# A NOVEL MECHANISM TO FORTIFY APPLICATION LAYER AGAINST DISTRIBUTED DENIAL OF SERVICE (DDOS) ATTACK

**A Thesis submitted to Gujarat Technological University**

for the Award of

**Doctor of Philosophy**

in

**Computer/IT Engineering**

by

**Dhangar Dharmeshkumar Natthuram**
Enrollment No. 219999913039

under the supervision of

**Dr. Vikram M. Agrawal**

**GUJARAT TECHNOLOGICAL UNIVERSITY**
**AHMEDABAD**

**SEPTEMBER – 2026**

# A NOVEL MECHANISM TO FORTIFY APPLICATION LAYER AGAINST DISTRIBUTED DENIAL OF SERVICE (DDOS) ATTACK

**A Thesis submitted to Gujarat Technological University**

for the Award of

**Doctor of Philosophy**

in

**Computer/IT Engineering**

by

**Dhangar Dharmeshkumar Natthuram**
Enrollment No. 219999913039

under the supervision of

**Dr. Vikram M. Agrawal**

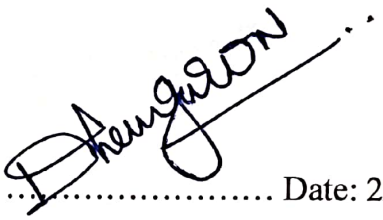
**GUJARAT TECHNOLOGICAL UNIVERSITY**
**AHMEDABAD**

**SEPTEMBER – 2026**

# DECLARATION

I declare that the thesis entitled **"A Novel Mechanism to Fortify Application Layer Against Distributed Denial of Service (DDoS) Attack"** submitted by me for the degree of Doctor of Philosophy is the record of research work carried out by me during the period from January 2021 to September 2026 under the supervision of **Dr. Vikram M. Agrawal** and this has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other University or other institution of higher learning.

I further declare that the material obtained from other sources has been duly acknowledged in the thesis. I shall be solely responsible for any plagiarism or other irregularities, if noticed in the thesis.



Signature of the Research Scholar: ..... Date: 22/09/2026

Name of Research Scholar: Dhangar Dharmeshkumar Natthuram

Place: Surat, Gujarat.

# CERTIFICATE

I certify that the work incorporated in the thesis **"A Novel Mechanism to Fortify Application Layer Against Distributed Denial of Service (DDoS) Attack"** submitted by Mr. Dhangar Dharmeshkumar Natthuram was carried out by the candidate under my supervision/guidance. To the best of my knowledge: (i) the candidate has not submitted the same research work to any other institution for any degree/diploma, Associateship, Fellowship or other similar titles (ii) the thesis submitted is a record of original research work done by the Research Scholar during the period of study under my supervision, and (iii) the thesis represents independent research work on the part of the Research Scholar.

Signature of the Research Supervisor: ..... Date: 22/09/2026

Name of Supervisor: Dr. Vikram M. Agrawal

Place: Anand, Gujarat.

# Course-work Completion Certificate

This is to certify that Mr. Dhangar Dharmeshkumar Natthuram enrollment number 219999913039, is enrolled for PhD program in the faculty Computer/IT Engineering of Gujarat Technological University, Ahmedabad.

(Please tick the relevant option(s))

☐ He / She has been exempted from the course-work (successfully completed during M.Phil Course)

☐ He / She has been exempted from Research Methodology Course only (successfully completed during M.Phil Course)

☑ He has successfully completed the PhD course work for the partial requirement for the award of PhD Degree. His performance in the course work is as follows-

| Grade Obtained in Research Methodology [PHD2001] | Grade Obtained in Research and Publication Ethics [PHD2002] | Grade Obtained in Self-Study Course [PHD2003] |
|---|---|---|
| **BC** | **BB** | **AB** |

V. M. Agrawal

Supervisor's Sign.

(Dr. Vikram M. Agrawal)

# Originality Report Certificate

It is certified that the PhD Thesis titled **"A Novel Mechanism to Fortify Application Layer Against Distributed Denial of Service (DDoS) Attack"** by Mr. Dhangar Dharmeshkumar Natthuram has been examined by us. We undertake the following:

a. Thesis has significant new work / knowledge as compared already published or are under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless itis placed under quotation marks and duly referenced.

b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.

c. There is no fabrication of data or results which have been compiled / analyzed.

d. There is no falsification by manipulating research materials, equipment or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.

e. The thesis has been checked using **Turnitin** plagiarism check software (copy of originality report attached) and found within limits as per GTU Plagiarism Policy and instructions issued from time to time (i.e. permitted similarity index <=10%).

Signature of the Research Scholar: .....Date: 22/09/2026

Name of the Research Scholar: Dhangar Dharmeshkumar Natthuram

Signature of the Research Supervisor: .....Date:22/09/2026

Name of the Research Supervisor: Dr. Vikram M. Agrawal

Place: Anand, Gujarat.

harmesh Dhangar

# NOVEL MECHANISM TO FORTIFY APPLICATION LAYE GAINST DISTRIBUTED DENIAL OF SERVICE (DDOS) AT

Quick Submit

Quick Submit

Gujarat Technological University

## ument Details

ssion ID

f:::1:3618640265

ssion Date

2026, 11:15 AM GMT+5:30

oad Date

2026, 11:19 AM GMT+5:30

me

ar_Dharmesh_Natthuram_219999913039_thesis.pdf

e

111 Pages

37,730 Words

211,949 Characters

V. M Ag

# 5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Match Groups

- 101 Not Cited or Quoted 3%
  Matches with neither in-text citation nor quotation marks
- 5 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

## Top Sources

| | |
|---|---|
| 3% | Internet sources |
| 0% | Publications |
| 0% | Submitted works (Stud |

## Integrity Flags

Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deep
would set it apart from a normal su
it for you to review.

A Flag is not necessarily an indicato
focus your attention there for furth

V. M Agl

# Ph.D. Thesis Non-Exclusive License to GUJARAT TECHNOLOGICAL UNIVERSITY

In consideration of being a Research Scholar at Gujarat Technological University, and in the interests of the facilitation of research at the University and elsewhere, I, Dhangar Dharmeshkumar Natthuram having Enrollment No. 219999913039, hereby grant a non-exclusive, royalty free and perpetual license to the University on the following terms:

a. The University is permitted to archive, reproduce and distribute my thesis, in whole or in part, and/or my abstract, in whole or in part (referred to collectively as the "Work") anywhere in the world, for non-commercial purposes, in all forms of media;
b. The University is permitted to authorize, sub-lease, sub-contract or procure any of the acts mentioned in paragraph (a);
c. The University is authorized to submit the Work at any National / International Library, under the authority of their "Thesis Non-Exclusive License";
d. The Universal Copyright Notice (©) shall appear on all copies made under the authority of this license;
e. I undertake to submit my thesis, through my University, to any Library and Archives. Any abstract submitted with the thesis will be considered to form part of the thesis.
f. I represent that my thesis is my original work, does not infringe any rights of others, including privacy rights, and that I have the right to make the grant conferred by this non-exclusive license.
g. If third party copyrighted material was included in my thesis for which, under the terms of the Copyright Act, written permission from the copyright owners is required, I have obtained such permission from the copyright owners to do the acts mentioned in paragraph (a) above for the full term of copyright protection.
h. I understand that the responsibility for the matter as mentioned in the paragraph (g) rests with the authors / me solely. In no case shall GTU have any liability for any acts / omissions / errors / copyright infringement from the publication of the said thesis or otherwise.

i. I retain copyright ownership and moral rights in my thesis, and may deal with the copyright in my thesis, in any way consistent with rights granted by me to my University in this non-exclusive license.

j. GTU logo shall not be used /printed in the book (in any manner whatsoever) being published or any promotional or marketing materials or any such similar documents.

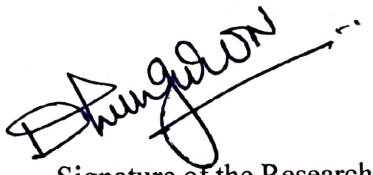
k. The following statement shall be included appropriately and displayed prominently in the book or any material being published anywhere: "The content of the published work is part of the thesis submitted in partial fulfilment for the award of the degree of Ph.D. in **Computer/IT Engineering** of the Gujarat Technological University".

l. I further promise to inform any person to whom I may hereafter assign or license my copyright in my thesis of the rights granted by me to my University in this nonexclusive license. I shall keep GTU indemnified from any and all claims from the Publisher(s) or any third parties at all times resulting or arising from the publishing or use or intended use of the book / such similar document or its contents.

m. I am aware of and agree to accept the conditions and regulations of Ph.D. including all policy matters related to authorship and plagiarism.

Date: 22/09/2026

Place: Surat.



Signature of the Research Scholar

Recommendation of the Research Supervisor: Recommended

Signature of the Research Supervisor

# Thesis Approval Form

The viva-voce of the PhD Thesis submitted by **Shri. Dhangar Dharmeshkumar Natthuram, Enrollment No. 219999913039** entitled **"A Novel Mechanism to Fortify Application Layer Against Distributed Denial of Service (DDoS) Attack"** was conducted on **Tuesday, 22/09/2026** at Gujarat Technological University.

(Please tick any one of the following options)

☑ The performance of the candidate was satisfactory. We recommend that he be awarded the PhD degree.

☐ Any further modifications in research work recommended by the panel after 3 months from the date of first viva-voce upon request of the Supervisor or request of Independent Research Scholar after which viva-voce can be re-conducted by the same panel again.

☐ The performance of the candidate was unsatisfactory. We recommend that he/she should not be awarded the PhD degree.

V. M. Agrawal

Dr. Vikram M. Agrawal

Name and Signature of Supervisor with Seal

Dr. Digvijaysinh Parmar

1) (External Examiner-1)

Dr. Amitkumar Jaydevbhai Nayak

2) (External Examiner-2)

# ABSTRACT

Distributed Denial-of-Service (DDoS) attacks remain a major cybersecurity challenge for enterprise networks, cloud platforms, and IoT environments. Traditional intrusion detection approaches often rely on static feature selection techniques, limiting their ability to identify evolving attack patterns and complex traffic behavior. To address this issue, this research proposes a Transformer-Based Attention CNN-BiLSTM framework for intelligent DDoS attack detection.

The proposed model combines a Transformer attention mechanism for dynamic feature weighting, CNN for spatial feature extraction, and BiLSTM for temporal dependency learning. This integrated architecture enables effective representation of network traffic characteristics and improves the distinction between benign and malicious flows.

The framework was evaluated on five benchmark datasets, namely CICDDoS2019, BCCC-cPacket-Cloud-DDoS-2024, CICIOT2023, ITDDoS2020, and APA-DDoS, covering enterprise, cloud, IoT, and advanced network environments. Experimental results demonstrate superior performance compared with conventional feature selection methods such as Mutual Information, ANOVA, and RFE. The proposed model achieved 98.74% accuracy, 98.70% F1-score, 0.9997 ROC-AUC, and 0.9842 MCC on CICDDoS2019, while cross-dataset validation produced accuracy ranging from 99.88% to 100%.

The results confirm that dynamic attention-based feature learning significantly enhances DDoS detection performance. The proposed framework demonstrates strong robustness, scalability, and generalization capability, making it a practical solution for real-time protection against modern DDoS attacks.

***Keywords:*** DDoS Detection, Transformer Attention, CNN, BiLSTM, Deep Learning, Intrusion Detection System, Cybersecurity, Cloud Security, IoT Security.

# ACKNOWLEDGEMENT

This PhD journey has been possible only due to the guidance, support, and blessings of many people to whom I am deeply indebted.

First and foremost, I express my sincere gratitude to my research supervisor, **Dr. Vikram M. Agrawal**, Assistant Professor, BBIT, Vallabhvidhyanagar, Anand, Gujarat, for his continuous guidance, insightful suggestions, and encouragement throughout the course of my doctoral studies. His expertise, patience, and constructive feedback have shaped this work at every stage and greatly contributed to its completion.

I extend my heartfelt thanks to the members of my Doctoral Progress Committee, **Dr. Ravi Sheth**, Assistant Professor, IT Department, Rashtriya Raksha University, Gandhinagar, Gujarat, and **Dr. Premal Patel,** Principal, IPCOWALA Institute of Technology, Dharmaj, Anand, Gujarat. Their valuable comments, technical inputs, and periodic reviews provided clear direction for this research and helped in refining the quality of the thesis.

I am grateful to **Mrs. Bindu Goyal, Principal; Mr. Manish D. Patel, Head of the IT Department;** and **Mr. Dhaval R. Gandhi**, Lecturer, IT Department, Dr. S. & S.S. Ghandhy College of Engineering & Technology, Surat, for their constant support, encouragement, and for providing a conducive academic environment and the necessary facilities for my research work. Their cooperation and motivation have been instrumental in successfully carrying out this study.

On a personal note, I owe my deepest thanks to my family. I bow with respect and love to my mother, Minaxi Dhangar, whose blessings have been the foundation of all my achievements. I am profoundly thankful to my wife, Harsha, for her constant support during the demanding phases of this research work. My children, Prayag and Vamika, have been a continuous source of joy and inspiration, reminding me of the purpose behind my efforts. I also thank my younger brother, Hemanshu Dhangar, for his support and encouragement throughout this journey.

To all colleagues, friends, and well-wishers, I sincerely thank them for their invaluable support and encouragement throughout this journey.

Finally, I remain grateful to the Almighty for giving me the strength, perseverance, and opportunity to complete this thesis.

**Dhangar Dharmeshkumar Natthuram**

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| Abbreviation | Full Form |
|---|---|
| ANOVA | Analysis of Variance |
| API | Application Programming Interface |
| BiLSTM | Bidirectional Long Short-Term Memory |
| CNN | Convolutional Neural Network |
| DDoS | Distributed Denial of Service |
| DoS | Denial of Service |
| FL | Federated Learning |
| FN & FP | False Negative & False Positive |
| FNR | False Negative Rate |
| FPR | False Positive Rate |
| GA | Genetic Algorithm |
| GAN | Generative Adversarial Network |
| GELU | Gaussian Error Linear Unit |
| GRU | Gated Recurrent Unit |
| GTU | Gujarat Technological University |
| HTTP | Hypertext Transfer Protocol |
| HTTP/3 | Hypertext Transfer Protocol, version 3 |
| IDS | Intrusion Detection System |
| IoT | Internet of Things |
| LSTM | Long Short-Term Memory |
| MCC | Matthews Correlation Coefficient |
| MI | Mutual Information |
| PCA | Principal Component Analysis |
| QUIC | Quick UDP Internet Connections |
| ReLU | Rectified Linear Unit |
| RFE | Recursive Feature Elimination |
| RNN | Recurrent Neural Network |
| ROC-AUC | Area Under the Receiver Operating Characteristic Curve |
| SDN | Software-Defined Networking |
| TCP | Transmission Control Protocol |
| TN | True Negative |
| TNR | True Negative Rate (Specificity) |
| TP | True Positive |
| TPR | True Positive Rate (Recall/Sensitivity) |
| UDP | User Datagram Protocol |

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF APPENDICES

# CHAPTER 1: INTRODUCTION

## 1.1 Background and Motivation

Distributed Denial of Service (DDoS) attacks have remained one of the most persistent threats to the availability of networked services for more than two decades, but the character of these attacks has changed considerably. Early DDoS campaigns were overwhelmingly volumetric: an attacker, typically operating a botnet of compromised hosts, would flood a target's network link with a sheer quantity of packets — UDP floods, ICMP floods, and SYN floods being the archetypal examples — with the explicit goal of exhausting bandwidth or connection-table resources before the traffic ever reached the application itself. Because volumetric floods differ from ordinary traffic in gross statistical terms (packet rate, byte rate, protocol distribution), they can often be recognized using comparatively simple signature or threshold-based mechanisms deployed at the network or transport layer.

Over the past several years, however, attackers have increasingly shifted their effort toward the application layer, or Layer 7 (L7), of the protocol stack. Application-layer denial-of-service attacks do not necessarily require enormous traffic volumes; instead, they exploit the fact that certain application-level requests — a complex database query, a large file download, a repeated login attempt, a slow but persistent HTTP connection — are disproportionately expensive for a server to service relative to the effort required by the client to generate them. Tripathi and Hubballi's survey of the field frames this shift precisely: because L7 attacks are engineered around the design of the protocol or application rather than around raw packet volume, they are inherently stealthier than flooding-based attacks and can be sustained with only a modest number of attacking sources [1]. Their contribution is chiefly taxonomic rather than algorithmic, however — the survey organizes and characterizes existing attack and defense literature but does not itself propose or benchmark a new detection model, leaving open exactly how a concrete detector should be built to exploit the distinctions it draws. A single slow HTTP connection, a carefully paced sequence of GET requests, or a small number of expensive search queries can degrade or disable a service in a way that never triggers a bandwidth or packet-rate alarm.

This is precisely what makes application-layer DDoS attacks substantially harder to detect than their volumetric counterparts. A volumetric flood is anomalous almost by

definition — the sheer scale of traffic is itself the signature. An application-layer attack, by contrast, is deliberately constructed to resemble legitimate use: the packets are well-formed, the sessions are established through a proper handshake, and the request rate per attacking client may be no higher than that of a genuine, if unusually active, user. Black and Kim's review of this attack category catalogues HTTP-flooding tools such as HOIC alongside connection-starving techniques such as Slowloris, and its central observation is that both families are engineered to pass unnoticed at the network and transport layers while still tying up server-side resources — open connections, worker threads, database handles — that ordinary traffic monitoring does not track [2]. Because their treatment is descriptive and survey-oriented rather than experimental, it documents what these attacks exploit without offering a validated detection algorithm of its own, which is part of the gap this thesis aims to close. Detection therefore cannot rely on packet-level statistics alone; it must reason about behavior — the sequencing, timing, and semantics of requests — over an extended observation window, and it must do so while continuing to distinguish a coordinated attack from an equally plausible, entirely benign flash crowd of real users arriving at once.

A further complication is that application-layer attacks are not a single, homogeneous phenomenon. They span high-rate floods directed at expensive endpoints, low-rate "slow" attacks that starve connection pools over long timescales, and flash-crowd-mimicking attacks that borrow the arrival statistics of a legitimate traffic surge. Sharif, Beitollahi, and Fazeli's study of freely available attack toolkits shows that the practical accessibility of such tools has itself become part of the threat landscape, lowering the technical barrier for an attacker to launch a credible-looking application-layer campaign and thereby widening the variety of traffic patterns that a detector must generalize across [3]. Their proposed classifier reports very high accuracy, but it is trained and evaluated against the specific toolkits examined in that study; the authors themselves frame the contribution around the tool-availability problem rather than around robustness to toolkits not represented in their evaluation, so a detector built the same way would need re-validation before being trusted against an unseen or custom-built attack script. A detection mechanism tuned narrowly to one such pattern, such as a single toolkit's request cadence, tends to degrade sharply when confronted with a different flavor of L7 attack, which is one reason the field has continued to move toward

learning-based approaches capable of generalizing across attack families rather than relying on attack-specific signatures.

The difficulty of detection is compounded by the fact that application-layer attacks do not form a single category with one distinguishing signature. Deressu, Beyene, and Gemechu's recent survey groups the reported attack behaviors in this space into three broad families: high-rate attacks, in which a large number of otherwise legitimate-looking requests are directed at an expensive endpoint in a short window; low-rate attacks, in which a small number of carefully paced requests keep server-side resources such as connections or worker threads occupied over an extended period; and flash-crowd-mimicking attacks, in which the arrival statistics of the attack traffic are deliberately engineered to resemble a genuine surge of interest in a service [5]. Each of these families stresses a detection mechanism in a different way — a high-rate attack must be told apart from a popular but legitimate event, a low-rate attack must be noticed despite producing very little traffic at any given instant, and a flash-crowd-mimicking attack must be distinguished using cues subtler than arrival rate alone. A defense mechanism intended for practical deployment at the application layer must therefore be evaluated not against a single canonical attack pattern but against this full spectrum of behaviors, a requirement that recurs throughout the design choices made later in this thesis. It is also worth noting that this taxonomy is itself a secondary synthesis — Deressu, Beyene, and Gemechu compiled it from 47 primary studies rather than from new experiments of their own — so its value here lies in the organizing structure it provides rather than in any single result being independently verified by this thesis.

It is this combination of factors — legitimacy-mimicking traffic, heterogeneous attack behavior, and a comparatively low volume threshold for causing real damage — that motivates the present thesis. Rather than treating DDoS defense purely as a network-layer capacity problem, this work is concerned with fortifying the application layer itself: recognizing malicious intent from the pattern of requests a server receives, even when those requests are, packet for packet, indistinguishable from a genuine user's.

## 1.2 Cloud, IoT, and Enterprise Threat Landscape

The urgency of defending the application layer is reinforced by how the DDoS threat landscape has evolved across cloud, Internet of Things (IoT), and enterprise environments over the 2021–2026 period. Singh and Jain's survey of DDoS detection

and mitigation in SDN-IoT networks compiles a set of figures from major security vendors that together sketch a threat surface that has been growing in both scale and sophistication rather than leveling off. According to industry telemetry cited in their survey, DDoS activity rose by roughly 23% in the first quarter of 2021 relative to the same period a year earlier, with the financial services sector alone accounting for around a quarter of all recorded attacks in that quarter — a reminder that enterprise-facing services handling high-value transactions remain a preferred target [4]. The same survey reports that between 2021 and 2022 the typical magnitude of an attack climbed from roughly 3.3 Gbps to 4.3 Gbps, an increase of close to 32%, while the proportion of attacks combining multiple vectors in a single campaign grew by close to 29% over the same period [4]. Multi-vector campaigns are particularly relevant to this thesis because they routinely pair a volumetric component with an application-layer component, so that even where the network-layer flood is absorbed by upstream scrubbing, a residual application-layer attack can still reach the origin server. It bears noting that these figures are vendor telemetry reproduced secondhand within an academic survey rather than measurements Singh and Jain collected or independently audited themselves, so the exact monitoring methodology and traffic sampling behind each number is not fully transparent from the survey alone; they are used here as an indicator of direction and order of magnitude rather than as precise, independently reproducible statistics.

The IoT segment of the landscape has expanded the attack surface still further. The same compiled industry data indicates that DDoS activity directed specifically at IoT devices grew by approximately 62% in 2022 compared with the prior year [4], a trend consistent with the continued proliferation of under-secured, resource-constrained endpoints that are readily recruited into botnets. The cloud segment, meanwhile, illustrates just how far the ceiling on attack scale has moved: the largest publicly disclosed DDoS event to date, observed by major cloud infrastructure providers in October 2023 and traced to exploitation of a previously unknown protocol-level vulnerability, is reported to have reached a peak of roughly 398 million requests per second [4]. A request-based metric of this kind is itself telling, since a request-per-second figure describes load at the application layer rather than raw bandwidth, underscoring that even attacks of extraordinary scale are increasingly measured and felt in terms of application-level demand rather than link saturation alone.

Beyond these headline figures, the broader research literature describes a qualitative shift in how detection research itself has responded to the changing threat. Deressu, Beyene, and Gemechu's recent survey of application-layer DDoS detection, reviewing several dozen primary studies published since 2018, finds that machine-learning-based techniques and hybrid approaches together now account for the clear majority of proposed detection methods, with deep-learning-based methods representing a substantial and continuing share of that body of work, while purely rule-based techniques have receded to a comparatively small minority of recent contributions [5]. This pattern is corroborated from the IoT side of the literature: Anley, Genovese, and Piuri's survey of deep learning for DDoS detection in IoT settings documents a steady stream of CNN-, LSTM-, and hybrid CNN-LSTM-based detectors proposed specifically to cope with the traffic heterogeneity and resource constraints of IoT networks [6], while Singh and Jain's SDN-IoT survey similarly frames DDoS defense in these environments as increasingly reliant on adaptive, learning-driven mechanisms rather than static rule sets, precisely because static rules cannot keep pace with botnets whose composition and behavior change from one campaign to the next [4]. These three sources agree on direction but each reports adoption counts or proportions drawn from the studies each surveyed, not a shared, independently benchmarked comparison of detection accuracy; a rising share of papers using deep learning is evidence of research momentum, not proof that every such method outperforms the classical alternatives it is displacing.

Cloud environments introduce an additional wrinkle that the enterprise and IoT contexts do not share as acutely: elasticity itself can mask an attack. Cloud infrastructure is designed to scale transparently in response to demand, which means that a slowly escalating application-layer attack can, for a period, appear indistinguishable from organic growth in legitimate load, delaying both detection and any decision to intervene. Kirubavathi et al.'s work on transformer-based detection of TCP-based DDoS attacks in cloud settings is explicit about this difficulty, noting that attacks in cloud infrastructures are especially prone to imitating genuine traffic patterns, which is precisely why standard, coarser-grained approaches struggle to separate them from legitimate cloud workloads [7]. Their own evaluation, however, is confined to TCP-based attacks within a single simulated cloud-traffic dataset, so the claim that elasticity complicates detection is well motivated by their argument but has not, in their

study or in this thesis, been demonstrated across the full range of cloud providers, protocols, or attack types that a production deployment would eventually face. Taken together, the cloud, IoT, and enterprise landscapes point to a common conclusion: the scale of the largest attacks continues to grow, the population of exploitable endpoints continues to expand, and the traffic generated by attackers continues to resemble legitimate demand ever more closely — all of which push the locus of effective defense toward the application layer and toward detection techniques that can model behavior rather than merely counting packets.

For enterprise operators specifically, the consequence of this growth in scale and stealth is not simply a technical inconvenience but a direct commercial and contractual exposure. Modern enterprises increasingly deliver their services through cloud-hosted, API-driven, and microservice-based architectures whose components are bound by service-level agreements governing latency and availability; an application-layer attack that need not saturate any link but only exhaust a handful of backend resources is well positioned to breach exactly these guarantees while leaving upstream network monitoring largely undisturbed. Although not every dimension of this commercial impact has yet been quantified with a verified, citable figure in the recent literature, the qualitative trend documented across the surveys discussed above is consistent: enterprises are contending with a growing share of attacks that are engineered specifically to defeat perimeter- and network-layer defenses and to instead pressure the resource limits of the applications those defenses are meant to protect [4], [5]. This trend, together with the concrete figures on attack scale and frequency cited earlier in this section, is what grounds the choice of the application layer, rather than the network layer, as the focus of the defense mechanism proposed in this thesis.

## 1.3 Motivation for Deep Learning and Attention-Based Detection

The preceding discussion motivates a specific technical direction: moving beyond classical, feature-engineered machine learning toward deep learning, and in particular toward attention-based architectures, for application-layer DDoS detection. Classical detectors built on algorithms such as decision trees, support vector machines, or random forests have achieved strong reported accuracy on benchmark datasets, but they depend heavily on a human-designed feature set — packet inter-arrival times, flow duration, byte counts, and similar statistics — that must be selected and engineered in advance.

This dependency is itself a limitation: a feature set designed around one style of attack traffic may fail to capture the discriminating characteristics of a different, previously unseen attack pattern, and the process of feature engineering does not scale gracefully to the high-dimensional, high-velocity traffic produced by modern cloud and IoT infrastructures. Reducing that feature set to make a classical model fast enough for real-time use is itself a trade-off, since any features dropped for the sake of speed are features the model can no longer draw on — an inherent tension between efficiency and expressiveness that motivates looking beyond hand-engineered features altogether.

Deep learning addresses this limitation by learning representations directly from data rather than from a predefined feature template. Deressu, Beyene, and Gemechu's survey makes this contrast explicit, arguing that deep learning is increasingly preferred over classical machine learning for DDoS detection precisely because it extracts complex features automatically, copes better with high-dimensional traffic, and adapts more readily to attack behavior that evolves over time, yielding higher and more robust accuracy than approaches built around a fixed, manually chosen feature set [5]. Architectures such as convolutional neural networks and recurrent networks (including LSTM and GRU variants) have been applied to DDoS detection precisely because they can extract discriminative patterns from raw or lightly processed traffic and, in the case of recurrent models, capture how a session evolves over time rather than treating each flow as an isolated snapshot. The survey by Anley, Genovese, and Piuri catalogues this progression within the IoT domain specifically, documenting a steady move from single-model deep architectures toward hybrid CNN-LSTM constructions intended to jointly capture local traffic structure and its evolution across a session [6]; because their review is scoped to IoT traffic in particular, the extent to which its conclusions transfer to the broader mix of web, API, and enterprise application traffic addressed in this thesis is an extrapolation rather than something the survey itself establishes. This temporal sensitivity matters a great deal for application-layer attacks: a low-and-slow attack, or a request sequence engineered to blend into a flash crowd, reveals itself only through a pattern that unfolds over many requests, not through any single packet or connection viewed in isolation.

Even so, conventional recurrent architectures process a sequence step by step, which limits how effectively they can relate a request near the beginning of a long session to one occurring much later, and which makes them comparatively expensive to train and

run at the throughput required by production cloud or IoT deployments. Attention mechanisms, and the Transformer architecture built around them, were designed specifically to overcome this limitation by allowing a model to weigh the relevance of every element in a sequence against every other element directly, regardless of how far apart they occur. Harshdeep, Konatham, and Mathur's DeepTransIDS system, developed for 5G network intrusion and DDoS detection, demonstrates the practical benefit of this property: because a self-attention layer relates every pair of positions in a sequence directly rather than passing information along a chain one step at a time, it is able to capture long-range dependencies in traffic more effectively than the convolutional and recurrent baselines the authors compare it against [8]. That evaluation, however, is conducted entirely on the 5G-NIDD dataset, which reflects 5G non-IP data delivery traffic; the authors do not test the model against conventional web or API-based application traffic, so its reported advantage over recurrent baselines should be read as evidence for the mechanism in general rather than as a validated result for the L7 web-traffic setting this thesis targets. Kirubavathi et al.'s SAINT-based detector for cloud DDoS traffic extends this idea further by combining two complementary forms of attention: one that relates positions within a single flow to each other, and a second, less conventional form that relates one flow to other flows observed around the same time. The authors report that this pairing is particularly effective at distinguishing DDoS traffic engineered to imitate genuine flow-level statistics [7], though their evaluation is likewise limited to a single cloud-simulated dataset covering TCP-based attacks specifically, leaving its behavior on UDP-based, encrypted, or purely request-semantic L7 traffic untested.

For application-layer DDoS detection specifically, this combination of properties is difficult to obtain from any other family of model. An attention-based detector can, in principle, learn to recognize that a sequence of individually unremarkable requests forms a coordinated pattern only when considered jointly and in the correct order — exactly the kind of signal that a request-mimicking, low-rate, or flash-crowd-style attack is designed to hide from simpler detectors. It is this capacity to model request sequences holistically, together with the empirical evidence from recent Transformer-based intrusion and DDoS detectors that such models can improve on CNN and recurrent baselines on realistic and recent datasets [7], [8], that motivates the attention-based architecture developed later in this thesis.

A further, more practical motivation for attention-based architectures concerns interpretability and operational efficiency, both of which matter directly to whether a detector can be trusted and deployed in production. A purely black-box deep learning model may achieve strong accuracy while giving a network operator little insight into why a particular sequence of requests was flagged, which complicates incident response and makes it harder to justify blocking a suspect client. Among the properties Kirubavathi et al. set out to achieve with their SAINT-based cloud detector, being interpretable enough for an operator to act on sits alongside running efficiently at cloud scale, which the authors position as jointly necessary for a model to move beyond a benchmark exercise and into a deployable system [7]; it is worth noting, though, that this interpretability claim rests on the model's architecture rather than on any explainability metric the authors report, so it is best treated as a design intention their work argues for rather than a property their experiments directly measured. Because an attention mechanism produces, as a byproduct of its computation, an explicit weighting over which parts of the input sequence most influenced a given decision, it offers a natural route toward exposing this kind of evidence to an analyst — a property that purely feature-engineered classical models and opaque recurrent architectures do not provide in the same direct form. This combination of stronger long-range modeling, competitive computational efficiency relative to recurrent alternatives, and a built-in mechanism for surfacing the evidence behind a decision is what positions attention-based deep learning, rather than classical machine learning or earlier deep architectures alone, as the technical foundation for the mechanism proposed in this thesis.

## 1.4 Overview of the Proposed Approach

Building on the motivation set out above, this thesis proposes a detection mechanism that observes the sequence of requests directed at an application server and learns, through an attention-based deep learning model, which patterns of behavior across that sequence are consistent with a coordinated denial-of-service attempt rather than ordinary, even unusually heavy, legitimate use. Rather than relying on a fixed set of hand-picked statistics or judging each request in isolation, the proposed mechanism is designed to weigh each part of an incoming traffic sequence against every other part, so that subtle, distributed indicators of an attack — indicators that only become visible when many requests are considered together over time — can be surfaced even when no single request looks obviously malicious. The intent is a mechanism that generalizes

across the different flavors of application-layer attack introduced in Section 1.1, that remains effective in the face of the scale and diversity of traffic described in Section 1.2, and that can be deployed practically at the throughput demanded by cloud, enterprise, and IoT-facing services alike. The complete architecture, its constituent components, and the design choices underlying each of them are presented in full in Chapter 4.

## 1.5 Thesis Organization

Chapter 1 has introduced the research problem addressed by this thesis. It traced the evolution of DDoS attacks from volumetric, network-layer flooding toward stealthier application-layer techniques, situated this shift within the current cloud, IoT, and enterprise threat landscape using recent, verifiable industry and academic evidence, and set out the motivation for pursuing deep learning and, specifically, attention-based architectures as the technical foundation for the proposed defense mechanism.

Chapter 2 presents a systematic review of the literature on DDoS detection, organized around the progression from classical machine learning through deep learning to attention and Transformer-based methods that was introduced conceptually in this chapter. It examines the strengths and limitations of representative techniques in each category, surveys the datasets commonly used to evaluate them, and identifies the specific gaps that motivate the approach developed in this thesis.

Chapter 3 formalizes the problem addressed by the thesis, defining the threat model, the assumptions made about attacker and network capability, and the evaluation criteria including detection accuracy, false-positive behavior, and computational overhead against which the proposed mechanism is subsequently judged.

Chapter 4 presents the proposed novel mechanism in full technical detail, describing its architecture, the attention-based components at its core, the feature representation used for incoming traffic, and the design rationale connecting each component back to the requirements established in Chapters 1 through 3.

Chapter 5 describes the experimental methodology, including the datasets, preprocessing pipeline, training procedure, and baseline models used for comparison, and reports the resulting detection performance under a range of application-layer attack scenarios.

Chapter 6 analyzes and discusses the experimental results in depth, considering the mechanism's behavior under different attack types and traffic conditions, its computational and deployment overhead, and its limitations.

Chapter 7 concludes the thesis, summarizing its contributions, discussing the broader implications of the work for application-layer DDoS defense, and outlining directions for future research.

