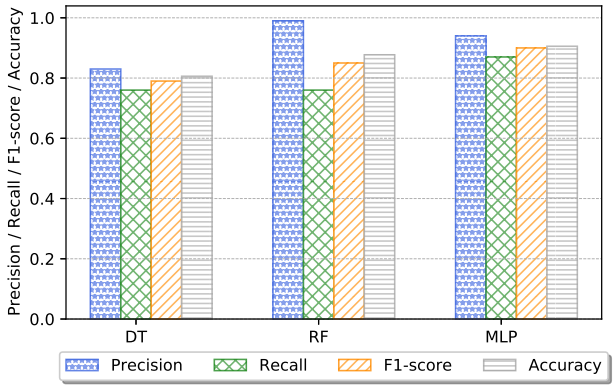


FIGURE 3.4 – La *Precision*, le *Recall*, le *F1-score* et l'*Accuracy* pour la topologie T1.

Nous pouvons voir que la valeur du *F1-score* pour la topologie T3 (figure 3.5) n'atteint pas 80% pour les deux modèles RF et MLP. Cela est dû à la faible valeur du *Recall*. Une faible valeur du *Recall* indique la présence d'un grand nombre de faux négatifs, ce qui représente de nombreux routeurs CE classifiés comme disponibles alors qu'ils sont indisponibles. Cela signifie qu'il y a des erreurs de configuration non-détectées. Nous abordons cette problématique en détails dans la sous-section relative à l'impact des types d'erreurs. En ce qui concerne la valeur de la *Precision*, contrairement à la valeur du *Recall*, elle est très élevée. Elle est égale à presque 100% pour la topologie T1 et à environ 90% pour la topologie T3, pour les deux modèles RF et MLP. Cela veut dire qu'il y a peu de faux positifs. Autrement dit, un routeur CE classifié comme indisponible est presque toujours effectivement indisponible.

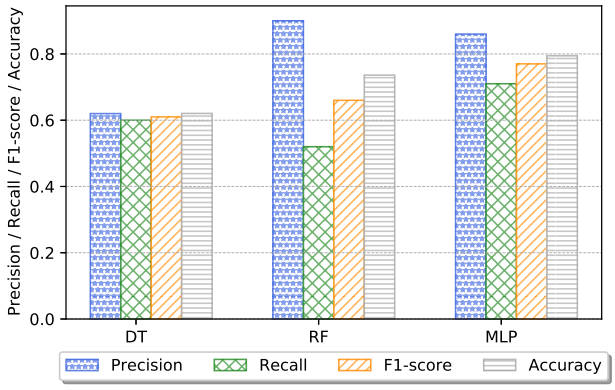


FIGURE 3.5 – La *Precision*, le *Recall*, le *F1-score* et l'*Accuracy* pour la topologie T3.

3.4.3.3 L'impact de la taille du réseau

Pour évaluer l'influence de la taille du réseau sur les performances de chaque modèle d'apprentissage (DT, RF et MLP), nous avons représenté sur le graphique de la figure 3.6 les valeur du *F1-score* obtenues avec les jeux de données de test des topologies T1, T2 et T3 avec 10 types d'erreurs de configuration.

Nous observons que lorsque la taille du réseau augmente, les trois modèles d'apprentissage deviennent moins performants. Nous pouvons expliquer cela par le fait que l'augmentation de la taille du réseau, c'est-à-dire l'ajout de routeurs PE, de routeurs CE, et de VRFs, implique l'ajout de propriétés et de labels pour chaque point de données (pour chaque configuration) dans le jeu de données d'entraînement. Cette augmentation du nombre de propriétés et de labels, sans augmenter la taille du jeu de données d'entraînement, rend le problème d'apprentissage plus complexe, d'où la baisse de la valeur du *F1-score* pour les trois modèles. En apprentissage automatique, ce problème est appelé : le problème de sous-apprentissage (*underfitting*) [121]. Il s'agit d'un scénario qui se produit lorsque le modèle d'apprentissage ne peut capturer les relations entre les données d'entraînement et les labels. Nous pouvons corriger cela en ajoutant plus de données pour l'entraînement ou en réajustant les paramètres des modèles d'apprentissage.

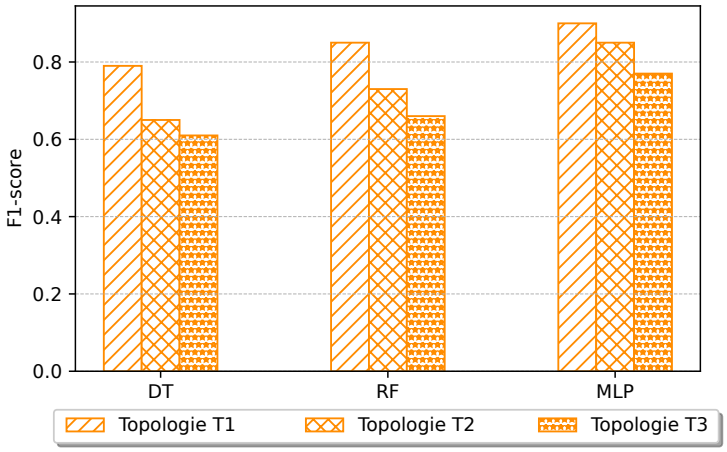


FIGURE 3.6 – La valeur du *F1-score* selon la taille du réseau.

3.4.3.4 L'impact des erreurs de configuration

Le graphique de la figure 3.7 illustre la valeur du *F1-score* des trois modèles d'apprentissage (DT, RF et MLP) pour la topologie T3 selon la catégorie d'erreur. Nous observons que les performances des trois modèles sont meilleures pour le jeu de données contenant des erreurs de configuration MP-BGP, moins bonnes pour le jeu de données contenant des erreurs de configuration sur les VRFs et encore moins bonnes pour le jeu de données contenant des erreurs de configuration sur le routage CE-PE. Cela signifie que les trois modèles entraînés apprennent mieux sur les erreurs MP-BGP et les erreurs sur les VRFs que sur les erreurs du routage CE-PE. Cela est confirmé par les résultats de test détaillés du modèle MLP sur la figure 3.8.

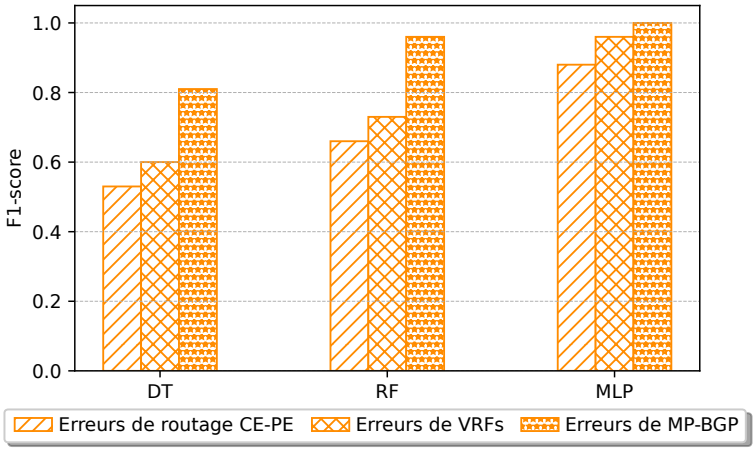


FIGURE 3.7 – La valeur du *F1-score* pour la topologie T3 selon la catégorie d'erreur.

Nous pouvons expliquer cela par le fait que l'impact des erreurs de configuration sur la disponibilité des sites clients est différent d'une catégorie d'erreur à l'autre. Comme présenté sur le tableau 3.1, une erreur sur la configuration du peering MP-BGP entre un routeur PE et le routeur RR rend tous les routeurs CE connectés à ce routeur PE indisponibles, alors qu'une erreur sur l'un des paramètres d'une VRF sur un routeur PE n'impacte que les routeurs CE appartenant au client concerné. Enfin, une erreur sur la configuration du routage CE-PE n'impacte que la disponibilité d'un seul routeur CE. Nous déduisons donc que les modèles d'apprentissage entraînés détectent et localisent plus facilement les incidents liés aux erreurs de configuration ayant un impact sur un plus grand nombre de routeurs CE.

La relation entre les propriétés dans le jeu de données d'entraînement et leurs types de données pourraient être une autre raison qui explique cette différence de perfor-

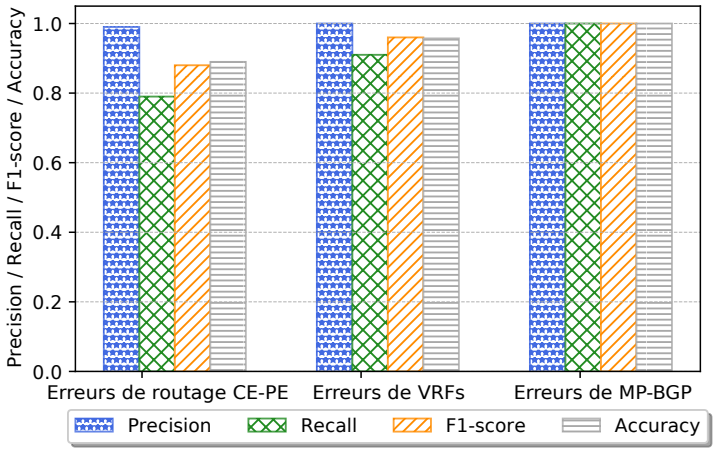


FIGURE 3.8 – La *Precision*, le *Recall*, le *F1-score* et l'*Accuracy* du modèle MLP pour la topologie T3 selon la catégorie d'erreur.

mances. Dans les configurations du routage CE-PE, les adresses et les masques IP des réseaux LAN des sites clients sont configurés une seule fois sur les routeurs CE, et n'ont aucun lien avec le reste de la configuration sur les autres routeurs. De plus, ils sont représentés dans les données d'entraînement par des propriétés numériques, avec une grande cardinalité (plage d'adressage IP) et sans aucune relation entre les valeurs. Cela rend difficile pour les modèles d'apprentissage de déterminer si ces deux propriétés (l'adresse IP et le masque) sont correctes ou erronées. En ce qui concerne les propriétés représentant les paramètres RD et RT des VRFs, bien qu'elles soient également numériques, elles doivent être identiques sur les différents sites d'un même client. Cette relation logique, étant apprise lors du processus d'entraînement, facilite la détection d'erreurs sur ces paramètres. Les valeurs du *Recall* présentées sur le graphique de la figure 3.8 confirment ces suppositions. Les faux négatifs sont plus présents dans le cas d'erreurs de routage CE-PE que dans le cas d'erreurs sur les VRFs, ce qui signifie que les erreurs sur les préfixes IP des réseaux LAN des sites clients sont moins détectées que les erreurs sur les parametres des VRFs (RD/RT). D'autre part, les éléments de configuration MP-BGP sont représentés par des propriétés binaires et ont un impact sur la disponibilité de plusieurs routeurs CE, d'où une valeur proche de 1 pour toutes les métriques sur la figure 3.8.

3.5 Conclusion

Le travail présenté dans ce chapitre consiste en l'application de méthodes d'apprentissage supervisé pour la détection et la localisation d'incidents provenant d'erreurs de configuration. Plus précisément, pour la détection et la localisation d'incidents de connectivité de bout-en-bout entre les différents sites distants de réseaux VPN BGP/MPLS. Nous avons formulé le problème de manière à ce que les modèles d'apprentissage puissent détecter et indiquer les sites VPN indisponibles suite à des erreurs de configuration. Un site VPN est considéré comme étant indisponible s'il est non-joignable par les autres sites du même réseau VPN. Le problème formulé consiste en une classification multi-label.

Ce travail nous a permis d'étudier la modélisation de données de configuration, et de déterminer l'approche d'apprentissage automatique la plus adaptée à notre problématique de thèse. Nous avons entraîné et évalué trois modèles d'apprentissage de différents types : un modèle classique (DT), un modèle de méthodes d'ensemble (RF) et un modèle de réseaux de neurones (MLP). Les résultats d'expérimentations ont montré de meilleures performances avec le modèle de réseaux de neurones. Cela nous oriente vers l'adoption, dans la seconde partie de notre travail, d'une approche basée sur les réseaux de neurones.

En revanche, nous avons constaté qu'en augmentant la taille du réseau, les trois modèles d'apprentissage entraînés deviennent moins performants. Nous avons argumenté cela par le fait que l'augmentation de la taille du réseau implique l'augmentation du nombre de paramètres de configuration, et donc l'augmentation du nombre de propriétés et de labels dans le jeu de données utilisé pour l'entraînement. Pour maintenir des performances d'apprentissage satisfaisantes et éviter un scénario de sous-apprentissage, il est donc nécessaire d'augmenter la taille du jeu de données d'entraînement. Une autre solution serait d'améliorer la modélisation de données afin de minimiser le nombre de propriétés non-représentatives et de permettre d'inclure des configurations de réseaux de différentes tailles dans un seul jeu de données. Cela permettra de détecter et de localiser les incidents liés aux erreurs de configuration indépendamment de la taille du réseau, car le modèle d'apprentissage automatique serait entraîné avec un jeu de données ayant une meilleure qualité et dans lequel des réseaux de différentes tailles sont représentés. Cependant, la méthode de modélisation de données adoptée dans cette première partie de notre travail ne permet pas une amélioration de ce type. Nous avons modélisé les données de configuration de manière statique, où chaque configuration est représentée dans le jeu de données sous la forme d'un vecteur de taille fixe contenant un ensemble de propriétés de valeurs numériques ou binaires. Pour permettre d'inclure des configurations de réseaux VPN de tailles différentes dans le même jeu de données, nous avons défini des hypothèses sur le nombre maximum et le nombre minimum de chaque paramètre de configuration (par exemple, le nombre de VRFs par routeur PE et le nombre

de routeurs CE par client). Cela permet de faire varier le nombre de réseaux VPN et le nombre de sites clients par réseau VPN, mais avec des conséquences négatives sur la qualité du jeu de données d'entraînement. En effet, les configurations ayant un nombre de routeurs CE ou un nombre de réseaux VPN plus petit que le nombre maximum défini sont représentées par des vecteurs de données contenant des propriétés inutiles qui ont un impact négatif sur le processus d'entraînement des modèles d'apprentissage automatique. Cela est une des causes du scénario de sous-apprentissage (*underfitting*). De plus, l'opérateur réseaux est limité et contraint par les nombres maximum et minimum des paramètres de configuration. Par exemple, si un client possède un nombre de sites qui est égal au nombre maximum de routeurs CE par client, il ne pourra pas vérifier sa configuration s'il envisage d'ajouter de nouveaux sites, à moins que l'opérateur réseau ne modifie le nombre maximum de routeurs CE et ne ré-entraîne le modèle d'apprentissage à nouveau. La solution envisageable pour répondre à ces limites consiste à adopter une approche de modélisation de données dynamique. Cela permettrait de créer un jeu de données composé de configurations de réseaux de tailles différentes, sans aucune contrainte ou hypothèse.

