Arts et Métiers Sciences et Technologies

ÉCOLE DOCTORALE SCIENCES ET MÉTIERS DE L'INGÉNIEUR

Laboratoire d'Innovation Numérique pour les Entreprises et les Apprentissages au service de la Compétitivité des Territoires (CESI LINEACT UR 7527) - Campus de Paris

THÈSE

Présentée par : **Mortada TERMOS**
Soutenue le : **21 Janvier 2026**

Pour obtenir le diplôme de **Doctorat** *délivré par* :

L'École Nationale Supérieure d'Arts et Métiers

Spécialité : **Informatique**

Intrusion Detection in IoT Networks Using Complex Networks and AI Techniques

Détection d'intrusions dans les réseaux IoT par réseaux complexes et techniques d'IA

THÈSE dirigée par :

Mourad ZGHAL, Enseignant Chercheur HDR
Ali JABER, Professeur

et co-encadrée par :

Mohamed-el-Amine BRAHMIA, Enseignant Chercheur HDR
Zakariya GHALMANE, Enseignant Chercheur

Jury

M. Pascal LORENZ	Professeur, Université de Haute-Alsace	Président
Mme. Lynda MOKDAD	Professeure, Université Paris-Est Créteil	Rapporteure
M. Rida KHATOUN	Professeur, Télécom ParisTech	Rapporteur
Mme. Hamida SEBA	Professeure, Université Claude-Bernard Lyon 1	Examinatrice
M. Mourad ZGHAL	Enseignant Chercheur (HDR), CESI École d'ingénieurs	Directeur
M. Ali JABER	Professeur, Université Libanaise	Co-directeur
M. Zakariya GHALMANE	Enseignant Chercheur, CESI École d'ingénieurs	Co-encadrant
M. Mohamed-el-Amine BRAHMIA	Enseignant Chercheur (HDR), CESI École d'ingénieurs	Co-encadrant
M. Ahmad FADALLAH	Enseignant Chercheur, Université des Sciences et des Arts du Liban	Invité

THÈSE

À ceux qui me sont chers,

Mortada

Acknowledgments

Je tiens à exprimer ma profonde gratitude à mes Directeurs de thèse, *Pr. Mourad ZGHAL* et *Pr. Ali JABER*, pour leur guidance scientifique, leur disponibilité et leur confiance tout au long de ce travail.

J'adresse mes sincères remerciements à mes Co-encadrants, *Dr. Mohamed-el-Amine BRAHMIA*, *Dr. Zakariya GHALMANE* et *Dr. Ahmad FADLALLAH*. Leurs échanges techniques, leurs conseils méthodologiques, leurs relectures attentives et leur exigence bienveillante ont structuré ma démarche de recherche et enrichi la qualité du manuscrit.

Je remercie chaleureusement les membres du jury: *Pr. Lynda MOKDAD* (Rapporteure), *Pr. Rida KHATOUN* (Rapporteur), *Pr. Hamida SEBA* (Examinatrice) et *Pr. Pascal LORENZ* (Examinateur), pour l'honneur qu'ils me font d'évaluer ce travail, pour la lecture attentive du manuscrit et pour leurs remarques constructives.

Je remercie également les institutions qui ont accueilli et soutenu ces recherches : l'*École Nationale Supérieure d'Arts et Métiers (ENSAM), Paris* au sein de l'*École doctorale SMI 432*, l'*Université Libanaise, Beyrouth* au sein de la *Doctoral School of Science & Technology (DSST)*, ainsi que les laboratoires *CESI LINEACT UR 7527, Strasbourg* et *PRASE, Beyrouth*, pour la qualité de l'environnement scientifique et des conditions de travail.

Enfin, j'adresse un remerciement tout particulier à ma famille et à mes amis pour leur soutien constant, leur patience et leurs encouragements. Leur présence à mes côtés dans les moments décisifs, leurs mots rassurants dans les périodes de doute et leur confiance indéfectible ont rendu cette aventure possible. À celles et ceux qui, de près ou de loin, ont contribué par un conseil, un sourire ou un geste de bienveillance : merci.

Résumé

Les systèmes de détection d'intrusions réseau (NIDS) surveillent le trafic afin d'identifier des activités malveillantes. Dans l'Internet des Objets (IdO), la grande échelle, la diversité des dispositifs et les ressources limitées rendent la détection particulièrement difficile. Une autre limite de nombreuses approches antérieures est de traiter le trafic comme des enregistrements tabulaires indépendants, en exploitant peu le contexte de graphe qui relie les appareils du réseau. Cette thèse adopte une approche "structure-first" : nous construisons des graphes à partir des flux (nœuds : adresses IP/équipements, arêtes : communications), analysons la structure du réseau via des mesures de centralité, puis utilisons ces signaux de centralité pour augmenter l'ensemble des caractéristiques de flux. Plus précisément, nous (i) intégrons la centralité dans des modèles d'apprentissage profond au moyen d'un pipeline *Graph Deep Learning based on Centrality* (GDLC), (ii) étudions l'apprentissage fédéré où un apprentissage local exploitant la structure du réseau améliore l'agrégation sans partage des données brutes, et (iii) concevons des réseaux de neurones de graphes (GNN) centrés sur les arêtes et adaptés aux propriétés du réseau. Sur des jeux de données largement utilisés, les caractéristiques de centralité ajoutent un pouvoir discriminant aux modèles profonds, qui détectent les attaques avec une confiance accrue. De plus, un apprentissage fédéré informé par la structure améliore la généralisation entre sites. En outre, les mesures de réseaux complexes permettent d'augmenter l'exactitude des GNN en enrichissant l'initialisation des embeddings et en guidant le choix des architectures. La thèse apporte ainsi une contribution fondée sur les graphes aux NIDS, où le pipeline proposé améliore de manière systématique la détection et la généralisabilité dans des contextes IdO.

Mots-clés : Cybersécurité, Systèmes de détection d'intrusions réseau, Internet des Objets, Réseaux complexes, Mesures de centralité, Apprentissage fédéré, Réseaux de neurones de graphes.

Abstract

Network Intrusion Detection Systems (NIDS) monitor network traffic to identify malicious activities. In Internet-of-Things (IoT) environments, large scale, device diversity, and constrained resources make detection particularly challenging. A further limitation of many prior approaches is the treatment of traffic as independent, tabular records, with little use of the communication graph context that links devices in the network. This thesis adopts a structure-first approach: we construct graphs from flows (nodes as IPs/devices, edges as communications), analyze network structure with centrality measures, and use centrality-derived features to augment the flow-feature set. Specifically, we (i) integrate centrality into deep learning models through a Graph Deep Learning based on Centrality (GDLC) pipeline, (ii) study federated training where local learning that uses network structure improves aggregation without sharing raw data, and (iii) design edge-centric graph neural networks (GNNs) tailored according to the properties of the network. Across widely used benchmarks, centrality features proved to add discriminative capabilities to deep learning models, which were able to detect attacks more confidently. Also, federated learning that uses the network structure improves generalization across sites. Furthermore, complex network measures were capable of improving GNNs' accuracy by augmenting the initialized embeddings and choosing suitable architectures. The thesis contributes a graph-based addition to NIDS, where the proposed graph-based pipeline consistently improves detection and generalizability in NIDS for IoT.

Keywords: Cybersecurity, Network Intrusion Detection Systems, Internet of Things, Complex Networks, Centrality Measures, Federated Learning, Graph Neural Networks.

Résumé long

Cette thèse s'inscrit dans le domaine de la cybersécurité des réseaux, et plus particulièrement dans la détection d'intrusions dans des environnements modernes marqués par l'essor de l'Internet des Objets (IoT), l'hétérogénéité des équipements connectés et l'augmentation constante du volume de trafic. Dans ces contextes largement distribués, les systèmes traditionnels de détection d'intrusions montrent leurs limites : dépendance aux signatures connues, incapacité à capturer des attaques coordonnées, forte sensibilité aux dérives de données et nécessité de centraliser les journaux réseau, ce qui pose des problèmes de confidentialité et de passage à l'échelle.

Face à ces défis, cette thèse explore l'apport combiné (i) de l'analyse des réseaux complexes, (ii) de l'apprentissage profond, (iii) de l'apprentissage fédéré, et (iv) des réseaux de neurones sur graphes (GNN) pour concevoir des systèmes de détection capables d'exploiter la structure relationnelle du trafic et d'opérer dans des environnements décentralisés. L'objectif est de dépasser les approches qui traitent les flux réseau comme des enregistrements indépendants, en modélisant explicitement les interactions entre dispositifs au moyen de graphes et de mesures topologiques.

La thèse propose ainsi une série de contributions complémentaires:

1. un cadre d'apprentissage profond enrichi par des mesures de centralité dérivées des réseaux complexes
2. une extension fédérée permettant d'intégrer ces mesures dans des scénarios distribués sous données non-IID, tout en préservant la confidentialité
3. des architectures GNN adaptées aux graphes de flux réseau, capables de tirer parti de l'information structurelle pour améliorer la détection des comportements anormaux.

Les résumés présentés dans les sections suivantes restituent, en langue française, les principales

idées, méthodologies et résultats des chapitres rédigés en anglais, et offrent une vue d'ensemble cohérente de l'évolution des travaux au fil du manuscrit.

Résumé du Chapitre I: État de l'art

Ce chapitre pose les fondations scientifiques et techniques de la thèse en retraçant l'évolution des systèmes de détection d'intrusion réseau (NIDS), depuis les approches classiques fondées sur des signatures jusqu'aux méthodes modernes basées sur l'intelligence artificielle (IA), l'apprentissage fédéré (FL), les réseaux de neurones sur graphes (GNN) et l'analyse de réseaux complexes. Il fournit le cadre conceptuel, terminologique et notationnel nécessaire à la compréhension des contributions proposées ultérieurement, et justifie les choix de modélisation, de construction de graphes et de jeux de données qui seront utilisés dans les Chapitres II à IV.

Dans un premier temps, le chapitre rappelle ce qu'est un NIDS, ses objectifs et ses principaux paradigmes de détection : approche par signatures, détection d'anomalies et approches basées sur l'IA. Les NIDS surveillent le trafic réseau au niveau de points d'observation stratégiques afin d'identifier des activités non autorisées ou malveillantes. Historiquement, les premiers systèmes reposaient sur des règles statiques et des seuils statistiques, en comparant les paquets ou les flux à des modèles d'attaque connus. Ce paradigme de détection par signatures offre une grande précision et peu de faux positifs pour les attaques connues, mais nécessite des mises à jour fréquentes et reste inopérant face aux attaques inédites, polymorphes ou fortement opaques. À l'inverse, les approches par anomalies construisent un modèle de comportement "normal" du réseau et déclenchent des alertes lorsque des écarts significatifs sont observés. Elles permettent, en principe, de détecter des attaques zéro-day ou peu fréquentes, mais restent confrontées à des difficultés majeures : dérive de concept (évolution du trafic légitime dans le temps), forte proportion de faux positifs, déséquilibre entre classes bénignes et malveillantes, manque d'explicabilité et vulnérabilité à des attaques qui imitent le comportement normal.

Face à ces limites, le chapitre introduit les approches basées sur l'IA comme un tournant important dans la conception des NIDS. Les premières générations de NIDS "intelligents" se sont appuyées sur des algorithmes d'apprentissage automatique classiques (arbres de décision, forêts aléatoires, SVM, Naive Bayes, régressions logistiques, méthodes ensemblistes, etc.). Ces modèles exploitent des vecteurs de

caractéristiques construits à partir de statistiques de flux, d'attributs protocolaires ou de mesures temporelles, et ont permis d'obtenir de très bonnes performances sur des jeux de données de référence tels que NSL-KDD, UNSW-NB15, CIC-IDS-2017 ou CIC-DDoS2019. Le chapitre souligne toutefois que ces modèles reposent fortement sur la qualité de l'ingénierie de variables et font l'hypothèse implicite que les enregistrements sont indépendants et identiquement distribués (IID), ce qui néglige la structure temporelle et topologique du trafic.

L'émergence de l'apprentissage profond (Deep Learning, DL) a ensuite profondément renouvelé la manière de concevoir les NIDS. Les réseaux de neurones profonds et les perceptrons multicouches (MLP) ont ouvert la voie à une extraction automatique de caractéristiques à partir de données de trafic de grande dimension, réduisant ainsi la dépendance à l'expertise manuelle. Les MLP et DNN se sont montrés particulièrement efficaces pour traiter de grands espaces de caractéristiques, notamment dans des environnements IoT, avec des performances dépassant souvent celles des méthodes classiques. Par la suite, les réseaux de neurones convolutifs (CNN) ont été utilisés pour capturer des corrélations locales et des motifs spatiaux entre caractéristiques, tandis que les architectures récurrentes (RNN, LSTM, GRU, BiLSTM) ont permis de modéliser les dépendances temporelles dans des séquences de paquets ou de flux. Des modèles hybrides de type CNN–LSTM combinent ces deux perspectives en extrayant des motifs locaux avant de modéliser leur évolution séquentielle, et des autoencodeurs empilés ont été employés pour la détection non supervisée d'anomalies via la reconstruction de trafic normal.

Le chapitre met également en évidence les hypothèses sous-jacentes à ces modèles profonds (distributions de données relativement stables entre entraînement et déploiement, disponibilité de grands ensembles labellisés, nécessité d'un prétraitement soigné des données) et leurs limitations pratiques : coût de calcul et énergétique important, difficulté à expliquer les décisions, sensibilité aux dérives de données et aux attaques adversariales, nécessité d'un réglage fin d'hyperparamètres et de pipelines d'ingestion robustes. Malgré ces contraintes, l'apprentissage profond a significativement amélioré les performances des NIDS, en particulier sur des jeux de données récents et riches, mais il reste encore difficile à déployer dans des environnements distribués et contraints en ressources.

Dans un second axe, le chapitre présente l'apprentissage fédéré (Federated Learning, FL) appliqué à la détection d'intrusion. L'apprentissage fédéré permet d'entraîner un modèle global sur des données géographiquement ou organisationnellement dispersées sans centraliser les données brutes. Chaque

client (passerelle IoT, véhicule autonome, système industriel, organisation...) effectue des mises à jour locales du modèle à partir de son propre trafic, puis envoie uniquement des gradients ou des paramètres à un serveur d'agrégation. Cette approche répond aux contraintes de confidentialité, de souveraineté des données, de bande passante et d'échelle rencontrées dans les réseaux modernes. Le chapitre décrit les variantes de FL (scénarios cross-device et cross-silo), les algorithmes d'agrégation tels que FedAvg et ses variantes (FedProx, FedNova, méthodes adaptatives), ainsi que leurs applications à des NIDS pour 5G, IoT et systèmes cyber-physiques. Des travaux récents montrent qu'il est possible d'obtenir des performances proches de l'apprentissage centralisé tout en conservant la localité des données et en partageant uniquement un "savoir" agrégé.

Cependant, l'apprentissage fédéré introduit de nouveaux défis : hétérogénéité (non-IID) des données entre clients, disparités de ressources, communication limitée, clients malveillants ou non fiables, fuites potentielles d'information à partir des gradients et attaques de type empoisonnement. Le chapitre insiste sur la nécessité de mécanismes d'agrégation robustes, de schémas de privatisation (quantification, bruit différentiel) et de conceptions de modèles adaptés à des dispositifs contraints. Il note également que, dans la littérature existante, les scénarios expérimentaux restent souvent simplifiés (clients simulés à partir d'un seul jeu de données) et ne rendent pas toujours compte de la diversité réelle des environnements où un NIDS fédéré serait déployé.

Le chapitre aborde ensuite la montée en puissance des réseaux de neurones sur graphes (Graph Neural Networks, GNN) pour la détection d'intrusion. Contrairement aux approches vectorielles qui traitent chaque flux comme un échantillon indépendant, les GNN exploitent explicitement la structure de graphe sous-jacente au trafic réseau : les entités (adresses IP, flux, sessions...) sont modélisées comme des nœuds et leurs interactions (communications, similarités, proximités temporelles) comme des arêtes. À travers des mécanismes de propagation de messages et d'agrégation locale, les GNN produisent des représentations de nœuds ou d'arêtes qui intègrent l'information contextuelle de voisinage et, sur plusieurs couches, des propriétés plus globales du graphe. Le chapitre rappelle les premiers travaux en apprentissage de représentations sur graphes, puis détaille des modèles plus récents tels que GraphSAGE et ses extensions adaptées aux flux réseau.

Plusieurs architectures spécifiques sont discutées, comme E-GraphSAGE, qui intègrent des caractéristiques d'arêtes issues des flux (par exemple, statistiques temporelles ou protocolaires) et utilisent des constructions de graphes basées sur les relations entre IP. Ces modèles ont montré des performances

très élevées sur des jeux de données récents (BoT-IoT, TON_IOT, CIC-IDS-2017, UNSW-NB15, etc.) en exploitant la dimension relationnelle du trafic. Le chapitre souligne cependant que la plupart de ces travaux supposent une topologie de graphe fixe durant l'entraînement, que la construction du graphe (choix des nœuds, définition des arêtes, fenêtrage temporel) joue un rôle crucial dans la qualité des résultats et que le passage à des graphes dynamiques ou à très grande échelle demeure un problème ouvert. De plus, une partie des jeux de données sur graphes disponibles reste limitée en taille ou présente des biais (par exemple, certains nœuds peuvent être presque identifiables par leur rôle dans la topologie), ce qui complique la généralisation.

Afin d'ancrer ces modèles dans un cadre théorique plus large, le chapitre introduit ensuite la notion de réseaux complexes. Un réseau complexe est un graphe dont la structure de connexions est ni parfaitement régulière ni purement aléatoire; il s'agit généralement de systèmes très interconnectés et fortement hétérogènes, tels que les réseaux sociaux, les graphes du Web, les réseaux biologiques ou les infrastructures de communication. Le chapitre rappelle les concepts de petits mondes, de réseaux sans échelle, et insiste sur la détection de communautés, c'est-à-dire de groupes de nœuds densément connectés entre eux et faiblement reliés au reste du graphe. Les communautés révèlent une organisation à un niveau méso-structural, au-delà des nœuds individuels, et sont essentielles pour interpréter des phénomènes coordonnés. Dans cette thèse, les communautés seront identifiées à l'aide de l'algorithme Infomap, qui modélise l'écoulement d'information par marches aléatoires et fournit des partitions précises et efficaces pour de grands graphes de trafic.

Le chapitre présente ensuite de manière systématique différents types de mesures issues de la théorie des réseaux complexes, organisées en trois catégories : mesures au niveau du graphe, au niveau des nœuds et au niveau des communautés. Les mesures globales, telles que la transitivité (ou coefficient de clustering global), la densité, la modularité de Newman et le paramètre de mixing, résument la structure d'ensemble du graphe, notamment le degré de connectivité, la présence de triangles et la force de la structure communautaire. Les mesures de centralité au niveau des nœuds – centralité de degré, d'intermédiarité (betweenness), de proximité (closeness), centralité propre (eigenvector), PageRank – quantifient le rôle ou l'importance de chaque nœud dans la propagation de l'information, la connectivité ou la structuration du réseau. Des mesures hiérarchiques, telles que le k-core et le k-truss, permettent d'identifier des sous-graphes denses et cohésifs correspondant à des "cœurs" structurés du réseau.

Enfin, des mesures au niveau des communautés sont discutées, notamment les centralités mod-

ulaires locales et globales (qui distinguent l'influence intra-communautaire et inter-communautaire), Comm centrality (qui favorise les nœuds à la fois hubs locaux et ponts entre communautés) et la Modularity Vitality (qui estime la contribution d'un nœud à la qualité de la partition communautaire). Ces mesures enrichissent fortement l'analyse et permettent d'identifier des nœuds critiques selon différentes perspectives (hubs internes, passerelles, nœuds qui structurent fortement leurs communautés, etc.). Cette panoplie de métriques constitue le socle des approches de la thèse, où des centralités seront injectées soit comme caractéristiques supplémentaires dans des modèles profonds (chapitres GDLC et Fed-GDLC), soit comme indicateurs guidant le design et la configuration de GNN.

La dernière partie du chapitre discute les limitations récurrentes de l'état de l'art et les formalisent sous forme de trois grandes lacunes qui motivent les contributions de la thèse.

- **(L1) Absence de contexte de graphe.** La majorité des travaux historiques considère encore les flux comme des enregistrements IID, ignorants des relations "qui communique avec qui" et de la fréquence des interactions. Dans ces conditions, les comportements coordonnés de type portscan, botnet ou attaques DDoS distribuées restent difficilement détectables : chaque flux pris isolément peut sembler bénin, alors que la structure de graphe (étoiles, plusieurs-vers-un, motifs de beaconing, pics de degré ou de PageRank) rend l'attaque évidente dès que les flux sont agrégés dans un réseau. Le chapitre illustre ces patrons topologiques typiques, qui ne deviennent visibles qu'après construction d'un graphe et calcul de mesures de centralité ou de motifs.

- **(L2) Jeux de données obsolètes ou biaisés.** Une part importante de la littérature s'appuie encore sur des jeux de données anciens ou limités, présentant des déséquilibres extrêmes entre classes, des fuites temporelles ou des chevauchements d'hôtes entre ensembles d'entraînement et de test. Ces biais conduisent à des performances artificiellement élevées mais peu transférables à des environnements réels. Le chapitre insiste sur la nécessité d'utiliser des corpus plus récents, plus variés et mieux contrôlés, ainsi que des protocoles d'évaluation qui respectent la temporalité et la séparation des rôles d'hôtes.

- **(L3) Lacunes de modélisation et de déploiement.** Les modèles ML classiques peinent à suivre la croissance des espaces de caractéristiques et les dérives de distribution, tandis que les modèles profonds restent coûteux, parfois opaques et difficiles à calibrer. En apprentissage fédéré, la plupart des travaux se concentrent sur des scénarios simplifiés (un seul jeu de données partitionné

en clients) et n'étudient pas suffisamment la généralisation globale, ni l'impact de la structure de graphe sur la convergence et la robustesse. Enfin, bien que les GNN aient montré un potentiel réel, la littérature intègre encore peu les mesures de réseaux complexes dans la boucle d'apprentissage et dans les choix architecturaux.

En conclusion, ce chapitre établit que, malgré des avancées importantes des NIDS basés sur l'IA, plusieurs verrous demeurent : manque d'exploitation systématique de l'information topologique, dépendance à des jeux de données limités, difficultés de déploiement dans des cadres distribués et hétérogènes, et sous-utilisation des mesures de réseaux complexes pour guider la modélisation. Ces constats motivent les contributions de la thèse, qui visent (i) à injecter des mesures de centralité dans des modèles profonds centralisés (GDLC), (ii) à étudier leur impact dans des scénarios d'apprentissage fédéré sous données non-IID, et (iii) à concevoir et adapter des GNN pour des graphes de flux en exploitant explicitement la structure de réseau et les propriétés issues de la théorie des réseaux complexes. Les chapitres suivants détaillent ces pipelines, les jeux de données utilisés et les protocoles d'évaluation mis en place pour répondre à ces limitations.

Résumé du Chapitre II, GDLC : un cadre de Deep Learning sur graphes basé sur les mesures de centralité

1. Introduction

Les systèmes de détection d'intrusion réseau (NIDS) ont pour objectif de distinguer un trafic bénin d'un comportement malveillant à partir de traces de communication. Avec la généralisation de l'Internet des Objets (IoT) et des systèmes cyber-physiques, la surface d'attaque s'élargit considérablement : de nouvelles formes d'attaques, souvent distribuées et furtives, ne correspondent plus aux signatures statiques sur lesquelles reposent les systèmes de détection traditionnels. Les approches à base de règles offrent de faibles taux de faux positifs mais échouent face aux attaques de type *zero-day* ou polymorphes, tandis que les méthodes par anomalies, plus flexibles, souffrent en général de taux d'alerte élevés.

L'essor des techniques de machine learning (ML) et de deep learning (DL) a permis d'améliorer significativement les performances des NIDS sur de nombreux jeux de données de référence. Des modèles tels que les arbres de décision, les forêts aléatoires, les SVM, ainsi que des architectures profondes

de type CNN ou RNN, apprennent des représentations discriminantes à partir de caractéristiques extraites des paquets ou des flux. Toutefois, la plupart de ces modèles restent centrés sur des caractéristiques locales par flux (durée, nombre d'octets, indicateurs de protocole, etc.) et ignorent la structure globale du réseau et les relations entre dispositifs. Dans des scénarios IoT où des botnets, des attaques par déni de service ou des campagnes de scanning s'organisent autour de nœuds centraux, cette vision purement locale devient insuffisante.

Pour dépasser cette limite, ce chapitre s'appuie sur les outils des réseaux complexes et, en particulier, sur les mesures de centralité. Celles-ci quantifient l'importance structurelle d'un nœud dans un graphe, en fonction de son degré, des chemins qui le traversent, de son appartenance à un noyau densément connecté ou de son rôle dans la structure de communautés. L'idée directrice est de projeter le trafic en un graphe de communication (nœuds = adresses IP, arêtes = flux) et d'extraire des descripteurs topologiques qui capturent des motifs d'attaque invisibles dans les seules caractéristiques de flux.

Le chapitre commence par introduire la construction de graphes de communication et les principaux types de mesures de centralité dans le contexte de la détection d'intrusion. Une première étude, basée sur l'ajout fixe de deux centralités (Degree, Betweenness) dans des modèles de ML classiques, montre que même des descripteurs structurels simples améliorent l'Accuracy et le F1-score sur les jeux de données IoT-NI et Edge-IIoTset. À partir de ces résultats encourageants, le chapitre propose ensuite un cadre plus général, le *Graph Deep Learning framework based on Centrality measures* (GDLC), qui sélectionne dynamiquement un ensemble de mesures de réseaux complexes adaptées à la topologie observée et les injecte dans des modèles de DL modernes (CNN, LSTM, GRU).

L'objectif global de ce chapitre est donc double : démontrer l'apport des mesures de centralité en complément des caractéristiques de flux et introduire un cadre générique, GDLC, capable d'adapter la sélection de ces mesures à des graphes IoT de taille et de structure variées, tout en améliorant les performances de détection et en réduisant les faux positifs.

2. Problématique

Les approches classiques de détection d'intrusion reposent principalement sur des caractéristiques agrégées par flux et sur des modèles qui considèrent chaque flux indépendamment des autres. Cette hypothèse est mal adaptée aux environnements IoT, où de nombreuses attaques se manifestent par des

motifs de communication collectifs : attaques par déni de service émanant d'un ensemble de dispositifs compromis vers un petit nombre de cibles, campagnes de scanning menées par des nœuds fortement connectés, ou encore attaques de type Man-In-The-Middle qui exploitent des positions centrales sur des chemins critiques. Or, ces motifs reposent sur la structure globale du graphe de communication, absente de l'espace de caractéristiques traditionnel.

Une première réponse consiste à enrichir les flux par des mesures de centralité calculées sur un graphe construit à partir du trafic. Toutefois, les premiers travaux se limitent à un petit sous-ensemble de mesures, comme Degree et Betweenness, choisi de manière fixe pour tous les scénarios. Cette stratégie présente deux limites majeures. D'une part, elle ne bénéficie pas de la richesse du catalogue des centralités classiques, hiérarchiques et modulaires disponible dans le champ des réseaux complexes. D'autre part, elle ne tient pas compte de l'hétérogénéité structurelle des jeux de données : densité, transivité, force de la structure de communautés et distribution des degrés peuvent varier fortement d'un environnement IoT à l'autre.

En parallèle, calculer l'ensemble des mesures possibles pour chaque nouveau graphe n'est pas une option réaliste. Certaines centralités sont coûteuses à évaluer (jusqu'à $O(VE)$), et beaucoup sont fortement corrélées. La redondance qui en résulte augmente inutilement la dimension de l'espace de caractéristiques, complexifie l'apprentissage et alourdit la charge de calcul, sans bénéfice clair en termes de performance. Dans un contexte IoT à grande échelle, où des millions de flux doivent être traités, cette explosion combinatoire pose un problème de passage à l'échelle.

Enfin, du point de vue opérationnel, les NIDS basés sur ML/DL restent confrontés à la question des faux positifs et des faux négatifs. Les faux positifs surchargent les équipes de sécurité et nuisent à la confiance dans le système, tandis que les faux négatifs laissent passer des attaques potentiellement graves. Les modèles profonds sur caractéristiques de flux obtiennent déjà des performances élevées, mais conservent des marges d'amélioration, en particulier pour les attaques rares ou structurellement spécifiques.

La problématique abordée dans ce chapitre peut ainsi se formuler comme suit : comment exploiter de manière systématique et scalable les mesures de réseaux complexes pour enrichir les NIDS, en sélectionnant dynamiquement un sous-ensemble de centralités adaptées à la topologie du graphe et en les intégrant dans des modèles de DL, tout en améliorant l'Accuracy et en réduisant significativement les faux positifs et faux négatifs sur des jeux de données IoT variés ?

3. Méthodologie

3.1 Construction de graphes à partir des flux

La première étape consiste à transformer le trafic en graphes de communication. Les enregistrements de flux agrègent des paquets partageant les mêmes adresses IP source et destination, ports, protocole et certaines caractéristiques temporelles. À partir d'un ensemble de flux $\mathcal{F}$, on construit un graphe orienté $G = (V, E)$ où chaque adresse IP distincte fournit un nœud et chaque flux (s, d) donne lieu à une arête orientée $e = (s, d)$. Plusieurs variantes sont possibles : représentation à arête unique (un seul lien par paire (s, d)), multigraphe (une arête par flux), ou graphe pondéré (poids proportionnel au nombre de flux). Le choix entre ces variantes induit un compromis entre compacité, niveau de détail et coût de calcul.

La dimension temporelle du trafic est prise en compte par la construction de graphes sur des fenêtres, par exemple un nombre fixé de flux ou un intervalle de temps donné. Cette approche par fenêtres successives fournit une série de graphes instantanés qui capturent l'évolution de l'état du réseau et facilitent la détection d'anomalies transitoires. D'autres schémas, comme la construction par sessions, regroupent les flux appartenant à une même conversation logique, ce qui est particulièrement pertinent dans des environnements IoT caractérisés par des communications périodiques.

Dans tous les cas, la construction des graphes s'accompagne d'un prétraitement des données : suppression des doublons, gestion des valeurs manquantes, traitement des valeurs extrêmes susceptibles de déformer la topologie, et prise en compte des limitations liées à la disponibilité des adresses IP dans certains jeux de données.

3.2 Intégration de centralités dans des modèles de ML

La première série d'expériences revisite le travail de Termos *et al.*, dans lequel des mesures de Degree et Betweenness sont calculées pour chaque nœud du graphe et réinjectées au niveau des flux. Concrètement, on associe à chaque flux les valeurs de centralité de ses extrémités, qui sont ensuite concaténées aux caractéristiques de trafic classiques. Cette représentation hybride est utilisée pour entraîner plusieurs classifieurs : arbre de décision, forêt aléatoire, réseau de neurones, AdaBoost et XGBoost.

Les expériences sont menées sur deux jeux de données IoT représentatifs : IoT-NI et Edge-IIoTset.

Ces graphes diffèrent par leur taille, leur densité, leur coefficient de clustering et leur degré maximal, ce qui permet de tester la robustesse de l'approche sur des topologies contrastées. Les résultats, synthétisés notamment dans la Figure 2.2, montrent que l'ajout de Degree et Betweenness améliore systématiquement l'Accuracy et le F1-score, en particulier pour les attaques peu fréquentes.

3.3 Cadre GDLC et sélection dynamique des mesures

Pour dépasser les limites d'un choix fixe de deux centralités, le cadre GDLC généralise l'approche en trois étapes. Tout d'abord, un graphe orienté de communication est construit comme précédemment. Ensuite, des mesures globales décrivant la structure du graphe sont calculées : densité, transivité et *mixing parameter*, qui reflète la force de la structure de communautés. Ces indicateurs guident la sélection d'un sous-ensemble de mesures de réseaux complexes jugé informatif et peu redondant.

GDLC considère plusieurs familles de mesures :

- des centralités classiques (Degree, Betweenness, Eigenvector, Closeness, PageRank) qui capturent différents aspects de l'influence et de la connectivité
- des mesures hiérarchiques (K-core, K-truss) qui identifient des sous-structures denses
- des centralités modulaires, locales ou globales, associées notamment aux notions de Comm Centrality et de Modularity Vitality.

La stratégie de sélection, formalisée dans l'algorithme présenté à la Figure 2.4 et dans l'Algorithme 2, repose sur les propriétés du graphe : lorsque densité et transivité sont élevées, seules quelques mesures orthogonales sont nécessaires, tandis que dans des graphes plus épars, un ensemble plus large est retenu. De même, un mixing parameter élevé favorise les centralités modulaires locales et Comm Centrality, alors qu'un mixing parameter faible conduit à privilégier des centralités modulaires globales et Modularity Vitality. Le résultat de cette sélection par jeu de données est résumé dans le Tableau 2.2.

Les mesures ainsi choisies sont ensuite concaténées aux caractéristiques de trafic pour former un vecteur de caractéristiques hybride, qui sert d'entrée à trois familles de modèles profonds : CNN unidimensionnels, LSTM et GRU. Les CNN traitent les flux individuellement, tandis que les LSTM et GRU exploitent la dimension séquentielle des flux pour modéliser les dépendances temporelles. Les architectures sont illustrées dans la Figure 2.6, avec des fonctions de coût adaptées aux tâches binaires

et multi-classes.

3.4 Jeux de données, validation SIR et cadre expérimental

GDLC est évalué sur deux jeux de données d'intrusion largement utilisés et structurés en format CICFlowMeter : CIC-ToN-IoT et CIC-IDS2017. Tous deux sont convertis en graphes de communication dirigés, puis normalisés après suppression des colonnes sujettes à fuite d'information (IP, timestamp). Le Tableau 2.1 récapitule les propriétés structurelles des graphes obtenus : CIC-ToN-IoT est un graphe sparse de grande taille avec un mixing parameter élevé, tandis que CIC-IDS2017 est plus petit, plus dense et doté d'une structure de communautés plus marquée.

Pour valider la pertinence des mesures de réseaux complexes, le modèle épidémique SIR est appliqué sur CIC-ToN-IoT. Dans ces simulations (Figure 2.7), une fraction de nœuds est initialement infectée, puis le processus de propagation et de guérison est simulé en comparant différentes stratégies d'immunisation : aléatoire ou guidée par des classements issus des centralités (Degree, Betweenness, PageRank, K-core).

Enfin, l'ensemble des expériences de classification est exécuté sur une station de travail commune (Intel® Xeon® Gold 6330, 65 Go de RAM, Windows Server 2022), afin de garantir des conditions de comparaison homogènes. Les modèles sont évalués en classification binaire et multi-classe à l'aide de métriques standard (Accuracy, Recall, Precision, F1-score, FPR, FNR), en distinguant les versions *baseline* (trafic seul) et GDLC (trafic + mesures de réseaux complexes), comme détaillé dans les Figures 2.8, 2.9, 2.10, 2.11 et les Tables 2.3, 2.4.

4. Résultats

Les premières expériences menées sur IoT-NI et Edge-IIoTset montrent que l'ajout de seulement deux mesures de centralité (Degree, Betweenness) aux caractéristiques de trafic améliore l'Accuracy et le F1-score pour l'ensemble des modèles de ML testés. Comme illustré dans la Figure 2.2, tous les classifieurs bénéficient de ce complément structurel, avec des gains particulièrement marqués pour les types d'attaques rares. Cette observation confirme que des informations structurelles simples, absentes des caractéristiques de flux classiques, apportent une valeur tangible pour la détection d'intrusion.

Dans le cadre de GDLC, la première série de résultats concerne la validation des mesures de réseaux complexes par le modèle SIR. Sur CIC-ToN-IoT (Figure 2.7), immuniser les nœuds les plus centraux,

selon Degree, Betweenness, PageRank ou K-core, réduit systématiquement la fraction de nœuds finalement atteints par l'épidémie par rapport à une immunisation aléatoire. Les mesures de Degree et de PageRank se distinguent par un effet légèrement plus prononcé, en cohérence avec la structure sparse et dominée par des hubs du graphe considéré. Ces résultats valident l'idée que les mesures retenues par GDLC identifient bien des nœuds structurellement critiques, dont la compromission ou la protection a un impact important sur la propagation d'attaques.

La sélection dynamique de mesures, telle que résumée dans le Tableau 2.2, conduit à des combinaisons différentes pour CIC-ToN-IoT et CIC-IDS2017. Dans CIC-ToN-IoT, caractérisé par une forte hétérogénéité et une structure de communautés relativement faible (mixing parameter élevé), GDLC retient des centralités classiques couplées à des centralités modulaires locales et à Comm Centrality, en plus des mesures hiérarchiques K-core et K-truss. À l'inverse, pour CIC-IDS2017, où la structure de communautés est plus marquée (mixing parameter faible), le cadre sélectionne des centralités modulaires globales et Modularity Vitality, toujours en combinaison avec des centralités classiques et des mesures hiérarchiques. Cette adaptation des caractéristiques à la topologie est un élément clé de la robustesse observée.

Sur CIC-ToN-IoT, les expériences de classification montrent un gain net de performance lorsque les modèles profonds sont enrichis par les mesures de réseaux complexes. La Figure 2.8 et le Tableau 2.3 indiquent des hausses d'Accuracy en classification binaire de l'ordre de plusieurs points pour CNN, LSTM et GRU, ainsi que des améliorations comparables en multi-classe. Par exemple, l'Accuracy atteint 96,67 % pour CNN, 98,90 % pour LSTM et 98,68 % pour GRU après intégration des caractéristiques CN. Ces gains s'accompagnent d'une baisse substantielle du FPR et du FNR (Figure 2.9), avec des réductions relatives de FPR pouvant dépasser 70 % par rapport aux baselines. Ces résultats suggèrent que les descripteurs structurels jouent un rôle important pour séparer les comportements bénins et malveillants, en particulier lorsque les modèles récurrents exploitent simultanément la dynamique temporelle des flux.

Des tendances similaires sont observées sur CIC-IDS2017. Comme illustré dans la Figure 2.10 et le Tableau 2.4, l'Accuracy en binaire grimpe jusqu'à environ 98,5–99 % selon l'architecture, avec des augmentations significatives de F1-score et une diminution conjointe des FPR/FNR (Figure 2.11). En multi-classe, bien que les gains en Accuracy soient plus modestes, les modèles GDLC conservent un avantage systématique sur leurs équivalents basés uniquement sur les caractéristiques de trafic. Le

bénéfice est particulièrement visible pour les modèles récurrents (LSTM, GRU), ce qui met en évidence la synergie entre informations temporelles et structurelles.

Pour mieux comprendre la manière dont les mesures de réseaux complexes se répartissent entre trafic bénin et attaques, le chapitre propose enfin une analyse basée sur la fonction de répartition empirique (ECDF) appliquée à CIC-ToN-IoT (Figure 2.12). Les courbes ECDF de Betweenness, Eigenvector et K-core montrent des différences de distribution entre les deux classes : les enregistrements d'attaque se concentrent à de faibles valeurs de Betweenness, présentent des valeurs d'Eigenvector centrality plus élevées et tendent à occuper des niveaux de K-core plus importants. Cette observation suggère que les nœuds impliqués dans les attaques ne sont pas nécessairement des ponts sur les plus courts chemins, mais qu'ils sont souvent insérés dans des régions denses et connectées à des voisins influents. De telles signatures structurelles complètent utilement les caractéristiques de trafic pour permettre aux modèles profonds de mieux discriminer les comportements.

En synthèse, les résultats de ce chapitre montrent que l'intégration de mesures de réseaux complexes sélectionnées dynamiquement au sein de modèles de DL permet d'augmenter l'Accuracy jusqu'à environ 7–8 points, tout en réduisant significativement les taux de faux positifs et de faux négatifs sur deux jeux de données d'intrusion de référence. GDLC fournit ainsi un NIDS *structure-aware* qui exploite à la fois la dynamique temporelle des flux et les rôles topologiques des dispositifs dans le graphe de communication. La principale limite identifiée réside dans l'hypothèse d'un cadre centralisé et d'un espace de caractéristiques homogène, peu réaliste dans des déploiements multi-organisations soumis à des contraintes de confidentialité. Les chapitres suivants, consacrés à Fed-GDLC et ADAP-GNN, étendent cette approche aux environnements fédérés et aux modèles à base de GNN, tout en conservant le principe central de guidage par les mesures de réseaux complexes.

Résumé du Chapitre III, Fed-GDLC: un cadre fédéré de deep learning sur graphes avec des caractéristiques basées sur les centralités pour la détection d'intrusion

1. Introduction

Ce chapitre s'intéresse à la détection d'intrusion dans des environnements modernes distribués, tels que les déploiements IoT et les villes intelligentes, où le trafic réseau est volumineux, sensible et

fortement hétérogène. Dans ces contextes, les flux sont naturellement répartis entre plusieurs entités (entreprises, hôpitaux, FAI, sites industriels, routeurs de bordure, etc.), ce qui rend la centralisation des données à des fins de détection d'intrusion à la fois coûteuse (bande passante, stockage), risquée (confidentialité, conformité réglementaire) et parfois juridiquement impossible. Les systèmes de détection d'intrusion réseau (NIDS) doivent donc être repensés pour fonctionner dans des environnements distribués où les données ne quittent pas leur domaine d'origine.

L'Apprentissage Fédéré (Federated Learning, FL) apparaît comme un paradigme adapté à ce défi. Il permet à plusieurs clients de collaborer pour entraîner un modèle global sans partager leurs données brutes : chaque client apprend localement sur ses propres flux, envoie des mises à jour de modèle à un serveur central, qui agrège ces mises à jour pour obtenir un modèle global, puis redistribue ce modèle aux clients. Cette boucle de communication se répète jusqu'à convergence. Dans le contexte des NIDS, ce schéma permet à des organisations distinctes de mutualiser leurs expériences d'attaque tout en préservant la localité et la confidentialité des traces réseau.

Ce chapitre prolonge les travaux précédents basés sur les réseaux complexes et les mesures de centralité. Dans une première étape, la méthodologie de centralités introduite dans le cadre GDLC est transposée telle quelle dans un environnement fédéré : les centralités sont calculées localement sur le graphe de communication de chaque client et intégrées comme caractéristiques supplémentaires dans un modèle de deep learning entraîné par FL. Cela permet de vérifier si les gains observés en contexte centralisé se maintiennent lorsque les données sont non-IID et réparties sur plusieurs sites.

Dans un second temps, le chapitre introduit Fed-GDLC, une extension fédérée du framework GDLC. L'idée centrale est de combiner (i) une sélection dynamique de mesures de centralité guidée par des métriques de graphe (densité, paramètre de mélange, structure de communautés) et (ii) un mécanisme de mise en commun des espaces de caractéristiques par une version fédérée de l'Analyse en Composantes Principales (FedPCA). Les clients construisent leurs graphes, calculent et sélectionnent localement des centralités non redondantes, enrichissent leurs flux avec ces mesures, puis envoient uniquement des statistiques agrégées au serveur pour aligner leurs espaces de caractéristiques dans un sous-espace commun. Sur ce sous-espace, des modèles profonds (CNN et CNN-LSTM) sont entraînés en mode fédéré.

L'objectif global de ce chapitre est donc double : démontrer que les centralités restent discriminantes dans un cadre fédéré de NIDS, et montrer que Fed-GDLC permet de gérer la forte hétérogénéité

des données et des graphes tout en respectant les contraintes de confidentialité et de scalabilité propres aux environnements IoT et smart cities.

2. Problématique

La problématique centrale abordée dans ce chapitre est de savoir comment exploiter l'information topologique issue des graphes de communication dans un environnement d'Apprentissage Fédéré pour la détection d'intrusion, tout en respectant les contraintes de confidentialité, d'hétérogénéité et d'évolutivité.

En premier lieu, la simple centralisation des flux réseau pose des problèmes majeurs. Sur le plan pratique, l'agrégation de volumes massifs de données issus de multiples réseaux (IoT, industriels, d'entreprise, etc.) engendre des coûts élevés en bande passante et en stockage. Sur le plan juridique et éthique, le transfert de telles données peut contrevenir à des réglementations sur la protection de la vie privée ou des données sensibles, en particulier dans les environnements médicaux ou critiques. L'Apprentissage Fédéré répond en partie à cette contrainte, mais introduit d'autres défis : les données sont non-IID, les tailles de jeux de données varient fortement d'un client à l'autre, et la qualité des graphes de communication peut différer selon les scénarios.

En second lieu, l'introduction de mesures de centralité dans ce cadre fédéré n'est pas triviale. Dans les travaux centralisés, les centralités sont calculées sur un graphe global, puis intégrées de manière uniforme dans les modèles. En contexte fédéré, chaque client possède son propre graphe, potentiellement très différent des autres en termes de densité, de structure de communautés ou de distribution d'attaques. La question est donc de savoir si des centralités calculées localement sur ces graphes hétérogènes conservent leur pouvoir discriminant lorsqu'elles sont agrégées au niveau global via FL.

En troisième lieu, la diversité des graphes pose un problème spécifique d'alignement des espaces de caractéristiques. Dans Fed-GDLC, chaque client applique le framework GDLC pour sélectionner dynamiquement un sous-ensemble de centralités en fonction de ses propres propriétés de graphe (densité, paramètre de mélange, structure communautaire). Cette sélection locale produit des ensembles de caractéristiques différents d'un client à l'autre, alors que l'Apprentissage Fédéré suppose un modèle global partagé fonctionnant sur un espace de caractéristiques commun. Comment unifier ces espaces sans demander à tous les clients de calculer toutes les centralités possibles, ce qui serait coûteux et

favoriserait le surapprentissage ?

Enfin, un dernier volet de la problématique touche au compromis entre performance, robustesse et coût computationnel. D'une part, il faut déterminer le nombre de composantes principales à retenir dans FedPCA pour capter suffisamment d'information sans surcharger la communication ni le calcul. D'autre part, il s'agit de choisir des architectures de deep learning (MLP, CNN, CNN-LSTM) capables de tirer parti des caractéristiques topologiques alignées tout en restant entraînables dans un cadre fédéré. La comparaison avec des approches plus naïves — Baseline sans graphes, usage de trois centralités fixes, ajout de toutes les centralités — doit permettre de clarifier les gains et les limites de Fed-GDLC.

En résumé, la problématique traitée dans ce chapitre consiste à concevoir, aligner et exploiter des caractéristiques basées sur les centralités dans un environnement d'Apprentissage Fédéré pour NIDS, en répondant aux enjeux de confidentialité, d'hétérogénéité des clients, d'efficacité computationnelle et de généralisation globale.

3. Méthodologie

La méthodologie du chapitre se déploie en deux grands volets : (i) l'intégration directe de centralités dans un scénario FL pour NIDS, puis (ii) le cadre Fed-GDLC, qui généralise cette approche en ajoutant une sélection dynamique et un alignement fédéré par FedPCA.

Dans le premier volet, la démarche consiste à reprendre la méthodologie centralisée de GDLC et à la transposer dans un cadre fédéré. Deux jeux de données de référence pour la détection d'intrusion, CIC-IDS2017 et CIC-ToN-IoT, sont utilisés pour simuler des clients FL. Chaque partition est convertie en graphe de communication à partir des paires IP source–destination, puis enrichie avec une sélection fixe de mesures de centralité. Les clients entraînent localement un modèle hybride CNN–LSTM, adapté à la fois aux motifs locaux (via 1D-CNN) et aux dépendances temporelles (via LSTM). Les mises à jour de modèle sont ensuite agrégées côté serveur, conformément au schéma classique de FL illustré dans Figure 3.1. Les performances sont évaluées principalement via la F1-score au fil des tours de fédération, en comparant un modèle basique (sans centralités) et un modèle enrichi par les centralités.

Le second volet introduit Fed-GDLC, qui structure la méthode en trois phases (Figure 3.4). La première phase applique localement GDLC sur chaque client. À partir des flux réseau, un graphe dirigé

non pondéré est construit, où les sommets représentent les adresses IP et les arêtes l'existence d'au moins un flux entre deux IP. Des mesures de graphe globales, notamment la densité et le paramètre de mélange, sont calculées pour résumer la connectivité et la structure communautaire. Ces métriques guident ensuite une sélection dynamique de mesures de centralité parmi plusieurs familles : centralités locales (degré), globales (intermédiarité, proximité, vecteur propre, PageRank), hiérarchiques (k-core, k-truss) et communautaires (centralités locales/globales de communauté, Comm centrality, modularity vitality), comme synthétisé dans Table 3.1. Le but est d'obtenir un ensemble compact et non redondant adapté à chaque graphe client. Les valeurs de centralité sélectionnées sont enfin injectées dans les enregistrements de flux en tant que caractéristiques supplémentaires pour les nœuds source et destination.