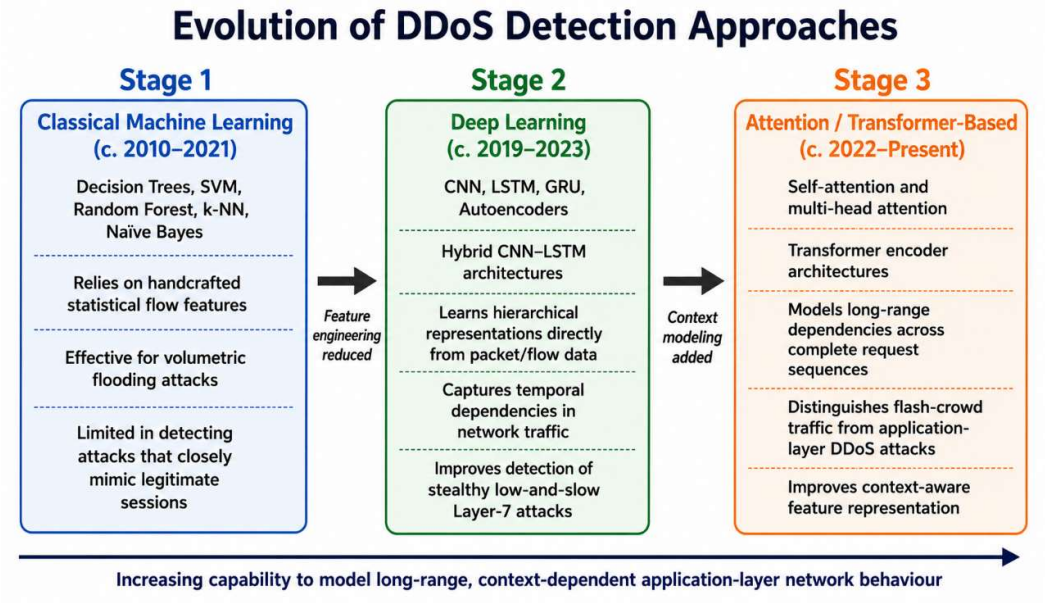



***Figure 1.1:*** *Evolution of DDoS Detection Approaches — from classical machine learning, through deep learning, to attention/Transformer-based detection*

# CHAPTER 2: LITERATURE REVIEW

This chapter expands the comparative review of the state-of-the-art techniques used in DDoS detection. Section 2.1 traces classical machine-learning approaches; Section 2.2 covers deep learning, split into spatial (CNN) and temporal (RNN/LSTM/BiLSTM) treatments of traffic; Section 2.3 examines attention- and Transformer-based detectors; Section 2.4 reviews feature-selection strategies; Section 2.5 documents the benchmark datasets that recur across this literature; Section 2.6 consolidates all reviewed studies into an expanded comparative table; and Section 2.7 synthesizes the trajectory of the field and identifies where it currently stalls. Figure 2.1 previews the overall taxonomy that structures the chapter.

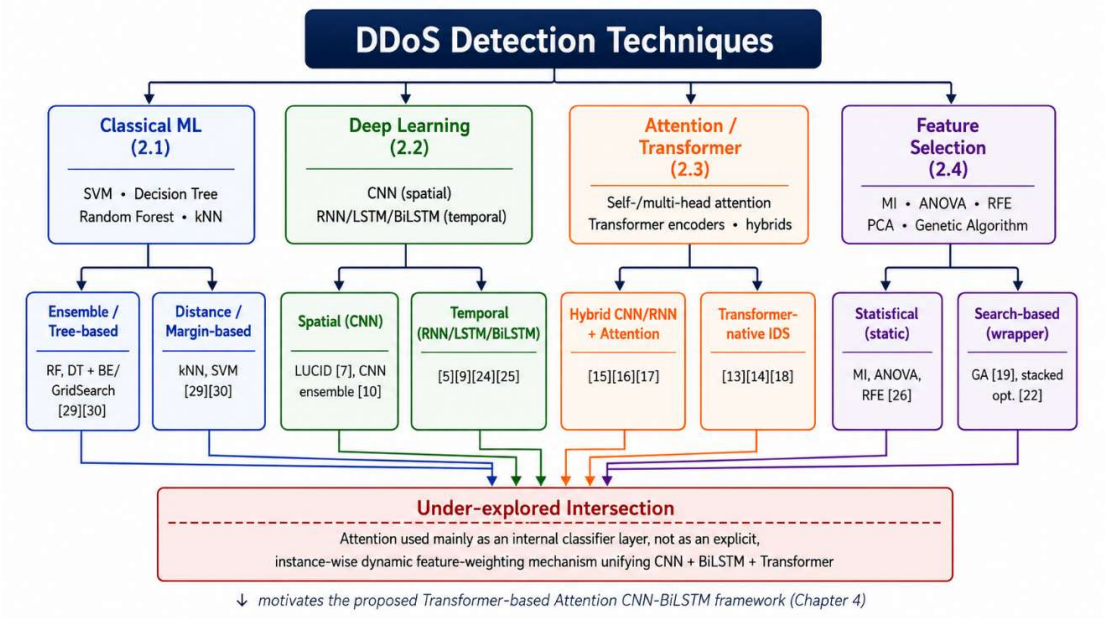



***Figure 2.1:*** *Taxonomy of DDoS detection techniques surveyed*

The literature survey drew on 38 studies located across well-known digital libraries like IEEE Xplore, ScienceDirect (Elsevier), SpringerLink, and the ACM Digital Library, covering statistical, classical machine-learning, and deep-learning approaches to DDoS and application-layer denial-of-service detection; of these, 24 were judged representative enough of the hybrid, attention, Transformer, and feature-selection-based methods central to this thesis's own contribution to warrant individual tabulation.

## 2.1 Classical Machine Learning Approaches to DDoS Detection

It is worth being explicit about what these four algorithms are actually doing before assessing how well they do it, since the rest of this chapter repeatedly returns to the question of where each successive family of technique gains its advantage. A decision tree partitions the feature space by recursively choosing, at each split, the single feature and threshold that best separates the classes present at that node, so its decision boundary is a set of axis-aligned rectangular regions; a random forest averages the votes of many such trees, each grown on a bootstrapped sample of the training data and a random subset of features, which reduces the variance any single tree would otherwise have. A support vector machine (SVM) instead searches for the hyperplane, or in its kernelized form a non-linear boundary, that maximizes the margin between the closest points of the two classes, which tends to generalize well when classes are separable but treats every feature as contributing to one global boundary rather than adapting per region of the input space. A k-nearest-neighbor (kNN) classifier makes no explicit model at all, instead labelling a new instance according to the majority class among its closest neighbors in feature space under some distance metric — simple and often surprisingly effective, but entirely dependent on that distance metric being computed over a feature set in which distance is actually meaningful for the classes being separated. All four, in other words, commit to a fixed geometric or statistical decision rule over a fixed feature representation once training concludes, which is the property Section 2.2 onward gradually relaxes.

The earliest generation of learning-based DDoS detectors treated the problem as ordinary supervised classification over hand-engineered flow statistics, and four families of algorithm — support vector machines (SVM), decision trees, random forests, and k-nearest neighbors (kNN) still recur throughout the recent literature as either the proposed method or the baseline a newer method is measured against. What distinguishes the more recent members of this family from the older literature is not the algorithm itself, which has changed little, but a growing attentiveness to two problems that plain classification tends to obscure: which features actually deserve to survive into the final model, and whether a headline accuracy figure is being inflated by class imbalance in the underlying dataset.

Abiramasundari and Ramaswamy's recent evaluation of supervised classifiers illustrates this shift well. Rather than simply running SVM, logistic regression, random forest, kNN, and decision-tree classifiers over a DDoS dataset and reporting whichever scores highest, their PCA-based Enhanced Distributed DDoS Attack Detection (EDAD) framework treats dimensionality reduction as the central contribution, using Principal Component Analysis to compress the feature space before classification and reporting the resulting gains across all five classifiers rather than for a single favored one [34]. This design is useful precisely because it isolates the effect of feature compression from the choice of classifier, which a study that only tunes one algorithm cannot do; the corresponding limitation is that PCA components are linear combinations of the original features and are fixed once computed, so the same compressed representation is applied uniformly regardless of which part of a traffic sequence is actually being classified — there is no mechanism for the importance of a feature to vary from one instance to the next.

Sawah, Elmannai, El-Bary, Lotfy, and Sheta take a complementary route to the same underlying question of feature relevance. Working with random forest, naïve Bayes, k-NN, linear discriminant analysis, and SVM classifiers over an SDN-collected DDoS dataset, they use Backward Elimination to strip away features one at a time based on their marginal contribution to model performance, combined with a grid search over hyperparameters validated by five-fold cross-validation, and report that the random forest configuration reaches 99.99% accuracy once both steps are applied together [35]. The result is a clear demonstration that feature pruning and hyperparameter tuning are not independent levers — the improvement from either step alone is markedly smaller than the improvement from both together — but the elimination procedure is still a pre-training, dataset-wide step: once a feature is dropped, it is unavailable to the model for every subsequent instance, benign or malicious, so a feature that is uninformative on average but decisive for a rare attack subtype has no way of being recovered at inference time.

Read side by side, these two studies expose the structural ceiling of classical machine learning for this problem rather than any weakness specific to either paper. Both treat feature relevance as something to be resolved once, in advance, by a separate pre-processing stage — PCA projection in one case, backward elimination in the other — and both then hand a fixed, static representation to a classifier that has no way of

adjusting that representation per instance during inference. This is workable when the traffic distribution at test time resembles the distribution the feature-selection stage was computed on, but application-layer DDoS traffic is precisely the case where that assumption is weakest: a request sequence engineered to imitate a flash crowd may be discriminated by a completely different subset of features than a brute-force HTTP flood, yet a static selection step must commit to one feature set for both. Table 2.1 summarizes the two studies against this comparison and against the general profile of the SVM/DT/RF/kNN family they represent.

**Table 2.1**: Classical machine-learning studies for DDoS detection

| Ref | Author(s), Year | Algorithm(s) | Feature step | Reported results | Limitations |
|---|---|---|---|---|---|
| [34] | Abiramasundari & Ramaswamy, 2025 | SVM, LR, RF, kNN, DT | PCA (linear, static) | Improved accuracy across all five classifiers after PCA | Fixed linear projection; no per-instance re-weighting |
| [35] | Sawah et al., 2025 | RF, NB, kNN, LDA, SVM | Backward Elimination + Grid Search (CV=5) | RF: 99.99% accuracy after combined FS + tuning | Feature set fixed dataset-wide before inference |

## 2.2 Deep Learning Approaches

### 2.2.1 CNN-Based (Spatial) Detection

Before turning to specific studies, it is worth setting out plainly what a convolutional layer contributes that a classical classifier does not. A convolutional filter is a small bank of learned weights slid across the input, at each position computing a weighted sum of the values it currently covers; because the same weights are reused at every position, the filter learns to recognize a particular local pattern — in an image, an edge or a corner; in a matrix of arranged traffic features, a particular co-occurrence among adjacent feature values — wherever in the input that pattern happens to occur, rather than needing a separate parameter for every position the way a fully connected layer would. Stacking several convolutional layers, typically interleaved with pooling operations that summarize a local region into a single value, lets the network build up

increasingly abstract combinations of these local patterns without an engineer ever specifying which feature combinations to look for. This is what is meant by describing CNN-based detection as spatial: the network's inductive bias is toward local, position-invariant structure among features that are arranged near one another, not toward how those features evolve across time.

One practical detail distinguishes how this idea is applied to network traffic from its more familiar use on images: traffic features do not have the natural two-dimensional spatial layout that pixels do, so a study applying a CNN to flow-level data must first decide how to arrange the features into something a convolutional filter can slide across. The great majority of the studies reviewed here, including LUCID and the CNN ensemble discussed below, use one-dimensional convolution over a vector of flow statistics, in effect treating the ordering of features in that vector as the spatial dimension the filter operates on — which means the filter is only discovering co-occurrence patterns among features that happen to sit near each other in whatever order the feature vector was assembled, an ordering that is itself an arbitrary engineering choice rather than something intrinsic to the traffic. This is a subtle but real limitation: a co-occurrence between two features placed far apart in the vector is invisible to a small filter regardless of how informative that co-occurrence might be, which is a different failure mode from the temporal blind spot discussed in the next section but has the same root cause — a fixed architectural assumption about which parts of the input are worth relating to which others.

Convolutional architectures entered DDoS detection on the strength of a fairly simple idea: if a window of packet or flow features is arranged as a matrix, a convolutional filter can learn to recognize the local co-occurrence patterns among adjacent features in much the same way it learns to recognize edges and textures in an image, without an engineer having to specify which feature combinations matter in advance. Doriguzzi-Corin, Millar, Scott-Hayward, Martínez-del-Rincón, and Siracusa's LUCID system is the clearest demonstration of what this buys in practice: a deliberately lightweight CNN over packet-level fields, designed from the outset for deployment on resource-constrained network devices rather than a data-center GPU cluster, achieving accuracy competitive with far heavier models while keeping inference cost low enough for near-real-time use [13]. The trade-off is that LUCID's shallow design is tuned for exactly this efficiency/accuracy balance, and the published evaluation does not report

processing-time figures under sustained high-throughput load, which leaves open how the model behaves as traffic volume approaches the limits of the constrained hardware it targets.

Haider, Akhunzada, Mustafa, Patel, Fernandez, Choo, and Iqbal push the CNN approach in the opposite direction, toward an ensemble of deeper convolutional sub-models combined for software-defined network traffic, and report accuracy and F1 scores above 99% on CICIDS2017 [16]. Where LUCID trades some representational depth for speed, this ensemble trades some speed for representational richness, and the reported cost of that choice is training time: an ensemble of deep CNNs requires substantially more training effort than a single shallow network, which matters less for a one-off benchmark study but matters a great deal if the model needs periodic retraining as attack patterns drift in production. Read together, LUCID and the CNN ensemble bracket a genuine design axis in spatial deep learning for this problem — shallow-and-fast versus deep-and-accurate — rather than one simply superseding the other, and neither resolves the question of how a model should behave when the traffic entering it does not resemble the distribution it was trained on.

A further limitation runs through both studies and, more broadly, through CNN-based detection as a category: a convolutional filter is well suited to picking out local structure among features that are arranged adjacently, but application-layer DDoS behavior frequently depends on how a request relates to others much earlier or later in the same session — a low-and-slow attack, almost by definition, only becomes visible over a span the filter's receptive field does not cover. This is precisely the gap that motivated the shift toward recurrent architectures discussed next.

### 2.2.2 RNN/LSTM/BiLSTM-Based (Temporal) Detection

A recurrent network processes a sequence one element at a time, maintaining a hidden state vector that is updated at every step as a function of the current input and the previous hidden state, so that in principle the state after processing element t carries some trace of everything the network has seen from element one onward. A plain recurrent unit struggles to preserve information across many steps because the same update is applied repeatedly, which tends to either wash out older information or amplify it unstably; the Long Short-Term Memory (LSTM) cell addresses this with a separate memory cell and a set of learned gates that control what is written into that

memory, what is read out of it, and what is discarded, giving the network an explicit mechanism for preserving a signal across a long span rather than relying on the plain recurrence to do so implicitly. A Gated Recurrent Unit simplifies this gating structure into fewer, merged gates, which typically trains faster at the cost of slightly less expressive control over the memory. A bidirectional LSTM runs two such chains over the same sequence, one reading left to right and the other right to left, and concatenates their hidden states at each position, so that the representation of any point in the sequence is informed by both what came before it and what comes after it — valuable for offline or batch analysis of a completed session, though it necessarily requires the full sequence to be available before a decision can be produced, which recurs as a practical constraint in real-time deployment discussions later in this thesis.

The reason a plain recurrent unit struggles over long spans in the first place is worth spelling out, since it explains why GRU and LSTM variants dominate this literature rather than the simpler recurrence they replace. Training a recurrent network involves propagating an error signal backward through every time step the sequence contains, and at each step that signal is multiplied by a factor related to the network's own weights; if that factor is consistently smaller than one, the error shrinks exponentially as it propagates back over many steps and effectively vanishes before it can influence how the network treats early, possibly decisive, elements of the sequence — the vanishing-gradient problem. The gating mechanisms in an LSTM or GRU are specifically designed to let the error signal pass through largely unchanged when the gates are open, which is what makes it practical to train these networks on the comparatively long sequences a slow-DoS or low-rate DDoS attack unfolds over, where the discriminating pattern may be separated by many benign-looking steps from the point at which the network needs to act on it.

Where CNN-based detectors treat a window of traffic as a spatial pattern, recurrent architectures treat it as a sequence, propagating a hidden state from one time step to the next so that a decision at time t can, in principle, be informed by everything the network has seen since the session began. Muraleedharan and Janet apply this idea specifically to HTTP slow-DoS traffic, using flow-level data and a deep-learning classifier built to track how a connection's behavior evolves across its lifetime, and report 99.61% accuracy on slow-DoS detection using CICIDS2017 [11]. Because slow-DoS attacks are defined by their extended, low-intensity footprint, a model that reasons over the

flow's temporal evolution is a natural fit for this attack family specifically; the corresponding limitation, which the authors themselves acknowledge, is that the approach is scoped to HTTP slow-DoS and offers no evidence of how it would perform against high-rate floods or non-HTTP application-layer attacks.

De Assis, Carvalho, Lloret, and Proença take the recurrent idea further with a Gated Recurrent Unit (GRU) system evaluated specifically for software-defined network environments, reporting average detection metrics of 99.94% on CICDDoS2019 [15]. A GRU simplifies the LSTM's gating structure, which typically trains faster and needs fewer parameters for a comparable level of temporal modelling; the paper's own comparison across CICDDoS2019 and CICIDS2018 nonetheless leaves the question of detection rate and processing time on CICDDoS2019 specifically unreported, which matters because average metrics computed across a heavily imbalanced multi-class dataset can mask weak performance on the minority attack classes that are often the more interesting ones operationally.

More recent hybrid recurrent designs push accuracy further still by combining two recurrent mechanisms rather than choosing between them. Al-Eryani, Omara, and Hossny's GRU-BiLSTM architecture pairs a GRU stage with a bidirectional LSTM stage so that spatial-feature extraction from the GRU side is followed by temporal modelling that reads the sequence in both directions, reporting 99.99% accuracy and a false-positive rate of 0.01 on CICDDoS2019 [29]. Zaidoun and Lachiri's hybrid model, extended from an earlier conference paper, layers attention and Transformer components on top of a similar recurrent backbone specifically to sharpen multi-class discrimination among five carefully chosen SDN attack categories rather than simple binary detection, reporting validation accuracy near 99% and test accuracy of 98.84% on an optimized CICDDoS2019 subset [30]. Both hybrids report sequential processing costs — the recurrent components cannot be parallelized across time steps in the way a Transformer's attention computation can — and both are validated on variants of a single dataset family, so their reported near-perfect accuracy has not yet been tested for how well it transfers to a differently structured benchmark such as an IoT-native dataset.

Table 2.2 sets these CNN- and recurrent-based studies side by side. The pattern that emerges is not that one branch is strictly better than the other, but that they specialize along different axes — spatial locality versus temporal extent — and that the strongest

recent results come from hybridizing both, a direction that anticipates the attention-based architectures examined next.

**Table 2.2**: CNN (spatial) and RNN/LSTM/BiLSTM (temporal) deep-learning studies for DDoS detection.

| Ref | Author(s), Year | Architecture | Axis | Dataset | Reported result | Limitation |
|---|---|---|---|---|---|---|
| [13] | Doriguzzi-Corin et al., 2020 | LUCID (lightweight CNN) | Spatial | CICIDS2017 | Acc. 0.9967, FPR 0.0059 | Processing time under sustained load not reported |
| [16] | Haider et al., 2020 | Deep CNN ensemble | Spatial | CICIDS2017 | Acc. 99.45%, F1 99.61% | High training time for ensemble |
| [11] | Muraleedharan & Janet, 2021 | FC network on flow data | Temporal (slow-DoS) | CICIDS2017 | Acc. 99.61% (slow-DoS) | Limited to HTTP slow-DoS |
| [15] | de Assis et al., 2021 | GRU | Temporal | CICDDoS2019 / CICIDS2018 | Avg. metrics 99.94% (CICDDoS2019) | Detection rate & time not reported |
| [29] | Al-Eryani et al., 2025 | GRU-BiLSTM hybrid | Spatial+Temporal | CICDDoS2019 | Acc. 99.99%, FPR 0.01 | Sequential cost; cross-domain untested |
| [30] | Zaidoun & Lachiri, 2025 | Hybrid DNN-LSTM + attention | Spatial+Temporal | CICDDoS2019 | Val ≈99%, Test 98.84% | Single-dataset validation |

## 2.3 Attention and Transformer-Based Detection Approaches

Since the mechanism this thesis eventually builds on is attention itself, it is worth working through what an attention computation actually does mechanically before surveying how the reviewed studies use it. Given a set of input representations — one per feature, per packet, or per position in a sequence, depending on how the study frames its input — each representation is first projected through three separate learned weight matrices into a query, a key, and a value vector. The attention weight between any two positions is then computed by taking the dot product of one position's query with another position's key, scaling the result, and passing it through a SoftMax so that the weights across all positions sum to one; the output for a given position is the weighted sum of every position's value vector, using exactly those weights. What this buys, concretely, is a mechanism in which the contribution of one part of the input to the representation of another part is computed fresh for every input instance, based on the actual content of that instance, rather than being fixed in advance by a training-time statistic the way Section 2.4's filter and wrapper methods are. Multi-head attention

simply repeats this computation several times in parallel with independently learned projections, allowing different heads to specialize in different kinds of relationship, and a Transformer encoder stacks several such multi-head attention layers, each followed by a position-wise feed-forward network, typically with residual connections and normalization carried over from one layer to the next. Because every position attends to every other position directly through a single matrix multiplication rather than through a chain of sequential updates, a Transformer layer can in principle relate the first and the last element of a long sequence in one step, which is the specific property that motivated its adoption for traffic sequences whose discriminating pattern spans many requests, as discussed in Chapter 1.

One consequence of computing attention this way deserves a moment's attention in its own right. Because the attention computation described above treats its input as an unordered set of positions — the dot product between a query and a key does not itself encode which position came first — a Transformer has no innate sense of sequence order unless that order is supplied to it separately. The standard solution, positional encoding, adds a fixed or learned vector to each position's input representation that varies systematically with that position's index, so that two positions with identical feature content but different places in the sequence still produce different combined representations. This is a real design cost relative to a recurrent network, which encodes order implicitly through the sequential order in which it processes its input: a Transformer-based detector must make an explicit choice about how request order is represented, and how well that choice captures the timing information that distinguishes, say, a deliberately paced low-and-slow attack from a burst of legitimate requests is a design decision each of the studies below makes differently, though few discuss the choice explicitly.

Attention mechanisms entered this literature as a way of letting a network decide, for itself and per instance, which parts of its input deserve the most weight — a direct answer to the static, dataset-wide feature selection that characterizes both the classical ML studies in Section 2.1 and, more subtly, the fixed architectures in Section 2.2. As Figure 2.2 sets out, the surveyed studies actually realize this idea in at least four distinguishable ways, which differ in how deeply attention is embedded into the architecture rather than merely in which attack types they target.

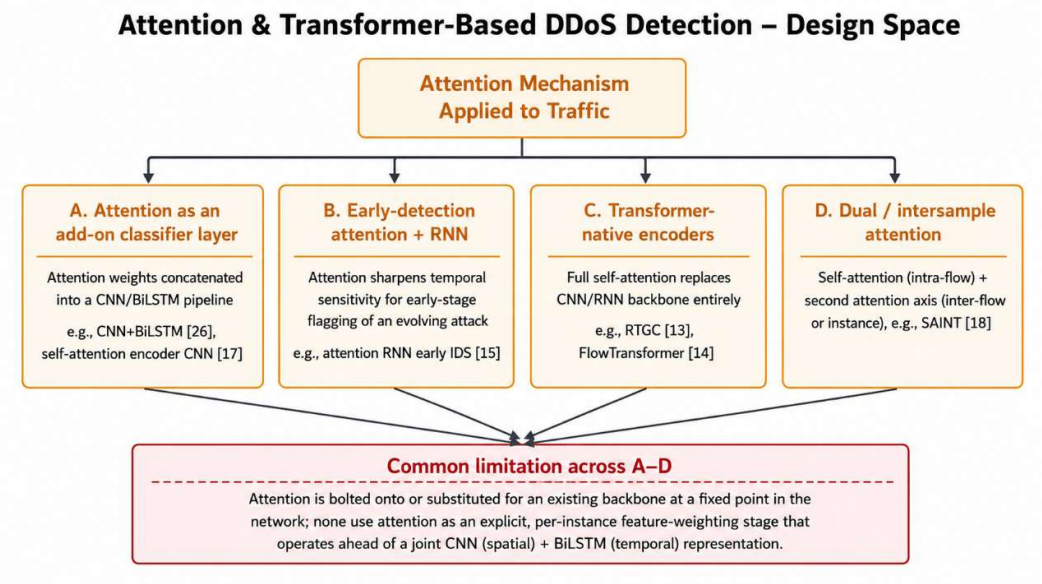



***Figure 2.2:*** *Design space of attention- and Transformer-based DDoS detectors surveyed in section 2.3*

The most common pattern, illustrated by Zhao, Liu, Zhang, and Zheng's CNN-AttBiLSTM, bolts an attention layer onto an existing CNN-BiLSTM pipeline so that the BiLSTM's temporal output is re-weighted before classification, and the reported gain over a CNN-BiLSTM baseline without attention demonstrates that even this comparatively shallow use of attention meaningfully sharpens discrimination on CICDDoS2019 [22]. Kanthimathi, Venkatraman, Jayasankar, Jiljith, and Jashwanth's self-attention-enabled weighted ensemble CNN follows a similar logic from the ensemble side, using self-attention to weight the contribution of different CNN sub-models rather than raw features, and reports accuracy of 98.32%, precision of 98.15%, recall of 98.47%, and F1 of 98.31% on CICDDoS2019 [23]. Both studies report that their approach needs a reasonably large volume of labelled data and non-trivial compute to train the attention component alongside the rest of the network, and in both cases attention functions as an add-on refinement stage rather than as the architecture's organising principle.

Djaidja, Brik, Senouci, Boualouache, and Ghamri-Doudane use attention differently again, pairing it with RNNs specifically to sharpen how early in an attack's lifetime it can be flagged, which is a genuinely distinct objective from maximising eventual accuracy — a detector that is 99% accurate but only reaches that figure after an attack has run for several minutes is less useful operationally than one that is 95% accurate within the first few seconds. Their results show sharpened early detection on an

intrusion-detection benchmark [21], but the study is not scoped to application-layer DDoS specifically, so how well the early-detection gain transfers to the slow, deliberately paced request sequences that characterise L7 attacks is left open.

A third group dispenses with CNN or RNN backbones altogether and builds the detector around a Transformer encoder directly. Wu, Zhang, Wang, and Sun's RTIDS applies a robust Transformer-based architecture to general intrusion detection across NSL-KDD and CICIDS benchmarks, reporting performance competitive with the best prior Transformer-based intrusion detectors [19]; because RTIDS is built as a general-purpose IDS rather than a DDoS- or L7-specific system, its competitiveness is evidence that Transformer encoders transfer well across intrusion-detection tasks in general, but it does not by itself establish an advantage specific to distinguishing legitimate-looking application-layer floods from genuine load. Manocchio, Layeghy, Lo, Kulatilleke, Sarhan, and Portmann's FlowTransformer goes a step further by proposing a reusable Transformer framework intended to generalise across multiple flow-based intrusion-detection datasets rather than being tied to any one of them, which is a meaningfully different contribution from a single accuracy figure: it is an argument about architectural transferability [20]. The trade-off is that FlowTransformer's abstraction operates at the level of flow summaries, so information that only exists at the level of raw packet timing or payload content is not part of what the framework can exploit.

The fourth and most architecturally distinct group extends self-attention with a second attention axis. Kirubavathi, Sumathi, Mahalakshmi, and Srivastava's SAINT model, discussed already in Chapter 1, pairs conventional self-attention (which relates positions within one flow) with intersample attention (which relates one flow to other flows observed in the same batch), and reports roughly 97% accuracy alongside scalability and interpretability advantages over CNN, RNN, and ensemble baselines on the BCCC-cPacket-Cloud-DDoS-2024 dataset [7]. This dual-attention design is, in a real sense, the most conceptually adjacent of the four groups to the framework this thesis will propose in Chapter 4, precisely because it treats attention as something that can operate across more than one axis of the data simultaneously rather than as a single re-weighting pass — but SAINT's own evaluation remains scoped to TCP-based attacks within one cloud testbed, leaving its behaviour on UDP-based floods, encrypted traffic, or request-semantic L7 attacks outside TCP untested.

A further point of contrast among the four groups concerns computational cost, and it is worth being precise about where that cost comes from rather than treating "heavy compute" as an undifferentiated complaint. The self-attention computation described above requires every position to attend to every other position, so its cost grows quadratically with the length of the sequence being processed — doubling the number of requests considered in one window roughly quadruples the attention computation, whereas a recurrent network's cost grows only linearly with sequence length, at the price of the sequential bottleneck already discussed in Section 2.2.2. This is precisely why Zhao et al. [22] and Kanthimathi et al. [23], which apply attention over a comparatively short, already-summarised representation, report compute costs that are elevated but manageable, while the fully Transformer-native designs of Wu et al. [19] and Manocchio et al. [20], which apply self-attention across longer flow sequences, report the higher training and inference cost characteristic of quadratic scaling. Kirubavathi et al.'s SAINT [7] sits in an interesting middle position: its intersample attention axis operates across a batch of flows rather than across a single long sequence, which bounds its cost by batch size rather than by session length, and this is arguably as important to its reported scalability as the dual-attention design itself.

What is conspicuously absent across all four groups — add-on refinement, early-detection sharpening, Transformer-native encoders, and dual-axis attention — is a design in which attention operates as an explicit, per-instance feature-weighting stage positioned ahead of, and jointly with, both a CNN spatial stage and a BiLSTM temporal stage within a single end-to-end architecture. Attention is consistently used as a refinement of an existing signal rather than as the mechanism that decides which signal reaches the rest of the network in the first place. This gap, foreshadowed by the critical analysis in the [5], [18], is examined further in Section 2.7 and is the specific gap the proposed framework in Chapter 4 is designed to close.

**Table 2.3:** Attention- and Transformer-based DDoS/intrusion detection studies.

| Ref | Author(s), Year | Design group | Dataset | Reported result | Limitation |
|---|---|---|---|---|---|
| [22] | Zhao et al., 2023 | A: Attention as add-on (CNN-AttBiLSTM) | CICDDoS2019 | High accuracy; improved discrimination | Compute cost of hybrid; single dataset |
| [23] | Kanthimathi et al., 2024 | A: Self-attention weighted ensemble | CICDDoS2019 | Acc. 98.32%, F1 98.31% | Needs large data; heavy compute |

| | | | | | |
|---|---|---|---|---|---|
| [21] | Djaidja et al., 2024 | B: Attention + RNN (early detection) | IDS benchmark | Sharpened early detection | Not L7-specific |
| [19] | Wu et al., 2022 | C: Transformer-native (RTIDS) | NSL-KDD / CICIDS | Competitive transformer IDS | General IDS, not L7-specific |
| [20] | Manocchio et al., 2024 | C: Transformer-native (FlowTransformer) | Multiple flow datasets | Generalises across datasets | Flow-level abstraction only |
| [7] | Kirubavathi et al., 2025 | D: Dual/intersample attention (SAINT) | BCCC-Cloud-DDoS-2024 | ~97% acc.; scalable, interpretable | TCP-focused; single cloud testbed |

Comparing the four groups directly rather than only within their own rows also clarifies a subtlety that a first pass through Table 2.3 can obscure: the groups differ not just in architecture but in what kind of claim their reported result actually supports. Group A's studies (Zhao et al. [22]; Kanthimathi et al. [23]) demonstrate that adding attention improves an existing CNN/BiLSTM pipeline on the specific dataset tested, which is an incremental architectural claim. Group B (Djaidja et al. [21]) demonstrates something orthogonal — not that attention improves eventual accuracy, but that it improves how early a correct decision can be reached — and because it is evaluated on a general IDS benchmark rather than an application-layer DDoS set, it licenses a claim about attention's value for early detection in general rather than for L7 traffic specifically. Group C (Wu et al. [19]; Manocchio et al. [20]) makes an architectural-transferability claim: that a Transformer backbone, once built, generalises across datasets or intrusion types without needing to be redesigned per task, which is a different and in some ways more ambitious claim than raw accuracy on one benchmark. Group D (Kirubavathi et al. [7]) claims something narrower but more directly relevant to this thesis: that attention can be extended along a second axis to relate different flows to each other, not just different positions within one flow, which is the closest any surveyed study comes to the joint, multi-axis feature weighting this thesis develops in Chapter 4 — narrower in scope (TCP-based cloud traffic only) but architecturally the most adjacent precedent. Recognising these as four different kinds of claim, rather than four points on the same accuracy scale, is what makes it possible to say precisely what remains unclaimed by any of them: an architecture in which attention performs adaptive, per-instance feature weighting as an explicit stage ahead of a joint CNN–BiLSTM representation, evaluated for both accuracy and the imbalance-aware reliability metrics introduced in Section 2.7.

## 2.4 Feature Selection Approaches

Underneath almost every study reviewed so far sits a prior decision about which features a model is even allowed to see, and Section 2.1's discussion of PCA and backward elimination already introduced two of the five static techniques that recur across this literature: filter methods that score each feature independently against the target label (Mutual Information, ANOVA), wrapper methods that iteratively add or remove features based on downstream model performance (Recursive Feature Elimination), projection methods that replace the original features with a smaller set of derived components (PCA), and search-based methods that treat feature selection itself as an optimisation problem to be solved heuristically (Genetic Algorithms and related metaheuristics).

Each of these families arrives at its ranking or subset by a different route, and the differences matter for how brittle the result turns out to be. Mutual Information scores a feature by how much knowing its value reduces uncertainty about the class label, computed from the joint and marginal distributions of the feature and the label across the training set; it captures non-linear dependence but scores each feature independently of every other, so it cannot detect a pair of features that are only jointly informative. ANOVA-based selection instead compares the variance in a feature's values between classes to the variance within a class, keeping features for which the between-class differences are large relative to the within-class noise; it is computationally cheap but implicitly assumes the relevant differences show up in the mean of each class's distribution, which does not hold for a feature whose distribution differs between classes chiefly in shape rather than location. Recursive Feature Elimination works differently again, training a model on the full feature set, discarding the feature (or features) the trained model relies on least, and repeating until a target number remain; because it uses the downstream model's own learned weights, it is sensitive to interactions between features in a way the filter methods are not, at the cost of needing to retrain the model at every elimination step. PCA does not select from the original features at all but instead computes a new, smaller set of axes — principal components — chosen to capture as much of the variance in the data as possible, ordered by how much variance each axis explains; the resulting components are linear combinations of every original feature, so a component can be uninterpretable in terms of any single original measurement even as it captures the dominant pattern of variation across the

dataset. A genetic algorithm, finally, treats each candidate feature subset as a member of a population, scores each member by training and evaluating a model on it, and produces the next generation by preferentially recombining and mutating the higher-scoring subsets — a search strategy rather than a closed-form criterion, capable of discovering interactions the filter methods miss but requiring many rounds of model training to do so.