Pour résumer les conclusions de la première partie de notre travail de thèse, nous avons constaté les limites de la modélisation de données statique que nous avons utilisée pour représenter les configurations de réseaux. Ainsi, pour la suite de notre travail, nous devons adopter une modélisation de données dynamique. En outre, en raison des bonnes performances obtenues avec le modèle MLP par rapport aux deux autres modèles (DT et RF), nous utiliserons un modèle d'apprentissage basé sur les réseaux de neurones.

Chapitre 4

Application des GNN pour détecter et localiser les erreurs de configuration

4.1 Introduction

Ce chapitre présente le travail réalisé dans la seconde partie de la thèse. L'objectif de cette partie est de proposer une solution basée sur une approche de réseaux de neurones de graphes (ou *Graph Neural Networks (GNN)* en anglais) pour détecter et localiser les erreurs de configurations dans le contexte d'un réseau d'opérateur. Le choix d'utiliser les GNN est basé sur les conclusions obtenues dans la première partie de notre travail. Nous devions proposer une solution avec des modèles d'apprentissage supervisé basés sur les réseaux de neurones et qui pourraient être alimentés par des données de configuration modélisées d'une manière dynamique. Les GNN représentent le type de modèles d'apprentissage automatique le plus adapté pour répondre à ces contraintes.

Comme dans la première partie, nous nous concentrons sur le cas du service de réseaux VPN BGP/MPLS de niveau 3. Nous rappelons que la configuration de ce service est répartie sur plusieurs routeurs de différents types (les routeurs CE, PE et RR). Dans notre approche, nous nous focalisons sur la vérification des parties de configuration les plus sujettes aux erreurs. Par conséquent, nous avons choisi de ne pas inclure la détection et la localisation des erreurs de configuration MP-BGP dans cette seconde partie de la thèse. Tout comme la configuration des protocoles IGP, MPLS et LDP dans le *backbone*, la configuration de MP-BGP est stable et change rarement. De plus, les résultats obtenus dans la première partie de notre travail ont montré que les erreurs de configuration MP-BGP sont relativement simples à détecter (voir section 3.4.3.4), en raison de leur impact important sur la disponibilité des sites distants des clients. Nous estimons que la détection et la localisation des erreurs de configuration MP-BGP ne nécessitent pas l'utilisation de méthodes d'apprentissage automatique.

Nous avons communiqué les contributions de cette seconde partie de notre travail sous la forme d'un article de recherche, intitulé : « *Detecting and locating configuration errors in IP VPNs with Graph Neural Networks* » [124]. Cet article a été publié dans le cadre de l'atelier international « *The Seventh IEEE/IFIP International Workshop*

on Analytics for Network and Service Management (AnNet 2022) », qui s'est tenu conjointement avec le symposium « *IEEE/IFIP Network Operations and Management Symposium (NOMS 2022)* » à Budapest, en Hongrie.

Ce chapitre est structuré de la manière suivante. Dans la section 4.2, nous expliquons les motivations pour lesquelles nous avons utilisé les GNN. Ensuite, la section 4.3 présente en détails la modélisation de données de configuration à l'aide des graphes. La section 4.4 présente le processus de génération de données, l'implémentation des modèles GNN et les résultats obtenus après la phase d'entraînement. Nous montrons par la suite dans la section 4.5 comment nous pourrions intégrer la solution proposée dans l'environnement de production de l'opérateur *IMS Networks*. Enfin, nous concluons ce chapitre dans la section 4.6.

4.2 Choix d'une approche basée sur les GNN

Dans la section 3.5, nous avons conclu que l'utilisation des modèles d'apprentissage automatique basés sur les réseaux de neurones est la meilleure approche pour détecter et localiser les erreurs de configuration dans les réseaux VPN de niveau 3. Nous avons également constaté que nous devons adopter une modélisation dynamique pour représenter les données de configuration. Pour répondre à cela, nous pensons qu'une solution prometteuse serait d'utiliser les GNN. Les GNN sont des modèles d'apprentissage automatique basés sur les réseaux de neurones, spécialement conçus pour traiter des données structurées sous forme de graphes (voir section 2.3.4 pour plus de détails). Les graphes offrent l'avantage de représenter les données d'une manière flexible. Les associer avec les réseaux de neurones nous donne la possibilité de traiter des données sans avoir besoin de fixer des contraintes ou des hypothèses sur leur taille. Les GNN peuvent prendre en entrée des graphes avec des nœuds et des liens de taille et de structure différentes. Dans notre cas, cela permettrait aux clients d'ajouter autant de sites distants qu'ils le souhaitent et de vérifier leurs configurations sans que l'opérateur réseaux ait besoin de ré-entraîner le modèle d'apprentissage. De même, l'opérateur n'est pas contraint par le nombre de clients VPN qu'il peut gérer. La modélisation des données sous forme de graphes est donc une modélisation dynamique qui permet de s'adapter à des configurations de tailles différentes, offrant ainsi une grande flexibilité dans le traitement des données.

Les données de configuration des réseaux peuvent être modélisées à l'aide des graphes, en représentant les différents routeurs par des nœuds de graphes et les connexions physiques et/ou logiques entre eux par des liens. Chaque configuration de routeur peut être représentée sous forme de propriétés à valeurs numériques ou binaires attribuées au nœud de graphe correspondant au routeur. Les graphes fournissent une représentation naturelle et intuitive des réseaux, permettant de modéliser efficacement les dépendances

entre les différents routeurs, telles que les dépendances de configuration. Cette modélisation facilite l'analyse des dépendances et la détection des erreurs potentielles. L'une des principales raisons pour lesquelles nous avons choisi les GNN est qu'ils permettent de prendre en compte la structure des graphes, ce qui est particulièrement important dans notre cas où la topologie du réseau est cruciale. Les GNN utilisent un mécanisme de propagation et d'agrégation d'informations basé sur la structure du graphe pour mettre à jour les représentations des nœuds. Ce mécanisme permet aux GNN de capturer et de modéliser efficacement les relations entre les nœuds, même dans des graphes dynamiques où les relations évoluent avec le temps.

Enfin, en complément des tâches de classification et de prédiction de graphes, les GNN peuvent également être utilisés pour des tâches de classification et de prédiction de nœuds. Cela est particulièrement pertinent pour notre cas, car nous cherchons à détecter et localiser des erreurs de configuration en identifiant les routeurs sur lesquels elles se produisent.

4.3 Modélisation des données à l'aide de graphes

Dans les problèmes de classification traités par des modèles d'apprentissage automatique basés sur les graphes, la définition de l'objectif d'apprentissage (classification de nœuds ou classification de graphes) dépend principalement de la modélisation en graphes des données. Dans notre travail, nous avons décidé de décomposer notre problème d'apprentissage en deux parties, en proposant deux modèles de GNN distincts. Chacun de ces deux modèles est conçu pour détecter et localiser les erreurs sur une partie différente de la configuration des réseaux VPN BGP/MPLS de niveau 3. L'objectif de notre travail est de couvrir, en combinant les deux modèles proposés, le plus d'erreurs de configuration possibles impactant la connectivité de bout-en-bout, indépendamment de la taille du réseau, du nombre de clients ou du nombre de sites par client.

Dans cette section, nous argumentons notre choix de décomposer le problème d'apprentissage en deux parties, puis nous détaillons la modélisation en graphes des données pour chacun des deux modèles GNN et présentons leurs objectifs d'apprentissage.

4.3.1 Décomposition du problème d'apprentissage

Indépendamment de la configuration du *backbone* de l'opérateur réseau, nous distinguons deux parties différentes de configuration permettant le déploiement et le bon fonctionnement d'un VPN de niveau 3. La première partie concerne la configuration de routage inter-sites (PE-PE). Elle comprend les paramètres et les politiques de routage des VRFs configurées sur les routeurs PE, qui contrôlent les interactions entre les

différents sites clients. La seconde partie concerne la configuration de routage entre les sites clients et le backbone (CE-PE). Elle comprend les paramètres de routage entre chaque routeur CE et le routeur PE auquel il est connecté, ainsi que les annonces de routes des sous-réseaux de chaque site client. Dans notre travail, nous avons choisi de modéliser, pour chacune de ces deux parties, un modèle de GNN dédié pour détecter et localiser les erreurs de configuration. Ce choix de séparer la modélisation des deux parties de la configuration est motivé par deux raisons principales : la différence des types de routeurs et l'interdépendance entre les paramètres de configuration.

Dans un modèle de GNN, les nœuds de graphes doivent être modélisés d'une manière homogène, c'est-à-dire qu'ils doivent avoir le même nombre, le même ordre et les mêmes types de propriétés. Dans notre contexte, nous nous intéressons aux configurations de deux types de routeurs différents : les routeurs PE et les routeurs CE. Modéliser les configurations de ces deux types de routeurs dans un même graphe avec les même propriétés est difficile, voire impossible. Par exemple, les configurations des VRFs ne sont présentes que sur les routeurs PE. Si nous modélisons toute la configuration sur un seul graphe, nous devrons inclure les propriétés correspondant aux paramètres des VRFs dans tous les nœuds, y compris ceux qui représentent les routeurs CE. Cela entraînerait un déséquilibre dans le jeu de données d'entraînement, réduisant ainsi sa qualité. L'inclusion de propriétés qui ne sont pertinentes que pour un des deux types de routeurs dans tous les nœuds engendre une surcharge inutile de propriétés invariantes pour certains nœuds, ce qui augmente considérablement le risque de sous-apprentissage (ou *underfitting* en anglais) [121].

En ce qui concerne l'interdépendance entre les paramètres de configuration, les deux parties de la configuration (la configuration de routage inter-sites et la configuration de routage site-backbone) ont des contraintes différentes. Pour vérifier la validité de la configuration de routage inter-sites pour un VPN, il est nécessaire de comparer la configuration des VRFs déployées sur tous les routeurs PE. En revanche, pour la vérification de la configuration de routage site-backbone, aucune comparaison avec les configurations des autres sites n'est requise. La configuration de routage site-backbone dépend uniquement des configurations présentes sur les deux routeurs concernés, c'est-à-dire le routeur CE et le routeur PE auquel le routeur CE est connecté. Cette différence de contraintes d'interdépendance entre les paramètres de configuration représente une raison de plus pour de ne pas fusionner les deux tâches d'apprentissage (la détection et la localisation d'erreurs pour les deux parties de la configuration) dans un seul modèle.

4.3.2 Modèle de routage PE-PE

Le modèle de routage PE-PE est proposé avec pour objectif de détecter et localiser les erreurs sur la partie de routage inter-sites de la configuration des réseaux VPN

BGP/MPLS de niveau 3. Nous listons ci-dessous les erreurs à détecter et à localiser par ce modèle, puis nous détaillons la modélisation en graphes adoptée ainsi que l'objectif d'apprentissage.

4.3.2.1 Liste des erreurs de configuration

Ce modèle cible les erreurs liées aux interactions entre les routeurs PE impliqués dans un réseau VPN, telles que l'absence de configuration d'une VRF sur un routeur PE ou la mauvaise configuration d'un paramètre spécifique d'une VRF (le RD, le RT, etc.). Nous listons toutes les erreurs ciblées par ce modèle dans le tableau 4.1.

Tableau 4.1 – Les erreurs de configuration à détecter et à localiser avec le modèle PE-PE

Erreur de configuration	**Routeur**
La VRF non-configurée	PE
Le RD mal configuré	PE
Le RT d'importation mal configuré	PE
Le RT d'exportation mal configuré	PE
Le sous-réseau à vérifier défini dans la politique de routage d'importation mal configuré	PE
Le RT à vérifier défini dans la politique de routage d'importation mal configuré	PE
Le sous-réseau à vérifier défini dans la politique de routage d'exportation mal configuré	PE
Le RT à appliquer défini dans la politique de routage d'exportation mal configuré	PE

La liste des erreurs de configuration que nous cherchons à détecter et à localiser avec ce modèle inclut quatre erreurs liées aux politiques de routage. Nous avons ajouté ces erreurs car dans cette partie de notre travail, en plus de la configuration classique de type « *Full-mesh* » des réseaux VPN de niveau 3, nous considérons une autre configuration plus complexe de type « *Hub-and-Spoke* » nécessitant l'utilisation de politiques de routage. Dans un réseau VPN adoptant une architecture *Hub-and-Spoke*, les sites distants (appelés « *Spokes* ») ne peuvent communiquer qu'avec un seul site appelé « *Hub* ». Pour mettre en place cette architecture, il existe plusieurs solutions. À *IMS Networks*, sur chacun des routeurs PE connectés aux sites *Spokes*, nous attribuons deux politiques de routage à la VRF appartenant au VPN *Hub-and-Spoke*, permettant l'importation et

l'exportation de routes VPN respectivement. Nous avons présenté plus de détails sur les deux types de configuration dans la sous-section 1.4.2.

