UTILIZING MACHINE LEARNING TO DETECT TOR TRAFFIC: A REALISTIC DATASET AND A PRELIMINARY ANALYSIS

by

Md Rafiqul Islam Sadik

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2025

To my beloved parents, whose unconditional love, sacrifices, and prayers have guided my path and strengthened my spirit, and to my wife, whose constant support, patience, and encouragement have been my greatest motivation. Their faith in me has made this journey possible. This thesis is dedicated to them with heartfelt gratitude and endless affection.

Table of Contents

List of Tables

List of Figures

Abstract

With the increasing use of anonymization technologies such as the Tor network, the ability to accurately differentiate Tor traffic from conventional Internet traffic has become an important challenge for network analysis and security monitoring.This thesis presents a fully controlled and reproducible framework for generating realistic Tor and Non-Tor traffic datasets to support the evaluation of encrypted traffic detection techniques. The framework integrates a Debian workstation, a Whonix gateway for Tor routing, and a noise-free AWS-based web server, combined with Selenium-driven automation to execute identical user activities over both Tor-based and Non-Tor-based network paths.Using this environment, a comprehensive dataset was generated across six application categories: web browsing, video streaming, file transfer, instant messaging, voice over IP, and video conferencing. A set of six machine-learning models-Decision Tree, Random Forest, XGBoost, Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) was evaluated on the generated dataset. Experimental results demonstrate that traditional tree-based models, particularly Random Forest and XGBoost, consistently outperform deep-learning approaches, achieving high detection accuracy in distinguishing Tor from Non-Tor network flows across all traffic types.These findings highlight both the effectiveness of classical machinelearning approaches and the importance of realistic dataset generation in advancing encrypted traffic classification research.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Dr. Qiang Ye and my colleague Suchith Sridhar Khajjayam, for their continuous guidance, encouragement, and insightful feedback throughout this research. Their expertise and patience have been invaluable to my academic and personal growth.

I am sincerely thankful to the Faculty of Computer Science at Dalhousie University for providing a supportive learning and research environment, as well as the computational resources needed to complete this work. I also extend my appreciation to my colleagues and friends in the lab, whose discussions, suggestions, and friendship made this journey more enjoyable and motivating.

I owe a special debt of gratitude to my parents for their endless love, sacrifices, and prayers, which have been a constant source of strength throughout my life. I am equally thankful to my wife for her unwavering support, patience, and encouragement during the challenging phases of this thesis. Without their belief in me, this work would not have been possible.

Chapter 1

Introduction

The rapid growth of encrypted communication on the Internet has significantly improved user privacy, data confidentiality, and protection against surveillance. At the same time, the widespread adoption of encryption has introduced new challenges for network monitoring [7] , intrusion detection, digital forensics, and cybercrime investigation. A prominent example of such encrypted and anonymized communication is the Tor network, which is widely used to conceal user identities and traffic origins. While Tor plays a legitimate role in safeguarding freedom of expression, circumventing censorship, and enabling secure access to information, it can also be misused for malicious activities such as cybercrime, botnet command-and-control, ransomware operations, and access to darknet marketplaces. Distinguishing Tor traffic from conventional Internet traffic has therefore become a crucial requirement for modern cybersecurity systems.

1.1 Background

Tor (The Onion Router) [45] is an anonymization network designed to provide strong privacy guarantees through multi-hop routing and layered encryption. Each user request is relayed through a chain of three Tor nodes—entry, middle, and exit—ensuring that no single entity knows both the sender and the receiver. By design, Tor obscures IP addresses, packet origins, and traffic patterns, making passive monitoring and traffic classification considerably more difficult.

In recent years, the demand for encrypted and anonymized communication has surged. The introduction of new transport protocols such as QUIC, the rise of video conferencing platforms, and the proliferation of cloud-based services have significantly changed traffic behavior on the Internet. These developments complicate the task of distinguishing Tor traffic from regular encrypted traffic, especially when both share similar protocol behaviors and flow-level characteristics. As a result, effective Tor

traffic detection must rely on robust statistical, temporal, and behavioral features rather than payload inspection, which is no longer feasible in an era dominated by TLS and end-to-end encryption.