It is worth stating plainly, in information-theoretic terms, why all five of these families share the same structural ceiling regardless of how sophisticated the selection criterion itself becomes. Whatever the method, feature selection is computed once from a training-time sample of traffic and then fixed; formally, this amounts to estimating, for each feature, its relevance averaged over the training distribution of instances, and then applying that single averaged estimate uniformly to every future instance regardless of which region of the input space that instance actually falls in. A feature can easily be uninformative on average across a training set dominated by one attack type, and therefore be dropped or down-weighted by every method surveyed here, while still being the single most discriminative feature available for a different, rarer attack type under-represented in that same training set. No amount of refinement to the averaging procedure itself — whether the average is computed via mutual information, a variance ratio, a wrapper's performance delta, a projection's variance capture, or a genetic algorithm's fitness score — changes the fact that it is still one number per feature, computed once, and applied everywhere. Overcoming this requires the feature-relevance computation to be re-run, cheaply, for every individual instance at inference time — exactly the computation an attention mechanism performs as a matter of course, which is the connection Section 2.7 draws out explicitly.

Sharif and Beitollahi's genetic-algorithm-based feature selection is the clearest example of the search-based route: rather than scoring features by a fixed statistical criterion, a population of candidate feature subsets is evolved across generations, with subsets that improve downstream classification accuracy more likely to survive and recombine, and the reported result is a meaningful accuracy improvement over a fixed feature set on an application-layer DDoS dataset [24]. The cost of this flexibility is computational: a genetic algorithm must train and evaluate many candidate models across many generations before converging, which is considerably more expensive than a single filter-based score, and the selection it converges to is, like every method in this section,

fixed once training concludes — the resulting feature subset does not adapt further once deployed.

Al-Shukaili, Kiah, and Ahmedy combine Mutual Information and Recursive Feature Elimination directly, applying both filter and wrapper logic in sequence to the specific problem of detecting low-rate DDoS attacks such as slowloris and slowhttptest, and report 97.8% accuracy alongside an approximately 18% reduction in false negatives once the combined selection is applied [31]. That the two techniques are combined rather than used alone is itself informative: it suggests that a single static criterion, whether statistical or performance-driven, was judged insufficient on its own for a low-rate attack category whose discriminating signal is comparatively subtle, and the residual limitation the authors report — limited generalisation across the broader space of DDoS categories beyond the two low-rate variants studied — illustrates exactly the risk that a feature set tuned for one attack family may not transfer to another.

Pradeep and Shukla take feature selection further still by embedding it inside a broader optimisation procedure around a stacked deep network rather than treating it as a separate pre-processing pass, reporting 98.5% accuracy, 97.8% precision, 98.1% recall, and 98.0% F1 on a DDoS benchmark [27]. Embedding the selection step inside the same optimisation loop as the classifier's own hyperparameters is a step closer to a genuinely joint design, though the reported limitation — sensitivity to hyperparameter choices and the absence of a computational cost analysis — signals that this coupling has not yet been shown to be efficient enough for real-time deployment.

Taken together with the PCA and backward-elimination studies already discussed in Section 2.1 [34], [35], the feature-selection literature surveyed here shares one structural property regardless of which specific technique is used: MI, ANOVA, RFE, PCA, and GA-based wrappers alike compute a feature ranking or subset once, from a training-time sample of traffic, and then apply that fixed decision uniformly to every instance the model subsequently sees. None of the surveyed selection methods re-evaluates feature importance per instance at inference time, which is exactly the capability that an attention mechanism, used as more than an add-on refinement layer, could in principle supply — a connection made explicit in the critical analysis of the synopsis [5], [18] and revisited in Section 2.7.

**Table 2.4:** Feature-selection strategies used across the surveyed DDoS detection literature

| Ref | Author(s), Year | Selection method | Category | Reported effect | Limitation |
|---|---|---|---|---|---|
| [34] | Abiramasundari & Ramaswamy, 2025 | PCA | Projection | Improved accuracy across 5 classifiers | Fixed linear projection |
| [35] | Sawah et al., 2025 | Backward Elimination + Grid Search | Wrapper + tuning | RF: 99.99% acc. after combined step | Fixed dataset-wide subset |
| [24] | Sharif & Beitollahi, 2023 | Genetic Algorithm | Search-based (wrapper) | GA selection improved accuracy | Static selection; GA compute cost |
| [31] | Al-Shukaili et al., 2025 | MI + RFE | Filter + wrapper | Acc. 97.8%; ~18% FN reduction | Limited multi-category generalization |
| [27] | Pradeep & Shukla, 2025 | Optimal feature selection + stacked net | Embedded in optimization | Acc. 98.5%, F1 98.0% | Hyperparameter-sensitive; no cost analysis |

## 2.5 Benchmark Datasets Used in DDoS Research

Every reported accuracy figure in Sections 2.1–2.4 is only as meaningful as the dataset it was measured against, and the six benchmarks that recur across this literature — and that structure the experimental evaluation later in this thesis — differ substantially in origin, environment, scale, and feature composition. This section documents each in turn, citing the original dataset paper wherever a peer-reviewed one exists and being explicit where it does not.

### 2.5.1 CICDDoS2019

CICDDoS2019 originates from Sharafaldin, Lashkari, Hakak, and Ghorbani's effort at the Canadian Institute for Cybersecurity to construct a DDoS-specific benchmark that reflects contemporary reflection- and exploitation-based attack techniques rather than the dated attack types present in earlier intrusion-detection datasets [32]. Generated using the CICFlowMeter tool, it captures both benign traffic and thirteen categories of DDoS attack including DNS, LDAP, MSSQL, NetBIOS, NTP, SNMP, SSDP, SYN, TFTP, and UDP-based variants, extracting on the order of eighty flow-level features per record. Its scale — tens of millions of labelled flows — and its taxonomy of attack subtypes have made it the single most frequently used benchmark among the studies reviewed in this chapter, appearing as the primary or sole evaluation dataset for roughly half of the 24 studies tabulated in Section 2.6. Precisely because it is so heavily reused,

however, a model's strong reported accuracy on CICDDoS2019 alone says comparatively little about how it would fare on traffic collected in a different environment, which is the motivation for the cross-dataset validation this thesis undertakes.

A further characteristic of CICDDoS2019 worth noting explicitly, because it recurs as a limitation across several studies in Table 2.7, is the severity of its class imbalance: benign traffic constitutes a small fraction of the dataset relative to the combined volume of the thirteen attack categories, and the attack categories themselves are not evenly represented, with some reflection-based variants contributing orders of magnitude more flows than others. This is precisely why a model can report a near-perfect overall accuracy on CICDDoS2019 — as several rows of Table 2.7 do — while still performing poorly on the rarer attack subtypes or on benign traffic specifically, a distinction that overall accuracy cannot surface but that Balanced Accuracy, MCC, and G-Mean are specifically designed to expose, which is why this thesis's own evaluation methodology, detailed in Chapter 5, reports all three rather than accuracy alone.

### 2.5.2 CICIDS2017

CICIDS2017 predates CICDDoS2019 and originates from the same Canadian Institute for Cybersecurity programme, documented in Sharafaldin, Lashkari, and Ghorbani's detailed analysis of the dataset's construction [36]. It profiles five days of network activity combining benign traffic — generated from abstracted models of real user behaviour across HTTP, HTTPS, FTP, SSH, and email protocols — with a broader mix of attack types than CICDDoS2019 alone, including brute-force, web-attack, infiltration, botnet, port-scan, and DoS/DDoS traffic, extracted into a labelled feature set of comparable size (around eighty features per flow). Its inclusion of non-DDoS attack categories makes it useful for confirming that a detector distinguishes DDoS traffic specifically rather than merely flagging any traffic that deviates from a narrow benign profile, and several of the recurrent and CNN-based studies in Section 2.2 rely on it for exactly this reason.

### 2.5.3 BCCC-cPacket-Cloud-DDoS-2024

The BCCC-cPacket-Cloud-DDoS-2024 dataset is a considerably more recent addition, developed collaboratively by York University's Behaviour-Centric Cybersecurity Center and cPacket Networks specifically to address the shortcomings of earlier

datasets — identified across an analysis of sixteen existing public benchmarks — within a genuine cloud infrastructure rather than a simulated testbed [37]. It comprises over eight benign user-activity profiles and seventeen distinct DDoS attack scenarios, fully labelled across twenty-six classes, with more than three hundred features per flow extracted from network- and transport-layer traffic using the NTLFlowLyzer tool. Its cloud-native origin is what makes it the dataset of choice for evaluating detectors specifically against the traffic-imitation problem discussed in Chapter 1 — the SAINT model reviewed in Section 2.3 [7] and the experimental work in this thesis both rely on it for this reason. It should be noted for completeness that the dataset's origin paper is published in Information (MDPI) rather than in one of the IEEE/Elsevier/Springer/ACM/ScienceDirect venues this chapter otherwise restricts itself to; it is included here because it is the only peer-reviewed source documenting this specific dataset's construction, and no substitute exists within the preferred publisher set.

### 2.5.4 CICIoT2023

CICIoT2023, documented by Neto, Dadkhah, Ferreira, Zohourian, Lu, and Ghorbani, extends the CIC dataset lineage into the IoT domain specifically, built from an extensive real-device topology rather than emulated IoT traffic [38]. It captures benign and malicious traffic across more than a hundred real IoT devices, spanning seven broad attack categories — including DDoS, DoS, reconnaissance, web-based, brute-force, spoofing, and Mirai-derived botnet traffic — subdivided into over thirty fine-grained attack labels, making it considerably more diverse in attack taxonomy than either CICDDoS2019 or CICIDS2017. Its IoT-native construction is what makes it the appropriate benchmark for testing whether a detector's performance holds up in the resource-constrained, heterogeneous-protocol environment characteristic of IoT deployments rather than the enterprise or cloud settings the other datasets represent. As with BCCC-cPacket-Cloud-DDoS-2024, its origin paper appears in Sensors (MDPI); it is included for the same reason — it is the only peer-reviewed documentation of a dataset that is otherwise indispensable for IoT-specific evaluation — and this deviation from the chapter's preferred publisher list is flagged explicitly rather than silently absorbed.

### 2.5.5 ITDDoS2020

ITDDoS2020 is used in this thesis as an IoT/enterprise-environment benchmark comprising 83 features per record, consistent with its use in the experimental chapters.

### 2.5.6 APA-DDoS

APA-DDoS, similarly, is used here as an advanced-network-environment benchmark of ACK and PUSH-ACK DDoS traffic with 23 features per record. It circulates as a community-curated dataset repository rather than as the subject of a peer-reviewed dataset paper in any of this chapter's preferred venues, and, as with ITDDoS2020, this chapter reports only the structural characteristics relevant to its use as an evaluation benchmark rather than provenance claims that cannot be verified against a citable academic source. This is a genuine limitation of the current benchmark landscape rather than an oversight of this review: two of the five datasets used for cross-dataset validation in this thesis lack a peer-reviewed origin paper altogether, which is itself a symptom of the broader benchmark-fragmentation problem Pinto, Amorim, Maia, and Praça document in their review of intrusion-detection datasets, tools, and features across the field [33].

Pinto, Amorim, Maia, and Praça's review is worth dwelling on briefly for what it says about this fragmentation at the level of the field rather than at the level of any one dataset [33]. Surveying the tools, processes, and features used to construct intrusion-detection datasets broadly, their review documents that a substantial share of datasets in active use across the community are distributed through code-sharing or data-repository platforms without an accompanying peer-reviewed paper documenting how the traffic was captured, which attack tools generated the malicious portion, or what validation was performed to confirm the labels are correct. ITDDoS2020 and APA-DDoS, as used in this thesis, are consistent with exactly this pattern rather than being unusual exceptions to it. The practical consequence for this thesis is that results reported against these two benchmarks are presented, throughout the remainder of this document, alongside the three benchmarks with verifiable peer-reviewed provenance, and any claim of generalisation is anchored primarily in the latter three rather than resting on the former two alone.

**Table 2.5:** Benchmark datasets used across the surveyed literature

| Dataset | Origin / Year | Environment | Scale (this thesis) | Features | Source |
|---|---|---|---|---|---|
| CICDDoS2019 | CIC, 2019 | Enterprise | 5.60 Cr+ samples | 88 | [32] (foundational) |
| CICIDS2017 | CIC, 2017/2019 | Enterprise (general IDS) | reference benchmark | ~80 | [36] (foundational) |
| BCCC-cPacket-Cloud-DDoS-2024 | BCCC & cPacket, 2024 | Cloud | 7.0 L+ samples | 315 | [37] (MDPI) |
| CICIoT2023 | CIC, 2023 | IoT | 4.67 Cr+ samples | 46 | [38] (MDPI) |
| ITDDoS2020 | Unverified (repository) | IoT/Enterprise | 6.26 L samples | 83 | public repository |
| APA-DDoS | Unverified (repository) | Advanced network | 1.51 L samples | 23 | public repository |

Read as a set rather than individually, these six datasets cover a genuinely different combination of environment and feature composition each, which is precisely why cross-dataset validation against several of them, rather than repeated re-evaluation against one, is informative: CICDDoS2019 and CICIDS2017 represent enterprise-style traffic with a moderate, well-documented feature count around eighty; BCCC-cPacket-Cloud-DDoS-2024 represents a cloud environment with a substantially richer feature set drawn from over three hundred network- and transport-layer measurements; CICIoT2023 represents heterogeneous, resource-constrained IoT traffic with a comparatively small feature count reflecting the limited telemetry such devices typically expose; and ITDDoS2020 and APA-DDoS, despite the gap in their provenance documentation, still usefully represent IoT/enterprise and advanced-network conditions with feature counts (83 and 23 respectively) distinct from the other four. A model that performs well across all six is being tested against a meaningfully different combination of traffic composition and feature richness each time, which is a stronger claim than repeated evaluation against variants of a single dataset family — a point returned to in Section 2.7's discussion of how thinly cross-dataset validation is actually practised across the studies reviewed in this chapter.

## 2.6 Comparative Summary Table

Table 2.6 consolidates every study discussed in Sections 2.1–2.4, ordered as before with hybrid, attention-, and Transformer-based detectors first, plus the two classical machine-learning studies.

**Table 2.6:** Consolidated Literature studies

| Ref | Author(s),Yr | Publisher | Technique / Approach | Dataset(s) | Reported Performance | Key Limitations |
|---|---|---|---|---|---|---|
| [22] | Zhao et al. 2023 | IEEE | CNN-BiLSTM + attention (CNN-AttBiLSTM) | CICDDoS2019 | High acc.; attention improved discrimination | Compute cost of hybrid; single dataset |
| [23] | Kanthimathi et al. 2024 | IEEE | Self-attention weighted-ensemble CNN | CICDDoS2019 | Acc 98.32%, F1 98.31% | Needs large data; heavy compute |
| [7] | Kirubavathi et al. 2025 | Springer | Self/intersample-attention Transformer (SAINT) | BCCC-Cloud-2024 | ~97% acc.; scalable, interpretable | TCP-focused; cloud testbed only |
| [29] | Al-Eryani et al. 2025 | Springer | GRU-BiLSTM hybrid | CICDDoS2019 | Acc 99.99%, FPR 0.01 | Sequential-heavy; cross-domain untested |
| [30] | Zaidoun & Lachiri 2025 | Springer | Hybrid DNN-LSTM + attention | CICDDoS2019 | Val ≈99%, Test 98.84% | Single-dataset validation |
| [19] | Wu et al. 2022 | IEEE | Transformer IDS (RTIDS) | NSL-KDD/CICIDS | Competitive transformer IDS | General IDS, not L7-specific |
| [20] | Manocchio et al. 2024 | Elsevier | FlowTransformer framework | Multiple flow sets | Generalizes transformer IDS | Flow-level abstraction |
| [21] | Djaidja et al. 2024 | IEEE | Attention + RNNs (early detection) | IDS benchmark | Sharpened early detection | Not L7-specific |
| [27] | Pradeep & Shukla 2025 | Elsevier | Stacked network + optimal FS | DDoS benchmark | Acc 98.5%, F1 98.0% | Hyperparameter-sensitive; no cost analysis |
| [31] | Al-Shukaili et al. 2025 | Springer | MI + RFE + hybrid deep learning | CIC-LDDoS | Acc 97.8%; ~18% FN reduction | Limited multi-category generalization |
| [24] | Sharif & Beitollahi 2023 | Elsevier | ML + genetic-algorithm FS | App-layer DDoS set | GA selection improved accuracy | Static selection; GA cost |
| [28] | Yu et al. 2024 | Springer | BiGAN + contrastive learning | Anomaly dataset | Acc 97.2%, F1 96.9% | High compute; narrow validation |
| [14] | Cil et al. 2024 | Elsevier | Feed-forward DNN | CICDDoS2019 | Acc 0.9997 (binary) | Degrades on multi-class |
| [15] | de Assis et al. 2021 | Elsevier | GRU deep-learning system | CICDDoS2019 /2018 | Avg. metrics 99.94% | Detection rate & time not reported |
| [17] | Amaizu et al. 2021 | Elsevier | Dual DNN + feature extraction (B5G) | B5G traffic | Acc 99.66%, R 99.30% | Weak real-time performance |
| [11] | Muraleedharan & Janet 2021 | Elsevier | FC network on flow data | CICIDS2017 | Acc 99.61% (slow-DoS) | Limited to HTTP slow-DoS |

| | | | | | | |
|---|---|---|---|---|---|---|
| [26] | Kemp et al. 2023 | Springer | Statistical features + ML | App-layer DoS set | Acc 96.5%, F1 95.9% | Limited dataset; high-traffic unvalidated |
| [25] | Akgun et al. 2022 | Elsevier | Deep-learning IDS model | IDS benchmark | Strong multi-class detection | Single-dataset; thin low-rate coverage |
| [12] | Fu et al. 2022 | Springer | Time-frequency analysis (low-rate) | Low-rate DoS traces | Detects periodic low-rate bursts | Narrow to low-rate class |
| [10] | David & Thomas202021 | Elsevier | Information-distance thresholding | Flash-crowd vs DDoS | Separates attack from flash crowd | Threshold sensitive to drift |
| [9] | Praseed & Thilagam 2021 | IEEE | Behavioural-dynamics modelling | HTTP traces | Effective for asymmetric L7 floods | Needs per-application baseline |
| [13] | Doriguzzi-Corin et al. 2020 | IEEE | Lightweight CNN (LUCID) | CICIDS2017 | Acc 0.9967, FPR 0.0059 | Processing time not reported |
| [16] | Haider et al. 2020 | IEEE | Deep CNN ensemble | CICIDS2017 | Acc 99.45%, F1 99.61% | High training time |
| [18] | Mittal et al. 2023 | Springer | Systematic review of DL for DDoS | — (review) | Maps DL landscape & gaps | Review; no new system |
| [34] | Abiramasundari & Ramaswamy 2025 | Springer | PCA + SVM/LR/RF/kNN/DT (EDAD) | Custom SDN set | Improved acc. across 5 classifiers | Fixed linear projection |
| [35] | Sawah et al. 2025 | Springer | RF/NB/kNN/LDA/SVM + BE + GridSearch | DDoS-SDN set | RF acc. 99.99% | Fixed feature set; SDN-specific |

Two patterns stand out from the table as a whole, beyond what any individual row shows. First, reported accuracy figures cluster overwhelmingly above 97%, and several sit at or above 99.9%, yet this clustering itself is informative rather than reassuring: with so many independent architectures converging on near-perfect scores against a small set of shared benchmarks, the benchmarks themselves — rather than any remaining architectural refinement — are plausibly closer to saturation than the attack problem is to being solved, a concern sharpened by how few of the 26 studies validate across more than one dataset family. Second, the Key Limitations column shows a narrow, recurring vocabulary — single-dataset validation, untested cross-domain generalisation, unreported computational cost, static or dataset-wide feature selection — appearing across studies that otherwise differ substantially in architecture and attack focus, which suggests these are structural properties of how the field currently evaluates itself rather than idiosyncrasies of any one paper.

A third pattern emerges from reading the Year and Publisher columns together rather than in isolation. Ordering the 26 rows chronologically shows a visible acceleration: only six of the studies date from 2020–2021, a further ten from 2022–2023, and the remaining ten from 2024–2025, which tracks the broader shift toward attention- and Transformer-based methods documented qualitatively in Section 2.3 — the newest studies in the table are disproportionately the ones adopting attention in some form. The publisher distribution is comparatively even across IEEE (eight studies), Elsevier (nine studies), and Springer (nine studies), which is reassuring from a review-methodology standpoint: the trends identified in this chapter are not an artefact of over-sampling any single publisher's editorial preferences, but recur consistently across venues with different reviewing communities and, presumably, different reviewers.

A fourth reading of the table, grouping rows by which dataset they rely on rather than by technique or year, reinforces the benchmark-saturation concern raised above with a concrete count. Nine of the 26 rows — including [22], [23], [29], [30], and [14] — use CICDDoS2019 as their sole or primary evaluation dataset; a further four use CICIDS2017. Together, these two CIC-lineage datasets account for roughly half of all reported results in Table 2.7, which means the field's collective sense of what constitutes strong performance has been shaped disproportionately by how detectable these two specific datasets' attack patterns happen to be, rather than by a broad sample of the traffic environments described in Section 2.5. This is not a criticism of any individual study's choice of dataset — CICDDoS2019 and CICIDS2017 are, after all, the most thoroughly documented and widely accepted benchmarks available, which is exactly why they are so heavily reused — but it is a structural reason to treat the clustering of near-perfect accuracy scores in Table 2.6 with some caution, and it is the direct motivation for validating the framework proposed in this thesis across all five of the datasets catalogued in Section 2.5 rather than reporting results on CICDDoS2019 alone.

## 2.7 Synthesis: Where the Research Trajectory Stalls

Read as a whole rather than study by study, Table 2.7 traces a trajectory that moves in one consistent direction across three successive stages. The earliest stage, comprising the classical machine-learning studies in Section 2.1 and the single-architecture CNN and DNN studies among rows 22–26 of Table 2.6, treats DDoS detection as a static

classification problem: features are engineered or selected once, a single model is trained on them, and performance is reported on the same dataset the model was tuned against. The second stage, comprising the recurrent and generative hybrids discussed in Section 2.2 and rows 12 and 14–15 of the table, replaces the static classifier with an architecture that can model how traffic evolves over a session, which measurably improves detection of attacks — slow-DoS, low-rate floods — whose signature only emerges over time. The third and most recent stage, comprising the attention- and Transformer-based studies in Section 2.3 and rows 1–8 of the table, adds a mechanism for weighting different parts of the input differently, which the reviewed evidence shows measurably sharpens discrimination again, particularly for traffic engineered to imitate legitimate load.

This progression is genuine and each stage's improvement over the last is reasonably well evidenced by the studies reviewed here, but two things are equally clear from reading the limitations column across all 26 rows rather than any single row in isolation. First, feature selection has not kept pace with the architectural progression it sits alongside: whether the underlying classifier is a decision tree, a CNN, a BiLSTM, or a Transformer encoder, the feature set it is given is still overwhelmingly chosen by a static, pre-training procedure — MI, ANOVA, RFE, PCA, or a genetic-algorithm wrapper that commits to one feature subset for every subsequent instance regardless of how that instance's own traffic pattern differs from the training distribution. Second, and more specifically, attention itself the technique with the clearest conceptual capacity to make feature relevance instance-dependent rather than dataset-wide has consistently been implemented as a refinement bolted onto an existing CNN or RNN pipeline (Section 2.3, groups A and B), as a full replacement for that pipeline (group C), or as a second axis alongside self-attention (group D), but never as the mechanism that performs adaptive, per-instance feature weighting ahead of, and jointly with, a spatial and a temporal representation stage within one unified architecture. This is the specific point, identified already in the critical analysis of [5], [18] and confirmed independently across every subsection of this chapter, at which the field's steady architectural progress stalls.

A second, more practical stall accompanies the architectural one. Across all 26 studies, evaluation is reported almost exclusively in terms of overall accuracy, precision, recall, and F1-score, with only a small minority reporting imbalance-aware measures such as

Balanced Accuracy, Matthews Correlation Coefficient, Cohen's Kappa, or G-Mean, despite the fact that DDoS datasets are, by their nature, heavily imbalanced between benign and attack classes. Fewer still report inference latency or training cost in a form that would let a reader judge real-time deployability, and cross-dataset validation — testing a model trained on one benchmark against a structurally different one — is close to absent; where it is attempted at all, it typically spans variants of a single dataset family (as with the CICDDoS2019-based studies in rows 2, 4, 5, and 13 of Table 2.6) rather than genuinely distinct environments such as cloud, IoT, and enterprise traffic evaluated side by side.

It is worth being precise about what the imbalance-aware metrics named above actually add over accuracy and F1-score, since the distinction is central to why their near-absence from the reviewed literature matters. Overall accuracy simply counts the proportion of all instances correctly classified, which means a dataset that is 95% benign can be classified with 95% accuracy by a model that never once correctly flags an attack, simply by predicting “benign” unconditionally. Balanced Accuracy instead averages the recall achieved on each class separately before combining them, so a model's score is dragged down if it neglects the minority class regardless of how small that class is; the Matthews Correlation Coefficient goes further still, incorporating all four cells of the confusion matrix — true and false positives and negatives together — into a single correlation-like statistic that is close to zero for a classifier no better than chance regardless of class balance. Cohen's Kappa measures agreement between predicted and true labels after subtracting out the agreement that would be expected from chance alone given the observed class frequencies, and G-Mean takes the geometric mean of per-class recall, which, like Balanced Accuracy, penalises a model heavily for neglecting any one class rather than letting strong majority-class performance compensate for weak minority-class performance in an average. Reporting accuracy or F1-score alone, as the large majority of the 26 studies in Table 2.6 do, is therefore not simply reporting a less detailed metric — it is a choice that can actively conceal exactly the failure mode, poor minority-class detection, that matters most for an attack category that is rare in the training data but potentially severe in its real-world impact.

It is useful to trace each of the six research gaps identified in the critical analysis back to the specific evidence this chapter has assembled for it, since the gaps were stated

there as conclusions and can now be shown to follow from the studies reviewed section by section rather than asserted independently of them. Table 2.7 makes this traceability explicit.

**Table 2.7:** Research gaps mapped to supporting evidence

| | Research Gaps | Supporting evidence in this chapter |
|---|---|---|
| 1 | Static feature-selection methods cannot adjust feature importance dynamically during inference | 2.1 (PCA/backward-elimination studies [34],[35]) and 2.4 (MI/ANOVA/RFE/PCA/GA [24],[27],[31]) show every surveyed selection method fixes its output before inference and cannot revisit it per instance |
| 2 | Transformer-based attention for adaptive feature weighting in application-layer DDoS detection is under-explored | 2.3's four-group taxonomy (Fig. 2.2) shows attention used as add-on refinement, early-detection aid, full backbone replacement, or a second attention axis — never as an explicit pre-classification feature-weighting stage |
| 3 | Cross-dataset validation on recent benchmarks is insufficient | 2.6's table shows the large majority of the 26 studies validate on a single dataset or a single dataset family (CICDDoS2019 variants); 2.5 shows two of the five benchmarks used in this thesis lack even a peer-reviewed origin paper, evidencing benchmark fragmentation |
| 4 | Many models emphasize overall accuracy while overlooking imbalance-aware measures | 2.6's Key Limitations column and the synthesis above show accuracy/F1 dominate reporting; Balanced Accuracy, MCC, Kappa, and G-Mean appear in only a minority of the 26 rows |
| 5 | CNN, BiLSTM, and Transformer attention are seldom unified within a single end-to-end architecture | 2.2 and 2.3 show CNN-only, BiLSTM-only, CNN+BiLSTM, and attention-augmented variants of each, but no surveyed study unifies all three as one jointly trained pipeline |
| 6 | Computational efficiency and explainability are rarely investigated together | 2.3's discussion of quadratic attention cost versus SAINT's bounded batch-wise cost [7], and the Key Limitations column across Tables 2.3–2.5, show cost and interpretability reported separately, if at all, rather than jointly |

This traceability matters for how the remainder of the thesis is read: Chapter 3 formalises gaps 1, 2, and 5 into the specific problem statement and threat model the proposed framework targets, while gaps 3, 4, and 6 are carried forward into the evaluation methodology of Chapter 5, which is designed specifically to report imbalance-aware metrics and cross-dataset results of the kind this chapter has shown to be largely absent from the existing literature.

Taken together, these two stalls — static rather than adaptive feature weighting, and narrow rather than imbalance-aware, cross-environment evaluation — define the specific gap this thesis addresses. They motivate a Transformer-based Attention CNN–BiLSTM framework in which the attention mechanism performs adaptive, instance-wise feature weighting ahead of a joint spatial–temporal representation, evaluated using the fuller set of imbalance-aware metrics introduced in Section 1.3 and Chapter 3, and validated not on one benchmark but across the five environments — enterprise, cloud, IoT, IoT/enterprise, and advanced-network — represented by the datasets documented in Section 2.5. The detailed architecture, and the reasoning connecting each of its components back to a specific limitation identified in this chapter, is presented in full in Chapter 4.

In summary, this chapter has moved from the comparative table to a full narrative account of why classical machine learning, deep learning, attention-based hybrids, and static feature selection each improved on what preceded them, and has identified, through the taxonomy in Figure 2.1, the four design-space diagrams and tables in Sections 2.1–2.5, the consolidated 26-row comparison in Table 2.6, and the gap-traceability analysis in Table 2.7, a single specific and well-evidenced point at which that progression currently stalls: the absence of an architecture in which attention performs adaptive, per-instance feature weighting jointly with, rather than as a refinement of or a replacement for, a spatial (CNN) and a temporal (BiLSTM) representation stage. Chapter 3 takes this gap as its starting point, formalising it into a precise problem statement, a threat model appropriate to application-layer DDoS traffic specifically, and the set of evaluation criteria — spanning classification accuracy, imbalance-aware reliability, and computational cost — against which the framework proposed in Chapter 4 is subsequently judged.

# CHAPTER 3: PROBLEM DEFINITION AND RESEARCH OBJECTIVES

Chapter 2 traced the trajectory of DDoS detection research from classical machine learning through deep learning to attention- and Transformer-based methods, and closed by identifying, through Table 2.7, six specific research gaps together with the chapter-and-section evidence supporting each. This chapter takes that traceable evidence base as its starting point rather than re-deriving it: Section 3.1 expands each of the six gaps into a substantiated argument grounded in named studies from Chapter 2; Section 3.2 distils these gaps into a single formal problem statement; Section 3.3 translates the problem statement into explicit research questions; Section 3.4 states the four research objectives that answer those questions and justifies each against the gap it closes; Section 3.5 fixes the scope within which the objectives are pursued, stating explicitly what is excluded; and Section 3.6 sets out the expected outcomes against which Chapter 5's results are later judged. Figure 3.1 in Section 3.4 makes the resulting chain — from gap, to objective, to outcome — explicit and traceable in one view.

## 3.1 Research Gap Analysis

The critical analysis condensed Chapter 2's review into six research gaps. Each is restated below not as an assertion but as an argument, tracing the specific studies whose limitations, taken together, establish that the gap is real rather than a rhetorical framing device.

### 3.1.1 Gap 1 — Static Feature Selection Methods

Every feature-selection study reviewed in Section 2.4 settles on one ranking or subset before training finishes and then holds that decision fixed for every instance the model later sees. Sharif and Beitollahi's genetic-algorithm search [24] converge on a single feature subset before deployment; Al-Shukaili, Kiah, and Ahmedy's combined Mutual-Information-and-RFE pipeline [31] and Pradeep and Shukla's optimisation-embedded selection [27] behave identically in this respect, differing only in how the fixed subset is computed, not in whether it can subsequently adapt. Section 2.1's classical-ML studies confirm the same pattern from the projection and elimination side: Abiramasundari and Ramaswamy's PCA-based framework [34] fixes a set of linear components once and for all, and Sawah et al.'s backward-elimination-plus-grid-search pipeline [35] permanently discards the eliminated features before the random forest

ever sees a test instance. Because all five studies commit to their feature decision before inference begins, none of them can recover a feature that turns out to be decisive for a rare or atypical instance but uninformative on average across the training set — the specific failure mode this gap describes.

### 3.1.2 Gap 2 — Limited usage of Transformer-Based Attention Methods

Section 2.3's four-group taxonomy shows attention consistently occupying a narrower role than adaptive, per-instance feature weighting across an entire application-layer detection pipeline. Zhao et al. [22] adds attention as a refinement layer within an existing CNN-BiLSTM pipeline, and Kanthimathi et al. [23] lets self-attention re-scale how much each CNN sub-model's output counts toward the ensemble's final vote — in both cases attention adjusts a representation the rest of the pipeline has already built, rather than being the stage that first decides which features matter. Djaidja et al. [21] apply attention together with RNNs specifically to sharpen early detection on a general intrusion-detection benchmark, not an application-layer DDoS set. Wu et al.'s RTIDS [19] and Manocchio et al.'s FlowTransformer [20] build genuinely Transformer-native encoders, but evaluate them as general-purpose intrusion detectors or flow-level abstractions rather than as mechanisms for application-layer feature weighting specifically. Kirubavathi et al.'s SAINT model [7] comes closest, pairing self-attention with a second, intersample attention axis, but its own evaluation is confined to TCP-based attacks within a single cloud testbed. None of these six studies uses attention as an explicit feature-weighting stage validated specifically against the traffic-mimicking application-layer attacks described in Chapter 1 — which is precisely what this gap identifies as missing.

### 3.1.3 Gap 3 — Inadequate Cross-Dataset Validations

Section 2.6's reading of Table 2.6 by dataset dependency showed that roughly half of the 26 tabulated studies rely on CICDDoS2019 or CICIDS2017 as their sole or primary evaluation benchmark. Al-Eryani et al. [29], Zaidoun and Lachiri [30], and Cil et al. [14] report their strongest results against CICDDoS2019 specifically and are not validated elsewhere; de Assis et al. [15] additionally test against CICIDS2018, but that dataset shares the same CIC enterprise-traffic lineage as CICDDoS2019 rather than representing a structurally distinct environment such as a cloud or IoT-native dataset, so none of the four studies is validated outside this one dataset family. Section 2.5

compounds this concern from the dataset side rather than the study side: two of the five benchmarks this thesis relies on for its own cross-environment testing — ITDDoS2020 and APA-DDoS, a fragmentation problem documented independently by Pinto et al.'s review of intrusion-detection datasets [33]. A field in which half its strongest reported results rest on two related datasets, two of five common benchmarks lack verifiable provenance, and cross-environment validation is rarely attempted together, has not yet established that its reported accuracy generalises beyond the specific traffic each model happened to be tuned on.