4.3.2.2 Modélisation en graphes et objectif d'apprentissage

Le modèle de routage PE-PE doit fournir pour chaque configuration erronée d'un réseau VPN client, le type d'erreur ainsi que le routeur PE sur lequel l'erreur est survenue. Pour ce faire, nous avons choisi de représenter la configuration de chaque réseau VPN séparément sur un graphe dédié. Tous les graphes partagent une même structure, composée de nœuds représentant les différents routeurs PE du réseau *backbone*. Nous connectons chaque nœud de graphe à tous les autres nœuds. Cela reflète la logique d'un réseau *backbone*, où chaque routeur PE doit être en mesure de communiquer avec tous les autres routeurs PE. La structure des graphes de routage PE-PE ne représente pas nécessairement la topologie physique du *backbone*. Chaque nœud de graphe contient un vecteur de propriétés à valeurs numériques ou binaires représentant la configuration de la VRF et des politiques de routage présentes sur le routeur PE correspondant. En outre, nous utilisons une propriété binaire sur chaque lien de graphe pour indiquer si les nœuds associés doivent échanger des informations pour le VPN représenté par le graphe. Cette propriété nous permet de représenter la topologie du VPN (*full-mesh*, *hub-and-spoke*, etc.) dans le graphe. La figure 4.1 présente la modélisation du modèle PE-PE pour trois réseaux VPN différents, dont l'architecture est détaillée dans la figure 4.2. Le client `A` dispose d'un réseau VPN de type *full-mesh*, les trois liens du graphe ont donc leur propriété binaire définie à 1. Le client `B` a un réseau VPN de type *hub-and-spoke*, le lien entre les deux nœuds représentant les routeurs PE connectés aux sites *spoke* a une propriété binaire définie à 0. Enfin, pour le client `C`, seulement un lien a sa propriété binaire définie à 1 car ce client n'est présent que sur deux sites.

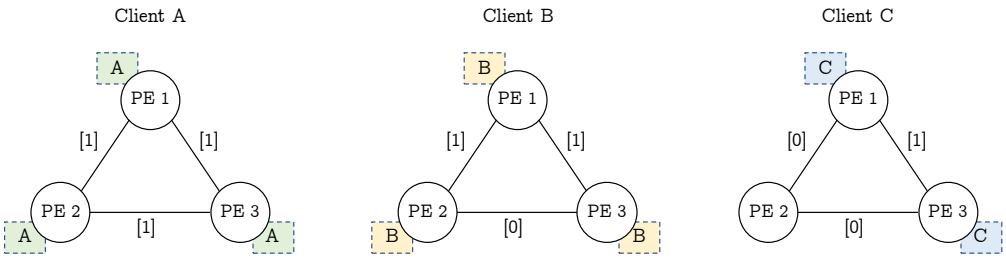


FIGURE 4.1 – La modélisation en graphes du routage PE-PE pour l'exemple de la figure 4.2.

Avec le modèle de routage PE-PE, notre objectif d'apprentissage est de réaliser une classification de nœuds. Pour chaque graphe représentant un VPN, nous aurons une sortie par nœud (c'est-à-dire, par routeur PE) indiquant la présence ou non d'une erreur de configuration. Cette sortie est sous la forme d'un vecteur de multiples classes, chaque

classe représentant une erreur potentielle. Cette classification nous permet de détecter et localiser, pour chaque VPN, les routeurs PE contenant des erreurs de configuration, et de les identifier précisément à l'aide du vecteur de sortie associé à chaque nœud. Pour chaque nœud, nous avons un total de 9 classes : une pour chaque ligne du tableau 4.1, ainsi qu'une classe indiquant une configuration valide.

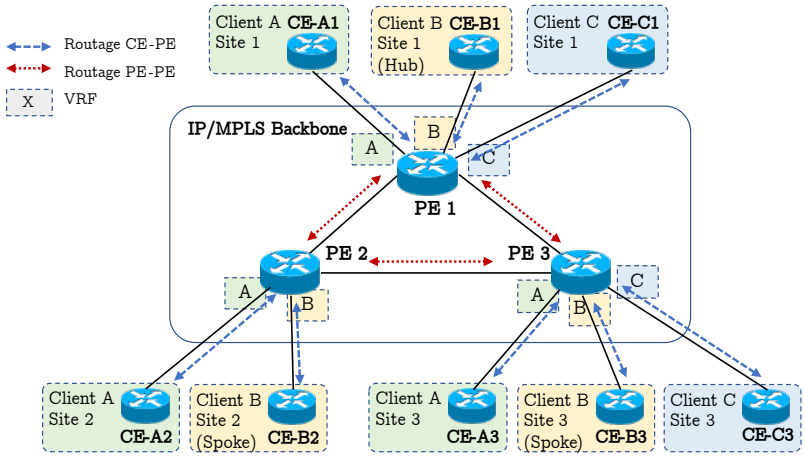


FIGURE 4.2 – Exemple de topologies VPN BGP/MPLS de niveau 3.

4.3.3 Modèle de routage CE-PE

L'objectif du modèle de routage CE-PE est de détecter et de localiser les erreurs de configurations ayant un impact sur l'acheminement des données entre chaque routeur CE et le routeur PE auquel il est connecté. Nous présentons ci-dessous la liste de ces erreurs, puis nous détaillons la modélisation en graphes du modèle et expliquons l'objectif d'apprentissage.

4.3.3.1 Liste des erreurs de configuration

Le modèle de routage CE-PE est conçu pour vérifier la configuration nécessaire permettant d'établir le routage entre chaque routeur CE et le routeur PE auquel il est connecté. Cette partie de la configuration comprend la configuration des interfaces physiques d'interconnexion entre les deux routeurs et la configuration du routage. La vérification de cette configuration doit permettre de détecter et de localiser les erreurs empêchant la connectivité entre les deux routeurs CE et PE. Le tableau 4.2 présente la liste des erreurs prises en compte dans notre travail pour être détectées et localisées par le modèle de routage CE-PE. Pour une meilleure lisibilité, nous avons regroupé les

erreurs de configuration en quatre catégories : les erreurs sur les VRFs, les erreurs sur les interfaces, les erreurs sur le routage eBGP et les erreurs sur le routage statique.

TABLEAU 4.2 – Les erreurs de configuration à détecter et à localiser avec le modèle CE-PE

La catégorie d'erreur	**L'erreur de configuration**	**Le routeur**
Les VRFs	La VRF non-configurée	PE
	Le RD mal configuré	PE
Les interfaces	L'adresse IP mal configurée	CE/PE
	Le masque IP mal configuré	CE/PE
	L'interface n'est pas assignée à une VRF ou la VRF assignée est erronée	PE
Le routage eBGP	Le routage BGP non-configuré	CE/PE
	Le numéro d'AS local mal-configuré	CE/PE
	L'adresse IP du voisin eBGP mal configurée	CE/PE
	Le numéro d'AS du voisin eBGP mal configuré	CE/PE
	Le sous-réseau annoncé mal configuré	CE
	Le routage IPv4 non-activé	CE/PE
Le routage statique	La redistribution de routes statiques dans BGP non-activée	PE
	La VRF assignée mal configurée	PE
	L'adresse IP du destination mal configurée	CE/PE
	L'adresse IP du prochain saut mal configurée	CE/PE

La configuration d'une interface physique d'un routeur PE, connectée à un routeur CE, nécessite l'attribution à la VRF du client auquel le routeur CE appartient. Cependant, cette VRF doit être configurée au préalable sur le routeur PE. Pour cette raison, nous avons inclus deux erreurs liées à la configuration des VRFs dans notre liste d'erreurs. En ce qui concerne le routage, nous avons pris en compte deux options : l'utilisation du routage statique et l'utilisation du protocole eBGP. Le modèle GNN doit être en mesure de détecter et de localiser les erreurs de configuration indépendamment de l'option utilisée.

4.3.3.2 Modélisation en graphes et objectif d'apprentissage

Nous avons choisi de représenter chaque connexion CE-PE par un graphe distinct, contenant deux nœuds : l'un représentant la configuration du routeur CE et l'autre représentant la configuration du routeur PE correspondant au routage avec ce routeur

CE. La sélection des propriétés a été effectuée en considérant la liste des erreurs et en analysant les paramètres de configuration qui peuvent être définis ou modifiés lors de l'ajout, de la mise à jour ou de la suppression d'un site VPN. La figure 4.3 illustre les graphes représentant la configuration du routage CE-PE pour les différents sites de l'exemple présenté dans la figure 4.2.

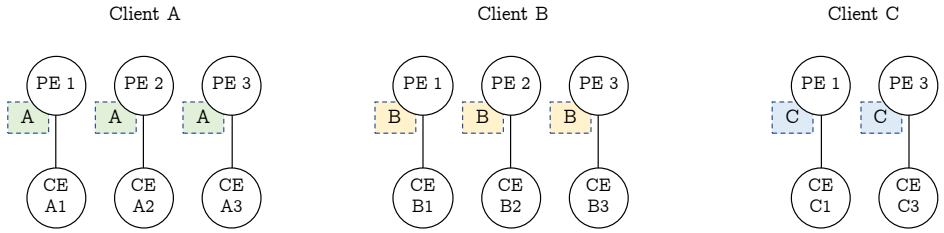


FIGURE 4.3 – La modélisation en graphes du routage CE-PE pour l'exemple de la figure 4.2.

L'objectif d'apprentissage défini pour ce modèle est différent par rapport au premier modèle (le modèle de routage PE-PE). En effet, nous avons choisi d'associer une sortie à chaque graphe plutôt qu'à chaque nœud. Ainsi, nous avons opté pour une approche de classification de graphes plutôt qu'une classification de nœuds. Cette approche de classification multi-classes consiste à attribuer à chaque graphe CE-PE un label, indiquant si la configuration représentée est valide ou indiquant l'erreur de configuration si la configuration est erronée. Le vecteur de classes possibles comporte un total de 30 classes : 29 classes correspondant aux différentes erreurs de configuration possibles, et une classe pour identifier une configuration valide. Nous avons défini la liste des classes de manière à ce que nous puissions localiser le routeur sur lequel l'erreur de configuration s'est produite. Par exemple, pour une erreur concernant le numéro d'AS local BGP, nous avons défini deux classes d'erreurs différentes : une classe indiquant que l'erreur s'est produite sur le routeur PE et une autre classe indiquant que l'erreur s'est produite sur le routeur CE.

4.4 Entraînement et évaluation des modèles GNN

Dans cette section, nous exposons tout d'abord le processus de génération de données, ainsi que l'implémentation et l'architecture des modèles GNN proposés. Ensuite, nous présentons et discutons les résultats obtenus après la phase d'entraînement.

4.4.1 Génération de données

Pour entraîner et évaluer les modèles GNN avec des données de configuration réalistes, comme dans la première partie de notre travail (voir section 3.3.3), nous avons procédé à la génération de données en se basant sur les configurations VPN existants des clients d'*IMS Networks*. *IMS Networks* possède environ une centaine de VPN client sur leur réseau de production, chacun contenant entre 10 et 30 sites distants répartis sur 20 routeurs PE. Ces réseaux VPN, pour la plupart, sont des réseaux VPN simples de type *full-mesh*. Cependant, certains réseaux VPN ont une topologie *hub-and-spoke* nécessitant la configuration de politiques de routage spécifiques. L'objectif de la génération de données dans cette seconde partie de notre travail de thèse est la création de deux jeux de données : un pour le modèle de routage PE-PE et un pour le modèle de routage CE-PE. Les approches d'apprentissage automatique supervisées nécessitent une grande quantité de données étiquetées. Nous avons donc été contraints de multiplier les réseaux VPN d'*IMS Networks* en générant des configurations aléatoires. Nous avons généré 50% de réseaux VPN de type *full-mesh* et 50% de réseaux VPN de type *hub-and-spoke*, afin d'avoir des jeux de données équilibrés. Cette opération est requise pour le modèle de routage PE-PE dans lequel des politiques de routage sont représentées par des propriétés sur les nœuds de graphes.

En ce qui concerne les erreurs de configuration, nous les avons injectées dans les configurations générées de réseaux VPN de manière aléatoire. Nous avons utilisé une fonction permettant de choisir aléatoirement le routeur et l'erreur de configuration tout en prenant en compte la contrainte d'équilibre dans les jeux de données. C'est-à-dire, dans chaque jeu de données, nous devons avoir toutes les classes représentées équitablement. Les erreurs injectées sont prises d'une liste prédéfinie de 38 types d'erreurs (voir tableaux 4.1 et 4.2). À chaque fois qu'une erreur est injectée, le label correspondant du nœud ou du graphe est mis à jour en conséquence.

Le résultat obtenu à la fin de cette phase de génération de données est 1000 configurations de réseaux VPN de niveau 3 de deux types : *full-mesh* et *hub-and-spoke*, contenant à la fois des configurations correctes et incorrectes. Chaque réseau VPN possède entre 10 et 30 routeurs CE, répartis sur 20 routeurs PE. Ces configurations sont utilisées pour créer deux jeux de données pour entraîner et évaluer les modèles GNN de routage PE-PE et de routage CE-PE. Le jeu de données de routage PE-PE contient 1000 graphes (un pour chaque réseau VPN) de 20 nœuds, chaque nœud étant étiqueté en fonction de la classe à laquelle il appartient (voir tableau 4.1). Le jeu de données de routage CE-PE contient 20013 graphes étiquetés (avec des classes provenant du tableau 4.2) correspondant à la somme de toutes les connexions CE-PE de tous les réseaux VPN. Pour les deux modèles, nous avons utilisé 70% des données pour l'entraînement, 10% pour la validation et 20% pour les tests.

4.4.2 Implémentation des modèles GNN

Nous avons utilisé pour l'implémentation des modèles GNN la bibliothèque *Spektral* [125]. Il s'agit d'une bibliothèque *Python* conçue avec pour objectif de faciliter la création et l'utilisation de modèles d'apprentissage profond basés sur les graphes. Elle est basée sur l'API *Keras* [122] et *Tensorflow* 2 [123], et elle fournit des implementations de plusieurs couches de convolution pour les GNN.

4.4.2.1 Caractéristiques du modèle de routage PE-PE

Pour rappel, l'objectif d'apprentissage défini pour le modèle de routage PE-PE consiste à effectuer une classification de nœuds. En d'autres termes, l'objectif est de déterminer la classe d'erreur de chaque nœud représentant la configuration d'un routeur PE dans chaque graphe de VPN. Le modèle GNN conçu pour cela est composé de deux couches de convolution « *Edge-Conditioned Convolutional layers (ECC)* » [126] permettant de prendre en compte les conditions sur les liens de graphes. Nous avons choisi d'utiliser ce type de couches de convolution en raison de la propriété définie sur chaque lien, qui conditionne la communication entre les nœuds de graphes. La figure 4.4 illustre l'architecture du modèle GNN implémenté à l'aide de *Spektral*.