La deuxième phase est consacrée à l'alignement des espaces de caractéristiques via FedPCA. Plutôt que de concaténer toutes les caractéristiques locales ou de centraliser les données brutes, chaque client calcule la matrice de covariance de ses caractéristiques enrichies et n'envoie que ces statistiques (et leur taille d'échantillon) au serveur. Le serveur agrège ces covariances pour construire une covariance globale, effectue une décomposition en valeurs propres, puis sélectionne les k premières composantes définissant une base de projection commune. Ces composantes sont renvoyées à tous les clients, qui projettent alors leurs données locales dans ce sous-espace de dimension réduite, comme décrit formellement dans l'algorithme FedPCA (Algorithm 4). Cette étape permet de disposer d'un espace de caractéristiques aligné entre clients, tout en préservant la localité des données brutes.

La troisième phase met en œuvre l'Apprentissage Fédéré proprement dit sur cet espace commun. Cinq jeux de données standardisés — CIC-Bot-IoT, CIC-ToN-IoT, CIC-IDS2017, CIC-DDoS-2019 et CIC-UNSW-NB15 — sont utilisés pour instancier des clients avec des propriétés de graphe et des déséquilibres de classes variés (Table 3.2 et Table 3.3). Les modèles sont implémentés en PyTorch, la fédération est simulée avec Flower, et les graphes ainsi que les centralités sont construits avec NetworkX. Chaque client entraîne localement son modèle pendant une époque à chaque tour de FL, avec un taux d'apprentissage de 5×10^{-5}. Pour gérer le fort déséquilibre de classes, une Focal Loss est utilisée avec des hyperparamètres $\alpha = 1.0$ et $\gamma = 2.0$, de manière à pénaliser davantage les exemples difficiles. L'agrégation des poids est réalisée via FedProx, ce qui permet de mieux gérer l'hétérogénéité des données clientes.

Plusieurs scénarios sont comparés : un Baseline sans centralités, une variante avec trois centralités

fixes (degré, intermédiarité, PageRank) appliquées uniformément à tous les clients, une variante "All Centralities" où toutes les mesures sont ajoutées sans sélection, et enfin Fed-GDLC, qui combine sélection dynamique locale et alignement FedPCA. Les modèles étudiés sont un MLP, un CNN et un CNN–LSTM, et les évaluations portent sur la F1-score globale sur un jeu de test serveur unifié, la perte moyenne de training par client et le temps d'exécution décomposé entre calcul des caractéristiques et entraînement fédéré.

4. Résultats

Les résultats sont présentés en plusieurs étapes, en commençant par l'impact des centralités dans un scénario FL simple, puis en évaluant en détail Fed-GDLC, son module FedPCA, les modèles de deep learning associés et le coût d'exécution.

Dans le premier ensemble d'expériences, où une sélection fixe de centralités est intégrée dans un cadre FL pour NIDS, les courbes de F1-score au fil des tours de fédération montrent que l'ajout de mesures de centralité améliore systématiquement les performances par rapport au modèle Baseline qui n'utilise que les caractéristiques de flux. Comme illustré par Figure 3.3, le modèle enrichi gagne en F1-score et présente une stabilité accrue dans les derniers tours, ce qui indique une meilleure généralisation à travers des clients aux distributions hétérogènes. Ce résultat confirme que l'information topologique extraite localement des graphes de communication conserve son intérêt dans un contexte fédéré.

Les expériences dédiées à FedPCA s'intéressent d'abord à la qualité de reconstruction des données projetées. En comparant la RMSE de reconstruction entre une PCA locale (par client) et la FedPCA partagée, Figure 3.8 montre que FedPCA induit une légère augmentation d'erreur, due au fait qu'elle doit représenter un sous-espace commun à des clients hétérogènes. Néanmoins, cette erreur diminue lorsque le nombre de composantes augmente, traduisant une meilleure capture de la variance globale. Sur le plan des performances de détection, l'étude de la F1-score en fonction du nombre de composantes FedPCA (Figure 3.9) révèle qu'une à deux composantes ne suffisent pas, et qu'il n'existe pas de "coude" marqué au-delà de trois composantes. La F1-score globale atteint un maximum pour six composantes, valeur adoptée dans la suite, tandis que des dimensions plus élevées entraînent des gains marginaux voire un surapprentissage, notamment à dix composantes.

La comparaison entre les différentes stratégies fédérées — Baseline, trois centralités fixes, All Centralities et Fed-GDLC — fait apparaître Fed-GDLC comme la meilleure option. Avec six com-

posantes FedPCA, le CNN–LSTM sous Fed-GDLC atteint une F1-score d'environ 75,1% sur le jeu de test unifié, surpassant la variante à trois centralités (71,3%) et le Baseline (68,2%). De manière surprenante, la variante All Centralities n'obtient que 61,4%, malgré une perte d'entraînement plus faible. Les courbes de F1-score et de perte au fil des tours (Figure 3.10 et Figure 3.11) montrent que l'ajout de toutes les centralités entraîne un surapprentissage aux distributions locales des clients : la perte de training diminue fortement, mais la F1-score de test est dégradée. À l'inverse, Fed-GDLC parvient à équilibrer la richesse de représentation et la capacité de généralisation, en s'appuyant sur une sélection dynamique de caractéristiques pertinentes et sur un alignement dans un sous-espace partagé de dimension modérée.

L'étude comparative des architectures de deep learning sous Fed-GDLC (Figure 3.12) confirme la supériorité du modèle CNN–LSTM dans ce cadre. Après 20 tours de FL avec six composantes FedPCA, le CNN–LSTM atteint environ 75,10% de F1-score, le CNN 70,16% et le MLP 63,36%. Pour situer ces résultats, une expérience centralisée est réalisée en fusionnant l'ensemble des jeux de données et en entraînant un CNN–LSTM sur ce corpus complet. Dans ce cas, la F1-score atteint 78,27%. Aucun modèle fédéré ne dépasse ce plafond, ce qui est attendu compte tenu des contraintes de confidentialité et de non-centralisation. Néanmoins, l'écart d'environ 3% entre le CNN–LSTM Fed-GDLC et son équivalent centralisé montre que la perte de performance due aux contraintes de FL reste modérée, tandis que les bénéfices en matière de confidentialité et de partage de connaissances entre organisations sont substantiels.

Enfin, l'analyse du temps d'exécution (Figure 3.13) décompose le coût total en deux composantes : le calcul des caractéristiques (centralités, PCA) et l'entraînement fédéré sur 20 tours. Les résultats montrent que, pour toutes les approches, le coût dominant provient clairement des tours de FL. L'ajout de centralités, en particulier sous Fed-GDLC, introduit un surcoût mesuré mais acceptable, largement compensé par les gains en performance. La variante All Centralities affiche le plus grand temps d'entraînement, confirmant que multiplier sans sélection les caractéristiques topologiques n'est ni efficient ni forcément bénéfique pour la généralisation.

La conclusion du chapitre souligne que les vecteurs de centralité sont (i) utiles dans des environnements décentralisés et (ii) alignables sans partage de trafic brut. L'approche Fed-GDLC montre qu'une sélection dynamique guidée par des mesures de graphe, combinée à un alignement FedPCA, permet de construire des NIDS fédérés robustes, exploitant efficacement la structure des graphes sous-

jacents. Les limites identifiées concernent notamment la robustesse face à des mises à jour malveillantes, l'efficacité communicationnelle et l'intégration de mécanismes d'agrégation sécurisée et de confidentialité différentielle. Ces résultats préparent la transition vers le chapitre suivant, qui abandonne l'ingénierie manuelle de caractéristiques au profit de l'apprentissage direct sur graphes via des réseaux de neurones de graphes. Ce passage s'effectue en initialisant les états de nœuds avec des centralités, en concevant des agrégateurs sensibles aux arêtes et en proposant ADAP-GNN, un cadre pour sélectionner dynamiquement l'architecture de GNN en fonction des propriétés des graphes, ouvrant la voie à des NIDS fédérés basés sur des GNN.

Résumé du Chapitre IV, ADAP-GNN : Un cadre adaptatif pour les GNN centrés sur les flux dans les systèmes de détection d'intrusion

1. Introduction

Ce chapitre s'inscrit dans la continuité des contributions précédentes en proposant un cadre unifié pour l'utilisation de réseaux de neurones de graphes (Graph Neural Networks, GNN) dans les systèmes de détection d'intrusion réseau (NIDS) basés sur les flux. Les NIDS modernes, en particulier dans les environnements IoT et IIoT, doivent traiter des volumes massifs de flux et détecter des attaques variées (DoS/DDoS, malwares, scans, etc.). Les approches d'apprentissage profond classiques (MLP, CNN, RNN) considèrent chaque flux indépendamment et ignorent la structure relationnelle entre dispositifs. Or, les graphiques offrent une représentation naturelle : les adresses IP sont modélisées comme des nœuds et les flux comme des arêtes dirigées portant les attributs du trafic.

Les GNN étendent les réseaux neuronaux aux données non euclidiennes en exploitant la connectivité du graphe. À travers un schéma de *message passing*, chaque nœud agrège l'information de son voisinage et apprend une représentation qui combine ses propres caractéristiques et celles de ses voisins. Des architectures majeures sont rappelées dans ce chapitre, notamment les GCN, GraphSAGE et GAT, qui diffèrent par la manière d'agréger et de pondérer les voisins.

Dans le contexte des NIDS, la particularité est que l'information pertinente se trouve principalement au niveau des arêtes (flux) plutôt qu'au niveau des nœuds (adresses IP). La plupart des jeux de données de flux ne fournissent pas de caractéristiques explicites pour les nœuds, ce qui crée un décalage entre la nature des données (centrée sur les arêtes) et la conception classique des GNN (centrée sur les

nœuds). Le chapitre commence donc par rappeler ce problème structurel, puis introduit des variantes *edge-centric* adaptées aux NIDS : E-GraphSAGE et E-ResGraphSAGE.

Une première amélioration consiste à exploiter les mesures de centralité issues de la théorie des réseaux complexes pour initialiser les nœuds avec des caractéristiques topologiques informatives (Figure 4.1, Table 4.1, Table 4.2). Cette intégration de connaissances structurelles améliore déjà les performances des modèles E-GNN pour la classification des flux.

Enfin, le cœur du chapitre est la proposition d'ADAP-GNN, un cadre adaptatif *property-aware* qui ne cherche plus un unique "meilleur" modèle, mais sélectionne automatiquement la configuration GNN la plus appropriée pour un jeu de données donné. ADAP-GNN s'appuie sur deux métriques simples mais interprétables calculées sur le graphe de flux : la proportion de nœuds attaquants et l'entropie de la centralité de degré. Ces propriétés guident deux décisions binaires : (i) utiliser ou non un échantillonnage de voisins, et (ii) utiliser ou non un mécanisme d'attention. L'ensemble des modèles et décisions est évalué sur neuf jeux de données NIDS hétérogènes, couvrant des réseaux IoT et IIoT de grande échelle (Table 4.3).

2. Problématique

L'adaptation des GNN aux NIDS se heurte à deux difficultés majeures. D'abord, la nature *edge-centric* du problème : chaque enregistrement de flux correspond à une arête dirigée entre deux adresses IP et porte l'essentiel de l'information (compteurs de paquets, volumes de bytes, durée, drapeaux de protocole). Les nœuds, eux, ne disposent pas de caractéristiques intrinsèques. Si l'on initialise les nœuds avec des vecteurs triviaux (par exemple des vecteurs de 1), le modèle ne dispose d'aucune information sur le rôle fonctionnel des dispositifs dans le réseau. La phase de message passing se trouve alors limitée par cette absence de signal initial.

Ensuite, les graphes de flux présentent souvent des arêtes parallèles entre une même paire de nœuds, correspondant à plusieurs flux successifs ou simultanés. Ces flux parallèles peuvent appartenir à des classes différentes (benin, scan, malware, etc.). Les architectures de GNN classiques, lorsqu'elles construisent les représentations d'arêtes uniquement à partir des embeddings des nœuds, ont tendance à "écraser" ces flux multiples en une représentation unique, ce qui entraîne une perte d'information et potentiellement des erreurs de classification pour les scénarios où des flux bénins et malveillants coexistent entre les mêmes nœuds.

Au-delà de ce décalage structurel, d'autres questions pratiques se posent. Premièrement, la **scalabilité** : sur des graphes de plusieurs millions de flux, agréger l'ensemble des voisins de chaque nœud est coûteux et sujet au bruit. L'échantillonnage de voisins réduit la complexité et joue un rôle de régularisation, mais il introduit un risque : sur des graphes où les attaquants sont rares, un échantillon aléatoire peut manquer les nœuds malveillants et dégrader les performances. Deuxièmement, le **choix de l'agrégation** : les mécanismes d'attention comme dans GAT permettent de pondérer différemment les voisins en fonction de leur importance, mais augmentent la complexité et peuvent sur-apprendre lorsque la structure du graphe est trop homogène.

Les expériences menées montrent qu'il n'existe pas de configuration GNN universellement meilleure. Sur certains jeux de données, E-GraphSAGE avec agrégation uniforme domine, tandis que sur d'autres, E-GAT avec attention fournit les meilleurs F1-scores (Table 4.4). De même, l'échantillonnage de voisins peut être bénéfique lorsque les attaquants sont nombreux, mais catastrophique lorsqu'ils sont rares (Table 4.5, Figure 4.9). Cela motive une **séléction adaptative** basée sur les propriétés du graphe plutôt qu'un réglage fixe de l'architecture.

Enfin, une autre question centrale est l'exploitation de l'information structurelle du graphe dès l'initialisation des nœuds. Les travaux antérieurs du manuscrit ont montré que les mesures de centralité améliorent les performances des modèles de NIDS basés sur l'apprentissage profond. Le chapitre IV cherche donc à combiner ces deux idées : (i) injecter les centralités dans l'initialisation des GNN *edge-centric*, et (ii) concevoir un mécanisme de sélection de modèle qui tienne compte d'indicateurs topologiques simples (proportion d'attaquants, entropie du degré) pour choisir entre agrégation complète ou échantillonnée, et entre poids uniformes ou attention.

3. Méthodologie

La méthodologie se déploie en trois blocs : les variantes edge-centric des GNN, l'initialisation guidée par les centralités, puis le cadre adaptatif ADAP-GNN accompagné de son protocole expérimental.

Modèles E-GraphSAGE et E-ResGraphSAGE.

E-GraphSAGE adapte GraphSAGE au contexte des flux en apprenant des représentations de nœuds à partir des arêtes incidentes. Les nœuds sont initialisés par un vecteur constant (de 1), puis, à chaque couche, un nœud v agrège les caractéristiques des arêtes $\mathbf{e}_{uv}$ issues de ses voisins $u \in \mathcal{N}(v)$ via

une fonction différentiable $\mathrm{AGG}^{(k)}$. L'agrégat est concaténé avec la représentation précédente du nœud et transformé par une couche linéaire suivie d'une non-linéarité. Après K couches, les embeddings de nœuds $\mathbf{z}_v$ sont combinés pour former les embeddings d'arêtes.

E-ResGraphSAGE étend ce schéma en introduisant des connexions résiduelles au niveau des arêtes. Plutôt que de construire l'embedding d'un flux uv uniquement à partir des embeddings des nœuds source et destination $\mathbf{z}_u, \mathbf{z}_v$, E-ResGraphSAGE réinjecte explicitement le vecteur de caractéristiques d'origine $\mathbf{e}_{uv}$ dans l'agrégation finale :

$$\mathbf{z}_{uv} = \mathrm{AGG}_{edge}(\mathbf{z}_u, \mathbf{z}_v, \mathbf{e}_{uv}).$$

Ainsi, chaque flux garde son individualité, même en présence d'arêtes parallèles entre les mêmes nœuds, ce qui réduit les erreurs de classification dans les scénarios où plusieurs types de trafic coexistent.

Initialisation informée par les centralités.

Pour dépasser l'initialisation uniforme par des vecteurs de 1, le chapitre propose une initialisation *centrality-informed* (Figure 4.1). Cinq mesures sont calculées pour chaque nœud : centralité de degré, d'intermédiarité (betweenness), de proximité (closeness), PageRank et K-truss. Les valeurs sont normalisées et concaténées pour former le vecteur de caractéristiques initial de chaque nœud. Cette stratégie est évaluée sur quatre jeux de données NIDS (NF-UQ-NIDS, NF-CSE-CIC-IDS2018, CCD-INID, X-IIoTID) présentant des densités et degrés moyens très contrastés (Table 4.1). Les résultats comparatifs sont rapportés dans Table 4.2 pour E-GraphSAGE et E-ResGraphSAGE.

Cadre adaptatif ADAP-GNN.

La contribution principale du chapitre est ADAP-GNN, qui sélectionne automatiquement la configuration GNN en fonction de deux métriques de graphe (Figure 4.4, Figure 4.5) :

- **Proportion d'attaquants p.** On compte le nombre de nœuds ayant initié au moins un flux malveillant $N_{\text{attackers}}$ et le nombre total de nœuds N_{total}, puis on définit

  $$p = \frac{N_{\text{attackers}}}{N_{\text{total}}}.$$

  Lorsque p est faible, les attaquants sont rares et l'échantillonnage de voisins risque de les omettre l'agrégation complète est alors privilégiée.

- **Entropie de la centralité de degré E_D.** On calcule la centralité de degré normalisée pour chaque nœud, puis la distribution de ces valeurs. L'entropie normalisée de cette distribution mesure l'hétérogénéité des degrés : des valeurs proches de 0 indiquent une topologie quasi régulière, tandis que des valeurs proches de 1 traduisent une forte diversité des connectivités. Lorsque E_D est élevé, un mécanisme d'attention peut exploiter cette hétérogénéité en pondérant différemment les voisins lorsqu'il est faible, l'attention tend à sur-apprendre et une agrégation uniforme est préférable.

Les seuils décisifs sont déterminés empiriquement par régression logistique. Pour la proportion d'attaquants, on affecte à chaque jeu de données une étiquette binaire indiquant si l'échantillonnage de voisins surpasse l'agrégation complète, puis on ajuste une sigmoïde reliant cette étiquette à p. Le point d'inflexion de la courbe, autour de $p^* = 0{,}022$, sert de seuil décisionnel (Figure 4.2). Un processus analogue est appliqué à l'entropie de degré pour déterminer un seuil $E^* = 0{,}344$ (Figure 4.3). Le croisement de ces deux décisions (échantillonnage oui/non, attention oui/non) définit quatre configurations possibles (E-GraphSAGE ou E-GAT, avec ou sans sampling/attention). Des exemples jouets (Figure 4.6, Figure 4.7) illustrent de manière qualitative l'impact de ces choix.

Protocole expérimental.

Tous les modèles GNN sont implémentés en PyTorch avec DGL. Les graphes de flux sont construits à partir des enregistrements réseau, avec des nœuds représentant les adresses IP et des arêtes représentant les flux. Les modèles utilisent deux couches de message passing, des embeddings de nœuds de dimension 128, des activations ReLU et l'optimiseur Adam avec un taux d'apprentissage de 0,005. Lorsqu'il est activé, l'échantillonnage sélectionne 25 voisins à la première couche et 10 à la seconde. Le nombre d'époques est adapté à la taille du jeu de données (de 500 à 5000), et le meilleur modèle est choisi selon le F1 pondéré sur l'ensemble de validation.

Neuf jeux de données publics sont considérés (Table 4.3) : CIC-ToN-IoT et sa version Snapshot, CIC-IDS-2017, NF-UQ-NIDS et sa variante Unified-Community, NF-CSE-CIC-IDS2018, CCD-INID-Centrality-Focused, Edge-IIoTset et X-IIoTID. Ils couvrent un large spectre de tailles, de proportions d'attaques, de structures de graphe, ainsi que les mesures p et E_D qui guident les décisions d'ADAP-GNN. Des modèles de DL classiques (MLP, CNN, GRU, LSTM) sont entraînés dans les mêmes conditions pour servir de baselines (Table 4.4).

4. Résultats

Les résultats se déclinent en quatre volets : l'apport global des GNN par rapport aux modèles DL classiques, l'intérêt de l'initialisation par centralités, puis l'analyse détaillée des effets de l'échantillonnage de voisins et du mécanisme d'attention dans le cadre ADAP-GNN.

Le tableau Table 4.4 compare les F1-scores obtenus par les modèles MLP, CNN, GRU, LSTM et par les modèles de graphes E-GraphSAGE et E-GAT. Dans la majorité des jeux de données, les modèles GNN surpassent clairement les baselines DL, confirmant que la prise en compte explicite de la structure du graphe améliore la détection. Par exemple, sur CIC-IDS-2017, E-GAT atteint un F1 de 98,58 contre 97,11 pour le meilleur modèle convolutionnel; sur CIC-ToN-IoT et sa version Snapshot, E-GraphSAGE dépasse les modèles séquentiels et convolutionnels.

Un cas emblématique est la détection de PortScan dans CIC-IDS-2017 (Figure 4.8). Les attaques de type scan de ports sont intrinsèquement multi-hôtes et se manifestent par des motifs de connectivité dans le graphe. Les GNN capturent cette dimension relationnelle : E-GraphSAGE et E-GAT obtiennent des F1 supérieurs à 99, alors que les modèles MLP, CNN, GRU et LSTM restent autour de 87–90. Cette analyse confirme l'intérêt des GNN pour les attaques qui se traduisent par des motifs structurés dans le réseau.

L'utilisation d'une initialisation centrality-informed dans E-GraphSAGE et E-ResGraphSAGE améliore systématiquement les performances sur les quatre jeux de données étudiés (Table 4.2). Par exemple, sur NF-UQ-NIDS, l'E-ResGraphSAGE passe d'un F1 pondéré de 97,03 à 97,70 avec centralités; sur NF-CSE-CIC-IDS2018, de 97,86 à 98,51; sur X-IIoTID, l'amélioration est encore plus marquée (de 83,83 à 86,64). Ces gains s'accompagnent d'une réduction des faux positifs. Cette expérience montre que fournir aux GNN des caractéristiques de nœuds dérivées de la topologie (degré, betweenness, closeness, PageRank, K-truss) renforce leur capacité à exploiter les motifs d'attaque qui se manifestent dans la structure du graphe.

Le tableau Table 4.5 et la figure Figure 4.9 analysent l'impact de l'échantillonnage en fonction de la proportion d'attaquants. Lorsque p est très faible (par exemple 0,000077 pour CIC-ToN-IoT ou 0,000523 pour CIC-IDS-2017), l'agrégation complète obtient de meilleurs résultats : sur CIC-ToN-IoT, le F1 passe de 82,09 (échantillonné) à 84,85 (complet); sur CIC-IDS-2017, de 98,13 à 98,32. Dans ces cas, l'échantillonnage peut manquer les rares voisins malveillants, comme l'illustrent les exemples

jouets de Figure 4.6.

En revanche, lorsque la proportion d'attaquants est plus élevée (par exemple 0,0736 pour NF-UQ-NIDS-Unified-Community ou 0,7762 pour Edge-IIoTset), l'échantillonnage devient bénéfique. Sur NF-UQ-NIDS-Unified-Community, le F1 passe de 87,77 (agrégation complète) à 91,03 (échantillonnage); sur Edge-IIoTset, de 95,91 à 96,41. Dans ces régimes, les attaquants sont suffisamment nombreux pour que l'échantillonnage conserve une fraction représentative des nœuds malveillants, tout en réduisant le coût de calcul et en jouant un rôle de régularisation. La figure Figure 4.10 montre que l'échantillonnage réduit en moyenne de moitié le temps d'inférence, ce qui est crucial pour une détection en quasi temps réel. Le seuil $p^* \approx 0{,}022$ utilisé par ADAP-GNN capture bien la zone de transition entre ces deux comportements.

Le tableau Table 4.6 et la figure Figure 4.11 examinent l'impact de l'attention par rapport à l'entropie de la centralité de degré. Sur des graphes à faible entropie (par exemple Edge-IIoTset avec $E_D \approx 0{,}0218$), l'agrégation uniforme est légèrement meilleure (95,91 contre 95,21 avec attention). Dans ces cas, la structure est relativement homogène et l'attention tend à sur-apprendre certaines configurations locales sans réel gain de généralisation. Sur NF-UQ-NIDS, attention dégrade nettement la performance (84,81 contre 87,22), et les courbes de perte (Figure 4.12) montrent une baisse plus rapide de la perte d'entraînement mais une perte de validation plus élevée, signe de sur-apprentissage.

À l'inverse, sur des graphes à entropie élevée, l'attention devient bénéfique. Sur CIC-IDS-2017 ($E_D \approx 0{,}44$), E-GAT surpasse E-GraphSAGE (98,58 contre 98,32). Sur NF-UQ-NIDS-Unified-Community ($E_D \approx 0{,}69$), le F1 progresse de 87,77 à 90,78 avec attention, et sur X-IIoTID ($E_D \approx 0{,}83$), de 92,14 à 94,77. Dans ces graphes plus hétérogènes, l'attention exploite la diversité des degrés pour mettre davantage de poids sur les voisins les plus informatifs, sans se concentrer excessivement sur quelques hubs. La figure Figure 4.13 montre que le surcoût temporel de l'attention en inférence reste modéré, ce qui en fait un compromis acceptable lorsque le gain de performance est significatif.

En résumé, ce chapitre montre que la performance des GNN pour la détection d'intrusion est fortement conditionnée par les propriétés structurelles des graphes de flux. Les contributions peuvent être résumées ainsi :

- adaptation edge-centric de GraphSAGE (E-GraphSAGE) et introduction d'E-ResGraphSAGE pour préserver l'individualité des flux parallèles grâce à des connexions résiduelles sur les arêtes

- initialisation des nœuds par des mesures de centralité (degré, betweenness, closeness, PageRank, K-truss), qui améliore systématiquement les F1-scores et réduit les faux positifs sur plusieurs jeux de données
- conception d'ADAP-GNN, cadre adaptatif qui sélectionne automatiquement l'utilisation de l'échantillonnage de voisins et du mécanisme d'attention à partir de la proportion d'attaquants et de l'entropie de la centralité de degré, avec des seuils appris ($p^* \approx 0{,}022$, $E^* \approx 0{,}344$)
- démonstration empirique, sur neuf jeux de données NIDS, que cette sélection guidée par les propriétés du graphe permet d'obtenir des modèles GNN à la fois plus précis et plus efficaces que les approches DL classiques ou que des configurations GNN fixes.

Ce cadre fournit ainsi une recette pragmatique pour les NIDS basés sur des GNN : (i) modéliser les flux comme arêtes d'un graphe, (ii) préserver l'information de chaque flux par des connexions résiduelles, (iii) enrichir les nœuds par des centralités topologiques, et (iv) ajuster échantillonnage et attention en fonction de métriques simples calculées sur le graphe. Les perspectives évoquées concernent l'extension à des graphes temporels et l'intégration dans des architectures fédérées où les propriétés de graphe et les modèles sont partagés de manière respectueuse de la confidentialité.

Contents

Acknowledgments **5**

Résumé **7**

Abstract **9**

Résumé long **11**

Résumé du Chapitre I 12

Résumé du Chapitre II 17

Résumé du Chapitre III 24

Résumé du Chapitre IV 31

List of Tables **43**

List of Figures **47**

Introduction **49**

Introduction **51**

1 State of the Art **59**

1.1 Network Intrusion Detection System 61

1.1.1 Definition 61

1.1.2 Signature-based Approach 61
1.1.3 Anomaly-based Approach 61
1.1.4 AI-based Approach 62
1.2 NIDS based on AI techniques 64
1.2.1 Traditional Machine Learning Models 64
1.2.2 Deep Learning Models 66
1.2.3 Federated Learning 69
1.2.4 Graph Neural Networks 71
1.3 Complex Networks 73
1.3.1 Definition 73
1.3.2 Communities 74
1.3.3 Graph-level Measures 75
1.3.4 Node-level Measures 76
1.3.5 Community-level Measures 78
1.4 Limitation of Previous Works 81
1.4.1 (L1) Absence of Graph Context 82
1.4.2 (L2) Outdated Benchmarks 83
1.4.3 (L3) Modeling and Deployment Gaps 85

2 GDLC: Graph Deep Learning framework based on Centrality measures 87

2.1 Centrality Measures in NIDS 88
2.1.1 Motivation 88
2.1.2 Graph Construction 89
2.2 Integrating Centrality Measures in Machine Learning 91
2.3 Graph Deep Learning framework based on Centrality measures (GDLC) 94
2.3.1 GDLC Framework 95

2.3.1.1 General Scheme 95
2.3.1.2 Dynamic Measure Selection 95
2.3.1.3 Toy Example and Deep Learning Models 97
2.3.2 Experimental Setup 100
2.3.3 Results 101
2.3.3.1 Importance of Complex Network Measures 101
2.3.3.2 Classification Results and Performance Analysis 103
2.4 Conclusion 108

3 Fed-GDLC: A Federated Graph Deep Learning Framework with Centrality-Aware Features for NIDS 111

3.1 Integrating Centrality Measures in Federated Learning-Based Intrusion Detection Systems 113
3.1.1 Experimental Setup 113
3.1.2 Results 115
3.2 Fed-GDLC 116
3.2.1 Architecture of Fed-GDLC 116
3.2.2 Selection and Computation 117
3.2.3 Fed-PCA for Feature Alignment 122
3.2.4 Deep Learning Models 124
3.2.5 Experimental Setup 125
3.2.6 Results 127
3.2.6.1 Evaluating FedPCA Dimensionality: Effects on Data Reconstruction and Intrusion Detection 128
3.2.6.2 Comparative Evaluation of Fed-GDLC and Other Federated Learning Approaches 130
3.2.6.3 Evaluating Fed-GDLC Models Compared to Centralized Training . . 131
3.2.6.4 Runtime Analysis 134

3.3 Conclusion 134

4 ADAP-GNN: Adaptive Property-Aware Graph Neural Network Framework 137

4.1 Adapting Graph Neural Networks for Flow-Based Intrusion Detection 139

4.1.1 Graph Neural Networks 139

4.1.2 Problem 141

4.1.3 Edge-Variant: E-GraphSAGE 141

4.1.4 Extension: E-ResGraphSAGE (Residual Connections) 143

4.2 Centrality Informed E-GraphSAGE 143

4.3 ADAP-GNN 146

4.3.1 Adaptive selection algorithm 146

4.3.1.1 Edge-centric GNN Models 147

4.3.1.2 Graph Metrics for Model Selection 147

4.3.1.3 Selection Algorithm 149

4.3.1.4 General Scheme of ADAP-GNN 152

4.3.1.5 Toy examples 152

4.3.2 Experimental Setup 154

4.3.3 Results 156

4.3.3.1 Performance Analysis of E-GraphSAGE and E-GAT in NIDS 158

4.3.3.2 Importance of Neighbor Sampling in NIDS 159

4.3.3.3 Importance of Attention Mechanism in NIDS 160

4.4 Conclusion 164

CONCLUSION 167

Bibliography 173

List of Tables

2.1 Structural properties of the networks created from CIC-ToN-IoT and CIC-IDS2017. . 101

2.2 Result of Measures selection according to the properties of the network 104

2.3 Performance Comparison of Baseline and GDLC-enhanced Deep Learning Models for Intrusion Detection on CIC-ToN-IoT 106

2.4 Performance Comparison of Baseline and GDLC-enhanced Deep Learning Models for Intrusion Detection on CIC-IDS2017 Dataset 107

3.1 Complex networks measures used in Fed-GDLC 120

3.2 Datasets used for federated evaluation in Fed-GDLC (attacks, imbalance, description, year) 126

3.3 Fed-GDLC: Client graph properties and selected GDLC type 127

4.1 Properties of graphs from datasets used in the experiments: average degree and density. 145

4.2 Centrality-informed E-GraphSAGE Results 145

4.3 ADAP-GNN datasets and properties 157

4.4 DL vs GNN F1-score across datasets 158

4.5 ADAP-GNN: Full vs. sampled F1 by attacker proportion 160

4.6 ADAP-GNN: uniform vs. attention weights F1 by Degree Centrality entropy 162

List of Figures

1.1 Classical centralities on a synthetic graph 77

1.2 Step-by-step illustration of hierarchical measures 79

1.3 Local vs global modular centralities 81

1.4 Modular centralities on the synthetic network: (a) Comm centrality and (b) Modularity Vitality. Larger, darker nodes indicate higher scores. 82

1.5 Portscan attack 83

1.6 Botnet attack 84

1.7 DDoS attack 84

2.1 Workflow of integrating Centrality measures with ML classifiers 92

2.2 Accuracy improvement of ML models on IoT-NI dataset 93

2.3 The general flow chart scheme of the GDLC Framework 96

2.4 The measure selection scheme applied in the GDLC framework 97

2.5 GDLC toy example with centrality and hierarchical measures 99

2.6 GDLC Deep Learning models architectures: CNN, LSTM, and GRU 99

2.7 Application of SIR model in GDLC 102

2.8 Applying GDLC to CIC-ToN-IoT: Accuracy before/after CN (binary/multi) 105

2.9 Applying GDLC to CIC-ToN-IoT: FPR/FNR comparison 105

2.10 Applying GDLC to CIC-IDS2017: Accuracy before/after CN (binary/multi) 106

2.11 Applying GDLC to CIC-IDS2017: FPR/FNR comparison 107

2.12 CIC-ToN-IoT: ECDF of CN measures by label 108

3.1 FL overview: client updates, server aggregation, global model 114
3.2 CNN-LSTM hybrid architecture used in FL 115
3.3 FL: F1 over rounds (baseline vs centrality-augmented) 116
3.4 Fed-GDLC workflow: local GDLC, FedPCA alignment, federated training 118
3.5 GDLC workflow: graph build, graph measures, dynamic selection, node measures, feature augmentation 121
3.6 Fed-GDLC deep learning architectures 125
3.7 Fed-GDLC experimental setup: GDLC locally, FedPCA alignment, FL rounds 128
3.8 Fed-GDLC: Reconstruction RMSE across PCA components for Local PCA and FedPCA 129
3.9 Fed-GDLC: F1-score vs. FedPCA components 130
3.10 F1-score over federated rounds 132
3.11 Average training loss over federated rounds 132
3.12 Fed-GDLC: F1 across rounds with centralized benchmark 133
3.13 Time breakdown: feature computation vs. FL training 134

4.1 Overview of the centrality-informed E-GraphSAGE 144
4.2 Threshold of the proportion of attackers in ADAP-GNN 150
4.3 Threshold of the Degree Centrality entropy in ADAP-GNN 151
4.4 ADAP-GNN decision chart 151
4.5 General scheme of ADAP-GNN 152
4.6 Toy example: attacker proportion vs. neighbor sampling 153
4.7 Toy example: degree entropy vs. attention 154
4.8 PortScan F1 on CIC-IDS-2017 (DL vs GNN) 159
4.9 ADAP-GNN: Effect of attacker proportion on sampling performance 161
4.10 ADAP-GNN: Sampling reduces inference time 161

4.11 Attention gains by degree-entropy order 163

4.12 Overfitting signal with attention 163

4.13 E-GAT testing time: uniform vs attention 164

Introduction

Introduction

Context of the Thesis

Internet of Things (IoT) and modern networks are exposed to a rapidly expanding attack surface, where the huge number of devices, the volume of communications, and the complexity of attacks are raising new challenges to cybersecurity [1–4]. Network Intrusion Detection Systems (NIDS) stand as a critical line of defense, yet many existing approaches are limited to new unseen attacks or do not generalize in practice [5–7]. Deep learning promised more robust representations of network traffic, and early NIDS works using MLP/RNN/CNN models demonstrated meaningful gains over classic machine learning by learning features directly from flows [8–10]. However, the majority of these models operate on flat tabular inputs and disregard the relational and topological structure of communication information that is naturally expressed through graphs and is central to understanding how attacks propagate in a network [11, 12].

Despite the progress of the deep learning approaches, a critical issue persists. Most NIDS pipelines ignore the graph structure of the network, and by this, they under-utilize interaction patterns that are often discriminative for complex attacks. These per-flow models treat connections as independent samples and optimize their classification learning on the flow features, such as packet counts and byte statistics. However, Distributed Denial-of-Service (DDoS), Botnets, and other scanning attacks are hard to detect at the per-flow level, as they may seem benign, but utilizing the patterns at the complex-networks level is very useful for enhancing the classification [13]. By their nature, the IoT networks can be expressed as graphs, where devices are considered as nodes and the flows are represented as edges [14]. Analyzing these graphs using complex networks measures or data-driven solutions like Graph Neural Networks (GNNs) helps reveal attack coordination that per-flow features cannot capture.

Current deep-learning–based network intrusion detection systems either neglect graph-structural

information entirely or use it without clear guidance on which topology-derived features matter, how to inject them into deep models, and whether they improve generalization across datasets and deployment settings such as federated learning with heterogeneous client data. In addition, the construction and training of graph neural networks for intrusion detection often lack principled links to complex-network theory, leaving open questions about topology-aware modeling choices and their robustness. This thesis, therefore, aims to quantify the added value of graph-derived measures, particularly centrality measures, when combined with flow features; to test whether such topology-aware features improve robustness and cross-dataset transfer; to examine their role in federated learning under data heterogeneity and drift; and to provide principled guidance for building and training GNN–based intrusion detection systems informed by complex-network measures.

Hypotheses

We formulate four hypotheses that directly address the above gap:

(H1) **Complex-network signal aids detection:** Information derived from the network's graph structure (e.g., roles, connectivity patterns, communities) improves intrusion detection beyond what per-flow features alone can capture.
Rationale: Many attacks emerge as coordinated, higher-order interaction patterns at the network level that are invisible in isolated flows.

(H2) **Centralities strengthen deep models:** Injecting centrality measures into deep learning (DL) pipelines improves detection accuracy relative to flow-only baselines.
Rationale: centralities summarize node roles (hubs, bridges, brokers) that provide transferable, topology-aware features complementary to learned flow features.

(H3) **Topology-driven stability in federated learning:** In federated learning (FL) with heterogeneous, non-identically distributed client data, topology-aware features help mitigate distributional drift by regularizing local models toward attack-relevant structure, improving global convergence and generalization.
Rationale: shared structural priors align diverse clients more effectively than purely local flow statistics, stabilizing aggregation.

(H4) **Complex-network guidance for GNN design:** The design and training of graph neural networks (GNN) for intrusion detection can be guided by complex-network measures, yielding performance and/or efficiency gains over unguided designs.
Rationale: aligning architectural and training choices with measured topology (e.g., role signals, centrality entropy, community structure) improves the bias–variance trade-off and resource use.

Research Questions

Motivated by these hypotheses, we organize the thesis around the following questions, each of which we answer explicitly through theoretical analysis and empirical evaluation, before summarizing the answers in the Conclusion:

RQ1 Do complex-network measures carry discriminative power for NIDS? When communications are mapped to graphs, can measures such as degree, betweenness, closeness, PageRank, and k-truss help in separating benign from attack behaviors across different datasets?

RQ2 From the broad selection of measures in complex networks, are they all beneficial? What criteria can be used to select the most suitable set of measures given the properties of a graph?

RQ3 What happens to centralities under Federated Learning? In federated deployments with heterogeneous client data, do centrality-informed local models and aggregation strategies preserve accuracy and reduce variance compared to FL systems relying purely on flow features?

RQ4 How can Graph Neural Networks be adopted for NIDS? For edge-centric traffic (flows/sessions), how can the original node-centric GNNs be applied to classify edges? Could centralities assist in the training and message-passing phase?

RQ5 How to select an optimal GNN architecture? Many architectural decisions of deep learning models are available, but neighbor sampling and using an attention mechanism are two main ones when it comes to building GNNs. When to sample the neighborhood in message passing, and when not to? When to add attention weights and when to use uniform aggregation?

RQ6 How graph-based approaches tackle scalability and interpretability. At the million-edge scale, what sampling, preprocessing, and explanation methods keep training tractable while yielding faithful attributions that analysts can trust?

Thesis Objective and Contributions

This thesis aims to put network structure at the center of intrusion detection by modeling communication as graphs and using complex-network measures that capture how devices interact with each other. Beyond using these measures to augment the base flow-features, we study how they can improve the accuracy of NIDS in centralized training and in privacy-aware federated learning environments where data are heterogeneous and non-IID across sites. Building on these insights, we design and evaluate graph-informed deep learning approaches tailored to NIDS, with a particular focus on GNN models that learn the graph structure from a data-driven ground.

The thesis advances intrusion detection by integrating the network topology as both hand-engineered features to augment flow-level features and as a data-driven Graph Neural Network. We studied how to transform the raw flows into graphs and how to extract features from them. We showed how these features can improve deep models in centralized and federated learning environments. Together, the following contributions connect complex-network analysis with modern deep learning and federated training, aiming for accuracy, robustness, and practicality at IoT scale.

1. **Complex-network view of NIDS.** We develop a pipeline that converts flows into a graph (nodes: IPs/devices; edges: flows/communications). We analyzed the graphs constructed from different intrusion detection benchmarks to study how IoT devices interact.

2. **Centrality-informed deep learning.** We propose and evaluate *centrality-informed* pipelines where these complex measures that interpret the structural patterns are used to complement the flow features in training the classifiers. We propose a set of rules to select the most expressive set of centrality measures according to the properties of the graph. After that, we used these measures, which quantify the importance of nodes in the graph, to augment the feature set that we train the deep learning models on. Across widely used datasets (e.g., CIC-IDS2017 [15], ToN-IoT [16], Edge-IIoTset [17]), we show consistent gains over purely tabular deep learning baselines.

3. **Federated NIDS under non-IID.** We study federated learning in privacy-constrained multi-site settings with non-IID traffic and heterogeneous graph structure. We adapted the approach we proposed in the centralized environment, so each site (FL client) selects the most beneficial set of centrality measures to augment its feature set, and all clients collaboratively train a global

model. We analyze the results across different datasets and demonstrate that incorporating topology-aware features improves generalizability and performance of the global model.

4. **Graph Neural Networks for NIDS.** We design and assess GNN-based approaches tailored to NIDS graphs by transforming them to edge-centric. We explore multiple GNN architectures, including inductive architectures such as GraphSAGE and attention-based GNNs such as GATs [18, 19]. We study the scalability and generalizability of using neighborhood sampling, and the robustness of the attention mechanism in the message passing, the core of the GNN training process. We quantify their impact on detection quality and deployment cost across benchmarks.

Organization of the manuscript

The manuscript is organized to move from foundations and related works to methods and experiments. Each chapter builds on the previous one, connecting complex-network insights with deep and graph-based learning for practical NIDS.

- **Chapter I: State of the Art.** We survey IDS for IoT and related settings, mapping the evolution from classical ML to deep learning and outlining the limitations of previous works. Particular attention is given to the lack of topology-aware modeling and the risks of outdated or leakage-prone benchmarking. This chapter clarifies open problems that motivate a graph-centric approach to intrusion detection.

- **Chapter II: Graph Deep Learning Framework based on Centrality measures.** We formalize the complex-network perspective by constructing graphs from traffic (nodes as devices, edges as communications). Centrality and related structural measures are introduced and analyzed for discriminative power across benchmarks such as CIC-IDS2017, ToN-IoT, and Edge-IIoTset. The findings establish how graph-level measures can guide the selection of the node-level and community-level metrics, and how the integration of the complex network measures improves the intrusion detection accuracy.

- **Chapter III: Federated Learning in NIDS.** We investigate federated training in privacy-constrained, multi-site environments where data and graph structures are inherently heterogeneous. The chapter characterizes how non-IID client distributions and topology changes affect the perfor-

mance of deep learning models, and how incorporating network-aware features mitigates heterogeneity effects and improves results across clients.

- **Chapter IV: Graph Neural Networks in NIDS.** We develop GNN-based approaches tailored to NIDS graphs, including edge-centric formulations and inductive/attention architectures such as GraphSAGE and GAT. We study graph construction choices, sampling strategies, and attention mechanism. Extensive experiments quantify detection quality, scalability, and deployment trade-offs across datasets.

- **Chapter V: Conclusion.** synthesizes the answers to RQ1-RQ6, discusses limitations, and expands perspectives for future research.

List of Publications

Journal Papers

- Mortada Termos, Zakariya Ghalmane, Mohamed-el-Amine Brahmia, Ahmad Fadlallah, Ali Jaber, and Mourad Zghal. "GDLC: A new Graph Deep Learning framework based on centrality measures for intrusion detection in IoT networks." Internet of Things, vol. 26, 101214, 2024. doi:10.1016/j.iot.2024.101214

- Under-review: Mortada Termos, Zakariya Ghalmane, Mohamed-el-Amine Brahmia, Ahmad Fadlallah, Ali Jaber, and Mourad Zghal. "ADAP-GNN: Adaptive Property-Aware Graph Neural Network for Intrusion Detection in IoT Networks." Computers and Electrical Engineering.

- Under-review: Mortada Termos, Zakariya Ghalmane, Mohamed-el-Amine Brahmia, Ahmad Fadlallah, Ali Jaber, and Mourad Zghal. "Fed-GDLC: A Federated Graph Deep Learning Framework with Centrality-Aware Features for Intrusion Detection in IoT Networks." Cluster Computing.

International Conference Proceedings

- Mortada Termos, Zakariya Ghalmane, Mohamed-el-Amine Brahmia, Ahmad Fadlallah, Ali Jaber, and Mourad Zghal. "Intrusion Detection System for IoT Based on Complex Networks and Machine Learning." IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), 2023. doi:10.1109/DASC/PiCom/CBDCom/Cy59711.2023.10361433

- Mortada Termos, Zakariya Ghalmane, Mohamed-el-Amine Brahmia, Ahmad Fadlallah, Ali Jaber, and Mourad Zghal. "Integrating Centrality Measures in Federated Learning-Based Intrusion Detection Systems." IEEE Wireless Communications and Networking Conference (WCNC), 2025. doi:10.1109/WCNC61545.2025.10978723

- Mortada Termos, Zakariya Ghalmane, Mohamed-el-Amine Brahmia, Ahmad Fadlallah, Ali Jaber, and Mourad Zghal. "Enhancing IoT Network Intrusion Detection with a New GraphSAGE Embedding Algorithm Using Centrality Measures." 10th International Conference on Internet of Things, Big Data and Security (IoTBDS), SciTePress, 329-336, 2025. doi: 10.5220/0013363000003944.

- Fouad Al Tfaily, Zakariya Ghalmane, Mortada Termos, Mohamed-el-Amine Brahmia, Ali Jaber, and Mourad Zghal. "Generating Realistic Cyber Security Datasets for IoT Networks with Diverse Complex Network Properties." 10th International Conference on Internet of Things, Big Data and Security (IoTBDS), SciTePress, 321-328, 2025. doi: 10.5220/0013359000003944.

Chapter 1

State of the Art

Contenu

1.1 Network Intrusion Detection System 61
1.1.1 Definition 61
1.1.2 Signature-based Approach 61
1.1.3 Anomaly-based Approach 61
1.1.4 AI-based Approach 62
1.2 NIDS based on AI techniques 64
1.2.1 Traditional Machine Learning Models 64
1.2.2 Deep Learning Models 66
1.2.3 Federated Learning 69
1.2.4 Graph Neural Networks 71
1.3 Complex Networks 73
1.3.1 Definition 73
1.3.2 Communities 74
1.3.3 Graph-level Measures 75
1.3.4 Node-level Measures 76
1.3.5 Community-level Measures 78
1.4 Limitation of Previous Works 81
1.4.1 (L1) Absence of Graph Context 82
1.4.2 (L2) Outdated Benchmarks 83
1.4.3 (L3) Modeling and Deployment Gaps 85

This chapter reviews the main scientific and technical foundations for this thesis, establishing the theoretical context, modeling choices, and datasets of Network Intrusion Detection Systems (NIDS). The goal is to provide a coherent overview of the evolution of intrusion detection techniques, from classical rule-based systems to modern Artificial Intelligence (AI)-based solutions. Moreover, this chapter

highlights how emerging paradigms such as Graph Neural Networks (GNNs), Federated Learning (FL), and complex network analysis have transformed the field. This chapter provides background information for the rest of the thesis and explains why each component is used later. In this chapter, we relate these concepts to the techniques established in this thesis: graph neural networks (GNNs) adapted to flow graphs, federated training that employs network structure under non-IID input, and centrality-informed deep learning (GDLC). The definitions, taxonomies, and observations presented here are not only survey material, but they are repeated and used in Chapters II–IV for feature design, model selection, assessment processes, and graph building choices. This chapter establishes the conceptual and notational foundations that the proposed methods directly depend on.