In addition to these broader trends, the growth of privacy-enhancing technologies has reshaped the landscape of modern network traffic. The increasing deployment of TLS 1.3, encrypted DNS protocols (DoH/DoT), and traffic-obfuscation tools has substantially reduced the visibility available to network operators and intrusion detection systems. As a result, traditional inspection methods such as deep packet inspection (DPI) have become far less effective, forcing a shift toward behavioral and statistical analysis. These developments make the classification of anonymized traffic—including Tor—both more relevant and more challenging than in earlier decades.

At the same time, the Tor ecosystem itself has evolved considerably. Recent updates to Tor's congestion-control algorithms, guard-node selection strategies, and circuit management policies have introduced new traffic patterns that differ from those documented in earlier research. The rise of mobile Tor usage, the increasing availability of pluggable transports, and the diversification of Tor user activities also contribute to greater variability in Tor traffic behavior. Consequently, many prior datasets and classification techniques no longer reflect the characteristics of modern Tor communication or the applications commonly used today.

These trends collectively highlight the urgency of developing up-to-date, controlled, and reproducible datasets that capture realistic Tor and Non-Tor communication patterns. Without such datasets, it becomes difficult to assess the accuracy of existing machine-learning models or to design robust classifiers that continue to operate effectively under evolving network conditions. Furthermore, as encrypted traffic continues to dominate global Internet usage, understanding the boundary between benign anonymized communication and potentially malicious activity becomes a critical research problem for both network security practitioners and privacy-preserving technologies. This thesis addresses this gap by offering a comprehensive and modern analysis of Tor traffic classification based on a rigorously controlled dataset and a systematic evaluation of traditional and deep-learning approaches.

1.2 Motivation

Although several Tor traffic datasets have been published in the past decade, such as ISCXTor2016 [30] and CIC-Darknet2020, [1] they are becoming increasingly outdated for modern research needs. These datasets were created many years ago, before the widespread adoption of contemporary applications such as Microsoft Teams, modern YouTube streaming formats, and other high-bandwidth, interactive services.

Prior datasets exhibit notable limitations:

- **Lack of comparability:** Tor and Non-Tor traffic captures often correspond to different browsing activities, file sizes, or video content. Such inconsistencies allow machine learning models to exploit application-level differences rather than intrinsic characteristics of Tor.

- **Presence of noise:** Real-world applications generate substantial background traffic such as advertisements, prefetching, analytics, recommendation engines, and auto-play content. These unpredictable factors make it difficult to isolate pure user activities.

- **Absence of modern protocols:** New protocols such as QUIC, adaptive video streaming formats, and updated transport-layer behavior are not adequately represented in older datasets.

- **Limited application diversity:** Many existing datasets focus heavily on web browsing and neglect real-time applications such as VoIP and video conferencing, which are prevalent today.

Another common but fundamentally unreliable approach to identifying Tor traffic is to compare packet destinations against the public list of Tor relays published by the Tor Project. Although this method appears straightforward, it does not work well in practice and fails under several realistic conditions.

1. **Tor over VPN hides all relay connections.** Many users connect to Tor through a VPN to add an additional privacy layer. In such cases, the client connects only to the VPN server, and all Tor relay interactions occur inside the

encrypted VPN tunnel. As a result, the network observer cannot see Tor relay IP addresses, making IP-based detection ineffective.

2. **Tor bridges are intentionally not included in public relay lists.** Bridges are unlisted Tor nodes designed to help users in censored regions bypass IP-based blocking. Since they are not published in any public directory, any detection method based on relay lists completely fails to identify bridge-based Tor traffic.

3. **Pluggable transports disguise Tor connections.** Many Tor users rely on pluggable transports such as `obfs4`, `meek`, and `snowflake`, which reshape and obfuscate traffic patterns. These transports modify packet characteristics and hide Tor handshakes, making the connection appear as ordinary HTTPS or random-looking encrypted traffic. Public relay IP lists provide no visibility into these obfuscated flows.

4. **Tor relays frequently change IP addresses.** The Tor network is dynamic; relays go online and offline regularly, and their IP addresses change over time. Public relay lists become outdated quickly, leading to false negatives and inconsistent detection results.

For these reasons, identifying Tor traffic using public relay IP lists is incomplete, unreliable, and easily circumvented. This limitation further motivates the need for machine-learning-based approaches that rely on flow-level statistical features rather than fixed relay addresses.

These limitations highlight the need for a new, up-to-date, and carefully controlled Tor traffic dataset that provides fair comparability between Tor and Non-Tor flows across a wide range of applications.