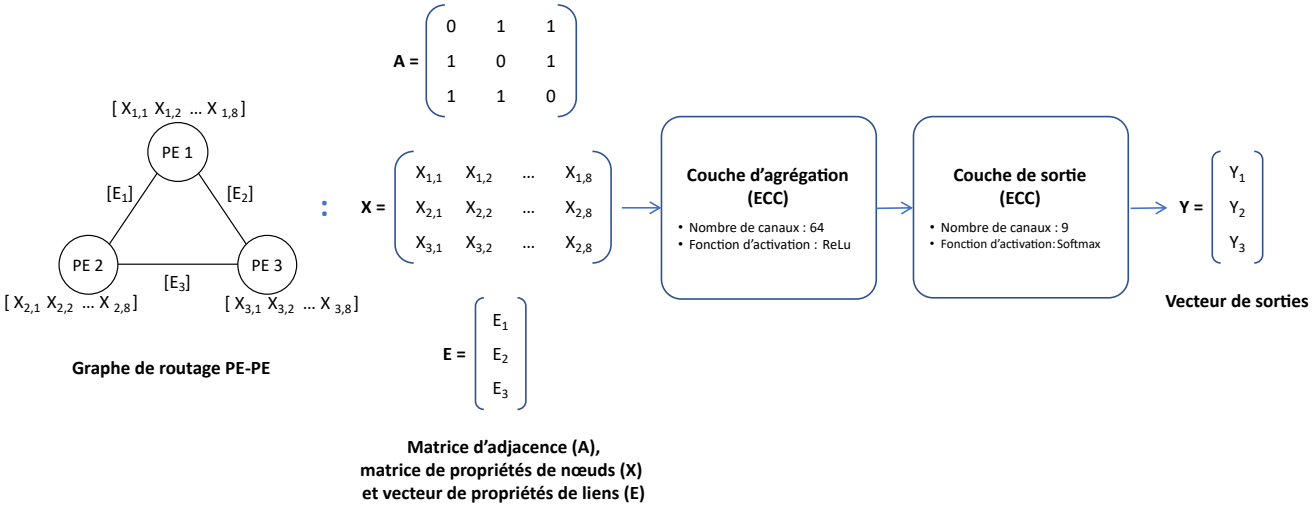


FIGURE 4.4 – Architecture du modèle GNN de routage PE-PE.

La première couche ECC est utilisée pour agréger les propriétés de chaque nœud avec celles de ses nœuds voisins. Nous avons utilisé une seule couche d'agrégation car nous avons estimé que cela est suffisant. En effet, étant donné que tous les nœuds sont connectés à tous les autres nœuds du graphe, une seule couche permet d'agréger les propriétés de tous les nœuds. Cette couche d'agrégation prend en entrée trois paramètres

représentant un graphe de routage PE-PE : i) la matrice d'adjacence (A), ii) la matrice de propriétés de nœuds (X) et iii) le vecteur de propriétés de liens (E). En sortie, elle retourne une matrice contenant un vecteur de 64 valeurs pour chaque nœud, qui représentent le résultat de l'agrégation des propriétés d'entrée. La fonction d'activation utilisée est la fonction ReLU.

Quant à la seconde couche ECC, elle est utilisée en tant que couche de sortie. Le résultat de cette couche doit déterminer la classe d'erreur de chaque nœud de graphe. Cette couche est composée de 9 neurones pour chaque nœud de graphe (un neurone par classe possible). En entrée, elle reçoit la matrice de sortie de la couche d'agrégation, en plus de la matrice d'adjacence (A) et le vecteur de propriétés de nœuds (E). La fonction d'activation utilisée est la fonction *Softmax*. Cette fonction calcule, au niveau de chaque neurone de sortie, la probabilité de la classe correspondante. La somme des probabilités calculées doit être égale à 1. La classe Y_i du nœud de graphe `i` considérée par le modèle GNN est celle ayant la probabilité la plus élevée.

4.4.2.2 Caractéristiques du modèle de routage CE-PE

L'objectif d'apprentissage du modèle de routage CE-PE est de classifier les graphes représentant les configurations de connexions CE-PE. Pour cela, nous avons créé un modèle GNN en utilisant trois couches différentes : une couche de convolution « *GCN* » [110], une couche de regroupement globale « *Global Sum* » et une couche de sortie « *Dense* ». La figure 4.5 présente l'architecture du modèle GNN proposé.

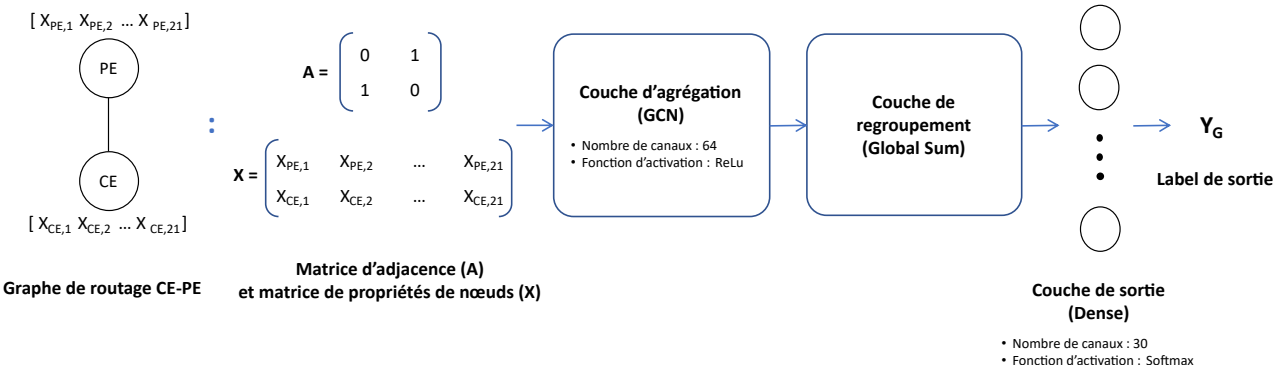


FIGURE 4.5 – Architecture du modèle GNN de routage CE-PE.

La couche GCN a le rôle d'une couche d'agrégation. Elle reçoit en entrée, pour chaque graphe de routage CE-PE, la matrice d'adjacence (A) et la matrice de propriétés de nœuds (X). Puis, elle renvoie en sortie une matrice contenant le résultat d'agrégation. Cette matrice contient deux vecteurs de 64 valeurs (un vecteur pour chaque nœud de graphe). Nous avons utilisé ici la fonction ReLU comme fonction d'activation.

Comme l'objectif d'apprentissage consiste à effectuer une classification de graphes, il est nécessaire d'avoir une représentation globale de haut niveau pour chaque graphe. La deuxième couche du modèle (la couche de regroupement globale « *Global Sum* ») nous permet d'obtenir cette représentation, en calculant la somme des valeurs de la matrice de sortie de la couche d'agrégation. Par la suite, cette représentation globale est utilisée par la troisième couche pour être classifiée. Nous avons utilisé pour cela la couche « *Dense* » de l'API *Keras*. Il s'agit d'une couche MLP composée de plusieurs neurones. Dans notre cas, elle est composée de 30 neurones qui correspondent aux classes possibles. La fonction d'activation que nous avons utilisée pour cette couche est la fonction *Softmax*.

4.4.3 Résultats et discussions

Comme expliqué dans la section 4.4.1, nous avons utilisé 80% des jeux de données générés pour l'entraînement et la validation. Avec les 20% restants, nous avons obtenu une valeur globale du *F1-score* (la moyenne pondérée des valeurs du F1-score de toutes les classes) de 94% pour le modèle de routage PE-PE et de 98% pour le modèle de routage CE-PE. Afin d'évaluer plus précisément ces résultats et de vérifier que les modèles entraînés peuvent généraliser correctement, nous avons décidé de les tester avec d'autres jeux de données, générés comme décrit dans la section 4.4.1, mais triés selon les types d'erreurs, la topologie des réseaux VPN et le nombre de sites distants par réseau VPN.

Les performances des modèles GNN ont été évaluées en utilisant la métrique *F1-score* ainsi que les métriques *Precision* et *Recall*. Vous trouverez plus de détails sur ces métriques d'évaluation dans la section 3.4.2.

4.4.3.1 Évaluation du modèle de routage PE-PE

Les figures 4.6 et 4.7 présentent en détail les résultats des performances du modèle de routage PE-PE en fonction des types d'erreurs et de la topologie des réseaux VPN, respectivement.

Dans la figure 4.6, nous illustrons les valeurs de la *Precision*, du *Recall* et du *F1-score* calculées en utilisant un jeu de données contenant 300 graphes, représentant chacun un réseau VPN de 10 à 30 sites distants (c'est-à-dire des routeurs CE) répartis aléatoirement sur 20 routeurs PE. Cela nous donne un total de 6000 nœuds PE à classifier. Les classes 1 à 8 correspondent aux erreurs de configuration listées dans le tableau 4.1, et la classe 0 indique qu'il n'y a pas d'erreur sur le nœud. En moyenne, nous observons que la valeur du *F1-score* est proche de 90% pour toutes les classes, sauf pour les classes 5 et 7 où elle est proche de 80%.

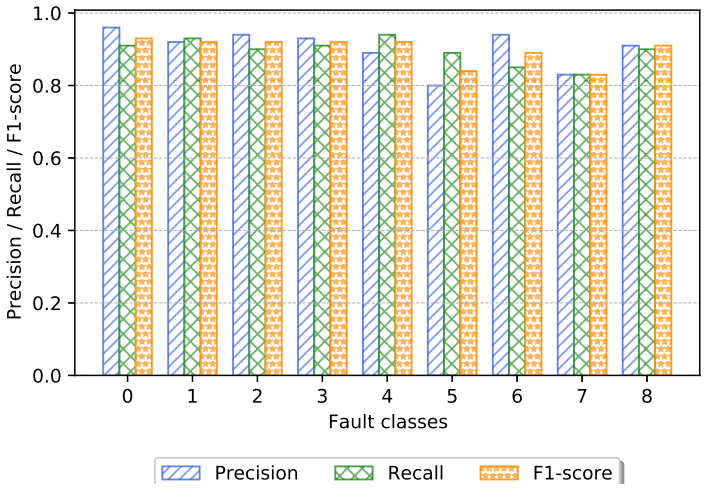


FIGURE 4.6 – La *Precision*, le *Recall* et le *F1-score* par classe d'erreur pour le modèle de routage PE-PE.

Les classes 5 et 7 représentent des erreurs sur les sous-réseaux à vérifier qui doivent être définis dans les politiques de routage d'importation et d'exportation, respectivement. Ces deux paramètres sont utilisés dans le cas des réseaux VPN de type *hub-and-spoke* dans le but de filtrer les routes à importer et à exporter depuis et vers la table VRF du client. Nous argumentons le fait que le modèle de routage PE-PE n'apprend pas aussi bien pour ces deux classes d'erreurs principalement par deux raisons. Premièrement, les paramètres de sous-réseaux n'ont pas d'autres propriétés dans le jeu de données avec lesquelles ils peuvent être comparés ou liés (ils sont présents ailleurs uniquement sur les routeurs CE des clients, dans le modèle de routage CE-PE). Deuxièmement, le jeu de données d'entraînement est déséquilibré pour ces types d'erreurs. En effet, les réseaux VPN de type *hub-and-spoke* représentent 50% des réseaux VPN dans le jeu de données d'entraînement, ce qui rend les erreurs de politiques de routage moins représentées que les autres erreurs (c'est-à-dire les classes 1 à 4) qui peuvent survenir sur n'importe quel type de réseaux VPN. Les classes 6 et 8 représentent également des erreurs sur les politiques de routage, mais concernent les erreurs sur les valeurs du RT qui sont définies dans les politiques d'importation et d'exportation, respectivement. Contrairement aux classes 5 et 7 avec les sous-réseaux, ces valeurs du RT sont liées à la valeur du RT existante dans la configuration de la VRF du client sur le même nœud PE. Les résultats obtenus pour ces deux classes montrent que le modèle peut apprendre la relation existante entre les propriétés représentant les paramètres de la VRF et les paramètres des politiques de routage.

Le graphique présenté dans la figure 4.7 confirme les résultats illustrés dans la figure

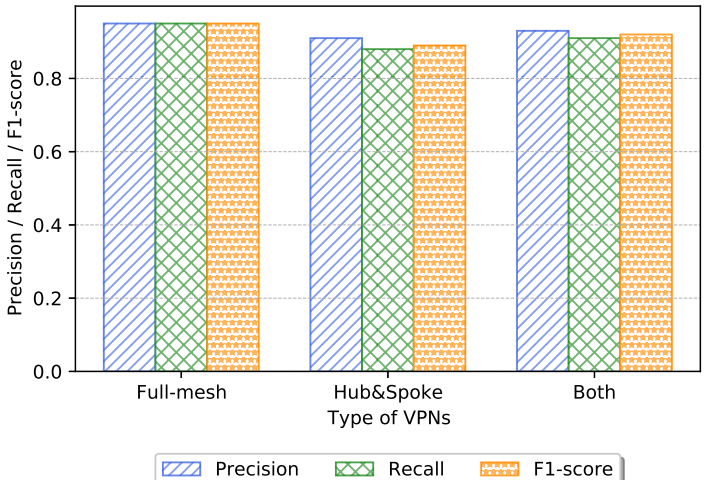


FIGURE 4.7 – La *Precision*, le *Recall* et le *F1-score* par type de VPN pour le modèle de routage PE-PE.

précédente (la figure 4.6). Les valeurs générales de la *Precision*, du *Recall* et du *F1-score* sont calculées en testant le modèle PE-PE sur trois jeux de données différents : i) un jeu de données composé uniquement de réseaux VPN de type *full-mesh*, ii) un jeu de données composé uniquement de réseaux VPN de type *hub-and-spoke* et iii) un jeu de données composé de réseaux VPN des deux types. Les trois jeux de données ont le même nombre de graphes et de nœuds que dans l'expérience précédente. Bien que le modèle apprend bien avec les deux topologies (*F1-score* $> 88\%$), il est clair sur ce graphique qu'il apprend mieux avec les réseaux VPN de type *full-mesh* (c'est-à-dire sans politiques de routage configurées).

4.4.3.2 Évaluation du modèle de routage CE-PE

La figure 4.8 présente les résultats de performances du modèle de routage CE-PE. Les tests ont été effectués sur le même jeu de données utilisé pour tracer le graphique de la figure 4.6. Étant donné qu'il y a de nombreux types d'erreurs dans ce modèle (30 classes), nous avons choisi, pour une meilleure lisibilité, de les regrouper selon la catégorie à laquelle elles appartiennent (voir le tableau 4.2).