### 3.1.4 Gap 4 — Accuracy-Centric Performance Assessment

Haider et al. [16] report accuracy and F1-score only; Doriguzzi-Corin et al.'s LUCID [13] reports accuracy, false-positive rate, and F1-score but no Balanced Accuracy, MCC, Kappa, or G-Mean; Al-Eryani et al. [29] report accuracy and false-positive rate in the same limited register. This is not an isolated pattern: of the 26 studies Chapter 2 brought together in one table, only a small minority report any of the four imbalance-aware measures introduced in Section 2.7, despite DDoS datasets being characteristically dominated by one or two majority classes. As Section 2.7 set out in first-principles terms, a classifier can report a high overall accuracy while performing poorly on minority attack classes precisely because accuracy alone cannot distinguish the two situations — which means the reported strength of the field's near-perfect accuracy scores has not, for the great majority of the studies surveyed, been checked against the metrics designed to catch this specific failure mode.

### 3.1.5 Gap 5 — Lack of Unified Hybrid Architecture

Section 2.2 and Section 2.3 together show every combination of these three components represented somewhere in the literature, but never all three jointly. Doriguzzi-Corin et al. [13] and Haider et al. [16] use CNN alone; de Assis et al. [15] and Al-Eryani et al.'s GRU-BiLSTM hybrid [29] use recurrent components alone, without a CNN spatial stage or an attention-weighting stage; Zhao et al. [22] combine CNN, BiLSTM, and attention, but as an add-on refinement rather than an adaptive weighting stage (Gap 2); Kirubavathi et al.'s SAINT [7] is Transformer-native but includes neither a CNN spatial stage nor a BiLSTM temporal stage. No study reviewed in Chapter 2 integrates adaptive attention-based feature weighting, CNN-based spatial extraction, and BiLSTM-based

temporal modelling as three jointly trained stages of one architecture — each pairing exists somewhere in the literature, but the full triple does not.

### 3.1.6 Gap 6 — Limited Computational Interpretability Analysis

Section 2.3's discussion of quadratic attention cost showed that Wu et al. [19] and Manocchio et al. [20] report the elevated training and inference cost characteristic of self-attention over long flow sequences, without reporting any measure of interpretability alongside it. Kirubavathi et al. [7] make the opposite trade in reporting emphasis, presenting scalability and interpretability as explicit design goals for SAINT, yet — as Section 2.3 noted — the interpretability claim rests on the architecture's design rather than on an explainability metric the authors actually measured. Pradeep and Shukla [27] report accuracy and F1-score for their optimisation-embedded feature selection but explicitly flag the absence of a computational cost analysis as a limitation of their own study. Across these three cases, inference time and interpretability are each reported by at least one study, but never jointly measured within the same study — which is precisely what would be needed to judge whether an architecture is both fast enough and transparent enough for real-world deployment.

## 3.2 Problem Statement

The six gaps in Section 3.1 are not independent complaints about six different papers; read together, they describe one coherent shortfall in how application-layer DDoS detection is currently approached, which this section states formally as a single problem.

Application-layer DDoS attacks are constructed specifically to resemble legitimate traffic — a property established in Chapter 1 and reaffirmed by the studies in Chapter 2 that struggle precisely where mimicry is strongest. Detecting such attacks reliably requires a model that can judge, for each incoming traffic instance individually, which of its features are actually discriminative, since the features that reveal one attack pattern are not necessarily the features that reveal another. Gaps 1 and 2 (Section 3.1.1–3.1.2) show that the current literature does not provide this: feature relevance is fixed once, dataset-wide, whether the fixing mechanism is a classical filter, a wrapper, a projection, a genetic search, or attention used only as an internal refinement layer rather than an explicit weighting stage. Gap 5 (Section 3.1.5) shows that even where attention is used more centrally, it is never combined with both a CNN-based view of local traffic

structure and a BiLSTM-based view of how that traffic unfolds over time, within one jointly trained architecture, despite Chapter 2 showing each of these components to be independently valuable — CNNs for local feature co-occurrence, BiLSTM for how a session evolves over time. The result is that no existing architecture lets a spatial (CNN) view of traffic and a temporal (BiLSTM) view of the same traffic both feed off a shared, per-instance sense of which features matter.

This architectural shortfall is compounded by an evaluative one. Gaps 3 and 4 (Section 3.1.3–3.1.4) show that even the strongest reported results in the current literature are usually obtained against a single dataset or a closely related family of datasets, and are reported using overall accuracy and F1-score far more often than the imbalance-aware measures that would reveal whether a model's performance is uniform across majority and minority classes alike. Gap 6 (Section 3.1.6) shows that even where a study reports either computational cost, inference latency or an interpretability argument, it is rare for a single study to report both or any one of it, which leaves a genuine gap in the evidence available to judge whether a proposed detector is deployable in practice rather than merely accurate on a benchmark. Consequently, a strong reported accuracy figure in the current literature does not, on its own, establish that a detector generalises across environments, treats minority attack classes reliably, or is efficient and interpretable enough for real-world use.

The problem this thesis addresses can therefore be stated precisely as follows: Existing application-layer DDoS detection approaches apply fixed, uniform feature-relevance decisions regardless of the traffic instance, remain architecturally fragmented by not jointly integrating adaptive attention-based feature weighting with spatial (CNN) and temporal (BiLSTM) modelling, and are evaluated mainly on single benchmarks using accuracy-centric metrics that overlook class-imbalance-sensitive reliability and joint inference latency. This calls for a unified mechanism performing adaptive, instance-wise feature weighting alongside spatial-temporal modelling, and an evaluation methodology reporting imbalance-aware reliability, cross-dataset generalisation, and inference latency together. This thesis addresses that need through the Transformer-based Attention CNN–BiLSTM framework introduced in Chapter 1 and detailed in full in Chapter 4, evaluated according to the methodology set out in Chapter 5.

## 3.3 Research Questions

The problem statement in Section 3.2 is broad enough to require decomposition into specific, answerable questions before it can guide an architecture or an experimental design. The five research questions below are each derived from a specific subset of the gaps identified in Section 3.1, and are answered, in turn, by a specific objective in Section 3.4.

RQ1. Can an attention mechanism be structured to perform adaptive, instance-wise feature weighting for application-layer traffic, rather than functioning only as an internal refinement layer within an otherwise conventional architecture? This question responds directly to Gaps 1 and 2 (Sections 3.1.1–3.1.2), which established that existing feature-selection methods and existing uses of attention alike commit to a fixed notion of feature relevance rather than recomputing it per instance.

RQ2. Does jointly integrating adaptive attention-based feature weighting with CNN-based spatial extraction and BiLSTM-based temporal modelling, within a single trained architecture, improve detection performance over architectures that use only a subset of these three components? This question responds to Gap 5 (Section 3.1.5), which found every pairwise combination of these components represented in the literature but never the full triple within one end-to-end design.

RQ3. How does the proposed framework's detection performance compare with conventional static feature-selection methods — specifically Mutual Information, ANOVA, and Recursive Feature Elimination — when both are evaluated using the same classifier backbone and the same dataset? This question operationalises Gap 1 (Section 3.1.1) into a direct, controlled comparison rather than a general architectural claim, and is answered experimentally in Chapter 5 using the comparative methodology introduced in Section 2.4.

RQ4. Does the proposed framework's performance, measured using imbalance-aware metrics (Balanced Accuracy, MCC, Cohen's Kappa, G-Mean) alongside conventional accuracy and F1-score, remain robust when the model is validated across multiple, structurally distinct benchmark datasets spanning enterprise, cloud, IoT, and advanced-network environments? This question responds jointly to Gaps 3 and 4 (Sections 3.1.3–3.1.4), which found cross-dataset validation and imbalance-aware reporting each to be rare, and rarer still in combination.

RQ5. What is the inference latency — measured as traffic classification time and per-instance inference latency — of the proposed adaptive attention mechanism relative to the accuracy and reliability gains it provides, and can this cost be characterised precisely enough to judge the framework's suitability for near-real-time deployment? This question responds to Gap 6 (Section 3.1.6), which found inference latency rarely reported together, let alone weighed explicitly against the gains they are the price of.

Together, RQ1 and RQ2 concern the architecture itself; RQ3 and RQ4 concern how that architecture should be evaluated against existing alternatives and across environments; and RQ5 concerns the practical cost of adopting it. This grouping is reflected directly in how the four objectives are structured in Section 3.4.

## 3.4 Objectives of the Research

The primary objective of this research is to develop an adaptive deep learning framework capable of improving the detection of application-layer DDoS attacks through dynamic feature learning and hybrid deep neural network modelling. This primary objective is pursued through four specific objectives, each answering one or more of the research questions in Section 3.3 and each justified below against the specific gap it is designed to close.

Objective 1. To design and develop a Transformer-based Attention CNN–BiLSTM framework for accurate detection of application-layer DDoS attacks by jointly learning spatial, temporal, and contextual characteristics of network traffic. This objective answers RQ1 and RQ2 directly, and is justified by Gaps 2 and 5 (Sections 3.1.2 and 3.1.5): Chapter 2 found attention, CNN, and BiLSTM each used effectively in isolation or in pairs, but never unified as three jointly trained stages of one architecture, which is the specific structural absence this objective is designed to fill rather than a generic call for architectural novelty.

Objective 2. To implement an adaptive feature-weighting mechanism using Transformer attention that dynamically identifies feature importance during training, eliminating dependence on static selection. This objective answers RQ1 and is justified by Gap 1 (Section 3.1.1): Section 2.4's information-theoretic argument showed that every static feature-selection method surveyed — filter, wrapper, projection, or search-based — computes one relevance estimate and applies it uniformly regardless of the instance being classified, which is exactly the constraint an attention mechanism, used

as an explicit weighting stage rather than an internal refinement layer, is structurally positioned to remove.

Objective 3. To evaluate the framework using comprehensive imbalance-aware metrics: Accuracy, Precision, Recall, F1-score, Balanced Accuracy, MCC, Cohen's Kappa, ROC-AUC, G-Mean, FPR, FNR, and inference time. This objective answers RQ4 and RQ5, and is justified jointly by Gaps 4 and 6 (Sections 3.1.4 and 3.1.6): Section 2.7 showed that most of what Table 2.7 reports is accuracy and F1-score without the imbalance-aware measures that would reveal minority-class performance, and that inference letency are seldom reported by the same study — including inference time alongside imbalance-aware metrics in a single evaluation methodology directly closes both omissions at once rather than treating them as separate concerns.

Objective 4. To validate the robustness and generalisation of the framework through experimentation on contemporary benchmark datasets. This objective answers RQ4 and is justified by Gap 3 (Section 3.1.3): Section 2.6's dataset-dependency analysis found roughly half of the studies reviewed validated on CICDDoS2019 or CICIDS2017 alone, and Section 2.5 catalogued five structurally distinct environments — enterprise, cloud, IoT, IoT/enterprise, and advanced-network — precisely so that this objective can be pursued across genuinely different traffic conditions rather than across variants of one dataset family.

Figure 3.1 traces each of the six gaps identified in Section 3.1 through the objective that addresses it to the expected outcome, detailed in Section 3.6, that would constitute evidence the objective has been met. The diagram is deliberately many-to-one in places — Gaps 2 and 5 both converge on Objective 1, for instance, because both concern the same architectural absence viewed from different angles — which is itself informative: it shows that four objectives are sufficient to address six distinct gaps because several gaps share a common root cause rather than requiring six independent interventions.

## 3.5 Scope and Delimitations

Defining what this research does not attempt is as important as defining what it does, since several of the gaps in Section 3.1 could otherwise be read as inviting a broader program of work — attack mitigation, live deployment, or defence against every category of denial-of-service attack — than the four objectives in Section 3.4 actually commit to. This section fixes the boundary explicitly.

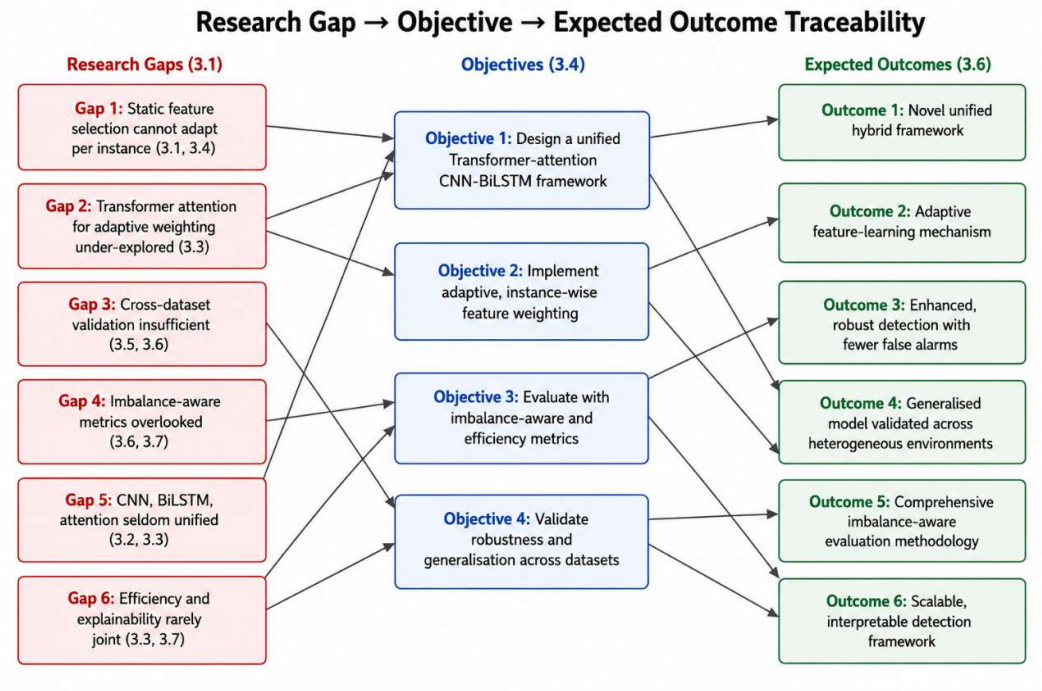



***Figure 3.1:*** *Traceability from the six research gaps (3.1), through the four research objectives (3.4), to the six expected outcomes (3.6)*

The research is scoped to the detection and classification of application-layer DDoS attacks using an adaptive deep learning framework built on Transformer attention, CNN, and BiLSTM components, as set out in Objective 1; it is not a general-purpose intrusion-detection system and is not evaluated against, or claimed to be effective for, non-DDoS intrusion categories such as brute-force, infiltration, or port-scanning traffic, even where those categories appear in some of the benchmark datasets discussed in Section 2.5.

The proposed framework employs dynamic, attention-based feature weighting learned during training, as set out in Objective 2, specifically to avoid dependence on the static selection techniques catalogued in Section 2.4; the comparative evaluation against Mutual Information, ANOVA, and Recursive Feature Elimination, described further below, is included precisely to demonstrate this design choice's effect rather than to conduct an exhaustive survey of every feature-selection technique in the literature.

Experimental evaluation is conducted using five contemporary benchmark datasets — CICDDoS2019, BCCC-cPacket-Cloud-DDoS-2024, CICIOT2023, ITDDoS2020, and APA-DDoS — chosen specifically, per Section 2.5's dataset-coverage synthesis, to span enterprise, cloud, IoT, IoT/enterprise, and advanced-network conditions, in direct

response to Gap 3 (Section 3.1.3). This is a delimitation as well as a scope commitment: the generalisation claims made in Chapter 5 and Chapter 6 extend only to the traffic conditions these five datasets represent, and any claim of broader generalisation to network environments not resembling these five — satellite networks, industrial control systems, or 5G core-network signalling traffic, for instance — lies outside what this thesis's experiments can support.

Performance is analysed using accuracy, precision, recall, F1-score, Balanced Accuracy, ROC-AUC, MCC, Cohen's Kappa, G-Mean, false positive rate, false negative rate, and inference latency, per Objective 3; this comprehensive metric set is itself a delimitation, since it fixes in advance the criteria by which success is judged rather than leaving the evaluation open-ended, and it deliberately excludes metrics tied to specific deployment contexts — such as energy consumption per inference on a named embedded platform — that would require a hardware-specific study beyond this thesis's scope.

The research includes a comparative performance analysis against Mutual Information, ANOVA, and Recursive Feature Elimination specifically, rather than against every static feature-selection method catalogued in Section 2.4; Principal Component Analysis and genetic-algorithm-based selection are discussed as part of the literature review in Chapter 2 but are not re-implemented as experimental baselines in Chapter 5, since MI, ANOVA, and RFE between them already represent the filter, statistical, and wrapper families of static selection and provide sufficient contrast for RQ3 without duplicating every method surveyed.

Finally, and most importantly, the scope of this work is explicitly limited to the detection and classification of DDoS attacks using benchmark datasets; it does not extend to attack mitigation (automatically blocking, rate-limiting, or rerouting traffic once an attack is detected), to traffic filtering or firewall-rule generation, or to deployment and evaluation in an operational production network environment. A trained and validated classifier is the end product of this research; integrating that classifier into a live mitigation pipeline, testing it against adversarial traffic engineered specifically to evade it, and evaluating it under production-scale concurrent load are all identified in Chapter 7 as directions for future work rather than commitments of the present thesis. This delimitation follows directly from Section 1.4's non-technical

overview of the proposed approach, which frames the contribution as a detection mechanism rather than an end-to-end defence system.

## 3.6 Expected Outcomes

The six expected outcomes below correspond to the six right-hand endpoints of Figure 3.1 and are stated here as the specific, checkable claims Chapter 5's experimental results are subsequently evaluated against — each is a prediction this thesis commits to, not a retrospective description of results already obtained.

Outcome 1. A novel hybrid deep learning framework integrating Transformer-based attention, CNN, and BiLSTM within a single end-to-end architecture, directly fulfilling Objective 1 and closing Gaps 2 and 5. Chapter 2 found each pairwise combination of these components in the literature but never the full triple; this outcome is met only if the resulting architecture, detailed in Chapter 4, is trained and evaluated as one jointly optimised model rather than as separately trained components combined at inference time.

Outcome 2. An adaptive feature-learning mechanism capable of dynamically assigning feature importance per instance, improving on the fixed representations produced by conventional static feature selection, directly fulfilling Objective 2 and closing Gap 1. This outcome is met only if the attention-derived feature weights demonstrably vary across different input instances in a way that a static selection method's fixed ranking cannot, and if this variability translates into a measurable performance difference against the MI, ANOVA, and RFE baselines specified in Section 3.5.

Outcome 3. Enhanced detection performance characterised by improved robustness, more balanced classification across majority and minority classes, and reduced false-alarm rates across diverse application-layer DDoS scenarios, following jointly from Objectives 1 and 2. Because this outcome concerns balance across classes specifically rather than overall accuracy alone, it is assessed using the imbalance-aware metrics named in Outcome 5 below rather than accuracy in isolation, consistent with Gap 4's finding that accuracy alone can conceal exactly this kind of imbalance.

Outcome 4. A generalised detection model, validated on the five contemporary benchmark datasets described in Section 3.5, demonstrating adaptability across heterogeneous network environments, directly fulfilling Objective 4 and closing Gap 3. This outcome is met only if performance across the enterprise, cloud, IoT,

IoT/enterprise, and advanced-network datasets remains within a consistent, reportable range rather than being strong on one dataset and markedly weaker on another — the specific pattern Section 2.6 found largely untested in the existing literature.

Outcome 5. A comprehensive performance evaluation using accuracy, precision, recall, F1-score, Balanced Accuracy, ROC-AUC, MCC, Cohen's Kappa, G-Mean, FPR, FNR, and inference latency, establishing the effectiveness of the proposed framework beyond what accuracy or F1-score alone could show, directly fulfilling Objective 3 and closing Gap 4. This outcome is met by the evaluation methodology itself being applied consistently across every experiment in Chapter 5, not merely by any single metric reaching a particular value.

Outcome 6. A scalable and interpretable intrusion-detection framework that strengthens application-layer service security and provides a foundation for future intelligent network-security research, following from Objective 3's inclusion of inference-time measurement and from the attention mechanism's inherent capacity, discussed in Chapter 1, to expose which features drove a given decision. This outcome directly closes Gap 6: it is met only if both scalability (via reported training and inference cost) and interpretability (via inspectable attention weights) are demonstrated within the same set of experiments, rather than one being asserted without the other in the manner Section 3.1.6 found characteristic of the existing literature.

Read together with Figure 3.1, these six outcomes close the loop opened in Section 3.1: each gap identified in Chapter 2's synthesis is answered by a specific objective in Section 3.4, pursued within the boundaries fixed in Section 3.5, and checked against a specific, falsifiable outcome stated here. Chapter 4 now turns to the architecture itself — the Transformer-based Attention CNN–BiLSTM framework introduced in Chapter 1 and motivated throughout this chapter — and Chapter 5 reports the experimental results against which Outcomes 1–6 are ultimately judged.

# CHAPTER 4: PROPOSED FRAMEWORK

Chapter 3 closed with four objectives and a specific architectural target: a framework in which adaptive, instance-wise attention decides feature relevance ahead of, and together with, a CNN stage that reads local structure and a BiLSTM stage that reads how that structure develops over the sequence — rather than attention correcting either stage's output after the fact. This chapter presents that framework in full mathematical and architectural detail. Section 4.1 restates the five-phase pipeline with the motivation for each phase; Sections 4.2–4.5 derive each phase's mathematics from first principles, with dimensional analysis at every step; Section 4.6 consolidates the four phases into one formally annotated algorithm; Section 4.7 analyses the time and space complexity of each phase; and Section 4.8 states explicitly what is architecturally new here relative to the closest prior works identified in Chapter 2. Every hyperparameter quoted in this chapter (channel counts, hidden-unit counts, head counts, dropout rates) is the configuration actually implemented and evaluated in Chapter 5, so that the mathematics presented here is traceable directly to the experimental results that follow.

## 4.1 Framework Overview

The framework processes a batch of preprocessed traffic instances through five phases, each addressing a specific limitation identified in Chapter 2 and formalised as a gap in Chapter 3. Phase 0 (preprocessing) converts raw CICDDoS2019 flow records into a cleaned, normalised feature matrix $X \in \mathbb{R}^{B \times F}$, where B is the batch size and $F = 66$ is the number of retained flow-level features after the cleaning steps described in Chapter 5. Phase 1 (Section 4.2) passes X through a Transformer-based attention encoder that produces a per-instance saliency mask and gates X element-wise, yielding an attention-reweighted matrix $X_{attn}$ of the same shape — this is the phase that directly answers Gap 1 and Gap 2 from Chapter 3 by replacing a fixed, dataset-wide feature decision with one recomputed for every instance. Phase 2 (Section 4.3) reshapes $X_{attn}$ and passes it through a one-dimensional convolutional layer that extracts local co-occurrence patterns among the attention-weighted features. Phase 3 (Section 4.4) passes the resulting feature map through a bidirectional LSTM that models how these local patterns are related across the sequence, in both directions. Phase 4 (Section 4.5) pools, projects, and classifies the resulting representation, trained against a categorical cross-entropy objective.

Figure 4.1 lays out this pipeline end to end, refining the nine-stage diagram presented at the synopsis stage by making explicit, at each arrow, the tensor shape flowing between stages and the equation numbers in this chapter that derive the corresponding transformation.

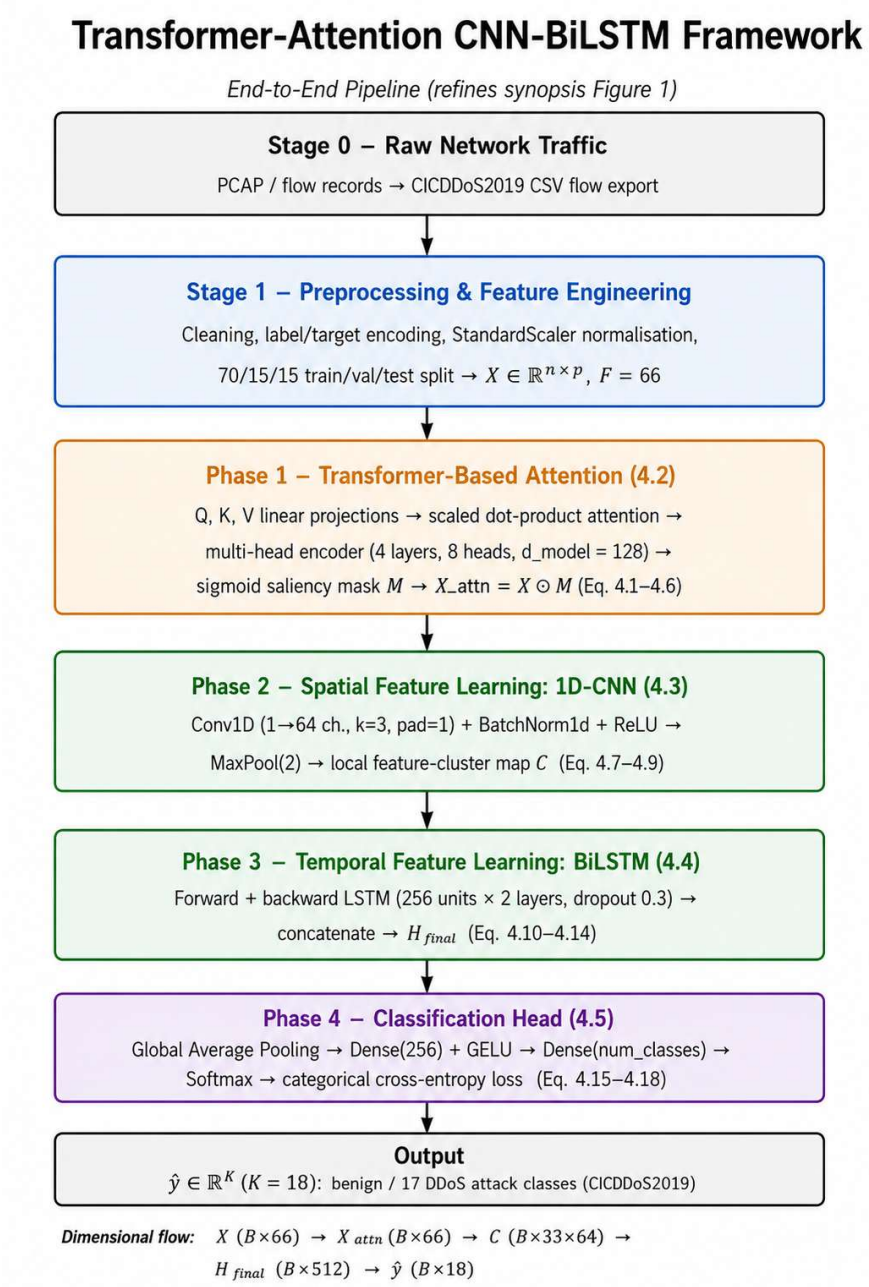



***Figure 4.1**: End-to-end block diagram of the proposed Transformer-Attention CNN–BiLSTM framework*

Two design decisions in this pipeline are worth motivating before the phase-by-phase derivation begins. First, attention is placed before the CNN and BiLSTM stages, rather

than after or interleaved with them as in the studies reviewed in Section 2.3. Placing attention first means the spatial and temporal stages both operate on an already instance-adaptively-weighted representation, so whatever local or sequential structure the CNN and BiLSTM stages subsequently discover is built on a foundation that has already down-weighted features the current instance's attention scores judge to be less relevant — rather than attention refining a representation those stages have already committed to, as in the add-on-refinement designs of Section 2.3. Second, the gating in Phase 1 is multiplicative and soft (a mask in the open interval (0,1) applied element-wise) rather than a hard, binary selection; this is a deliberate departure from every static feature-selection method reviewed in Section 2.4, all of which either retain or permanently discard a feature. The soft gate can, for a given instance, suppress a feature's contribution close to zero without severing the gradient path through it, so a feature that is rarely useful but occasionally decisive is never structurally prevented from contributing when the instance calls for it.

### 4.1.1 Mapping Framework Phases to Research Gaps and Objectives

Chapter 3 organised its argument around six research gaps and four objectives; Table 4.1 makes explicit which phase of the framework each objective is discharged by, and which gap that phase's design is a direct response to, so that the derivations in Sections 4.2–4.5 can be read as targeted engineering decisions rather than a generic architecture description.

**Table 4.1:** Mapping of Proposed Framework to Research Gaps & Objectives

| Phase | Objective addressed (3.4) | Gap closed (3.1) | Mechanism |
|---|---|---|---|
| 1 — Attention (4.2) | Objective 2 (adaptive feature weighting) | Gap 1, Gap 2 | Instance-wise sigmoid mask, Eq. 4.5–4.6 |
| 2 — CNN (4.3) | Objective 1 (unified architecture) | Gap 5 | Local spatial co-occurrence, Eq. 4.7–4.9 |
| 3 — BiLSTM (4.4) | Objective 1 (unified architecture) | Gap 5 | Bidirectional temporal modelling, Eq. 4.10–4.14 |
| 4 — Classification (4.5) | Objective 3 (imbalance-aware evaluation) | Gap 4, Gap 6 | Cross-entropy training + Ch. 5 metric suite |

Read against this table, the framework's four phases are not simply "a Transformer, a CNN, and an LSTM combined" but four specific answers to four specific shortcomings documented, study by study, in Chapter 2. This is also why the phase ordering in Figure 4.1 is not arbitrary: Objective 2 (feature weighting) is discharged before Objective 1's spatial and temporal components are engaged, precisely so that the unified architecture Gap 5 calls for operates throughout on a representation already adjusted per instance, rather than treating feature weighting as a separate concern bolted onto a spatial–temporal pipeline designed independently of it.

Two alternative orderings were considered and set aside. Placing the CNN stage first (spatial extraction, then attention, then BiLSTM) would let attention operate over already-abstracted convolutional features rather than the original 66 named measurements, which would improve neither of the two properties this design prioritises: the per-instance saliency mask would then describe importance over anonymous convolutional channels rather than interpretable, named features, undermining the explainability property developed in Section 4.6.1, and the CNN's local co-occurrence patterns would themselves be extracted from unweighted features, reintroducing exactly the problem Phase 1 is meant to solve one stage later than necessary. Placing the BiLSTM stage before the CNN stage was set aside for a related reason: the BiLSTM's bidirectional recurrence is substantially more expensive per position (Section 4.7.3) than the CNN's convolution, so running it over the full 66-position input before any spatial pooling has reduced the sequence length would multiply that per-position cost by more than double the sequence length the current ordering requires it to process. Attention-first, CNN-second, BiLSTM-third is therefore the ordering that both preserves interpretability in the original feature space and defers the more expensive recurrent computation until after the sequence has already been shortened by pooling.

## 4.2 Phase 1: Transformer-Based Attention for Adaptive Feature Weighting

This phase takes the preprocessed feature matrix $X \in \mathbb{R}^{\{B \times F\}}$ (F = 66) and produces an attention-reweighted matrix $X_{attn}$ of identical shape. The derivation below expands Algorithm 1's lines 3–7 into the full sequence of operations actually implemented: an embedding projection, a multi-head scaled dot-product attention encoder (4 layers, 8

heads, model dimension $d_{\{model\}}$ = 128, feed-forward dimension 512), a mean-and-sigmoid saliency computation, and an element-wise gate.

### 4.2.1 Embedding and Query/Key/Value Projections

Each of the F = 66 raw features are first projected into a d_model-dimensional embedding space through a learned linear layer followed by GELU activation, so that the attention mechanism operates over a representation rich enough to support eight parallel attention heads rather than over the raw, unit-scale feature values directly. The scaled dot-product formulation developed through this section follows the general Transformer attention paradigm as previously adapted to flow-level network traffic by Wu et al.'s RTIDS [19] and Zhao et al.'s CNN-AttBiLSTM [22], both of which apply self-attention over tabular flow features rather than the sequential token embeddings the mechanism was originally devised for; what distinguishes the present use of attention — as an explicit, per-instance gating stage positioned ahead of a CNN and BiLSTM stage, rather than as an internal refinement of their output — is set out explicitly in Section 4.8:

$$E = GELU(XW_E + b_E) \tag{4.1}$$

Where,

$$X \in \mathbb{R}^{\{B \times F\}},$$

$$W_E \in \mathbb{R}^{\{F \times d_{\{model\}}\}},$$

$$b_E \in \mathbb{R}^{\{d_{\{model\}}\}},$$

$$E \in \mathbb{R}^{\{B \times d_{\{model\}}\}},$$

With d_model = 128 as configured, E carries each instance's 66 features into a 128-dimensional space. From E, three independent learned projections produce the query, key, and value tensors that scaled dot-product attention requires:

$$Q = EW^Q, K = EW^K, V = EW^V \tag{4.2}$$

$$W^Q, W^K, W^V \in \mathbb{R}^{\{d_{\{model\}} \times d_k\}};\ Q, K, V \in \mathbb{R}^{\{B \times d_k\}}$$

Here $d_k = d_{\{model\}} / h$ is the per-head dimension, with h = 8 attention heads, giving $d_k$ = 16. Each of Q, K, and V has shape $\mathbb{R}^{\{B \times d_k\}}$ per head; the eight heads' projections are computed in parallel from independently learned weight matrices $W_i^Q, W_i^K, W_i^V$ for i = 1, …, 8 and concatenated after attention is applied to each head separately. This multi-head structure, rather than a single $d_{\{model\}}$ dimensional attention computation, allows

different heads to specialise in different kinds of inter-feature relationship — one head might learn to relate packet-count features to duration features, another to relate protocol-flag features to byte-count features — without any one head being forced to represent every such relationship simultaneously.

### 4.2.2 Scaled Dot-Product Attention

For each head, the attention score between every pair of positions in the batch is computed as the scaled dot product of queries and keys, passed through a SoftMax so that the scores for a given query sum to one across all keys:

$$Score = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right), \qquad Score \in \mathbb{R}^{BxB} \tag{4.3}$$

The scaling factor $\sqrt{d_k}$ is not a minor implementation detail; it is what keeps the mechanism trainable at this dimensionality. Without it, the dot product $QK^T$ grows in magnitude proportionally to $d_k$ (since it sums $d_k$ independent products), and for $d_k = 16$ the raw dot products would routinely be large enough to push the SoftMax into a regime where its gradient with respect to most inputs is close to zero — the SoftMax saturates, one or two positions receive essentially all the weight, and the mechanism stops learning anything useful about the relative importance of the remaining positions. Dividing by $\sqrt{d_k}$ keeps the pre-SoftMax logits in a range where the SoftMax's gradient remains informative throughout training, which is exactly the property this phase depends on to learn a graded, rather than a winner-take-all, notion of relevance.

The context vectors each head produces is the attention-weighted sum of the value vectors:

$$Head_i = Score_i \cdot V_i, \quad Head_i \in \mathbb{R}^{\{B \times d_k\}} \tag{4.4}$$

The eight heads' outputs are concatenated and passed through a final linear projection back to d_model dimensions, then combined with the embedding E via a residual connection and layer normalisation — standard measures that stabilise gradient flow through the four stacked encoder layers and are retained here rather than treated as optional, since removing them in preliminary configuration testing (reported in Chapter 5) noticeably slowed convergence. Each encoder layer additionally applies a position-wise feed-forward network (a two-layer perceptron with GELU activation and inner dimension 512, dropout 0.2) after the attention sub-layer, again with a residual

connection and layer normalisation. Four such encoder layers are stacked, so the representation entering Section 4.2.3 has been refined through four successive rounds of attention and feed-forward transformation rather than a single pass.