1.3 Problem Statement

Due to evolving Internet protocols, increased traffic complexity, and the limitations of existing datasets, there is a pressing need to determine whether machine learning and deep learning approaches can still accurately differentiate Tor and Non-Tor traffic

under realistic and controlled conditions. The central problem addressed in this thesis is:

> *Can Tor and Non-Tor traffic be reliably distinguished using machine learning techniques when the traffic is generated under fully controlled, comparable, and noise-free conditions across modern application scenarios?*

To answer this question, a new dataset must be generated that eliminates inconsistencies between Tor and Non-Tor traffic, provides accurate comparability, and reflects real-world usage patterns across multiple application domains.

This thesis aims to answer the following research questions:

1. **Dataset Generation:** How can a fully controlled framework be designed to generate realistic, comparable Tor and Non-Tor traffic across diverse applications?
2. **Feature Extraction:** Are flow-level statistical features sufficient for representing Tor and Non-Tor network behavior in a machine learning setting?
3. **Model Performance:** How effectively can traditional machine learning models and deep learning models classify Tor vs. Non-Tor traffic across different traffic categories?
4. **Application Dependence:** How does classification performance vary across applications such as web browsing, video streaming, file transfers, instant messaging, VoIP, and video conferencing?
5. **Impact of Controlled Environments:** Does removing noise and enforcing identical activity sequences improve or reduce model performance?

1.4 Contributions

The key contributions of this thesis are summarized as follows:

- **A Fully Controlled Framework:** A VirtualBox-based architecture using Debian and Whonix is developed to isolate and capture Tor and Non-Tor traffic along two separate network interfaces.

- **Noise-Free Server Configuration:** A custom AWS-hosted server is deployed to eliminate background noise during web browsing, video streaming, and file transfers, ensuring clean and reproducible traffic traces.
- **Comprehensive Dataset:** Six different application categories are generated, including web browsing, video streaming, file transfer, instant messaging, VoIP, and video conferencing, producing an up-to-date and diverse Tor dataset. [42]
- **Flow-Level Feature Extraction:** A Scapy-based pipeline is implemented to extract 48 statistical features from raw packet captures, enabling machine learning analysis.
- **Machine Learning Evaluation:** Six models—Decision Tree, Random Forest, XGBoost, MLP, CNN, and RNN—are trained and evaluated across all application categories.
- **Extensive Comparative Analysis:** Classification performance is analyzed across different resolutions, file sizes, traffic types, and application scenarios, offering deep insights into Tor detection behavior.

1.5 Thesis Organization

The remainder of this thesis is organized as follows:

- **Chapter 2** presents a comprehensive literature review covering the Tor network and related anonymity systems, machine learning and deep learning techniques for encrypted traffic classification, censorship resistance and pluggable transports, as well as publicly available Tor and darknet datasets. It also identifies key research gaps that motivate the design of a new, realistic Tor vs. Non-Tor dataset.
- **Chapter 3** describes the methodology used to generate the proposed dataset. It details the VirtualBox-based testbed architecture, including the Debian Workstation and Whonix Gateway, the UDP-blocking configuration, the design of the noise-free AWS server, the traffic generation procedures for six application

categories, and the custom flow-based feature extraction pipeline. Ethical and privacy considerations related to data collection are also discussed.

- **Chapter 4** presents the machine-learning based Tor traffic detection framework. It explains the data pre-processing steps, the configuration and training of six models (Decision Tree, Random Forest, XGBoost, MLP, CNN, and RNN), and the evaluation procedure. The chapter then reports and analyzes experimental results across all application scenarios, highlighting performance trends and comparative insights between traditional ML and deep learning approaches.

- **Chapter 5** concludes the thesis by summarizing the main findings and contributions. It also outlines several directions for future work, including extended data collection, advanced obfuscation and adversarial techniques, and the exploration of more sophisticated sequence-modeling architectures.

In addition, the **Appendix** provides supplementary tables and detailed quantitative results for all application categories and models, supporting the analyses presented in the main chapters.