Dans l'ensemble, le modèle de routage CE-PE apprend très bien pour toutes les catégories d'erreurs. Il est capable de détecter et d'identifier les erreurs de configuration, qu'il s'agisse d'un routage statique ou du protocole de routage eBGP. Nous avons une valeur du *F1-score* proche de 100% pour toutes les catégories. La nature moins complexe des graphes (seulement deux nœuds) ainsi que la taille du jeu de données d'entraînement

pour ce modèle (près de 30 000 graphes) peuvent expliquer ces très bons résultats.

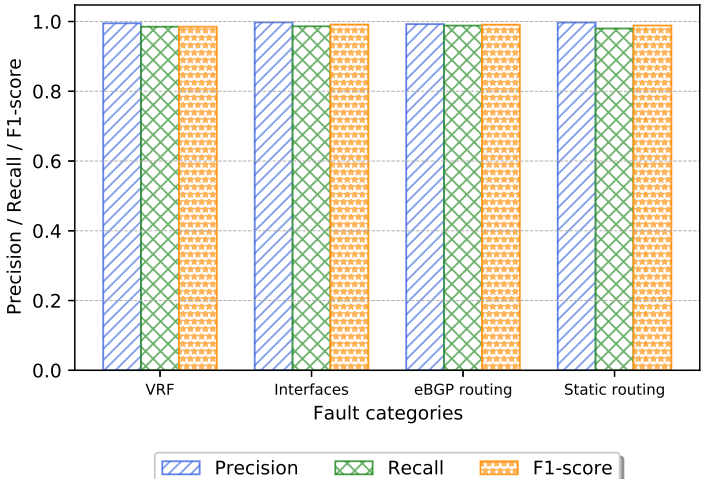


FIGURE 4.8 – La *Precision*, le *Recall* et le *F1-score* par catégorie d'erreurs pour le modèle de routage CE-PE.

4.4.3.3 Impact de la taille du VPN sur les performances des deux modèles

La figure 4.9 présente la valeur du *F1-score* des modèles de routage PE-PE et de routage CE-PE en utilisant six (6) jeux de données différents, chacun contenant 300 réseaux VPN. Nous avons défini pour chaque jeu de données un nombre fixe de routeurs CE par réseau VPN afin d'évaluer l'impact de la taille du VPN sur les performances des modèles (les routeurs CE sont distribués de manière aléatoire sur 20 routeurs PE pour chaque réseau VPN).

Nous pouvons observer que la variation du nombre de routeurs CE par réseau VPN n'impacte pas les performances du modèle de routage CE-PE. Cela est logique car la variation du nombre de routeurs CE n'a aucune influence sur les graphes CE-PE, elle ne fait que changer le nombre de graphes à classifier. Quant au modèle de routage PE-PE, la valeur du *F1-score* globale pour 10, 20 et 30 routeurs CE par réseau VPN est similaire à la moyenne de la valeur du *F1-score* tracée dans la figure 4.6 (environ 90%). Ce résultat est cohérent puisque le modèle a été entraîné avec un jeu de données contenant des réseaux VPN avec 10 à 30 routeurs CE. Un résultat plus intéressant est la valeur du *F1-score* pour les réseaux VPN avec 3, 5 et 40 routeurs CE. Bien que ces tailles de réseaux VPN n'étaient pas présentes dans le jeu de données d'entraînement, la précision de classification est très bonne (une valeur de *F1-score* inférieur de seulement 3% ou 4%), ce qui indique que le modèle GNN de routage PE-PE entraîné généralise

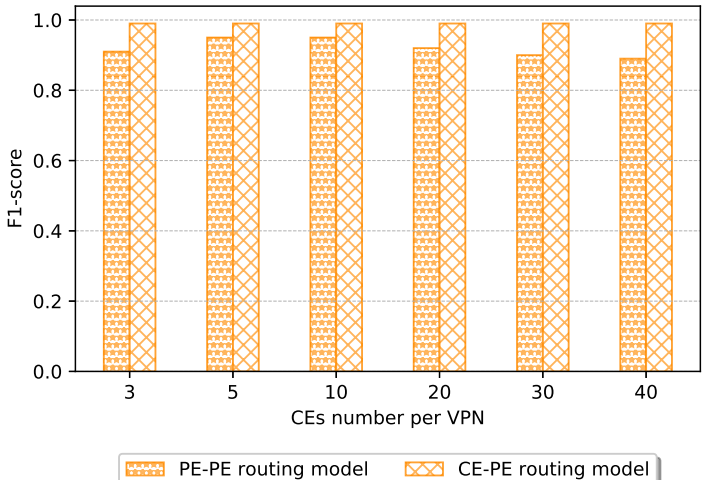


FIGURE 4.9 – La valeur du *F1-score* selon le nombre de sites par VPN.

bien, sans sur-apprentissage (*overfitting*).

4.5 Intégration dans l'environnement de production

L'objectif de la thèse n'est pas de proposer une solution complète de vérification de configuration prête à être déployée en production. Nous nous sommes plutôt concentrés sur la formulation du problème d'apprentissage, le choix de l'approche à adopter, la modélisation des données et la conception du modèle d'apprentissage. Toutefois, les résultats encourageants que nous avons obtenus nous ont incités à réfléchir à la définition d'une architecture logicielle qui permettrait d'intégrer la vérification de configuration du service VPN de niveau 3 dans les processus de déploiement et d'exploitation des réseaux à *IMS Networks*. L'objectif de cette démarche est de montrer, à travers une architecture simple, comment notre proposition peut être utilisée au quotidien par des ingénieurs de déploiement et/ou d'exploitation au sein d'un opérateur réseaux. Souvent, comprendre la cause d'incidents et les corriger représente une tâche chronophage. La vérification de la configuration par un outil basé sur l'apprentissage automatique, avant de déployer de nouveaux routeurs ou de modifier une configuration existante, permettrait d'améliorer la fiabilité et de gagner du temps. Les performances des modèles d'apprentissage automatique offrent un niveau d'assurance important quant à l'exactitude des configurations déployées, ce qui permet d'éviter de provoquer des incidents liés à des erreurs de configuration.

La figure 4.10 illustre un aperçu de l'architecture proposée pour intégrer la vérification de configuration pour le service VPN de niveau 3 dans l'environnement de production. Nous avons opté pour une architecture composée de déférents modules indépendants et qui interagissent entre eux : i) une interface API fournissant des fonctions de vérification aux utilisateurs, ii) des modules de construction de graphes de configuration à partir des configurations soumises par les utilisateurs, iii) des modules de vérification de configuration contenant les modèles d'apprentissage automatiques validés et déployés, et iv) une base de données orientée graphe pour stocker les configurations valides qui ont déjà été vérifiées pour les réutiliser lors d'autres vérifications. Cette architecture modulaire permet une plus grande flexibilité et une meilleure gestion des différents composants, facilitant ainsi la maintenance et l'évolutivité de l'ensemble du système.

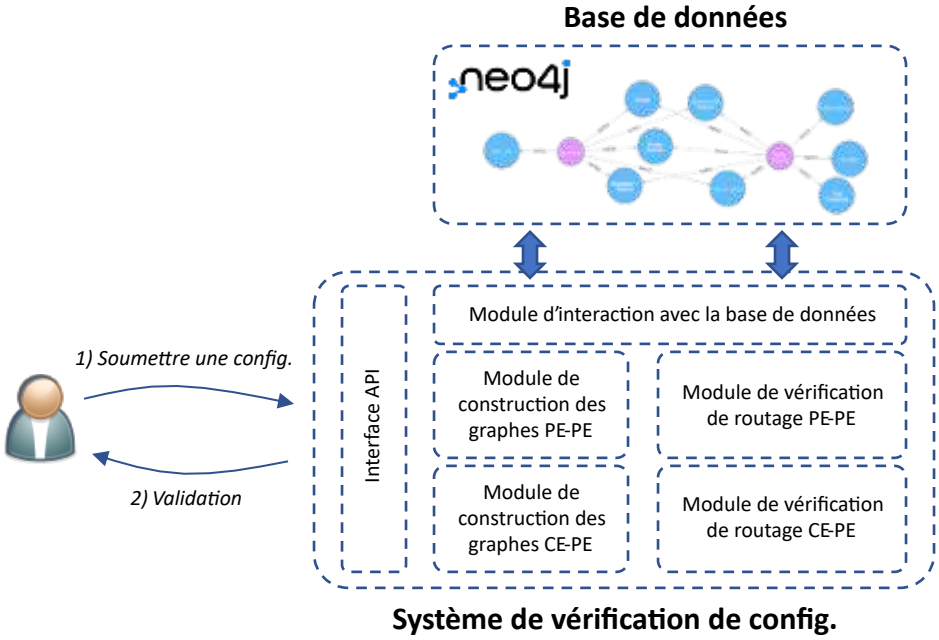


FIGURE 4.10 – Aperçu de l'architecture du système de vérification de configuration.

4.5.1 Fonctionnalités de vérification

L'interface API pourrait être utilisée par une application pour permettre aux ingénieurs réseaux de soumettre une configuration au système de vérification. Si la configuration est validée par le système de vérification, l'ingénieur réseaux peut la déployer sur le(s) routeur(s) concerné(s) en production. Si la configuration est invalidée, la réponse de la requête initiée par l'ingénieur réseaux doit indiquer le type d'erreur de configuration

et le routeur sur lequel l'erreur s'est produite. Nous proposons de fournir, via l'interface API, trois fonctionnalités de vérification de configuration de VPN de niveau 3 :

1. Pour tous les réseaux VPN de l'opérateur.
2. Pour tous les sites VPN d'un client donné.
3. Pour un seul site VPN d'un client donné.

La vérification de toutes les configurations de VPN de niveau 3 de l'opérateur réseaux pourrait être effectuée lors de la première utilisation du système de vérification, même s'il n'y a pas d'erreurs à détecter. Cette opération est nécessaire pour initialiser la base de données du système de vérification, dont l'utilité est détaillée dans la sous-section 4.5.4. La deuxième fonctionnalité, c'est-à-dire la vérification des configurations de tous les sites VPN d'un client donné, pourrait être utilisée pour chaque ajout d'un nouveau VPN client sur le réseau *backbone* de l'opérateur. En ce qui concerne la vérification de configuration pour un seul site VPN, les ingénieurs réseaux pourraient effectuer cette opération à chaque ajout d'un nouveau site ou modification d'un site existant.

Une première version de cette interface API a été réalisée avec pour objectif de présenter le système de vérification aux équipes de déploiement et d'exploitation réseaux à *IMS Networks*.

4.5.2 Construction des graphes de configuration

Dans le domaine des réseaux, une configuration est spécifiée sous forme de texte structuré qui décrit les paramètres de configuration d'un routeur ou d'un commutateur (*switch)*. Cette description peut varier selon le type d'équipement réseau et le constructeur. D'un autre côté, en apprentissage automatique, les données doivent être présentées sous la forme d'un ensemble de propriétés à valeurs numériques ou binaires, telles que des vecteurs, des matrices ou des graphes. Dans notre cas, les données sont encodées sous forme de graphes, composés de nœuds et de liens ayant des propriétés représentant les paramètres de configuration.

Pour faciliter l'utilisation du système de vérification, il est nécessaire de permettre aux ingénieurs réseaux de soumettre les configurations sous leur format ordinaire, c'est-à-dire sous forme de texte structuré. Pour parvenir à cela, nous proposons d'ajouter deux modules de construction des graphes de configuration dans le système : un module pour la construction des graphes du modèle de routage PE-PE et un module pour la construction des graphes du modèle de routage CE-PE. Ces deux modules ont pour rôle de créer, à partir des configurations soumises par les ingénieurs réseaux, des graphes de configuration PE-PE et CE-PE qui seront utilisés comme données d'entrée pour les modèles GNN de vérification. Ceci est assuré en effectuant des opérations de transfor-

mation de format pour extraire la sémantique de la configuration et l'exprimer sous la forme de graphes avec des propriétés sur les nœuds et sur les liens.

Au cours de la dernière année de la thèse, un étudiant de Master 1 a rejoint notre équipe en tant que stagiaire pour nous aider dans ce travail. L'une de ses missions consistait à développer un service permettant d'extraire les paramètres de configuration à partir de fichiers texte et de les présenter sous forme d'un ensemble de propriétés numériques ou binaires. Pour ce faire, il a utilisé une bibliothèque d'analyse de texte semi-structuré en *Python* appelée « *Template Text Parser (TTP)* » [127]. Le résultat a été une extraction complète des informations de configuration manipulées dans les graphes PE-PE et CE-PE.

4.5.3 Modules de vérification automatique

Après avoir entraîné, évalué et validé les deux modèles GNN de détection et de localisation d'erreurs de configuration pour les deux parties de routage PE-PE et de routage CE-PE, respectivement, nous avons enregistré leurs poids en utilisant une fonction de l'API *Keras* [122]. Ces poids peuvent être ensuite importés dans les deux modèles qui seront utilisés en production, à l'aide d'une autre fonction fourni par l'API *Keras*. Dans le système de vérification des configurations, chacun des deux modèles GNN entraînés et validés est intégré dans un module dédié.

Afin de pouvoir présenter ce travail aux ingénieurs réseaux d'*IMS Networks*, nous avons déployé les deux modèles GNN sur une machine virtuelle de test. Nous les avons ensuite interrogés en utilisant la première version de l'interface API que nous avions développée.

4.5.4 Base de données pour stocker les configurations vérifiées

Le choix d'intégrer une base de données dans le système de vérification a été fait dans le but de simplifier au maximum son utilisation. Le rôle de cette base de données est de stocker toutes les configurations, vérifiées et validées, des réseaux VPN clients de l'opérateur. Après chaque vérification, si les configurations soumises sont correctes, elles sont stocker automatiquement dans la base de données. Ainsi, lorsqu'un nouveau site est créé ou qu'un site existant est modifié, il n'est pas nécessaire de soumettre toute la configuration du VPN pour vérifier si la nouvelle configuration permet au site d'être joignable par les autres sites du VPN. Il suffit de soumettre uniquement la partie ajoutée ou modifiée de la configuration. Les modules de construction des graphes de configuration interrogent automatiquement la base de données pour compléter la configuration soumise par le reste de la configuration du réseau VPN.