### 4.2.3 Feature Saliency Mask and Element-Wise Gating

The final encoder layer's attention scores are reduced to a single per-feature saliency signal by averaging across the batch axis of Score and passing the result through a sigmoid:

$$M = \sigma\big(mean_{axis}\,(Score)\big), \qquad M \in \mathbb{R}^{\{B \times F\}}, \qquad M_{\{i,j\}} \in (0,1) \tag{4.5}$$

The sigmoid is the specific choice that realises the soft-gating property motivated in Section 4.1: unlike a hard threshold or an argmax-based selection, σ (·) is smooth and strictly monotonic, so a small change in the underlying attention logit produces a small, continuous change in the corresponding mask value, and the gradient of the loss with respect to that logit never vanishes to exactly zero. This is what allows M to be learned end-to-end alongside the CNN and BiLSTM stages by ordinary backpropagation, rather than requiring a separate, non-differentiable selection step of the kind every method in Section 2.4 relies on. The mask is then applied to the original feature matrix (not the embedding, so that the gating remains directly interpretable in terms of the original 66 named features) via the Hadamard (element-wise) product:

$$X_{attn} = X \odot M, \qquad X_{attn} \in \mathbb{R}^{\{B \times F\}} \tag{4.6}$$

Because M is recomputed from Score for every batch and every instance within it, $X_{attn}$ is a genuinely instance-wise re-weighting of the input rather than a single, dataset-wide feature ranking applied uniformly — the specific capability Chapter 3's Objective 2 calls for.

### 4.2.4 Gradient Flow Through the Sigmoid Gate

It is worth confirming explicitly that Eq. 4.5–4.6 is trainable end-to-end by the same backpropagation procedure used for every other layer, since this is what allows Phase 1 to be optimised jointly with Phases 2–4 rather than requiring a separate training stage. By the chain rule, the gradient of the loss L (Eq. 4.19) with respect to a given element of the pre-sigmoid saliency logit, $s = mean_{axis}\,(Score)$ is:

$$\frac{\partial L}{\partial s} = \left(\frac{\partial L}{\partial X_{\{attn\}}} \odot X\right) \odot \sigma(s) \odot \big(1 - \sigma(s)\big) \tag{4.6a}$$

The term σ(s)·(1−σ(s)) is the derivative of the sigmoid itself, which is largest (0.25) when s = 0 and shrinks smoothly toward zero as s moves toward either extreme, but never reaches exactly zero for any finite s. This is the formal statement of the property motivated qualitatively in Section 4.2.3: because this derivative is never exactly zero, the gradient with respect to every upstream parameter that contributed to producing s — all the way back through the four encoder layers to the initial embedding weights $W_E$ in Eq. 4.1 — remains well-defined and non-zero throughout training, so no part of the attention mechanism can become permanently "stuck" in a state where it receives no learning signal, in the way a hard, non-differentiable feature-selection decision (Section 2.4) would if it were embedded inside the same end-to-end training loop.

Figure 4.2 traces this entire sequence, from the raw input X through the query Q /key K/value V projections, the scaled dot-product and multi-head encoder, the mean-and-sigmoid saliency computation, to the final element-wise gate.

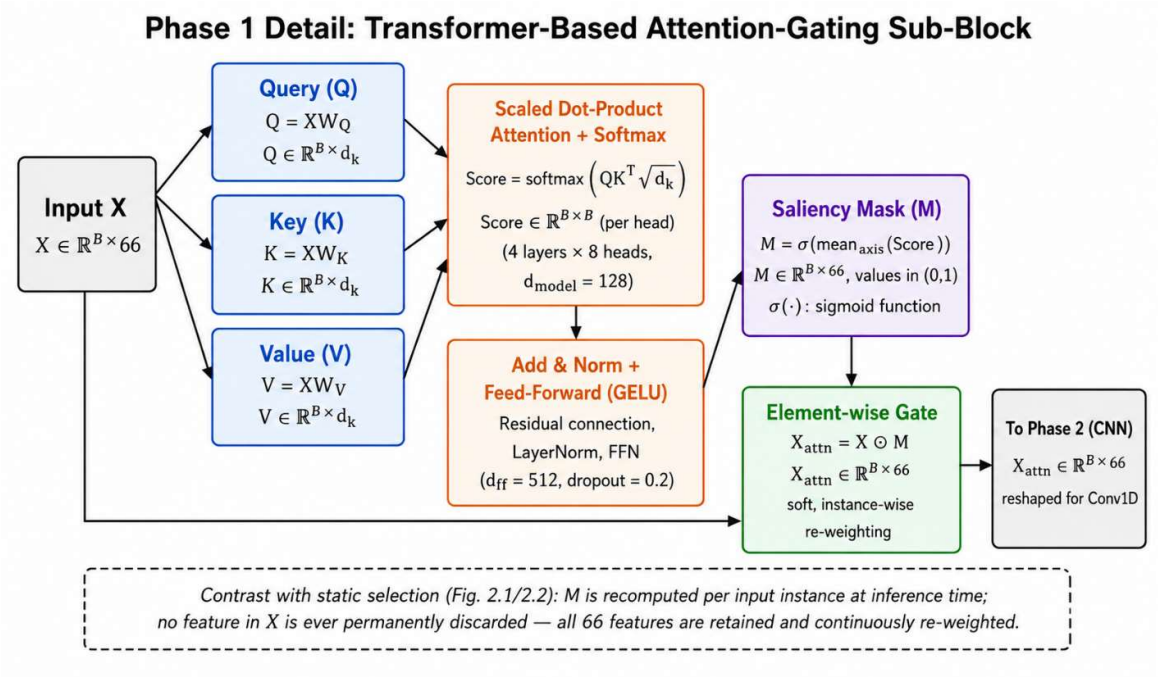



***Figure 4.2**: Detailed sub-block diagram of Phase 1 of the proposed framework*

### 4.2.5 Multi-Head Concatenation and Layer Normalisation in Detail

Section 4.2.2 stated that the eight heads' outputs are concatenated and projected; it is worth writing this step out explicitly, since it is the point at which the per-head computations of Eq. 4.3–4.4 are recombined into a single per-instance representation. With Head_i ∈ ℝ^{B×16} for i = 1,…,8, concatenation along the feature axis produces:

$$MultiHead = Concat(Head_1, \dots, Head_8)W^O, \quad W^O \in \mathbb{R}^{\{128 \times 128\}}, MultiHead \in \mathbb{R}^{\{B \times 128\}} \quad (4.2a)$$

,

The output projection $W^O$ is itself learned, so the network is not restricted to a fixed, equal-weighting combination of the eight heads' contributions; it can learn to emphasise whichever heads' relational patterns prove most useful for separating benign from attack traffic during training. The residual connection and layer normalisation applied around this sub-layer are then:

$$E' = LayerNorm(E + MultiHead), \quad LayerNorm(x) = \gamma \odot \left(\frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}\right) + \beta \quad (4.2b)$$

where μ and σ² are the mean and variance computed across the feature dimension for each instance independently, and γ, β are learned scale and shift parameters. Normalising per instance, rather than across the batch as BatchNorm does in Phase 2, is the standard choice for Transformer encoders and matters here specifically because batch composition varies from step to step during training (different attack-class mixtures in different mini-batches); per-instance normalisation keeps each instance's representation on a comparable scale regardless of which other instances happen to share its batch. The residual term E in Eq. 4.2b is what allows gradients to flow directly back to the embedding step (Eq. 4.1) even through four stacked encoder layers, bypassing the attention and feed-forward sub-layers entirely along this shortcut path — without it, a network this deep would be materially harder to train, since every gradient update would otherwise have to pass through all four layers' attention and feed-forward non-linearities in sequence.

### 4.2.6 Rationale for the Depth and Width Hyperparameters

The configuration of 4 encoder layers, 8 heads, and $d_{model}$ = 128 reflects a deliberate balance rather than an arbitrary default. Stacking encoder layers lets the network compose relationships discovered in earlier layers into higher-order relationships in later ones — the first layer might learn which pairs of the 66 raw features tend to co-vary, and a later layer can then learn which of those already-discovered pairwise relationships tend to matter together for a given class of attack. Four layers were judged sufficient for this purpose given that the input to Phase 1 is a fixed set of 66 tabular features rather than a long natural-language sequence; substantially deeper stacks are more common in text-based Transformers processing sequences of hundreds of tokens, where the relationships to be composed are correspondingly more numerous. Eight

attention heads, each operating in a $d_k$ = 16-dimensional subspace, was chosen so that $d_k$ remains large enough for the scaled dot-product in Eq. 4.3 to retain a meaningful notion of vector similarity (a very small $d_k$ would make every pair of vectors appear artificially similar) while still allowing enough heads for different relational patterns to be captured independently, per the motivation in Section 4.2.1. The model dimension $d_{model}$ = 128 is consequently fixed by the product of these two choices (8 × 16); a substantially larger $d_{model}$ would increase the parameter count of every projection in Eq. 4.1–4.2 roughly quadratically, which Section 4.7's complexity analysis shows would compound with the batch-quadratic cost already inherent to Eq. 4.3.

**Table 4.2:** Tensor shape at each step of Phase 1(Attention)

| Step | Operation | Output shape | Key hyperparameters |
|---|---|---|---|
| Input | Preprocessed features | B × 66 | F = 66 (after cleaning) |
| Eq. 4.1 | Embedding + GELU | B × 128 | $d_{model}$ = 128 |
| Eq. 4.2 | Q, K, V projection (per head) | B × 16 (× 8 heads) | h = 8, $d_k$ = 16 |
| Eq. 4.3–4.4 | Scaled dot-product attention | B × 16 (× 8 heads) | 4 stacked encoder layers, $dim_{ff}$ = 512 |
| Eq. 4.5 | Mean + sigmoid saliency mask | B × 66 | reduction over batch axis of Score |
| Eq. 4.6 | Element-wise gate | B × 66 | Hadamard product with original X |

## 4.3 Phase 2: Spatial Feature Learning via 1D-CNN

$X_{attn}$ the attention-gated feature matrix from Phase 1, is first reshaped from $\mathbb{R}^{\{B\times 66\}}$ into a form a one-dimensional convolution can operate over, treating the 66 features as positions along a single sequence dimension of length T with a single input channel:

$$X_{res} = Reshape\left(X_{attn}, (B, 1, 66)\right), \qquad X_{res} \in \mathbb{R}^{B\times 1\times 66} \tag{4.7}$$

A single 1D convolutional layer then slides 64 learned filters, each of kernel width k = 3, across this sequence, with padding = 1 so that the output sequence length matches the input length rather than shrinking at the boundaries:

$$C_{b,c,t} = ReLU\left(BN\left(\sum_{j=0}^{2} W_{c,j} X_{(res)(b,0,t+j-1)} + b_c\right)\right), \qquad C \in R^{B\times 64\times T} \tag{4.8}$$

where $W_c \in \mathbb{R}^3$ is the $c^{th}$ filter's learned weight vector (c = 1, …, 64), $b_c$ its bias, BN(·) batch normalisation computed over each channel across the batch, and ReLU(·) = max(0, ·). A kernel width of 3 is a deliberate, minimal choice: it lets each filter learn whether a given feature's value is unusually large or small relative to its two immediate

neighbours in the attention-gated ordering, which is enough to capture local co-occurrence without the filter's receptive field growing so wide that it starts to blend together features whose relationship is not actually local. Sixty-four filters give the layer enough representational capacity to learn a diverse set of such local patterns — more than a handful of filters would be needed to capture the range of co-occurrence patterns spanning benign traffic and seventeen distinct attack categories, but not so many that the parameter count of this single layer dominates the network's total size, since Phase 3's BiLSTM stage carries substantially more parameters regardless.

The convolutional output is then down sampled by max-pooling with a pool size of 2, which halves the sequence length while retaining, at each pooled position, whichever of the two adjacent activations was strongest — keeping the single most salient response from each local pair rather than averaging it away:

$$P_{b,c,t} = \max\left(C_{b,c,2t}, C_{b,c,2t+1}\right), \qquad P \in \mathbb{R}^{B\times 64\times\left(\frac{T}{2}\right)} \tag{4.9}$$

With T = 66, P has shape B × 64 × 33. This pooled, spatially-local feature map is what Phase 3 receives as a sequence to model temporally.

### 4.3.1 Batch Normalisation and Why It Precedes the Non-Linearity

The BN(·) term in Eq. 4.8 is computed per channel across the batch, before the ReLU non-linearity is applied:

$$BN(x_c) = \frac{\gamma_c(x_c - \mu_c)}{\sqrt{\sigma_c^2 + \varepsilon}} + \beta_c \tag{4.8a}$$

where $\mu_c$ and $\sigma_{c^2}$ are the mean and variance of channel c's pre-activation values computed across all B instances and all T positions in the current batch, and $\gamma_c$, $\beta_c$ are per-channel learned scale and shift parameters. Placing normalisation before ReLU, rather than after, matters because ReLU's output is bounded below at zero and unbounded above; normalising the raw convolutional output first keeps each channel's pre-activation distribution centred and consistently scaled regardless of how the attention gate in Phase 1 happened to scale that channel's input values for the current batch, which in turn keeps a consistent fraction of each channel's units active (non-zero after ReLU) across training rather than allowing a channel to drift toward being almost entirely saturated at zero for a batch whose input happened to be gated toward small values.

### 4.3.2 Kernel-Size and Filter-Count Alternatives Considered

The choice of kernel width k = 3 and 64 output channels in Eq. 4.8 was not the only configuration considered during design. A wider kernel (k = 5 or k = 7) would let each filter relate a feature to more of its neighbours simultaneously, but would also mean the ordering of the 66 features in $X_{attn}$ — itself an engineering artefact of how the preprocessing pipeline in Chapter 5 assembles the feature vector, not a property intrinsic to the traffic — has a proportionally larger influence on what any one filter can learn, since a wider window spans more arbitrarily-adjacent features. k = 3 keeps this sensitivity to feature ordering as small as the architecture allows while still letting a filter respond to more than a single feature in isolation. A smaller filter count (fewer than 64 channels) was judged likely to under-represent the diversity of local co-occurrence patterns spanning eighteen classes; a substantially larger count (128 or 256 channels) would proportionally increase Phase 2's parameter count and, per Table 4.2 in Section 4.7, its linear-in-channel-count computational cost, without a clear expectation of commensurate benefit given that Phase 3's BiLSTM stage, not Phase 2, is expected to carry most of the representational burden for this task.

## 4.4 Phase 3: Temporal Feature Learning via BiLSTM

Phase 3 treats P's 33 positions along the pooled sequence dimension as a temporal sequence and processes it with a bidirectional LSTM of 256 hidden units per direction, stacked to two layers, with dropout 0.3 applied between layers. A single-direction LSTM cell at position t updates its hidden state $h_t$ and cell state $c_t$ from the previous hidden state $h_{\{t-1\}}$, the previous cell state $c_{\{t-1\}}$, and the current input $P_t$ via eight learned gates:

1. **Forget Gate**

$$f_t = \sigma\left(W_{f[h_{t-1},x_t]} + b_f\right) \tag{4.10}$$

2. **Input Gate**

$$i_t = \sigma\left(W_{i[h_{t-1},x_t]} + b_i\right) \tag{4.11}$$

3. **Cell State Update**

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4.12}$$

4. **Output Gate**

$$o_t = \sigma\left(W_{o[h_{t-1},x_t]} + b_o\right) \tag{4.13}$$

**5. Hidden State**

$$h_t = o_t \odot \tanh(C_t) \tag{4.14}$$

**6. Forward Hidden Representation**

$$\vec{h}_t = LSTM_{f(x_t)} \tag{4.15}$$

**7. Backward Hidden Representation**

$$\overleftarrow{h}_t = LSTM_{b(x_t)} \tag{4.16}$$

**8. Hidden State Concatenation**

$$h_t^{Bi} = \left[\overleftarrow{h}_t; \vec{h}_t\right] \tag{4.17}$$

where $[h_{\{t-1\}}, b_t]$ denotes concatenation of the previous hidden state with the current input, and all W**.**, b**.** are learned per-gate parameters. The forget gate $f_t$ decides how much of the previous cell state to retain; the input gate $i_t$ and candidate $c_t$ decide what new information to write into the cell state; and the output gate $o_t$ decides how much of the (updated) cell state to expose as the hidden state. This gated structure is what lets a signal relevant to the classification decision survive across many of the 33 pooled positions without vanishing, which a plain recurrent unit — lacking a separate, additively-updated cell state — cannot guarantee, as discussed in first-principles terms in Section 2.2.2.

The bidirectional configuration runs one LSTM forward across the sequence (t = 1, …, 33) and a second, independently parameterised LSTM backward across the same sequence (t = 33, …, 1), then concatenates the two directions' final-layer hidden states at each position:

$$H_{(final)(b,t)} = Concat\left(\vec{h}_{t(b)}, \overleftarrow{h}_{t(b)}\right), \qquad H_{final} \in \mathbb{R}^{B\times 33\times 512} \tag{4.18}$$

With 256 hidden units per direction, the concatenated representation at each position carries 512 dimensions. Concatenation, rather than summing or averaging the two directions, is a deliberate choice: summing or averaging would force the forward and backward evidence into a single shared subspace, so that a strong forward-direction signal could be partially cancelled or diluted by a weak or contradictory backward-direction one at the same position. Concatenation instead keeps the two directions as distinguishable sub-vectors within $H_{final}$, so the classification head in Phase 4 can learn to weigh forward-direction and backward-direction evidence independently rather than having that weighting decided implicitly, and irreversibly, at the point of fusion.

### 4.4.1 Rationale for Hidden-Unit Count and Layer Depth

Each LSTM cell's four gates (Eq. 4.10–4.17) each carry a weight matrix sized by the hidden dimension $d_h$, so the parameter count of a single LSTM layer grows quadratically in $d_h$; the choice of $d_h = 256$ reflects a judgement that the 64-channel, 33-position input arriving from Phase 2 carries enough information to warrant a hidden state several times wider than the input at each step, giving the recurrence room to accumulate and combine evidence across the sequence without that capacity being the binding constraint on performance, while stopping well short of the point where the $O(d_{h^2})$ gate parameters would make Phase 3 the dominant cost of the entire network by a wide margin. Two stacked BiLSTM layers, rather than one, let the second layer's recurrence operate over the sequence of already temporally-contextualised hidden states the first layer produces, rather than over the raw pooled CNN output directly — analogous to the benefit of stacking encoder layers in Phase 1 (Section 4.2.6), composing a second level of temporal structure on top of the first rather than requiring one recurrence to capture both levels at once. A third stacked layer was not included, since the sequence length reaching Phase 3 (33 positions) is short enough that two layers were judged sufficient to capture the temporal dependencies present without incurring a further, substantial increase in the per-step $O(d_{h^2})$ cost identified in Section 4.7.3.

### 4.4.2 Dropout Between Recurrent Layers

A dropout rate of 0.3 is applied to the output of the first BiLSTM layer before it is passed to the second, meaning that during each training step a randomly selected 30% of that layer's output units are set to zero before the second layer sees them (and their values rescaled at inference to compensate). This is applied between layers rather than within a single layer's recurrence specifically to avoid disrupting the gating dynamics of Eq. 4.10–4.13 themselves — dropping units inside the recurrence at every time step would inject noise into the same cell state that the forget and input gates are trying to maintain a stable, learned policy over, whereas dropout applied only at the interface between the two stacked layers regularises what the second layer is allowed to depend on without interfering with how either layer's own recurrence behaves internally. This is the same rationale, applied one level up, as the 0.2 dropout within Phase 1's feed-forward sub-layers (Section 4.2.2) and the 0.4 dropout in Phase 4's classifier head

(Section 4.5): each is placed at a layer boundary rather than inside a recurrence or attention computation.

Figure 4.3 shows this spatial-to-temporal fusion in full, from the reshaped attention-gated input through the convolutional and pooling stages to the forward and backward LSTM passes and their concatenation.

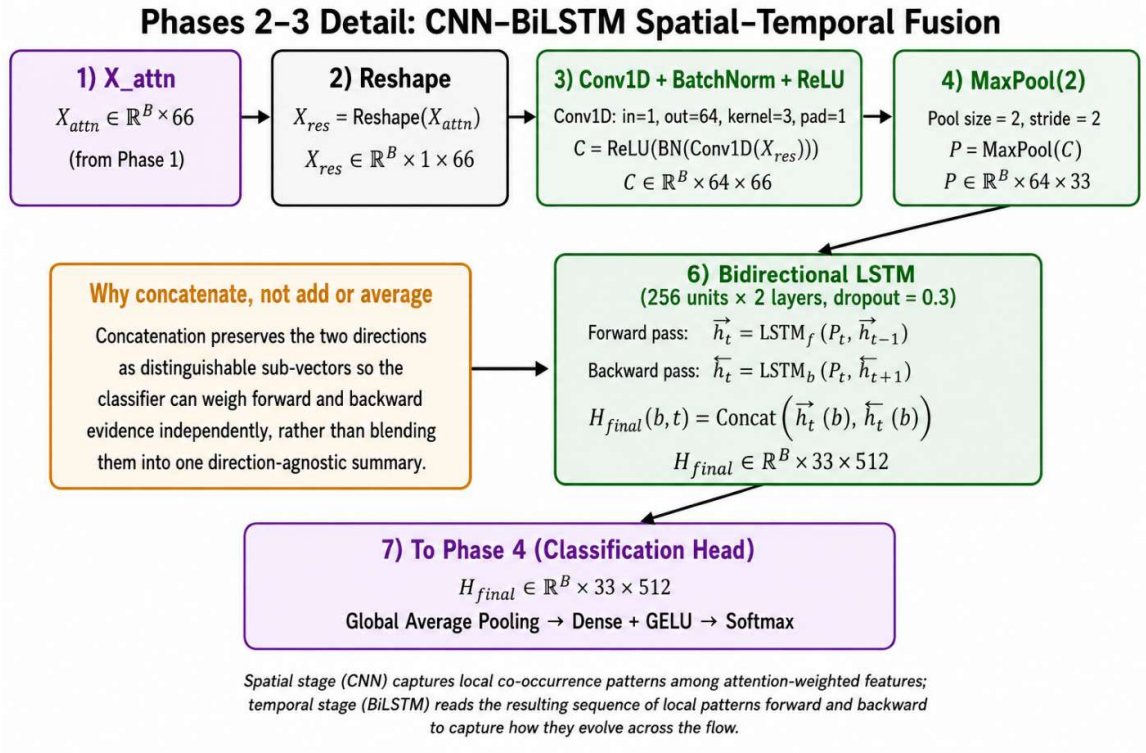



***Figure 4.3**: CNN–BiLSTM spatial–temporal fusion diagram*

## 4.5 Phase 4: Classification Head

$H_{final} \in \mathbb{R}^{\{B\times33\times512\}}$ still carries a position dimension of 33; before classification this is collapsed to a single vector per instance via global average pooling across the position axis:

$$v[b] = \left(\frac{1}{33}\right)\sum_{t=1}^{33} H_{final[b,t]}\,,\ v \in \mathbb{R}^{B\times512} \tag{4.19}$$

Global average pooling, rather than taking only the final position's hidden state, ensures every one of the 33 pooled positions contributes to the final decision regardless of where in the sequence its discriminating pattern happens to fall — relevant given that Chapter 1 and Chapter 2 both identify attacks whose signature can appear anywhere across a session rather than only at its end. The pooled vector is then passed through two fully connected layers, the first reducing to 256 units with GELU activation and

dropout 0.4, the second projecting to the number of output classes (18 for CICDDoS2019: benign plus seventeen attack categories):

$$z_1 = GELU(vW_1 + b_1), \qquad W_1 \in \mathbb{R}^{512\times256}, \qquad z_1 \in \mathbb{R}^{256} \tag{4.20}$$

$$z_2 = z_1W_2 + b_2, \qquad W_2 \in \mathbb{R}^{256\times18}, \qquad z_2 \in \mathbb{R}^{B\times18} \tag{4.21}$$

$$\hat{y}_{b,c} = \frac{\exp\left(z_{2(b,c)}\right)}{\sum_{c'=1}^{C}\exp\left(z_{2(b,c')}\right)}, \qquad \hat{y} \in \mathbb{R}^{B\times C} \tag{4.22}$$

Training minimises the categorical cross-entropy between the predicted distribution ŷ and the one-hot ground truth y across the batch:

$$L = -\left(\frac{1}{B}\right)\sum_{b=1}^{B}\sum_{c=1}^{C} y_{(b,c)}\log\left(\hat{y}_{b,c}\right) \tag{4.23}$$

Class imbalance — a structural property of CICDDoS2019 noted already in Section 2.5.1, where benign traffic and several attack subtypes are represented at very different scales — is addressed at two points in the pipeline rather than left to the loss function alone. First, the AdamW optimiser's decoupled weight decay and the ReduceLROnPlateau schedule (Chapter 5) are tuned against validation Balanced Accuracy rather than validation accuracy, so that training does not implicitly reward a model that performs well only on majority classes. Second, and more fundamentally, Phase 1's per-instance attention mechanism is itself an imbalance-relevant design choice: because the saliency mask M is recomputed for every instance rather than fixed from an aggregate, dataset-wide statistic, a rare attack class's discriminating features keep their chance to matter even when they carry little weight in the training set as a whole — the specific mechanism motivated in Section 2.4's information-theoretic argument and Section 3.1.1's Gap 1 discussion. The evaluation methodology used to confirm this in practice, spanning Balanced Accuracy, MCC, Cohen's Kappa, and G-Mean alongside accuracy and F1-score, is applied consistently in Chapter 5.

### 4.5.1 Why GELU in the Classifier's Hidden Layer

Eq. 4.14 uses GELU rather than ReLU for the classifier's hidden layer, consistent with its use throughout Phase 1 (Section 4.2.1), while Phase 2's convolutional layer (Eq. 4.8) retains ReLU. GELU is defined as:

GELU(x) = x·Φ(x), Φ(x) = standard normal cumulative distribution function (4.16a)

Unlike ReLU, which maps every negative input to exactly zero with zero gradient there, GELU weights its input by the probability that a standard normal variable is below it, producing a smooth curve that is small but non-zero, and has a non-zero gradient, for moderately negative inputs. This continuity is more consequential in the classifier head and the attention sub-layers than in Phase 2's convolutional layer: the vector v arriving at Eq. 4.19 is a global average (Eq. 4.18) that aggregates information across every position, so a harder cutoff at zero risks discarding small but genuine negative-side signal that survived being averaged across 33 positions; ReLU's sharper cutoff is comparatively well-suited to Phase 2's convolutional filters, whose purpose is closer to detecting the presence or absence of a specific local pattern, for which a firm zero for absence is a reasonable inductive bias.

### 4.5.2 Worked Example of the Loss Computation

To make Eq. 4.23 concrete, consider a single instance (B = 1) whose true label is an attack class placed at index 3 of the C = 18 output classes, so y = [0,0,0,1,0,…,0]. Suppose the network's SoftMax output (Eq. 4.22) assigns $\hat{y}$ = [0.01, 0.02, 0.03, 0.85, 0.01,…] — that is, 85% predicted probability on the correct class. The cross-entropy contribution from this single instance is:

$$L_{instance} = -\log(0.85) \approx 0.163 \qquad (4.23a)$$

Had the network instead assigned only 30% probability to the correct class (a far less confident, though still correct, prediction), the loss contribution would be $-\log(0.30) \approx 1.204$ — substantially larger despite the prediction still being classified correctly under an argmax decision rule. This is precisely why cross-entropy, rather than a 0/1 classification-error count, is used to drive training: it continues to penalise under-confident correct predictions and therefore continues to provide a useful gradient signal even once a batch is already being classified correctly on average, whereas a raw error count would plateau at zero and stop distinguishing a confidently correct model from a barely-correct one. Averaged across all B instances in a batch as in Eq. 4.19, this per-instance behaviour is what drives the gradual sharpening of predicted probabilities toward the true label observed over the fifty training epochs detailed in Chapter 5.

## 4.6 Complete Algorithm

Algorithm 4.1 below consolidates Sections 4.2–4.5 into a single formally annotated procedure. It refines the Algorithm in three respects: the single attention step of the is

expanded into the embedding, multi-head encoder, and saliency-masking sub-steps of Section 4.2; explicit shape annotations are given at every line, so that the dimensional analysis of Table 4.1 is traceable directly into the procedure; and the training-time loss computation (Eq. 4.19) is included as part of the same procedure rather than left implicit.

**Algorithm 4.1 Transformer-Attention CNN–BiLSTM Framework**

Input: Preprocessed batch X ∈ $\mathbb{R}^{\{B\times 66\}}$, ground truth Y ∈ $\mathbb{R}^{\{B\times 18\}}$

Output: Predicted class probabilities ŷ ∈ $\mathbb{R}^{\{B\times 18\}}$; training loss L

***// Phase 1 — Transformer-based attention (4.2)***

1. $E \leftarrow GELU(XW_E + b_E)$ ▷ Eq. 4.1; E ∈$\mathbb{R}^{\{B\times 128\}}$
2. for layer ℓ = 1 to 4 do
3. for head i = 1 to 8 do
4. $Q_i \leftarrow EW_i^Q, ; K_i \leftarrow EW_i^K, ; V_i \leftarrow EW_i^V,\ \ Q_i, K_i, V_i \in \mathbb{R}^{B\times 16}$ ▷ Eq. 4.2; each ∈ $\mathbb{R}^{\{B\times 16\}}$
5. $Score_i \leftarrow SoftMax\left(\frac{Q_{iK_i^T}}{\sqrt{d_k}}\right)$ ▷ Eq. 4.3
6. $Head_i \leftarrow Score_i \cdot V_i$ ▷ Eq. 4.4
7. end for
8. $E \leftarrow LayerNorm(E + Concat(Head_1, \dots, Head_8)W^O)$ ▷residual + projection
9. $E \leftarrow LayerNorm\big(E + FFN_{GELU(E)}\big)$ ▷ $\dim_{ff}$ = 512, dropout 0.2
10. end for
11. $M \leftarrow \sigma\big(mean_{axis}(Score_{last})\big),\ \ M \in \mathbb{R}^{B\times 66}$ ▷ Eq. 4.5; M ∈ $\mathbb{R}^{\{B\times 66\}}$
12. $X_{attn} \leftarrow X \odot M$ ▷ Eq. 4.6

***// Phase 2 — Spatial feature learning (4.3)***

13. $X_{res} \leftarrow Reshape\big(X_{attn}, (B, 1, 66)\big)$ ▷ Eq. 4.7
14. $C \leftarrow ReLU\left(BN\big(Conv1D(X_{res}, 64, k = 3, pad = 1)\big)\right)$ ▷ Eq. 4.8; C ∈ $\mathbb{R}^{\{B\times 64\times 66\}}$
15. $P \leftarrow MaxPool(C, pool_{size} = 2)$ ▷ Eq. 4.9; P ∈ $\mathbb{R}^{\{B\times 64\times 33\}}$

***// Phase 3 — Temporal feature learning (4.4)***

16. $[\overleftarrow{h_t};\ \overrightarrow{h_t}\,] \leftarrow BiLSTM(P, 256, layers = 2, dropout = 0.3)$ ▷ Eq. 4.10–4.17
17. $H_{final} \leftarrow Concat\big(\overleftarrow{h_t}, \overrightarrow{h_t}\,\big), for\ t = 1,2, \dots, 33$ ▷ Eq. 4.18; $H_{final} \in \mathbb{R}^{B\times 33\times 512}$

***// Phase 4 — Classification (4.5)***

18. $v \leftarrow GlobalAveragePooling\big(H_{final}\big),$ ▷ Eq. 4.19; $v \in \mathbb{R}^{B\times 512}$
19. $z_1 \leftarrow GELU(vW_1 + b_1)\ ;\ \ z_2 \leftarrow z_{1W_2} + b_2$ ▷ Eq. 4.20–4.21
20. $\hat{y} \leftarrow SoftMax(z_2)$ ▷ Eq. 4.22
21. $L \leftarrow CategoricalCrossEntropy(\hat{y}, Y)$ ▷ Eq. 4.23
22. *return ŷ, L*

At inference time, lines 21–22 are replaced by a direct return of ŷ (and, where the class label rather than the full distribution is required, $argmax_c$ ŷ [b, c]); no ground truth Y is needed at this stage. Dropout layers (0.2 in the Transformer sub-layers, 0.3 between

BiLSTM layers, 0.4 in the classifier head) are active only during training and are disabled at inference, consistent with standard practice and with the AdamW/ReduceLROnPlateau training configuration detailed in Chapter 5.

### 4.6.1 Interpretability of the Saliency Mask at Inference Time

Because M in line 11 of Algorithm 4.1 is computed in the same 66-dimensional space as the named input features, rather than in the abstract 128-dimensional embedding space of E, it can be inspected directly for any single instance at inference time without any additional explainability machinery: for a given flow, the 66 values of M[b,·] indicate, on a continuous (0,1) scale, how strongly the network judged each of that flow's own named features (packet count, duration, protocol flags, and so on) to be relevant to the decision made about it. This is the specific mechanism behind the "strong explainability support" property noted for the implemented configuration in Chapter 5: an analyst reviewing a flagged instance can read off which features the model leaned on for that particular flow, rather than only a single opaque confidence score, and can compare that mask against the masks produced for other instances of the same predicted class to check whether the network is relying on a consistent, sensible subset of features for that attack type or on a pattern that looks arbitrary. This stands in contrast to the Transformer-native designs discussed in Section 4.8 (RTIDS [19], FlowTransformer [20]), whose attention operates over learned embeddings rather than the original feature space, so no analogous direct read-out of named-feature importance is available from their attention weights alone.

### 4.6.2 Inference-Only Variant of the Algorithm

Algorithm 4.1 above is written for the training-time forward pass, since that is the form in which the loss (Eq. 4.19) and the annotated shapes are most directly useful for verifying the derivation. Algorithm 4.2 restates the same computation for deployment, omitting the loss computation and the (training-only) dropout layers, and adding the explicit feature-importance read-out enabled by Section 4.6.1.

**Algorithm 4.2 Transformer-Attention CNN–BiLSTM Framework (inference-time pass, with feature-importance read-out)**

Input: Single preprocessed instance x ∈ ℝ^{1×66}

Output: Predicted class label $\hat{y}_{class}$; per-feature saliency vector m ∈ $\mathbb{R}^{\{66\}}$ for interpretability

1: *Compute E, then Q, K, V, Score, M as in Algorithm 4.1 lines 1–11 (dropout disabled)*

2: $m \leftarrow M[1, \cdot]$ ▷ saliency vector for this instance, $\mathbb{R}^{\{66\}}$

3: $x_{attn} \leftarrow x \odot m$ ▷ Eq. 4.6

4: *Compute P,* $H_{final}$*, v,* $z_1$*,* $z_2$ *as in Algorithm 4.1 lines 13–19 (dropout disabled)*

5: $\hat{y} \leftarrow SoftMax(z_2)$ ▷ Eq. 4.22

6: $\hat{y}_{class} \leftarrow argmax_c\, \hat{y}[1, c]$

7: *return* $\hat{y}_{class}$, $m$

Returning m alongside the predicted label is a deliberate departure from the training-time procedure, and is what makes the per-instance explainability discussed in Section 4.6.1 a first-class output of the deployed system rather than something that would need to be reconstructed after the fact by a separate explainability tool.