Chapter 2

Related Work

This chapter reviews the foundational work related to anonymization networks, encrypted traffic analysis, flow-based feature extraction, and machine learning techniques used for traffic classification. It also examines existing Tor traffic datasets and highlights research gaps that motivate the development of a fully controlled and up-to-date Tor dataset. The review is organized into six major sections: an overview of the Tor network, Foundations of Tor and Anonymity Networks, Machine Learning for Tor and Encrypted Traffic, Deep Learning for Encrypted Traffic Classification, publicly available existing Tor datasets.

2.1 Tor Network

Tor is a widely deployed anonymity system designed to hide the relationship between a user and the online services they access. It achieves this by forwarding traffic through a sequence of independently operated relays, each of which only knows its immediate predecessor and successor. This layered routing strategy prevents any single point in the network from reconstructing the full communication path. By encrypting traffic multiple times before it leaves the client machine, Tor ensures that intermediate relays cannot inspect the payload or identify the original sender. The network has become an essential tool for individuals who require strong privacy protections, including journalists, activists, and users living under restrictive censorship regimes. At the same time, researchers continue to study Tor's limitations, particularly the fact that observable metadata—such as packet timing and flow patterns—can still leak information even when content remains encrypted. As a result, Tor sits at the center of ongoing debates about online privacy, surveillance resistance, and the future of anonymous communication technologies.

The Tor network (The Onion Router) [45] is one of the most prominent systems

designed to provide online anonymity through multi-hop routing and layered encryption. Originally developed by the U.S. Naval Research Laboratory, Tor routes user traffic through three relays: an entry (guard) node, a middle relay, and an exit node. Each relay decrypts only one layer of the encrypted packet, ensuring that no single relay knows both the source and destination of the communication.

The fundamental architecture of the Tor network was first introduced by Dingledine et al., who presented Tor as a second-generation onion-routing system designed to support low-latency anonymous communication [12]. Their work established core mechanisms such as layered encryption, multi-hop relaying, and fixed-size cell transmission—concepts that continue to underpin modern traffic analysis research.

Subsequent empirical studies expanded the understanding of Tor's operational and behavioral characteristics:

- **McCoy et al.** conducted one of the earliest large-scale measurement studies of the Tor network, revealing insights into relay operations, traffic distribution, and user behavior patterns [33].

These foundational works underscore the inherent tension between Tor's anonymity goals and its susceptibility to traffic fingerprinting based on observable metadata features.

Beyond Tor, several earlier anonymity systems laid the groundwork for low-latency anonymous communication:

- **Chaum's Mix Networks** introduced the principle of layered encryption combined with batching and message mixing to obscure sender–receiver correlations [9]. While highly secure, their reliance on batching made them unsuitable for interactive, low-latency applications.

- **JAP (Java Anon Proxy)** and the **Freedom Network** explored decentralized anonymization architectures but exhibited scalability and performance limitations, preventing wide deployment [12].

Tor improved on these prior systems through innovations such as fixed-size TLS-encrypted cells, circuit-based multiplexing, and directory authorities, enabling interactive anonymity at Internet scale.

Further research strengthened empirical understanding of Tor's design trade-offs:

- **Hopper et al.** analyzed Tor's susceptibility to predecessor attacks and showed how long-lived circuits could allow adversaries to infer client activity patterns [23].
- **Johnson et al.** investigated the guard node selection process and demonstrated that adversarial relays could still mount effective long-term correlation attacks despite Tor's protections [27].
- **Jansen et al.** introduced *Shadow*, a scalable discrete-event simulator enabling realistic Tor experimentation without interacting with the live network, significantly improving reproducibility in Tor research [25].

These works collectively reveal that despite Tor's cryptographic protections, metadata leakage—particularly timing and packet-size information—remains a persistent vulnerability exploited by traffic analysis and machine-learning models.

2.2 Machine Learning and Deep Learning

This section provides a brief overview of the fundamental concepts of machine learning (ML) and deep learning (DL), both of which form the core analytical approaches used in this research. These techniques enable automated pattern discovery from flow-level statistical features and support the development of accurate models for Tor traffic detection. Recent studies show that ML and DL techniques are increasingly effective for classifying encrypted and anonymized network traffic [5, 32].

2.2.1 Overview of Machine Learning

Machine learning (ML) refers to a collection of computational techniques that enable systems to learn patterns from data and make predictions or decisions without being explicitly programmed. A widely accepted definition describes ML as the process by which a program improves its performance on a task through experience [34]. ML algorithms learn by identifying statistical relationships within labeled or unlabeled datasets and using these learned patterns to generalize to previously unseen samples.