Nous considérons que les graphes sont le meilleur moyen de représenter les données de réseaux. C'est pourquoi nous avons choisi d'utiliser une base de données NoSQL orientée graphe. Ce choix s'explique également par le fait que nous manipulons des données de configuration modélisées sous forme de graphes.

Pendant son stage à *IMS Networks*, le stagiaire a pu conçevoir, créer et initialiser une première version de la base de données en utilisant l'outil d'extraction des paramètres de configuration à partir de fichiers texte qu'il avait développé. Pour cela, il a utilisé la base de données orientée graphe *Neo4j* [128].

4.6 Conclusion

Dans ce chapitre, nous avons présenté une approche basée sur les GNN pour la vérification de configuration dans le contexte d'un réseau d'opérateur. Nous avons défini comme problème d'apprentissage la détection et la localisation d'erreurs de configuration pour le service de réseaux VPN BGP/MPLS de niveau 3. Cela inclut la détection de l'erreur de configuration, l'identification du routeur sur lequel l'erreur est présente, l'identification du VPN client concerné et l'identification du type d'erreur. Le choix d'utiliser une approche basée sur les GNN nous a permis d'utiliser les réseaux de neurones avec des données de configuration modélisées de manière dynamique à l'aide des graphes, ce qui nous a donné la possibilité d'entraîner les modèles d'apprentissage sans avoir besoin de poser des contraintes ou des hypothèses sur la taille des réseaux VPN.

Nous avons proposé deux modèles de GNN permettant de détecter et de localiser les erreurs sur deux parties de configuration distinctes : la partie de configuration de routage inter-sites (PE-PE) et la partie de configuration de routage site-backbone (CE-PE). Pour le modèle de routage PE-PE, nous avons considéré deux architectures différentes de réseaux VPN : une architecture *full-mesh* simple et une architecture *hub-and-spoke* nécessitant la configuration de politiques de routages. Nous avons modélisé la configuration de chaque VPN dans un graphe dédié, ce qui nous a permis de vérifier la configuration de chaque VPN séparément. Pour ce modèle, l'objectif d'apprentissage consiste à effectuer une classification de nœuds afin de détecter et d'identifier les erreurs pour chaque routeur PE (chaque routeur PE est représenté par un nœud de graphe). En ce qui concerne le modèle de routage CE-PE, nous avons modélisé chaque configuration de connexion CE-PE par un graphe contenant deux nœuds (un nœud pour le routeur CE et un nœud pour le routeur PE). L'objectif d'apprentissage défini pour ce modèle est la classification de graphes.

Les résultats d'évaluations sont prometteurs pour les deux modèles. Après la phase d'entraînement, nous avons réalisé des tests avec différents jeux de données triés par types d'erreurs, type de VPN et taille de VPN. Nous avons obtenu des valeurs de *F1-*

score allant de 80% à 90% pour le modèle de routage PE-PE, et proches de 100% pour le modèle de routage CE-PE. De plus, les tests effectués sur des jeux de données contenant des réseaux VPN différents de ceux présents dans le jeu de données d'entraînement (c'est-à-dire, avec un nombre différent de sites et un placement différent sur les routeurs PE) ont indiqué une baisse de performance de seulement 4% pour le modèle de routage PE-PE, et aucune baisse de performance pour le modèle de routage CE-PE. Cela signifie que les deux modèles fonctionnent très bien sur des données non vues précédemment, et donc généralisent bien.

À la fin de ce chapitre, nous avons présenté une architecture modulaire permettant d'intégrer l'outil de vérification de configuration dans un environnement de production. Cette architecture est composée d'une interface API, de deux modules de construction de graphes respectivement pour les parties de configuration de routage PE-PE et CE-PE, de deux modules de vérification automatique de configuration toujours pour les parties de routage PE-PE et CE-PE, et d'une base de données NoSQL orientée graphe pour stocker les configurations vérifiées et validées. Nous n'avons pas été en mesure de développer et de déployer l'intégralité de l'architecture proposée en raison d'un manque de temps. Cependant, certaines parties ont été réalisées, notamment une première version de l'interface API, un service permettant d'extraire les paramètres de configuration à partir de fichiers texte, ainsi qu'une première version de la base de données des configurations validées.

Conclusion générale

Rappel du contexte et de la problématique

Au cours de cette thèse, nous avons exploré la problématique de la détection et de la localisation d'erreurs de configuration dans le contexte d'un réseau d'opérateur, en se concentrant sur le service de réseaux VPN BGP/MPLS de niveau 3. Notre objectif principal était de proposer une nouvelle approche basée sur des modèles d'apprentissage automatique, qui permettrait aux ingénieurs réseaux de vérifier les configurations avant chaque ajout, suppression ou modification dans le réseau de production.

Nous avons entamé notre travail en réalisant un état de l'art des réseaux d'opérateurs et des configurations nécessaires pour déployer les services réseaux fournis aux clients. Cette étape nous a permis de mieux comprendre le contexte de la thèse et la problématique liée à la vérification des configurations. Un opérateur réseaux doit fournir des services à ses clients tout en assurant un niveau élevé de qualité de service, de fiabilité et de disponibilité. Pour y parvenir, il est indispensable de mettre en place des processus permettant de prévenir les incidents ou de les corriger rapidement lorsqu'ils surviennent. La vérification des configurations avant chaque déploiement est l'un de ces processus pour éviter les incidents. À *IMS Networks*, l'expérience a démontré qu'un grand nombre d'incidents est causé par des erreurs humaines, y compris les erreurs de configuration lors de l'ajout de nouveaux équipements réseaux (routeurs ou commutateurs) ou de la modification des équipements existants. Face à ce besoin, nous avons constaté les limites des méthodes classiques de vérification. Ces méthodes sont généralement basées sur des règles et des contraintes qui doivent être mises à jour manuellement. La mise à jour manuelle de cette base de règles est très chronophage et ne garantit pas sa complétude par rapport à l'ensemble des erreurs de configuration possibles. De plus, au-delà de la détection et la localisation des erreurs de syntaxe et des erreurs liées à la structure générale d'une configuration, un opérateur réseaux doit être en mesure de vérifier si les configurations répondent aux besoins spécifiques de chaque client, ce qui est impossible avec les méthodes classiques de vérification de configuration.

Afin d'atteindre notre objectif, nous avons poursuivi notre travail en effectuant un état de l'art des méthodes d'apprentissage automatique et de leur application dans le contexte des réseaux. Cette démarche nous a permis d'explorer les différentes techniques d'apprentissage automatique existantes, telles que l'apprentissage supervisé, l'apprentissage non-supervisé et l'apprentissage par renforcement, et de comprendre comment

ces approches peuvent être appliquées pour résoudre des problèmes spécifiques liés aux réseaux. En examinant les réussites et les défis rencontrés dans ce domaine, nous avons acquis une meilleure compréhension des opportunités offertes par l'apprentissage automatique pour améliorer la vérification de configuration de routage dans le contexte des réseaux d'opérateurs.

Synthèse des contributions

Les contributions de cette thèse ont été réalisées en deux parties. La première partie avait pour objectif la détection et la localisation d'incidents issus d'erreurs de configuration en utilisant des méthodes d'apprentissage automatique. Quant à la seconde partie, l'objectif consistait à proposer une approche basée sur les réseaux de neurones de graphes (GNN) pour vérifier les configurations de réseaux VPN de niveau 3, c'est-à-dire, detecter et localiser les erreurs de configuration impactant la connectivité de bout-en-bout. Nous présentons ci-dessous un synthèse pour chacune des deux parties.

Détection et localisation d'incidents provenant d'erreurs de configuration

Dans la première partie, nous avons proposé d'appliquer des méthodes d'apprentissage supervisé pour détecter et localiser les incidents résultant d'erreurs de configuration. En d'autres termes, l'objectif d'apprentissage est d'indiquer pour chaque site VPN s'il est indisponible à cause d'erreurs de configuration. Un site VPN est considéré comme étant indisponible s'il est non-joignable par les autres sites du même réseau VPN. Le problème d'apprentissage formulé consiste en une classification multi-label, où nous attribuons un label pour chaque site VPN afin de représenter son état de disponibilité.

Après avoir formulé le problème, nous avons poursuivi cette partie de notre travail en modélisant les paramètres de configuration des réseaux VPN BGP/MPLS de niveau 3 en un ensemble de propriétés, de valeurs numériques ou binaires, structurées sous la forme d'un vecteur de taille fixe. Ces propriétés incluent les paramètres de configuration de tous les réseaux VPN de clients présents dans le réseau de l'opérateur. La sélection des propriétés a été effectuée en se basant sur les configurations des clients d'*IMS Networks* et sur la liste des erreurs à couvrir.

Par la suite, nous avons créé trois jeux de données différents (en faisant varier la taille du réseau) contenant à la fois des configurations valides et des configurations erronées, générées et étiquetées en se basant sur la modélisation de données établie. Nous avons utilisé ces jeux de données pour entraîner et évaluer trois modèles d'apprentissage de différents types : un modèle classique d'arbres de décision (DT), un modèle de

méthodes d'ensemble (RF) et un modèle de réseaux de neurones (MLP). Les résultats d'expérimentations ont montré de meilleures performances avec le modèle de réseaux de neurones. Cependant, des limites concernant la modélisation de données ont été constatées. La modélisation que nous avons adoptée ne permet pas un passage à l'échelle flexible. Pour maintenir des performances d'apprentissage satisfaisantes en cas d'augmentation de la taille du réseau, il est nécessaire d'augmenter la taille du jeu de données d'entraînement. De plus, la taille fixe du vecteur de propriétés nous a contraints de définir des hypothèses et des critères sur le nombre maximal de routeurs PE, sur le nombre maximal de clients et sur le nombre maximal de sites par client. Par conséquent, si un client dépasse le nombre maximal de sites distants, l'opérateur réseau devra réentraîner le modèle d'apprentissage avec un nouveau jeu de données en modifiant la valeur de ce nombre maximal. La même chose s'applique si l'opérateur dépasse le nombre maximal de clients ou le nombre maximal de routeurs PE.

Application des GNN pour détecter et localiser les erreurs de configuration

Dans la seconde partie, nous avons modifié la formulation du problème afin de détecter et de localiser les erreurs de configuration, et donc traiter la problématique initiale définie pour la thèse. Pour ce faire, nous avons choisi d'adopter une approche basée sur les réseaux de neurones de graphes (GNN). Ce choix est basé sur les conclusions de la première partie de notre travail. L'adoption des GNN nous a permis d'utiliser des modèles de réseaux neurones avec des données modélisée de manière dynamique et flexible à l'aide des graphes. Cela nous a donné la possibilité d'entraîner les modèles d'apprentissage sans avoir besoin de poser des contraintes ou des hypothèses sur le nombre de réseaux VPN ou sur leur taille.

Nous avons proposé, dans cette seconde partie, deux modèles GNN différents : le modèle de routage PE-PE et le modèle de routage CE-PE. Le modèle de routage PE-PE est dédié à la vérification de configuration inter-sites, c'est-à-dire, la détection et la localisation d'erreurs sur la partie de la configuration comprenant les paramètres et les politiques de routage des VRFs configurées sur les routeurs PE, qui contrôlent les interactions entre les différents sites clients. Nous avons considéré pour ce modèle deux architectures différentes de réseaux VPN : une architecture *full-mesh* simple et une architecture *hub-and-spoke* nécessitant la configuration de politiques de routages. Nous avons modélisé la configuration de chaque VPN dans un graphe dédié, ce qui nous a permis de vérifier la configuration de chaque VPN séparément. Pour ce modèle, l'objectif d'apprentissage consiste à effectuer une classification de nœuds afin de détecter et d'identifier les erreurs pour chaque routeur PE (chaque routeur PE est représenté par un nœud de graphe). Quant au modèle de routage CE-PE, il est dédié à la vérification de configuration sur la partie de routage site-backbone (CE-PE). Pour cela, nous avons modélisé chaque configuration de connexion CE-PE par un graphe contenant

deux nœuds (un nœud pour le routeur CE et un nœud pour le routeur PE). L'objectif d'apprentissage défini pour ce modèle est la classification de graphes. La combinaison de ces deux modèles nous permet de couvrir un plus grand nombre possible d'erreurs de configuration impactant la connectivité de bout-en-bout dans une architecture de réseaux VPN BGP/MPLS de niveau 3, indépendamment de la taille du réseau, du nombre de clients ou du nombre de sites par client.

Les résultats d'évaluation sont intéressants pour les deux modèles. Nous avons obtenu des valeurs de *F1-score* allant de 80% à 90% pour le modèle de routage PE-PE, et proches de 100% pour le modèle de routage CE-PE. De plus, les tests effectués sur des jeux de données contenant des réseaux VPN différents de ceux présents dans le jeu de données d'entraînement (c'est-à-dire, avec un nombre différent de sites et un placement différent sur les routeurs PE) ont indiqué une baisse de performance de seulement 4% pour le modèle de routage PE-PE, et aucune baisse de performance pour le modèle de routage CE-PE. Cela signifie que les deux modèles fonctionnent très bien sur des données non vues précédemment, et sont donc capables de généraliser leur traitement.

Pour terminer, nous avons proposé une architecture modulaire permettant d'intégrer l'outil de vérification de configuration dans un environnement de production.

Perspectives

Dans le cadre de cette thèse, plusieurs perspectives peuvent être envisagées pour enrichir et étendre les travaux réalisés. Nous décrivons ci-dessous quelques-unes des pistes possibles.

Considérer les diverses architectures de sites VPN

Il existe plusieurs architectures pour connecter un site VPN au *backbone* BGP/MPLS de l'opérateur réseaux (voir section 1.4.3 pour plus de détails). Dans notre travail, nous avons considéré uniquement les sites VPN simples comprenant un seul routeur CE connecté au réseau *backbone* via l'un des routeurs PE de l'opérateur réseau. Une amélioration du modèle de routage CE-PE proposé dans la seconde partie de nos contributions serait de modifier la modélisation des données et la formulation du problème pour prendre en compte les différentes architectures possibles de sites VPN.