1.1 Network Intrusion Detection System

The chapter begins with an introduction to NIDS, explaining its objectives, mechanisms, and categorization into signature-based, anomaly-based, and AI-based systems.

1.1.1 Definition

Network Intrusion Detection Systems (NIDS) are security tools that inspect network traffic in order to detect and report unauthorized or malicious activity. Unlike host-based intrusion detectors, which reside on individual machines, NIDS operate at key points within a network to monitor packets as they traverse from one device to another [20]. In general, a NIDS captures network flows and uses an analysis engine that applies detection rules or learned models to identify anomalies and malicious behavior. Early systems were built around static pattern matching and statistical thresholds [21], and while these approaches remain widely used, modern NIDS increasingly incorporate machine learning and other artificial intelligence (AI) techniques to improve their effectiveness against sophisticated cyber threats [22].

1.1.2 Signature-based Approach

The earliest NIDS implementations relied on signature matching: they compared the network traffic against a database of known attack patterns [23]. Signature-based systems offer high precision and produce few false positives because they explicitly encode what an attack looks like. Signature-based detectors excel at identifying known exploits, however, they struggle to recognize novel or polymorphic attacks and require continual updates to remain effective [24]. As attacks become more complex, and different mutation of new attacks are used, pattern matching becomes lacking and the limitations of signature-based NIDS become more clear.

1.1.3 Anomaly-based Approach

Anomaly-based NIDS work on detecting malicious or abnormal behavior by first modeling what is considered “normal” in network traffic and then raising alerts when deviations from that baseline are observed [5, 25]. Unlike signature-based systems that rely on predefined rules or attack patterns, anomaly-based approaches are capable of identifying previously unseen or zero-day attacks, as they

do not depend on prior knowledge of the attack signatures [26]. Instead, they learn the characteristics of benign behavior and detect anomalies based on statistical irregularities, structural deviations, or learned embeddings in high-dimensional feature spaces [27].

In the literature, a wide range of methods have been proposed for anomaly detection, each of which presents different theories about how normal and deviations occur. Anomaly detection techniques can be broadly grouped into four main categories. The first includes statistical or threshold-based methods, which rely on statistics such as mean, variance, or entropy to characterize normal activity. The second category includes clustering and density-based approaches such as k-means, DBSCAN, and Gaussian Mixture Models. These methods assume that benign traffic forms dense regions in the feature space, while attacks appear as sparse or distant outliers. The third category includes one-class models such as Isolation Forests and Autoencoders trained exclusively on normal data. The fourth is the hybrid and ensemble approaches, where the strengths of multiple paradigms are combined [6].

Despite significant progress, anomaly-based systems continue to face several practical and theoretical limitations. One of the most persistent issues is concept drift, where the definition of normal behavior evolves over time due to changes in user activity, network topology, or software updates. Unless models are periodically retrained or adapted, their detection accuracy can deteriorate. Another common problem is the high false-positive rate, since not every deviation from normalcy represents an attack, and rare but legitimate activities may be misclassified as malicious. Also, the imbalance between benign and attack samples complicates both training and evaluation, as most network traffic is non-malicious. Moreover, explainability and interpretability remain open concerns, as most modern anomaly detectors operate as black boxes. Finally, adversarial mimicry, where attackers mimic normal activities to trick the detectors and blend in [6].

1.1.4 AI-based Approach

Artificial Intelligence (AI) has emerged as a major driver in the evolution of Network Intrusion Detection Systems (NIDS), addressing many of the limitations found in both signature-based and anomaly-based approaches. While signature-based systems struggle to recognize novel threats and anomaly-based systems often suffer from high false-positive rates, AI-based NIDS leverage data-driven models capable of learning complex, non-linear relationships between network features to improve adaptability and accuracy [7, 28]. These models learn discriminative representations directly from

network traffic or flow statistics, enabling automated pattern discovery that reduces dependence on handcrafted rules or static thresholds.

Early AI-based systems primarily relied on traditional machine learning algorithms such as decision trees, random forests, support vector machines (SVM), and ensemble classifiers [29, 30]. These methods required feature engineering to extract statistical, time-based, or protocol-specific attributes from network flows. Although they achieved competitive results on structured datasets such as NSL-KDD [31] and UNSW-NB15 [32], their performance often degraded when deployed in dynamic or heterogeneous network environments. Also, they work on the assumption of stationarity and that the feature distribution remains stable over time, which rarely holds in real-world networks subject to evolving attack patterns and changing traffic behavior [33].

The emergence of deep learning significantly advanced AI-based NIDS by introducing models capable of automatic feature extraction and hierarchical pattern recognition. Convolutional Neural Networks (CNN) have been used to capture spatial correlations between features, while Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) model temporal dependencies in sequential traffic data [34, 35]. Autoencoders and Variational Autoencoders (VAE) have been adopted for unsupervised anomaly detection by reconstructing normal traffic patterns and flagging deviations with high reconstruction error [10]. These approaches demonstrated superior detection accuracy compared to classical methods, particularly when trained on large-scale datasets, yet they remain limited by their dependence on centralized data collection, heavy computational demands, and vulnerability to adversarial manipulation [10].

Recent research trends have extended AI-based NIDS beyond standalone models toward more context-aware and distributed architectures. Hybrid deep learning frameworks combine multiple neural structures, such as CNN-LSTM or CNN-GRU, to exploit both spatial and temporal correlations in network flows [9]. Meanwhile, Federated Learning (FL) has gained attention as a means of collaboratively training intrusion detection models across distributed environments without sharing raw data [36]. This paradigm enhances privacy and scalability in large IoT or industrial networks but introduces new challenges related to data heterogeneity, communication efficiency, and robustness against poisoning attacks.

Another emerging direction in AI-based intrusion detection is the exploitation of structural and relational information found in communication networks. Traditional feature vectors capture only

local characteristics of flows or hosts, ignoring the graph topology that defines how devices interact. Recent advances leverage graph-based learning and Graph Neural Networks (GNNs) to model these dependencies explicitly [37]. By representing devices as nodes and communication links as edges, GNNs learn representations that integrate both node attributes and neighborhood context, allowing the detection of subtle relational anomalies that conventional models overlook.

Overall, AI-based NIDS represent a shift from static, manually tuned detection toward adaptive, data-driven intelligence. The growing integration of deep learning, federated paradigms, and graph representations continues to push the field toward more resilient, scalable, and context-aware detection frameworks. The next section provides a detailed discussion of these AI techniques, outlining how traditional machine learning, deep learning, federated learning, and graph neural network models have each contributed to modern intrusion detection systems.

1.2 NIDS based on AI techniques

AI-driven NIDS have advanced from traditional classifiers to deep, federated, and graph-based models. In this section, we will present the related works for each category, highlighting its modeling assumptions, strengths, limitations, and deployment considerations.

1.2.1 Traditional Machine Learning Models

Classical machine learning (ML) techniques have introduced the first wave of AI-driven IDS, advancing the field beyond static signature-based methods. They require a set of engineered features such as the statistics summarizing packet sequences, protocol headers, or host behavior, and they usually operate under the supervised classification paradigm [38]. Common algorithms include k-nearest neighbors (KNN), decision trees (DTs), random forests (RFs), support vector machines (SVMs), logistic regression, naive Bayes classifiers, and gradient-boosted trees such as XGBoost [39–41]. Unsupervised approaches, such as k-means or DBSCAN clustering and one-class SVM, have also been explored for anomaly detection [42].

KNN assigns an unknown flow to the class of its closest training instances and therefore excels at discovering previously unseen intrusions without requiring explicit patterns. Decision trees and RFs learn hierarchical rules that partition the feature space, RFs combine multiple trees to improve gener-

alization and provide measures of feature importance. SVMs perform linear or non-linear separation in a high-dimensional space and support different kernel functions to handle nonlinearities. Naive Bayes assumes conditional independence of features and offers a probabilistic interpretation, while logistic regression provides a simple linear decision boundary and serves as a baseline. XGBoost builds an ensemble of trees by iteratively fitting residuals and incorporates regularization to reduce overfitting. Also, an ensemble of models can be trained to increase the model's accuracy and stability [43]. These algorithms are attractive because they are relatively easy to train, require modest computational resources and memory, and their predictions can often be interpreted by domain experts. Unsupervised methods, although less accurate, can flag novel anomalies without labeled data.

Researchers have applied these models to benchmark datasets such as NSL-KDD [31], UNSW-NB15 [32], CIC-IDS-2017[16], and CIC-DDoS2019 [44]. Talukder *et al.* propose a stacking feature-embedded random forest for big and imbalanced intrusion datasets [40]. Their pipeline uses random oversampling and principal component analysis to address class imbalance and high dimensionality and obtains 99.95% accuracy on UNSW-NB15 and 99.99% on CIC-IDS-2017. Abdelaziz *et al.* study a random forest-based NIDS for CIC-IDS-2017 and use permutation feature importance to identify 26 salient features; the resulting RF achieves a weighted F1-score of 99.8% and a macro F1-score of 93.31% while reducing the feature set by more than half [45]. Waghmode *et al.* develop a quantum-inspired least-square SVM (LS-SVM) classifier with exhaustive feature selection; their model achieves 99.3–99.5% accuracy on NSL-KDD and CIC-IDS-2017 while maintaining training times under 3s and almost perfect precision and recall [41].

Traditional ML methods often assume that the available features already capture the information needed to distinguish normal traffic from attacks. In practice, feature vectors are usually treated as independent and identically distributed (IID), so temporal patterns, packet ordering, and network topology are left out. The chosen features typically compress low-level statistics (for example, packet counts, byte volumes, and flag ratios) or incorporate domain-specific attributes crafted by experts. Many studies also assume that training data are balanced and representative of what happens in deployment, an assumption that rarely holds when attacks are relatively low. Supervised classifiers depend on labeled data, and unsupervised approaches usually presume a clear gap between normal behavior and anomalies, and that anomalous points sit far from a "normal" centroid. These simplifications make modeling easier, but they are fragile: once normal and attack distributions overlap or

shift over time, misclassifications become more likely to happen.

Classical ML algorithms bring several practical advantages to intrusion detection: they are simple to deploy, relatively easy to interpret, and inexpensive to run. With careful feature engineering, these methods can reach high accuracy on cleanly separable datasets, and their modest CPU and memory needs make them feasible to run on fog and edge devices [46]. That said, these benefits come with trade-offs: performance is highly sensitive to feature quality and class imbalance, and many early ML-based IDS relied on outdated datasets without proper re-balancing [47]. Most models also treat records independently, overlooking temporal structure and context across flows. For that reason, coordinated and slow-burning attacks may slip through undetected. In order to achieve the best results, deploying ML-based NIDS requires careful feature engineering, data pre-processing and tuning of model hyperparameters.

1.2.2 Deep Learning Models

The emergence of Deep Learning (DL) has significantly transformed Intrusion Detection Systems (IDS), pushing the boundaries of detection capabilities beyond traditional machine learning approaches. Early works explored the adoption of deep architectures such as deep neural networks (DNNs) and multi-layer perceptrons (MLPs), motivated by their ability to automatically learn complex representations of high-dimensional network traffic [48]. Unlike shallow models that heavily relied on manual feature engineering, deep learning provided end-to-end feature extraction and classification, thereby enhancing detection accuracy on modern datasets [49]. MLPs were among the first to be employed in IDS due to their scalability and ability to handle large feature spaces effectively. For example, Vishwakarma and Kesswani introduced a deep neural network-based real-time IDS for IoT environments, evaluated on several benchmark datasets. The model achieved 99.48% accuracy on NF-TON-IOT [50] and 99.21% on NF-CSE-CICIDS2018 [50], outperforming several baseline models. While MLPs are less specialized in capturing temporal dependencies, their strength lies in handling large, high-dimensional input features efficiently [51]. Building on this line of work, Luu *et al.* proposed a deep neural network trained with magnet loss to enforce better separation between benign and malicious embeddings, and validated it on recent IoT-focused benchmarks, including CIC-ToN-IoT [15], CIC-BoT-IoT [52], and CICIoT2023 [53], where it achieved strong performance and demonstrated the effectiveness of tailored representation learning objectives for modern IoT NIDS [54].

Building on this foundation, convolutional neural networks (CNNs) were adopted for IDS tasks, capitalizing on their ability to extract spatial hierarchies from input data. CNNs are particularly effective in capturing local dependencies within traffic features [55]. For instance, Qiu *et al.* combine fuzzy logic with CNN ensembles; their cooperative IDS decomposes an IoT network into subnets, selects features via fuzzy ranking and trains local CNN detectors. The ensemble exchanges detection information among observer nodes and achieves 99.72% accuracy on NSL-KDD and 98.36% on UNSW-NB15 [56]. CNNs significantly reduce reliance on handcrafted features; however, they lack mechanisms to capture long-term temporal dependencies, which are often essential in intrusion scenarios. Recurrent architectures addressed this limitation by explicitly modeling sequential dependencies. RNNs, LSTMs, and GRUs became popular choices for IDS as they could capture temporal traffic patterns more effectively than CNNs or MLPs [57, 58]. For example, [8] implemented an RNN-based IDS on the NSL-KDD dataset, achieving 83.28% accuracy for binary classification and 81.29% for multi-class classification. While less accurate than CNNs or MLPs on large-scale modern datasets, this work highlighted the sequential modeling potential of RNNs. More advanced recurrent structures such as BiLSTMs overcame some of the limitations of vanilla RNNs, particularly the vanishing gradient problem. In this direction, [59] proposed a Bi-LSTM-based collaborative IDS integrated with blockchain for IoT and cloud environments, achieving 99.41% accuracy on UNSW-NB15 and 98.91% on BoT-IoT datasets. Nevertheless, recurrent architectures are computationally expensive, and their performance degrades when tasked with learning very long dependencies, a drawback that motivated the development of hybrid approaches.

Hybrid deep models bring together convolutional feature extractors and recurrent sequence learners to capture both local patterns and temporal dynamics. A representative design is CNN-LSTM: convolutional layers act as feature extractors over per-flow or per-packet representations, distilling local patterns (e.g., byte/flag n-grams, short burst statistics), while the LSTM layers model temporal dependencies across sequences of events, allowing the detector to track how attacks unfold over time. Halbouni *et al.* report that this coupling improves detection quality on benchmark IDS datasets, achieving high accuracy with a reduced false-alarm rate in both binary and multiclass settings, compared with single-architecture baselines. In practice, such hybrids benefit from careful preprocessing (normalization, sequence construction, padding/length control), attention to class imbalance, and regularization to mitigate overfitting from the increased capacity. Deployed well, CNN-LSTM models

offer a pragmatic middle ground: richer representation power than classical ML or pure CNNs, while retaining a relatively straightforward training pipeline that can be tuned for real-time inference on constrained platforms. [60]. In parallel, unsupervised learning models such as autoencoders have been explored for scalable intrusion detection in big data contexts. Autoencoders reduce dimensionality and uncover latent feature representations that enhance downstream classification. For example, [61] proposed a stacked autoencoder followed by classification models such as SVM, decision trees, and random forest, evaluated on the UNB ISCX 2012 dataset. This hybrid autoencoder-classifier pipeline achieved competitive results in accuracy, f1-score, sensitivity, and precision, while offering efficiency for real-time intrusion detection in massive network traffic environments.

Deep models generally assume access to large, labelled datasets for training. More assumptions are needed for specific models, for example, RNNs process sequences and assume that future states depend on previous states. This requires the data to be time-sorted and then windowing traffic into fixed-length sequences. In general, deep learning models presume that the training and deployment distributions are similar and that the feature space remains stable. To address class imbalance, researchers apply oversampling, undersampling, hybrid sampling, or generative models to synthesize minority class samples. For all of these models, a good data preprocessing is needed such as features normalization and data cleaning.

Deep learning (DL) brings clear strengths to intrusion detection while introducing practical trade-offs that shape how systems are built and deployed. At their best, DL architectures learn rich, non-linear patterns and generalize to previously unseen variants. More information can be analyzed using different architectures, such as the temporal aspect that is exploited using RNNs. Yet these gains come with costs: large, clean labels are scarce; high-capacity models overfit on limited or skewed data; training is compute-intensive; RNNs can still struggle with very long horizons without careful design; confidence can be miscalibrated under distribution shift [62]; adversarial perturbations pose risks [63]; interpretability remains partial; and performance often hinges on sensitive hyperparameters and may not transfer across networks or time [26, 47]. Effective deployment therefore balances accuracy with resources: some pipelines achieve sub-35 ms inference but lean on GPUs; edge deployment may require lightweight models, careful versioning for heterogeneous hardware, and, critically, robust data pipelines [64]. Moreover, consistent normalization and sequence windowing is needed overtime.

In summary, deep learning has expanded the design space of IDS with diverse architectures, each

offering unique strengths. MLPs and DNNs excel in handling large-scale feature spaces, CNNs capture spatial correlations effectively, and RNN variants such as LSTM and BiLSTM model sequential dependencies in network traffic. Hybrid CNN-RNN models provide strong accuracy by combining spatial and temporal learning, while autoencoders enable efficient unsupervised feature extraction for large-scale deployments. However, challenges such as computational overhead, model explainability, and adaptability to evolving cyber threats remain open research directions in deep learning-based IDS.

1.2.3 Federated Learning

Federated learning (FL) is a distributed training paradigm that addresses privacy and data ownership concerns by allowing devices to collaboratively train a global model without sharing raw data. Clients locally compute model updates on their own data and share only gradients or model parameters with a central aggregator [65]. This approach works well for intrusion detection in IoT environments, vehicles, and industrial control systems where sensitive data cannot be centralized. FL also enables the utilization of heterogeneous data from multiple sources, potentially improving generalization. However, it introduces unique challenges such as communication overhead, client heterogeneity (non-IID data), and vulnerability to poisoning attacks [66].

Historically, most NIDS were trained in a centralized fashion, aggregating traffic logs in a data center. As privacy regulations tightened and data volumes exploded, transmitting raw packets to a central location became impractical. FL offers a compromise: training occurs at the data source, and only model updates are transmitted to an aggregator [67]. FL setups vary from cross-device settings to cross-silo settings. In the first, hundreds or thousands of edge devices with small local datasets and intermittent connectivity are involved in the training, while in the second a few organizations (e.g., hospitals, factories) having large datasets train the global model. The server coordinates training rounds by broadcasting the current model, collecting updates, and computing an aggregation, typically a weighted average (FedAvg) [68]. Variants such as FedProx, FedNova, and adaptive federated algorithms modify the aggregation to handle heterogeneous client behavior or learning rates [69]. In NIDS, clients may correspond to IoT gateways, autonomous vehicles, industrial control systems, or corporate edge devices, where each may observe different traffic patterns and attack types [70].

Recent work on federated learning (FL) for intrusion detection spans privacy-preserving training across heterogeneous devices, communication-efficient optimization, and robustness to non-IID data.

IoTDefender pioneers a federated transfer-learning design for 5G IoT, pushing partial models to the mobile-edge (MEC) so sites can fine-tune on local traffic while sharing only high-level knowledge, reducing raw-data exposure and improving cross-site reuse of representations [71]. In industrial CPS settings, DeepFed couples deep intrusion detectors with cross-silo aggregation to respect plant-level data locality; the authors report competitive accuracy without centralizing sensitive logs, highlighting FL's practicality for regulated environments [72]. Beyond basic FedAvg, communication and security concerns are addressed by schemes such as NIDS-FGPA, which aggregates clients using gradient-similarity with secure aggregation to curb inference attacks and dampen the impact of poisoned or highly skewed updates [73]. A complementary line optimizes the client and model side: Karunamurthy *et al.* target IoT constraints with an "optimal" FL-based IDS that balances detection gains against bandwidth and energy limits [74]. Together, these studies point to a practical recipe for FL-NIDS: reuse transferable features at the edge, secure and weight updates to handle non-IID and adversarial clients, and co-design models and protocols to fit the tight resource budgets common in IoT and industrial deployments.

FL-based IDS assumes that local clients (IoT devices, vehicles, industrial gateways) hold private data that cannot be shared with a central server. Each client trains a local model on its own traffic and sends an update to a coordinating server. The server aggregates the updates to form a new global model and broadcasts it back to the clients. This process repeats for multiple rounds until convergence. This setup quietly relies on several expectations: that clients share a common model architecture (even if local hyperparameters differ), and that the aggregator is trusted. Also, participation is partial and sometimes asynchronous, so the server must tolerate dropouts. Moreover, all parties operate in a broadly compatible feature space, since the models should have the same architecture. Above all, clients should have enough compute and memory to run a few local epochs. Finally, privacy is treated as "no raw data leaves the device," but gradients can leak information; robust aggregation (e.g., clipping, similarity screening, median-type rules) helps against poisoning and inference, though most defenses still presume that a majority of clients behave honestly.

Federated intrusion detection brings clear benefits but also practical challenges that shape how it should be designed and deployed. By training models where the data live, FL preserves privacy and regulatory compliance while reducing exposure of raw traffic that may contain personal or proprietary information. Also, it aggregates knowledge from diverse sources to improve generalization, mitigate

single points of failure, and support continual, incremental updates as new traffic arrives. Compared with centralized training, sending compressed or quantized model updates can ease bandwidth demands and lower the risk of data interception, enabling cross-silo collaboration where organizations jointly learn without sharing logs. Moreover, the local model could be tuned for the client, while generalizing the global model for all. These strengths meet head-on with constraints: non-IID data across clients can slow or destabilize convergence, and client heterogeneity creates stragglers and dropouts that bias training. Moreover, privacy is not absolute: gradients can leak information and are vulnerable to inversion or poisoning, so robust aggregation and differential privacy are used at the cost of extra compute, bandwidth, and sometimes accuracy. Model size and tuning also matter: lightweight networks may underfit, while larger architectures may be infeasible on small devices. Also, hyperparameters such as local epochs, learning rates, and aggregation frequency strongly affect fairness and convergence, yet principled guidelines for IDS remain limited [66]. Effective deployment, therefore, couples algorithmic choices with systems engineering: select an aggregation rule (e.g., FedAvg, FedProx, FedNova, FedOpt, FedSAM) that balances non-IID robustness and speed, design participation schedules that tolerate asynchrony, and prioritize reliable, informative clients. Secure channels with an appropriate privacy stack are needed, and preprocessing must remain consistent across sites. FL-based NIDS can bring a great addition to the security of IoT networks, but multiple architectural choices and deployment decisions must be taken carefully.

1.2.4 Graph Neural Networks

Graph neural networks (GNNs) are a family of deep models that learn over graphs, which are data structures made of nodes (entities) and edges (relations). Instead of treating each sample independently, GNNs pass messages along edges so that a node's representation is shaped by its neighbors and, over multiple layers [75, 76]. This lets the model capture patterns that span many entities rather than only per-record statistics [77]. In other words, GNNs extend deep learning to non-Euclidean data by learning representations on graphs. In the context of NIDS, graph construction transforms raw packets into a structured representation that exposes the relational patterns needed for topology-aware detection. A graph can be constructed by representing IP addresses, flows, or even combined modalities as nodes, while edges may encode communication (e.g., a flow from a source IP to destination IP), similarity temporal proximity (in case when flows are represented as nodes). By modeling these

interactions, GNNs encode the topology of a network and can detect attacks that have patterns across entities rather than within single flows [78].

Early research explored graph embedding and representation learning. For instance, Xiao *et al.* [79] proposed constructing first-order and second-order graphs to capture local host-level and global traffic-level perspectives, enabling the extraction of latent features for anomaly detection without expert-driven feature engineering. Another milestone was the use of GraphSAGE [18], which aggregates node neighborhood information to generate embeddings. Although initially applied to citation graphs, GraphSAGE paved the way for neighborhood-sampling strategies in intrusion detection tasks.

Building on these foundations, recent work has tailored GNNs for NIDS. E-GraphSAGE [80] extended GraphSAGE by incorporating edge features from flow data, enabling fine-grained intrusion detection in IoT networks with strong results on BoT-IoT and TON_IOT datasets. NF-GNN [81] introduced a flow graph representation specifically for malware detection, leveraging edge feature-based GNNs and achieving high supervised classification performance on CICAndMal2017. FN-GNN [82] advanced this line by initializing nodes from flow features and constructing edges through IP relationships, combining GCN and GraphSAGE operators to achieve state-of-the-art results on CIC_IDS_2017 and UNSW-NB15. Another direction explored self-supervised learning: Anomal-E [83] leveraged edge embeddings and unsupervised anomaly detection algorithms (PCA, Isolation Forest, CBLOF, HBOS) to detect both known and novel attacks, outperforming feature-based baselines on modern benchmarks.

GNN-based IDS assume that network traffic can be converted into graphs, and most models assume that the graph topology is static during training. In case of dynamic changes in network topology, they are handled by constructing separate snapshots or by incremental updates. Some models build temporal graphs where edges have timestamps and message passing occurs in chronological order. The choice of graph influences the model's capacity and complexity: fine-grained graphs capture detailed interactions but may be huge, whereas coarse-grained graphs lose information. Also, the quality of the graph construction is critical, where incorrect or incomplete graphs can mislead the model. In general, most GNN approaches rely on labeled samples for supervised training, though self-supervised and unsupervised learning are gaining popularity [84].

Graph neural networks (GNNs) offer a natural way to model how security events relate, aggregating information along edges to capture both local neighborhoods and broader topology. So behaviors that look harmless in isolation (e.g., many low-intensity flows) can reveal coordinated patterns such as

botnets, DDoS, or distributed scans once projected onto a host or service graph. At the same time, real-world use face some limitations: transforming high-volume traffic into reliable graphs is a bottleneck. Also, many models overfit to a particular topology and degrade when the network shifts, and naïve message passing scales poorly without sampling or clustering. Most models are limited to static graphs, yet, dynamic graphs complicate construction and generalization. On experimental side, datasets often contain a limited set of IP addresses that can inadvertently act as labels. While promising, the field is still in its early stages, and future research must focus on unifying graph construction methodologies, scaling models for real-time use, and leveraging self-supervised or contrastive learning to improve generalization across heterogeneous network environments [85].

The AI-based approaches reviewed above consider traffic events as independent records and learn decision boundaries in a high-dimensional feature space. However, they typically do not encode any physical or topological information about which devices talk to which. This limitation becomes critical for coordinated or multi-stage malicious behaviors such as scans, botnets, and DDoS. For this reason, in this thesis, we are exploring how to enhance NIDS using the graph information. We now turn to a complex-network view where traffic is represented as graphs, allowing us to capture connectivity patterns that feature-only models overlook. In the next section, we define what a complex network is, how to analyze it, and what measures can be used in it.

1.3 Complex Networks

In this section, we introduce complex networks as a tool to analyze systems of interacting entities in IoT networks. We then outline the network types, community concepts, and the measures used in complex networks on the graph, node, and community levels. These measures will be used later to characterize traffic and enhance intrusion detection.

1.3.1 Definition

Complex networks are sets of interconnected objects with an intricate internal organization [86, 87]. They appear widely in both natural and engineered systems and provide a common framework for modeling and interpreting phenomena across biology, social science, knowledge graphs, transportation, and information technology [88–91]. As a result, the study of complex networks has become broadly

applicable: researchers aim to uncover recurring patterns and structural principles that explore the interwoven nature of our connected world [92–109].

A network consists of points (nodes) linked in pairs by lines (edges). A complex network is one whose topology shows non-trivial connection patterns, meaning neither perfectly regular nor purely random [110], and is typically highly interconnected and heterogeneous. Interest in analyzing such systems grew with the move from small, simple graphs to networks with thousands or millions of nodes [88]. This shift was catalyzed by two widely observed real-world classes that are neither regular nor random: small-world and scale-free networks, which arise in diverse settings including social networks, the Internet's webgraph, and protein–protein interaction maps.

In real-world graphs, detecting communities is key to revealing organization beyond individual nodes. Communities are clusters of nodes with dense internal links and relatively few connections to other groups; clear, well-separated modules indicate strong community structure. Measures such as Modularity [86] and the Mixing Parameter quantify this structure. In our work, we identify communities using Infomap, a widely adopted algorithm with strong benchmark performance [111], providing practical and informative partitions for our analysis.

1.3.2 Communities

Communities are cohesive groups of nodes with many internal links and comparatively few external ones [112–114]. Defining the communities reveal how a network is organized beyond the level of individual vertices. The strength and clarity of such organization can be quantified with standard measures such as Modularity Q [86] and the Mixing Parameter, which together indicate how much connectivity is retained within groups versus routed across them. In intrusion-detection settings, a clear community structure often surfaces coordinated behaviors: traffic that appears harmless per-flow can align into distinctive patterns once viewed through community boundaries. In our work, communities are detected with Infomap [111], a well-validated method that models information flow via random walks and finds partitions that yield compact descriptions of that flow; its accuracy and efficiency on benchmark graphs make it a practical choice for large, dynamic traffic networks.

1.3.3 Graph-level Measures

These indicators summarize properties of the entire graph and give information about the interaction between nodes [115]. Capturing global structure provides a clearer understanding of the network's behavior and how best to analyze it.

a. Transitivity:

Also known as the *global clustering coefficient*, transitivity quantifies the chance that if node A connects to node B and node B connects to node C, then A also connects to C [86]. It is computed as:

$$T = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples of nodes in the network}} \tag{1.1}$$

b. Density:

Density measures what fraction of all possible edges actually appear in the network and can be interpreted as the probability that a random pair of nodes is linked [86]. Knowing the density helps characterize the graph and anticipate how centrality measures may correlate.

$$D = \frac{2E}{N(N-1)} \tag{1.2}$$

Here, E denotes the number of edges and N the number of nodes.

c. Mixing Parameter:

The mixing parameter is the proportion of edges that run between communities (inter-community edges). It takes values in $[0, 1]$, where 0 indicates perfectly separated modules and 1 indicates complete mixing with no discernible modular structure. Denoted by μ:

$$\mu = \frac{\text{number of inter-community edges}}{\text{total number of edges in the network}} \tag{1.3}$$

d. Modularity:

Modularity evaluates how strongly a network partitions into meaningful communities or modules, highlighting its modular organization [86]. Larger values signal clearer community structure compared to a random graph with the same degree sequence. With the usual notation, modularity Q is:

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta g_i g_j \tag{1.4}$$

where m is the total number of edges; A is the adjacency matrix with $A_{ij} = 1$ if nodes i and j are connected and 0 otherwise; k_i is the degree of node i; $\frac{k_i k_j}{2m}$ is the expected edge count under a null model; and $\delta g_i g_j$ is the Kronecker delta, equal to 1 when i and j belong to the same community and 0 otherwise.

1.3.4 Node-level Measures

A core task in network analysis is to understand the role and influence of each node. Node-level measures provide quantitative ways to assess this importance. Below we summarize widely used indicators, from classical centralities to hierarchical measures. Each metric offers a distinct perspective, helping to reveal how the network is organized and how nodes contribute to its behavior. Throughout this section, we illustrate concepts on a small synthetic graph generated with the LFR (Lancichinetti–Fortunato–Radicchi) benchmark [116]. Figure 1.1 visualizes the example network alongside several classical centralities.

Degree Centrality. A local centrality that counts a node's immediate neighbors. Nodes with high Degree centrality maintain many direct links, reflecting strong local connectivity and potential influence [86]. The Degree centrality C_D of node i is:

$$C_D(i) = \frac{d_i}{n-1} \tag{1.5}$$

where d_i is the number of neighbors of node i and n is the total number of nodes. To illustrate its effect, Figure 1.1(b) sizes and colors nodes by their Degree values. Degree emphasizes local neighborhoods rather than global placement, making it simple and computationally inexpensive.

Betweenness Centrality. A global measure capturing how often a node lies on shortest paths between other nodes. High-betweenness nodes act as bridges that channel flows across the graph [86, 99]. It is defined by:

$$C_B(i) = \sum_{s \neq i \neq t \in V} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{1.6}$$

where V is the node set, σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(i)$ counts those paths that pass through i. Figure 1.1(c) highlights how Betweenness can differ from Degree; for example, a node with many neighbors near the periphery (e.g., node "22") may rank lower because it seldom intermediates paths.

Closeness Centrality. A global measure of how near a node is, on average, to all others via shortest paths. Nodes with high Closeness can, in principle, spread information efficiently [86]. As seen in Figure 1.1(d), it often favors centrally located nodes. The formula is:

$$C_C(i) = \frac{n-1}{\sum_{j \neq i} d_{ij}} \tag{1.7}$$

where n is the number of nodes and d_{ij} is the shortest-path distance between i and j.

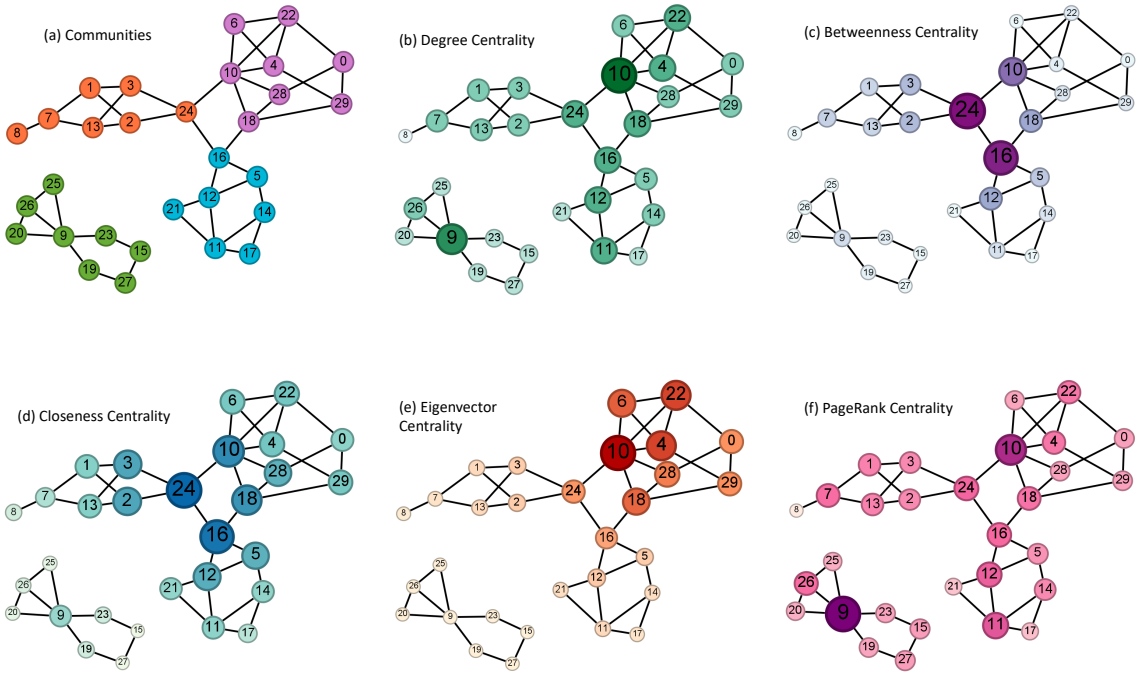


Figure 1.1: Visualization of the synthetic network and the influence of each node according to classical centralities: (a) Communities in the graph with different colors, (b) Degree, (c) Betweenness, (d) Closeness, (e) Eigenvector, and (f) PageRank. Node size and color reflect relative importance; larger and darker nodes denote higher centrality values.

Eigenvector Centrality. Not all neighbours contribute equally: connections to well-connected nodes can amplify a node's importance [86]. Figure 1.1(e) shows Eigenvector centrality C_E, where, for instance, node "10" and its vicinity receive high scores. It is calculated as:

$$C_E(i) = \frac{1}{\lambda} \sum_{j \in N(i)} A_{ij} C_E(j) \tag{1.8}$$

with λ a scalar, $N(i)$ the neighbors of i, and A_{ij} the adjacency matrix.

PageRank Centrality. A global centrality applicable to general graphs in which a node's score depends on the scores of nodes linking to it. Computed iteratively, each node's new score accumulates contributions from its incoming edges [86]. Nodes with high PageRank tend to be authoritative or influential. Figure 1.1(f) shows PageRank that is evaluated via:

$$C_{PR}(i) = \alpha \sum_j A_{ij} \frac{C_{PR}(j)}{k_j^{out}} + \beta \tag{1.9}$$

where α is a damping factor (typically in $(0, 1)$), k_j^{out} is the out-degree of node j, and β is a constant teleportation term.

Hierarchical measures evaluate node importance with respect to layered or core–periphery structure.

K-core. A hierarchical measures that represent the maximal subgraph in which every node has degree at least k. It is obtained by recursively pruning nodes with degree $< k$ until none remain, revealing densely knit, central regions of the graph [117]. The procedure is as follows:

1. Remove all nodes with degree $< k$.
2. Repeat until no such nodes remain; the residual is the k-core.

K-truss. A hierarchical measures that represent at what level the node set in terms of cohesive subgraph where edges participate in many triangles, the basic three-node building blocks. Beyond simple adjacency (as in k-core), edge strength is gauged by shared neighbors. In a k-truss, each edge belongs to at least $(k-2)$ triangles, yielding a tightly knit subgraph [115, 118]. Iterative procedure:

1. Identify all triangles in the graph.
2. For each edge, count its incident triangles; remove edges in fewer than $(k-2)$ triangles.
3. Repeat until convergence; remaining edges and endpoints form the k-truss.

Applying the hierarchical measures on the toy example is presented in the Figure 1.2.

1.3.5 Community-level Measures

Community-level measures evaluate how nodes contribute within and across their communities. Prior work shows that injecting community information into centrality definitions often improves their discriminative power [112, 119–123]. Broadly, these measures fall into three groups. *Local-aware*

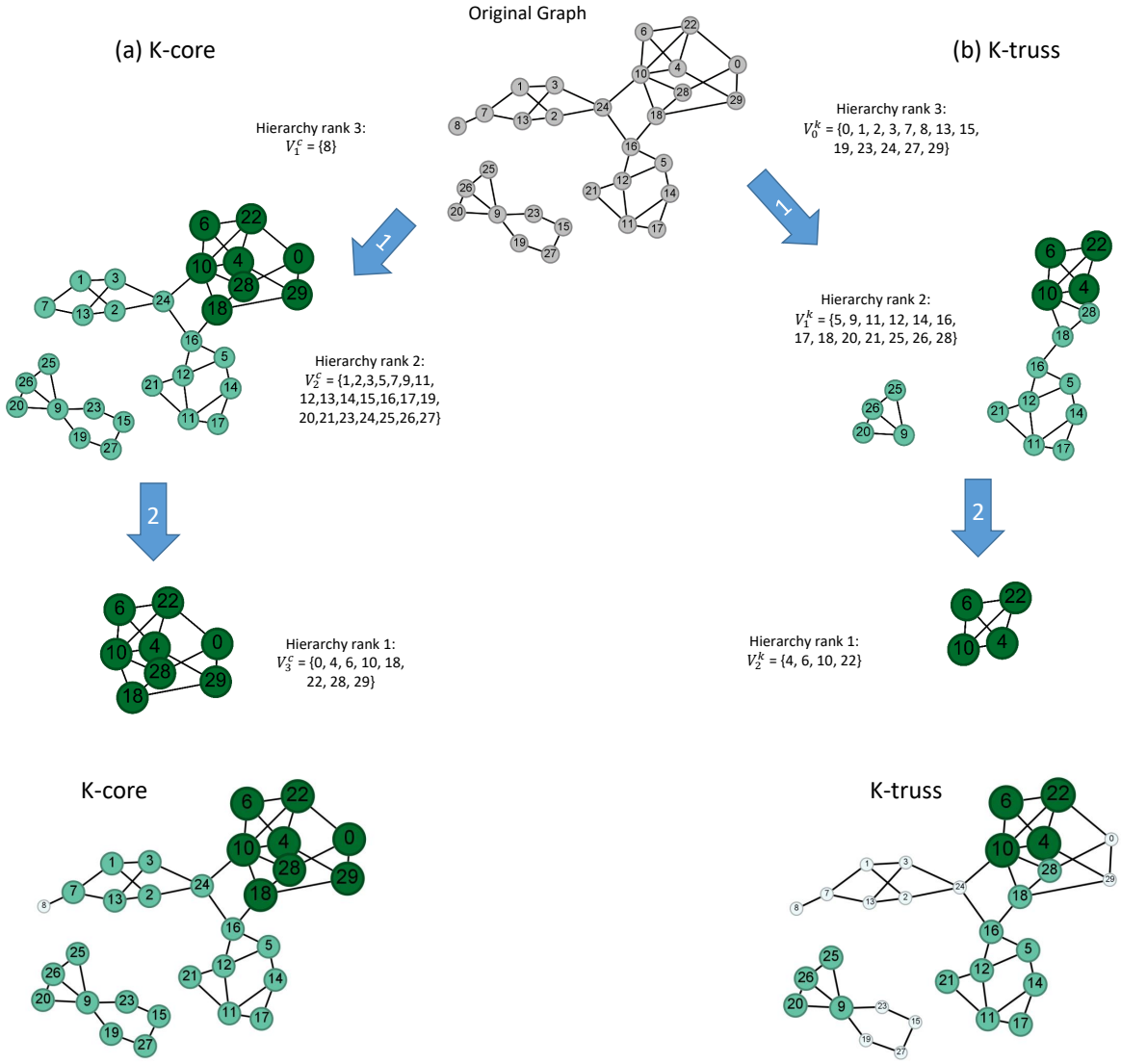


Figure 1.2: Step-by-step illustration of hierarchical levels in the synthetic network using k-core and k-truss. Nodes with higher hierarchical levels appear larger and darker.

modular centralities (e.g., the local variant of Degree) focus only on a node's links inside its own community. *Global-aware* modular centralities (e.g., the global Degree variant and Comm centrality) consider links to nodes in other communities. A third, *mixed* family accounts for both intra- and inter-community ties (e.g., Modularity Vitality).

Local Modular Centralities. These are the local components derived from classical centralities and quantify a node's influence via its intra-community edges. The local layer is formed by removing all inter-community edges (i.e., links between nodes in different communities) and treating each community as a separate graph; centralities computed on these induced subgraphs reveal influence in the immediate neighborhood [98, 122, 124]. In Figure 1.3(b),(e), Degree and Betweenness in the local view illustrate how bridge nodes lose prominence when only internal links are considered.

Global Modular Centralities. These are the global components of modular centrality, again derived from classical measures, but emphasizing inter-community structure. By removing intra-community links and retaining only edges across communities, we assess how nodes operate at the "global" level, particularly whether they act as bridges among groups. Nodes isolated in this reduced graph receive a global modular score of zero [98, 122, 124]. As shown in Figure 1.3(c),(f), hubs that are strong only within their own communities tend to drop in rank, whereas bridge nodes gain importance (e.g., nodes "10" and "12" diminish relative to their classical scores due to limited cross-community connectivity).

Comm Centrality. Proposed by Gupta *et al.* [119], Comm centrality selects nodes that are simultaneously strong hubs within their community and effective bridges across communities. It is degree-based and combines intra- and inter-community links non-linearly by squaring the inter-community term, thereby emphasizing bridging roles. This behavior is visible in Figure 1.4(a) for nodes "10", "18", "16", and "24". The score is:

$$C_C(i) = (1+\mu_c) * \left(\frac{k_i^{in}}{max_j(k_j^{out}\ \forall j \in C)} * R \right) + (1-\mu_c) * \left(\frac{k_i^{out}}{max_j(k_j^{in}\ \forall j \in C)} * R \right)^2 \tag{1.10}$$

where μ_c is the fraction of inter-community links over all links in community c, and R is a scaling constant balancing intra- and inter-community terms.

Modularity Vitality. Introduced by Magelinski *et al.* [120], this measure quantifies a node's contribution to community structure, positive or negative, using its effect on Newman's Modularity

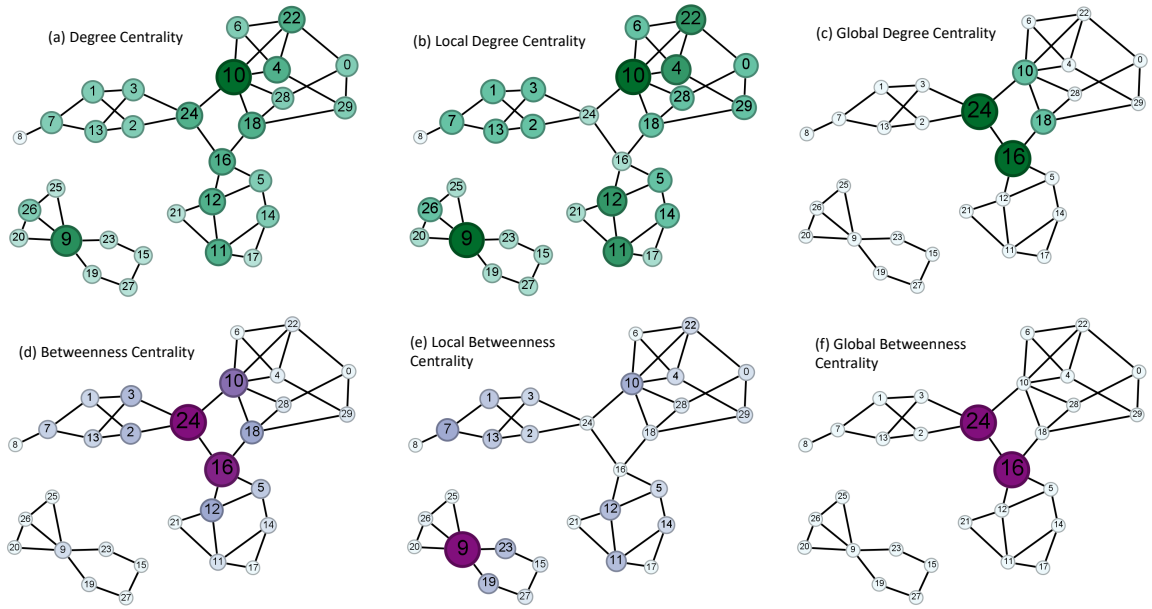


Figure 1.3: Influence of each node in the synthetic network for Degree and Betweenness and their modular counterparts: (a) Degree, (b) Local Degree, (c) Global Degree, (d) Betweenness, (e) Local Betweenness, and (f) Global Betweenness. Node color and size encode relative importance: larger, darker nodes indicate higher centrality.

[86]. Specifically, it computes the change in modularity caused by removing node i:

$$\alpha_{MV}(i) = Q(G) - Q(G \setminus \{i\}) \tag{1.11}$$

where $Q(G)$ is the modularity of the original graph and $Q(G\setminus\{i\})$ is the modularity after deleting node i. As illustrated in Figure 1.4(b), this metric tends to favour internal hubs over bridges: removing a well-connected hub often alters community quality more than removing a connector that, while important for inter-group flow, may leave the overall partition largely intact.

1.4 Limitation of Previous Works

Prior works, and especially ML/DL-based NIDS, have achieved strong results on benchmark datasets. Despite this progress, the literature still faces persistent limitations that motivate a new approach in this thesis. Below, we group them into three themes and illustrate where graph information is necessary.

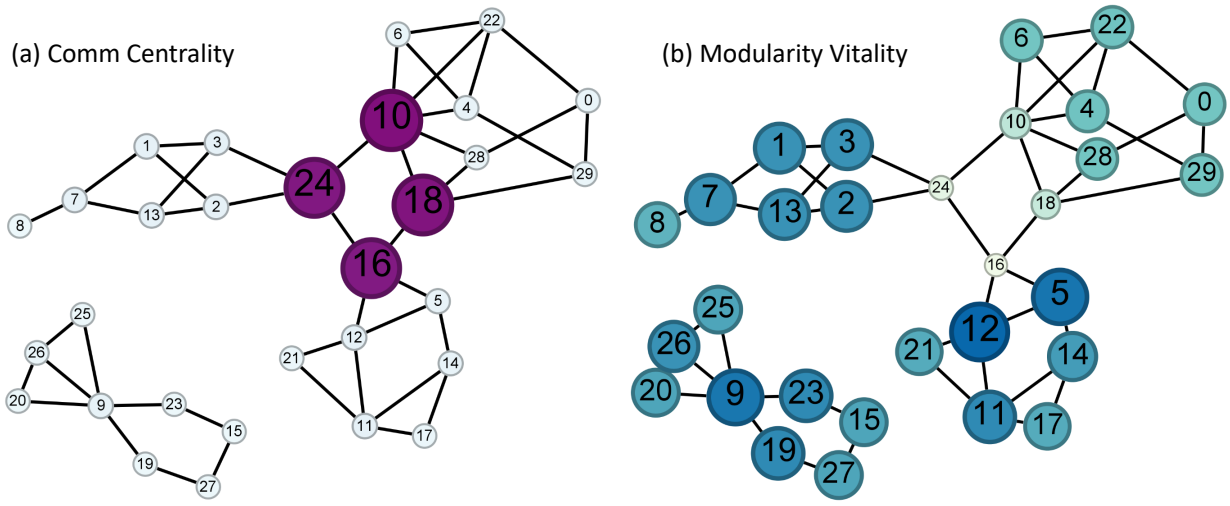


Figure 1.4: Modular centralities on the synthetic network: (a) Comm centrality and (b) Modularity Vitality. Larger, darker nodes indicate higher scores.