In this research, we focus specifically on supervised learning, where the goal is to learn a mapping from flow-level network features to a corresponding class label (Tor or Non-Tor). Supervised learning is particularly well suited for traffic classification tasks because it leverages labeled examples to capture discriminative patterns in encrypted flows.

The input to the learning models consists of flow-level statistical features extracted from Tor and Non-Tor packet captures. These features encode behavioral attributes such as packet-size distributions, flow durations, inter-arrival times, directional byte counts, and other indicators that characterize encrypted network communication. Because Tor encrypts payload contents, detection must rely entirely on such side-channel metadata rather than content inspection, a strategy that has been widely demonstrated in prior encrypted-traffic studies [35, 3, 48].

Several well-established supervised machine-learning algorithms were selected for this study due to their strong performance, interpretability, and robustness. The core classifiers include:

- **Decision Trees (DT)** recursively partition the feature space to form an interpretable set of if–else classification rules [8]. Their transparency makes them useful for understanding how specific flow characteristics contribute to Tor-versus-Non-Tor discrimination.

- **Random Forests (RF)** extend decision trees by aggregating the predictions of many randomized trees, substantially improving generalization and reducing the risk of overfitting [7]. RF models are particularly robust when dealing with noisy or heterogeneous network-flow features.

- **XGBoost (XGB)** ia highly optimized gradient boosting framework, constructs trees sequentially so that each new tree corrects the errors of its predecessors[10]. This allows XGBoost to capture complex nonlinear relationships between flow features, making it particularly effective for encrypted traffic classification.

Tree-based algorithms are advantageous in this domain because they naturally handle heterogeneous feature scales, noisy measurements, and nonlinear interactions

without requiring strict assumptions about data distributions. Moreover, they provide feature importance scores, enabling meaningful interpretation of which flow characteristics— such as burstiness, inter-arrival time variability, or directional packet sizes—are most useful for distinguishing Tor from Non-Tor flows.

Prior work consistently demonstrates that supervised ML methods can classify encrypted traffic with high accuracy even when payload information is unavailable [35, 3]. Their computational efficiency further makes them suitable for real-time or near-real-time deployment. In particular, ensemble methods such as Random Forests have been shown to be strong, robust, and surprisingly competitive across a wide variety of real-world classification tasks [15]. These advantages motivate the use of DT, RF, and XGBoost as core components in the Tor traffic detection framework developed in this thesis.

2.2.2 Overview of Deep Learning

Deep learning (DL) is a specialized subfield of machine learning based on artificial neural networks with multiple hidden layers capable of learning hierarchical representations of data. Unlike traditional ML models—which rely on manually engineered statistical features—deep learning architectures automatically extract complex abstractions from raw or minimally processed inputs [20]. Through layered nonlinear transformations, DL models can capture intricate patterns that may be difficult to specify explicitly, making them particularly attractive for domains involving high-dimensional or temporal data.

In the context of encrypted traffic analysis, deep learning offers the potential to learn latent behavioral characteristics of flows without requiring extensive manual feature engineering. Prior research has shown that neural networks can model packet sequences, timing structures, and flow dynamics more expressively than many classical methods [39, 31]. These capabilities are particularly relevant because encrypted traffic—including Tor—conceals payloads, leaving only metadata patterns for classification.

In this thesis, we examine three widely used deep learning architectures: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network(RNN).

- **Multilayer Perception (MLP)** serves as a baseline fully connected model, transforming flow-level statistical features through stacked dense layers to learn nonlinear decisionboundaries. Although originally designed for tabular data, MLPs remain competitive in many traffic-classification tasks due to their simplicity and ability to approximate arbitrary functions.

- **Convolutional Neural Networks (CNNs)** were originally developed for image and spatial pattern recognition, but they have also been adapted to network traffic classification by applying one-dimensional convolutions to sequential packet features [39]. The convolutional filters enable CNNs to detect local temporal relationships such as short-term burst patterns or repeated structural motifs within flows. However, CNNs may struggle with long-range dependencies or irregular traffic patterns, especially when applied to aggregated flow-level statistics rather than raw packet sequences.