Étendre la solution pour couvrir d'autres types de services

L'extension de la solution pour couvrir d'autres services que les réseaux VPN de niveau 3 constitue une perspective intéressante pour cette thèse. Les techniques d'appren-

tissage automatique et les modèles GNN proposés pour la vérification des configurations des réseaux VPN de niveau 3 pourraient être adaptés et appliqués à d'autres services réseaux, tels que l'accès Internet, les réseaux VPN de niveau 2, les réseaux « *Software Defined Wide Area Network (SD-WAN)* » ou d'autres technologies de routage et de commutation.

En étendant la solution à d'autres types de services, il serait possible d'élargir la portée et l'applicabilité des modèles proposés, offrant ainsi aux opérateurs réseaux une gamme plus large d'outils pour vérifier et valider les configurations de diverses infrastructures de réseaux et de services. De plus, cela permettrait d'identifier et de résoudre les problèmes de configuration dans des contextes variés, améliorant ainsi la qualité de service, la fiabilité et la disponibilité de l'ensemble du réseau.

Intégrer la solution dans l'environnement de production

À la fin de la thèse, nous avons proposé une architecture modulaire visant à intégrer l'outil de vérification de configuration dans l'environnement de production. Cette architecture comprend une interface API, deux modules de construction de graphes pour les parties de configuration de routage PE-PE et CE-PE, deux modules de vérification automatique de configuration pour les parties de routage PE-PE et CE-PE, ainsi qu'une base de données NoSQL orientée graphe pour stocker les configurations vérifiées et validées. Bien que nous n'ayons pas eu le temps de développer et de déployer l'ensemble de l'architecture proposée, certains composants ont été mis en place, tels qu'une première version de l'interface API, un service d'extraction des paramètres de configuration à partir de fichiers texte, et une première version de la base de données des configurations validées.

En tant que perspective, la finalisation de cette architecture modulaire pourrait être envisagée. L'achèvement et le déploiement de l'architecture permettraient une intégration plus fluide de l'outil de vérification dans l'environnement de production, offrant ainsi aux opérateurs réseaux un moyen efficace et automatisé de détecter et de résoudre les erreurs de configuration.

Bibliographie

[1] C. Servin et J. Arnaud, *Réseaux et télécoms*, 4e éd. Dunod, 2013.

[2] J. Kurose et K. Ross, *Computer Networking : A Top-down Approach*, 7e éd. Pearson, 2017.

[3] *The Network Admin's Guide to IP Transit*, Catchpoint, `https://www.catchpoint.com/network-admin-guide/ip-transit` (visité le 12 septembre 2022).

[4] B. Gregory, *Networking for Nerds : Do You Know the Difference between IP Peering vs. IP Transit for Enterprise Internet Interconnection ?* The Equinix Blog, 10 décembre 2018, `https://blog.equinix.com/blog/2018/12/10/networking-for-nerds-do-you-know-the-difference-between-ip-peering-vs-ip-transit-for-enterprise-internet-interconnection/?lang=ja` (visité le 12 septembre 2022).

[5] R. A. Steenbergen, *A Guide to Peering on the Internet*, NANOG 51, 30 janvier 2011, `https://www.slideshare.net/RichardSteenbergen/a-guide-to-peering-on-the-internet` (visité le 12 septembre 2022).

[6] N. Chatzis, G. Smaragdakis, A. Feldmann et W. Willinger, « There is More to IXPs than Meets the Eye, » *SIGCOMM Comput. Commun. Rev.*, t. 43, *5*, p. 19-28, nov. 2013. doi : 10.1145/2541468.2541473.

[7] D. Mills, *Exterior Gateway Protocol formal specification*, RFC 904, avr. 1984. doi : 10.17487/RFC0904.

[8] Y. Rekhter, S. Hares et T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC 4271, jan. 2006. doi : 10.17487/RFC4271.

[9] T. Li, R. Chandra et P. S. Traina, *BGP Communities Attribute*, RFC 1997, août 1996. doi : 10.17487/RFC1997.

[10] D. Tappan, S. R. Sangli et Y. Rekhter, *BGP Extended Communities Attribute*, RFC 4360, fév. 2006. doi : 10.17487/RFC4360.

[11] R. Chandra, T. J. Bates, Y. Rekhter et D. Katz, *Multiprotocol Extensions for BGP-4*, RFC 4760, jan. 2007. doi : 10.17487/RFC4760.

[12] J. Moy, *OSPF Version 2*, RFC 2328, avr. 1998. doi : 10.17487/RFC2328.

[13] A. Viswanathan, E. C. Rosen et R. Callon, *Multiprotocol Label Switching Architecture*, RFC 3031, jan. 2001. doi : 10.17487/RFC3031.

[14] K. Gurpreet et D. Kumar, « MPLS Technology on IP Backbone Network, » *International Journal of Computer Applications*, t. 5, août 2010. DOI : 10.5120/885-1257.

[15] B. Thomas, L. Andersson et I. Minei, *LDP Specification*, RFC 5036, oct. 2007. DOI : 10.17487/RFC5036.

[16] *Protocole MPLS*, FRAMEIP.COM, https://www.frameip.com/mpls/ (visité le 29 avril 2023).

[17] R. Callon, *Use of OSI IS-IS for routing in TCP/IP and dual environments*, RFC 1195, déc. 1990. DOI : 10.17487/RFC1195.

[18] R. Venkateswaran, « Virtual private networks, » *IEEE Potentials*, t. 20, *1*, p. 11-15, 2001. DOI : 10.1109/45.913204.

[19] *The OSI-Model in a simple way*, https://osi-model.com/ (visité le 25 octobre 2022).

[20] *L2VPN vs L3VPN*, Un brin de réseau, https://www.brindereseau.fr/?article=l2vpn-vs-l3vpn (visité le 25 octobre 2022).

[21] Y. Rekhter et E. C. Rosen, *BGP/MPLS IP Virtual Private Networks (VPNs)*, RFC 4364, fév. 2006. DOI : 10.17487/RFC4364.

[22] P. L. Higginson, B. Hinden, P. F. Hunt et al., *Virtual Router Redundancy Protocol*, RFC 2338, avr. 1998. DOI : 10.17487/RFC2338.

[23] Y. Li, X. Yin, Z. Wang et al., « A Survey on Network Verification and Testing With Formal Methods : Approaches and Challenges, » *IEEE Communications Surveys & Tutorials*, t. 21, *1*, p. 940-969, 2019. DOI : 10.1109/COMST.2018.2868050.

[24] N. Feamster et H. Balakrishnan, « Detecting BGP configuration faults with static analysis, » *in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI) - Volume 2*, 2005, p. 43-56.

[25] A. Fogel, S. Fung, L. Pedrosa et al., « A General Approach to Network Configuration Analysis, » *in Proccedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, Oakland, CA : USENIX Association, mai 2015, p. 469-483, ISBN : 978-1-931971-218.

[26] A. Gember-Jacobson, R. Viswanathan, A. Akella et R. Mahajan, « Fast Control Plane Analysis Using an Abstract Representation, » *in Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16)*, Florianopolis, Brazil : Association for Computing Machinery, 2016, p. 300-313, ISBN : 9781450341936. DOI : 10.1145/2934872.2934876.

[27] R. Beckett, A. Gupta, R. Mahajan et D. Walker, « A General Approach to Network Configuration Verification, » *in Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*, Los Angeles, CA, USA : Association for Computing Machinery, août 2017, p. 155-168, isbn : 9781450346535. doi : 10.1145/2541468.2541473.

[28] J. Selig, *What Is Machine Learning ? A Definition*, expert.ai, 14 mars 2022, https://www.expert.ai/blog/machine-learning-definition/ (visité le 21 octobre 2022).

[29] M. I. Jordan et T. M. Mitchell, « Machine learning : Trends, perspectives, and prospects, » *Science*, t. 349, *6245*, p. 255-260, 2015. doi : 10.1126/science.aaa8415.

[30] I. El Naqa et M. J. Murphy, « What Is Machine Learning ? » *In Machine Learning in Radiation Oncology : Theory and Applications*, I. El Naqa, R. Li et M. J. Murphy, éd. Springer International Publishing, 2015, p. 3-11. doi : 10.1007/978-3-319-18305-3_1.

[31] A. L. Samuel, « Some Studies in Machine Learning Using the Game of Checkers, » *IBM Journal of Research and Development*, t. 3, *3*, p. 210-229, 1959. doi : 10.1147/rd.33.0210.

[32] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997, isbn : 978-0-07-042807-2.

[33] E. Alpaydin, *Introduction to machine learning*, 3e éd. The MIT Press, 2014, isbn : 978-0-262-02818-9.

[34] M. Iqbal et Z. Yan, « Supervised machine learning approaches : a survey, » *International Journal of Soft Computing*, t. 5, p. 946-952, avr. 2015. doi : 10.21917/ijsc.2015.0133.

[35] T. Hastie, R. Tibshirani et J. Friedman, « Unsupervised Learning, » *in The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer New York, 2009, p. 485-585. doi : 10.1007/978-0-387-84858-7_14.

[36] R. S. Sutton et A. G. Barto, *Reinforcement learning : an introduction*, 2e éd. The MIT Press, 2018, isbn : 978-0-262-03924-6.

[37] B. Jijo et A. Mohsin Abdulazeez, « Classification Based on Decision Tree Algorithm for Machine Learning, » *Journal of Applied Science and Technology Trends*, t. 2, p. 20-28, jan. 2021. doi : 10.38094/jastt20165.

[38] S. Tangirala, « Evaluating the Impact of GINI Index and Information Gain on Classification using Decision Tree Classifier Algorithm*, » *International Journal of Advanced Computer Science and Applications*, t. 11, *2*, 2020. doi : 10.14569/IJACSA.2020.0110277.

[39] P. Geurts, A. Irrthum et L. Wehenkel, « Supervised learning with decision tree-based methods in computational and systems biology, » *Molecular BioSystems*, t. 5, *12*, p. 1593-1605, 2009. doi : 10.1039/B907946G.

[40] T. G. Dietterich, « Ensemble Methods in Machine Learning, » *in Multiple Classifier Systems (MCS 2000)*, Springer, Berlin, Heidelberg, 2000. doi : 10.1007/3-540-45014-9_1.

[41] G. Biau et E. Scornet, « A random forest guided tour, » *TEST 25*, p. 197-227, juin 2016. doi : 10.1007/s11749-016-0481-7.

[42] M. Ahmed, R. Seraj et S. M. S. Islam, « The k-means Algorithm : A Comprehensive Survey and Performance Evaluation, » *Electronics*, t. 9, *8*, 2020. doi : 10.3390/electronics9081295.

[43] P. Rai et S. Shubha, « A Survey of Clustering Techniques, » *International Journal of Computer Applications*, t. 7, oct. 2010. doi : 10.5120/1326-1808.

[44] C. Kim, *Quick Guide to K-Means Clustering with Python example (Scikit-learn)*, Medium, 28 juillet 2022, https://medium.com/@chyun55555/quick-guide-to-k-means-clustering-with-python-example-scikit-learn-6efe8e319893 (visité le 10 novembre 2022).

[45] Y. LeCun, Y. Bengio et G. Hinton, « Deep learning, » *Nature*, t. 521, *7553*, p. 436-444, 2015. doi : 10.1038/nature14539.

[46] I. Goodfellow, Y. Bengio et A. Courville, *Deep Learning*. MIT Press, 2016, isbn : 9780262035613.

[47] G. E. Hinton, S. Osindero et Y.-W. Teh, « A fast learning algorithm for deep belief nets, » *Neural computation*, t. 18, *7*, p. 1527-1554, 2006.

[48] J. Yosinski, J. Clune, Y. Bengio et H. Lipson, « How transferable are features in deep neural networks ? » *In Advances in Neural Information Processing Systems (NIPS)*, 2014, p. 3320-3328.

[49] B. Maurice, *Fonctionnement du neurone artificiel*, Deeply Learning, 22 septembre 2018, https://deeplylearning.fr/cours-theoriques-deep-learning/fonctionnement-du-neurone-artificiel/ (visité le 21 avril 2023).

[50] D. E. Rumelhart, G. E. Hinton et R. J. Williams, « Learning representations by back-propagating errors, » *Nature*, t. 323, *6088*, p. 533-536, 1986. doi : 10.1038/323533a0.

[51] X. Glorot, A. Bordes et Y. Bengio, « Deep sparse rectifier neural networks, » *in Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, p. 315-323.

[52] C. M. Bishop, *Pattern recognition and machine learning*. Information Science et Statistics, 2006, isbn : 9780387310732.

[53] S. Ruder, « An overview of gradient descent optimization algorithms, » *arXiv preprint arXiv :1609.04747*, 2016.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov, « Dropout : A simple way to prevent neural networks from overfitting, » *Journal of Machine Learning Research*, t. 15, *1*, p. 1929-1958, 2014.

[55] S. Ioffe et C. Szegedy, « Batch normalization : Accelerating deep network training by reducing internal covariate shift, » *in Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015, p. 448-456.

[56] F. Rosenblatt, « The perceptron : A probabilistic model for information storage and organization in the brain, » *Psychological Review*, t. 65, *6*, p. 386-408, 1958. doi : 10.1037/h0042519.

[57] H. Taud et J. Mas, « Multilayer Perceptron (MLP), » *in Geomatic Approaches for Modeling Land Change Scenarios*. Springer International Publishing, 2018, p. 451-455. doi : 10.1007/978-3-319-60801-3_27.

[58] D. P. Kingma et J. Ba, « Adam : A method for stochastic optimization, » *in Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[59] *Histoire du Deep Learning*, Natural solutions, 09 avril 2018, https://www.natural-solutions.eu/blog/histoire-du-deep-learning (visité le 21 avril 2023).

[60] K. Simonyan et A. Zisserman, « Very deep convolutional networks for large-scale image recognition, » *in Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[61] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang et S. Y. Philip, « A Comprehensive Survey on Graph Neural Networks, » *IEEE Transactions on Neural Networks and Learning Systems*, t. 32, *1*, p. 4-24, 2020. doi : 10.1109/TNNLS.2020.2978386.

[62] Y. LeCun, L. Bottou, Y. Bengio et P. Haffner, « Gradient-based learning applied to document recognition, » *Proceedings of the IEEE*, t. 86, *11*, p. 2278-2324, 1998. doi : 10.1109/5.726791.