1.4.1 (L1) Absence of Graph Context

First and most critically, past pipelines largely ignore the communication graph among devices. Without visibility on who-talks-to-whom and how frequently, flows are treated as IID samples, and traffic is flattened into independent records. This discards the interaction context that is often where coordinated behavior lives. In such settings, detectors may confidently label each flow as benign while missing the organization that makes those flows suspicious when seen together [125]. Work in this direction has grown recently, but at the start of this thesis, it was still early-stage, leaving great room for investigation.

When per-flow features dominate, the model cannot exploit topological information such as hubs, bridges, neighborhoods, or how communities are connected. These patterns are central to practical threats:

- **Portscan:** Scans produce star-like neighborhoods; one scanner touches many distinct targets/ports in a short window, searching for vulnerabilities [126]. A graph pattern can be seen as degrees spike, and local clustering is low around the scanner, as presented in Figure 1.5.

- **Botnet:** Compromised IoT devices beacon to a small controller set and later participate in synchronized actions [127]. Many-to-few beacons (bots → controller) raising in-degree/role asymme-

try at controllers; temporal co-activation across a subset of bots; concentration of paths through controller nodes (PageRank/flow betweenness), as shown in Figure 1.6.

- **Distributed Denial-of-Service (DDoS):** Attacks amplify many-to-one or many-to-few traffic toward targets or reflectors [128, 129]. In this kind of attacks, sharp increase in in-degree and edge multiplicity at victims/reflectors, as shown in Figure 1.7.

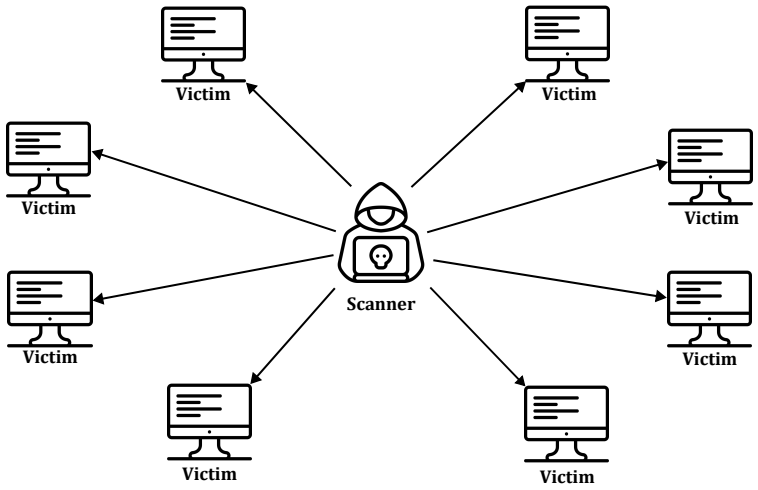


Figure 1.5: Portscan example: a single scanner fans out to multiple targets within a short time window, forming a star-like neighborhood. Each flow seen alone might be classified as benign.

These signals are invisible to feature-only IID modeling yet become explicit once flows are unified as a graph and summarized by centrality (degree, betweenness, PageRank), motif counts, or k-truss/role descriptors.

1.4.2 (L2) Outdated Benchmarks

Many studies still rely on legacy datasets with severe class imbalance, stale traffic patterns, or leakage-prone protocols, yielding optimistic numbers that do not transfer beyond the dataset itself [7, 33, 130]. Typical issues include: (i) temporal leakage (packets/flows from the same session split across train/test), (ii) host leakage (the same IP roles appearing in both splits), and (iii) class imbalance across malicious and benign activities. As a result, models overfit to the characteristics of the dataset rather than to stable behavioral patterns.

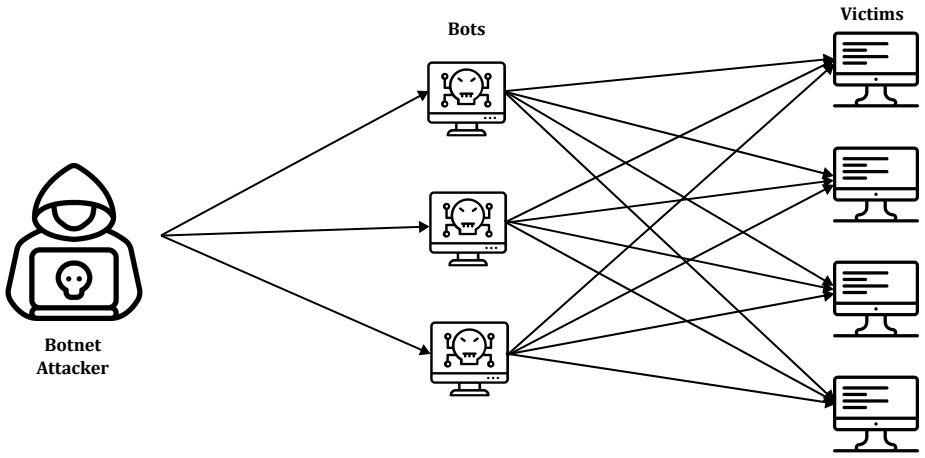


Figure 1.6: Botnet example: one controller (attacker) issues commands to three bots, and each bot targets all four victims.

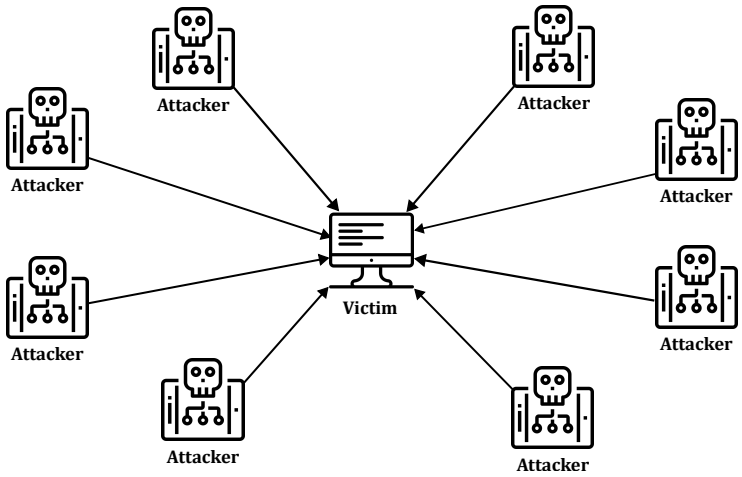


Figure 1.7: Distributed Denial-of-Service (DDoS) example: numerous sources direct traffic toward a single victim (many-to-one), creating an in-degree spike at the target.

1.4.3 (L3) Modeling and Deployment Gaps

Classical ML struggles as feature spaces grow and distributions drift, and deep models improve representation power but remain harder to interpret and calibrate. Previous works on Federated learning (FL) do not consider cross-silo experiments or using different datasets to create clients, but they split one dataset into different chunks to create clients. This experimental setup does not evaluate the global generalization of deep learning models. Also, the graph information was not tested in an FL environment to study its effect on generalization. Finally, while GNNs promise relational awareness, most pipelines still underuse the complex-network measures during graph construction and model design.

Given the early stage of GNN research in intrusion detection at the time of initiation, and the limited treatment of centrality-driven analysis in the literature, our study first established the efficacy of centrality measures under both centralized and federated training. After that, we transitioned to GNNs to broaden the investigation and contribute to the emerging graph-based line of work.

The concepts introduced in this section: complex networks, communities, centrality metrics, FL, and GNNs, directly inform our methods in the next chapters. The next chapters detail the proposed pipelines: centrality-informed deep learning (GDLC), federated training that uses network structure under non-IID data, and GNN designs for network flows, along with the datasets and evaluation protocols used in experiments.

Chapter 2

GDLC: Graph Deep Learning framework based on Centrality measures

Contenu

2.1 Centrality Measures in NIDS 88
2.1.1 Motivation 88
2.1.2 Graph Construction 89
2.2 Integrating Centrality Measures in Machine Learning 91
2.3 Graph Deep Learning framework based on Centrality measures (GDLC) 94
2.3.1 GDLC Framework 95
2.3.2 Experimental Setup 100
2.3.3 Results 101
2.4 Conclusion 108

Network intrusion detection systems (NIDS) aim to distinguish malicious behavior from benign network traffic. Recent attacks targeting Internet-of-Things (IoT) deployments and cyber-physical systems demonstrate that classical rule-based detectors are no longer sufficient, as novel attacks often do not match known signatures, and complex botnets can hide within a flood of seemingly normal flows. A promising direction is to model network traffic as a graph and analyze structural properties of this graph using tools from complex networks. Centrality measures quantify the importance of nodes or edges, and these measures can be used as additional features for machine learning and deep learning models [131]. This chapter surveys and unifies a series of recent contributions that leverage centrality in NIDS, moving from early machine learning augmentation techniques through the Graph Deep Learning framework based on Centrality measures (GDLC).

In this chapter, we begin with an introduction on graphs and centrality measures in the context of intrusion detection, presented in Section 2.1. This section provides the foundations of graph construction, the motivation for using centralities, and how they capture structural attack patterns. Building on this background, each of the following sections corresponds to a separate published paper. Specifically, Section 2.2 reviews the early experiment on the topics, where centrality measures were appended as features to traditional machine learning models. Section 2.3 presents the first contribution: the Graph Deep Learning with Centrality (GDLC) framework, which generalizes this idea by dynamically selecting and integrating diverse centralities within deep learning models.

2.1 Centrality Measures in NIDS

As IoT deployments spread throughout critical infrastructure, transportation, healthcare, and smart homes, the attack surface grows rapidly. Traditional signature-based Network Intrusion detection systems perform exact pattern matching of each network flow against a database of known malicious flows. Although this approach exhibits low false positive rates, it cannot detect zero-day or polymorphic attacks, as there is no existing signature to match. Anomaly-based detectors build a model of normal traffic and flag deviations as suspicious, offering greater flexibility at the cost of higher false alarm rates. Recently, researchers have adopted supervised machine learning (ML) and deep learning (DL) techniques to learn discriminative patterns from labeled datasets. Models like decision trees, random forests, support vector machines, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and their variants achieve superior accuracy in many benchmark studies. Despite these advances, most ML and DL models operate on per-flow features (such as durations, byte counts, flags) and ignore the relationship between devices. To expand the models to these patterns, Complex Networks analysis can be used on IoT networks.

2.1.1 Motivation

Complex network analysis provides a rich set of tools to characterize graph structure and nodes influence [86]. A complex network is a graph whose topology has non-trivial yet not completely random features, such as broad degree distributions, high clustering, community structure and small-world behavior. Many real-world networks display these properties, such as social networks, transportation networks and IoT infrastructures. To quantify how important a node is, Centrality measures are used.

Different measures give different perspectives of this importance [132]. For example, Degree Centrality counts the number of immediate neighbors. While Betweenness centrality counts the number of shortest paths that pass through a node. Eigenvector and PageRank centralities give higher scores to nodes connected to other high-scoring nodes. Also, some measures focus on the importance of the node on the community structure of a graph, such as the Modular Centrality variants. These measures capture aspects of connectivity that are not present in one-flow raw traffic features [133, 134].

Beyond flow features, attacks often has distinctive structural patterns. For instance, Denial of Service (DoS) traffic may originate from multiple compromised devices targeting a small set of servers. Attacks such as scanning and botnet propagation involve a few highly connected nodes contacting numerous victims. Also, Man-In-The-Middle (MITM) attacks appear as nodes intercepting traffic on critical paths. It is necessary to describe the network as a graph and extract data regarding the importance, influence, and position of each device within that graph in order to capture such patterns.

The work by Termos et al. [135] was one of the first to demonstrate that incorporating centrality measures into ML-based NIDS improves detection performance. They constructed a directed graph from network flows, where nodes represent IP addresses and edges represent connections. For each node they computed Degree and Betweenness centralities, then appended these values as additional features to the flow records. When training decision trees, random forests, neural networks, AdaBoost and XGBoost on the IoT-NI and Edge-IIoTset datasets, they observed that augmenting the feature set with centralities improved accuracy by up to 4.52% compared to using only standard traffic features. Motivated by this success, subsequent work generalized the idea by dynamically selecting a broader set of centralities and integrating them into deep learning models, the framework was named GDLC [136].

2.1.2 Graph Construction

The first step in leveraging centrality for intrusion detection is to convert the sequence of traffic flows into a graph. A flow summarizes multiple packets that share a source IP address s, a destination IP address d, protocol, and temporal attributes. From a set of flows $\mathcal{F}$, we build a directed graph $G = (V, E)$ as follows. Each unique IP address appearing as a source or destination forms a node $v \in V$. For each flow (s, d) we add a directed edge $e = (s, d) \in E$. If multiple flows occur between the same pair of devices, we may represent them as a single edge, assign a weight equal to the number of flows,

or represent each one as a separate edge. This representation captures the essential communication structure of the network, enabling us to later extract centrality metrics that reflect the importance and structural roles of specific devices.

A key design choice is how to represent repeated or parallel interactions between the same nodes. In a single-edge representation, all flows between (s, d) are collapsed into one directed edge. This choice reduces redundancy and ensures a compact graph, but may conceal detailed interaction dynamics. Alternatively, in a multi-graph representation, every distinct flow instance is preserved as a separate edge, which increases resilience but also amplifies the graph size and computational complexity. Finding a common space, weighted graph can be seen as a balance between these extremes.

Another dimension in graph construction concerns the temporal aspect of traffic. Since flows occur in time, it is possible to build graphs over fixed windows (e.g., every 1000 flows or every 5 minutes) and analyze them as dynamic snapshots. This window-based graph construction preserves evolving network states and makes it possible to detect anomalies that appear only within specific intervals. An alternative is the session-based graph, where flows belonging to the same session or conversation are grouped to form higher-level structures. This design is particularly useful in IoT networks, where many devices exhibit periodic communication patterns, and deviations from these patterns may signal malicious activities.

Beyond the classical flow graph, other representations can be explored to better capture the relationships between flows. One such approach is the line graph transformation, where nodes correspond to flows themselves and edges connect flows that share a common endpoint (source or destination). This construction highlights correlations between flows rather than devices and is particularly suited for intrusion detection tasks that rely on detecting abnormal communication patterns across multiple flows. However, this approach increase complexity dramatically, which reduces its applicability.

Importantly, the choice of graph representation has a direct impact on the interpretability of centrality measures. For example, in a single-edge representation, Degree centrality reflects the diversity of connections of a node, while in a weighted graph it reflects both diversity and interaction intensity. Similarly, Betweenness centrality in a multi-graph emphasizes nodes lying on multiple distinct flow paths, whereas in an aggregated graph it emphasizes critical relay devices regardless of repeated connections. The decision to use single-edge, weighted, or temporal graphs thus depends on the trade-off between scalability and accuracy. In large-scale IoT environments with millions of flows, lightweight

aggregated graphs may be preferable for real-time detection, while multigraph or line graph representations may be more suitable for offline forensic analysis.

Finally, graph construction must also address several preprocessing challenges that arise in real-world intrusion detection datasets. These include the removal of duplicate records, the exclusion of missing values, and the treatment of outliers that may distort the topology of the resulting graph. Also, the availability of IP addresses for source and destination poses a challenge, since not all datasets and benchmarks provide them explicitly. Such limitations complicate the mapping of flows to graph nodes.

2.2 Integrating Centrality Measures in Machine Learning

This section revisits our earliest experiments in combining centrality measures with traditional machine learning models for intrusion detection [135]. Machine learning models for network intrusion detection traditionally rely on statistical features extracted from packet or flow data, such as duration, packet size, or byte rate. While effective in capturing traffic behavior at the local level, these features often overlook the global structural dependencies inherent in communication patterns. To overcome this limitation, network analysis and centrality measures are integrated into machine learning workflows to enrich feature representation and improve generalization across heterogeneous IoT environments. The results provide compelling evidence that the augmentation of centrality measures improves detection and lay the foundation for the more general Graph Deep Learning framework based on Centrality measures (GDLC) presented in Section 2.3.

Once traffic is represented as a directed graph $G = (V, E)$, centrality values can be computed for every node $v \in V$ and then mapped back to the corresponding flows. Centrality measures quantify the importance of nodes within a network by analyzing their topological position. In this section we restricted attention to two widely used measures that capture complementary aspects of importance: Degree (local connectivity) and Betweenness (global reachability). When combined with traffic-based attributes, they enable classifiers to distinguish between benign and malicious entities not only by behavioral patterns but also by structural significance. We summarize the end-to-end workflow in Figure 2.1.

Traditional machine learning models were employed to evaluate the effect of augmenting features

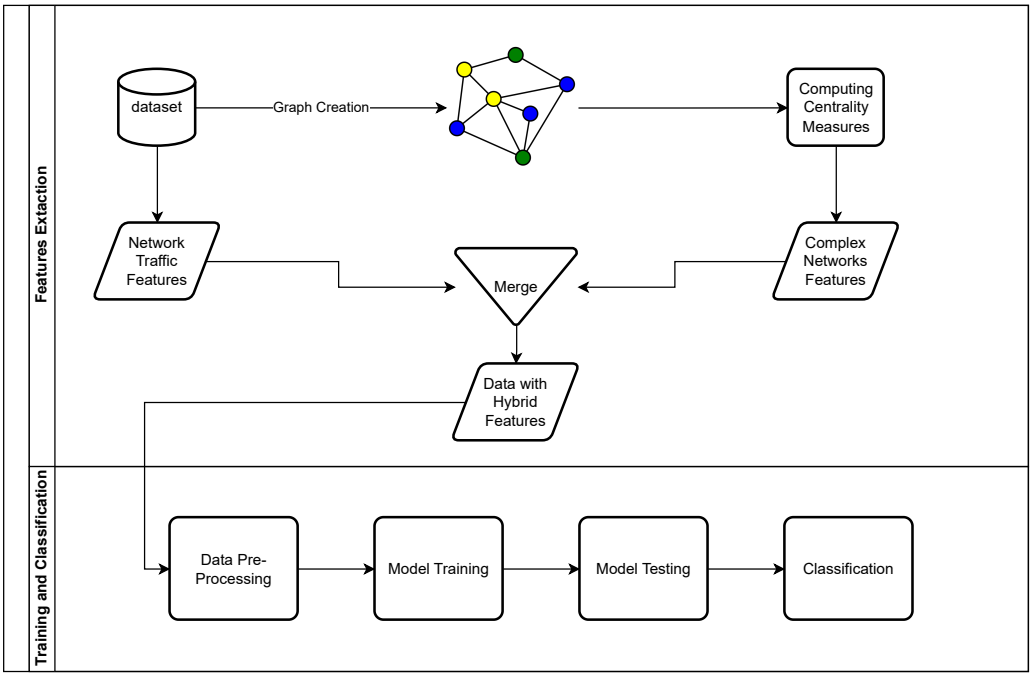


Figure 2.1: Overall workflow of integrating Centrality measures in the process of training ML classifiers. Starting from the dataset of flows, a directed graph is constructed. Centrality measures are computed from the graph to obtain complex network features. Both traffic-based and centrality-based features are then merged into a hybrid representation, which undergoes preprocessing before being used for model training, testing, and final classification.

with centralities. In this experiment we used: Decision Tree (DT), Random Forest (RF), Neural Network (NN), AdaBoost, and XGBoost. These models were selected for their complementary properties: tree-based models (DT and RF), parametric baseline model (NN), and the boosting-based models. Each model was trained using the same set of network traffic features with and without centrality augmentation, allowing a direct comparison of performance gains.

We evaluate the proposed centrality-augmented features on two benchmark IoT intrusion detection datasets, IoT-NI and Edge-IIoTset, through a series of binary classification experiments. Both IoT-NI and Edge-IIoTset differ in scale and properties, which directly influences the performance of machine learning models. IoT-NI forms a communication graph with 58,316 nodes and 58,538 edges (density 1.721×10^{-5}), a clustering coefficient of 4.55×10^{-4}, an average degree of 2.0076, and a maximum degree of 43,462 [137]. Edge-IIoTset is substantially larger, with 176,695 nodes and 178,062 edges (density 5.703×10^{-6}), a clustering coefficient of 1.634×10^{-5}, an average degree of 2.015, and a maximum degree of 163,795 [138]. As shown in Figure 2.2, the addition of centrality-based features significantly improved classification accuracy and F1-score across all datasets, with particularly notable gains in detecting low-frequency attack types. This demonstrates that centrality measures effectively capture structural cues missed by statistical flow-based attributes.

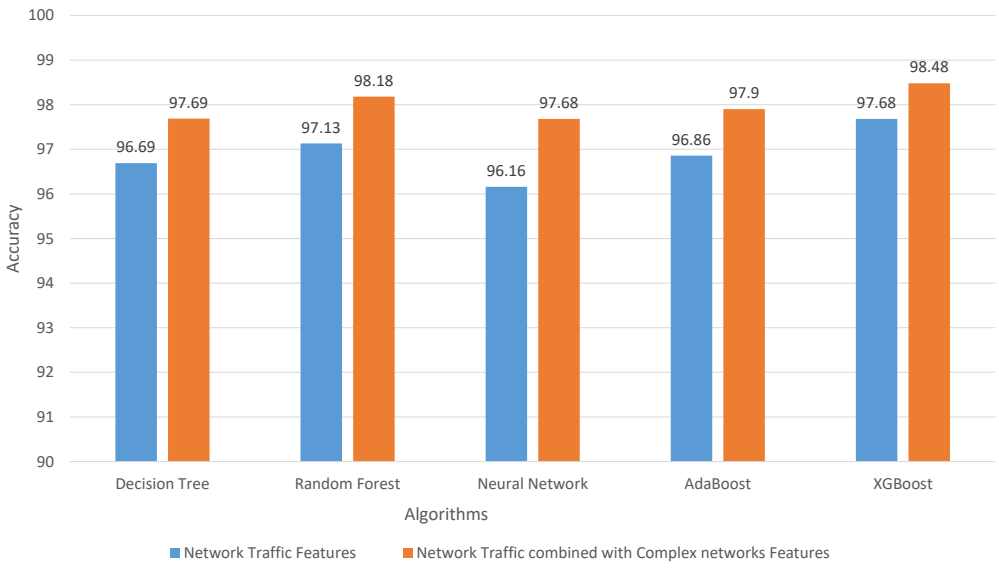


Figure 2.2: Accuracy of five machine learning models on the IoT-NI dataset before and after centrality augmentation. All classifiers benefited from the inclusion of Degree and Betweenness measures.

In summary, this early study demonstrated that incorporating simple structural features into

traditional ML workflows for intrusion detection yields tangible benefits. While the approach is limited by its reliance on a fixed choice of centralities and by the static nature of the augmentation, it provided clear evidence that graph-derived measures can enrich flow-level representations. These findings directly motivated the development of the more general Graph Deep Learning framework based on Centrality measures (GDLC) described in Section 2.3, which extends the idea of centrality augmentation to a dynamic and adaptive setting, integrates a broader range of measures, and leverages deep learning models capable of capturing temporal and hierarchical patterns in IoT traffic.

2.3 Graph Deep Learning framework based on Centrality measures (GDLC)

Building on the encouraging results of the previous section, where the augmentation of flow records with even simple centralities like Degree and Betweenness can improve intrusion detection for classical machine learning models, we now tackle one critical limitation of that approach. In the previous work, building the structural context of IoT traffic relies on a fixed set of only two centralities chosen to all datasets. However, in the Complex Networks field there are many centrality measures with different types and goals. To overcome this gap, the Graph Deep Learning framework based on Centrality measures (GDLC) was proposed to dynamically select the most suitable and informative measures based on the network's topological properties [136]. Moreover, previous work was limited to shallow ML models, while in this work modern Deep Learning models that can extract complex patterns from data were used. Specifically, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) were selected, these models have achieved impressive performance on raw traffic statistics, yet they still treat flows independently. In this way GDLC generalizes the idea of centrality augmentation and integrates it into modern deep architectures.

In the GDLC paradigm the graph is built in the same manner as in the previous section: each unique IP address becomes a node, and an edge is created for each flow from a source to a destination. Graph-level measures such as transitivity, density, and mixing parameter provide an overview of the network's structure, while node-level centrality, hierarchical measures, and modular centralities characterize the role of individual devices. Classical measures such as Degree, Betweenness, Eigenvector, Closeness and PageRank capture different aspects of influence. Hierarchical measures like K-core and K-truss identify dense substructures, and modular centralities distinguish between local and global importance with respect to community structure. A preliminary analysis using the Susceptible-Infectious-Recovered

(SIR) epidemic model confirms that immunizing high-centrality nodes greatly slows the spread of an outbreak, validating the relevance of these measures for identifying critical devices.

2.3.1 GDLC Framework

2.3.1.1 General Scheme

The pipeline of GDLC begins by converting raw network flows into a directed graph. Each unique IP address becomes a node and each flow forms a directed edge, resulting in a network that reflects the communication structure between IoT devices. Then, graph-level measures that describe the graph and connectivity of devices are computed. Using these measures, the appropriate subset of centralities and hierarchical measures is dynamically selected rather than blindly appending all available metrics. These metrics include local measures, such as Degree centrality, as well as global measures like Betweenness and PageRank, in addition to heirarchical measures such as K-core and modular centralities like Modularity Vitality. The selected centralities are then concatenated to the raw traffic features to form a rich feature vector describing each flow. Finally, this representation is fed into deep neural networks. GDLC explores three architectures: 1) a Convolutional Neural Network (CNN) that captures local patterns in the feature sequence, 2) a Long Short-Term Memory (LSTM) network to model temporal dependencies, and 3) a Gated Recurrent Unit (GRU) network that offers comparable accuracy with fewer parameters. These models operate on the augmented feature sequence, learning to discriminate between benign and malicious traffic based on both raw statistics and structural context. The overall pipeline of GDLC is illustrated in Figure 2.3. By coupling dynamic centrality selection with deep learning, GDLC aims to provide a flexible and adaptive NIDS capable of coping with diverse IoT environments [136].

2.3.1.2 Dynamic Measure Selection

Not all centrality measures carry the same information for a given network. Some may be redundant, while others may have little correlation with actual attack spread. However, computing all possible centrality and hierarchical measures is computationally expensive and often redundant, as many measures are strongly correlated. GDLC therefore incorporates a dynamic selection strategy based on the global properties of the graph. Density and transitivity indicate whether classical centralities are correlated with hierarchical measures, while the mixing parameter quantifies the strength

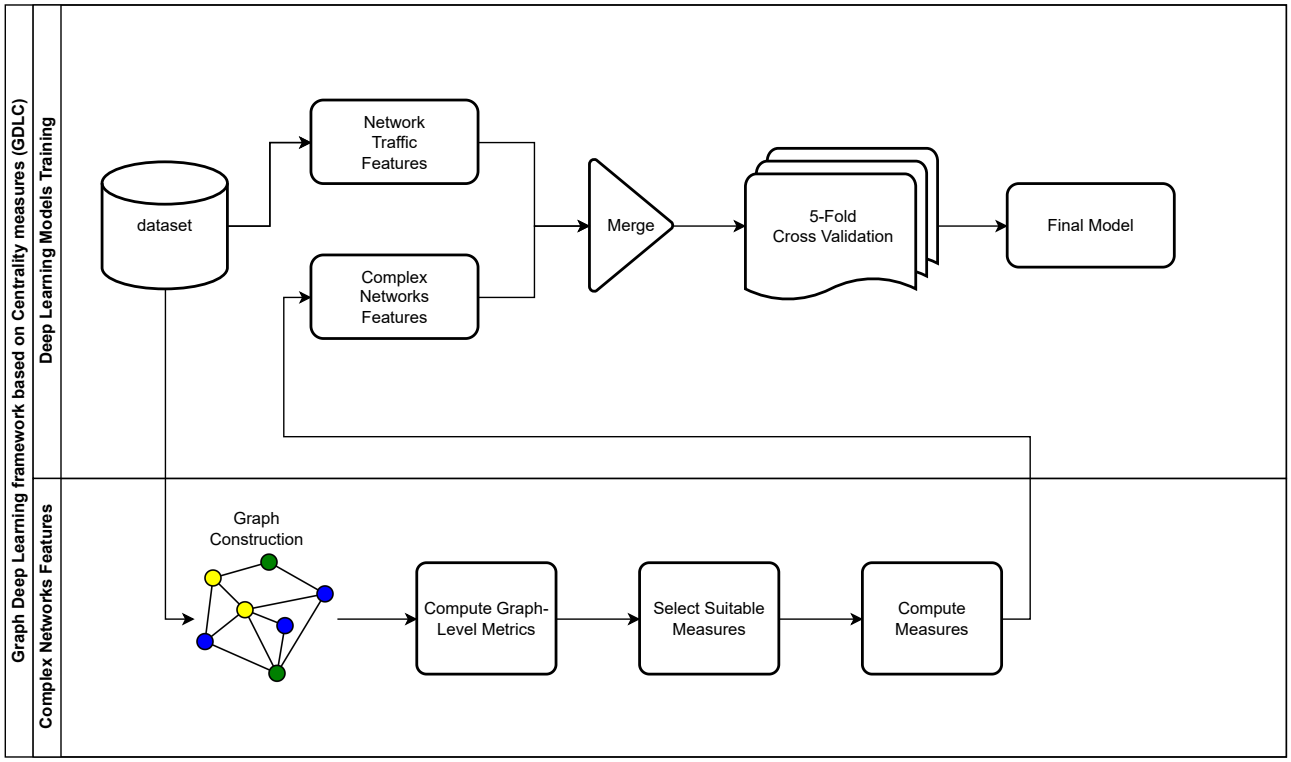


Figure 2.3: The general flow chart scheme of the GDLC Framework

Algorithm 1 Graph Deep Learning framework based on Centrality measures (GDLC)

Input: Network traffic data

Output: Intrusion detection results

1: Construct the graph from the network traffic data
   Node: Represents a network device with an IP address
   Edge: Represents one or multiple connections between nodes
2: Select and compute suitable complex network measures, based on the network properties (refer to Algorithm 2 for details)
3: Add computed measures as features to the dataset
4: Data preprocessing:
   Handle missing data
   Feature analysis
   Sort data by time
   Convert categorical features to numerical
   Normalize features
5: Train and test using cross-validation
   Apply cross-validation with appropriate algorithms
6: **return** Intrusion detection results

of community structure and guide the choice of local or global modular centralities. When density and transitivity are high, only a few orthogonal measures (Betweenness, PageRank, K-core and K-truss) are required. When these values are low, a more comprehensive set of measures is computed. Similarly, if the mixing parameter is high (weak community structure) then local modular centralities and Comm centrality are preferred, whereas a low mixing parameter prompts the use of global modular centralities and Modularity Vitality. The selection procedure is summarized in Figure 2.4, and formalized in Algorithm 2.

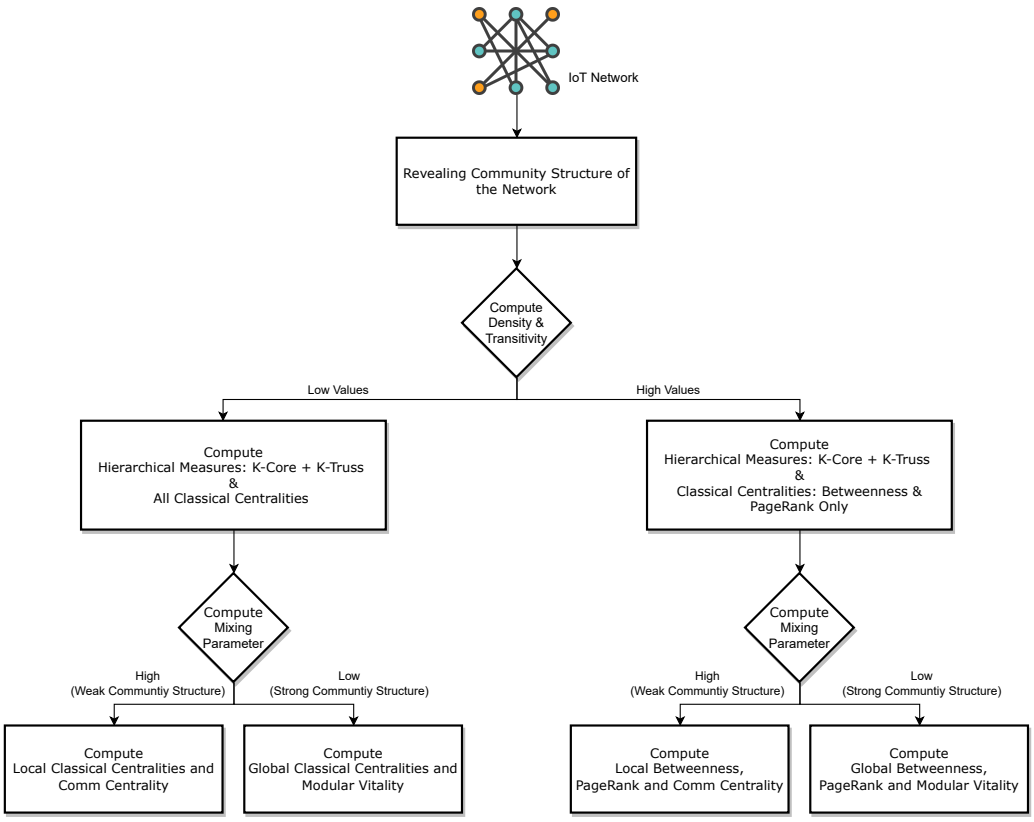


Figure 2.4: The measure selection scheme applied in the GDLC framework

2.3.1.3 Toy Example and Deep Learning Models

To illustrate how the measure selection works in practice, Figure 2.5 depicts a small directed graph along with the values of several centrality and hierarchical measures. The graph has low transitivity and density but a strong community structure; according to our selection strategy we therefore compute K-core and K-truss as well as the global variants of PageRank. The figure shows that classical

Algorithm 2 The measure selection algorithm used in the GDLC framework

Input: Graph G
Output: Selected complex network features

```
Compute graph-level measures of graph G
if G has high transitivity and density then
    Compute Betweenness, PageRank, K-truss and K-core
    if G has high mixing parameter then
        Compute Local Betweenness, Local PageRank and Comm Centrality
    else
        Compute Global Betweenness, Global PageRank and Modularity Vitality
    end if
else
    Compute all centrality and hierarchical measures
    if G has high mixing parameter then
        Compute local modular centralities and Comm Centrality
    else
        Compute global modular centralities and Modularity Vitality
    end if
end if
return Selected complex network features
```

PageRank and its local modular form produce similar values and rankings inside the communities, for that reason computing them both is redundant. Whereas global PageRank emphasizes bridge nodes compared to PageRank. It also demonstrates the orthogonality between classical centralities and hierarchical measures in sparse networks.

The selected measures are then concatenated with the original flow features and fed into deep learning models. To explore different patterns in the data, we adopted two kinds of deep learning models with different input perspectives. The one-dimensional Convolutional Neural Network (CNN) takes individual flow feature vectors as input. While the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks process sequences of flows, capturing temporal dependencies in the time-series. The architectures are shown in Figure 2.6. In all cases we use a sigmoid activation with binary cross-entropy loss for binary classification, and a softmax activation with categorical cross-entropy for multi-class classification. A discussion of computational complexity in the original paper shows that, although the cost of computing certain centralities can be as high as $O(VE)$, the overall overhead is acceptable compared to training deep learning models.

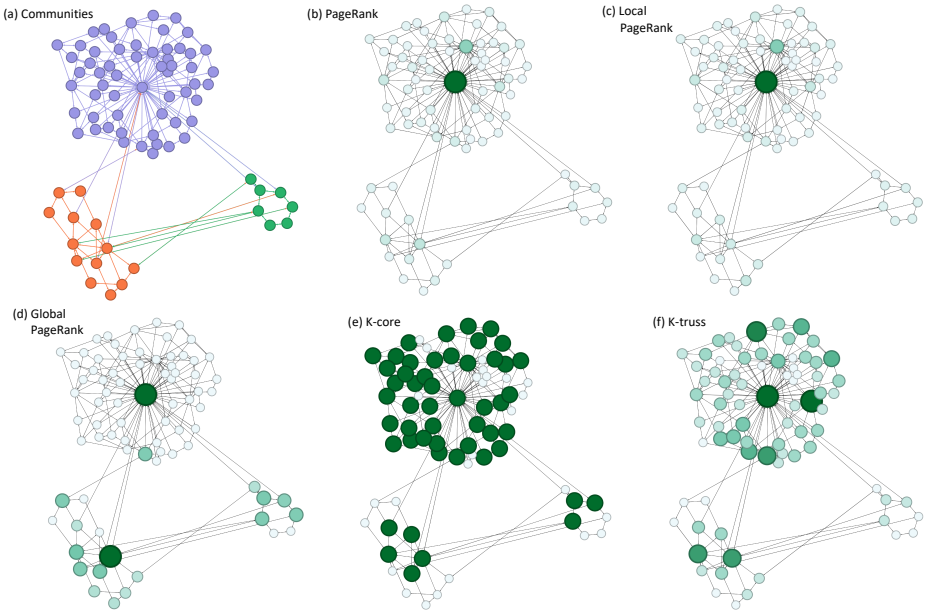


Figure 2.5: Visualization of the communities (a) in the toy example graph and the influence of each node according to the measures: (b) PageRank, (c) Local PageRank, (d) Global PageRank, (e) K-core, and (f) K-truss. The importance and influence of each node are represented by its colour and size; the larger and darker the node, the more influential it is according to these measures.

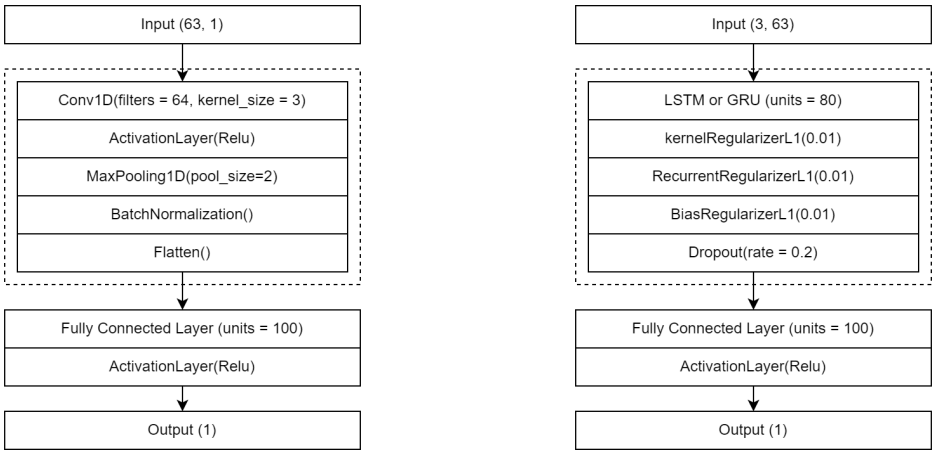


Figure 2.6: Architectures view of the adopted deep learning models: CNN, LSTM, and GRU. For binary classification we used sigmoid activation with binary cross-entropy loss. For multi-class classification we used softmax activation with sparse categorical cross-entropy loss.

2.3.2 Experimental Setup

To evaluate the GDLC framework, we conducted experiments on two publicly available intrusion detection datasets with different scales and structures: CIC-ToN-IoT [139] and CIC-IDS2017 [16]. Both datasets are provided in CICFlowMeter format, ensuring a consistent set of traffic features across experiments. The records in these datasets are network flows, which are abstractions of multiple packets sharing source and destination IPs, ports, protocol, and temporal attributes, and can be represented as edges in a directed communication graph. Below we summarize the datasets used in our study, including their origins and key characteristics:

- **TON-IoT:** Released by N. Moustafa in 2019 [139], this dataset captures heterogeneous telemetry from a realistic medium-scale Industrial IoT (IIoT) testbed organized into edge, fog, and cloud layers. It includes device telemetry, OS logs, and network traffic. A subsequent representation by M. Sarhan et al. transformed the raw PCAP traces into CSV files with CICFlowMeter features [15]. The resulting dataset comprises over 80 statistical traffic features such as timestamps, IP addresses, ports, protocols, and labels. In total, it contains 5,351,760 samples, of which 2,836,524 (53.0%) are malicious and 2,515,236 (47.0%) are benign. Nine categories of attacks are represented, including scanning, DoS/DDoS, ransomware, backdoor, injection, XSS, password cracking, and MITM.

- **CIC-IDS-2017:** Introduced by I. Sharafaldin et al. in 2017 [16], this dataset simulates real user activity for 25 individuals interacting over common protocols such as HTTP, HTTPS, FTP, SSH, and email. It is provided in two forms: raw PCAP files and fully labeled CSV files with CICFlowMeter features. The dataset contains 3,119,345 records, of which 846,248 (27.1%) are attacks and 2,273,097 (72.9%) are benign. It includes eight attack categories: FTP-Patator, SSH-Patator, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS.

Preprocessing and Graph Construction: For each dataset, we constructed a directed, unweighted graph where nodes represent devices and edges represent flows between them. Duplicate records were removed to ensure data integrity. After graph creation, original IP and timestamp columns were dropped to prevent data leakage. We normalized numerical features to ensure consistent scaling and conducted feature analysis to filter redundant attributes.

Dataset Properties: The resulting networks differ markedly in size and topology, as shown in Table 2.1 which reports their key graph-level measures. CIC-ToN-IoT is large, sparse, and exhibits a weak community structure (high mixing parameter), while CIC-IDS2017 is smaller, denser, and exhibits a strong community structure. These differences directly affect which centrality measures are most informative.

Table 2.1: Structural properties of the networks created from CIC-ToN-IoT and CIC-IDS2017.

Dataset	Nodes	Edges	Avg. Degree	Transitivity	Density	Mixing Parameter
CIC-ToN-IoT	143,809	353,604	4.9176	0.144	1.709e-05	0.567
CIC-IDS2017	14,137	64,714	9.155	0.0002	0.0003	0.0

Hardware Setup: All experiments were executed on the Numerilab workstation at CESI LINEACT (Intel®Xeon®Gold 6330 @ 2.00GHz, 65GB RAM, Windows Server 2022). This unified setup ensures a fair evaluation of GDLC across datasets, highlighting both the adaptability of the dynamic centrality selection and the complementary strengths of CNN, LSTM, and GRU in intrusion detection.

2.3.3 Results

In this section, we present the results of two complementary experiments. The first evaluates the ability of complex network measures to identify structurally important nodes within IoT graphs, using the Susceptible-Infectious-Recovered (SIR) model as a validation tool. The second investigates how incorporating the selected measures into GDLC enhances intrusion detection performance. In order to achieve that, deep learning models (CNN, LSTM, GRU) were trained and tested on traffic records both before and after augmentation with centrality features, in both binary and multi-class classification settings.

2.3.3.1 Importance of Complex Network Measures

We begin by assessing the relevance of complex network measures in identifying influential nodes within IoT communication graphs. To achieve this, we applied the Susceptible-Infectious-Recovered (SIR) epidemic model [140] on the constructed networks. The SIR model, widely used to study disease propagation, allows us to evaluate how different immunization strategies affect the spread of an epidemic through the network. In each simulation, a fraction of nodes is initially infected

and may subsequently transmit the infection with probability τ or recover with probability γ. By comparing scenarios in which nodes are immunized at random versus immunizing those ranked highest by centrality, we can directly measure the ability of these measures to pinpoint structurally critical devices.

Figure 2.7 shows the outcome on the CIC-ToN-IoT network for Degree, Betweenness, PageRank, and the hierarchical K-core measure. Immunizing top-ranked nodes consistently reduced the fraction of recovered nodes, whereas random immunization was the least effective. All measures performed well, with Degree and PageRank yielding slightly stronger effects. This behavior is consistent with the dataset's topology, which is sparse and hub-dominated, making it easier to identify the few nodes that strongly control information flow. These findings validate the use of centrality and hierarchical measures as discriminative features: they highlight critical nodes that, if compromised, can impact large parts of the network, and conversely, if protected, can limit the spread of attacks. Incorporating such information into the GDLC workflow provides complementary context to raw traffic statistics and temporal features.

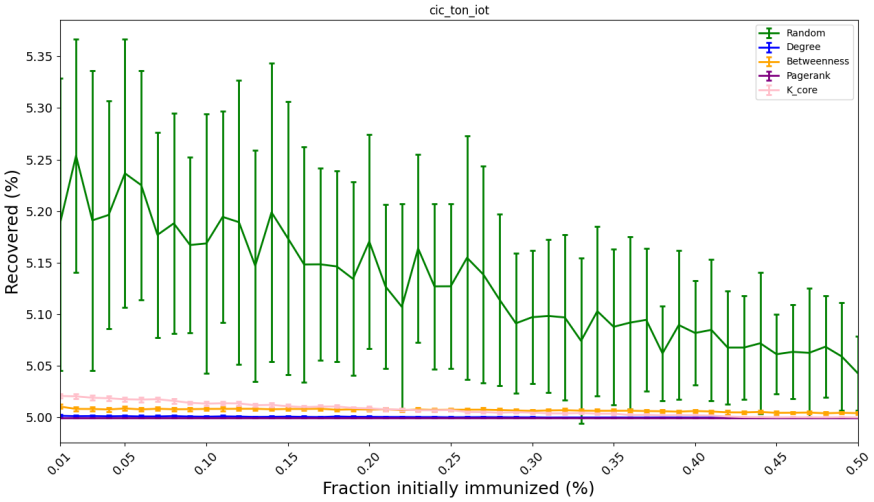


Figure 2.7: Evaluation of the SIR epidemic model on the CIC-ToN-IoT network. The figure compares random immunization with immunizing the top-ranked nodes according to different centrality measures. The infection rate was set to $\tau = 0.1$ and the recovery rate to $\gamma = 0.5$. Each curve reports the average of 30 simulations with 5% of the nodes initially infected.

2.3.3.2 Classification Results and Performance Analysis

Augmenting traffic features with dynamically selected centralities consistently improved intrusion detection across datasets and architectures. Compared to baselines trained only on CICFlowMeter features, the GDLC models achieved higher accuracy and F1-scores alongside clearly lower false positive and false negative rates. The gains are attributable to two factors: (i) the structural signal captured by centrality and hierarchical measures, and (ii) the selection policy that retains only measures aligned with the graph's topology, avoiding redundant features.

To evaluate the performance of our models, this study adopted two publicly available datasets: CIC-IDS2017, and CIC-TON-IoT. Each dataset provides a diverse set of benign and attack scenarios that allow a comprehensive assessment across multiple network environments, traffic patterns, and device heterogeneity. The experiments were designed to quantify the contribution of complex network measures when combined with standard network traffic features and to determine the extent to which centrality-aware embeddings improve detection performance in practice. Models are optimized under consistent hyperparameter settings and evaluated using Accuracy, Recall, Precision, F1-Score, False Positive Rate (FPR), and runtime. The following experiments compares baseline deep learning models (CNN, LSTM, GRU) trained solely on network traffic (NT) features against their counterparts augmented with complex network (CN) features derived from graph constructions of the underlying flows. The CN features include degree-based measures, betweenness, closeness, PageRank, k-truss, and other graph descriptors computed over the device-interaction graphs. Across datasets, the centrality-enriched variants consistently deliver higher Accuracy and F1-Score, with notable reductions in FPR, indicating improved separability between benign and malicious behaviors. These gains are robust to model choice, with recurrent and convolutional architectures both benefiting from the additional structural signal introduced by the graph-derived features.

the property-aware selection adapts the centrality set to each dataset's topology. For CIC-ToN-IoT [139], characterized by low transitivity and density with a high mixing parameter, local modular measures including Comm Centrality complement classical and hierarchical features. In contrast, CIC-IDS2017 [16] exhibits a low mixing parameter, where global modular measures and Modularity Vitality are more informative alongside Degree, Betweenness, Closeness, Eigenvector, and PageRank centrality measures and the hierarchy measures k-core and k-truss. The selected Complex Networks

Table 2.2: Result of Measures selection according to the properties of the network

Dataset	classical	modular	hierarchical
CIC-ToN-IoT	Degree, Betweenness, Closeness, Eigenvector, & Pagerank	Local Modular Centralities & Comm Centrality	K-core & K-truss
CIC-IDS2017	Degree, Betweenness, Closeness, Eigenvector, & Pagerank	Global Modular Centralities & Modularity Vitality	K-core & K-truss

measures are listed for each dataset in the Table 2.2.