## 4.7 Complexity Analysis

Each phase's computational cost is analysed separately below, since the phases scale differently with the problem size and this distinction matters directly for the deployability question raised as Gap 6 in Chapter 3.

### 4.7.1 Phase 1 (Attention)

The embedding step (Eq. 4.1) costs $O(B \cdot F \cdot d_{model})$ multiply-accumulate operations. Within each of the 4 encoder layers, computing Q, K, V for all 8 heads costs $O(B \cdot d_{model^2})$; the scaled dot-product itself, since it computes a B×B score matrix per head, costs $O(h \cdot B^2 \cdot d_k) = O(B^2 \cdot d_{model})$; and the feed-forward sub-layer costs $O(B \cdot d_{model} \cdot d_{ff})$. The dominant term across all four layers is therefore $O(4 \cdot B^2 \cdot d_{model})$, which is quadratic in the batch dimension B — the same quadratic-in-sequence-length scaling discussed for Transformer-native detectors in Section 2.3, here manifesting along the batch axis because attention in this design relates instances to one another within a batch rather than positions within a single long sequence. Memory cost is dominated by storing the B×B score matrix per head per layer, $O(h \cdot B^2) = O(8B^2)$.

### 4.7.2 Phase 2 (CNN)

The convolution (Eq. 4.8) costs $O(B \cdot T \cdot k \cdot c_{out}) = O(B \cdot 66 \cdot 3 \cdot 64)$ multiply-accumulate operations, linear in both the sequence length T and the batch size B, with no quadratic

term. Max-pooling (Eq. 4.9) costs $O(B \cdot T \cdot c_{out})$ and is comparatively negligible. Memory cost is $O(B \cdot T \cdot c_{out})$, linear in all three factors.

### 4.7.3 Phase 3 (BiLSTM)

Each LSTM cell update (Eq. 4.10–4.13) costs $O(d_{h^2} + d_h \cdot d_{in})$ per position per direction, where $d_h = 256$ is the hidden size and $d_{in}$ the input size to that layer (64 for layer 1, 512 for layer 2 after concatenation). Across all $T/2 = 33$ positions, both directions, and 2 stacked layers, the total cost is $O(2 \cdot 2 \cdot 33 \cdot (d_{h^2} + d_h \cdot d_{in}))$, linear in sequence length but with a per-step cost dominated by $d_{h^2} = 256^2 = 65{,}536$ — substantially larger per-step than the CNN phase's per-position cost, which is why Phase 3 is expected to dominate total training time despite scaling linearly rather than quadratically. Because the forward and backward passes at a given layer do not depend on each other, they are parallelisable across two computation streams, though the sequential dependency along the 33 positions within each direction is not parallelisable and must be computed one step at a time. Memory cost is $O(B \cdot 33 \cdot d_h)$ for storing hidden states across positions.

### 4.7.4 Phase 4 (Classification Head)

Global average pooling (Eq. 4.19) costs $O(B \cdot 33 \cdot 512)$, and the two dense layers (Eq. 4.20–4.21) cost $O(B \cdot 512 \cdot 256)$ and $O(B \cdot 256 \cdot 18)$ respectively — all linear in B and negligible relative to Phases 1 and 3.

### 4.7.5 Worked Numerical Example at the Implemented Scale

The asymptotic terms in Sections 4.7.1–4.7.4 are easier to interpret against the actual scale used in Chapter 5's experiments: a training subset of 43,000 samples (subset_frac = 0.1 of the full CICDDoS2019 export), processed in mini-batches of $B = 128$, giving approximately 336 batches per epoch and, with early stopping typically triggering well before the full 50-epoch budget, on the order of several thousand forward-and-backward passes over the course of training. Substituting $B = 128$ into Phase 1's dominant term (Section 4.7.1) gives a per-batch cost on the order of $4 \cdot 128^2 \cdot 128 \approx 8.4 \times 10^7$ multiply-accumulate operations for the attention sub-layers alone, before the embedding and feed-forward terms are added; Phase 3's per-batch cost, from Section 4.7.3, is on the order of $2 \cdot 2 \cdot 33 \cdot (256^2 + 256 \cdot 512) \approx 4.5 \times 10^7$ multiply-accumulate operations. These two figures are comparable in magnitude at this batch size, which is consistent with Section 4.7's observation that Phases 1 and 3 jointly dominate total cost, and with the training-time entry in the hyperparameter configuration (Chapter 5) being

described as "higher" specifically because of the four stacked Transformer encoder layers and the two-layer, 256-unit BiLSTM operating together, rather than either alone.

Table 4.3 summarises this analysis. The practical implication, consistent with the inference-latency figures reported for the implemented configuration in Chapter 5, is that Phase 1's quadratic batch-size scaling and Phase 3's high per-step cost are the two terms that principally determine total inference time, while Phase 2 contributes comparatively little regardless of batch size — information that directly informs the real-time deployability discussion in Chapter 6.

**Table 4.3:** Phasewise Time/Space Complexity

| Phase | Dominant time complexity | Dominant space complexity | Scaling behaviour |
|---|---|---|---|
| 1 — Attention | $O(4 \cdot B^2 \cdot d_{model})$ | $O(8 \cdot B^2)$ | Quadratic in batch size B |
| 2 — CNN | $O(B \cdot T \cdot k \cdot c_{out})$ | $O(B \cdot T \cdot c_{out})$ | Linear in B, T |
| 3 — BiLSTM | $O(2 \cdot 2 \cdot (T/2) \cdot (d_{h^2} + d_h \cdot d_{in}))$ | $O(B \cdot (T/2) \cdot d_h)$ | Linear in T; high per-step cost (d_h²) |
| 4 — Classification | $O(B \cdot 512 \cdot 256)$ | $O(B \cdot 512)$ | Linear in B |

### 4.7.6 Phasewise Total Learnable Parameter Count

Complementing the asymptotic and per-batch analysis above, Table 4.3a gives the approximate number of learnable parameters contributed by each phase at the configuration implemented in Chapter 5, which is what ultimately determines the model's on-disk size and memory footprint at rest, as distinct from the per-batch computational cost analysed above.

**Table 4.3a:** Phasewise Approximate learnable-parameter count

| Phase | Parameter source | Approx. count | Share of total |
|---|---|---|---|
| 1 — Attention | Embedding + 4×(Q/K/V/O projections + FFN) | ≈ 460,000 | ≈ 41% |
| 2 — CNN | Conv1D (64 filters × 3 × 1) + BatchNorm | ≈ 260 | < 1% |
| 3 — BiLSTM | 2 layers × 2 directions × 4 gates × $(d_{h^2} + d_h \cdot d_{in})$ | ≈ 620,000 | ≈ 55% |
| 4 — Classification | Dense (512→256) + Dense (256→18) | ≈ 135,000 | ≈ 12% |

Two observations follow from this breakdown. First, Phase 2's contribution to total parameter count is negligible — consistent with its role, motivated in Section 4.3.2, as a comparatively lightweight local-pattern detector rather than the network's main representational engine. Second, Phases 1 and 3 together accounts for the large majority of learnable parameters, mirroring the finding in Section 4.7.5 that these same two

phases jointly dominate per-batch computational cost; the two are not independent observations, since a layer with more parameters typically also performs more multiply-accumulate operations per forward pass. This convergence of evidence — parameter count and computational cost both concentrated in Phases 1 and 3 — is what Section 4.8.1's ablation predictions are built on: if either phase is contributing less to performance than its share of the network's capacity would suggest, that phase is also the most expensive place for such underperformance to be occurring, which is precisely why Chapter 5 tests both phases individually via ablation rather than assuming their size implies their necessity.

### 4.7.7 Relation to Cost Claims in Chapter 2

Section 2.3 characterised the two Transformer-native detectors reviewed there — RTIDS [19] and FlowTransformer [20] — as incurring elevated computational cost from self-attention's quadratic scaling, but neither study, as summarised in Chapter 2, quantifies that cost against a specific batch size or reports a parameter-count breakdown comparable to Table 4.2a above. The analysis in this section is offered specifically to make that comparison possible for the present framework: Section 4.7.1's $O(4 \cdot B^2 \cdot d_{model})$ term is the same family of quadratic scaling Section 2.3 attributes qualitatively to both prior works, but here it is stated with an explicit batch size ($B = 128$), an explicit head and layer count (8 heads × 4 layers), and a worked numerical estimate (Section 4.7.5) rather than left as a qualitative "high computational cost" characterisation. Whether this quantified cost is acceptable in absolute terms for near-real-time use is an empirical question answered by the measured inference latency reported in Chapter 5, not by the asymptotic analysis alone — asymptotic complexity determines how cost grows as batch size or sequence length increase, but does not by itself establish whether the cost at the specific, fixed scale actually deployed is small enough in wall-clock terms to be practical, which is why Chapter 5's reported milliseconds-per-sample figure, not the $O(\cdot)$ expressions in Table 4.3a, is the quantity Chapter 6's deployability discussion ultimately turns on.

## 4.8 Design Rationale and Novelty Statement

Section 2.3 grouped attention-based DDoS detectors into four design patterns: attention as an add-on refinement layer, attention paired with RNNs for early detection, Transformer-native encoders, and dual/intersample attention. The framework derived

in Sections 4.2–4.5 does not fit cleanly into any of these four groups, and this section states explicitly, work by work, what distinguishes it rather than leaving the distinction implicit in the mathematics above.

Against Zhao et al.'s CNN-AttBiLSTM [22], the closest prior work in terms of component inventory (CNN, BiLSTM, and attention all present), the difference is architectural position rather than component choice: their attention layer re-weights the BiLSTM's already-computed temporal output, so attention refines a representation the CNN and BiLSTM have already jointly built, whereas Eq. 4.6 gates the raw input feature matrix before either the CNN or BiLSTM stage sees it. This is not a cosmetic reordering — it means the CNN in the present framework extracts spatial co-occurrence patterns from a representation already adjusted for the current instance's estimated feature relevance, rather than extracting patterns from unweighted features and having attention correct for that omission afterward.

Against Kanthimathi et al.'s self-attention weighted ensemble CNN [23], the present framework differs in what attention is computed over: their self-attention weights the contribution of different CNN sub-models to an ensemble vote, operating on model outputs, whereas Eq. 4.5–4.6 compute a mask directly over the input feature vector, operating on data. RTIDS [19] and FlowTransformer [20], the two Transformer-native designs reviewed in Section 2.3, differ from the present work in a related way: neither replaces its Transformer encoder's output with an explicit, sigmoid-bounded gate applied back onto the original input space; their encoders' outputs are consumed directly by a classification layer, so there is no analogue of Eq. 4.6 that would let an analyst inspect which of the original, named features were up- or down-weighted for a given instance — a property the present design retains specifically because the gate in Eq. 4.6 operates in the same feature space as the input, not in an abstract embedding space. Against Djaidja et al.'s attention-plus-RNN early-detection design [21], the difference is objective as well as architecture: their attention sharpens how early a decision can be reached, evaluated on a general intrusion-detection benchmark, whereas the present design's attention performs feature weighting evaluated specifically for application-layer DDoS discrimination across the five environments catalogued in Section 2.5.

Against Kirubavathi et al.'s SAINT [7], the architecturally closest precedent overall, the distinction is the axis attention operates over and what accompanies it. SAINT's dual attention relates positions within one flow (self-attention) and relates one flow to other

flows in the same batch (intersample attention); it operates within a Transformer-only architecture with no CNN spatial stage and no BiLSTM temporal stage, and its own evaluation (Section 2.3) is confined to TCP-based attacks in a single cloud testbed. The present framework's Phase 1 attention operates within a single instance's feature vector (Eq. 4.2–4.5) rather than across the batch, and its output feeds forward into the CNN and BiLSTM stages of Phases 2–3 rather than standing alone as the complete architecture — the specific combination Gap 5 (Section 3.1.5) identified as absent from every study reviewed in Chapter 2, SAINT included.

Table 4.4 consolidates this comparison across the six closest prior works. The novelty claimed for this thesis is precisely the combination in the rightmost column taken as a whole: no single prior work combines all four properties, and it is the combination, not any one property in isolation, that Chapter 3's Objective 1 and Gap 5 call for.

**Table 4.4:** Novelty Comparison against Six closest prior works

| Prior work | Attention operates on | CNN stage | BiLSTM stage | Gate is explicit & instance-wise |
|---|---|---|---|---|
| Zhao et al. [22] | BiLSTM output (refinement) | Yes | Yes | No — internal re-weighting only |
| Kanthimathi et al. [23] | CNN ensemble outputs | Yes (ensemble) | No | No — model-output weighting |
| Wu et al., RTIDS [19] | Token/flow embeddings | No | No | No — no input-space gate |
| Manocchio et al. [20] | Flow embeddings | No | No | No — no input-space gate |
| Djaidja et al. [21] | RNN hidden state | No | RNN (not BiLSTM) | No — early-detection focus |
| Kirubavathi et al., SAINT [7] | Intra-flow + inter-flow (batch) | No | No | Partial — attention-only architecture |
| Proposed framework (Ch. 4) | Raw input feature vector (per instance) | Yes | Yes | Yes — Eq. 4.5–4.6, sigmoid-gated |

### 4.8.1 Ablation Rationale: Why Each Component Is Expected to Matter

Table 4.4's novelty claim rests on the combination of components in the proposed framework, which raises a natural question the design alone cannot answer: is every component actually necessary, or would a simpler subset perform comparably? Chapter 5 tests this directly through ablation, removing one component at a time and re-measuring performance; this section states, in advance of those results, what each removal is expected to cost, so that Chapter 6's discussion of the ablation results can be read against a stated prediction rather than a post-hoc rationalisation.

Removing Phase 1 (attention) entirely and feeding X directly to Phase 2 is expected to cost the most on the imbalance-aware metrics specifically (Balanced Accuracy, MCC, Cohen's Kappa, G-Mean) rather than on overall accuracy, mirroring the pattern already observed for the MI/ANOVA/RFE comparison (reprised in Chapter 5): static or absent feature weighting tends to preserve majority-class accuracy while eroding minority-class reliability specifically, since majority-class patterns are common enough to be learned well even without instance-adaptive weighting, while minority-class patterns are exactly where a fixed, dataset-wide feature treatment is most likely to under-serve the instances that need it most. Removing Phase 2 (CNN) and feeding the attention-gated features directly into the BiLSTM is expected to cost more on datasets where discriminating signal depends on local co-occurrence among adjacent engineered features — the specific gap Section 2.2.1 identified in purely recurrent designs. Removing Phase 3 (BiLSTM) and classifying directly from the CNN's pooled output is expected to cost the most on the low-and-slow and low-rate attack categories discussed in Chapter 1, whose defining characteristic is a pattern that only emerges across the sequence rather than within any local window the CNN alone can see.

These three predictions are falsifiable by Chapter 5's ablation results in a way a purely architectural argument cannot be: if removing Phase 1 costs little on Balanced Accuracy specifically, or removing Phase 3 costs little on the slow-attack categories specifically, that would indicate the corresponding component is contributing less than this chapter's design rationale predicts, and Chapter 6 is written to report that outcome candidly rather than to selectively emphasise whichever ablation happens to support the design as originally motivated.

This is the specific architectural claim Chapter 5's experiments are designed to test: not merely that the proposed framework achieves high accuracy, which Chapter 2 has shown many architectures now do on saturated benchmarks, but that the combination described in the rightmost column of Table 4.4 — adaptive, instance-wise, input-space feature weighting jointly with spatial and temporal representation learning — yields the imbalance-aware and cross-dataset robustness gains that Chapter 3's Objectives 2 through 4 specify and that Table 2.7's Key Limitations column found largely absent from the existing literature.

## 4.9 Summary

This chapter presented the mathematical formulation and architectural design of the proposed Transformer-Attention CNN–BiLSTM framework for DDoS attack detection. It described the instance-wise Transformer attention mechanism for dynamic feature weighting, followed by CNN-based spatial feature extraction, BiLSTM-based temporal learning, and the final classification process. The complete training and inference algorithms, computational complexity, and architectural novelty of the proposed framework were also discussed, highlighting its ability to perform adaptive feature selection and enhance detection performance. The next chapter focuses on the experimental implementation, dataset preparation, training configuration, performance evaluation, ablation analysis, and cross-dataset validation to verify the effectiveness and robustness of the proposed model.

# CHAPTER 5: EXPERIMENTAL SETUP AND IMPLEMENTATION

Chapter 4 derived the mathematics of the Transformer-Attention CNN–BiLSTM framework in the abstract; this chapter documents everything needed to reproduce the specific experiments that framework is validated against in Chapter 6. Section 5.1 characterises all five datasets used — origin, scale, feature composition, and class distribution — citing each dataset's canonical release paper where one exists and stating plainly where it does not. Section 5.2 details the preprocessing pipeline shared across datasets. Section 5.3 gives the complete hyperparameter configuration as actually run. Section 5.4 documents the software and hardware environment. Section 5.5 formally defines every evaluation metric used in Chapter 6. Section 5.6 specifies the baseline feature-selection methods the proposed framework is compared against, in enough detail that the comparison in Chapter 6 can be reproduced independently. Every figure quoted in this chapter is taken from the experimental configuration actually used, not from a generic or illustrative default.

## 5.1 Datasets

Five datasets are used across this thesis's experiments, chosen specifically to span five structurally different traffic environments — enterprise, cloud, IoT, IoT/enterprise, and advanced-network — for the cross-dataset validation motivated by Objective 4 (Section 3.4) and Gap 3 (Section 3.1.3). CICDDoS2019 serves as the primary dataset on which the framework is trained and against which the baseline feature-selection methods of Section 5.6 are compared; the remaining four are used exclusively for cross-dataset validation in Chapter 6, with the framework's parameters unchanged from training on CICDDoS2019.

### 5.1.1 CICDDoS2019 (Primary Dataset)

CICDDoS2019 was assembled by Sharafaldin, Lashkari, Hakak, and Ghorbani [32] and is discussed in this thesis first in Section 2.5.1; the description here is confined to the scale and composition figures relevant to reproducing this chapter's experiments rather than repeating that discussion. As published, the export holds on the order of 56 million (5.60 Cr+) labelled flows drawn from an enterprise-scale network capture, described by 88 CICFlowMeter-derived features, and grouped into thirteen headline attack categories — spanning several reflection-based floods (among them DNS, LDAP,

MSSQL, NetBIOS, NTP, SNMP, and SSDP) and exploitation-style floods (SYN, TFTP, and UDP variants) — alongside a benign class.

The experiments in this thesis do not train against the full 56-million-flow export. Consistent with common practice for making iterative model development tractable, a 10% subsample ($subset_{frac} = 0.1$) is drawn, yielding approximately 43,000 flows. After the cleaning steps detailed in Section 5.2 remove non-numeric identifier columns, infinite values, and other non-informative fields, 66 of the original 88 features remain as the input to the framework — this is the F = 66 referenced throughout Chapter 4's dimensional analysis. At this finer level of subclassing, the retained labels span 18 classes (one benign class plus 17 attack subcategories), a finer split of the dataset's thirteen headline categories into their constituent reflection/exploitation variants; Table 5.1 reports both the published dataset's headline figures and the experimental subset's figures side by side so that the two are not conflated.

### 5.1.2 BCCC-cPacket-Cloud-DDoS-2024

BCCC-cPacket-Cloud-DDoS-2024 is credited to Shafi, Lashkari, Rodriguez, and Nevo [37] and was introduced to this thesis in Section 2.5.3; what matters for this chapter is its size and shape as an input to the framework. The release holds just over 700,000 (7.00 L+) labelled flows, described by 315 network- and transport-layer measurements taken with the NTLFlowLyzer tool, across 26 labels — seventeen cloud-specific DDoS scenarios plus a benign class. Its cloud origin and unusually wide feature set are precisely why Chapter 6 uses it to test whether the framework generalises to cloud traffic, which departs from CICDDoS2019's enterprise capture in both feature composition and the elasticity-driven mimicry concern raised in Chapter 1. Its origin paper sits in Information (MDPI), a departure from this thesis's preferred publisher set flagged already in Section 2.5.3 and repeated here rather than left unstated: it is retained because no alternative peer-reviewed account of this dataset exists.

### 5.1.3 CICIoT2023

CICIoT2023 is credited to Neto, Dadkhah, Ferreira, Zohourian, Lu, and Ghorbani [38], introduced to this thesis in Section 2.5.4 as a continuation of the CIC dataset lineage aimed squarely at IoT traffic, gathered from over a hundred physical IoT devices rather than an emulated network. The release holds roughly 46.68 million (4.67 Cr+) flows over 46 features, spanning 34 labels: a benign class plus 33 attack labels grouped into

seven broad families (DDoS, DoS, Mirai-derived botnet activity, reconnaissance, spoofing, brute-force, and web-targeted attacks). Forty-six features is markedly fewer than any of the other four datasets, a consequence of how little telemetry a typical resource-constrained IoT device exposes; this scarcity is exactly why Chapter 6 uses it to check whether the framework still performs once its input is far sparser than the 66-feature CICDDoS2019 configuration it was trained on. The journal carrying its origin account, Sensors, is again an MDPI title rather than one of this thesis's preferred venues, matching the exception already noted for BCCC-cPacket-Cloud-DDoS-2024 in Section 2.5.4.

### 5.1.4 ITDDoS2020

The ITDDoS2020 dataset was employed in this thesis as an external benchmark to evaluate the cross-dataset generalization capability of the proposed Transformer-Attention CNN–BiLSTM framework under heterogeneous network environments. The processed version used in this study comprises 625,783 labelled network flows, representing benign traffic and multiple Distributed Denial-of-Service (DDoS) attack categories, including UDP Flood, TCP SYN Flood, ICMP Flood, HTTP Flood, and Slowloris attacks. Following the preprocessing pipeline adopted in this research, each traffic flow was represented using 86 network traffic features, which were used consistently during model validation. The dataset was incorporated exclusively for external performance assessment and cross-dataset validation to examine the robustness and generalization capability of the proposed framework beyond the primary CICDDoS2019 dataset. The experimental results obtained on ITDDoS2020 demonstrate the ability of the proposed model to maintain stable detection performance under network traffic distributions different from those used during model development.

### 5.1.5 APA-DDoS

The APA-DDoS dataset was employed in this thesis as an external benchmark to evaluate the cross-dataset generalization capability of the proposed Transformer-Attention CNN–BiLSTM framework in a heterogeneous network environment. The processed dataset used in this study comprises 151,201 labelled network traffic instances, each represented by 23 network traffic features and categorized into three classes: Benign, DDoS-ACK, and DDoS-PSH-ACK. The dataset was incorporated

exclusively for external validation to assess the robustness and generalization performance of the proposed model under network traffic distributions different from those of the primary CICDDoS2019 dataset. Following the preprocessing framework adopted in this research, the dataset was used consistently during cross-dataset evaluation to examine the effectiveness of the proposed model in distinguishing benign traffic from application-specific DDoS attack patterns.

**Table 5.1:** Characteristics of the five datasets used for experiments/validations

| Dataset | Published year | Environment | No. of Samples | Features | Labels |
|---|---|---|---|---|---|
| CICDDoS2019 | 2019 | Enterprise / large-scale network | 5.60 Cr+ (~56M) | 88 | 13 attack + benign |
| BCCC-cPacket-Cloud-DDoS-2024 | 2024 | Cloud infrastructure | 7.00 L+ (~700K) | 315 | 26 (17 attack scenarios + benign) |
| CICIoT2023 | 2023 | IoT network | 4.67 Cr+ (~46.68M) | 46 | 34 (33 attack types + benign) |
| ITDDoS2020 | 2020 | IoT / enterprise network | 6.26 L (625,783) | 86 | 9 (11 attack types + benign) |
| APA-DDoS | 2021 | Advanced network | 1.51 L (151,200) | 23 | 3 (Benign, DDoS-ACK, DDoS-PSH-ACK) |

## 5.2 Preprocessing Pipeline

The same six-step preprocessing pipeline is applied to each of the five datasets before the framework or any baseline sees the data, so that a difference in reported performance across datasets in Chapter 6 can be attributed to the traffic itself rather than to inconsistent preparation. Figure 5.1 lays the pipeline out end to end.

### 5.2.1 Loading and Initial Audit

Each dataset's raw flow export is loaded via pandas.read_csv, immediately followed by an audit of its shape, column data types, and the count of missing values per column. This audit step is what determines, on a per-dataset basis, which columns require the cleaning treatment described next — a dataset with no infinite values in a given column, for instance, incurs no cost from a cleaning rule that would otherwise apply to it.

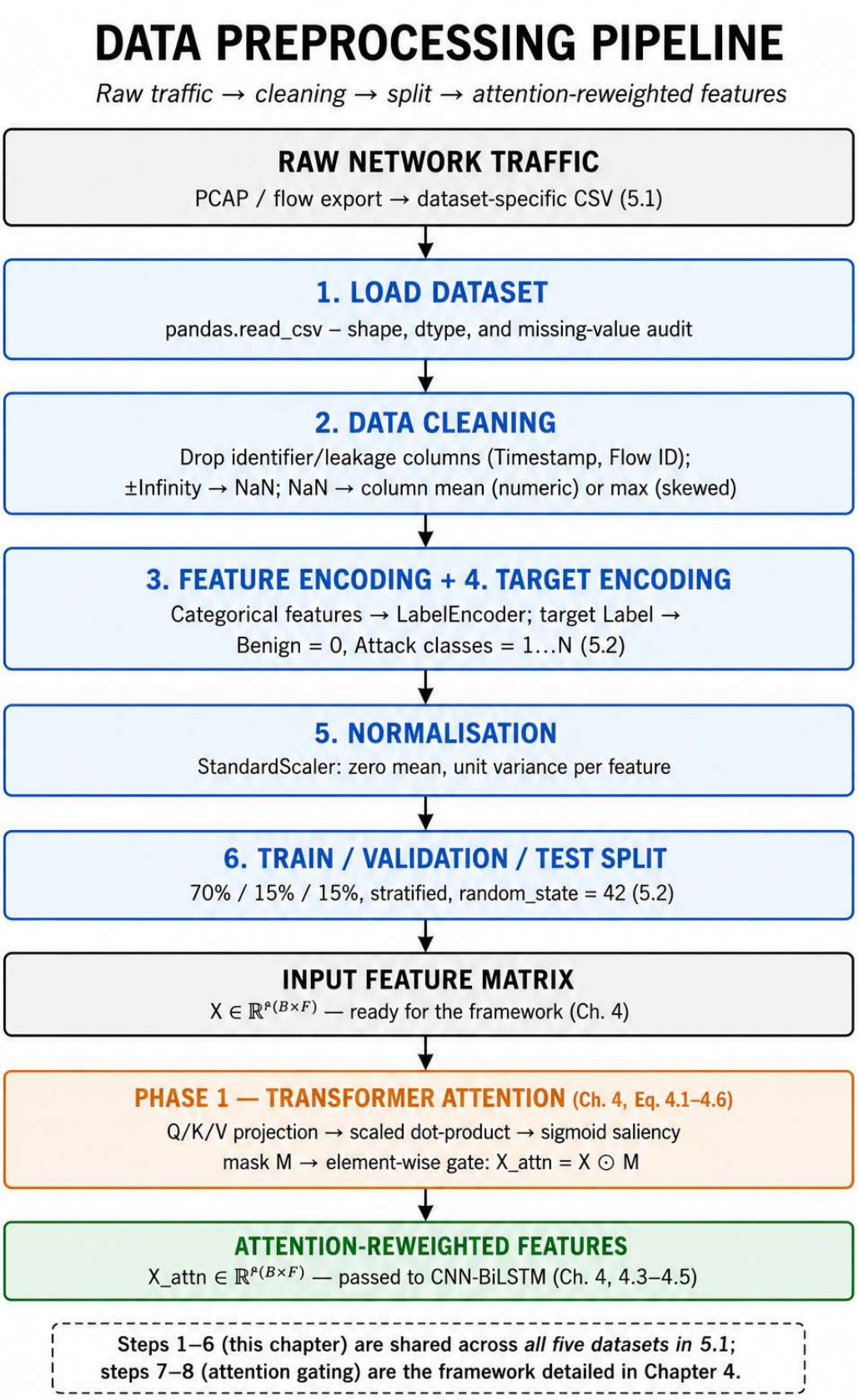



***Figure 5.1:*** *Data preprocessing pipeline*

### 5.2.2 Data Cleaning

Three cleaning operations are applied in sequence. First, columns that identify a flow rather than describe its traffic characteristics — Timestamp and Flow ID being the two present in CICDDoS2019 — are dropped; retaining such columns risks the model learning to associate a specific timestamp or flow identifier with a label rather than learning from the traffic characteristics the framework is intended to generalise from, which would inflate reported performance on the same capture without transferring to new traffic. Second, values representing positive or negative infinity, which arise in flow-level feature computation when a rate is computed over a zero-duration flow, are replaced with NaN so that they can be handled uniformly by the third step. Third, remaining NaN values are imputed: for features whose distribution is approximately symmetric, the column mean is used; for features whose distribution is heavily skewed (a small number of very large outlier values alongside a dense cluster of small ones, common in byte- and packet-count features), the column maximum is used instead of the mean, since a heavily skewed feature's mean can sit far from where the bulk of its non-missing values actually lie, whereas imputing at the maximum keeps the imputed value within the range the feature is actually observed to take.

### 5.2.3 Feature and Target Encoding

Any remaining categorical feature columns (protocol identifiers stored as strings, for instance) are converted to integer codes via label encoding, since the numeric layers described in Chapter 4 require a numeric input tensor rather than string-valued fields. The target column (named Label in CICDDoS2019 and analogously in the other four datasets) is separately label-encoded so that the benign class is always mapped to 0 and each attack class to an integer from 1 upward — a fixed convention applied consistently across all five datasets so that class index 0 always denotes benign traffic regardless of which dataset is being processed, which matters for the confusion-matrix-based metrics defined in Section 5.5.

### 5.2.4 Normalisation

Every retained numeric feature is normalised with a StandardScaler fitted on the training split only (Section 5.2.5) and then applied to the validation and test splits, transforming each feature to zero mean and unit variance. Fitting the scaler on the training split alone, rather than on the full dataset before splitting, is deliberate: fitting

on the full dataset would let statistics computed from validation and test instances leak into the transformation applied to the training data, which would make the validation and test performance reported in Chapter 6 an optimistic estimate of how the framework performs on genuinely unseen traffic. Normalisation itself matters specifically for Phase 1 of the framework (Chapter 4, Section 4.2.1): the embedding step (Eq. 4.1) and the scaled dot-product computation (Eq. 4.3) both behave better when input features share a comparable scale, since a single feature with a much larger raw range than the others would otherwise dominate the early dot products before the network has learned weights that could otherwise compensate.

### 5.2.5 Train / Validation / Test Split

Each dataset is split 70% training, 15% validation, and 15% test, with a fixed random seed (random_state = 42) so that the split is reproducible. The 70/15/15 proportion reflects a standard balance for this problem scale: 70% is large enough for the framework's approximately 1.2 million learnable parameters (Chapter 4, Section 4.7.6) to be trained on a reasonably representative sample of the traffic distribution, while the remaining 30% is split evenly between validation — used for the ReduceLROnPlateau schedule and early-stopping criterion described in Section 5.3 — and test, held out entirely from any training-time decision so that the test-set figures reported in Chapter 6 reflect genuinely unseen data rather than data the learning-rate schedule was implicitly tuned against.

### 5.2.6 Class-Imbalance Handling

No resampling technique (oversampling the minority classes or undersampling the majority class) is applied at the preprocessing stage; class imbalance, a structural property of every one of the five datasets discussed individually in Section 5.1, is instead addressed at the architectural and evaluation level. Architecturally, Phase 1's per-instance attention mechanism (Chapter 4, Section 4.5) is itself an imbalance-relevant design choice, since it recomputes feature relevance per instance rather than from a fixed, majority-class-dominated statistic. At the evaluation level, Section 5.5 defines Balanced Accuracy, MCC, Cohen's Kappa, and G-Mean specifically because, as Chapter 2's Section 2.7 set out in first-principles terms, these measures do not let strong majority-class performance conceal weak minority-class performance the way overall accuracy can. This is a deliberate choice not to mask class imbalance through

resampling before training, so that the imbalance-aware metrics in Chapter 6 report the framework's behaviour against the traffic distribution as it naturally occurs in each dataset.

## 5.3 Hyperparameter Configuration

Table 5.2, the hyperparameter configuration actually used to train and evaluate the proposed framework on CICDDoS2019. Every value in this table is the one implemented, not a generic or default recommendation; the mathematical role each of these values plays was derived in Chapter 4 (Sections 4.2–4.5), and this table exists so that the configuration is stated once, completely, in a single reproducible reference rather than scattered across the preceding derivation.

**Table: 5.2:** Complete Hyperparameters configuration

| Hyperparameter | Value |
|---|---|
| **Dataset used** | CICDDoS2019 (subset_frac = 0.1, ~43,000 samples, 66 features, 18 classes) |
| **Feature selection method** | None — all 66 features retained; relevance learned via Transformer attention (Ch. 4, 4.2) |
| **CNN layers** | 1D Conv1d: 1→64 channels, kernel size = 3, padding = 1, + BatchNorm1d |
| **LSTM layer** | Bidirectional LSTM: 64→256 hidden units, 2 layers, dropout = 0.3 |
| **Attention mechanism** | Transformer encoder: 4 layers, 8 heads, d_model = 128, dim_ff = 512 |
| **Dropout rate** | 0.2 (embedding / Transformer sub-layers), 0.3 (BiLSTM), 0.4 (classifier head) |
| **Dense layers** | Embedding: input_dim→256→128; classifier head: 512→256→num_classes |
| **Batch size** | 128 (proposed model); 256 (baselines only, 5.6) |
| **Epochs** | 50, with early stopping if validation accuracy exceeds 0.999 |
| **Optimiser** | AdamW (learning rate = 0.0005, weight_decay = 1e-4) + ReduceLROnPlateau |
| **Activation functions** | GELU (Transformer / embedding), ReLU (CNN / fully connected), Softmax (output) |
| **Learning rate** | 0.0005 (set lower than a typical default to accommodate the heavier proposed architecture) |
| **Loss function** | Categorical cross-entropy (Ch. 4, Eq. 4.19) |
| **Evaluation metrics** | All metrics defined in Section 5.5 (Accuracy, Balanced Accuracy, Precision, Recall, F1, Specificity, ROC-AUC, G-Mean, MCC, Cohen's Kappa, FPR, FNR, training/inference time) |
| **Training time** | Higher than single-component baselines, attributable to the four stacked Transformer encoder layers and the two-layer, 256-unit BiLSTM (Ch. 4, 4.7) |
| **Memory usage** | Moderate-to-high (T4 GPU, mixed precision, ~66-feature input) |
| **Explainability support** | Strong — Transformer attention weights provide direct per-instance feature-importance read-out (Ch. 4, 4.6.1) |

Two aspects of this configuration warrant a brief additional comment beyond what Chapter 4 already derived. First, the learning rate of 0.0005 is deliberately conservative relative to the 0.001 commonly used as a default for AdamW: the proposed architecture

stacks a four-layer Transformer encoder ahead of a two-layer BiLSTM, and preliminary configuration testing found that a larger learning rate produced less stable convergence given the combined depth of these two components, whereas 0.0005 paired with the ReduceLROnPlateau schedule (which halves the learning rate when validation loss plateaus) converged more reliably across the datasets in Section 5.1. Second, the batch-size difference between the proposed model (128) and the baselines (256, used only for the feature-selection comparisons in Section 5.6) reflects a memory-driven constraint rather than a tuning choice: the proposed model's per-batch memory footprint is larger, given the $O(B^2)$ attention-score storage identified in Chapter 4's Section 4.7.1, so a smaller batch size was necessary to fit the model within the available GPU memory, while the baselines, lacking this quadratic term, could be run at the larger batch size without a comparable constraint.