[63] A. Krizhevsky, I. Sutskever et G. E. Hinton, « Imagenet classification with deep convolutional neural networks, » *in Advances in neural information processing systems*, 2012, p. 1097-1105.

[64] R. Lambert, *Focus : Le Réseau de Neurones Convolutifs*, Pensée Artificielle, 11 janvier 2019, https://penseeartificielle.fr/focus-reseau-neurones-convolutifs/ (visité le 20 avril 2023).

[65] R. Girshick, J. Donahue, T. Darrell et J. Malik, « Rich feature hierarchies for accurate object detection and semantic segmentation, » *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, p. 580-587. doi : 10.1109/CVPR.2014.81.

[66] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals et G. E. Dahl, « Neural Message Passing for Quantum Chemistry, » *in Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017, p. 1263-1272.

[67] T. N. Kipf et M. Welling, « Semi-Supervised Classification with Graph Convolutional Networks, » *in Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[68] M. Abadi, A. Agarwal, P. Barham et al., *TensorFlow : Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015.

[69] Y. Li, D. Tarlow, M. Brockschmidt et R. Zemel, « Gated Graph Sequence Neural Networks, » *in Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[70] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò et Y. Bengio, « Graph Attention Networks, » *in Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[71] S. Hochreiter et J. Schmidhuber, « Long Short-Term Memory, » *Neural Computation*, t. 9, *8*, p. 1735-1780, 1997. doi : 10.1162/neco.1997.9.8.1735.

[72] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner et G. Monfardini, « The Graph Neural Network Model, » *IEEE Transactions on Neural Networks*, t. 20, *1*, p. 61-80, 2009. doi : 10.1109/TNN.2008.2005605.

[73] J. L. Elman, « Finding structure in time, » *Cognitive Science*, t. 14, *2*, p. 179-211, 1990. doi : 10.1207/s15516709cog1402_1.

[74] M. Wang, Y. Cui, X. Wang, S. Xiao et J. Jiang, « Machine Learning for Networking : Workflow, Advances and Opportunities, » *IEEE Network*, t. 32, *2*, p. 92-99, 2018. doi : 10.1109/MNET.2017.1700200.

[75] R. Boutaba, M. Salahuddin, N. Limam et al., « A Comprehensive Survey on Machine Learning for Networking : Evolution, Applications and Research Opportunities, » *Journal of Internet Services and Applications 9*, 2018. doi : 10.1186/s13174-018-0087-2.

[76] Z. Chen, J. Wen et Y. Geng, « Predicting future traffic using Hidden Markov Models, » *in IEEE 24th International Conference on Network Protocols (ICNP)*, 2016, p. 1-6. doi : 10.1109/ICNP.2016.7785328.

[77] P. Poupart, Z. Chen, P. Jaini et al., « Online flow size prediction for improved network routing, » *in 2016 IEEE 24th International Conference on Network Protocols (ICNP)*, 2016, p. 1-6. doi : 10.1109/ICNP.2016.7785324.

[78] Z. Aouini, A. Kortebi, Y. Ghamri-Doudane et I. L. Cherif, « Early classification of residential networks traffic using C5.0 machine learning algorithm, » *in 2018 Wireless Days (WD)*, 2018, p. 46-53. doi : 10.1109/WD.2018.8361693.

[79] R. Chauhan et S. Kumar, « Packet Loss Prediction Using Artificial Intelligence Unified with Big Data Analytics, Internet of Things and Cloud Computing Technologies, » *in 2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, 2021, p. 01-06. doi : 10.1109/ISCON52037.2021.9702517.

[80] Z. Ge, J. Hou et A. Nayak, « GNN-based End-to-end Delay Prediction in Software Defined Networking, » *in 2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2022, p. 372-378. doi : 10.1109/DCOSS54816.2022.00066.

[81] Z. Wang, M. Zhang, D. Wang et al., « Failure prediction using machine learning and time series in optical network, » *Opt. Express*, t. 25, *16*, p. 18 553-18 565, août 2017. doi : 10.1364/OE.25.018553.

[82] S. Shahkarami, F. Musumeci, F. Cugini et M. Tornatore, « Machine-Learning-Based Soft-Failure Detection and Identification in Optical Networks, » *in 2018 Optical Fiber Communications Conference and Exposition (OFC)*, 2018, p. 1-3.

[83] C. Wang, L. Zhang, Z. Li et C. Jiang, « SDCoR : Software Defined Cognitive Routing for Internet of Vehicles, » *IEEE Internet of Things Journal*, t. 5, *5*, p. 3513-3520, 2018. doi : 10.1109/JIOT.2018.2812210.

[84] M. Kiran, B. Mohammed et N. Krishnaswamy, « DeepRoute : Herding Elephant and Mice Flows with Reinforcement Learning, » *in Machine Learning for Networking (MLN 2019)*, S. Boumerdassi, É. Renault et P. Mühlethaler, éd., Springer International Publishing, 2020, p. 296-314. doi : 10.1007/978-3-030-45778-5_20.

[85] M. Bahnasy, F. Li, S. Xiao et X. Cheng, « DeepBGP : A Machine Learning Approach for BGP Configuration Synthesis, » *in Proceedings of the Workshop on Network Meets AI & ML (NetAI '20)*, août 2020, p. 48-55. doi : 10.1145/3405671.3405816.

[86] F. Geyer et S. Schmid, « DeepMPLS : Fast Analysis of MPLS Configurations Using Deep Learning, » *in 2019 IFIP Networking Conference (IFIP Networking)*, 2019, p. 1-9. doi : 10.23919/IFIPNetworking.2019.8816842.

[87] A. Nagaraja, S. Aljawarneh et P. H. S, « PAREEKSHA : A Machine Learning Approach for Intrusion and Anomaly Detection, » *in Proceedings of the First International Conference on Data Science, E-Learning and Information Systems*, 2018. doi : 10.1145/3279996.3280032.

[88] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin et J. Aguilar, « Towards the Deployment of Machine Learning Solutions in Network Traffic Classification : A Systematic Survey, » *IEEE Communications Surveys & Tutorials*, t. 21, *2*, p. 1988-2014, 2019. doi : 10.1109/COMST.2018.2883147.

[89] T. Zhang et S. Mao, « Machine Learning for End-to-End Congestion Control, » *IEEE Communications Magazine*, t. 58, *6*, p. 52-57, 2020. doi : 10.1109/MCOM.001.1900509.

[90] H. Jiang, Q. Li, Y. Jiang et al., « When machine learning meets congestion control : A survey and comparison, » *Computer Networks*, t. 192, 2021. doi : 10.1016/j.comnet.2021.108033.

[91] M. Nouioua, P. Fournier-Viger, G. He, F. Nouioua et Z. Min, « A Survey of Machine Learning for Network Fault Management, » *in Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics : Theories and Applications*. Springer International Publishing, 2021, p. 1-27. doi : 10.1007/978-3-030-66288-2_1.

[92] Z. Mammeri, « Reinforcement Learning Based Routing in Networks : Review and Classification of Approaches, » *IEEE Access*, t. 7, p. 55 916-55 950, 2019. doi : 10.1109/ACCESS.2019.2913776.

[93] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky et S. Uhlig, « Software-Defined Networking : A Comprehensive Survey, » *Proceedings of the IEEE*, t. 103, *1*, p. 14-76, 2015. doi : 10.1109/JPROC.2014.2371999.

[94] F. Yang, S. Wang, J. Li, Z. Liu et Q. Sun, « An overview of Internet of Vehicles, » *China Communications*, t. 11, *10*, p. 1-15, 2014. doi : 10.1109/CC.2014.6969789.

[95] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters et J. Schmidhuber, « Natural Evolution Strategies, » *Journal of Machine Learning Research*, t. 15, *27*, p. 949-980, 2014.

[96] T. Salimans, J. Ho, X. Chen, S. Sidor et I. Sutskever, « Evolution Strategies as a Scalable Alternative to Reinforcement Learning, » 2017. doi : 10.48550/ARXIV.1703.03864.

[97] F. Stulp et O. Sigaud, « Robot Skill Learning : From Reinforcement Learning to Evolution Strategies, » *Paladyn, Journal of Behavioral Robotics*, t. 4, *1*, p. 49-61, 2013. doi : 10.2478/pjbr-2013-0003.

[98] W. Jiang, « Graph-based deep learning for communication networks : A survey, » *Computer Communications*, t. 185, p. 40-54, 2022. doi : 10.1016/j.comcom.2021.12.015.

[99] D. Pujol-Perich, J. Suárez-Varela, M. Ferriol et al., « IGNNITION : Bridging the Gap between Graph Neural Networks and Networking Systems, » *IEEE Network*, t. 35, *6*, p. 171-177, 2021. doi : 10.1109/MNET.001.2100266.

[100] D. P. Giakatos, S. Kostoglou, P. Sermpezis et A. Vakali, « Benchmarking Graph Neural Networks for Internet Routing Data, » 2022. doi : 10.48550/ARXIV.2210.14189.

[101] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros et A. Cabellos-Aparicio, « Unveiling the Potential of Graph Neural Networks for Network Modeling and Optimization in SDN, » *in Proceedings of the 2019 ACM Symposium on SDN Research (SOSR '19)*, Association for Computing Machinery, 2019, p. 140-151. doi : 10.1145/3314148.3314357.

[102] F. Geyer et G. Carle, « Learning and Generating Distributed Routing Protocols Using Graph-Based Deep Learning, » *in Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks (Big-DAMA '18)*, Association for Computing Machinery, 2018, p. 40-45. doi : 10.1145/3229607.3229610.

[103] *AS-rank dataset*, CAIDA, https://asrank.caida.org/ (visité le 07 décembre 2022).

[104] *AS-relationships dataset*, CAIDA, https://publicdata.caida.org/datasets/as-relationships/ (visité le 07 décembre 2022).

[105] *The Interconnection Database*, PeeringDB, https://www.peeringdb.com/ (visité le 07 décembre 2022).

[106] *AS hegemony*, Internet Health Report, https://ihr.iijlab.net/ihr/hegemony/ (visité le 07 décembre 2022).

[107] *Country-level Transit Influence (CTI)*, CAIDA, https://github.com/CAIDA/mapkit-cti-code (visité le 07 décembre 2022).

[108] *Stanford ASdb Dataset*, Stanford University, https://asdb.stanford.edu/ (visité le 07 décembre 2022).

[109] W. Hamilton, Z. Ying et J. Leskovec, « Inductive Representation Learning on Large Graphs, » *in Advances in Neural Information Processing Systems*, t. 30, 2017.

[110] T. N. Kipf et M. Welling, « Semi-Supervised Classification with Graph Convolutional Networks, » 2016. doi : 10.48550/ARXIV.1609.02907.

[111] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò et Y. Bengio, « Graph Attention Networks, » 2017. doi : 10.48550/ARXIV.1710.10903.

[112] A. Grover et J. Leskovec, « Node2vec : Scalable Feature Learning for Networks, » *in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, Association for Computing Machinery, 2016, p. 855-864. doi : 10.1145/2939672.2939754.

[113] T. Shapira et Y. Shavitt, « BGP2Vec : Unveiling the Latent Characteristics of Autonomous Systems, » *IEEE Transactions on Network and Service Management*, 2022. doi : 10.1109/TNSM.2022.3169638.

[114] E.-H. Mohammedi, E. Lavinal et G. Fleury, « Configuration Faults Detection in IP Virtual Private Networks Based on Machine Learning, » *in Machine Learning for Networking (MLN'2020)*, Springer, 2021, p. 40-56. doi : 10.1007/978-3-030-70866-5_3.

[115] G. Tsoumakas et I. Katakis, « Multi-label classification : An overview, » *International Journal of Data Warehousing and Mining (IJDWM)*, t. 3, *3*, p. 1-13, 2007. doi : 10.4018/jdwm.2007070101.

[116] F. Hvilshøj, *An Introduction to Balanced and Imbalanced Datasets in Machine Learning*, Encord, 14 septembre 2022, https://encord.com/blog/an-introduction-to-balanced-and-imbalanced-datasets-in-machine-learning (visité le 23 janvier 2023).

[117] F. Pedregosa, G. Varoquaux, A. Gramfort et al., « Scikit-Learn : Machine Learning in Python, » *The journal of Machine Learning Research*, t. 12, p. 2825-2830, nov. 2011.

[118] *sklearn.tree.DecisionTreeClassifier*, Scikit-learn documentation, https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (visité le 01 février 2023).

[119] *sklearn.ensemble.RandomForestClassifier*, Scikit-learn documentation, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (visité le 01 février 2023).

[120] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever et R. Salakhutdinov, « Dropout : A Simple Way to Prevent Neural Networks from Overfitting, » *Journal of Machine Learning Research*, t. 15, *56*, p. 1929-1958, 2014. adresse : http://jmlr.org/papers/v15/srivastava14a.html.

[121] W. Koehrsen, *Overfitting vs. Underfitting : A Complete Example*, Towards Data Science, 28 janvier 2018, https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765 (visité le 09 janvier 2023).

[122] A. Gulli et S. Pal, *Deep Learning with Keras*. Packt Publishing, 2017, isbn : 978-1-78712-903-0.

[123] M. Abadi, P. Barham, J. Chen et al., « TensorFlow : A System for Large-Scale Machine Learning, » *in Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI 2016)*, USENIX Association, 2016, p. 265-283.

[124] E.-H. Mohammedi, E. Lavinal et G. Fleury, « Detecting and locating configuration errors in IP VPNs with Graph Neural Networks, » *in IEEE/IFIP Network Operations and Management Symposium (NOMS 2022)*, 2022, p. 1-6. doi : 10.1109/NOMS54207.2022.9789800.

[125] D. Grattarola et C. Alippi, « Graph Neural Networks in TensorFlow and Keras with Spektral [Application Notes], » *IEEE Computational Intelligence Magazine*, t. 16, *1*, p. 99-106, 2021. doi : 10.1109/MCI.2020.3039072.

[126] M. Simonovsky et N. Komodakis, « Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs, » *in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 29-38. doi : 10.1109/CVPR.2017.11.

[127] *Template Text Parser documentation*, https://ttp.readthedocs.io/en/latest/index.html (visité le 07 avril 2023).

[128] *Neo4j Graph Database*, https://neo4j.com/product/neo4j-graph-database/ (visité le 10 avril 2023).