Applying the proposed methodology to the CIC-ToN-IoT dataset, characterized by low transitivity and density with a high mixing parameter, confirms the positive impact of complex network (CN) features on intrusion detection. Under 5-fold time-series cross-validation in the binary setting, all deep learning backbones (CNN, LSTM, GRU) benefited from CN augmentation, as seen in Figure 2.8. Accuracy increased from the baselines to 96.67% for CNN (+6.60), 98.90% for LSTM (+5.32), and 98.68% for GRU (+5.79). Multi-class experiments showed similarly strong gains: CNN reached 85.37% (+4.95), LSTM 85.70% (+6.27), and GRU 85.66% (+5.73), with corresponding F1-Score improvements of +4.60, +5.33, and +5.31, respectively as shown in Table 2.3.

Crucially for IDS design, CN features also drove down false alarms. As shown in Figure 2.9(a), the False Positive Rate (FPR) dropped substantially compared to the traffic-only baselines; specifically, it dropped by 52.56% of its original value for CNN, 76.97% for LSTM, and 75.11% for GRU. Moreover, the False Negative Rate (FNR) improved as well (Figure 2.9(b)). These reductions indicate better separation between benign and malicious behaviors once structural information about node role and influence is injected. The relatively larger accuracy jump observed for CNN suggests that, in this dataset, local structural cues captured by CN features are particularly informative; nevertheless, the best absolute accuracies remain with LSTM/GRU, underscoring the complementary value of temporal patterns. Overall, the results validate that the graph-derived features which reflect each device's position, connectivity, and importance, consistently enhances detection performance across architectures.

For CIC-ToN-IoT specifically, LSTM was already the strongest baseline and its advantage was further reinforced with CN enrichment. This corroborates the hypothesis that structural descriptors (e.g., centrality- and hierarchy-based measures) cooperate with temporal signals from sequential traffic

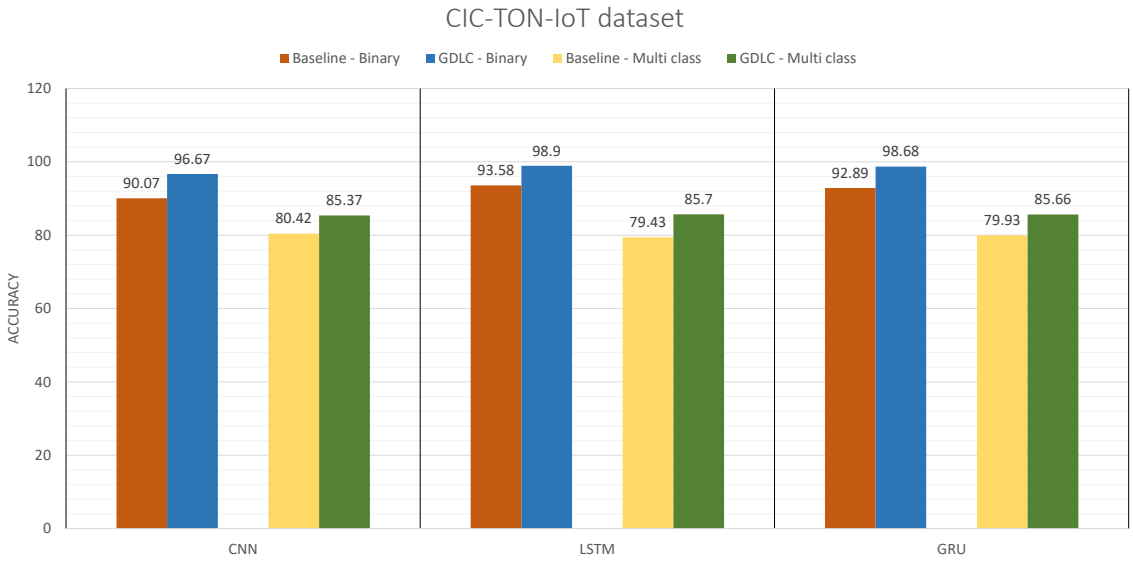
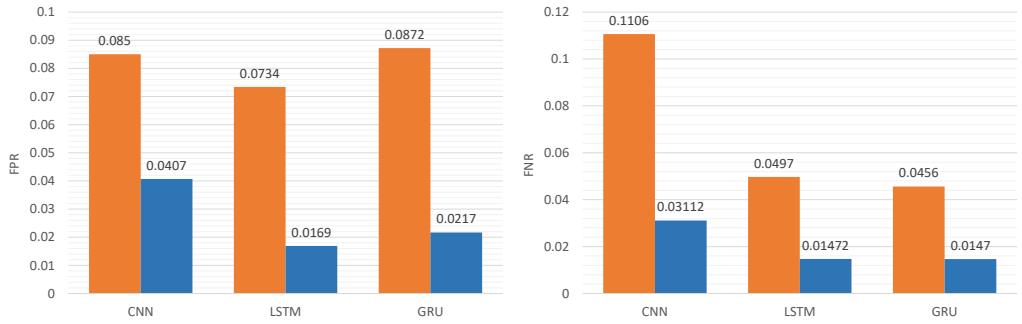


Figure 2.8: CIC-ToN-IoT dataset: Accuracy comparison before and after integrating complex network features, for both binary and multi-class classification.

to improve discriminability.



(a) False Positive Rate (FPR) (b) False Negative Rate (FNR)

Figure 2.9: FPR and FNR comparison between deep learning models on the CIC-ToN-IoT dataset.

Applying the methodology to the CIC-IDS2017 dataset (characterized by low transitivity, density, and mixing parameter), again shows clear gains from integrating complex network (CN) features. As illustrated in Figure 2.10, binary classification accuracy improves across all backbones: CNN rises by +6.18 to 98.49%, LSTM by +7.77 to 98.55%, and GRU by +2.85 to 99.05%. F1-Score also increases substantially after CN integration: +5.11 for CNN, +6.97 for LSTM, and +2.71 for GRU as shown in Table 2.4. Consistent with these trends, Figure 2.11 shows that both the False Positive Rate (FPR) and False Negative Rate (FNR) decline relative to the traffic-only baselines. In the

Table 2.3: Performance Comparison of Baseline and GDLC-enhanced Deep Learning Models for Intrusion Detection on CIC-ToN-IoT

	Binary Classification				Multi-class Classification			
Algorithm	Accuracy	F1-Score	FPR	FNR	Accuracy	F1-Score	FPR	FNR
Baseline CNN	90.07%	89.18%	0.085	0.1106	80.42%	75.89%	0.0509	0.1957
GDLC CNN	96.67%	94.37%	0.0407	0.0311	85.37%	80.49%	0.0377	0.1462
Baseline LSTM	93.58%	92.88%	0.0734	0.0497	79.43%	75.22%	0.0519	0.2056
GDLC LSTM	98.903%	98.55%	0.0169	0.0147	85.701%	80.55%	0.0393	0.1429
Baseline GRU	92.89%	91.81%	0.0872	0.0456	79.93%	75.29%	0.0530	0.2006
GDLC GRU	98.68%	98.37%	0.0217	0.0147	85.66%	80.607%	0.0396	0.1433

multi-class setting, the accuracy gains mirror the binary case: +3.10 for CNN, +1.65 for LSTM, and +0.89 for GRU. Overall, GDLC variants exhibit better FPR/FNR profiles, particularly for the temporal models (LSTM, GRU), underscoring that structural features complement temporal dynamics to enhance pattern distinguishing.

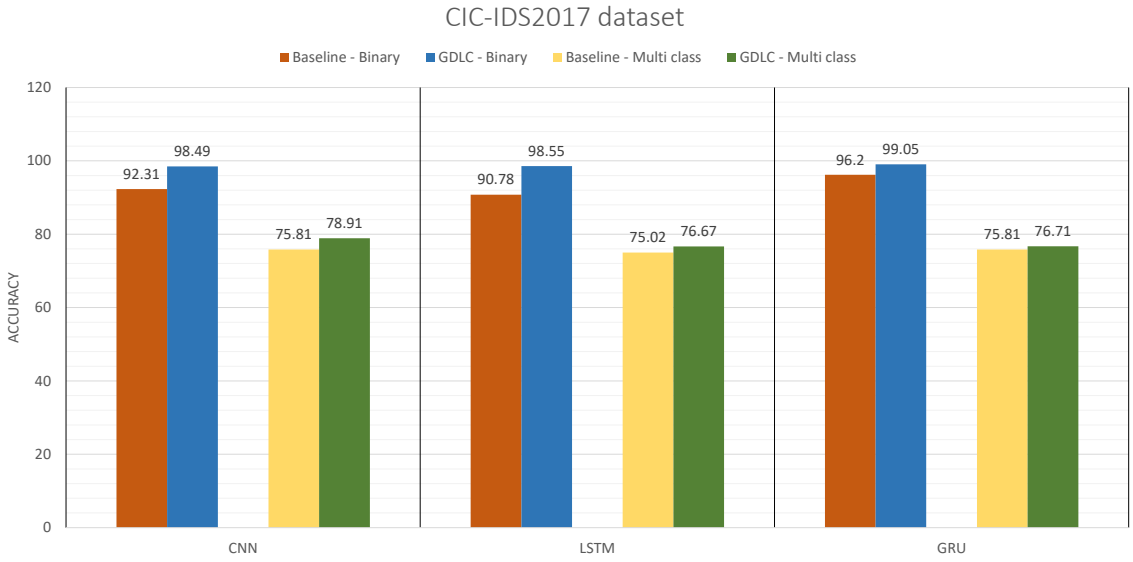
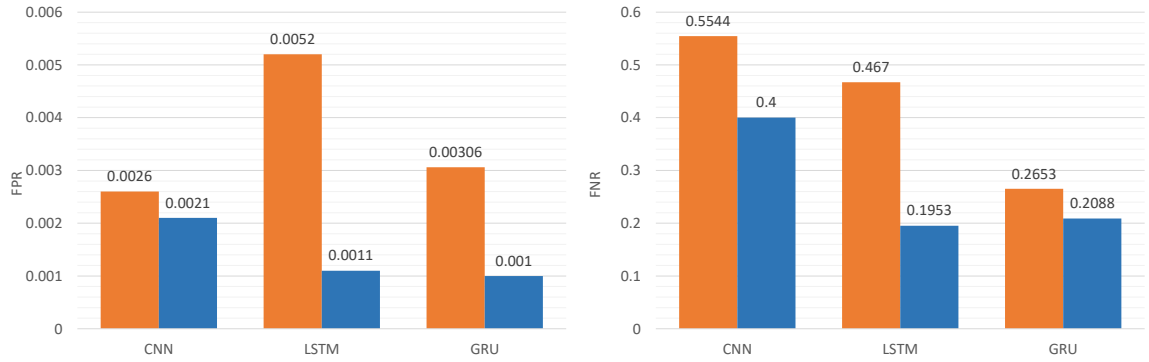


Figure 2.10: CIC-IDS2017 dataset: Comparison of Accuracy before and after integrating complex network features, for both binary and multi-class classification.

For CIC-IDS2017 specifically, LSTM exhibits the strongest improvement post-integration. This finding reinforces that providing each model with information about a node's position, influence, and relationships across the network is beneficial; the effect is especially pronounced for temporal

architectures (LSTM, GRU), highlighting the synergy between sequential traffic patterns and graph-derived structural cues.



(a) False Positive Rate (FPR)

(b) False Negative Rate (FNR)

Figure 2.11: FPR and FNR comparison between deep learning models for the CIC-IDS2017 dataset.

Table 2.4: Performance Comparison of Baseline and GDLC-enhanced Deep Learning Models for Intrusion Detection on CIC-IDS2017 Dataset

	Binary Classification				Multi-class Classification			
Algorithm	Accuracy	F1-Score	FPR	FNR	Accuracy	F1-Score	FPR	FNR
Baseline CNN	92.31%	92.702%	0.0354	0.4290	75.82%	70.51%	0.0313	0.2417
GDLC CNN	98.49%	97.81%	0.0021	0.4	78.91%	74.11%	0.0279	0.2108
Baseline LSTM	90.78%	91.46%	0.01604	0.3916	75.02%	70.104%	0.0323	0.2497
GDLC LSTM	98.55%	98.43%	0.0011	0.1953	76.67%	71.97%	0.0302	0.2332
Baseline GRU	96.20%	96.12%	0.00306	0.2653	75.81%	70.507%	0.0295	0.2418
GDLC GRU	99.05%	98.83%	0.001	0.2088	76.705%	71.99%	0.0285	0.2329

To investigate how complex network (CN) features relate to attack behavior, we applied the Empirical Cumulative Distribution Function (ECDF) to the CIC-ToN-IoT dataset. The ECDF summarizes how observations accumulate over the range of a variable and is useful for visualizing distributional differences between classes. Figure 2.12 presents ECDF curves for three CN measures: Betweenness centrality, Eigenvector centrality, and the K-core hierarchical measure. These metrics were computed on device-interaction graphs built from the IoT network. We split records by label (benign vs. attack) and plot the ECDFs separately. The curves highlight distinct relationships: (i) attack records concentrate at lower Betweenness values, indicating that attackers rarely lie on shortest paths; (ii) attacks exhibit relatively higher Eigenvector centrality, suggesting connectivity to influential neighbors; and

(iii) attackers tend to occupy high K-core values, implying membership in dense subgraphs. In short, attackers are neither bridges nor isolated, they are embedded near influential nodes in dense regions. This is an informative structural signature that complements traffic features and helps deep models uncover more discriminative patterns.

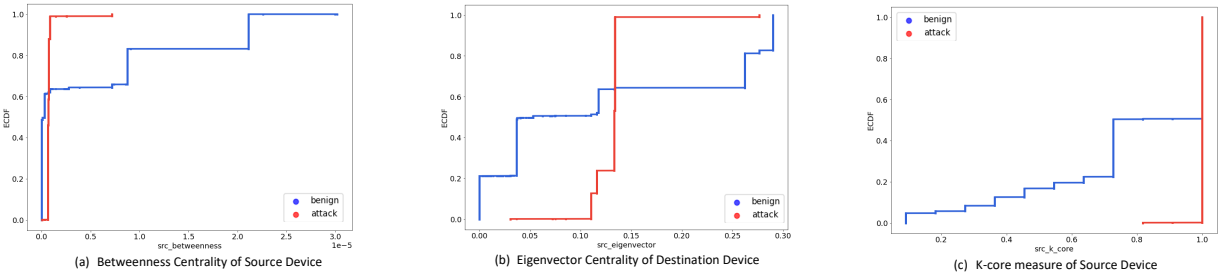


Figure 2.12: Empirical Cumulative Distribution Function (ECDF) of three complex network measures (Betweenness, Eigenvector, and K-core) on the CIC-ToN-IoT dataset, plotted separately for benign and attack labels.

In summary, GDLC couples dynamically selected CN measures with deep learning models to enhance NIDS accuracy. After constructing the graph, centralities are chosen according to the graph's measured properties; the SIR model applied on the cybersecurity datasets validated the effectiveness of these measures for identifying important nodes. We then trained CNN, LSTM, and GRU on the enriched feature sets. Across datasets, GDLC yields notable gains, accuracy improvements up to 7.7%, and consistently outperforms traffic-only baselines in both binary and multi-class settings. While CNN shows a significant accuracy increase, LSTM and GRU generally achieve the best absolute performance, reflecting the complementary between temporal dynamics and structure-aware features. Importantly, all three models exhibit a consistent and significant reduction in False Positive Rate (FPR), a crucial property for practical IDS deployments.

2.4 Conclusion

This chapter investigated how graph structure can be utilized to enhance the intrusion detection accuracy by integrating centrality measures alongside the flow features. We began by motivating using graph information in NIDS and how graph construction is done. Using these foundations, we first

revisited our earliest machine learning experiments where Degree and Betweenness were appended to flow features. On IoT-NI and Edge-IIoTset, this simple augmentation consistently improved accuracy and F1, indicating that even minimal structural features help models to detect attacks that would be missed if only regular flow features were used.

Building on that evidence, we introduced GDLC, a general framework that dynamically selects a centrality set based on measured graph properties (density, transitivity, mixing parameter) before concatenating the resulting complex-network (CN) features with traffic features and training deep models. The selection mechanism ensures selecting the least correlated classical, modular, and hierarchical measures by analyzing the topology of the graph: for instance, local modular centralities and Comm centrality are favored when community mixing is high, while global modular centralities and Modularity Vitality are selected when mixing is low. The experimental study validated these design choices on CIC-ToN-IoT and CIC-IDS2017. First, SIR-based simulations showed that immunizing nodes ranked by centrality/hierarchy reduces epidemic reach substantially more than random immunization, confirming that the selected measures pinpoint structurally critical devices. Second, when integrated into CNN/LSTM/GRU deep learning models, CN features yielded consistent gains across binary and multi-class settings. ECDF analyses provided an interpretable reasoning for the observed performance gains.

In closing, this chapter established a coherent pipeline: construct communication graphs from flows, compute graph-level descriptors, dynamically select a de-correlated set of classical, modular, and hierarchy centrality measures, and append them to traffic features for deep models. The result is a structure-aware NIDS that improves accuracy and, crucially, reduces false alarms across datasets and architectures. However, a constraint remains: GDLC assumes centralized availability of data and a homogeneous augmented feature space, which is unrealistic in multi-organization deployments with privacy and compliance constraints. The next chapter (Fed-GDLC) addresses this constraint. We keep GDLC's local graph construction and dynamic centrality selection at each client, but replace raw-feature aggregation with privacy-preserving statistical alignment to obtain a common representation for global training. This directly tackles non-IID partitions, feature heterogeneity, and data-governance limitations without moving raw traffic. Chapter 4 (ADAP-GNN) tackles another constraint, where we move from "features from graphs" to "learning on graphs" by using Graph Neural Networks (GNNs).

Chapter 3

Fed-GDLC: A Federated Graph Deep Learning Framework with Centrality-Aware Features for NIDS

Contenu

3.1 Integrating Centrality Measures in Federated Learning-Based Intrusion Detection Systems . . . 113
3.1.1 Experimental Setup . . . 113
3.1.2 Results . . . 115
3.2 Fed-GDLC . . . 116
3.2.1 Architecture of Fed-GDLC . . . 116
3.2.2 Selection and Computation . . . 117
3.2.3 Fed-PCA for Feature Alignment . . . 122
3.2.4 Deep Learning Models . . . 124
3.2.5 Experimental Setup . . . 125
3.2.6 Results . . . 127
3.3 Conclusion . . . 134

Modern networked environments, such as IoT deployments and smart cities, generate vast, sensitive, and highly heterogeneous traffic that is naturally distributed across devices, organizations, and administrative domains [141]. Centralizing such data for intrusion detection is often impractical (bandwidth, storage), risky (privacy, compliance), and sometimes legally constrained. Federated Learning (FL) addresses these constraints by enabling collaborative model training without requiring the transfer of raw data to a single location. In FL, clients compute local updates that are securely aggregated into a global model. In network intrusion detection systems (NIDS), this concept is needed:

edge routers, enterprises, hospitals, ISPs, and industrial sites can collaboratively learn robust intrusion detectors while preserving data locality [142–145].

This chapter studies FL for NIDS through two complementary works. First, we adapt our earlier centrality-aware methodology to a federated setting, showing that complex-network features computed locally over each client's communication graph remain effective under non-IID data and privacy constraints. Second, we introduce Fed-GDLC, a federated extension of the GDLC framework that dynamically selects topology-aware centralities per client and then aligns heterogeneous feature spaces with a federated PCA step before training deep models (CNN and CNN-LSTM) with server-side aggregation. In these experiments, we demonstrate why FL is well-suited for graph-enriched NIDS in decentralized IoT and smart-city infrastructures where privacy, heterogeneity, and scale are needed.

3.1 Integrating Centrality Measures in Federated Learning-Based Intrusion Detection Systems

Building on the centralized NIDS results with complex networks features presented earlier, this section carries the same methodology into a federated setting. We keep the centrality computation and feature integration, but move training to distributed clients with server-side aggregation, addressing privacy and data-movement constraints. This work highlights the ability of centrality measures to remain effective in distributed training: computed locally on client graphs, they augment the flow features, improves deep models under non-IID data, and without sharing original flow data [146].

The application of Federated Learning (FL) in NIDS represents a promising approach for achieving collaborative learning while preserving data privacy [147]. FL enables multiple clients to jointly train a model without centralizing their data. In FL, model parameters or gradients are aggregated on a central server to build a global model that captures knowledge from diverse data sources. This paradigm is particularly well-suited for NIDS, where data is distributed across different smaller networks or organizations, and centralizing traffic data may be impractical or legally constrained [148]. Training proceeds iteratively through communication rounds, during which clients compute local updates based on their local data and send them to the server for aggregation. This process continues until the global model converges to a satisfactory performance level. By carefully orchestrating the number of communication rounds, update frequencies, and aggregation strategies, it is possible to balance accuracy, convergence speed, and resource consumption, achieving collaborative learning without significant overhead [149, 150].

3.1.1 Experimental Setup

To simulate a realistic simulation, we used different datasets to act as different clients across an FL framework. In this experiment, we used two public datasets commonly used in the evaluation of intrusion detection systems: CIC-IDS2017 and CIC-Ton-IoT. The datasets were partitioned to simulate multiple clients and a server in an FL setup, reflecting heterogeneous environments. Each partition was converted into a graph based on the IP address of the network flows and a large number of features. Clients train local models on their respective partitions and share updates with a central server for aggregation. This setup allows us to observe the global model's convergence behavior across

heterogeneous data distributions and to evaluate performance and compare the baseline approach with our approach. The architecture of our simulation is presented in Figure 3.1.

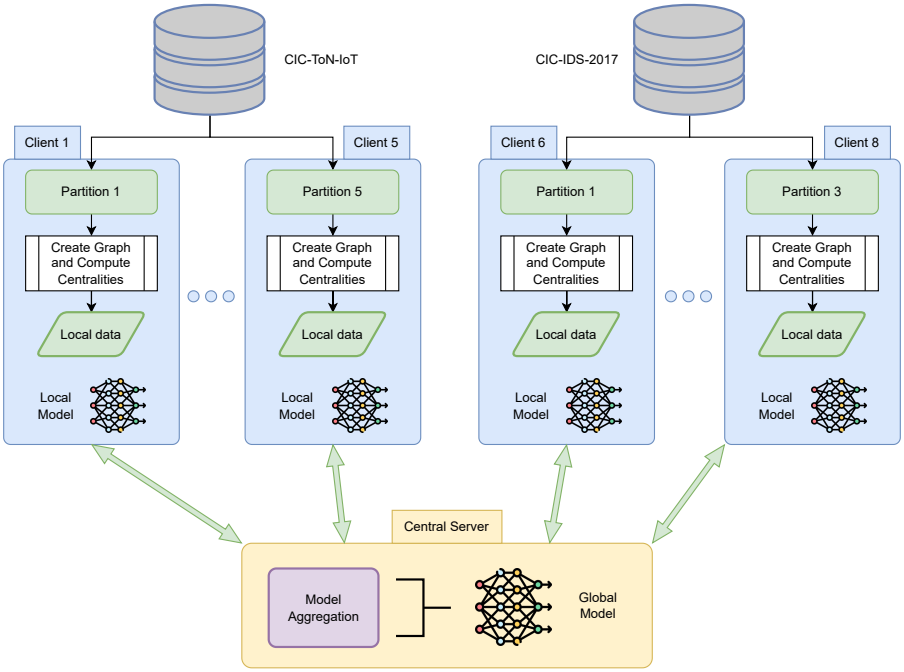


Figure 3.1: Schematic overview of our Federated Learning (FL) method showing client-server interactions, where clients train local models on private data, share updates with the server for aggregation, and receive the global model. A global model is then trained collaboratively across clients.

In our proposed approach, we utilize a hybrid architecture to learn and classify network traffic sequences by leveraging both spatial and temporal patterns present in network data. Figure 3.2 displays the hybrid model used in our experiments, it is created by combining a Convolutional Neural Network (CNN) with a Long Short-Term Memory (LSTM). The model uses the Adam optimizer and a learning rate scheduler to stabilize training dynamics. Layer Normalization is applied to improve convergence and reduce internal covariate shift, which is often more effective than Batch-Normalization in FL applications [151].

To assess the effectiveness of the integration of centrality measures into our FL-based NIDS, we utilized multiple metrics that capture complementary aspects of classification performance, including Accuracy, Precision, Recall, and F1-Score. F1-Score, the harmonic mean of Precision and Recall, is particularly informative in imbalanced datasets, as it considers both false positives and false negatives. By reporting these metrics across federated rounds and partitions, we provide a comprehensive view of

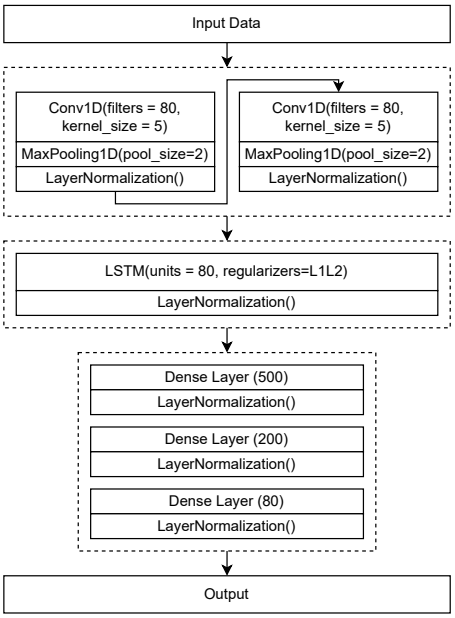


Figure 3.2: Architecture of the adopted CNN-LSTM deep learning model for intrusion detection. The model uses Layer Normalization and Dropout, is trained with the Adam optimizer and a learning-rate scheduler, and employs Softmax with sparse categorical cross-entropy loss.

how the model evolves and generalizes in the presence of heterogeneous client data. In our study, we illustrated the performance improvements through comparative plots between the baseline traffic-only model and our centrality-augmented approach, highlighting the enhanced accuracy and reliability of our FL-based NIDS.

3.1.2 Results

In this study, we compared the performance of a baseline federated learning-based NIDS with a novel approach that incorporates centrality measures into the model's feature space [146]. The evaluation was conducted over multiple communication rounds in the federated setting, tracking the evolution of the models' F1-scores over time. As shown in Figure 3.3, the results demonstrate that the integration of centrality measures consistently improves detection performance compared to the baseline. The centrality-augmented model shows a higher F1-score and achieves better stability in later rounds, indicating a more robust generalization across heterogeneous client distributions.

The above results motivate a deeper analysis of design choices and their deployment in a federated learning (FL) environment. The next chapter introduces Fed-GDLC, where centrality measures are

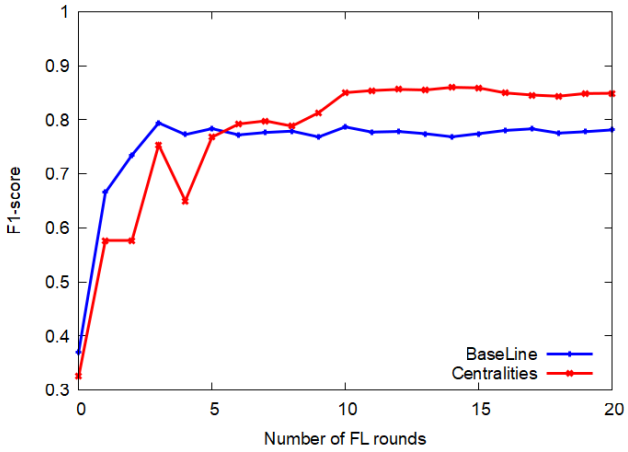


Figure 3.3: F1-score comparison over multiple federated learning rounds for the baseline and the centrality-augmented models, demonstrating higher overall model performance due to the integration of centrality measures.

selected dynamically using GDLC at each client. This gives rise to heterogeneous feature sets across clients, which we address using FedPCA to align and unify the integrated features into a common representation.

3.2 Fed-GDLC

Following the first FL experiment that integrated a fixed set of centrality measures (degree, betweenness, PageRank) into the flow-features, Fed-GDLC generalizes the approach by adapting the GDLC framework into federated settings. In Fed-GDLC, clients compute and integrate topology-aware centrality features locally, while the server performs aggregation. To ensure a common features space, Fed-GDLC aligns the features using a Federated version of PCA, without sharing raw flows. The outline of the workflow of Fed-GDLC can be presented in three phases: (i) local graph construction with dynamic centrality selection and feature augmentation, (ii) feature alignment using FedPCA to obtain a common feature space across clients, and (iii) federated optimization with server-side aggregation where only model updates are exchanged.

3.2.1 Architecture of Fed-GDLC

We begin with a high-level view of how components interact across clients and the server, clarifying where graph construction, centrality selection, FedPCA alignment, and the federated optimization

loop occur in practice. The end-to-end Fed-GDLC pipeline is shown in Figure 3.4. In phase one, each client builds a graph from source-destination IP pairs, computes graph measures locally, selects a non-redundant set of centralities suited to its topology, computes those node-level scores, and augments every flow record with the source and destination values. In phase two, the focus shifts to feature alignment: a federated Principal Component Analysis (PCA) is applied to obtain a shared subspace across clients. Each client forms a local covariance matrix and sends it to the server, which aggregates covariances, extracts the top-k eigenvectors, and broadcasts the common projection basis back to all clients. In this way, the necessary statistics are shared without exposing the raw data. In phase three, the system enters the federated training loop: at each round, the server shares global weights, clients train locally on private data, and only weight updates are returned. The server averages these updates to refresh the global model. Repeating this cycle enables collaborative learning under strict data-locality constraints while leveraging the structural graph context derived from each client's network.

3.2.2 Selection and Computation

The first phase of the Fed-GDLC is applying the Graph Deep Learning with Centralities (GDLC) locally at each client. The GDLC [136] is a framework designed to enhance intrusion detection by adding new features derived from the graph topology and device importance in the network. Rather than treating flows in isolation, the GDLC adds high-value structural context into each record using the centrality and hierarchical measures values of source and destination devices. These measures are carefully chosen to give the most information in the fewest number of features, balancing the additional complexity with the additional context. The selection method relies on graph-level measures to select the suitable centrality measures according to each graph. In the following paragraphs and table, all used measures are presented and defined.

In Fed-GDLC, graph information is grouped into measures spanning three categories: graph, node, and community to guide feature construction and selection across clients. Graph-level measures summarize the global topology and help decide when richer node or community-aware features are necessary. Specifically, density captures overall connectivity [152], while the mixing parameter reflects how strongly communities are separated or mixed [153]. These signals are used to guide which centralities are most informative for the current network structure, without sharing raw flows among clients.

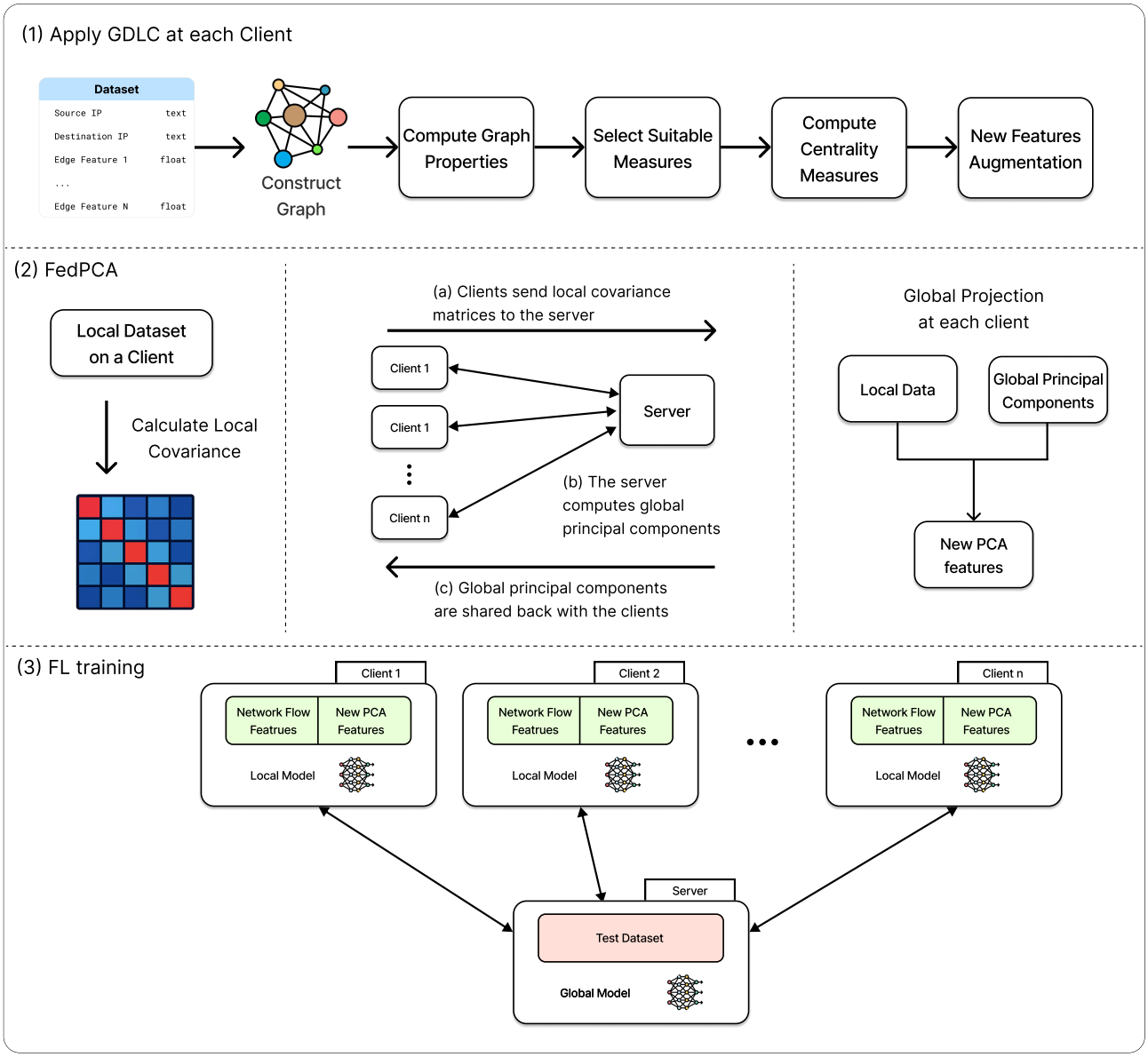


Figure 3.4: Overview of Fed-GDLC across clients and server: phase 1, clients build flow graphs, select non-redundant centralities, compute node scores, and augment records. Phase 2, clients send local covariances, the server aggregates and extracts top-k components, then returns a shared projection. Phase 3, the server shares weights, clients train locally and send updates for aggregation, preserving data locality.

At the node level, Fed-GDLC uses different centrality groups that capture local, global, and hierarchical pattern and impact. Degree centrality quantifies immediate neighborhood importance (local) [154], betweenness and closeness emphasize shortest-path inluence and reachability (global) [154, 155], and eigenvector/PageRank propagate importance through influential neighbors and random-walk (global) [154, 156]. Hierarchical measures such as the k-core and k-truss identify nodes embedded in densely connected substructures [157]. During local training, these metrics work together to provide topology-aware features that enhance traffic properties.

Community-level measures quantify how nodes act within and across detected communities, which we obtain using Infomap [111]. Local community-aware variants compute classical centralities inside each community (intra-community influence), whereas global community-aware variants operate on the inter-community graph to highlight bridges [158]. Two additional measures combine both perspectives: modularity vitality measures a node's contribution to the quality of the partitioning [120], and "Comm" centrality balances intra and inter-links to highlight hubs, bridges, or both [158]. A concise overview of all measures used in Fed-GDLC appears in Table 3.1.

From all these node-level and community-level measures, the Fed-GDLC applies a client-side topology-aware selection that first computes graph-level measures and use them to select the most informative non-redundant group of centralities accordingly, computes them, and appends the resulting scores as features to each flow record. Figure 3.5 summarizes the end-to-end workflow and its main processing stages. The framework includes five steps:

1. **Graph construction**: From raw traffic logs, form a directed, unweighted graph $G = (V, E)$ where V are unique IPs and $(a \to b) \in E$ if a flow exists from a to b; multiple flows between the same pair collapse into one edge.

2. **Graph-level characterization**: Compute Density and the Mixing Parameter to summarize overall connectivity and the strength of community structure.

3. **Dynamic measure selection**: Use Density (connectivity) and Mixing (community separation) to choose a compact, non-redundant set:

   - High density ($D > 0.01$) $\Rightarrow$ select Betweenness, PageRank, and hierarchical measures (k-core, k-truss).

Table 3.1: Overview of graph, node, hierarchical, and community-level measures employed in Fed-GDLC.

Scope	Measure	Definition
Graph-level	Density [152]	Fraction of realized edges among all possible node pairs, indicating overall connectivity.
	Mixing parameter [153]	Share of edges linking different communities; lower values imply stronger community structure.
Node (local)	Degree centrality [154]	Immediate neighborhood size normalized by network size (local hubness).
Node (global)	Betweenness [155]	Proportion of shortest paths that traverse a node, capturing bridging roles.
	Closeness [154]	Inverse of average shortest-path distance to others, reflecting reachability.
	Eigenvector [154]	Importance influenced by the importance of neighbors.
	PageRank [156]	Random-walk-based prestige with damping/teleportation.
Node (hierarchical)	k-core [157]	Maximal subgraph in which every node has degree at least k (found via recursive pruning).
	k-truss [157]	Subgraph where each edge is supported by at least $(k-2)$ triangles (iterative edge pruning).
Community-level	Local community-aware [158]	Classical centrality computed within each detected community (intra-community influence).
	Global community-aware [158]	Classical centrality on the inter-community graph (bridge roles between communities).
	Comm centrality [158]	Degree-based blend of intra- and inter-community links (inter-links squared).
	Modularity vitality [120]	Node's contribution to network modularity (positive or negative).

- Otherwise ⇒ include classical centralities (Degree, Closeness, Eigenvector, Betweenness, PageRank) plus hierarchical measures.
- Strong community structure (low mixing, $\mu < 0.01$) ⇒ add global community-aware variants and Modularity Vitality.
- Weak community structure (High mixing, $\mu > 0.01$) ⇒ add local community-aware variants and Comm centrality.

4. **Node-level computation**: For every node, compute the centralities selected in the previous step.
5. **Feature augmentation**: Append the source- and destination-node scores to each flow record to create the final feature set.

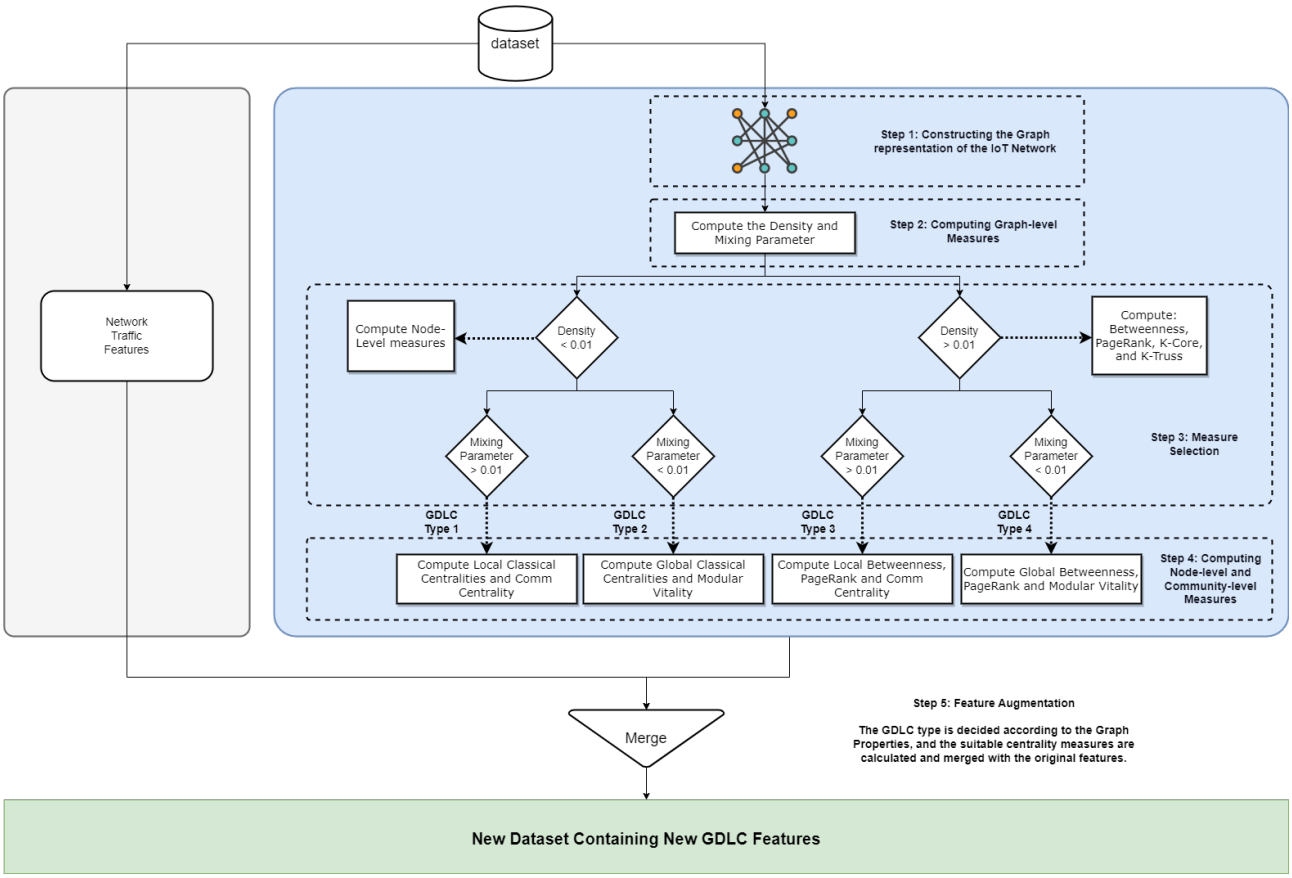


Figure 3.5: GDLC workflow overview: (1) construct a directed, unweighted graph from network flows; (2) compute graph-level measures (Density, Mixing Parameter); (3) dynamically select non-redundant node- and community-level centralities based on these properties; (4) compute the selected measures per node; and (5) augment each flow record with source/destination node features.

3.2.3 Fed-PCA for Feature Alignment

Federated Principal Component Analysis (FedPCA) adapts the classical PCA algorithm to the federated learning setting [159]. It serves a critical purpose in Fed-GDLC by unifying the different features augmented using the dynamic selection of the GDLC. Without exchanging raw data, the FedPCA enables multiple clients to jointly derive a common low-dimensional feature space.

Principal Component Analysis (PCA) is a classical technique for linear dimensionality reduction and feature extraction. PCA transforms n possibly correlated features into a new set of uncorrelated variables called principal components, ordered by the amount of variance they capture. That means the majority of the information in the older features will remain after the top components are chosen. PCA preserves the original dataset's most informative structure while reducing noise and redundancy by keeping only the top k components. Algorithm 3 describes these steps of standard PCA.

Algorithm 3 Algorithmic Steps for Standard Principal Component Analysis (PCA)

Input: Data matrix $X \in \mathbb{R}^{m \times n}$ (rows are samples, columns are features), target dimension $k \leq n$
Output: Transformed data $X' \in \mathbb{R}^{m \times k}$

1: **Standardize features:** Subtract the mean μ_j and divide by the standard deviation σ_j for each feature.
2: **Compute covariance matrix** C**:** $C \leftarrow \frac{1}{m-1} X^\top X$
3: **Eigen decomposition,** calculate eigenvalues λ_i and eigenvectors v_i: $C\, v_i = \lambda_i\, v_i, \quad i = 1, \ldots, n$
4: **Eigenvalues Sorting:** sort eigenvalues λ_i in descending order, and reorder corresponding eigenvectors v_i accordingly.
5: **Select top-**k **components,** to form the projection matrix W: $W \leftarrow [\, v_1, v_2, \ldots, v_k \,] \in \mathbb{R}^{n \times k}$
6: **Project data:** Project the original dataset X onto the new feature space: $X' \leftarrow X\, W$
7: **return** X'

When integrating the GDLC framework with federated learning (FL), the dynamic measure-selection that is applied at each client separately yields client-specific subsets of centrality features reflecting each local graph's characteristics. This heterogeneity creates a core difficulty in FL, where the global model assumes a consistent feature space across participants. Although standard PCA could compress heterogeneous centrality sets into a fixed-size representation, doing so would necessitate pooling raw data centrally, contradicting FL's decentralization and privacy tenets. To address this, we employ Federated Principal Component Analysis (FedPCA), a two-stage approach that preserves data locality while harmonizing feature representations across clients.

FedPCA is grounded in the protocol of Grammenos *et al.* [159], which proposes streaming, asynchronous, and differentially private subspace estimation. In our setting, we adopt a simplified one-shot, synchronous variant that retains the essential privacy and decentralization guarantees while reducing algorithmic complexity. The procedure consists of three main steps:

1. *Local Covariance Computation:* Each client computes the covariance matrix of its selected centrality features and transmits only these summary statistics to the server.
2. *Federated Aggregation:* The server aggregates local covariance matrices into a global covariance and performs eigen-decomposition to obtain the global principal components, which are then broadcast to clients.
3. *Global Projection:* Each client projects its local data onto the shared global principal components.

In contrast to naively concatenating client-specific feature spaces, FedPCA aggregates covariance statistics instead of exchanging raw features. For each client i, let C_i denote the covariance matrix computed from its local graph-based features, and let m_i be the corresponding sample size. The client sends only (C_i, m_i) to the server, which then constructs the global covariance as:

$$C_{\text{global}} = \frac{1}{\sum_i m_i} \sum_i m_i C_i.$$

This weighted aggregation ensures that each client influences the global covariance in proportion to the size of its dataset. The server then performs an eigen-decomposition of C_{global} to obtain the top-k principal components, which are broadcast back to all clients for local feature projection.

The resulting projected features are not additional raw attributes, but compact low-dimensional embeddings that capture the shared variance across heterogeneous local feature spaces. Operating on these aligned embeddings allows the global model to work in a common latent subspace, while all original traffic records and raw features remain local to each client. We considered non-linear PCA variants (e.g., Kernel PCA), but ultimately adopted the linear formulation to limit computational and communication overhead in the federated setting. Non-linear extensions could be incorporated in future work, for instance via privacy-preserving kernel approximation techniques.

Executing these steps maps all clients' centrality-augmented features into a shared lower-dimensional subspace, despite differences arising from local graph properties. This yields a unified feature repre-

sentation for the global FL model without exposing raw data. Full algorithmic details appear in Algorithm 4.

Algorithm 4 Algorithmic Steps for Federated Principal Component Analysis (FedPCA)

Input: For each client $i = 1, \ldots, N$, local data matrix $X_i \in \mathbb{R}^{m_i \times n}$ (standardized centrality features), target dimension $k \leq n$

Output: Each client's transformed data $X_i' \in \mathbb{R}^{m_i \times k}$

1: **Local covariance computation:**
$C_i \leftarrow \frac{1}{m_i - 1} X_i^\top X_i$

2: **Aggregation at server:** Clients send C_i; server computes:
$M \leftarrow \sum_{i=1}^{N} m_i$,
$C_{\text{global}} \leftarrow \sum_{i=1}^{N} C_i$

3: **Eigen decomposition at server:** Calculate eigenvalues λ_j and eigenvectors v_j:
$C_{\text{global}}\, v_j = \lambda_j\, v_j, \quad j = 1, \ldots, n$

4: **Eigenvalues sorting:** Sort (λ_j, v_j) pairs in descending order by λ_j.

5: **Select top-k components** to form projection matrix W:
$W \leftarrow [\, v_1, v_2, \ldots, v_k \,] \in \mathbb{R}^{n \times k}$

6: **Broadcast components:** Server sends W to all clients.

7: **Local projection:** Each client computes
$X_i' \leftarrow X_i\, W$

8: **return** Each client's X_i'

3.2.4 Deep Learning Models

In this work, we adopted two effective deep architectures for NIDS: CNN and the heavier CNN-LSTM, which have outperformed traditional classifiers on benchmark traffic datasets [160]. Augmenting inputs with GDLC's centrality and hierarchical measures provides structural context from IoT communication graphs, which improves robustness in heterogeneous environments. Each model treats the GDLC-augmented flow as a one-dimensional signal and applies 1D-CNN layers to learn local patterns and enrich feature representations. In the hybrid model, the convolutional output feeds an LSTM layer to capture longer-range dependencies, before flattening and passing through dense layers for classification, as shown in Figure 3.6. This combination enables the detection of more complex attack behaviors.