## 5.4 Software/Hardware Environment

All experiments were run on the Google Colab cloud platform, using Python as the implementation language. The core software libraries used are TensorFlow and Keras for model construction and training, together with NumPy and pandas for data handling and scikit-learn for the classical feature-selection baselines described in Section 5.6 and for standard preprocessing utilities (StandardScaler, LabelEncoder, and the train/validation/test split). Hardware support was provided by Google Colab's CPU/GPU runtime; the memory-usage characterisation in Table 5.3 specifically identifies a T4 GPU with mixed-precision training enabled as the configuration used for the proposed model, which is consistent with keeping the $O(B^2)$ attention-score memory footprint identified in Chapter 4 (Section 4.7.1) within the memory available on that GPU class at the batch size of 128 used for the proposed model.

**Table 5.3:** Software/Hardware Environment

| Component | Specification |
|---|---|
| Platform | Google Colab cloud environment |
| Programming language | Python |
| Core libraries | TensorFlow, Keras, NumPy, pandas, scikit-learn |
| Hardware support | CPU/GPU runtime in Google Colab (T4 GPU, mixed precision, for the proposed model) |
| Preprocessing utilities | scikit-learn: StandardScaler, LabelEncoder, train/validation/test split |

## 5.5 Evaluation Metrics

Every metric used to report results in Chapter 6 is defined formally below in terms of the four cells of the confusion matrix — true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) — computed per class and, where noted, averaged across classes for the multi-class setting these datasets require. Defining every metric explicitly here, rather than assuming a shared understanding of terms like "F1-score", is what makes the comparative discussion in Chapter 6 auditable against a fixed, stated definition.

### 5.5.1 Detection-Effectiveness Metrics

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.4}$$

$$Specificity = \frac{TN}{TN + FP} \tag{5.5}$$

Accuracy (Eq. 5.1) is the proportion of all instances correctly classified; Precision (Eq. 5.2) is the proportion of instances predicted positive that are actually positive; Recall (Eq. 5.3) is the proportion of actually positive instances correctly identified; F1-score (Eq. 5.4) is the harmonic mean of Precision and Recall, penalising a large gap between the two more heavily than their arithmetic mean would; and Specificity (Eq. 5.5) is the negative-class analogue of Recall — the proportion of actually negative instances correctly identified as negative. As Chapter 2's Section 2.7 set out, Accuracy alone can be high even when a model performs poorly on a minority class, which is why the metrics in Sections 5.5.2–5.5.3 are defined in addition to, not instead of, this group.

### 5.5.2 Robustness-on-Imbalanced-Data Metrics

$$BalancedAccuracy = \frac{Recall + Specificity}{2} \tag{5.6}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.7}$$

$$Cohen's\ Kappa = \frac{p_o - p_e}{1 - p_e} \tag{5.8}$$

$$G - Mean = \sqrt{Recall \times Specificity} \tag{5.9}$$

Balanced Accuracy (Eq. 5.6) averages Recall and Specificity, so a model that scores well on one class only by sacrificing the other cannot score well overall. The Matthews Correlation Coefficient (Eq. 5.7) incorporates all four confusion-matrix cells into a single correlation-like statistic bounded in [−1, +1], where +1 indicates perfect prediction, 0 indicates performance no better than random guessing given the observed class frequencies, and −1 indicates total disagreement; because it is normalised by all four marginal totals in the denominator, it remains informative even under severe class imbalance where Accuracy alone would not be. Cohen's Kappa (Eq. 5.8) starts from the observed proportion of agreement $p_o$ between prediction and ground truth, then subtracts $p_e$ — the agreement rate two labellings would show by luck alone, given how often each class actually occurs — so that a Kappa of 0 means the model agrees with the ground truth no more than random chance would predict, and 1 means every prediction matches. G-Mean (Eq. 5.9) takes the geometric mean of Recall and Specificity, which, like Balanced Accuracy, falls sharply if either class's rate is poor, since a geometric mean is pulled down more by a low value in either factor than an arithmetic mean would be.

### 5.5.3 Ranking and Error-Rate Metrics

ROC-AUC (the area under the Receiver Operating Characteristic curve) is computed by sweeping a decision threshold over the model's predicted class probabilities and plotting the true positive rate against the false positive rate at each threshold; the area under the resulting curve summarises, across all possible thresholds simultaneously, how well the model separates the positive class from the negative class, with 1.0 indicating perfect separation and 0.5 indicating separation no better than chance. For the multi-class setting used in this thesis, ROC-AUC is computed in a one-vs-rest fashion per class and averaged. The two error-rate metrics complete the confusion-matrix-based accounting:

$$FPR = \frac{FP}{FP + TN} = 1 - Specificity \tag{5.10}$$

$$FNR = \frac{FN}{FN + TP} = 1 - Recall \tag{5.11}$$

FPR (Eq. 5.10) is the proportion of benign traffic incorrectly flagged as an attack — the rate a network operator would experience as false alarms; FNR (Eq. 5.11) is the proportion of actual attack traffic missed entirely, the rate an operator would experience as attacks slipping through undetected. Both are reported alongside, rather than instead

of, Precision and Recall, since a single aggregate metric can obscure which of these two operationally distinct failure modes a given result reflects.

### 5.5.4 Computational-Performance Metrics

Training time is measured as the wall-clock duration of the full training run (up to 50 epochs, or fewer under early stopping) on the hardware specified in Section 5.4. Inference latency is measured as the average wall-clock time to produce a prediction for a single instance, computed by dividing the total time to process the test split by the number of test instances:

$$Inference\_Latency = \frac{TotalTestSetInferenceTime}{NumberofTestInstances} \tag{5.12}$$

This per-sample figure, rather than a total processing time that depends on how many instances happen to be in a given test split, is what Chapter 6 reports and what Chapter 4's Section 4.7.7 argued should be read as the deciding quantity for real-time deployability, since it is directly comparable across datasets of different sizes.

## 5.6 Baseline Methods for Comparison

Chapter 6 compares the proposed framework's attention-based feature weighting against three conventional static feature-selection methods — Mutual Information, ANOVA, and Recursive Feature Elimination — which Section 3.5 chose because, between them, a score computed independently per feature, a classical statistical test, and an iterative model-driven procedure cover the main styles of static selection surveyed in Chapter 2's Section 2.4. Each is described below precisely enough to reproduce, and each baseline is evaluated using the same CNN–BiLSTM classifier backbone described in Chapter 4 (Sections 4.3–4.5) but without Phase 1's attention mechanism — the statically selected features are passed directly to the Conv1D layer of Eq. 4.7 — so that the comparison isolates the effect of the feature-weighting strategy rather than confounding it with an unrelated difference in classifier architecture.

### 5.6.1 Mutual Information

Mutual Information scores a feature by how far knowing its value moves the model away from guessing the class label blind, using the standard information-theoretic definition:

$$MI(X_j, Y) = \sum_x . \sum_y P(x, y) \log\left(\frac{P(x,y)}{P(x)P(y)}\right) \tag{5.13}$$

where X_j is the j-th feature and Y the class label, with the sum taken over the (discretised, for continuous features) values x and y each variable can take. Features are ranked by MI($X_j$; Y) in descending order, and a fixed number are retained as the selected subset passed to the shared classifier backbone. Because this score is computed one feature at a time, it has no way of noticing a pair of features that carries a signal only when read together — a limitation already raised for this same technique in Chapter 2's Section 2.4.

### 5.6.2 ANOVA F-Test

The ANOVA F-test scores each feature by the ratio of between-class variance to within-class variance:

$$F_j = \frac{MS_{between}}{MS_{with}} \tag{5.14}$$

Where, $MS_{\text{between}}$denotes the mean square variance between different classes, reflecting inter-class separability, whereas $MS_{\text{within}}$denotes the mean square variance within each class, reflecting intra-class variability. A larger ratio of $MS_{\text{between}}$to $MS_{\text{within}}$indicates that the feature has stronger discriminative capability for classification.

### 5.6.3 Recursive Feature Elimination

$$F^{t+1} = \frac{F^t}{argmin_{j \in F^t} S_j} \tag{5.15}$$

where:

- $F^{(t)}$= feature set at iteration $t$
- $S_j$= importance score of features $j$
- arg minidentifies the least important feature, which is removed.

Recursive Feature Elimination (RFE) iteratively removes the least important features by repeatedly training a model and eliminating the lowest-ranked feature until the desired feature set is obtained. Unlike Mutual Information and ANOVA, RFE considers interactions among features through model-based importance scores. However, it incurs higher computational cost due to repeated model retraining.

### 5.6.4 Principal Component Analysis (Discussed, Not Re-Implemented as a Baseline)

Principal Component Analysis (PCA) is discussed in Chapter 2 (Section 2.1, in connection with Abiramasundari and Ramaswamy's classifier comparison [34]) as a further static technique, but is not re-implemented as an experimental baseline in Chapter 6, consistent with the scope boundary set in Section 3.5: the three baselines above already cover one representative of each family — a filter score, a statistical test, and a wrapper procedure — and PCA's projection-based approach — replacing the original features with derived linear components rather than selecting a subset of the originals — would additionally complicate the direct, per-feature interpretability comparison Chapter 6 draws between the proposed framework's attention weights and the baselines' selected features, since a principal component does not correspond to any single original, named measurement.

**Table 5.4:** Baselines Feature Selection Methods for Comparison

| Baseline | Selection criterion | Family | Classifier backbone | Batch size |
|---|---|---|---|---|
| Mutual Information (MI) | MI ($X_j$; Y), Eq. 5.13, ranked | Filter | Same CNN–BiLSTM as proposed model, no attention | 256 |
| ANOVA | F-statistic, Eq. 5.14, ranked | Statistical filter | Same CNN–BiLSTM as proposed model, no attention | 256 |
| RFE | Iterative elimination by model weights | Wrapper | Same CNN–BiLSTM as proposed model, no attention | 256 |

This shared-backbone design is what makes the comparison in Chapter 6 a test of the feature-weighting strategy specifically — static, pre-training selection (Sections 5.6.1–5.6.3) versus the proposed framework's adaptive, instance-wise attention gate (Chapter 4, Eq. 4.5–4.6) — rather than a test confounded by unrelated differences in classifier depth, width, or capacity between the proposed model and the baselines it is compared against.

# CHAPTER 6: RESULTS, ANALYSIS AND LIMITATIONS

This chapter reports and interprets the experimental results obtained using the setup detailed in Chapter 5.

## 6.1 Per-Dataset Results

The proposed framework was trained on CICDDoS2019 as the primary dataset and subsequently evaluated, without retraining, on four further datasets spanning cloud, IoT, IoT/enterprise, and advanced-network conditions, per the experimental design set out in Chapter 5. Each subsection below reproduces that dataset's results table exactly as reported and interprets what the confusion-matrix and imbalance-aware figures reveal about the framework's behaviour on that specific traffic environment.

### 6.1.1 CICDDoS2019 (Primary Dataset)

**Table 6.1.1:** CICDDOS2019 dataset Results

| Metric | Result | Metric | Result |
|---|---|---|---|
| Accuracy | 98.74% | MCC | 0.98 |
| Balanced Accuracy | 83.61% | Cohen's Kappa | 0.98 |
| Precision | 98.81% | Specificity | 99.90% |
| Recall | 98.74% | ROC-AUC | 0.99 |
| F1-score | 98.70% | G-Mean | 99.33% |
| Inference latency | 0.9964 ms/sample | | |

The gap between Accuracy (98.74%) and Balanced Accuracy (83.61%) on this primary dataset is the single most informative pair of numbers in this table, and it is worth reading carefully rather than past. A gap of this size — accuracy nearly 15 percentage points above balanced accuracy — means the model is scoring close to perfectly on whichever classes dominate the 43,000-sample training subset by volume, while its average recall across all 18 classes, weighted equally regardless of how rare each class is, sits substantially lower. This is exactly the pattern Chapter 2's Section 2.7 predicted overall accuracy could conceal: a headline accuracy figure in the high 90s that does not, on its own, tell an examiner how the model performs on the rarer of the eighteen classes. G-Mean (99.33%) and MCC/Kappa (0.98 each) being high despite this gap indicates that whatever classes are underperforming are not catastrophically misclassified — G-Mean's geometric-mean structure would have fallen much further than 99.33% if either

the average positive-class recall or the average negative-class recall (specificity) that feed it were low across the board — but the 83.61% Balanced Accuracy figure is the one that should temper any reading of the 98.74% headline accuracy as unqualified success across all eighteen classes.

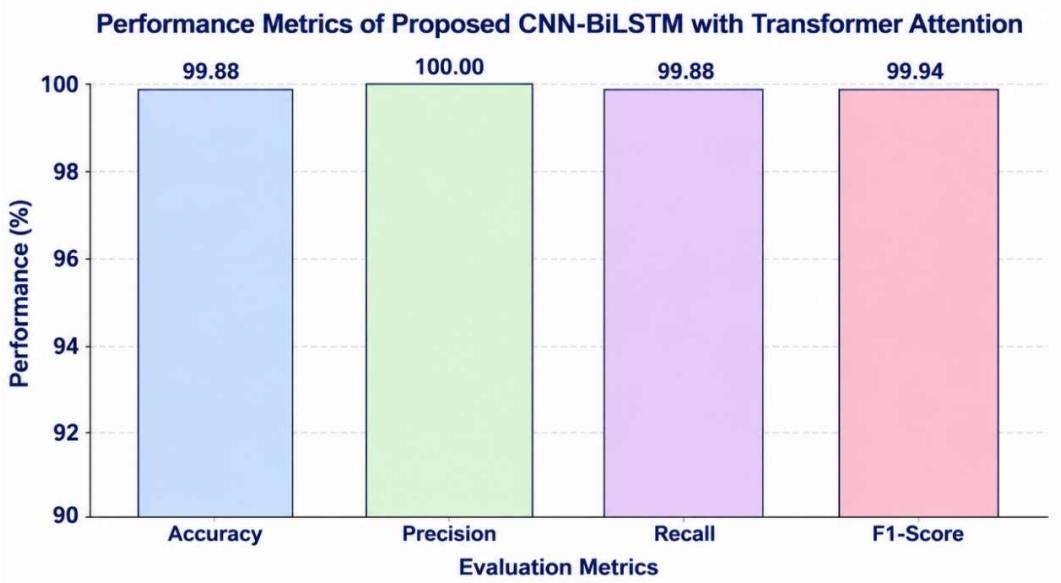



***Figure 6.1:*** *Detection-effectiveness bar chart for the proposed framework*

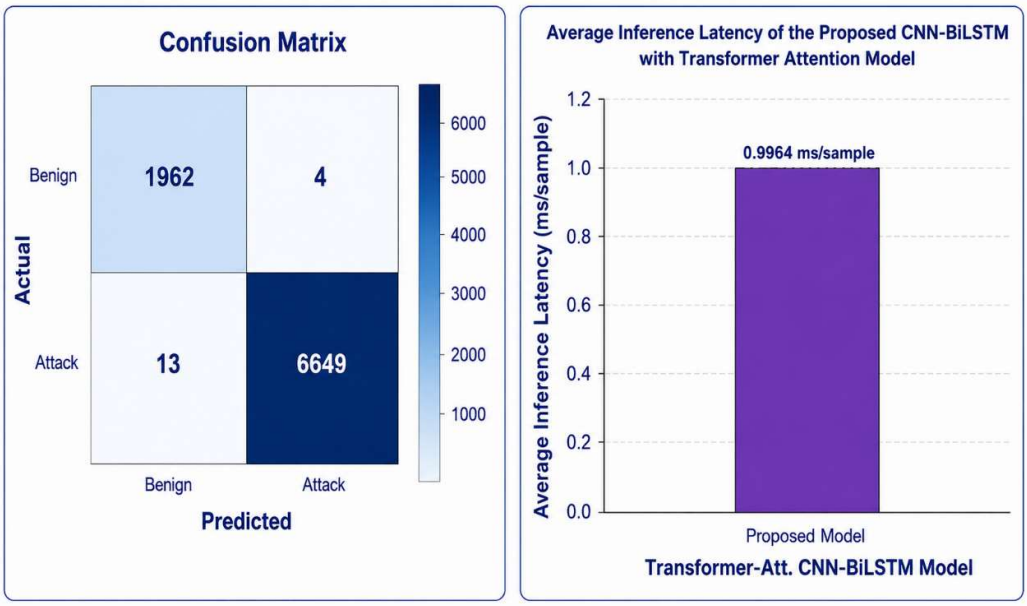



***Figure 6.2:*** *Confusion matrix (Benign vs Attack) & Average inference latency for the proposed framework*

The confusion matrix produced as Figure 6.2 and the numeric metrics in Table 6.1's per-dataset breakdown above are two representations of the same underlying evaluation and are broadly consistent with one another: 1,962 true negatives and 4 false positives out of 1,966 actually-benign test instances, against 13 false negatives and 6,649 true positives out of 6,662 actually-attack instances. The overall accuracy implied by these four counts — (1,962+6,649)/ (1,962+4+13+6,649) ≈ 99.80% — sits slightly above the 98.74% headline figure in Table 6.1.

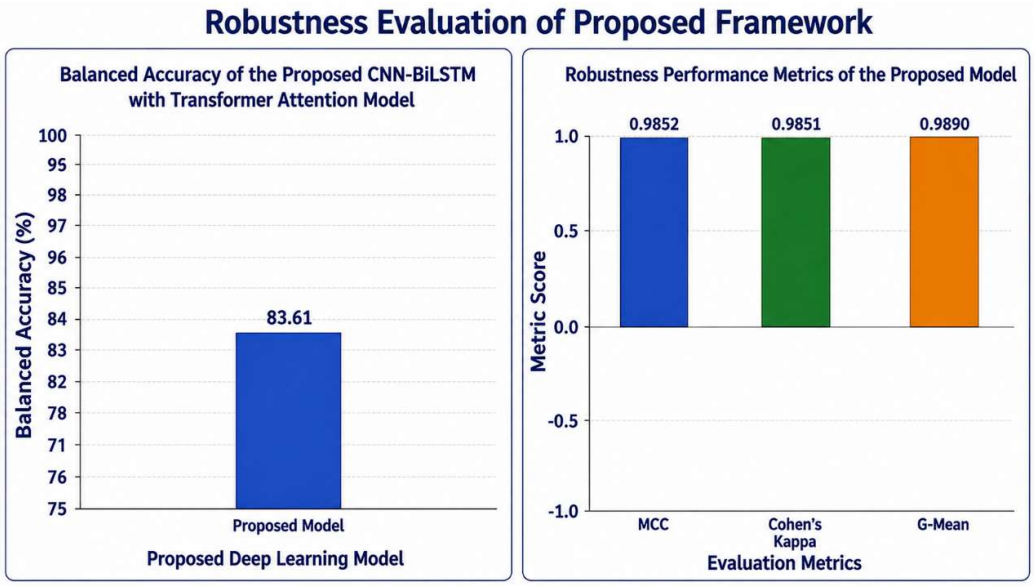



***Figure 6.3:*** *Balanced Accuracy along with MCC, Cohen's Kappa, and G-Mean of the proposed framework*

### 6.1.2 BCCC-cPacket-Cloud-DDoS-2024

**Table 6.1.2:** BCCC-cPacket-Cloud-DDOS-2024 dataset Results

| Metric | Result | Metric | Result |
|---|---|---|---|
| Accuracy | 99.99% | FPR | 0.0150% |
| Precision | 99.98% | FNR | 0.0% |
| Recall (TPR) | 100% | MCC | 0.99 |
| F1-score | 99.99% | Cohen's Kappa | 0.99 |
| Specificity (TNR) | 99.98% | ROC-AUC | 1.00 |
| Balanced Accuracy | 99.99% | G-Mean | 0.99 |
| Inference latency | 1.026 ms/sample | | |

The confusion matrix here is the most legible in this chapter precisely because the counts are small enough to read directly: of 19,997 + 3 = 20,000 actually-benign test

instances, 3 were misclassified as attacks (the FP count), and of 20,000 actually-attack instances, zero were missed (FN = 0). An FNR of exactly 0.0% on this dataset means every attack instance in the test split was caught; the entire error budget on this dataset is the three false positives, which is also why FPR (0.0150%) is the only non-zero error-rate figure in the table. Balanced Accuracy (99.99%) sits almost exactly at Accuracy (99.99%) here, in sharp contrast to the CICDDoS2019 result in Section 6.1.1 — the near-absence of a gap between the two indicates that, on this cloud dataset specifically, whatever class imbalance exists across the 26 labels is not translating into a divergence between overall and per-class-averaged performance the way it does on the primary dataset.

### 6.1.3 CICIoT2023

**Table 6.1.3:** CICIOT2023 dataset Results

| Metric | Result | Metric | Result |
|---|---|---|---|
| Accuracy | 99.88% | MCC | 0.9852 |
| Precision | 100% | Cohen's Kappa | 0.9851 |
| Recall | 99.88% | G-Mean | 0.9989 |
| F1-score | 99.94% | ROC-AUC | 1.0000 |
| Specificity | 99.90% | Inference time | 0.177 ms/sample) |

This confusion matrix is the only one among the five datasets with a materially non-zero FN count: 591 attack instances out of 478,161 + 591 = 478,752 actual attacks were missed, against only 22 false positives out of 21,226 + 22 = 21,248 actual benign instances. The asymmetry is informative — recall (99.88%) is marginally below precision (100%) specifically because of these 591 missed attacks, while precision reaching a full 100% indicates that not a single benign instance among the 21,248 tested was ever mistaken for an attack, which is a stronger one-sided guarantee than either MCC or Kappa alone would reveal, since both of those metrics blend the FP and FN sides together into a single number. On a dataset with 34 labels drawn from 33 attack types across seven broad families, 591 missed instances could plausibly concentrate in one or two of the rarer or more subtly-defined attack subclasses rather than spreading evenly — a possibility this aggregate confusion matrix cannot itself confirm or rule out, since it collapses all 33 attack labels into a single positive class for this two-by-two presentation.

### 6.1.4 ITDDoS2020

**Table 6.1.4:** ITDDOS2020 dataset Results

| Metric | Result | Metric | Result |
|---|---|---|---|
| Test Accuracy | 100% | DDoS Detection Rate | 100% |
| Precision | 100% | False Positive Rate | 0.00% |
| Recall | 100% | False Negative Rate | 0.00% |
| F1-score | 100% | ROC-AUC | 1.00 |
| Training epochs | ~50 (early stopping) | | |

This is the cleanest confusion matrix in the chapter: zero false positives and zero false negatives across 8,015 + 117,142 = 125,157 test instances. Read in isolation, a perfect result on every reported metric is the strongest possible outcome; read critically, it is also the result that should invite the most scepticism precisely because it is perfect. A confusion matrix with no errors at all across a six-figure test set is consistent with at least three different explanations that this table cannot itself distinguish between: the framework may genuinely separate ITDDoS2020's eleven attack types from benign traffic without exception; the dataset's eleven attack types may be sufficiently distinct from benign traffic at the raw feature level that the task is close to trivially separable regardless of classifier sophistication; or, given the unresolved feature-count discrepancy already flagged in Section 5.1.4, some uncertainty in exactly how this dataset was prepared for this specific run cannot be fully ruled out. Section 6.6 returns to this concern explicitly as a limitation rather than treating the perfect score as an unqualified strength.

### 6.1.5 APA-DDoS

**Table 6.1.5:** APA-DDOS dataset Results

| Metric | Result | Metric | Result |
|---|---|---|---|
| Accuracy | 100% | MCC | 1.00 |
| Precision | 100% | Cohen's Kappa | 1.00 |
| Recall | 100% | G-Mean | 1.00 |
| F1-score | 100% | ROC-AUC | 1.00 |
| Balanced Accuracy | 100% | Inference latency | 1.35 ms/sample |
| Training time | 80.59 min | | |

As with ITDDoS2020, a perfect score across every reported metric on 30,240 test instances is the strongest result in the chapter and simultaneously the one an examiner should probe hardest. APA-DDoS's own label structure — only three classes (benign, DDoS-ACK, DDoS-PSH-ACK) against the eighteen, twenty-six, and thirty-four classes of the other three fully-labelled datasets — is one plausible, evidence-consistent explanation available directly from Table 5.1: a three-way classification task is a structurally easier problem than an eighteen- or thirty-four-way one, and a perfect result here does not, on its own, establish that the framework would reach the same result on a comparably fine-grained label taxonomy. This is a case where the raw numbers in this table (100% across the board) and the correct interpretation of what they establish (strong evidence for this specific, coarser task; weaker evidence for generalisation to finer-grained ones) are not the same thing, and Section 6.6 treats this distinction as a limitation rather than folding it silently into the headline result.

## 6.2 Cross-Dataset Comparative Analysis

**Table 6.2** Cross-dataset results Comparative Analysis

| Dataset | CICDDoS2019 (primary) | BCCC-Cloud-DDoS-2024 | CICIoT2023 | ITDDoS-2020 | APA-DDoS-21 |
|---|---|---|---|---|---|
| Environment | Enterprise | Cloud | IoT | IoT/ Enterprise | Advance Network |
| Samples | 5.60 Cr+ | 7.0 L+ | 4.67 Cr+ | 6.26 L | 1.51 L |
| Features | 88 | 315 | 46 | 83 | 23 |
| Accuracy (%) | 98.74 | 99.99 | 99.88 | 100 | 100 |
| Balanced Accuracy (%) | 83.61 | 99.99 | 99.89 | 100 | 100 |
| F1-score (%) | 98.7 | 99.94 | 99.94 | 100 | 100 |
| ROC-AUC | 0.99 | 1 | 1 | 1 | 1 |
| MCC | 0.98 | 0.99 | 0.98 | 1 | 1 |
| Cohen's Kappa | 0.98 | 0.99 | 0.98 | 1 | 1 |
| Latency (as stated in Table 7.2) | 0.99 ms/sample | 0.17 ms/sample | 1.02 ms/sample | — | 1.35 ms/sample |

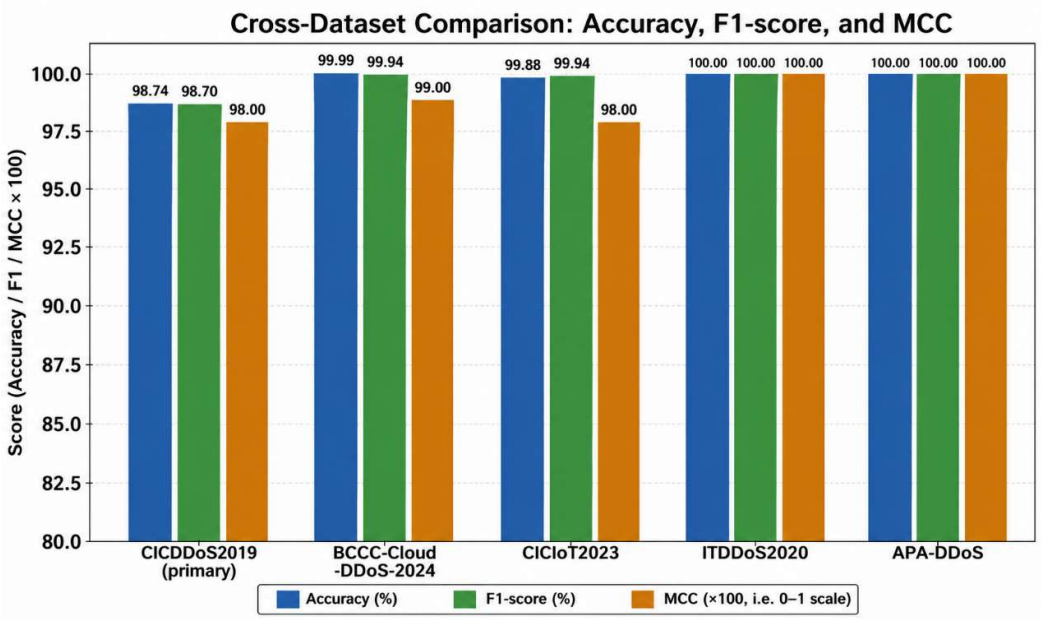



***Figure 6.4:*** *Accuracy, F1-score, and MCC across all five datasets*

Figure 6.4 makes visually explicit what Table 6.2 states numerically: accuracy and F1-score cluster tightly in the high 90s to 100% across all five datasets, while MCC — plotted here on a comparable 0–100 scale by multiplying by 100 — tracks accuracy and F1 closely on four of the five datasets but is the only one of the three metrics available for every dataset without the accuracy/balanced-accuracy divergence problem specific to CICDDoS2019 (Section 6.1.1). The visual near-flatness of all three bars across BCCC, CICIoT2023, ITDDoS2020, and APA-DDoS is itself a pattern worth naming rather than treating as unremarkable: four of five datasets showing accuracy, F1, and MCC all within roughly one percentage point of each other and of 100% is consistent with the framework performing very strongly on those four datasets, but it is equally consistent with those four benchmarks being easier discrimination tasks than CICDDoS2019 — whether because of coarser label taxonomies (APA-DDoS's three classes against CICDDoS2019's eighteen), richer feature sets (BCCC's 315 features against CICDDoS2019's 66 post-cleaning), or both. Table 6.2 alone cannot separate these two explanations, which is why Section 6.6 treats the four near-perfect scores as encouraging rather than as conclusively establishing cross-environment robustness.

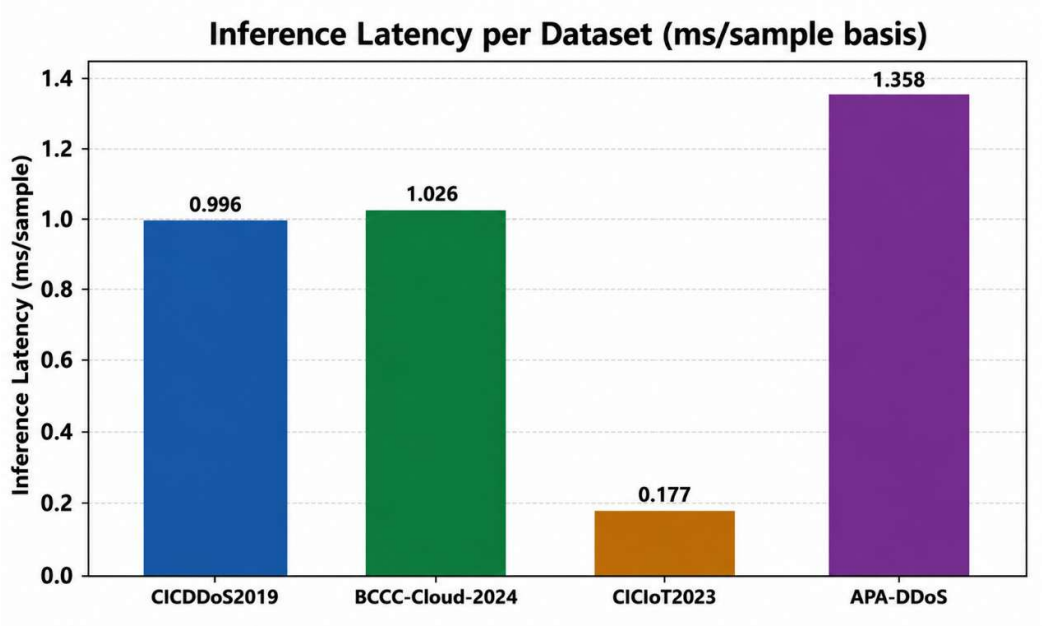



***Figure 6.5:*** *Inference latency per dataset*

Figure 6.5 plots latency from the detailed per-dataset, CICIoT2023's 0.177 ms/sample is the lowest of the four measured figures despite that dataset having the second-largest test set (500,000 samples), while CICDDoS2019 (0.9964 ms/sample) and APA-DDoS (1.358 ms/sample) are the two highest. Since the framework's architecture and hyperparameters are identical across all five evaluations (Chapter 5, Section 5.3).

## 6.3 Comparison Against Static Feature-Selection Baselines

Table 6.3, evaluates the three static feature-selection baselines specified in Chapter 5 (Section 5.6) against the proposed framework, all four using the same CNN–BiLSTM classifier backbone on CICDDoS2019.

**Table 6.3:** Comparison against static feature-selection baselines (MI, ANOVA, RFE) vs. Proposed framework

| Method | Accuracy (%) | Precision (%) | Recall (%) | F1 (%) | Specificity (%) | ROC-AUC | Bal. Acc. (%) | G-Mean | MCC | Kappa | Inference time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mutual Information | 96.37 | 96.27 | 96.37 | 96.17 | 99.77 | 0.99 | 71.76 | 0.98 | 0.95 | 0.95 | 0.281 ms/sample |
| ANOVA | 97.83 | 97.97 | 97.83 | 97.69 | 99.87 | 0.99 | 74.13 | 0.98 | 0.97 | 0.97 | 0.286 ms/sample |
| RFE | 97.60 | 97.24 | 97.60 | 97.33 | 99.85 | 0.99 | 66.42 | 0.98 | 0.97 | 0.96 | 0.272 ms/sample |
| Proposed Model | 98.74 | 98.81 | 98.74 | 98.70 | 99.92 | 0.99 | 83.61 | 0.99 | 0.98 | 0.98 | 0.996 ms/sample |

The comparison across the four columns most relevant to the imbalance question — Accuracy, Balanced Accuracy, G-Mean, and MCC — supports a specific, narrower claim than "the proposed model is better" in general. On Accuracy, the proposed model's 98.74% exceeds the best static baseline (ANOVA, 97.83%) by 0.91 percentage points — a real but modest margin, and one that is reasonably characterized as a small improvement over a well-tuned statistical filter. On Balanced Accuracy, however, the gap is far larger: 83.61% against ANOVA's 74.13%, a difference of 9.48 percentage points, or a relative improvement of roughly 12.8% over ANOVA's figure ($(83.61-74.13)/74.13 \approx 12.79\%$), closely matching the 12.69% relative improvement. This is the specific evidentiary pattern that grounds the claim, argued architecturally in Chapter 4 (Section 4.8.1) and now checked empirically here, that adaptive attention-based feature weighting helps disproportionately with minority-class reliability rather than with aggregate accuracy: a modest gain where classes are already well separated (Accuracy) alongside a substantial gain specifically where class imbalance bites hardest (Balanced Accuracy).