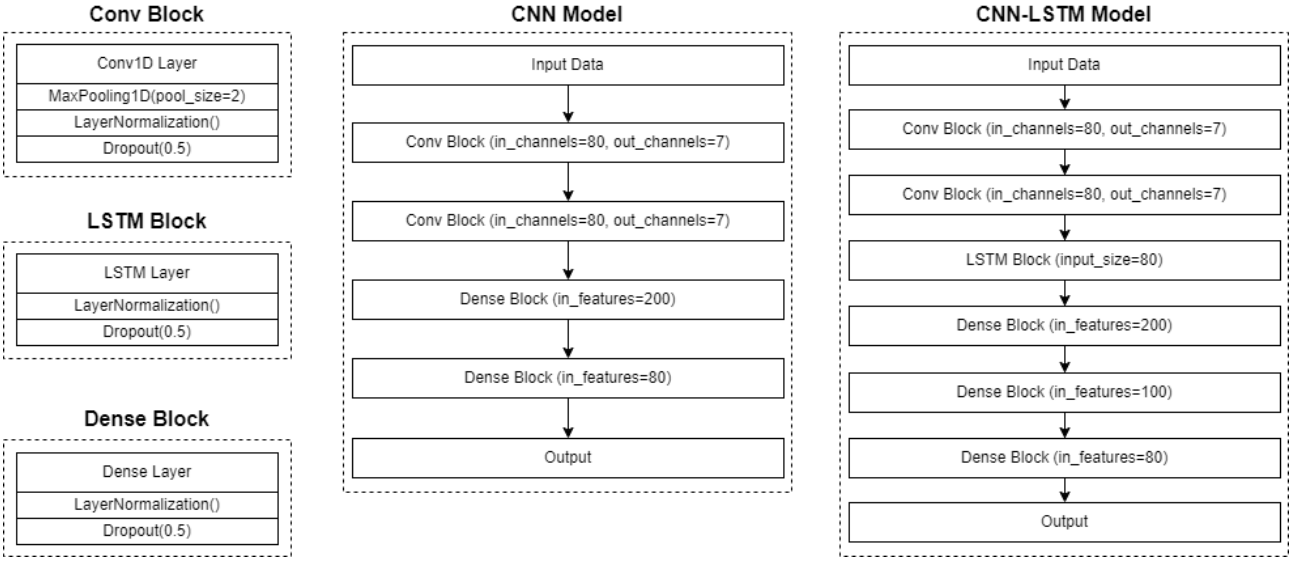


Figure 3.6: Architecture of the CNN and the CNN-LSTM model used in our experiments. Each input flow is first augmented with GDLC-derived centrality and hierarchical features, then processed by two 1D convolutional blocks, an LSTM block, followed by multiple dense layers for classification. Outputs use sigmoid with binary cross-entropy for binary tasks, or softmax with categorical cross-entropy for multi-class tasks.

3.2.5 Experimental Setup

The experimental setup used to evaluate the Fed-GDLC framework is represented and discussed in this subsection. The general settings for the experiments were configured to mimic real-world settings using five different datasets: CIC-Bot-IoT, CIC-ToN-IoT, CIC-IDS2017, CIC-DDoS-2019, and CIC-UNSW-NB15. First, the environment is presented, then each dataset is described, and finally the federated learning setup is displayed.

Our experiments used standard Python tooling: deep models in PyTorch, a simulated federated setting with Flower, and graph construction in addition to centrality computation using NetworkX. In each federated round, every client trained its local model for one epoch with a learning rate of 5×10^{-5}, then evaluation was performed on a held-out test set. To assess Fed-GDLC, we report the weighted F1-score, which accounts for class-frequency imbalance across labels. During training, we employed Focal Loss to mitigate skewed class distributions by down-weighting easy examples and emphasizing hard ones, using $\alpha = 1.0$, $\gamma = 2.0$, and a "sum" reduction to preserve the contribution of minority-class errors in the aggregated loss.

In a federated learning setup, clients must share a common feature space to enable a single global model [148]. For that reason, we have chosen datasets that are standardized to a consistent format despite having different original distributions. We instantiate clients using 1M-record subsets from CIC-Bot-IoT, CIC-ToN-IoT, CIC-IDS2017, CIC-DDoS-2019, and CIC-UNSW-NB15. Table 3.2 summarizes each dataset's attack types, the percentage of attack records, a brief description, and publication year.

Table 3.2: Federated evaluation datasets and properties. We use 1M-flow subsets from CIC-Bot-IoT, CIC-ToN-IoT, CIC-IDS2017, CIC-DDoS-2019, and CIC-UNSW-NB15 to instantiate clients with a shared feature format. The table reports attack types, percentage of attack records (class imbalance), a brief description, and the publication year.

Dataset	***Types of Attacks***	***Percentage of attacks***	***Description***	***Year***
CIC-Bot-IoT [161]	DDoS, DoS, Scanning, and Data infiltration	99.34%	Generated from a realistic IoT test built in the Cyber Range Lab at UNSW Canberra, containing a combination of normal and botnet traffic.	2019
CIC-ToN-IoT [15]	DoS, DDoS, MITM, Scanning, Injection, Password, Backdoor, Ransomware, XSS	53.24%	CICFlowMeter representation of the ToN-IoT dataset focuses on evaluating cybersecurity applications in the Internet of Things (IoT).	2020
CIC-IDS-2017 [16]	DoS, DDoS, Bot, PortScan, Infiltration, Web Attacks, Patator, Heartbleed	21.03%	A modern, realistic network simulation for testing intrusion detection and prevention systems using benign traffic.	2017
CIC-DDoS-2019 [162]	10 kinds of Reflection-based DDoS, and 3 kinds of Exploitation-based attacks	99.84%	A collection of network-flow records focusing on modern DDoS attacks, designed to address gaps in older datasets.	2019
CIC-UNSW-NB15 [32]	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms	2.88%	A collection of network traffic consisting of benign activities and nine categories of synthetic attacks.	2015

Applying Fed-GDLC's measures selection algorithm at each client is driven by graph-level prop-

erties of the local network. Table 3.3 reports, for each client, the node and edge counts, average degree (graph size), density (connectivity), mixing parameter (community structure), and the resulting GDLC type chosen by our dynamic algorithm. Density and mixing parameter jointly determine which centrality group should be computed, as outlined in Figure 3.5. For example, CIC-Bot-IoT and CIC-IDS-2017 have a moderate density with low mixing and are assigned GDLC type "Two," the sparse CIC-ToN-IoT graph is assigned type "One," and the higher-density CIC-UNSW-NB15 corresponds to type "Four."

Table 3.3: Graph-level properties per client (nodes, edges, average degree, density, mixing parameter) and the resulting GDLC type chosen by the Fed-GDLC selection policy. Density and mixing guide which centrality families are computed, as outlined in Figure 3.5.

Dataset	Nodes	Edges	Avg. Degree	Density	Mixing Parameter	GDLC Type
CIC-Bot-IoT	122	163	2.672	0.013	0.007	Two
CIC-ToN-IoT	64,690	88,613	2.739	2.218e-05	0.194	One
CIC-IDS-2017	17,503	80,790	9.232	2.6e-04	0.0	Two
CIC-DDoS-2019	425	669	3.148	0.004	0.0	Two
CIC-UNSW-NB15	42	289	13.761	0.167	0.0	Four

The federated learning setup for Fed-GDLC is shown in Figure 3.7. Each client represents a distinct dataset (CIC-Bot-IoT, CIC-ToN-IoT, CIC-IDS2017, CIC-DDoS-2019, CIC-UNSW-NB15), mirroring real-world heterogeneity in networks and traffic patterns. First, every client independently applies the GDLC pipeline to augment its local features. The server then runs the FedPCA step of Fed-GDLC to align feature spaces produced by the dynamic selection algorithm, without sharing raw data. With a unified subspace, training starts using standard federated rounds: clients update their models locally and send the model weight updates only to the server, which aggregates these updates using FedProx to form a new global model and sends it back to all clients. This iterative process continues until convergence, yielding a privacy-preserving, context-aware intrusion detector that uses structural properties across diverse IoT environments.

3.2.6 Results

Here we present the experimental evaluation of the proposed Fed-GDLC framework. We begin by assessing the FedPCA step: reconstruction RMSE of Local PCA is compared with that of FedPCA

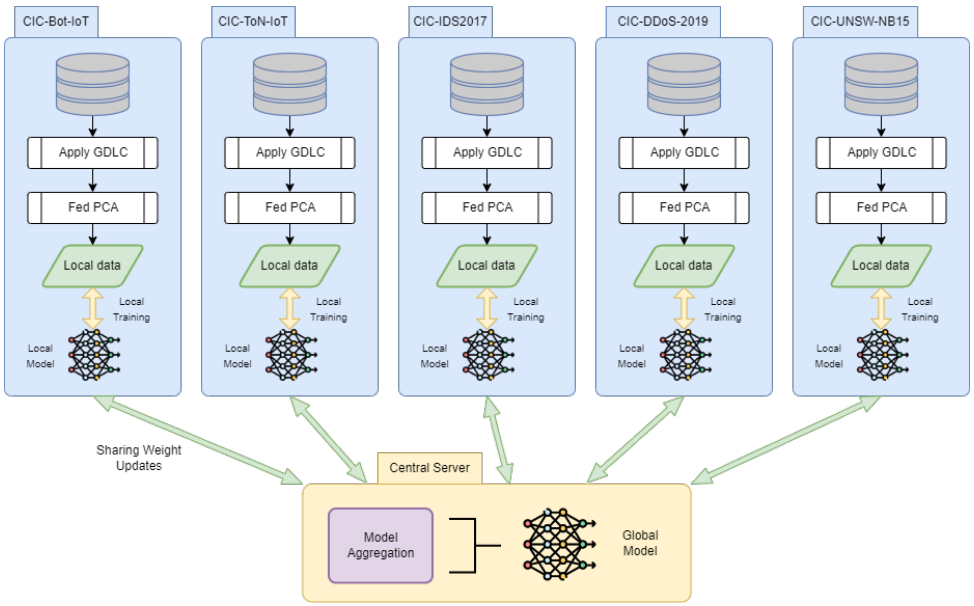


Figure 3.7: Federated learning architecture for Fed-GDLC. Each client corresponds to a distinct dataset (CIC-Bot-IoT, CIC-ToN-IoT, CIC-IDS2017, CIC-DDoS-2019, CIC-UNSW-NB15) and applies the GDLC feature-engineering pipeline locally. The server then runs FedPCA to align feature spaces without sharing raw data, followed by iterative federated rounds: clients train on private data and send weight updates, and the server aggregates them (FedProx) and broadcast the global model.

across clients, and we analyze how the number of FedPCA components affects intrusion-detection performance. Next, we compare four federated approaches: Baseline (no graph information), three randomly selected centralities (shared across clients), All Centralities, and Fed-GDLC. The comparison is done by testing the global model after each round of training on a unified server-side test set merged from all datasets. Finally, we provide a time-cost analysis to quantify the computational overhead of feature augmentation. These results show that CNN-LSTM within Fed-GDLC offers a robust, dynamic solution for intrusion detection in heterogeneous environments.

3.2.6.1 Evaluating FedPCA Dimensionality: Effects on Data Reconstruction and Intrusion Detection

In this first experiment, we examine how the dimensionality of the shared FedPCA subspace impacts both feature reconstruction and end-to-end detection performance. First, we compare reconstruction RMSE for FedPCA against Local PCA (taken as a benchmark) across varying numbers of components to quantify information loss introduced by unifying client data. Then, using CNN-LSTM within Fed-GDLC and sweeping 3-10 FedPCA components, we report the global F1-score to identify

the dimensionality that maximizes detection accuracy while keeping communication and computational overhead reasonable.

Principal Component Analysis (PCA) projects the original feature space onto a lower-dimensional subspace by preserving components with the largest eigenvalues of the covariance matrix only. Removing the smaller components result in some information loss. FedPCA adopts the same principle but derives a shared subspace from aggregated per-client covariance estimates, which may not align exactly with each client's local principal directions and thus introduces a small additional reconstruction error. To assess the FedPCA step in Fed-GDLC, we measure reconstruction RMSE on each client for both Local PCA (computed separately per client) and FedPCA (shared subspace). Figure 3.8 reports this comparison across the five datasets, averaging RMSE over component counts from 1 to 10. As expected, reconstruction error decreases with more components, while FedPCA exhibits slightly higher RMSE than Local PCA because it must encode across heterogeneous datasets. This reasonable increase is the trade-off for obtaining a common, privacy-preserving subspace across clients.

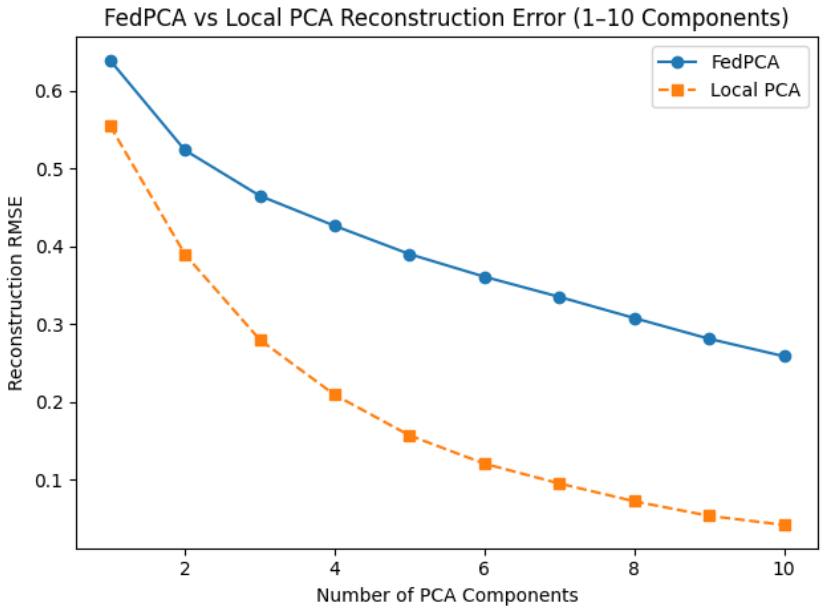


Figure 3.8: Reconstruction RMSE of Local PCA and FedPCA as a function of the number of principal components. FedPCA yields slightly higher RMSE, reflecting a shared subspace across heterogeneous clients, while increasing components reduces error by capturing more variance.

As shown in Figure 3.9, one or two FedPCA components are insufficient to capture the information carried by the centrality features. Between three and ten components, no sharp "elbow" is visible, as

additional dimensions yield only marginal RMSE reductions. To select a practical dimensionality for Fed-GDLC, we evaluated the global model with three to ten components using identical architectures and hyperparameters. The resulting test F1-scores are plotted in Figure 3.9. Performance peaks at six components, indicating that most critical variance is captured within this subspace. Beyond six, gains diminish and signs of overfitting appear (notably at ten components). Higher dimensional settings retain more client-specific detail, improving local fit but hurting generalization on the unified server-side test set. Accordingly, we adopt six components as a balanced choice between information preservation and robustness.

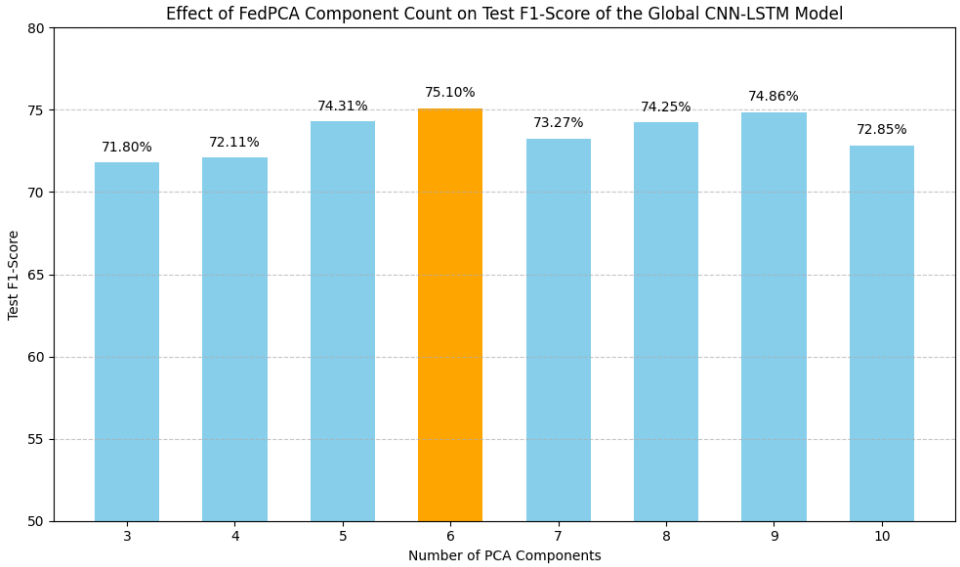


Figure 3.9: Effect of the FedPCA dimensionality on test F1-score for the global model. Performance peaks at six components, indicating a balance between retained variance and overfitting risk; beyond six, gains diminish or degrade.

3.2.6.2 Comparative Evaluation of Fed-GDLC and Other Federated Learning Approaches

In this section, we compare Fed-GDLC to alternative federated learning approaches using two metrics: the server-side test F1-score and the average training loss across clients. A single CNN-LSTM configuration with identical hyperparameters is used to ensure a fair comparison. Fed-GDLC employs a dynamic selection of the most suitable node-level and community-level measures to encode graph structure and device-relationship patterns in IoT networks. These features are then aligned across clients before augmenting the feature space. To quantify the benefit of this integration, we compare

Fed-GDLC with three baselines: (i) a Baseline FL setup with no additional measures, (ii) Three Selected Centralities: Degree, Betweenness, and PageRank chosen without Fed-GDLC and applied uniformly across clients (no alignment required), and (iii) All Centralities: adding every centrality considered in this study with no selection policy. We report and discuss two metrics for all approaches: the global model's F1-score on the unified test set and the mean training loss across clients. While training loss can be a weak proxy in centralized settings, under FL it reveals how well models fit heterogeneous local data, enabling comparison with global test performance to diagnose overfitting. Figure 3.11 plots the average per-round training loss across clients, and Figure 3.10 shows the server-side F1-score over rounds.

The evaluation results identify a clear winner: Fed-GDLC with six FedPCA components attains the highest test F1-score of approximately 75.1%, exceeding the three randomly selected centrality measures approach at 71.3% and the Baseline at 68.2%. Unexpectedly, the "All Centralities" variant achieves only 61.4%. In the plots, federated rounds begin at zero, where models are evaluated with randomly initialized weights. Consequently, all curves start low at round zero, offering a snapshot of how detection capability improves over time. Regarding training loss, the Baseline shows the slowest decline and the highest overall loss. Adding any centrality information lowers local training loss: using three randomly selected measures reduces it substantially relative to the Baseline, with Fed-GDLC slight edge over it. Notably, "All Centralities" reaches the lowest training loss of all, yet this does not carry over to the test set, where it resulted with the poorest F1-score. This indicates overfitting to client-specific distributions. By contrast, Fed-GDLC strikes an effective balance, mitigating both underfitting and overfitting by selecting only the most informative local features and aligning them using a low-dimensional shared subspace with FedPCA. Despite reasonable added complexity, Fed-GDLC offers the best trade-off among convergence speed, robustness, and final predictive performance.

3.2.6.3 Evaluating Fed-GDLC Models Compared to Centralized Training

To assess how different model architectures behave under the proposed Fed-GDLC approach, we plot the server-side F1-score for three deep learning models (CNN-LSTM, CNN, MLP). In this experiment, all models are trained with Fed-GDLC for 20 federated rounds, using six FedPCA components, the setting that yields the best performance, as shown in Section 3.2.6.1. For context, we also evaluate a centralized CNN-LSTM to represent the best attainable outcome without FL constraints. In the

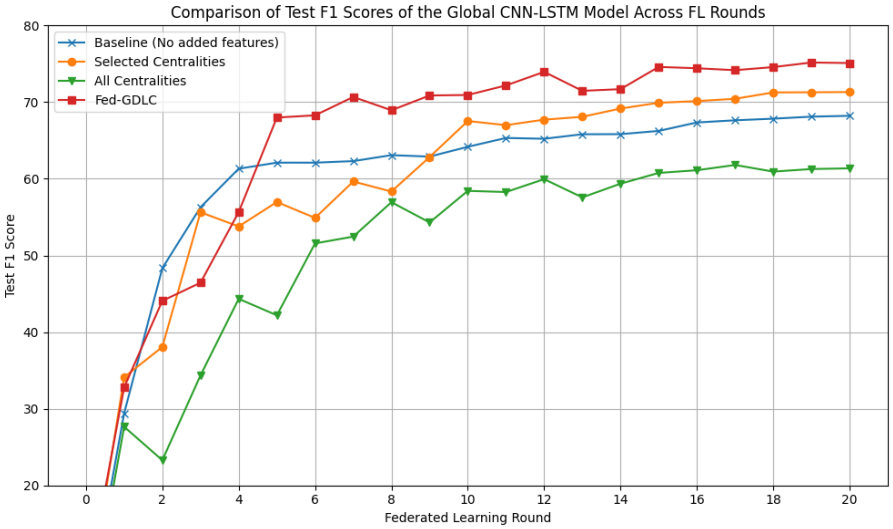


Figure 3.10: Server-side F1-score of the global CNN-LSTM across federated rounds (0-20) for four approaches: Baseline, Selected Centralities, All Centralities, and Fed-GDLC (six FedPCA components). Fed-GDLC attains the highest final F1-score, indicating the best generalization.

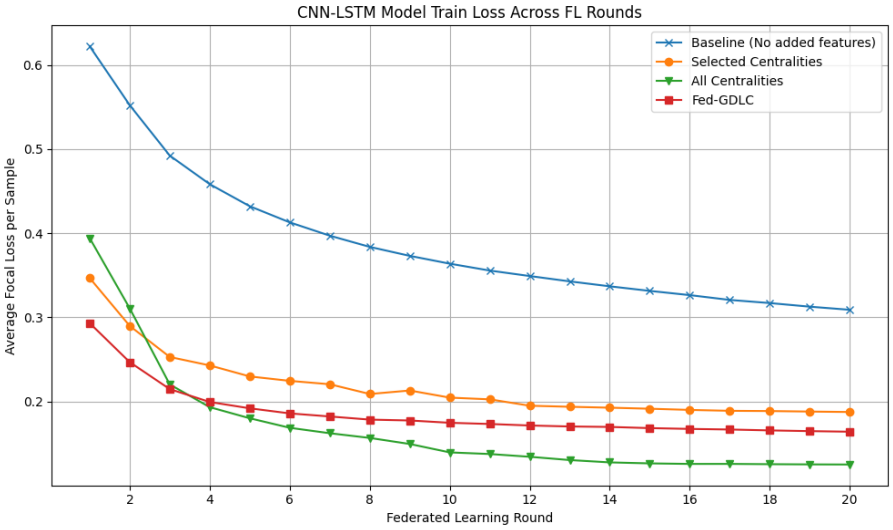


Figure 3.11: Average client-side training loss of the CNN-LSTM over federated rounds (0-20) for the same four approaches. Fed-GDLC shows efficient convergence; the All Centralities variant reaches the lowest training loss yet exhibits signs of overfitting relative to test performance.

centralized setup, all local datasets are merged into a single training set, and after 20 epochs, the model is evaluated on the same test set used for the global FL model. Results appear in Figure 3.12, where the FL curves are plotted over rounds and a horizontal dashed line marks the final F1-score of the centralized model. As expected, no FL-trained model exceeds the centralized CNN-LSTM, which achieves 78.27%. Under Fed-GDLC, all three models improve steadily from round 0: CNN-LSTM attains the highest scores, surpassing 70% by round 10 and peaking at 75.10% by round 20, the CNN follows closely, reaching 70.16% at round 20, and the MLP trails, with a maximum of 63.36% at round 20.

Under federated learning constraints, the CNN-LSTM emerges as the most effective architecture within the Fed-GDLC framework. Its highest F1-score at round 20 (only slightly below the centralized counterpart) demonstrates strong capacity to capture complex patterns across heterogeneous datasets. While smaller, lighter models do improve with collaborative training, they fall short of delivering consistently reliable attack detection. The reasonable 3% gap between CNN-LSTM under Fed-GDLC and the centralized benchmark underscores its advantage in balancing the privacy-accuracy trade-off. Closing the remaining gap will likely require more expressive architectures, whereas simpler models have a larger penalty in privacy-preserving federated settings.

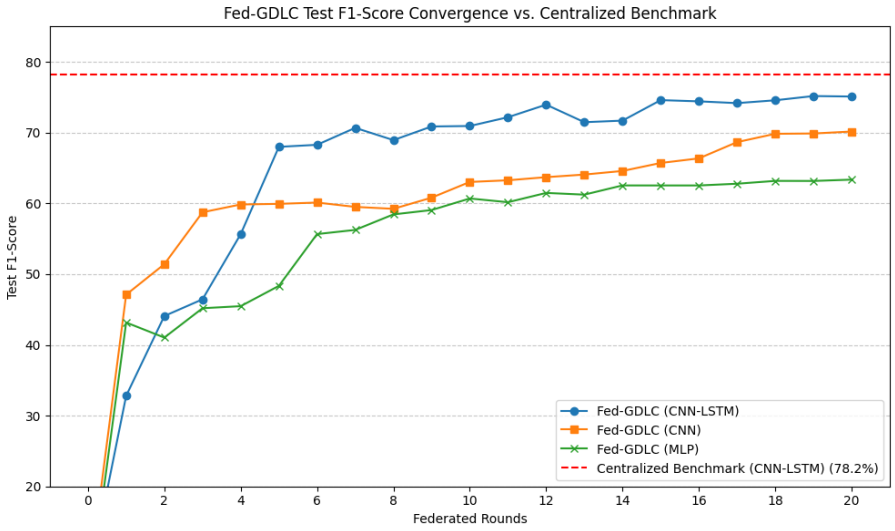


Figure 3.12: Server-side F1-score over 20 federated rounds for CNN-LSTM, CNN, and MLP trained with Fed-GDLC (six FedPCA components). The centralized CNN-LSTM benchmark performance is denoted by a red dashed line at 78.27%.

3.2.6.4 Runtime Analysis

In this subsection, the computational overhead of both the feature-augmentation stage and the federated learning process is studied in order to evaluate the Fed-GDLC's feasibility. Figure 3.13 decomposes runtime for the four approaches into two phases: Feature Computation (centralities or PCA) and training over 20 federated learning rounds. Execution time (in seconds) is visualized as horizontal stacked bars. As expected, adding more features lengthens training, most notably in the "All Centralities" variant where no selection policy is applied. This comparison highlights that the cost of computing centralities is relatively small compared to the expense of 20 FL rounds. Overall, Fed-GDLC introduces some additional overhead, but it remains minor relative to the accuracy gains obtained by leveraging the new topology-aware features.

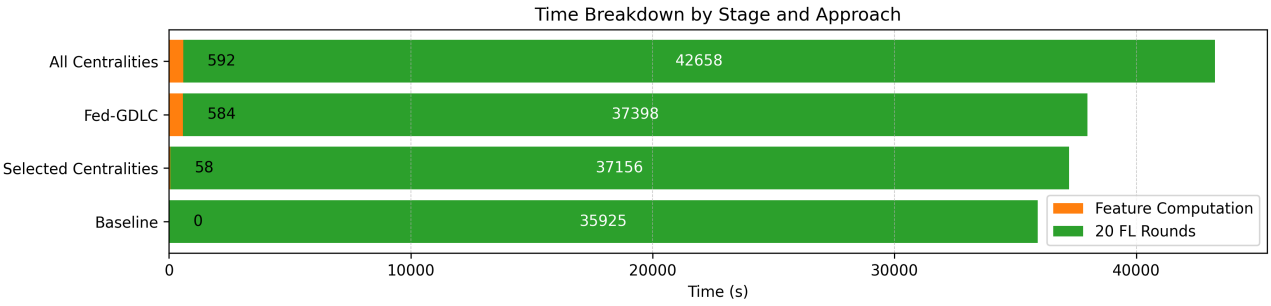


Figure 3.13: Execution-time breakdown by approach, separating feature computation (centralities or PCA) from 20 federated training rounds. Results for Baseline, Selected Centralities, Fed-GDLC, and All Centralities are shown as stacked horizontal bars. Feature extraction adds modest overhead compared to the dominant cost of FL training.

3.3 Conclusion

This chapter established the case for federated intrusion detection in decentralized IoT and smart-city settings and validated two key ideas. First, locally computed centrality measures on client graphs have kept their discriminative value under FL. Notably, they improved round-wise F1 and stabilized training across heterogeneous, non-IID clients without sharing raw traffic. Second, Fed-GDLC extends this benefit by (i) dynamically selecting non-redundant centralities guided by graph-level measures (density and mixing parameter) and (ii) aligning client-specific features using FedPCA. In our experiments, a CNN-LSTM trained with Fed-GDLC achieved the strongest global F1-score, outperforming a traffic-only baseline, a three centrality case, and naïve "All Centralities". Specifically, the latter exhib-

ited overfitting, where results showed low training loss but poorer generalization. We further showed that six FedPCA components have a favorable balance between information loss and robustness, and that the time cost of computing centralities is modest compared to the dominant cost of federated rounds.

Taken together, these results indicate that privacy-preserving, graph-informed feature engineering strengthens FL-based NIDS. They also clarify practical design trade-offs: dynamic selection beats a uniform pre-defined set of features, low-dimensional alignment works under heterogeneity, and deep hybrid models (CNN–LSTM) are effective at exploiting enriched features. Limitations remain around robust aggregation against poisoned updates, communication-efficient training, and secure aggregation with differential privacy.

This chapter demonstrated that centrality vectors are (i) useful in decentralized settings and (ii) alignable without sharing raw traffic. Yet, hand-engineered features underutilize relational patterns and are limited in large graphs. Chapter 4 shifts from "features from graphs" to "learning on graphs" using the message-passing of Graph Neural Networks. Concretely, we (a) initialize node states with centralities (and a ones-vector) as inductive priors, (b) design edge-aware aggregators and residual edge pathways to preserve individual edge properties, and (c) propose ADAP-GNN, a rule-based framework to select the suitable GNN architecture according to graph properties. This moves the burden of representation from manual feature design to data-driven graph learning, and prepare for future works on federated GNNs in NIDS.

Chapter 4

ADAP-GNN: Adaptive Property-Aware Graph Neural Network Framework

Contenu

4.1 Adapting Graph Neural Networks for Flow-Based Intrusion Detection 139
4.1.1 Graph Neural Networks 139
4.1.2 Problem 141
4.1.3 Edge-Variant: E-GraphSAGE 141
4.1.4 Extension: E-ResGraphSAGE (Residual Connections) 143
4.2 Centrality Informed E-GraphSAGE 143
4.3 ADAP-GNN 146
4.3.1 Adaptive selection algorithm 146
4.3.2 Experimental Setup 154
4.3.3 Results 156
4.4 Conclusion 164

Graph-structured data appears in a wide range of applications, ranging from social interactions to communication networks and biological systems. Traditional deep learning models are designed for regular Euclidean domains such as images or sequences, while Graph Neural Networks (GNNs) extend these models to non-Euclidean domains by exploring the connectivity information found in a graph [163]. GNNs operate by iteratively aggregating and transforming information from a node's neighborhood so that each node learns an embedding that reflects both its own features and the features of its neighbors. This message-passing paradigm enables the extraction of local and global structural patterns that are inaccessible to models that ignore relationships among data points [164].

In the context of cybersecurity and Network Intrusion Detection Systems (NIDS), graphs provide

a natural representation for flows between devices in a network. Each device can be treated as a node and each flow or communication between two devices is modeled as an edge connecting the corresponding nodes [12]. GNNs are thus well suited for learning from such structured data because they allow a model to exploit the topological information present in the network. The objective of this chapter is to provide the necessary background on graph neural networks, describe how they can be applied to NIDS, and present the contributions of this thesis toward improving edge-centric GNNs for intrusion detection.

4.1 Adapting Graph Neural Networks for Flow-Based Intrusion Detection

The increasing scale and complexity of Internet of Things (IoT) networks have made them attractive targets for adversarial attacks. Traditional signature-based NIDS identify attacks by comparing network traffic to a database of known patterns, but they cannot detect previously unseen (zero-day) attacks. Machine-learning and deep-learning approaches learn statistical patterns from data and have achieved promising results in detecting novel threats [165, 166]. However, most existing models process each network flow in isolation and fail to exploit the relational structure of the network. For example, a port-scan attack may appear benign when analyzing a single flow but is revealed by the pattern of many short connections originating from the same attacker [126].

Graph-based methods address this limitation by modeling the network as a graph and employing complex network analysis or GNNs to learn from both node and edge relationships [167]. Each device is represented as a node, and each flow between two devices corresponds to an edge. The feature vector associated with an edge consists of the attributes available in the flow record, such as packet counts, durations, and protocol information. In most flow datasets there are no predefined node features, so node embeddings must be learned by aggregating edge features. Centrality measures derived from the graph structure can also provide informative node attributes that encode the importance of each device in the network [113]. A hand-engineered graph-based NIDS would use complex network measures to identify anomalies. For example, Degree and Betweenness centralities were incorporated into machine-learning models to improve classification accuracy [135]. Data-driven approaches leverage GNNs to process graph-structured data directly [85]. However, a key challenge arises when GNN are adapted to NIDS, since GNNs rely on node features, and the flow data available in NIDS datasets don't provide these features. In this section, we explain how GNN works, formulate the problem, and explain how it was tackled in the literature.

4.1.1 Graph Neural Networks

Graph neural networks are designed to operate directly on graphs. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each node $v \in \mathcal{V}$ may possess an associated feature vector $\mathbf{h}_v^0$, and each edge $uv \in \mathcal{E}$ may carry its own attribute vector $\mathbf{e}_{uv}$. GNNs generalize the idea of filters in convolutional neural networks to arbitrary graphs by propagating information along the

edges, considering that an image is just a special case of graph where pixels are nodes connected in a grid.

The basic building block of many GNNs is the message-passing layer. In a layer k, each node v collects messages from its neighbors $N(v)$ and aggregates them into a single summary vector $\mathbf{h}_{N(v)}^{k}$. This aggregated message is combined with the node's previous representation $\mathbf{h}_{v}^{k-1}$ through a learnable function to produce the updated representation $\mathbf{h}_{v}^{k}$. Formally, a typical message-passing iteration can be written as

$$\begin{aligned} \mathbf{h}_{\mathcal{N}(v)}^{(k)} &= \mathrm{AGG}^{(k)}\Big(\{\mathbf{h}_{u}^{(k-1)} : u \in \mathcal{N}(v)\}\Big), \\ \mathbf{h}_{v}^{(k)} &= \sigma\Big(\mathbf{W}^{(k)}\big[\mathbf{h}_{v}^{(k-1)} \,\|\, \mathbf{h}_{\mathcal{N}(v)}^{(k)}\big]\Big). \end{aligned} \tag{4.1}$$

where AGG^{k} is an aggregation function such as mean, sum, or max, σ denotes a nonlinear activation, $[\cdot\|\cdot]$ denotes vector concatenation, and $\mathbf{W}^{k}$ is a trainable weight matrix. After K layers, each node v obtains an embedding $\mathbf{z}_{v} = \mathbf{h}_{v}^{K}$ that depends on its features and the features of its K-hop neighborhood.

Several architectures have been proposed under this general paradigm [168]. Graph Convolutional Networks (GCNs) perform a spectral convolution on the graph Laplacian and have been widely applied to semi-supervised node classification [169]. GraphSAGE introduces inductive node embeddings by learning aggregator functions on sampled neighborhoods [18]. The sampling mechanism allows GraphSAGE to scale to large graphs and the learning of aggregator functions allows it to generalize to unseen nodes in different graphs [170]. Graph Attention Networks (GAT) extend the aggregation step with an attention mechanism that learns to weigh the contributions of different neighbors [19]. For node v, an attention coefficient $\alpha_{v,u}$ is computed for each neighbor u and normalized so that $\sum_{u \in N(v)} \alpha_{v,u} = 1$. The node's representation is then updated using a weighted sum of the transformed neighbor representations:

$$\begin{aligned} e_{v,u}^{(k)} &= \mathrm{LeakyReLU}\Big(\mathbf{a}^{\top}\big[\mathbf{W}^{(k)}\mathbf{h}_{v}^{(k-1)} \,\|\, \mathbf{W}^{(k)}\mathbf{h}_{u}^{(k-1)}\big]\Big), \\ \alpha_{v,u}^{(k)} &= \mathrm{softmax}_{u \in \mathcal{N}(v)}\big(e_{v,u}^{(k)}\big) = \frac{\exp\big(e_{v,u}^{(k)}\big)}{\sum_{w \in \mathcal{N}(v)} \exp\big(e_{v,w}^{(k)}\big)}, \\ \mathbf{h}_{v}^{(k)} &= \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{v,u}^{(k)}\, \mathbf{W}^{(k)}\mathbf{h}_{u}^{(k-1)}\right). \end{aligned} \tag{4.2}$$

Where $e_{v,u}^{(k)} \in \mathbb{R}$ is the unnormalized attention score between nodes v and u at layer k, $\alpha_{v,u}^{(k)}$ is the corresponding normalized attention coefficient obtained by applying a softmax over the neighbors of v, and $\mathbf{a} \in \mathbb{R}^{2d_k}$ is a learnable attention vector (with d_k the feature dimension at layer k).

4.1.2 Problem

Adapting graph neural networks (GNNs) to network intrusion detection systems (NIDS) presents a fundamental challenge: the problem is inherently edge-centric, yet most GNN architectures are designed for node classification and rely on node features. In flow datasets, each record corresponds to a communication between a source and a destination IP and is naturally modeled as a directed edge. These edges carry the main attributes of the traffic, such as byte counts, packet sizes, durations, and protocol flags. In contrast, the nodes, representing IP addresses or devices, do not come with built-in features. Assigning trivial or constant vectors to nodes provides no meaningful information and fails to capture the functional role of devices in the network.

This mismatch highlights the core difficulty: without node features, traditional message-passing schemes are unable to initialize or propagate informative representations. As a result, adapting GNNs to NIDS requires either architectural changes that shift the focus of learning from nodes to edges, or the design of methods to derive meaningful node attributes from the available edge information. Moreover, NIDS graphs often contain multiple parallel edges between the same pair of nodes, since one device can generate many flows toward another in a short period of time. These parallel flows may belong to different classes, benign or malicious, which makes it essential to preserve the individuality of each edge rather than collapsing them into a single connection.

Therefore, the principal problem in applying GNNs to NIDS has two main aspects: (i) the lack of predefined node features, which forces the development of strategies to extract or infer them from edge data, and (ii) the edge-centric nature of the task, which requires adaptations to the GNN architecture so that flow-level attributes are retained and used effectively during training. Addressing these issues is the base for building effective GNN-based intrusion detection system.

4.1.3 Edge-Variant: E-GraphSAGE

To address the limitations identified in adapting GNNs to NIDS, researchers proposed an edge-centric adaptation of the GraphSAGE framework, known as E-GraphSAGE. Instead of assuming the

availability of informative node features, this model learns node representations directly from the edges, which carry the relevant flow information. Each node is initialized with a simple placeholder vector made up of ones, and through successive layers, its embedding is updated by aggregating the feature vectors of the edges connected to it.

In E-GraphSAGE, at layer k, node v aggregates information from all its neighbors $u \in \mathcal{N}(v)$ through their connecting edges e_{uv}, producing an intermediate representation of its neighborhood:

$$\mathbf{h}_{\mathcal{N}(v)}^{(k)} = \mathrm{AGG}^{(k)}\Big(\Big\{ \mathbf{e}_{uv}^{(k-1)} : u \in \mathcal{N}(v) \Big\}\Big). \tag{4.3}$$

This aggregated vector is then concatenated with the node's previous embedding and passed through a trainable transformation followed by a non-linear activation. After several iterations, each node acquires an embedding that reflects both its position in the graph and the attributes of its associated flows.

In the original formulation, the edge embeddings were created only from the learned representations of the source and destination nodes. This design meant that multiple flows between the same pair of nodes could be collapsed into a single representation, losing important flow-specific information. Later works addressed this limitation by introducing residual connections, where the raw edge features are reintroduced alongside the node embeddings during classification. This modification allows each flow to keep its individuality and prevents the loss of critical details in scenarios with parallel edges.

The original E-GraphSAGE implementation also considered all neighbors of each node during aggregation, which can be inefficient and prone to noise in large graphs. To improve scalability, later adaptations introduced neighbor sampling, where only a subset of edges are aggregated at each step. This reduces computation while maintaining sufficient information for accurate learning. Further extensions incorporated attention mechanisms to weigh neighbors differently, allowing the model to emphasize the most relevant connections. Together, these modifications strengthen the ability of E-GraphSAGE to operate effectively in the edge-centric setting of NIDS, where flow-level details and scalability are equally important.

4.1.4 Extension: E-ResGraphSAGE (Residual Connections)

While E-GraphSAGE successfully adapts GraphSAGE to the edge-centric nature of NIDS, it suffers from an important limitation: the final edge embedding is constructed only from the embeddings of the source and destination nodes. This design implicitly assumes that all flows between the same pair of nodes share the same class. In practice, however, multiple parallel flows may exist between two devices, and these flows can belong to different categories, such as benign traffic mixed with malicious scans. Collapsing them into a single representation will results in information loss and misclassification. To address this limitation, E-ResGraphSAGE was introduced by incorporating residual connections into the architecture. In this model, the original edge features are preserved and injected directly into the final classification stage, alongside the node embeddings learned from the message-passing process. The edge embedding at layer K is obtained as

$$\mathbf{z}_{uv} = \mathrm{AGG}_{edge}(\mathbf{z}_u, \mathbf{z}_v, \mathbf{e}_{uv}) \,, \tag{4.4}$$

where $\mathbf{z}_u$ and $\mathbf{z}_v$ denote the final embeddings of the source and destination nodes after K layers, and $\mathbf{e}_{uv}$ represents the original flow feature vector. The aggregation function AGG_{edge} combines these three components into an enhanced edge representation that retains both the structural context and the raw flow-level attributes.

By explicitly reintroducing $\mathbf{e}_{uv}$, E-ResGraphSAGE ensures that each flow remains distinguishable even when multiple connections exist between the same pair of nodes. This modification improves the model's ability to classify parallel flows accurately, especially in scenarios where subtle variations in protocol flags, packet sizes, or timing distinguish malicious behavior from benign activity.

4.2 Centrality Informed E-GraphSAGE

While E-GraphSAGE and its residual extension address some limitations of applying GNNs to NIDS, they still rely on an uniform node initialization (vectors of ones). This way, all devices in the network are treated equally and their structural roles are ignored. To overcome this limitation, we proposed in our earlier work a centrality-informed initialization strategy, providing the model with meaningful node features derived from graph topology [171]. Centrality measures capture both local

connectivity and global influence of nodes, making them a good candidate to embed the structural context into the learning process of GNN. In this work, we integrated five measures: Degree, Betweenness, Closeness, PageRank, and K-truss, where we normalized their values and concatenated them to form the initial feature vector of each node. The overall architecture is summarized in Figure 4.1, where centrality-based initialization is incorporated into the E-GraphSAGE pipeline.

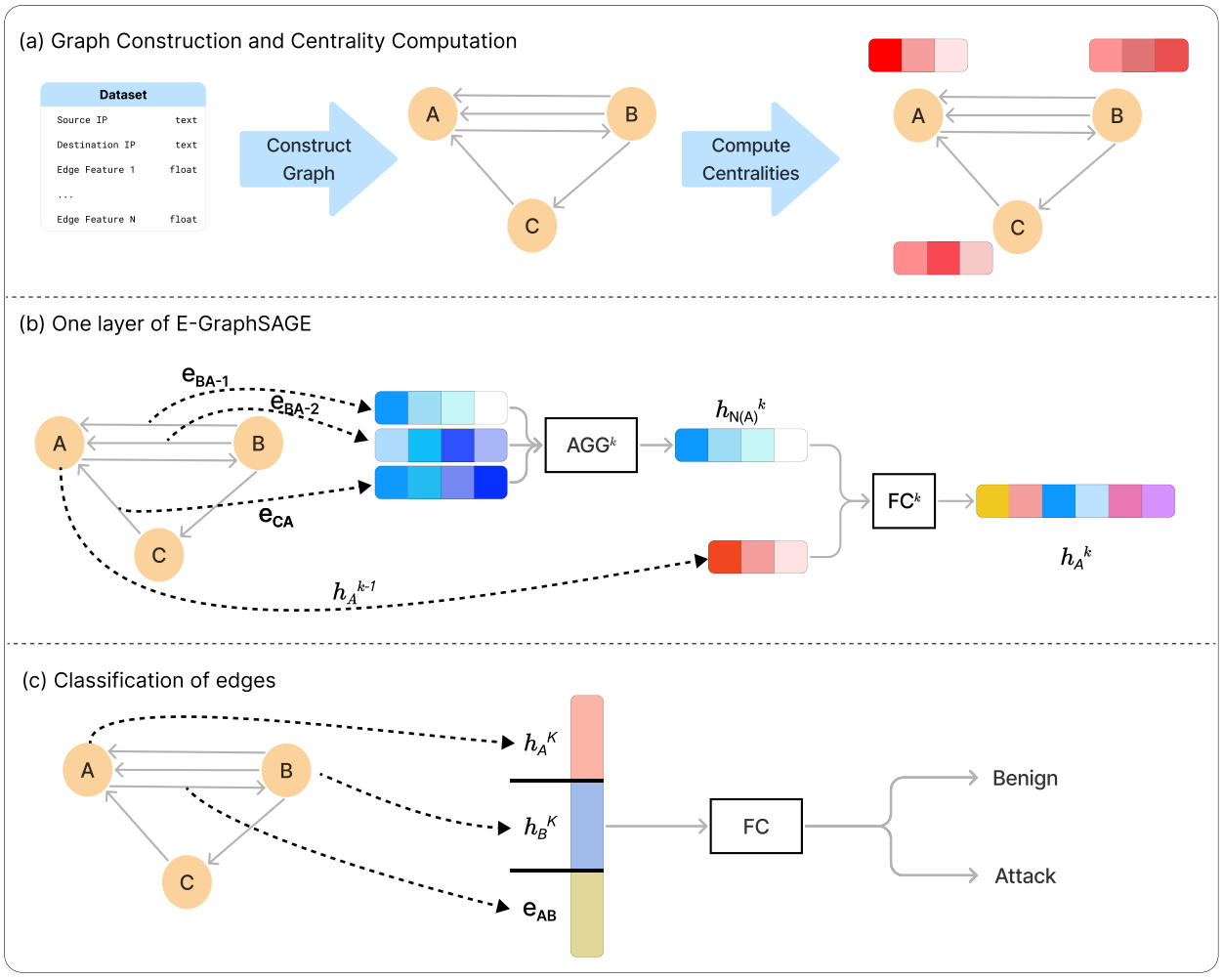


Figure 4.1: Overview of the centrality-informed E-GraphSAGE: (a) graph construction and centrality computation, (b) node embedding updates via edge aggregation, and (c) edge classification with residual connection.

To evaluate this approach, we conducted experiments on four public NIDS datasets: NF-UQ-NIDS, NF-CSE-CIC-IDS2018, CCD-INID, and X-IIoTID. These datasets cover diverse IoT and industrial scenarios, with varying graph densities and connectivity patterns. Their structural properties are

summarized in Table 4.1.

Table 4.1: Properties of graphs from datasets used in the experiments: average degree and density.

Dataset	***Average Degree***	***Density***
NF-UQ-NIDS	114.82	0.001
NF-CSE-CIC-IDS2018	79.83	0.00072
CCD-INID	453.17	2.697
X-IIoTID	4076.23	22.15

Results demonstrate that embedding centrality measures improves flow classification performance across all datasets. The weighted F1-score increased consistently compared to the baseline E-GraphSAGE with trivial initialization, while the false positive rate was also reduced. A summary of results is presented in Table 4.2.

Table 4.2: Weighted F1-scores of E-GraphSAGE and Residual E-GraphSAGE across datasets, showing improvements with centrality-informed initialization.

Dataset	**Model**	***F1-Score (Traditional)***	***F1-Score (Informed Initialization)***
NF-UQ-NIDS	E-GraphSAGE	97.01	97.62
	E-ResGraphSAGE	97.03	**97.70**
NF-CSE-CIC-IDS2018	E-GraphSAGE	97.78	98.42
	E-ResGraphSAGE	97.86	**98.51**
CCD-INID	E-GraphSAGE	94.52	94.77
	E-ResGraphSAGE	96.43	**96.77**
X-IIoTID	E-GraphSAGE	83.76	86.62
	E-ResGraphSAGE	83.83	**86.64**

These experiments confirm that structural priors help GNNs better exploit the edge-centric data of NIDS [171]. However, this solution focuses solely on feature initialization. It does not address other important factors such as scalability, neighborhood sampling, or attention-based weighting. To overcome these broader challenges, we introduce in the next section our main contribution, ADAP-GNN, a general framework that dynamically adapts GNN architectures to graph properties.

4.3 ADAP-GNN

The edge-centric models introduced earlier address specific limitations of the standard node-centric GNNs. However, the question of which architectural choices are most appropriate is not answered when choosing the right model for a given intrusion detection scenario. In the real world, networks differ widely in size, density, class balance, and topological heterogeneity, and no single GNN architecture consistently dominates across all conditions. To tackle this challenge, in this section, we introduce an Adaptive Property-Aware Graph Neural Network (ADAP-GNN) framework that dynamically selects between different edge-centric graph neural networks based on measurable properties of the underlying graph. This section describes the motivation for ADAP-GNN, details the architectural choices of GNNs, defines the graph metrics used in the selection, and presents the overall selection algorithm. Moreover, the effects of sampling and attention mechanisms are illustrated with toy examples, and then a comprehensive experimental evaluation across nine intrusion detection datasets is provided.

In ADAP-GNN, our goal is to build a systematic way to choose between two main choices in a GNN architecture: (i) using the complete set of neighbors versus sampling a subset during aggregation, and (ii) employing a uniform aggregation versus an attention mechanism. To apply the first one, we extend the E-GraphSAGE model to include optional neighbor sampling, and we present E-GAT, the edge-centric variant of the Graph Attention Network. We then define two graph metrics: (i) the proportion of attackers in the network and (ii) the entropy of the Degree Centrality. The values of these metrics guide the selection of model configurations. ADAP-GNN combines these components into a single framework: given a dataset, we construct a flow graph, compute its properties, and dynamically select the appropriate configuration before training and deployment.

4.3.1 Adaptive selection algorithm

In this section, we introduce the ADAP-GNN framework that uses the graph properties to select a suitable GNN configuration to enhance intrusion detection. Starting from the presentation of the edge-centric GNN models, we move on to define the graph and dataset measures we use in the selection algorithm that will be utilized.

4.3.1.1 Edge-centric GNN Models

In flow-based intrusion detection, each IP address is represented as a node and each communication between two devices is represented as a directed edge. Because nodes have no attributes, their features are initialized with a constant vector of ones and refined through message-passing layers. After K layers, each node v has an embedding $\mathbf{z}_v$, and the final edge embeddings are formed by combining the embeddings of the endpoints with the original edge features.

The E-GraphSAGE architecture adapts the inductive GraphSAGE model to the edge-centric setting. At layer k, a node v aggregates the feature vectors of its incoming edges $\mathbf{e}_{uv}^{(k-1)}$ for all neighbors $u \in N(v)$ using a differentiable function AGG^k. The aggregated message $\mathbf{h}_{N(v)}^k$ is concatenated with the node's previous representation and transformed by a learnable matrix $\mathbf{W}^k$ followed by a non-linear activation as explained in Section 4.1.3. To improve scalability and generalization, neighbor sampling can be used: instead of considering all neighbors of v, a random subset $\tilde{N}(v)$ of fixed size is selected. This technique reduces computational overhead and acts as a regularizer, but it may miss rare attack patterns if only a few nodes initiate malicious flows. After K layers, the node embeddings are combined with the raw edge features via an edge aggregation function AGG_{edge} to produce an embedding $\mathbf{z}_{uv}$ for each flow.

Graph Attention Networks extend the message-passing algorithm of GCN and GraphSAGE by learning to assign different weights to neighbors. We adapted this mechanism to the edge-centric setting to produce E-GAT. This was done by concatenating the embeddings of the source and target nodes to compute an attention score for each incident edge. For layer k, a raw attention score is computed, and the scores are then normalized using a softmax. These scores are used to weight the effect of each neighboring node on the process of updating the node's representation. The whole process of the E-GAT model is described in Algorithm 5.

4.3.1.2 Graph Metrics for Model Selection

To guide model selection, ADAP-GNN uses simple properties of the graph that are interpretable and easy to compute. The suitability of sampling and attention depends on two metrics from the flow graph: the proportion of attackers and the Degree centrality entropy.

The proportion of attackers is the proportion of the nodes that have initiated an attack to the

Algorithm 5 E-GAT Model

Input: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$,
Depth K of the GNN model,
Edge features $\mathbf{e}_{u\mathrm{v}}$ for each $(u, \mathrm{v}) \in \mathcal{E}$,
Non-linear activation function σ

Output: New edge embeddings $\mathbf{z}_{u\mathrm{v}}$ for each $u\mathrm{v} \in \mathcal{E}$

Initialize node features $\mathbf{h}_{\mathrm{v}}^0$ as a vector of ones $\forall \mathrm{v} \in \mathcal{V}$
for $k = 1$ **to** K **do**
  for each edge $u\mathrm{v} \in \mathcal{E}$ **do**
    $\mathbf{e}_{u\mathrm{v}}^{(k)} = \mathrm{CONCAT}(\mathbf{h}_u^{(k-1)}, \mathbf{h}_{\mathrm{v}}^{(k-1)})$
    $\mathbf{e}_{u\mathrm{v}}^{(k)} = \mathrm{LeakyReLU}(\mathbf{W}_{\mathrm{att}}^{(k)} \cdot \mathbf{z}_{u\mathrm{v}}^{(k)})$
  end for
  for each node $\mathrm{v} \in \mathcal{V}$ **do**
    $\alpha_{u\mathrm{v}}^{(k)} = \dfrac{\exp(\mathbf{e}_{u\mathrm{v}}^{(k)})}{\sum_{w \in N(\mathrm{v})} \exp(\mathbf{e}_{w\mathrm{v}}^{(k)})} \quad \forall u \in N(\mathrm{v})$
    $\mathbf{h}_{N(\mathrm{v})}^k = \sum_{u \in N(\mathrm{v})} \alpha_{u\mathrm{v}}^{(k)} \, \mathbf{z}_{u\mathrm{v}}^{(k)}$
    $\mathbf{h}_{\mathrm{v}}^k = \sigma\left(\mathbf{W}^k \cdot \mathrm{CONCAT}\left(\mathbf{h}_{\mathrm{v}}^{k-1}, \mathbf{h}_{N(\mathrm{v})}^k\right)\right)$
  end for
end for
Final node embeddings: $\mathbf{z}_{\mathrm{v}} = \mathbf{h}_{\mathrm{v}}^K \ \forall \mathrm{v} \in \mathcal{V}$
for each edge $u\mathrm{v} \in \mathcal{E}$ **do**
  $\mathbf{z}_{u\mathrm{v}} = \mathbf{W}^{edge} \cdot \mathrm{CONCAT}\left(\mathbf{z}_u, \mathbf{z}_{\mathrm{v}}, \mathbf{e}_{u\mathrm{v}}\right)$
end for
return $\mathbf{z}_{u\mathrm{v}}$ for each $u\mathrm{v} \in \mathcal{E}$

total number of nodes in the graph. Let $N_{\text{attackers}}$ be the number of nodes that have initiated at least one malicious flow, and let N_{total} be the total number of nodes. We define the attacker proportion as:

$$p = \frac{N_{\text{attackers}}}{N_{\text{total}}} \tag{4.5}$$

When p is small, malicious flows originate from a few devices and sampling neighbors risks omitting these critical nodes. In this case, full aggregation is recommended. Conversely, if many nodes participate in attacks, the risk of excluding all attackers is low, and sampling can be safely used to reduce computation.

The second metric is the entropy [172] of the Degree centrality [86], which quantifies heterogeneity in node connectivity to values between zero and one. It captures how concentrated or diverse node degrees are across the graph: values near zero reflect an almost regular topology, while values near one indicate a wide spread of degrees. For a graph with n nodes, the Degree Centrality of node i is:

$$C_D(i) = \frac{d_i}{n-1}, \tag{4.6}$$

where d_i is the sum of the in-degree and out-degree of i. Let $\mathcal{C}_D$ be the set of distinct Degree Centrality values and let p_c denote the fraction of nodes having centrality $c \in \mathcal{C}_D$. The entropy of the degree distribution is

$$E_D = \frac{-\sum_{c \in \mathcal{C}_D} p_c \log(p_c)}{\log(|\mathcal{V}|)}. \tag{4.7}$$

If E_D is high (close to 1), the degrees are widely spread and a uniform aggregation may not distinguish between hubs and peripheral nodes. An attention mechanism can then help emphasise informative neighbors. If E_D is low, most nodes have similar degrees except for a few outliers; in this case attention may overfit and uniform weights suffice.

4.3.1.3 Selection Algorithm

The decision logic of ADAP-GNN is displayed in Figure 4.4. The proportion of attackers determines whether neighbor sampling is applied: when p is below a dataset-dependent threshold, full aggregation is preferred to avoid missing rare attacks; when p is high, sampling reduces computation and serves

as a regularizer. The degree entropy governs the use of attention: high entropy motivates attention to differentiate between diverse node roles, whereas low entropy favors uniform weighting to prevent overfitting. These two binary decisions produce four possible configurations. In practice the thresholds for "high" and "low" are determined empirically from the datasets, but the qualitative pattern is consistent across our experiments. In our experiments, the threshold of the proportion of attackers is 0.022, and the threshold of the Degree centrality entropy is 0.344. These thresholds were determined by holding experiments on 9 datasets, and using logistic regression to determine the thresholds. A binary outcome indicating whether neighbor sampling performed better than full aggregation was assigned to each dataset. Next, we model the likelihood that neighbor sampling should be chosen as a function of the percentage of attackers in the graph by fitting a logistic function. The sigmoid curve's inflection point, or the point where the model predicts a 50% likelihood of favoring sampling, was used to determine the decision threshold. A threshold of roughly $p^* = 0.022$ was obtained from this investigation. Consequently, only when the percentage of attackers above this threshold does ADAP-GNN advise turning on neighbor sampling. Our selection mechanism's property-aware design is supported by the fitted curve and threshold point displayed in Figure 4.2. The mechanism was applied in Figure 4.3 to determine the threshold point of Degree Centrality entropy.

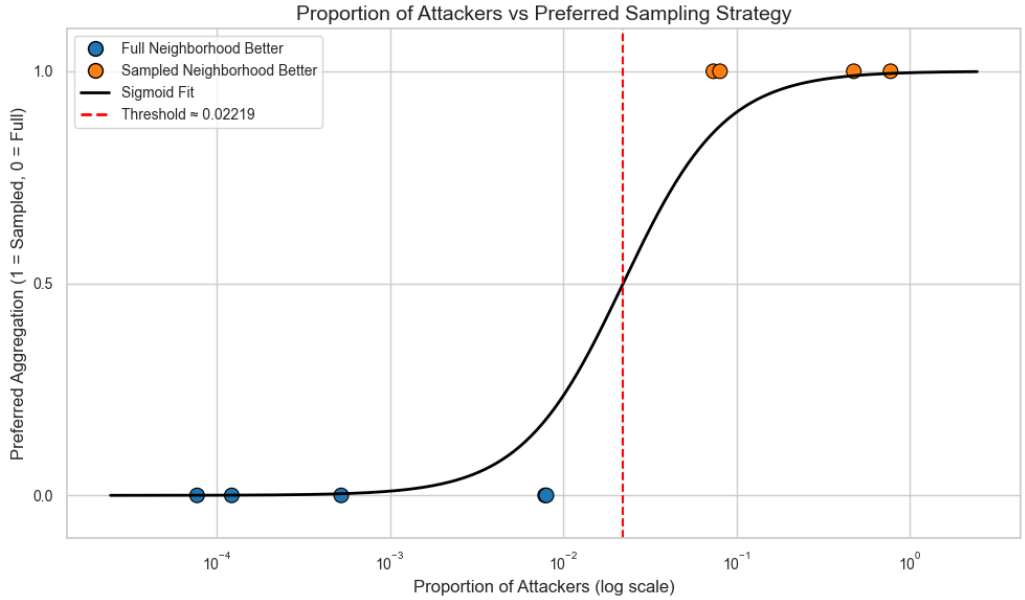


Figure 4.2: Logistic regression used to determine the threshold on the proportion of attackers. The decision threshold $p^* = 0.022$ corresponds to the inflection point of the fitted sigmoid curve.

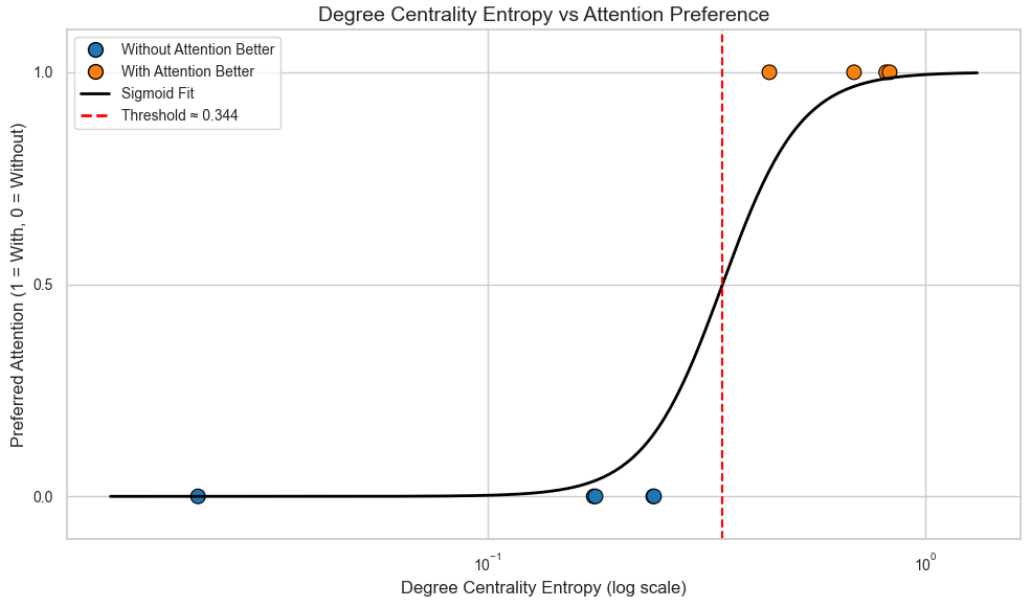


Figure 4.3: Logistic regression to determine the threshold on Degree Centrality entropy. The attention mechanism is recommended when the entropy exceeds $E^* = 0.344$.

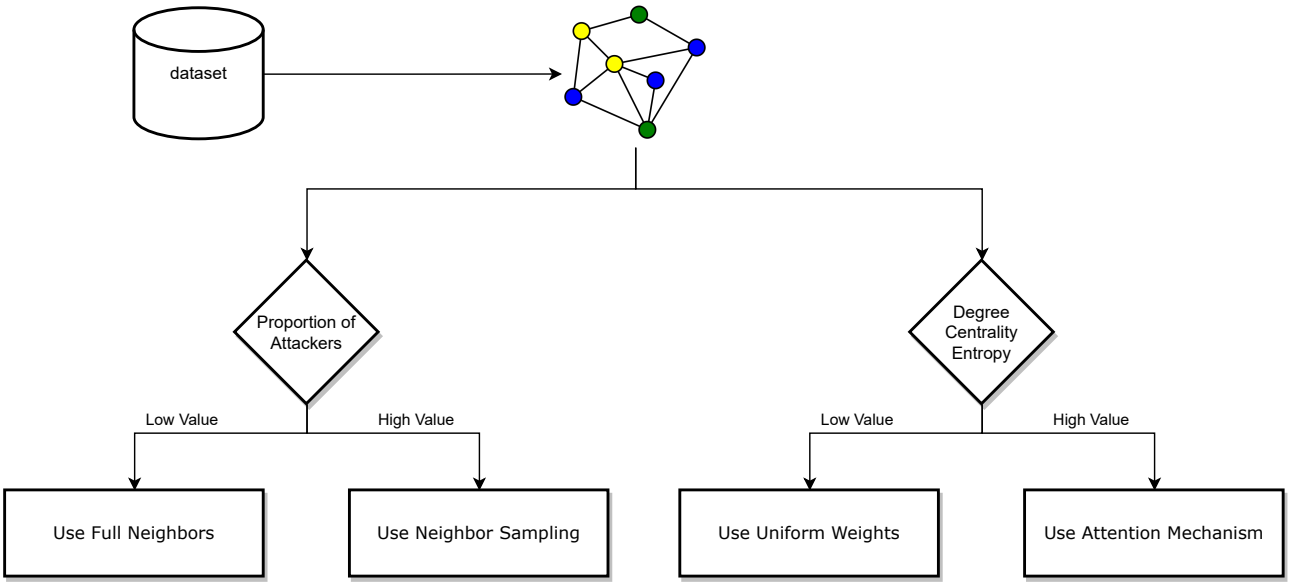


Figure 4.4: Decision chart of the ADAP-GNN selection algorithm. The left branch uses the attacker proportion to decide whether to sample neighbors, and the right branch uses the degree entropy to decide whether to employ an attention mechanism.

4.3.1.4 General Scheme of ADAP-GNN

The workflow of the proposed framework is summarized in Figure 4.5: starting from a collection of flow records, we first build a directed multi-graph in which vertices represent IP addresses and edges represent network flows. All available flow attributes are treated as edge features. We then compute the attacker proportion and degree entropy described above. These metrics drive two binary decisions: whether to use neighbor sampling, and whether to employ an attention mechanism. According to the decisions, we instantiate either E-GraphSAGE or E-GAT with or without sampling and attention. The selected model is trained on the constructed graph and applied to classify each flow as benign or malicious.

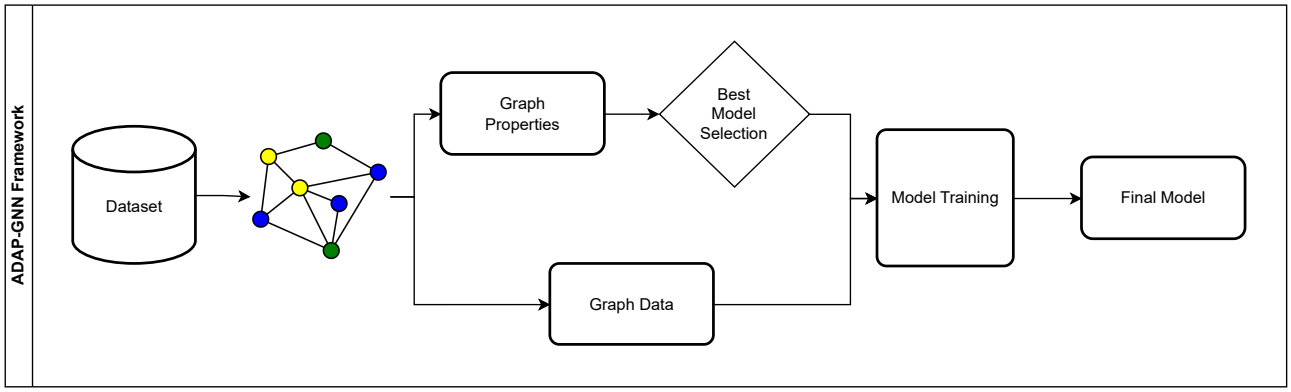


Figure 4.5: General scheme of the ADAP-GNN framework: The flow dataset is transformed into a graph where nodes are devices and edges are flows. Graph properties (attacker proportion and degree entropy) are calculated and used to choose the GNN configuration. The selected model is trained and then used to classify flows.

4.3.1.5 Toy examples

To make the link between graph properties and GNN configuration concrete, two didactic toy cases are provided. Beyond the baseline GNN setup, we consider the two architectural choices: neighbor sampling and attention mechanism. The first example (Figure 4.6) displays how the proportion of attackers drives the sampling decision, and the second (Figure 4.7) illustrates how degree-centrality entropy decides what kind of attention should be used. Each figure shows a small directed graph centered on a node v with neighbors $u_1, u_2, \ldots$ to keep the aggregation step visually traceable. Nodes are circles: red indicates that the node has issued at least one attack, while yellow denotes benign behavior. When node sizes differ, larger circles indicate higher Degree Centrality (i.e., more connections).

Directed edges are drawn as arrows, colored red for attack flows and yellow for benign flows.

The effect of neighbor sampling is tightly related to the fraction of attackers present in the network. In Figure 4.6(a), only two of the thirteen nodes are attackers, yielding a low attacker proportion (≈ 0.15). Applying stochastic neighbor sampling in this setting can easily drop the few informative neighbors and thus erase important attack information and patterns, as shown by the sampled (blue) region that excludes all attackers. In contrast, when the attacker proportion is higher, selecting random neighbors remains likely to capture at least some malicious nodes. In Figure 4.6(b), five of thirteen nodes are attackers (≈ 0.38), and the sampled neighborhood still has attack edges around v. With the presence of these attacks, aggregation can therefore pass useful adversarial patterns to the classifier even under sampling.

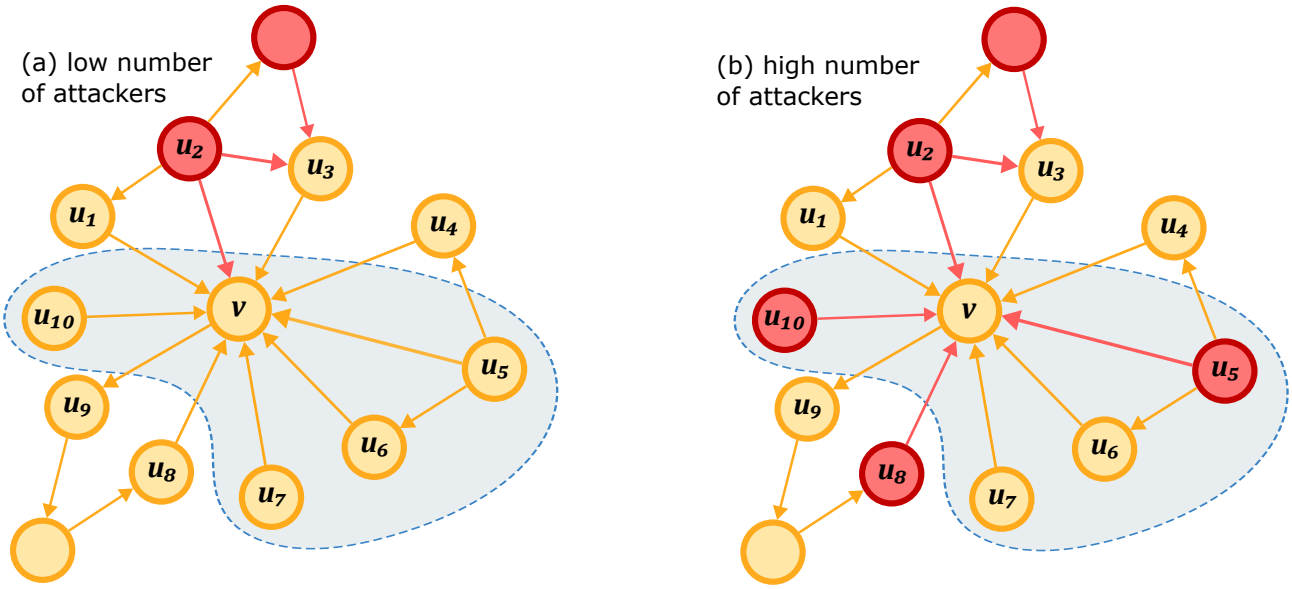


Figure 4.6: Toy example showing how the attacker proportion affects neighbor sampling. Red nodes/arrows indicate attackers/attack flows, and yellow denote benign devices/flows. The blue region marks sampled neighbors used for aggregation. (a) Low attacker proportion: sampling misses all attackers. (b) Higher attacker proportion: sampling keeps some attackers, preserving useful patterns and features.

In GNNs, aggregation determines how a node gets information from its neighborhood. Adding attention lets the model weigh neighbors by relevance instead of assigning uniform weights to all of them, but this additional capability can overfit when the neighborhoods are too homogeneous in general with few outliers. In ADAP-GNN, the Degree centrality entropy is used to guide this trade-off. Figure 4.7(a) shows a low-entropy graph: aside from v and a single prominent neighbor u_1, most

nodes share the same small degree. Here, attention tends to lock onto the dominant pattern and may over-specialize. By contrast, Figure 4.7(b) has a high-entropy graph with diverse degrees and no single node type overshadowing others. In this heterogeneous setting, attention can productively differentiate contributions across neighbors, amplifying informative nodes without overfitting.

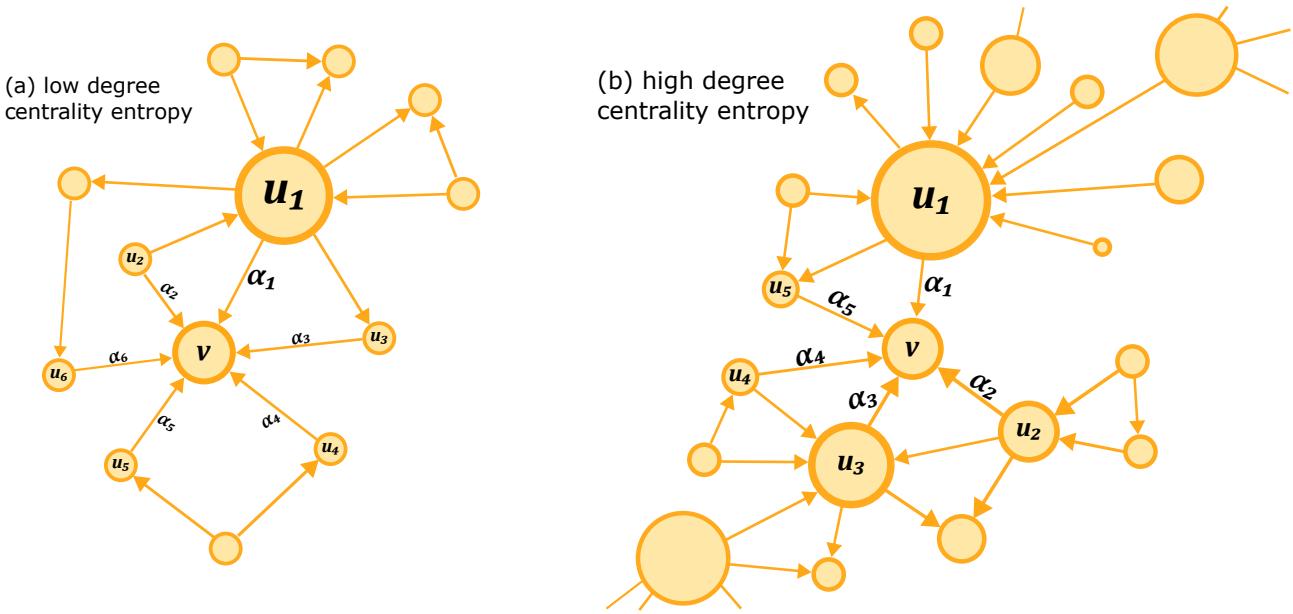


Figure 4.7: Toy example contrasting low and high entropy degree distributions and their impact on attention mechanism. In the graph, the node size encodes Degree Centrality. (a) Low entropy: most nodes share the same degree, so attention tends to overfit to a dominant neighbor. (b) High entropy: diverse degrees allow attention to weight neighbors meaningfully without overfitting on a single pattern.

4.3.2 Experimental Setup

This section reports our experiments with E-GraphSAGE and E-GAT for NIDS, and evaluates the proposed ADAP-GNN across nine datasets that differ in size, topology, and format. We first specify the training setup and then outline the datasets and present their properties.

We implemented the GNN models in PyTorch with the Deep Graph Library (DGL). Each dataset was split into training, validation, and test partitions. After every epoch, we evaluated on the validation set to monitor overfitting; the checkpoint with the best validation F1-score was kept and later evaluated on the test set. The number of training epochs was scaled with dataset size, ranging from 500 for the largest datasets up to 5000 for the smallest, in order to balance computational cost with

convergence, and to avoid early stopping while the loss was still decreasing meaningfully. We used the Adam optimizer with a learning rate of 0.005. The GNN depth was fixed to two message-passing layers, sampling 25 neighbors in the first layer and 10 in the second when sampling was enabled. Node embeddings had dimension 128 with ReLU activations. For a fair comparison across highly imbalanced datasets, we used the weighted F1-score as the primary metric.

In this experiment, we employed nine publicly available datasets covering a wide selection of traffic conditions and attack families. Table 4.3 summarize, for each dataset, its size (number of flow records), main attack types, fraction of attack records, together with the graph metrics used by ADAP-GNN (proportion of attacker nodes and degree-centrality entropy) and the resulting configuration decisions (neighbor sampling and attention mechanism).

The nine datasets used in this experiment are:

CIC-ToN-IoT: The CICFlowMeter representation of the TON-IoT dataset focuses on evaluating cybersecurity applications in the Internet of Things (IoT) [15]. The dataset contains nine type of attacks including: Denial-of-Service (DoS), Distributed Denial-of-Service (DDoS), and Man-In-The-Middle (MITM).

CIC-ToN-IoT-Snapshot: A modified version of the CIC-ToN-IoT dataset created by extracting a specific time window from the original dataset [173]. This approach aims to introduce a dataset with different properties while preserving the original network relationships and ensuring that every node retains realistic attributes rather than being arbitrarily generated.

CIC-IDS-2017: A modern, realistic, and diverse dataset to test Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs). Sharafaldin *et al.* simulated a real network environment containing benign and the most up-to-date common attacks [16]. Of these attacks, there are FTP-Patator, Botnet, DoS, and DDoS.

NF-UQ-NIDS: A compilation of multiple network flow datasets into a comprehensive universal NIDS benchmark with diverse attack scenarios [50]. Attacks from the datasets are unified under parent classes like DoS, DDoS, brute force, and injection. The dataset contains over 11.9 million records, out of which 76.77% are benign flows and 23.23% are attacks.

NF-UQ-NIDS-Unified-Community: A modified version of the NF-UQ-NIDS dataset created by a community mixing modification [173]. The transformation achieves a more unified community

structure while maintaining the relative proportions of attack and normal traffic.

NF-CSE-CIC-IDS2018: A NetFlow-based version of the original CSE-CIC-IDS2018, offering a total of 8,392,401 flows [50]. It has 87.86% benign flows and 12.14% attack samples. BruteForce, Bot, DoS, DDoS, Infiltration, and Web Attacks are the six attack types found in this dataset.

CCD-INID-Centrality-Focused: The Center for Cyber Defense IoT Network Intrusion Dataset (CCD-INID) provides an advanced modern dataset that reflects IoT networks activities to help improving the security of this type of networks [174]. We used a centrality-based modified version that focuses on nodes with high centrality values [173]. The modified version contains 64,199 records, where 52% are benign and 48% are attacks.

Edge-IIoTset: Ferrag et al. [17] used a wide variety of IoT device types to develop the Edge Industrial IoT dataset (Edge-IIoTset). The "selected for ML and DL" subset of this dataset was employed in our experiment. Of the 2,219,201 classified traffic records in the subset, 72.8% are benign, while the remaining records are attack activities. DoS/DDoS attacks, information gathering, MITM, injection, and malware attacks are the five primary categories of attacks in the subset.

X-IIoTID: The extensive and connectivity-agnostic X-IIoTID dataset was created to tackle the particular difficulties associated with securing Industrial Internet of Things (IIoT) systems. In addition to providing multiview features including network traffic, host resources, logs, and warnings, it records a variety of attack scenarios, modern IIoT connectivity protocols, and multivendor device behaviors [175]. Of the 631,814 records in the X-IIoTID dataset, 51.4% are benign and 48.6% are attacks.

4.3.3 Results

This section reports the performance of E-GraphSAGE and E-GAT for intrusion detection, and analyzes how ADAP-GNN behaves across datasets with different graph properties. The presentation is organized in three parts. First, we compare E-GraphSAGE and E-GAT against conventional deep learning baselines to highlight the value of relational patterns in network traffic. Second, we study the impact of neighbor sampling, focusing on how the proportion of attackers influences aggregation. A timing comparison is included to show the efficiency gains of sampling. Third, we examine the attention mechanism through the lens of the Degree centrality entropy, identifying when attention improves performance. We also report the inference-time overhead of attention.

Table 4.3: Datasets, traffic scale, and ADAP-GNN graph properties/decisions. The table lists record counts (Size), the percentage of attack records (% Attacks), the proportion of attacker nodes (Proportion of attackers), degree-centrality entropy (Degree entropy), and the configuration selected by ADAP-GNN for neighbour sampling (NS) and attention (Att.).

Dataset	***Size***	***% Attacks***	***Proportion of attackers***	***Degree Entropy***	***NS***	***Att.***
CIC-ToN-IoT [15]	5,350,583	53.01%	0.000077	0.1749	✗	✗
CIC-ToN-IoT-Snapshot [173]	4,915,090	57.24%	0.00012	0.1765	✗	✗
CIC-IDS-2017 [16]	2,814,878	19.5%	0.00052	0.4405	✗	✓
NF-UQ-NIDS [50]	10,729,039	21.33%	0.0079	0.2402	✗	✗
NF-UQ-NIDS-Unified-Community [173]	2,203,216	29.89%	0.0736	0.6877	✓	✓
NF-CSE-CIC-IDS2018 [50]	7,314,661	8.64%	0.00786	0.2392	✗	✗
CCD-INID-Centrality-Focused [173]	64,199	47.95%	0.4757	0.8146	✓	✓
Edge-IIoTset [17]	2,031,921	29.35%	0.776	0.0218	✓	✗
X-IIoTID [175]	631,814	48.59%	0.0803	0.8297	✓	✓

Dataset	*MLP*	*CNN*	*GRU*	*LSTM*	*E-GraphSAGE*	*E-GAT*
CIC-IDS-2017	92.11	97.11	95.39	95.19	98.32	**98.58**
CIC-ToN-IoT	77.05	79.68	76.72	76.53	**84.85**	84.61
CIC-ToN-IoT-Snapshot	76.56	78.68	75.36	76.29	**85.04**	84.63
NF-UQ-NIDS	81.68	85.61	85.78	85.49	**87.22**	84.81
NF-UQ-NIDS-Unified-Community	87.43	87.39	83.25	84.22	87.77	**90.78**
NF-CSE-CIC-IDS2018	84.12	91.32	89.56	90.45	**95.79**	95.41
X-IIoTID	87.25	89.35	90.27	89.86	92.14	**94.77**
CCD-INID-Centrality-Focused	80.43	83.51	82.41	82.65	85.32	**87.47**
Edge-IIoTset	98.09	**99.12**	98.12	98.01	95.91	95.21

Table 4.4: Comparison of F1-scores among traditional deep learning models (MLP, CNN, GRU, LSTM) and graph-based approaches (E-GraphSAGE, E-GAT) across multiple intrusion detection datasets. Bold values indicate the best performance for each dataset.

4.3.3.1 Performance Analysis of E-GraphSAGE and E-GAT in NIDS

GNNs are designed to exploit relational structure in graphs, which makes them powerful in the NIDS context. To show how effective they are, we benchmark the GNN models against widely used deep learning models: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM). All methods share identical preprocessing, splits, and evaluation settings for fairness. Final F1-scores are listed in Table 4.4. While convolutional and recurrent models benefit from additional structure, graph-based approaches achieve the highest performance on most datasets, underscoring the benefit of relational modeling. Notably, there is no single GNN winner: E-GAT leads on several datasets, whereas E-GraphSAGE tops others, motivating a property-aware selection strategy.

To further illustrate the benefit of graph structure, we analyze PortScan detection in CIC-IDS-2017. Port scanning explores open ports to gather information and search for potential vulnerabilities [176]; because it spans interactions across multiple hosts, it is more naturally captured in a graph than in isolated records. Figure 4.8 shows that GNNs markedly outperform deep learning baselines on this

attack class: MLP achieves 87.54, CNN 90.35, GRU 90.23, and LSTM 90.22, while E-GraphSAGE reaches 99.06 and E-GAT 0.99.28. This gap highlights how relational context improves detection.

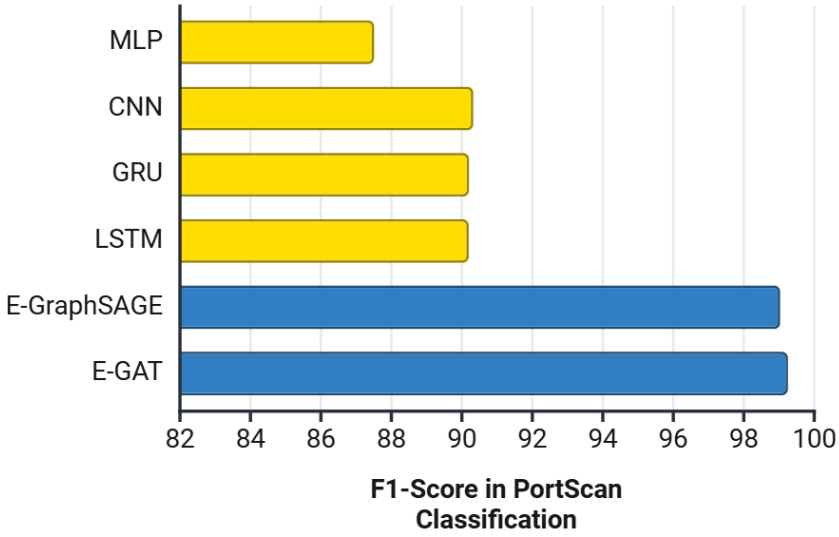


Figure 4.8: F1-scores for port scan detection in CIC-IDS-2017. Yellow bars: deep learning models (MLP, CNN, GRU, LSTM). Blue bars: graph-based models (E-GraphSAGE, E-GAT). Graph-based approaches substantially outperform conventional models on this multi-host attack.

4.3.3.2 Importance of Neighbor Sampling in NIDS

Having confirmed the advantage of GNNs in NIDS, we now examine how configuration choices affect performance. Neighbor sampling replaces full-neighbor aggregation with a random subset, reducing computation, improving runtime, and acting as a regularizer. However, there is a risk of omitting rare, informative neighbors when the number of attackers is insufficient. Therefore, ADAP-GNN prevents sampling when the attacker proportion is low. Table 4.5 lists F1-scores for full versus sampled aggregation along with attacker proportions. Full aggregation tends to win when attackers are rare (e.g., CIC-ToN-IoT at 0.000077 and CIC-IDS-2017 at 0.000523), whereas sampling wins as attacker proportion grows (e.g., NF-UQ-NIDS-Unified-Community at 0.073620 and Edge-IIoTset at 0.776236). This pattern supports the selection rule embedded in ADAP-GNN.

Figure 4.9 visualizes these trends: full-neighbor aggregation (red) dominates when attackers are scarce, since sampling risks excluding critical malicious neighbors; as the attacker proportion increases,

Dataset	*Proportion of Attackers*	*Full Neighborhood*	*Sampled Neighborhood*
CIC-ToN-IoT	0.000077	**84.85**	82.09
CIC-ToN-IoT-Snapshot	0.000122	**85.04**	82.34
CIC-IDS-2017	0.000523	**98.32**	98.13
NF-CSE-CIC-IDS2018	0.007862	**95.79**	94.46
NF-UQ-NIDS	0.007998	**87.22**	85.84
NF-UQ-NIDS-Unified-Community	0.073620	87.77	**91.03**
X-IIoTID	0.080357	92.14	**91.64**
CCD-INID-Centrality-Focused	0.475728	85.32	**85.55**
Edge-IIoTset	0.776236	95.91	**96.41**

Table 4.5: GNN performance sorted by the proportion of attackers. The table compares F1-scores using full versus sampled neighborhood aggregation, where bold indicates the higher score per dataset. Sampling helps when attackers are more prevalent.

sampled aggregation (blue) overtakes.

From a systems perspective, neighbor sampling halves inference time on average (Figure 4.9), a useful gain for timely detection. While sampling regularizes and speeds up inference, its use should be guided by the proportion of attackers to avoid discarding rare but informative edges.

4.3.3.3 Importance of Attention Mechanism in NIDS

ADAP-GNN also adapts the use of attention based on the Degree centrality entropy. Low entropy indicates connectivity concentrated in a few nodes, where attention can overfit. On the other hand, high entropy indicates heterogeneous neighborhoods, where attention can add value by increasing the weights of informative neighbors. Table 4.6 compares configurations with and without attention, sorted by entropy. Attention improves performance on higher-entropy graphs (e.g., CIC-IDS-2017: 98.58 vs. 98.32; NF-UQ-NIDS-Unified-Community: 90.78 vs. 87.77; X-IIoTID: 94.77 vs. 92.14), but is counterproductive on low-entropy graphs (e.g., Edge-IIoTset: 95.21 vs. 95.91). For all datasets, attention is either beneficial or with a marginal reduction in accuracy, only in NF-UQ-NIDS, uniform weights are significantly better, an example where attention overfits.

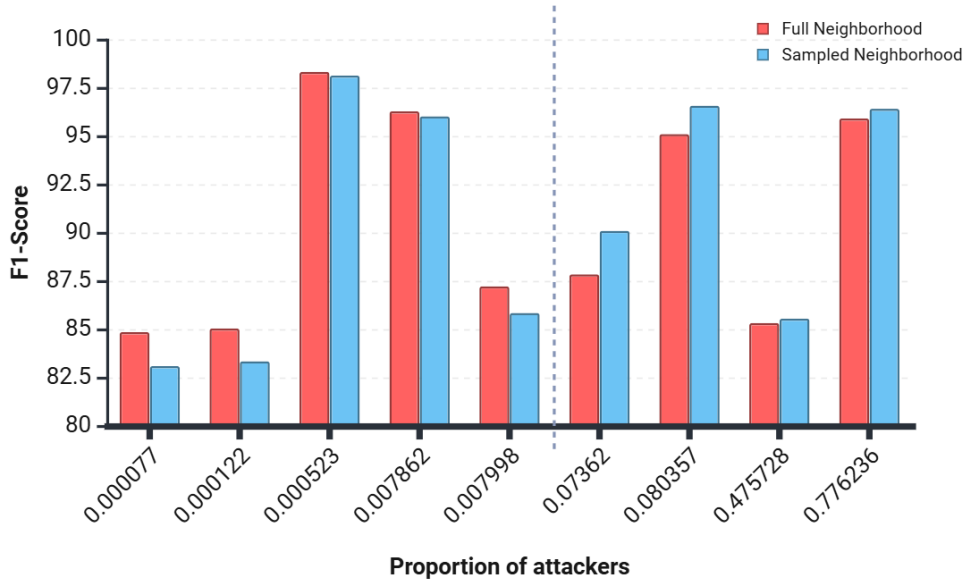


Figure 4.9: F1-scores of E-GraphSAGE with full versus sampled neighborhood aggregation across datasets ordered by attacker proportion. A vertical dashed line marks the approximate regime where sampling begins to dominate.

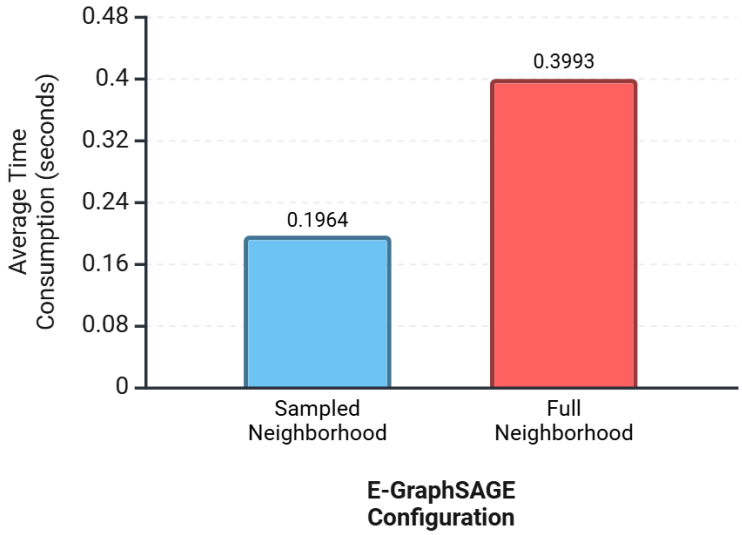


Figure 4.10: Average testing time (seconds) of E-GraphSAGE across datasets. Blue: sampled neighborhood, and red: full neighborhood. Sampling yields consistent runtime reductions.

Dataset	*Degree Centrality Entropy*	*Without Attention*	*Attention*
Edge-IIoTset	0.021798	**95.91**	95.21
CIC-ToN-IoT	0.174882	**84.85**	84.61
CIC-ToN-IoT-Snapshot	0.176466	**85.04**	84.63
NF-CSE-CIC-IDS2018	0.239182	**95.79**	95.41
NF-UQ-NIDS	0.240163	**87.22**	84.81
CIC-IDS-2017	0.440548	98.32	**98.58**
NF-UQ-NIDS-Unified-Community	0.687675	87.77	**90.78**
CCD-INID-Centrality-Focused	0.814577	85.32	**87.47**
X-IIoTID	0.829736	92.14	**94.77**

Table 4.6: Datasets ordered by the Degree centrality entropy, comparing F1-scores without and with attention. Bold indicates the higher score per dataset. Attention helps at higher entropy, but may overfit at low entropy.

Figure 4.11 visualizes this dependency: uniform aggregation (red) slightly dominates at low entropy, while attention (green) improves results at higher entropy. NF-UQ-NIDS is the main exception, where attention underperforms due to overfitting.

Training/validation loss curves on NF-UQ-NIDS (Figure 4.12) show lower training loss with attention but higher validation loss, consistent with overfitting; this trend persisted across dropout and weight-decay settings. Consequently, attention should be reserved for higher-entropy graphs where the added capacity generalizes.

Finally, Figure 4.13 reports inference time with and without attention. Although attention introduces extra computation for dynamic weights, the added cost is modest relative to its accuracy gains on high-entropy datasets. This aligns with ADAP-GNN's strategy of enabling attention only when warranted.

In summary, graph properties tightly govern which GNN configuration is preferable. Neighbor sampling trades a small risk of information loss (when attackers are scarce) for substantial speed and regularization benefits (when attackers are prevalent). Attention adds discriminative power in heterogeneous graphs but risks overfitting in homogeneous ones. These trade-offs are converted by

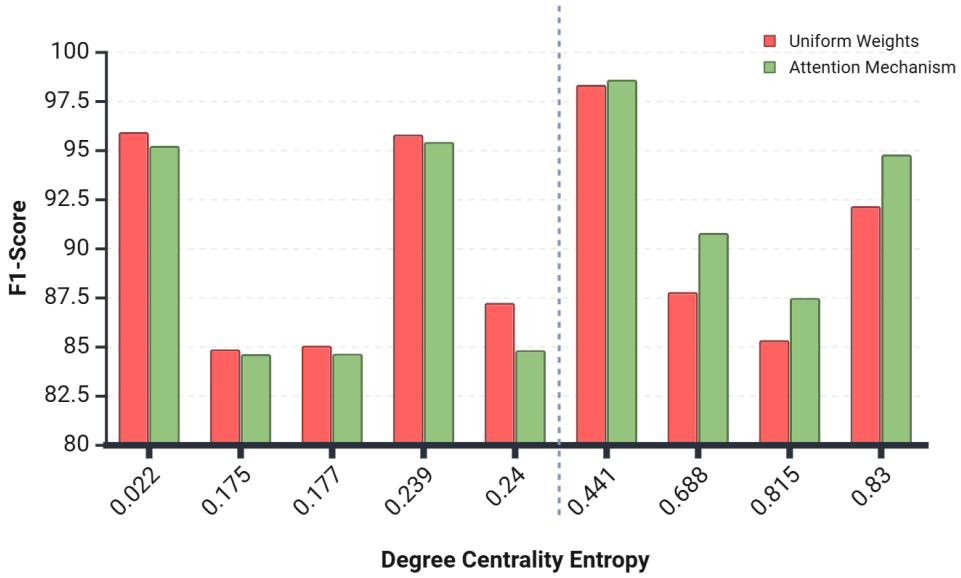


Figure 4.11: F1-scores of E-GraphSAGE with uniform aggregation versus attention across datasets ordered by the Degree centrality entropy. A vertical dashed line marks where attention begins to dominate.