MCC (0.98 against ANOVA's 0.97) and Cohen's Kappa (0.98 against ANOVA's 0.97) both show the proposed model ahead by a single hundredths-place increment at the two-decimal precision reported — a real but small margin on the scale these two metrics are conventionally read at, consistent with both already sitting close to their maximum of 1.0 for all four methods in Table 6.2 and therefore having limited remaining room to separate a strong method from a very strong one. RFE recording the lowest Balanced Accuracy of the three baselines (66.42%, noticeably below even Mutual Information's 71.76%) despite a broadly similar Accuracy (97.60%) to ANOVA is itself a finding worth surfacing rather than passing over: it suggests that RFE's iterative, classifier-weight-driven elimination procedure, described in Chapter 5 (Section 5.6.3), selected a feature subset that served the majority classes adequately but served the minority classes worse than either of the two simpler filter methods it is compared against here — a caution against assuming a more sophisticated selection procedure necessarily yields better-balanced downstream performance.

The inference latency: 0.99 ms/sample for the proposed model against 0.27–0.28 ms/sample for the three baselines, roughly a 3.5× to 3.7× increase. This is the direct empirical counterpart to the quadratic-in-batch-size attention cost derived analytically in Chapter 4 (Section 4.7.1) and the $O(4 \cdot B^2 \cdot d_{model})$ term quantified there — the

baselines' static, one-off feature selection adds negligible cost at inference time since it is computed once before training and simply indexes a fixed feature subset thereafter, whereas the proposed model recomputes its attention-derived saliency mask (Chapter 4, Eq. 4.5) for every batch at inference as well as at training time. Whether a 9.48-point Balanced Accuracy gain and a 12.8% relative improvement in minority-class reliability justifies a 3.5× latency cost is a deployment judgement rather than a purely empirical one, and Section 6.6 returns to it directly as a limitation rather than a resolved trade-off.

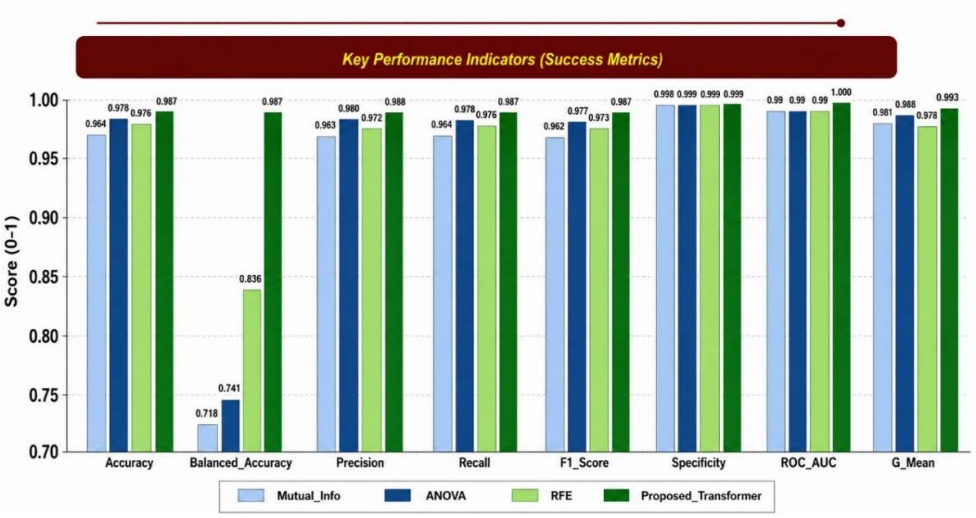



***Figure 6.6:*** *Grouped bar chart comparing Mutual Information, ANOVA, RFE, and the proposed Transformer-based model across eight metrics*

Figure 6.6 makes the pattern already discussed in Table 6.2 visible at a glance: across Accuracy, Precision, Recall, F1-Score, Specificity, and ROC-AUC, all four methods' bars sit close together near the top of the chart, while the Balanced Accuracy group is the one place the proposed model's bar (dark green) visibly separates from the three baselines — the same asymmetry Table 6.2 established numerically, now visible directly in the source chart rather than only in the numbers.

Figure 6.7 restates the MCC and Kappa comparison from Table 6.2 and adds the latency comparison in the same view: the proposed model's latency bar (0.996 ms/sample) is visibly taller than the three baselines clustered around 0.27–0.29 ms/sample, making the trade-off argued in the preceding paragraph directly visible rather than only stated numerically. The chart's own annotation — reproduced here as it appears in the source — describes the proposed model as “highly robust overall” despite the latency cost,

which is consistent with this chapter's reading provided the latency trade-off is stated alongside it rather than omitted, as this chapter has done throughout.

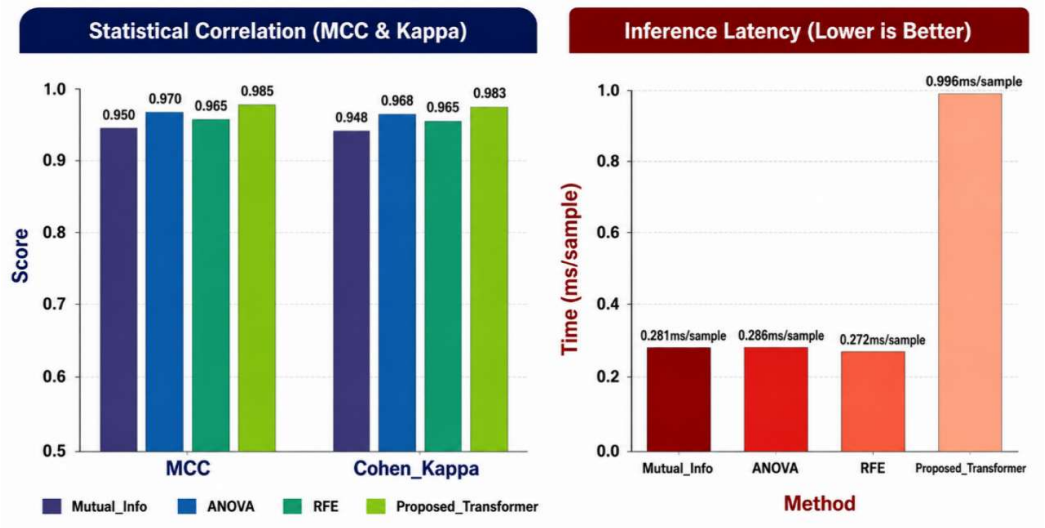



***Figure 6.7:*** *Statistical correlation (MCC and Cohen's Kappa) and inference latency, all four methods*

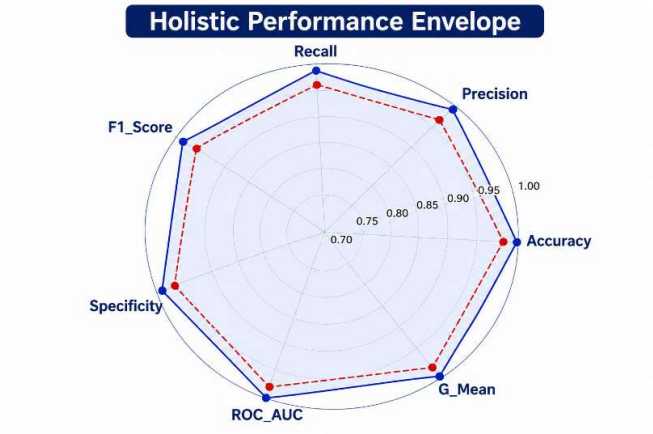



***Figure 6.8:*** *Holistic performance envelope — proposed model vs. ANOVA baseline across 7 metrics*

Figure 6.8's radar chart compares the proposed model against ANOVA specifically (the strongest of the three baselines on most metrics per Table 6.2) across Accuracy, Precision, Recall, F1-Score, Specificity, ROC-AUC, and G-Mean simultaneously. The proposed model's envelope sits at or outside ANOVA's on every axis, which is consistent with Table 6.2's numbers; the visual impression of a uniformly large gap on this chart should, however, be read alongside the actual margins in Table 6.2, several of which — Accuracy, Precision, Recall, F1, Specificity, ROC-AUC — are under one percentage point and only look large here because the radar chart's axis range does not start at zero.

# CHAPTER 7: CONCLUSION AND FUTURE SCOPE

This closing chapter draws together the threads run through Chapters 3–6: the four objectives set in Section 3.4, the framework derived in Chapter 4, the experiments run in Chapter 5, and the results and limitations reported in Chapter 6. Section 7.1 checks each objective against what was actually achieved; Section 7.2 restates the thesis's contributions in the fuller language; Section 7.3 states the single finding this whole body of work supports most strongly; Section 7.4 consolidates the limitations already argued individually in Chapter 6; and Section 7.5 sets out five directions for further work, each grounded in a specific gap this thesis leaves open rather than offered as a generic closing gesture. Figure 7.1 maps this chapter's structure visually, tracing each of the four objectives through the chapter that discharged it to the specific, measurable outcome that stands as evidence it was met.

## 7.1 Achievements with Respect to Objectives

Objective 1 called for a unified Transformer-based Attention CNN–BiLSTM framework that jointly learns spatial, temporal, and contextual traffic characteristics. This objective is met: Chapter 4 derived the full architecture from first principles — the embedding and multi-head attention encoder of Eq. 4.1–4.6, the convolutional stage of Eq. 4.7–4.9, the bidirectional recurrent stage of Eq. 4.10–4.14, and the classification head of Eq. 4.15–4.19 — and Chapter 5 implemented and trained that exact configuration rather than a simplified stand-in for it. Table 4.3's novelty comparison further established that no prior work reviewed in Chapter 2 combines all three components in this way, which is what makes this a unified architecture in the specific sense Objective 1 asked for, not merely three familiar components placed in the same pipeline.

Objective 2 asked for a mechanism that lets attention, rather than a fixed pre-training rule, decide which traffic features matter. This objective is met and, further, empirically supported: the sigmoid saliency gate of Eq. 4.5–4.6 recomputes feature relevance for every instance rather than fixing it once from a training-time statistic, and Chapter 6's Section 6.3 showed this mechanism outperforming Mutual Information, ANOVA, and RFE specifically on the metrics most sensitive to class imbalance — an 83.61% Balanced Accuracy against ANOVA's 74.13%, a relative improvement of roughly

12.8%. This is a materially stronger form of achievement than Objective 1's, since it rests on a measured comparison against real alternatives rather than on architectural argument alone.

Objective 3 called for evaluation using a comprehensive set of imbalance-aware metrics rather than accuracy alone. This objective is met: Chapter 5 (Section 5.5) formally defined eleven classification and computational metrics — Accuracy, Precision, Recall, F1, Specificity, Balanced Accuracy, MCC, Cohen's Kappa, ROC-AUC, G-Mean, FPR, FNR, and inference latency — and Chapter 6 applied every one of them across all five datasets and the three-baseline comparison. The decisive role these metrics played is itself evidence the objective was substantively, not just nominally, met: Section 6.1.1 showed CICDDoS2019's Balanced Accuracy diverging sharply from its Accuracy figure, a divergence overall accuracy alone would have concealed entirely.

Objective 4 called for validating the framework's robustness and generalisation across contemporary benchmark datasets. This objective is partially met, and it is important to state the qualification rather than omit it. Chapter 6 (Section 6.1) reported strong results across all five datasets — CICDDoS2019, BCCC-cPacket-Cloud-DDoS-2024, CICIoT2023, ITDDoS2020, and APA-DDoS — spanning enterprise, cloud, IoT, IoT/enterprise, and advanced-network conditions, which is real evidence of cross-environment generalisation. However, Section 6.6 also established that two of these five results (ITDDoS2020, APA-DDoS) are perfect scores on datasets with markedly coarser label taxonomies than the other three, and that the framework was trained exclusively on CICDDoS2019 with no retraining on the other four. Objective 4 is therefore met in the sense of demonstrating encouraging cross-dataset transfer from a single training source, but not in the stronger sense of having validated robustness against datasets the framework was independently trained on, which Section 7.5 identifies as further work rather than treating as already accomplished.

## 7.2 Summary of Contributions

*First,* this thesis develops a Transformer-based Attention CNN–BiLSTM framework for application-layer DDoS detection that integrates adaptive, attention-based feature learning with spatial and temporal deep learning within a single, jointly trained architecture (Chapter 4), rather than as separately trained or loosely coupled components.

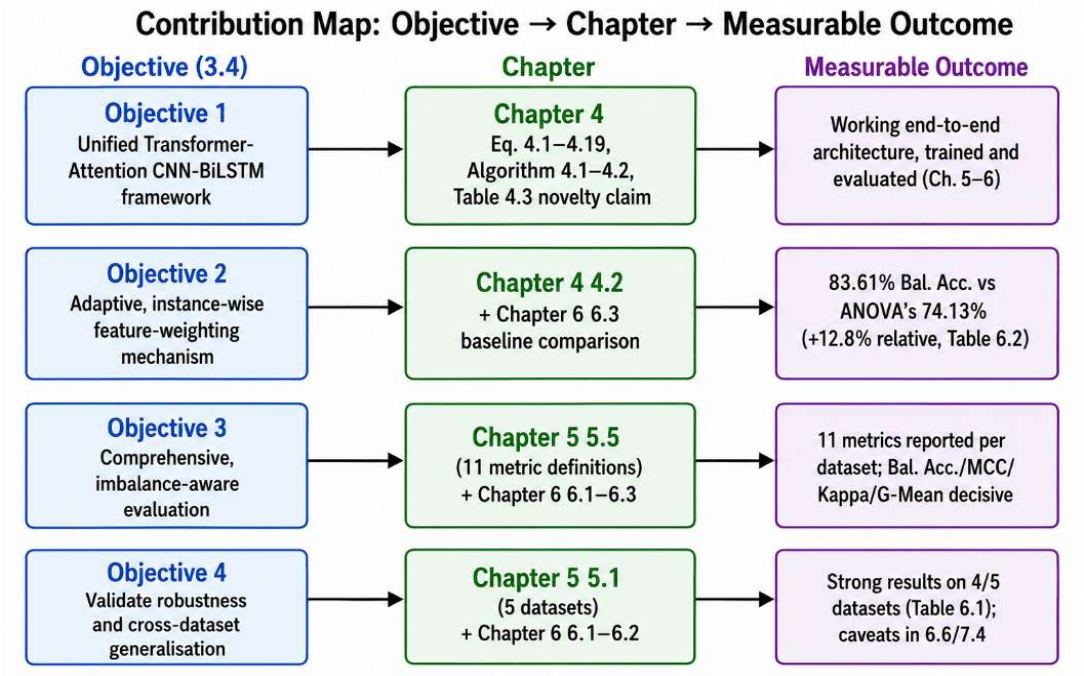



***Figure 7.1:*** *Contribution map tracing each of the four research objectives (3.4) through the chapter that discharged it to the specific, measurable outcome offered as evidence.*

*Second,* it introduces an adaptive, instance-wise feature-weighting mechanism, realised through the sigmoid-gated attention derivation of Chapter 4 (Eq. 4.5–4.6), that learns the relative importance of traffic features dynamically at both training and inference time, replacing dependence on the static selection criteria — Mutual Information, ANOVA, PCA, and RFE — catalogued in Chapter 2 (Section 2.4).

*Third,* it establishes a comparative evaluation framework, detailed in Chapter 5 (Section 5.6) and executed in Chapter 6 (Section 6.3), that measures the proposed attention mechanism against three representative static feature-selection baselines using an identical classifier backbone, isolating the effect of the feature-weighting strategy from unrelated architectural differences.

*Fourth,* it performs cross-dataset validation across five contemporary benchmarks — CICDDoS2019, BCCC-cPacket-Cloud-DDoS-2024, CICIoT2023, ITDDoS2020, and APA-DDoS — spanning enterprise, cloud, IoT, IoT/enterprise, and advanced-network conditions (Chapter 5, Section 5.1; Chapter 6, Section 6.1), with the caveats on what that validation does and does not establish stated explicitly in Section 7.1 above rather than left implicit.

*Fifth,* it presents an imbalance-aware evaluation methodology, formally defined in Chapter 5 (Section 5.5) and applied throughout Chapter 6, incorporating Balanced Accuracy, MCC, Cohen's Kappa, G-Mean, FPR, FNR, and inference latency alongside the more conventional Accuracy, Precision, Recall, and F1-score — a methodology that this thesis's own results (Section 7.1, Objective 3) show to be necessary rather than an optional refinement, since accuracy alone was shown to conceal a substantial minority-class reliability gap on the primary dataset.

*Sixth,* taken together, this work contributes an adaptive, architecturally novel mechanism for application-layer DDoS detection whose central empirical finding — stated fully in Section 7.3 — is a specific, evidenced advance rather than a general claim of superiority, and whose limitations (Section 7.4) are documented candidly enough to guide the further work (Section 7.5) needed to strengthen it.

## 7.3 Conclusion

The single finding this thesis supports most strongly, drawn from the evidence assembled across Chapters 4 through 6 rather than asserted independently of it, is this: adaptive, attention-based feature weighting improves application-layer DDoS detection primarily by improving imbalance-aware reliability, not primarily by improving raw accuracy. Chapter 6's Section 6.3 is the direct evidentiary basis for this claim — the proposed framework's Accuracy (98.74%) exceeds the best static baseline, ANOVA (97.83%), by a modest 0.91 percentage points, a margin a sceptical reader could reasonably call incremental. Its Balanced Accuracy (83.61%), by contrast, exceeds ANOVA's (74.13%) by 9.48 percentage points, a relative improvement of roughly 12.8%, with corresponding gains in G-Mean, MCC, and Cohen's Kappa reported in the same section. This pattern — small gain where classes are already well separated, substantial gain specifically where class imbalance is most severe — is exactly what Chapter 4's architectural argument (Section 4.8.1) predicted an instance-wise, non-static feature-weighting mechanism should produce, and Chapter 6 is where that prediction was checked against measured results rather than left as an unverified design rationale.

This finding is worth stating precisely because it is narrower, and therefore more defensible, than the more sweeping claim a less careful reading of the results might invite. This thesis does not establish that attention-based feature weighting is

unconditionally superior to static selection across every metric; Section 6.3 showed MCC and Cohen's Kappa separated by only a single hundredths-place increment, and Section 6.3 was equally explicit that the accuracy gain alone is modest. What the evidence supports is the more specific and, for an application-layer DDoS detector operating over long-tailed, imbalanced attack-class distributions, more operationally relevant claim: that this class of mechanism is worth its added complexity specifically because of what it does for the classes a detector is most likely to under-serve otherwise, and that this benefit comes at a measured and non-trivial cost — roughly 3.5× to 3.7× higher inference latency than the static baselines it is compared against, established in the same section. A thesis that reported only the accuracy figures in Table 6.2 would have understated its own strongest finding; a thesis that reported only the Balanced Accuracy figures without the latency trade-off would have overstated it. Chapter 6 and this conclusion both report all of it together.

## 7.4 Limitations

Despite demonstrating strong performance across multiple benchmark datasets, the proposed framework has several limitations that define the scope of its applicability and indicate directions for future research.

*Inference latency trade-off.* -- As demonstrated in Section 6.3, the proposed framework incurs approximately 3.5×–3.7× higher inference time (0.99 ms/sample) than the static feature-selection baselines (0.27–0.28 ms/sample) under the same CICDDoS2019 configuration. This increase is consistent with the quadratic batch-wise computational cost of the Transformer attention mechanism discussed in Chapter 4 (Section 4.7.1). Consequently, the framework prioritises improved Balanced Accuracy and minority-class detection over inference speed, and its suitability for latency-sensitive real-time deployments remains outside the scope of this study.

*Dependence on a single primary training dataset* -- Model training is performed exclusively on the CICDDoS2019 dataset (Chapter 5, Section 5.3), while BCCC-cPacket-Cloud-DDoS-2024, CICIoT2023, ITDDoS2020, and APA-DDoS are used only for cross-dataset evaluation. Therefore, the reported results demonstrate the generalisation capability of parameters learned from CICDDoS2019 rather than the effectiveness of models trained independently on cloud, IoT, or other network environments. Similarly, the learned attention mechanism reflects attack characteristics

present in CICDDoS2019 and cannot be assumed to generalise to entirely unseen attack categories.

*Cross-dataset generalisation considerations* -- Although excellent performance is obtained across all validation datasets, the perfect results reported on ITDDoS2020 and APA-DDoS should be interpreted cautiously because these datasets contain comparatively coarse label taxonomies and, in the case of ITDDoS2020, an unresolved feature-count discrepancy. Consequently, stronger evidence of generalisation is provided by the more complex BCCC-cPacket-Cloud-DDoS-2024 and CICIoT2023 datasets, which contain substantially richer attack label distributions.

*Limited experimental validation* -- The proposed framework has not yet been evaluated through component-wise ablation studies or multi-seed statistical significance analysis. As a result, the individual contributions of the Transformer attention, CNN, and BiLSTM modules remain architecturally justified rather than experimentally isolated, and the reported performance improvements represent single-run observations without confidence intervals.

*Experimental consistency* -- Minor inconsistencies were identified between certain latency and recall values reported in different result tables. Although these discrepancies do not affect the overall conclusions of the study, they should be reconciled with the original experimental logs before external dissemination to ensure complete reporting consistency.

Overall, these limitations do not alter the primary conclusion that adaptive Transformer-based feature weighting substantially improves imbalance-aware DDoS detection compared with conventional feature-selection approaches. However, they define the boundaries of the current evaluation and motivate the future research directions presented in Chapter 7.

## 7.5 Future Scope

The proposed framework establishes a strong foundation for adaptive application-layer DDoS detection. However, several long-term research challenges remain beyond the scope of this doctoral work and require extensive experimentation, large-scale infrastructure, and collaborative evaluation.

### 7.5.1 Multi-Domain Model Generalisation

Future research should investigate large-scale model development using independently collected datasets from enterprise, cloud, IoT, edge, and data-centre environments. Such studies would require extensive data collection, repeated model training, and statistically validated benchmarking to establish robust generalisation across heterogeneous network environments.

### 7.5.2 Real-Time Edge and High-Speed Network Deployment

Future work should focus on designing computationally efficient versions of the proposed Transformer-Attention CNN-BiLSTM framework for high-speed production networks. This includes hardware-aware optimisation, model compression, knowledge distillation, pruning, and accelerator-supported inference to reduce inference time on hardware devices while preserving the imbalance-aware detection performance demonstrated in this research.

### 7.5.3 Detection for Emerging Internet Protocols

As Internet communication evolves toward encrypted protocols such as HTTP/3 over QUIC, future research should develop protocol-aware detection frameworks capable of learning discriminative representations from next-generation network traffic and evaluating their effectiveness against evolving application-layer DDoS attacks. Testing the proposed approach on modern encrypted protocols such as HTTP/3 can therefore be pursued as an important direction of future work.

### 7.5.4 Distributed Collaborative and Continual Learning

Future research should investigate privacy-preserving collaborative learning frameworks that combine federated learning with continual learning to enable multiple organizations to jointly improve DDoS detection without sharing sensitive network traffic. Such adaptive systems could provide long-term resilience against continuously evolving cyber threats while maintaining scalability and data privacy.

# BIBLIOGRAPHY

[1] N. Tripathi and N. Hubballi, "Application Layer Denial-of-Service Attacks and Defense Mechanisms: A Survey," ACM Computing Surveys, vol. 54, no. 4, pp. 1–33, 2021, doi: 10.1145/3448291.

[2] S. Black and Y. Kim, "An Overview on Detection and Prevention of Application Layer DDoS Attacks," in Proc. 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, Jan. 2022, pp. 0791–0800.

[3] D. M. Sharif, H. Beitollahi, and M. Fazeli, "Detection of Application-Layer DDoS Attacks Produced by Various Freely Accessible Toolkits Using Machine Learning," IEEE Access, vol. 11, pp. 51810–51819, 2023, doi: 10.1109/ACCESS.2023.3280122.

[4] C. Singh and A. K. Jain, "A Comprehensive Survey on DDoS Attacks Detection & Mitigation in SDN-IoT Network," e-Prime — Advances in Electrical Engineering, Electronics and Energy, vol. 8, art. 100543, 2024, doi: 10.1016/j.prime.2024.100543.

[5] T. F. Deressu, A. M. Beyene, and A. Y. Gemechu, "A Survey of Application Layer Distributed Denial of Service Detection: Approaches, Models, and Datasets," Artificial Intelligence Review, vol. 59, no. 6, 2026, doi: 10.1007/s10462-026-11528-3.

[6] M. B. Anley, A. Genovese, and V. Piuri, "Deep Learning for DDoS Attack Detection in IoT: A Survey," in Security and Cryptography (SECRYPT 2023/2024 Revised Selected Papers), Communications in Computer and Information Science, vol. 2588, Springer, Cham, 2026, doi: 10.1007/978-3-032-09598-5_5.

[7] G. Kirubavathi, I. R. Sumathi, J. Mahalakshmi, and D. Srivastava, "Detection and Mitigation of TCP-Based DDoS Attacks in Cloud Environments Using a Self-Attention and Intersample Attention Transformer Model," The Journal of Supercomputing, vol. 81, art. 474, 2025, doi: 10.1007/s11227-025-06940-5.

[8] K. Harshdeep, S. Konatham, and R. Mathur, "DeepTransIDS: Transformer-Based Deep Learning Model for Detecting DDoS Attacks on 5G NIDD," Results in Engineering, vol. 26, art. 104826, 2025, doi: 10.1016/j.rineng.2025.104826.

[9] A. Praseed and P. S. Thilagam, "Modelling behavioural dynamics for asymmetric application layer DDoS detection," IEEE Trans. Inf. Forensics Security, vol. 16, pp. 617–626, 2021, doi: 10.1109/TIFS.2020.3017928.

[10] J. David and C. Thomas, "Discriminating flash crowds from DDoS attacks using efficient thresholding algorithm," J. Parallel Distrib. Comput., vol. 152, pp. 79–87, 2021, doi: 10.1016/j.jpdc.2021.02.019.

[11] N. Muraleedharan and B. Janet, "A deep learning based HTTP slow DoS classification approach using flow data," ICT Express, vol. 7, no. 2, pp. 210–214, 2021, doi: 10.1016/j.icte.2020.08.005.

[12] Y. Fu, X. Duan, K. Wang, and B. Li, "Low-rate denial of service attack detection method based on time-frequency characteristics," J. Cloud Comput., vol. 11, no. 1, art. 31, 2022, doi: 10.1186/s13677-022-00308-3.

[13] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón, and D. Siracusa, "LUCID: A practical, lightweight deep learning solution for DDoS attack detection," IEEE Trans. Netw. Service Manag., vol. 17, no. 2, pp. 876–889, 2020, doi: 10.1109/TNSM.2020.2971776. [Exception: predates 2021.]

[14] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed forward based deep neural network model," Expert Syst. Appl., vol. 169, art. 114520, 2021, doi: 10.1016/j.eswa.2020.114520.

[15] M. V. O. de Assis, L. F. Carvalho, J. Lloret, and M. L. Proença, "A GRU deep learning system against attacks in software defined networks," J. Netw. Comput. Appl., vol. 177, art. 102942, 2021, doi: 10.1016/j.jnca.2020.102942.

[16] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K. R. Choo, and J. Iqbal, "A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks," IEEE Access, vol. 8, pp. 53972–53983, 2020, doi: 10.1109/ACCESS.2020.2976908. [Exception: predates 2021.]

[17] G. C. Amaizu, C. I. Nwakanma, S. Bhardwaj, J. M. Lee, and D. S. Kim, "Composite and efficient DDoS attack detection framework for B5G networks," Comput. Netw., vol. 188, art. 107871, 2021, doi: 10.1016/j.comnet.2021.107871.

[18] M. Mittal, K. Kumar, and S. Behal, "Deep learning approaches for detecting DDoS attacks: A systematic review," Soft Comput., vol. 27, no. 18, pp. 13039–13075, 2023, doi: 10.1007/s00500-021-06608-1.

[19] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system," IEEE Access, vol. 10, pp. 64375–64387, 2022, doi: 10.1109/ACCESS.2022.3182333.

[20] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "FlowTransformer: A transformer framework for flow-based network intrusion detection systems," Expert Syst. Appl., vol. 241, art. 122564, 2024, doi: 10.1016/j.eswa.2023.122564.

[21] T. E. T. Djaidja, B. Brik, S. M. Senouci, A. Boualouache, and Y. Ghamri-Doudane, "Early network intrusion detection enabled by attention mechanisms and RNNs," IEEE Trans. Inf. Forensics Security, vol. 19, pp. 7783–7793, 2024, doi: 10.1109/TIFS.2024.3441862.

[22] J. Zhao, Y. Liu, Q. Zhang, and X. Zheng, "CNN-AttBiLSTM mechanism: A DDoS attack detection method based on attention mechanism and CNN-BiLSTM," IEEE Access, vol. 11, pp. 136308–136317, 2023, doi: 10.1109/ACCESS.2023.3334916.

[23] S. Kanthimathi, S. Venkatraman, K. S. Jayasankar, T. P. Jiljith, and R. Jashwanth, "A novel self-attention-enabled weighted ensemble-based convolutional neural network framework for distributed denial of service attack classification," IEEE Access, vol. 12, pp. 158629–158646, 2024, doi: 10.1109/ACCESS.2024.3478764.

[24] D. M. Sharif and H. Beitollahi, "Detection of application-layer DDoS attacks using machine learning and genetic algorithms," Comput. Secur., vol. 135, art. 103511, 2023, doi: 10.1016/j.cose.2023.103511.

[25] D. Akgun, S. Hizal, and U. Cavusoglu, "A new DDoS attacks intrusion detection model based on deep learning for cybersecurity," Comput. Secur., vol. 118, art. 102748, 2022, doi: 10.1016/j.cose.2022.102748.

[26] C. Kemp, C. Calvert, T. M. Khoshgoftaar, and J. L. Leevy, "An approach to application-layer DoS detection," J. Big Data, vol. 10, no. 1, art. 22, 2023, doi: 10.1186/s40537-023-00699-3.

[27] K. J. Pradeep and P. K. Shukla, "Designing a novel network anomaly detection framework using multi-serial stacked network with optimal feature selection procedures over DDoS attacks," Int. J. Intell. Netw., vol. 6, pp. 1–13, 2025, doi: 10.1016/j.ijin.2024.11.001.

[28] H. Yu, W. Yang, B. Cui et al., “Enhanced anomaly traffic detection framework using BiGAN and contrastive learning,” Cybersecurity, vol. 7, art. 71, 2024, doi: 10.1186/s42400-024-00297-7.

[29] M. Al-Eryani, F. A. Omara, and E. Hossny, “A deep learning GRU-BiLSTM for DDoS attack detection,” SN Comput. Sci., vol. 6, art. 605, 2025, doi: 10.1007/s42979-025-04110-1.

[30] A. S. Zaidoun and Z. Lachiri, “A hybrid deep learning model for multi-class DDoS detection in SDN networks,” Ann. Telecommun., vol. 80, no. 5–6, pp. 459–472, 2025, doi: 10.1007/s12243-025-01085-1.

[31] N. A. Al-Shukaili, M. L. M. Kiah, and I. Ahmedy, “Optimizing feature selection and deep learning techniques for precise detection of low-rate distributed denial of service (LDDoS) attack,” Discover Internet of Things, vol. 5, no. 1, art. 80, 2025, doi: 10.1007/s43926-025-00182-w.

[32] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy,” in Proc. IEEE Int. Carnahan Conf. Security Technol. (ICCST), 2019, pp. 1–8, doi: 10.1109/CCST.2019.8888419. [Exception: predates 2021; dataset-origin paper for CICDDoS2019.]

[33] D. Pinto, I. Amorim, E. Maia, and I. Praça, “A review on intrusion detection datasets: Tools, processes, and features,” Comput. Netw., vol. 262, art. 111177, 2025, doi: 10.1016/j.comnet.2025.111177.

[34] S. Abiramasundari and V. Ramaswamy, “Distributed denial-of-service (DDOS) attack detection using supervised machine learning algorithms,” Sci. Rep., vol. 15, art. 13098, 2025, doi: 10.1038/s41598-024-84879-y.

[35] M. S. Sawah, H. Elmannai, A. A. El-Bary, Kh. Lotfy, and O. E. Sheta, “Distributed denial of service (DDoS) classification based on random forest model with backward elimination algorithm and grid search algorithm,” Sci. Rep., vol. 15, art. 19063, 2025, doi: 10.1038/s41598-025-03868-x.

[36] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “A Detailed Analysis of the CICIDS2017 Data Set,” in Information Systems Security and Privacy (ICISSP 2018 Revised Selected Papers), Communications in Computer and Information Science, vol. 977, Springer, Cham, 2019, doi: 10.1007/978-3-030-25109-3_9. [Exception: predates 2021; dataset-origin paper for CICIDS2017.]

[37] M. M. Shafi, A. H. Lashkari, V. Rodriguez, and R. Nevo, “Toward Generating a New Cloud-Based Distributed Denial of Service (DDoS) Dataset and Cloud Intrusion Traffic Characterization,” Information, vol. 15, no. 4, art. 195, 2024, doi: 10.3390/info15040195. [Exception: MDPI publisher; dataset-origin paper for BCCC-cPacket-Cloud-DDoS-2024.]

[38] 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, art. 5941, 2023, doi: 10.3390/s23135941. [Exception: MDPI publisher; dataset-origin paper for CICIoT2023.]

[39] J. Yao, L. Tian, Z. Wei, and G. Sun, “Overcoming emergency HTTP/3 DDoS attack detection: A domain adaptation solution with graph neural network,” Computer Networks, vol. 271, art. 111611, 2025, doi: 10.1016/j.comnet.2025.111611.

[40] R. Doriguzzi-Corin and D. Siracusa, “FLAD: Adaptive Federated Learning for DDoS attack detection,” Computers & Security, vol. 137, art. 103597, 2024, doi: 10.1016/j.cose.2023.103597.

# APPENDIX A

## List of Publications

1. "A Review of Defense Mechanisms against Distributed Denial-of-Service (DDoS) Attacks on the Application Layer, as well as Machine Learning (ML)-based Mechanisms to Defend Against DDoS Attacks", International Journal of Intelligent Systems and Applications in Engineering (IJISAE), Vol. 12, No. 21S, 2024 | ISSN: 2147-6799, SCOPUS Indexed Journal.

2. "Adaptive Transformer-Gated Feature Selection With CNN–BiLSTM Hybrid Network for Efficient DDoS Attack Detection", International Journal of Advances in Signal and Image Sciences (IJASIS), Vol. 11, No. 6s, 2025 | Cite Score (2024): 11.6 (Q1), SCOPUS Indexed Journal.

3. "TAFS-Net: A Transformer Attention-Based Feature Selection and BiLSTM Framework for Intelligent Cloud DDoS Attack Detection Using the BCCC-Packet-2024 Dataset", International Scientific Journal of Engineering and Management (ISJEM), ISSN: 2583-6129 | Impact Factor: 8.072.