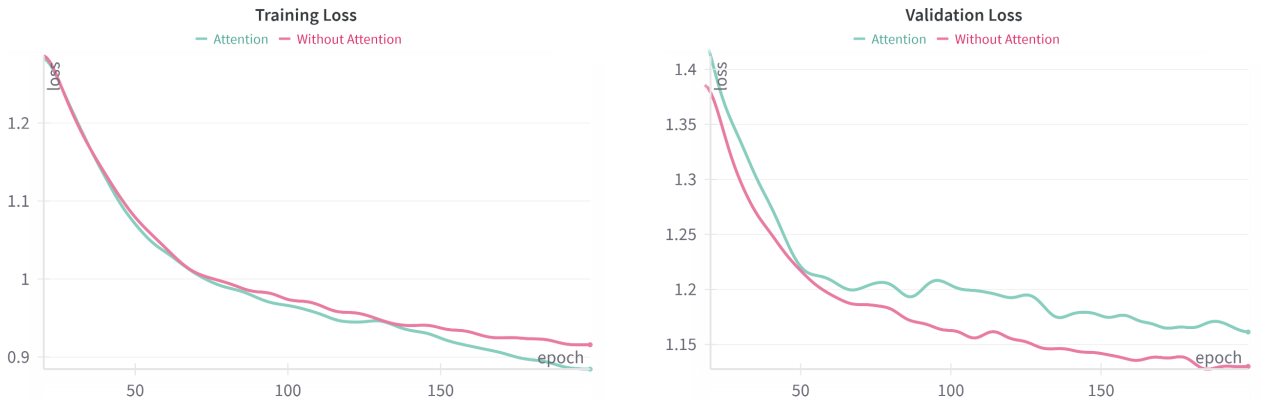


Figure 4.12: Training and validation loss with and without attention on NF-UQ-NIDS. Attention achieves lower training loss but higher validation loss, indicating overfitting.

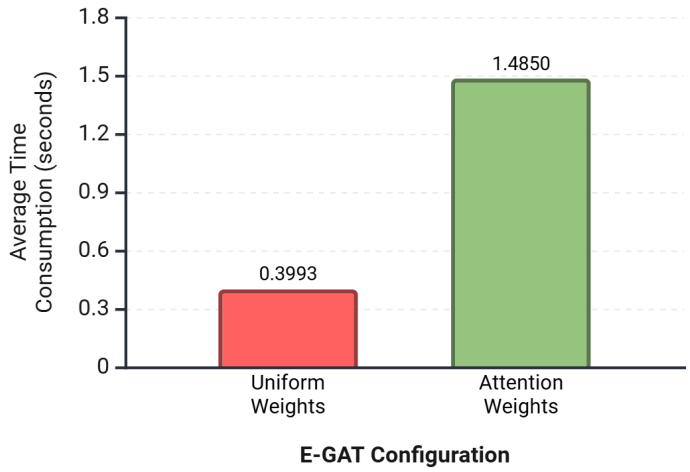


Figure 4.13: Average testing time (seconds) of E-GAT across datasets. Red for uniform aggregation, and green for attention. The overhead of attention is modest and justified when it improves accuracy.

ADAP-GNN into a property-aware selection algorithm that provides precise and effective intrusion detection in a variety of contexts.

4.4 Conclusion

This chapter developed a graph-based view of flow intrusion detection and showed how to overcome the core mismatch between edge-centric NIDS data and node-centric GNN architectures. We began by formalizing message passing and motivating graphs for cybersecurity, then introduced E-GraphSAGE as an edge-centric adaptation that learns node representations directly from incident flows. To prevent parallel flows between the same node pair from collapsing into a single representation, we incorporated residual connections (E-ResGraphSAGE), re-injecting raw edge features at classification time so that individual flows remain distinguishable. Building on our earlier complex-network insights, we proposed a centrality-informed initialization that integrates nodes with structural features in the form of centrality measures: Degree, Betweenness, Closeness, PageRank, and K-truss. Across NF-UQ-NIDS, NF-CSE-CIC-IDS2018, CCD-INID, and X-IIoTID, this initialization consistently increased weighted F1 versus trivial all-ones starts, while also reducing false positives. This proved that topology-aware initialization of node features helps GNNs capture relational attack patterns that edge-only models

miss.

We then generalized from "one best model" to ADAP-GNN, a property-aware framework that chooses between full vs. sampled neighborhoods and uniform vs. attention-based aggregation using two interpretable graph metrics: the proportion of attacker nodes (guiding sampling) and degree-centrality entropy (guiding attention), with empirically derived thresholds (≈ 0.022 and ≈ 0.344, respectively). Experiments on nine datasets showed that GNNs (E-GraphSAGE/E-GAT) outperformed strong deep-learning baselines (MLP/CNN/GRU/LSTM). A PortScan case in CIC-IDS-2017 highlighted the value of relational context (E-GraphSAGE 99.06, E-GAT 99.28) over record-wise models. Experiments also displayed that the chosen configuration tracks graph conditions: (i) when attackers are rare, full aggregation beats sampling (e.g., CIC-ToN-IoT 84.85 vs. 82.09), and when attackers are prevalent, sampling improves accuracy and roughly halves inference time. (ii) When degree entropy is high, attention helps (e.g., X-IIoTID 94.77 vs. 92.14), but at low entropy it can overfit (e.g., NF-UQ-NIDS $84.81 < 87.22$). Taken together, these results establish a clear formula for practical NIDS with GNNs: preserve flow individuality using residual information, initialize nodes with structural centralities, and adapt sampling/attention to measured graph properties. Future improvements extend to temporal graphs, more complex attention mechanisms, and integration with federated settings where graph features and model updates remain privacy-preserving.

Conclusion

The work presented in this thesis addresses one of the fundamental challenges of modern cybersecurity: the detection of intrusions in Internet of Things (IoT) networks. The guiding hypothesis underlying this research is that the structural organization of interactions between connected devices, often overlooked in traditional approaches, constitutes a decisive source of information for improving the accuracy, robustness, and generalizability of intrusion detection systems. This conviction led to the development of a structure-first approach, grounded in complex network theory and machine learning, whose three main contributions are: the GDLC, the Fed-GDLC, and the ADAP-GNN frameworks. These contributions represent a logical and coherent progression toward more integrated, distributed, and intelligent detection mechanisms.

The first contribution, the GDLC (Graph Deep Learning based on Centrality) framework, demonstrated that topological measures derived from communication graphs significantly enhance the representation of network flows used by detection models. These measures include degree, closeness, betweenness, eigenvector centralities, and PageRank. Integrating these measures within conventional deep learning architectures, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (LSTM and GRU), led to measurable and consistent improvements across multiple benchmark datasets. On the CIC-ToN-IoT and CIC-IDS2017 datasets, the average F1-score increased by up to 5.33 percentage points, compared to baseline models, while the average false positive rate decreased by up to 76.97%. Collectively, this contribution addresses **H1** and **H2**: it validates structure-derived signal as a source of discriminative information beyond flow features (H1), and shows that injecting a compact, non-redundant set of centralities into CNN/LSTM/GRU consistently boosts performance and lowers false positives across datasets (H2). Ablation studies confirm that the gains stem from centrality information rather than increased model capacity. Taken together, these empirical gains provide a positive answer to **RQ1**: topological (complex-network) measures do carry discriminative

power for NIDS. They also address **RQ2**: the largest and most robust improvements arise from a minimal, non-redundant subset of measures selected according to the graph's structural properties. This contribution also speaks to RQ6. Using a Susceptible–Infected–Recovered (SIR) model, we showed that the selected centralities align with propagation control, providing a mechanistic interpretation of why they aid detection. In parallel, empirical cumulative distribution functions (ECDFs) of centrality values improved the interpretability of these measures in the cybersecurity context.

The second contribution, Fed-GDLC, extended this principle to federated learning, where data privacy and heterogeneous client distributions pose major challenges. Building on a federated principal component analysis (FedPCA) step and local selection of the most relevant centrality metrics, multiple clients collaboratively trained a global model without exchanging raw traffic, even when each client had distinct IoT network properties. Under non-Independent and Identically Distributed (non-IID) settings, topology-aware features regularized the FL process and improved both fairness and efficiency: over federated rounds, the F1-score rose from 68.2% to 75.1%, after adding the most relevant centrality metrics. This contribution validates **H3**, demonstrating the stabilizing and unifying role of topology in distributed, heterogeneous learning environments, and directly addresses **RQ3** by showing that centrality-informed local models and FedPCA-aligned aggregation preserve and improve accuracy while reducing variance compared to FL systems relying purely on flow features.

The third contribution, *Adaptive Property-Aware Graph Neural Network (ADAP-GNN)*, is the natural extension of the centrality-based approaches introduced in GDLC and Fed-GDLC: instead of using centralities only as auxiliary features, ADAP-GNN uses measured topology to steer the design and configuration of edge-centric GNNs. Concretely, it integrates complex-network measures with GNNs in two stages: (i) a centrality-informed initialization of node embeddings for E-GraphSAGE and E-ResGraphSAGE, and (ii) adaptive decisions on attention and neighbor sampling driven by graph properties such as Degree centrality entropy and the proportion of attackers. The centrality-informed initialization improved the weighted F1-score of strong E-GNN baselines by about three points, with the most pronounced gains on heterogeneous and high-entropy graphs. Moreover, the adaptive mechanisms achieved a balanced trade-off between accuracy and efficiency, improving detection stability while reducing inference cost by up to half through entropy-aware sampling and selective attention. Taken together, these results confirm **H4**: complex-network measures can guide not only feature design but also architectural and training choices in GNNs, yielding models that are both more accurate and

more efficient than unguided, one-size-fits-all configurations. This contribution also directly addresses **RQ4**, **RQ5**, and **RQ6**: it demonstrates how edge-centric GNNs can be adapted for intrusion detection by incorporating centrality measures into the message-passing process (RQ4); how topology-aware properties such as centrality entropy can inform when to sample or apply attention for optimal performance (RQ5); and how graph-based methods can sustain scalability and interpretability through efficient sampling and explainable propagation patterns, even at the million-edge scale (RQ6).

Taken together, these contributions illustrate the methodological coherence of this work: GDLC established the relevance of structural metrics in a centralized context; Fed-GDLC ensured their transferability and privacy preservation in a federated setting; and ADAP-GNN advanced toward direct and adaptive neural exploitation of network structure. These three frameworks thus form a logical chain of innovation, spanning from graph-based feature engineering to explicit neural modeling of inter-device interactions. They collectively demonstrate that a structure-centered approach to intrusion detection provides a unifying perspective linking topological analysis, deep learning, and federated learning.

From a scientific standpoint, the contributions of this research are manifold. First, it establishes a formal link between complex network theory and deep learning architectures applied to cybersecurity, offering an interpretable approach to model behavior. Second, it introduces the integration of federated PCA in the context of intrusion detection systems, allowing feature space alignment without compromising the privacy of local data. Finally, it proposes an adaptive GNN architecture selection mechanism guided by structural metrics, paving the way for truly autonomous and self-configurable models.

Our approach relies on a static graph construction option that may affect deployment in resource-constrained environments. A dynamic representation of the graph will improve robustness and usability of the system in real-world deployment. Also, training GNNs on very large, evolving graphs remains computationally demanding; neighbor sampling and mini-batching alleviate but do not remove this cost. Finally, while centralities improve performance, they add feature-engineering overhead and must be recomputed in streaming settings.

The results obtained demonstrate that a structure-centered approach, combining topological modeling, deep learning, and knowledge federation, provides a robust and scalable alternative to traditional intrusion detection methods. By placing network structure at the core of algorithmic reasoning, this

thesis contributes to the emergence of a new generation of intelligent cybersecurity systems capable of understanding, learning, and anticipating abnormal behaviors in increasingly dynamic connected environments. Through the integration of graph theory, artificial intelligence, and distributed systems engineering, it ultimately opens the way for explainable, adaptive, and real-time operational NIDSs tailored to the critical infrastructures of tomorrow.

Research Perspectives

To translate our findings into more robust and deployable NIDS, we extend this work along six prioritized directions. These directions deepen our centrality-augmented pipelines, advance federated and graph neural methods under non-IID dynamics, and target operational gaps in privacy, evaluation, and continuous learning.

1. *Federated GNNs for Non-IID Traffic and Drift:* To further improve generalizability and application of GNN in NIDS, it should be adapted to a federated graph neural network (FL-GNN) approach where each client holds a heterogeneous, evolving subgraph. This adaptation needs extensive testing and research, as many research questions can be raised: (i) What weights should be aggregated? Including the GNN layers or just the classification layer. (ii) Should the architecture of the global GNN model adapt to the local networks? (iii) How does this approach affect the explainability of GNN in these networks?

2. *Graph Construction at Stream Time:* To improve the applicability of graph-based approaches in NIDS, a move should be made from static snapshots to streaming, incremental graphs that update at flow-arrival time and reflect the evolving attack surface. Rolling windows would be maintained with incremental centralities (temporal betweenness and temporal PageRank). Online graph building should be implemented, answering questions as: what should be the window length? Or should a temporal graph be used, where each edge and node has a time to live in the graph? In the experiments, metrics should cover alert latency, stability of centralities over time, and storage overhead, in addition to accuracy. The implementation of graph construction at stream time is very important to apply graph-based approaches in real-world scenarios.

3. *Privacy & Security in Federated Settings:* To make FL-based NIDS feasible in real-world applications, where even the training workflow is exposed to poisoning, the aggregation process should be robust to Byzantine clients (malicious participants) and associated with explicit privacy guarantees. The models should have resilience against poisoning attacks, using robust aggregation techniques, gradient clipping, and reputation weighting (down-weight anomalous contributors).

4. *Evaluation Beyond Accuracy (Operational & Cost-Aware)* To better reflect the usefulness of the models, evaluation should move beyond accuracy toward time and cost-sensitive metrics. Time-

to-detection needs to be reported alongside traditional precision–recall, with clear definitions and confidence intervals. Practical load should be quantified through false-positive cost curves and analyst-time per incident, raising the question: (i) Which thresholds minimize cost at a fixed recall? and (ii) How stable are alerts across hours and configuration changes? Deployment readiness should be evaluated by memory footprint and energy cost.

5. *Continuous Learning for New and Evolving Attacks* To remain effective under changing behaviors, the detector should adopt continuous learning. To do that, good drift indicators must be used, and complex-networks measures must be explored to detect the drift in graphical information. Another question that may arise is about the level of updates: a low-rank update or full fine-tuning? Also, previous knowledge must not be lost in the process, and the evaluation should quantify adaptation time and stability.

6. *Explainability for Centrality-Augmented Models and GNNs* To increase trust and actionability, explanations should be provided on how the graph information is used and what patterns are being predicted. This may vary from edge-level to node-level and even graph-level patterns. For instance, in the case of graphs created in communication sessions, the whole graph may be analyzed to detect the type of activity in this session. Providing explanations for such cases improves the trustworthiness of the system.

Bibliography

[1] M. U. Farooq, M. Waseem, S. Mazhar, A. Khairi, and T. Kamal, "A review on internet of things (iot)," *International journal of computer applications*, vol. 113, no. 1, 2015.

[2] A. Kumar, K. Abhishek, M. Ghalib, A. Shankar, and X. Cheng, "Intrusion detection and prevention system for an iot environment," *Digital Communications and Networks*, vol. 8, no. 4, pp. 540–551, 2022.

[3] P. K. Sadhu, V. P. Yanambaka, and A. Abdelgawad, "Internet of things: Security and solutions survey," *Sensors*, vol. 22, no. 19, 2022.

[4] I. H. Sarker, A. I. Khan, Y. B. Abushark, and F. Alsolami, "Internet of things (iot) security intelligence: A comprehensive overview, machine learning solutions and research directions," *Mobile Networks and Applications*, vol. 28, pp. 296–312, Feb 2023.

[5] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Comput. Surv.*, vol. 54, oct 2021.

[6] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, p. 102675, 2022.

[7] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[8] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[9] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, "Dl-ids: Extracting features using cnn-lstm hybrid network for intrusion detection system," *Security and Communication Networks*, vol. 2020, p. 8890306, Aug 2020.

[10] S. Ravindran and V. Sarveshwaran, "Deep learning towards intrusion detection system (ids): Applications, challenges and opportunities," *Journal of Mobile Multimedia*, vol. 19, no. 5, pp. 1299–1329, 2023.

[11] Y. Li, Z. Zhou, R. Li, F. Shi, J. Guo, and Q. Liu, "Gogddos: A multi-classifier for ddos attacks using graph neural networks," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1462–1467, 2023.

[12] D. Pujol-Perich, J. Suarez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of graph neural networks for robust intrusion detection," *SIGMETRICS Perform. Eval. Rev.*, vol. 49, p. 111–117, June 2022.

[13] A. E. Attar, A. Wehby, F. Chbib, H. A. Mehrez, A. Fadlallah, J. Hachem, and R. Khatoun, "Analysis of machine learning algorithms for ddos attack detection in connected cars environment," in *2023 Eighth International Conference On Mobile And Secure Services (MobiSecServ)*, vol. CFP23RAC-ART, pp. 1–7, 2023.

[14] A. Fadlallah, H. Sbeity, M. Malli, and P. Lteif, "Application of attack graphs in intrusion detection systems: An implementation," *International Journal of Computer Networks*, vol. 8, pp. 2016–2017, 04 2016.

[15] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating standard feature sets towards increased generalisability and explainability of ml-based network intrusion detection," *Big Data Research*, vol. 30, p. 100359, 2022.

[16] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," 01 2018.

[17] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications: Centralized and federated learning," 2022.

[18] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 1025–1035, Curran Associates Inc., 2017.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[20] Z. Wang, Y. Liu, D. He, and S. Chan, "Intrusion detection methods based on integrated deep learning model," *Computers & Security*, vol. 103, p. 102177, 2021.

[21] M.-e.-A. Brahmia, S. Babouche, S. Ouchani, and M. Zghal, "An adaptive attack prediction framework in cyber-physical systems," in *2022 Ninth International Conference on Software Defined Systems (SDS)*, pp. 1–7, 2022.

[22] A. Heidari and M. A. Jabraeil Jamali, "Internet of things intrusion detection systems: a comprehensive review and future directions," *Cluster Computing*, vol. 26, pp. 3753–3780, Dec 2023.

[23] A. Zohourian, S. Dadkhah, H. Molyneaux, E. C. P. Neto, and A. A. Ghorbani, "Iot-prids: Leveraging packet representations for intrusion detection in iot networks," *Computers & Security*, vol. 146, p. 104034, 2024.

[24] L. Diana, P. Dini, and D. Paolini, "Overview on intrusion detection systems for computers networking security," *Computers*, vol. 14, no. 3, 2025.

[25] M. Gamal, H. M. Abbas, N. Moustafa, E. Sitnikova, and R. A. Sadek, "Few-shot learning for discovering anomalous behaviors in edge networks," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 1823–1837, 2021.

[26] J.-S. Lee, Y.-C. Chen, C.-J. Chew, C.-L. Chen, T.-N. Huynh, and C.-W. Kuo, "Conn-ids: Intrusion detection system based on collaborative neural networks and agile training," *Computers & Security*, vol. 122, p. 102908, 2022.

[27] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019*, 2019.

[28] A. Pinto, L.-C. Herrera, Y. Donoso, and J. A. Gutierrez, "Survey on intrusion detection systems based on machine learning techniques for the protection of critical infrastructure," *Sensors*, vol. 23, no. 5, 2023.

[29] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[30] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks," *Internet of Things*, vol. 16, p. 100462, 2021.

[31] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[32] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, 2015.

[33] S. Layeghy and M. Portmann, "Explainable cross-domain evaluation of ml-based network intrusion detection systems," *Computers and Electrical Engineering*, vol. 108, p. 108692, 2023.

[34] J. Fan and K. Ling-zhi, "Intrusion detection algorithm based on convolutional neural network," *DEStech Transactions on Engineering and Technology Research ICETA*, vol. 37, no. 12, pp. 1271–1275, 2017.

[35] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South African Computer Journal*, vol. 56, 2015.

[36] S. I. Popoola, G. Gui, B. Adebisi, M. Hammoudeh, and H. Gacanin, "Federated deep learning for collaborative intrusion detection in heterogeneous networks," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pp. 1–6, 2021.

[37] M. Zhong, M. Lin, C. Zhang, and Z. Xu, "A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges," *Computers and Security*, vol. 141, p. 103821, 2024.

[38] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," *Internet of Things (Netherlands)*, vol. 7, 2019.

[39] H. Bao and J. Gao, "Network intrusion detection based on improved knn algorithm," *Scientific Reports*, vol. 15, p. 29842, Aug 2025.

[40] M. A. Talukder, M. M. Islam, M. A. Uddin, K. F. Hasan, S. Sharmin, S. A. Alyami, and M. A. Moni, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," *Journal of Big Data*, vol. 11, p. 33, Feb 2024.

[41] P. Waghmode, M. Kanumuri, H. El-Ocla, and T. Boyle, "Intrusion detection system based on machine learning using least square support vector machine," *Scientific Reports*, vol. 15, p. 12066, Apr 2025.

[42] A. El Attar, A. Fadlallah, F. Chbib, and R. Khatoun, "Unsupervised learning algorithms for denial of service detection in vehicular networks," in *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pp. 1–7, 2022.

[43] M. A. Hossain and M. S. Islam, "Ensuring network security with a robust intrusion detection system using ensemble-based machine learning," *Array*, vol. 19, p. 100306, 2023.

[44] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, 2019.

[45] M. T. Abdelaziz, A. Radwan, H. Mamdouh, A. S. Saad, A. S. Abuzaid, A. A. AbdElhakeem, S. Zakzouk, K. Moussa, and M. S. Darweesh, "Enhancing network threat detection with random forest-based nids and permutation feature importance," *Journal of Network and Systems Management*, vol. 33, p. 2, Oct 2024.

[46] C. A. de Souza, C. B. Westphall, R. B. Machado, L. Loffi, C. M. Westphall, and G. A. Geronimo, "Intrusion detection and prevention in fog based iot environments: A systematic literature review," *Computer Networks*, vol. 214, p. 109154, 2022.

[47] N. Mishra and S. Pandya, "Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review," *IEEE Access*, vol. 9, pp. 59353–59377, 2021.

[48] A. J. Mohammed, M. H. Arif, and A. A. Ali, "A multilayer perceptron artificial neural network approach for improving the accuracy of intrusion detection systems," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 4, 2020.

[49] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, P. S. Yu, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1149–1170, 2023.

[50] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," in *Big Data Technologies and Applications* (Z. Deze, H. Huang, R. Hou, S. Rho, and N. Chilamkurti, eds.), (Cham), pp. 117–135, Springer International Publishing, 2021.

[51] M. Vishwakarma and N. Kesswani, "Dids: A deep neural network based real-time intrusion detection system for iot," *Decision Analytics Journal*, vol. 5, p. 100142, 2022.

[52] M. Sarhan, S. Layeghy, and M. Portmann, "Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection," *Big Data Research*, vol. 30, p. 100359, 2022.

[53] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[54] C. D. Luu, V. H. Nguyen, V. Q. Nguyen, and N.-S. Vu, "Novel deep learning-based iot network attack detection using magnet loss optimization," *Internet of Things*, vol. 33, p. 101680, 2025.

[55] I. Al-Turaiki and N. Altwaijry, "A convolutional neural network for improved anomaly-based network intrusion detection," *Big Data*, vol. 9, no. 3, pp. 233–252, 2021.

[56] X. Qiu, L. Shi, and P. Fan, "A cooperative intrusion detection system for internet of things using fuzzy logic and ensemble of convolutional neural networks," *Scientific Reports*, vol. 15, p. 15934, May 2025.

[57] S. Althubiti, W. Nick, J. Mason, X. Yuan, and A. Esterline, "Applying long short-term memory recurrent neural network for intrusion detection," in *SoutheastCon 2018*, pp. 1–5, 2018.

[58] P. Dixit and S. Silakari, "Analysis of state-of-art attack detection methods using recurrent neural network," in *Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences* (M. Dua, A. K. Jain, A. Yadav, N. Kumar, and P. Siarry, eds.), (Singapore), pp. 795–804, Springer Singapore, 2022.

[59] O. Alkadi, N. Moustafa, B. Turnbull, and K.-K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting iot and cloud networks," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9463–9472, 2021.

[60] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Cnn-lstm: Hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, pp. 99837–99849, 2022.

[61] S. N. Mighan and M. Kahani, "A novel scalable intrusion detection system based on deep learning," *International Journal of Information Security*, vol. 20, pp. 387–403, Jun 2021.

[62] G. Olímpio, L. Camargos, R. S. Miani, and E. R. Faria, "Model update for intrusion detection: Analyzing the performance of delayed labeling and active learning strategies," *Computers & Security*, vol. 134, p. 103451, 2023.

[63] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 538–566, 2023.

[64] O. H. Abdulganiyu, T. A. Tchakoucht, and Y. K. Saheed, "Towards an efficient model for network intrusion detection system (ids): systematic literature review," *Wireless Networks*, vol. 30, pp. 453–482, Jan 2024.

[65] B. Fakher, M.-e.-A. Brahmia, I. Bennis, and A. Abouaissa, "Fedcsa: A novel federated learning client selection with anomaly detection approach for iot systems," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2025.

[66] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, vol. 195, pp. 346–361, 2022.

[67] O. Belarbi, T. Spyridopoulos, E. Anthi, I. Mavromatis, P. Carnelli, and A. Khan, "Federated deep learning for intrusion detection in iot networks," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pp. 237–242, 2023.

[68] M. Arbaoui, M.-E.-A. Brahmia, and A. Rahmoun, "Towards secure and reliable aggregation for federated learning protocols in healthcare applications," in *2022 Ninth International Conference on Software Defined Systems (SDS)*, pp. 1–3, 2022.

[69] J.-F. Dollinger, M.-E.-A. Brahmia, and M. Zghal, "Hyperparameter impact on computational efficiency in federated edge learning," in *2024 International Wireless Communications and Mobile Computing*, pp. 849–854, IEEE, 2024.

[70] M. H. Bhavsar, Y. B. Bekele, K. Roy, J. C. Kelly, and D. Limbrick, "Fl-ids: Federated learning-based intrusion detection system using edge devices for transportation iot," *IEEE Access*, vol. 12, pp. 52215–52226, 2024.

[71] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "Iotdefender: A federated transfer learning intrusion detection framework for 5g iot," in *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, pp. 88–95, 2020.

[72] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2021.

[73] J. Wang, K. Yang, and M. Li, "Nids-fgpa: A federated learning network intrusion detection algorithm based on secure aggregation of gradient similarity models," *PLOS ONE*, vol. 19, pp. 1–33, 10 2024.

[74] A. Karunamurthy, K. Vijayan, P. R. Kshirsagar, and K. T. Tan, "An optimal federated learning-based intrusion detection for iot environment," *Scientific Reports*, vol. 15, p. 8696, Mar 2025.

[75] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[76] Z. Ghalmane and B. Daoud, "Graph-based learning for multimodal route recommendation," in *Proceedings of the 10th International Conference on Internet of Things, Big Data and Security - IoTBDS*, pp. 313–320, INSTICC, SciTePress, 2025.

[77] L. Waikhom and R. Patgiri, "A survey of graph neural networks in various learning paradigms: methods, applications, and challenges," *Artificial Intelligence Review*, vol. 56, pp. 6295–6364, Jul 2023.

[78] R. Xu, G. Wu, W. Wang, X. Gao, A. He, and Z. Zhang, "Applying self-supervised learning to network intrusion detection for network flows with graph neural network," *Computer Networks*, vol. 248, p. 110495, 2024.

[79] Q. Xiao, J. Liu, Q. Wang, Z. Jiang, X. Wang, and Y. Yao, "Towards network anomaly detection using graph embedding," in *Computational Science – ICCS 2020* (V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, eds.), (Cham), pp. 156–169, Springer International Publishing, 2020.

[80] Q. Niyaz, W. Sun, A. Javaid, and M. Alam, "E-graphsage: A graph neural network based intrusion detection system for iot," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4598–4608, 2021.

[81] J. Busch, A. Kocheturov, V. Tresp, and T. Seidl, "Nf-gnn: Network flow graph neural networks for malware detection and classification," in *ACM International Conference Proceeding Series*, pp. 121–132, Association for Computing Machinery, 7 2021.

[82] D. H. Tran and M. Park, "Fn-gnn: A novel graph embedding approach for enhancing graph neural networks in network intrusion detection systems," *Applied Sciences (Switzerland)*, vol. 14, 8 2024.

[83] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-e: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, 2022.

[84] A. Chaudhary, H. Mittal, and A. Arora, "Anomaly detection using graph neural networks," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 346–350, 2019.

[85] T. Bilot, N. E. Madhoun, K. A. Agha, and A. Zouaoui, "Graph neural networks for intrusion detection: A survey," *IEEE Access*, vol. 11, pp. 49114–49139, 2023.

[86] M. Newman, *Networks*. Oxford university press, 2018.

[87] J. C. Burguillo, *Complex Networks*, pp. 35–56. Cham: Springer International Publishing, 2018.

[88] S. Boccaletti, P. De Lellis, C. del Genio, K. Alfaro-Bittner, R. Criado, S. Jalan, and M. Romance, "The structure and dynamics of networks with higher order interactions," *Physics Reports*, vol. 1018, pp. 1–64, 2023.

[89] C. Sergiou, M. Lestas, P. Antoniou, C. Liaskos, and A. Pitsillides, "Complex systems: A communication networks perspective towards 6g," *IEEE Access*, vol. 8, pp. 89007–89030, 2020.

[90] H. Hamizeh, E. Mugellini, O. Abou Khaled, and P. Cudré-Mauroux, "Socialmatching++: A novel approach for interlinking user profiles on social networks," *Proceedings of the PROFILES@ ISWC 2017: Sementic web conferene, 22 october 2017, Vienna, Austria*, 2017.

[91] Z. GHALMANE, M. Zghal, *et al.*, "Road networks: A new filtering approach to extract backbones using community structure," 2023.

[92] Z. Ghalmane, M.-E.-A. Brahmia, M. Zghal, and H. Cherifi, "A stochastic approach for extracting community-based backbones," in *Complex Networks and Their Applications XI* (H. Cherifi, R. N. Mantegna, L. M. Rocha, C. Cherifi, and S. Micciche, eds.), (Cham), pp. 55–67, Springer International Publishing, 2023.

[93] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. El Hassouni, "Community-based method for extracting backbones," in *The International Conference on Complex Networks (CompleNet) 2022*, 2022.

[94] Z. Gao, W. Dang, X. Wang, X. Hong, L. Hou, K. Ma, and M. Perc, "Complex networks and deep learning for eeg signal analysis," *Cognitive Neurodynamics*, vol. 15, pp. 369–388, Jun 2021.

[95] Z. Ghalmane, C. Cherifi, H. Cherifi, *et al.*, "Extracting backbones in weighted modular complex networks. scirep 10, 15539 (2020)."

[96] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. El Hassouni, "Author correction: Extracting backbones in weighted modular complex networks," *Scientific Reports*, vol. 11, p. 8957, 2021.

[97] Z. Ghalmane, M. El Hassouni, C. Cherifi, and H. Cherifi, "Influential spreaders in networks with community structure," in *International Conference on Complex Systems (ICCS)*, p. 1, 2020.

[98] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. E. Hassouni, "Centrality in complex networks with overlapping community structure," *Scientific Reports*, vol. 9, 2019.

[99] Z. Ghalmane, M. E. Hassouni, and H. Cherifi, "Betweenness centrality for networks with non-overlapping community structure," in *2018 IEEE Workshop on Complexity in Engineering (COMPENG)*, pp. 1–5, 2018.

[100] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. El Hassouni, "Interactions between overlapping nodes and hubs in complex networks with modular structure," in *Conference on Complex Systems (CCS)*, p. 1, 2020.

[101] Z. Ghalmane, M. El Hassouni, C. Cherifi, and H. Cherifi, "Influential spreaders in modular networks," in *NetSci*, p. 1, 2020.

[102] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. El Hassouni, "Localization of hubs in complex networks with overlapping modular structure," in *NetSci International School and Conference on Network Science*, p. 1, 2020.

[103] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. El Hassouni, "Localization of hubs in modular networks," in *10th International Conference on Complex Systems (ICCS 2020)*, p. 1, 2020.

[104] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. El Hassouni, "Searching for influential nodes in modular networks," in *10th International Conference on Complex Systems (ICCS 2020)*, p. 2, 2020.

[105] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. Hassouni, "Characterizing the relation between hubs and overlapping nodes in modular networks," in *COMPLEX NETWORKS 2019*, 2019.

[106] Z. Ghalmane, M. El Hassouni, C. Cherifi, and H. Cherifi, "Centrality in networks with overlapping communities," in *7th IEEE International Conference on Complex Networks and Their Applications (CNA 2018)*, 2018.

[107] Z. Ghalmane, M. El Hassouni, and H. Cherifi, "Targeted immunization in networks with non-overlapping community structure," in *BOOK OF ABSTRACTS*, p. 104, 2017.

[108] Z. Ghalmane, S. Rajeh, C. Cherifi, H. Cherifi, and M. El Hassouni, "Finding influential nodes in networks with community structure," in *Network Modeling, Learning and Analysis (NMLA), WorldCIST2020 Workshop*, p. 3, 2020.

[109] Z. Ghalmane, C. B. Cherifi, H. Cherifi, and M. El Hassouni, "A backbone extraction method for complex weighted networks," in *Networks, Modèles & Analyse des Réseaux: Approches Mathématiques & Informatiques (MARAMI 2020)*, p. 4, CEUR Workshop Proceedings, 2020.

[110] J. C. Z. Ivan and Burguillo, "Complex systems," *Self-organizing Coalitions for Managing Complexity: Agent-based Simulation of Evolutionary Game Theory Models using Dynamic Social Networks for Interdisciplinary Applications*, pp. 11–34, 2018.

[111] P. Nerurkar, M. Chandane, and S. Bhirud, "A comparative analysis of community detection algorithms on social networks," in *Computational Intelligence: Theories, Applications and Future Directions - Volume I* (N. K. Verma and A. K. Ghosh, eds.), (Singapore), pp. 287–298, Springer Singapore, 2019.

[112] Z. Ghalmane, M. E. Hassouni, and H. Cherifi, "Immunization of networks with non-overlapping community structure," *Social Network Analysis and Mining*, vol. 9, 2019.

[113] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. E. Hassouni, "Extracting modular-based backbones in weighted networks," *Information Sciences*, vol. 576, pp. 454–474, 2021.

[114] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. E. Hassouni, "Exploring hubs and overlapping nodes interactions in modular complex networks," *IEEE Access*, vol. 8, pp. 79650–79683, 2020.

[115] S. Rajeh, M. Savonnet, E. Leclercq, and H. Cherifi, "Interplay between hierarchy and centrality in complex networks," *IEEE Access*, vol. 8, pp. 129717–129742, 2020.

[116] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, p. 046110, Oct 2008.

[117] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *2011 IEEE 27th International Conference on Data Engineering*, pp. 51–62, 2011.

[118] Z. Ghalmane, M. El Hassouni, C. Cherifi, and H. Cherifi, "k-truss decomposition for modular centrality," in *2018 9th International Symposium on Signal, Image, Video and Communications (ISIVC)*, pp. 241–248, 2018.

[119] N. Gupta, A. Singh, and H. Cherifi, "Centrality measures for networks with community structure," *Physica A: Statistical Mechanics and its Applications*, vol. 452, pp. 46–59, 2016.

[120] T. Magelinski, M. Bartulovic, and K. M. Carley, "Measuring node contribution to community structure with modularity vitality," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 707–723, 2021.

[121] M. M. Tulu, R. Hou, and T. Younas, "Identifying influential nodes based on community structure to speed up the dissemination of information in complex network," *IEEE Access*, vol. 6, pp. 7390–7401, 2018.

[122] Ghalmane, Zakariya, El Hassouni, Mohammed, Cherifi, Chantal, and Cherifi, Hocine, "Centrality in modular networks," *EPJ Data Sci.*, vol. 8, no. 1, p. 15, 2019.

[123] Z. Ghalmane, M. El Hassouni, C. Cherifi, and H. Cherifi, "Centrality in modular networks. epj data sci 8 (1): 15," 2019.

[124] Z. Ghalmane, C. Cherifi, H. Cherifi, and M. El Hassouni, "A community-aware backbone extractor for weighted networks," in *9th International Conference on Complex Networks and Their Applications*, pp. p–3, 2020.

[125] T. N. I. Alrumaih and M. J. F. Alenazi, "Cgaad: Centrality- and graph-aware deep-learning model for detecting cyberattacks targeting industrial control systems in critical infrastructure," *IEEE Internet of Things Journal*, vol. 11, no. 13, pp. 24162–24182, 2024.

[126] A. Q. M. Algaolahi, A. A. Hasan, A. Sallam, A. M. Sharaf, A. A. Abdu, and A. A. Alqadi, "Port-scanning attack detection using supervised machine learning classifiers," in *2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, pp. 1–5, 2021.

[127] A. Alharbi and K. Alsubhi, "Botnet detection approach using graph-based machine learning," *IEEE Access*, vol. 9, pp. 99166–99180, 2021.

[128] A. Mustapha, R. Khatoun, S. Zeadally, F. Chbib, A. Fadlallah, W. Fahs, and A. El Attar, "Detecting ddos attacks using adversarial neural network," *Computers & Security*, vol. 127, p. 103117, 2023.

[129] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," *Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018*, 2018.

[130] M. Catillo, A. Pecchia, M. Rak, and U. Villano, "Demystifying the role of public intrusion datasets: A replication study of dos network traffic data," *Computers & Security*, vol. 108, p. 102341, 2021.

[131] K. Ait Rai, M. Machkour, and J. Antari, "Influential nodes identification in complex networks: a comprehensive literature review," *Beni-Suef University Journal of Basic and Applied Sciences*, vol. 12, p. 18, Feb 2023.

[132] O. Ugurlu, "Comparative analysis of centrality measures for identifying critical nodes in complex networks," *Journal of Computational Science*, vol. 62, p. 101738, 2022.

[133] S. Rajeh, M. Savonnet, E. Leclercq, and H. Cherifi, "Comparative evaluation of community-aware centrality measures," *Quality & Quantity*, vol. 57, pp. 1273–1302, Apr 2023.

[134] V. Iswarya, V. Govindasamy, and V. Akila, "A survey on the identification of influential spreaders in complex networks," in *2024 2nd International Conference on Computer, Communication and Control (IC4)*, pp. 1–4, 2024.

[135] M. Termos, Z. Ghalmane, M.-E.-A. Brahmia, A. Fadlallah, A. Jaber, and M. Zghal, "Intrusion detection system for iot based on complex networks and machine learning," in *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence*

and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 0471–0477, 2023.

[136] M. Termos, Z. Ghalmane, M. el Amine Brahmia, A. Fadlallah, A. Jaber, and M. Zghal, "Gdlc: A new graph deep learning framework based on centrality measures for intrusion detection in iot networks," *Internet of Things*, vol. 26, p. 101214, 2024.

[137] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, "Iot network intrusion dataset," 2019.

[138] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022.

[139] N. Moustafa, "Ton iot datasets," 2019.

[140] R. Vega, L. Flores, and R. Greiner, "Simlr: Machine learning inside the sir model for covid-19 forecasting," *Forecasting*, vol. 4, no. 1, pp. 72–94, 2022.

[141] M. Arbaoui, M.-E.-A. Brahmia, and A. Rahmoun, "A review of iot architectures in smart healthcare applications," in *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 1–8, 2022.

[142] R. Zhao, Y. Wang, Z. Xue, T. Ohtsuki, B. Adebisi, and G. Gui, "Semisupervised federated-learning-based intrusion detection method for internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8645–8657, 2023.

[143] G. de Carvalho Bertoli, L. Alves Pereira Junior, O. Saotome, and A. L. dos Santos, "Generalizing intrusion detection for heterogeneous networks: A stacked-unsupervised federated learning approach," *Computers & Security*, vol. 127, p. 103106, 2023.

[144] F. Al Tfaily, Z. Ghalmane, M. e. A. Brahmia, H. Hazimeh, A. Jaber, and M. Zghal, "Graph-based federated learning approach for intrusion detection in iot networks," *Scientific Reports*, vol. 15, no. 1, p. 41264, 2025.

[145] M. Arbaoui, M.-E.-A. Brahmia, A. Rahmoun, and M. Zghal, "Optimizing shapley value for client valuation in federated learning through enhanced gtg-shapley," in *2024 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 1528–1533, 2024.

[146] M. Termos, Z. Ghalmane, M.-e.-A. Brahmia, A. Fadlallah, A. Jaber, and M. Zghal, "Integrating centrality measures in federated learning-based intrusion detection systems," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2025.

[147] J.-M. Shao, G.-Q. Zeng, K.-D. Lu, G.-G. Geng, and J. Weng, "Automated federated learning for intrusion detection of industrial control systems based on evolutionary neural architecture search," *Computers & Security*, vol. 143, p. 103910, 2024.

[148] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138509–138542, 2021.

[149] M. Arbaoui, M.-e.-A. Brahmia, A. Rahmoun, and M. Zghal, "Federated learning survey: A multi-level taxonomy of aggregation techniques, experimental insights, and future frontiers," *ACM Trans. Intell. Syst. Technol.*, vol. 15, Nov. 2024.

[150] A.-R.-E. M. Baahmed, J.-F. Dollinger, M.-E.-A. Brahmia, and M. Zghal, "Hyperparameter impact on computational efficiency in federated edge learning," in *2024 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 0849–0854, 2024.

[151] Z. Du, J. Sun, A. Li, P.-Y. Chen, J. Zhang, H. H. Li, and Y. Chen, "Rethinking normalization methods in federated learning," in *Proceedings of the 3rd International Workshop on Distributed Machine Learning*, DistributedML '22, (New York, NY, USA), p. 16–22, Association for Computing Machinery, 2022.

[152] T. Kobayashi and M. Génois, "The switching mechanisms of social network densification," *Scientific Reports*, vol. 11, p. 3160, Feb 2021.

[153] S. Kojaku, F. Radicchi, Y.-Y. Ahn, and S. Fortunato, "Network community detection via neural embeddings," *Nature Communications*, vol. 15, p. 9446, Nov 2024.

[154] S. Shvydun, "Centrality in complex networks under incomplete data," *PLOS Complex Systems*, vol. 2, pp. 1–22, 05 2025.

[155] N. Xiang, Q. Wang, and M. You, "Estimation and update of betweenness centrality with progressive algorithm and shortest paths approximation," *Scientific Reports*, vol. 13, p. 17110, Oct 2023.

[156] M. Charikhi, "Association of the pagerank algorithm with similarity-based methods for link prediction in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 637, p. 129552, 2024.

[157] H. Chen, A. Conte, R. Grossi, G. Loukides, S. P. Pissis, and M. Sweering, "On breaking truss-based and core-based communities," *ACM Trans. Knowl. Discov. Data*, vol. 18, Apr. 2024.

[158] S. Rajeh, M. Savonnet, E. Leclercq, and H. Cherifi, "Characterizing the interactions between classical and community-aware centrality measures in complex networks," *Scientific Reports*, vol. 11, p. 10088, May 2021.

[159] A. Grammenos, R. Mendoza Smith, J. Crowcroft, and C. Mascolo, "Federated principal component analysis," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 6453–6464, Curran Associates, Inc., 2020.

[160] P. L. S. Jayalaxmi, R. Saha, G. Kumar, M. Conti, and T.-H. Kim, "Machine and deep learning solutions for intrusion detection and prevention in iots: A survey," *IEEE Access*, vol. 10, pp. 121173–121192, 2022.

[161] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[162] M. Ramzan, M. Shoaib, A. Altaf, S. Arshad, F. Iqbal, A. K. Castilla, and I. Ashraf, "Distributed denial of service attack detection in network traffic using deep learning algorithm," *Sensors*, vol. 23, no. 20, 2023.

[163] B. Khemani, S. Patil, K. Kotecha, and S. Tanwar, "A review of graph neural networks: concepts,

architectures, techniques, challenges, datasets, applications, and future directions," *Journal of Big Data*, vol. 11, p. 18, Jan 2024.

[164] L. Wu, P. Cui, J. Pei, and L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore, 2022.

[165] S. A. Abdulkareem, C. Heng Foh, M. Shojafar, F. Carrez, and K. Moessner, "Network intrusion detection: An iot and non iot-related survey," *IEEE Access*, vol. 12, pp. 147167–147191, 2024.

[166] T. Sowmya and E. Mary Anita, "A comprehensive review of ai based intrusion detection system," *Measurement: Sensors*, vol. 28, p. 100827, 2023.

[167] B. Zhang, J. Li, C. Chen, K. Lee, and I. Lee, "A practical botnet traffic detection system using gnn," in *Cyberspace Safety and Security* (W. Meng and M. Conti, eds.), (Cham), pp. 66–78, Springer International Publishing, 2022.

[168] G. Gkarmpounis, C. Vranis, N. Vretos, and P. Daras, "Survey on graph neural networks," *IEEE Access*, vol. 12, pp. 128816–128832, 2024.

[169] F. Liang, C. Qian, W. Yu, D. Griffith, and N. Golmie, "Survey of graph neural networks and applications," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 9261537, 2022.

[170] X. Liu, M. Yan, L. Deng, G. Li, X. Ye, and D. Fan, "Sampling methods for efficient training of graph convolutional networks: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 2, pp. 205–234, 2022.

[171] M. Termos, Z. Ghalmane, M. Brahmia, A. Fadlallah, A. Jaber, and M. Zghal, "Enhancing iot network intrusion detection with a new graphsage embedding algorithm using centrality measures," in *Proceedings of the 10th International Conference on Internet of Things, Big Data and Security - IoTBDS*, pp. 329–336, INSTICC, SciTePress, 2025.

[172] D. Ellerman, *New foundations for information theory: logical entropy and Shannon entropy*. Springer Nature, 2021.

[173] F. Al Tfaily, Z. Ghalmane, M. Termos, M.-e.-A. Brahmia, A. Jaber, and M. Zghal, "Generating realistic cyber security datasets for iot networks with diverse complex network properties," in

Proceedings of the 10th International Conference on Internet of Things, Big Data and Security - IoTBDS, 2025.

[174] Z. Liu, N. Thapa, A. Shaver, K. Roy, M. Siddula, X. Yuan, and A. Yu, "Using embedded feature selection and CNN for classification on CCD-INID-V1-A new IoT dataset," *Sensors (Basel)*, vol. 21, July 2021.

[175] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3962–3977, 2022.

[176] D. Ono, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "A proposal of port scan detection method based on packet-in messages in openflow networks and its evaluation," *International Journal of Network Management*, vol. 31, no. 6, p. e2174, 2021.

Mortada TERMOS

Intrusion Detection in IoT Networks Using Complex Networks and AI Techniques

Résumé : Les systèmes de détection d'intrusion réseau (NIDS) font face à des défis majeurs dans les environnements de l'Internet des objets (IoT) en raison de leur échelle, de leur hétérogénéité et de ressources limitées. De plus, de nombreuses approches considèrent le trafic comme des enregistrements indépendants, en négligeant la structure du réseau. Cette thèse modélise les flux de trafic sous forme de graphes, où les nœuds représentent les dispositifs et les arêtes les communications entre eux. Nous intégrons ces caractéristiques dans des modèles d'apprentissage profond via une chaîne de traitement Graph Deep Learning basée sur les mesures de centralité (GDLC), étendons l'approche à l'apprentissage fédéré afin d'améliorer la confidentialité et la généralisation inter-sites, et concevons des réseaux de neurones sur graphes centrés sur les arêtes, adaptés aux propriétés du réseau. Des expérimentations menées sur des jeux de données de référence montrent que les caractéristiques basées sur la centralité améliorent la précision de détection, la confiance et la capacité de généralisation entre sites, démontrant ainsi l'efficacité des NIDS basés sur les graphes pour l'IoT.

Mots clés : Cybersécurité, Systèmes de détection d'intrusions réseau, Internet des Objets, Réseaux complexes, Apprentissage fédéré.

Abstract : Network Intrusion Detection Systems (NIDS) face significant challenges in Internet-of-Things (IoT) environments due to scale, heterogeneity, and limited resources. Moreover, many approaches treat traffic as independent records, neglecting network structure. This thesis models traffic flows as graphs, where nodes represent devices and edges represent the communications between them. We integrate these features into deep learning models through a Graph Deep Learning based on Centrality (GDLC) pipeline, extend the approach to federated learning to enhance privacy and cross-site generalization, and design edge-centric graph neural networks tailored to network properties. Experiments on standard benchmarks show that centrality-based features consistently improve detection accuracy, confidence, and generalization across sites, demonstrating the effectiveness of graph-based NIDS for IoT.

Keywords : Cybersecurity, Network Intrusion Detection Systems, Internet of Things, Complex Networks, Federated Learning.