- **Recurrent Neural Networks (RNNs)** RNN-based models, particularly those built on Long Short-Term Memory (LSTM) units, capture long-range temporal dependencies by maintaining internal memory states across sequence elements. This makes them suitable for modeling communication behaviors that unfold over time, such as voice calls, video conferencing, or cumulative flow patterns [21]. Prior studies have demonstrated the usefulness of LSTMs in encrypted traffic classification and Tor traffic analysis, as they can learn subtle sequential behaviors that are difficult for static models to capture [31, 36].

Despite their expressive power, deep learning models present challenges in terms of training cost, hyperparameter sensitivity, and susceptibility to overfitting, especially on datasets with limited variability. Moreover, DL models often lack the interpretability of tree-based algorithms, making it difficult to understand which flow characteristics drive classification decisions. Recent work highlights additional concerns, such as vulnerability to adversarial perturbations and performance instability under distribution shifts [41]. These considerations are particularly important for practical deployment in real-world network monitoring systems.

Overall, while deep learning architectures provide a powerful and flexible framework for modeling encrypted traffic, our experiments show that traditional ML algorithms continue to outperform DL models in flow-based Tor traffic detection. This is largely because flow-level statistical features are highly structured and discriminative, reducing the relative advantage of deep hierarchical models. Nonetheless, deep learning remains an important part of the comparative analysis in this thesis and represents a promising direction for future work involving raw packet-level sequences or fine-grained temporal representations.

2.3 Machine Learning for Tor and Encrypted Traffic Classification

Several works have explored traditional machine-learning methods for classifying encrypted or Tor traffic. Lashkari et al. [30] analyzed Tor flows using timing-based statistical features and demonstrated that tree-based models can effectively distinguish Tor from non-Tor traffic. Similarly, Hodo et al. [22] adopted machine-learning techniques to detect non-Tor traffic within mixed encrypted environments. In broader encrypted-traffic classification literature, Alshammari and Zincir-Heywood conducted multiple influential studies on machine-learning approaches for SSH, Skype, and VoIP classification [3, 4]. Their findings reinforced the value of handcrafted statistical features such as packet lengths, flow durations, and inter-arrival times.

Beyond Tor-specific research, classical ML has long been the dominant approach for encrypted traffic classification. Early ML-based traffic classification (pre-dating Tor) showed that statistical flow attributes provide strong discriminative power even when payloads are encrypted. Key foundational studies include:

- **Moore and Zuev:** Introduced Bayesian statistical fingerprinting for encrypted traffic classification, demonstrating strong accuracy using flow-level features [35].
- **Karagiannis et al.:** Showed that behavioral flow statistics can outperform deep packet inspection (DPI) for application identification, even without payload visibility [28].
- **Williams et al.:** Evaluated supervised learning on NetFlow records and demonstrated that simple statistical features are sufficient for robust encrypted traffic classification [48].

These classical results established the foundation for later Tor-specific studies by confirming that encrypted communication leaks distinguishable side-channel patterns in packet sizes, timing, and directional flow behavior.

Subsequent work further strengthened the role of machine learning in encrypted traffic classification. For example:

- **Kim et al.:** Applied ensemble learning techniques to improve classification accuracy for encrypted VoIP and VPN traffic, highlighting the value of model diversity and feature aggregation [29].

Collectively, these studies emphasize that statistical features—including packet-size distributions, flow durations, jitter, burstiness, and entropy—are highly predictive, forming the foundation for the flow-level features extracted and evaluated in this thesis.

2.4 Deep Learning for Tor and Encrypted Traffic Classification

Deep learning has gained significant attention in encrypted traffic classification due to its ability to automatically learn hierarchical representations from raw packet or flow-level data. Several key studies have advanced the understanding of deep learning methods in this domain:

- **Lotfollahi et al.** introduced *Deep Packet*, a deep-learning framework based on CNNs and stacked autoencoders that classifies encrypted network traffic without requiring manual feature engineering [31]. Their results demonstrated that deep models can extract discriminative patterns even when payloads are fully encrypted.

- **Rezaei and Liu** provided a comprehensive survey of deep-learning techniques for encrypted traffic classification, highlighting recurring challenges such as model scalability, lack of interpretability, and sensitivity to robustness issues [40].

Deep learning has been applied specifically to Tor traffic analysis as well:

- **Pathmaperuma et al.** evaluated deep-learning models for Tor traffic characterization and reported promising improvements over classical machine- learning approaches [37].

- **Rust-Nguyen et al.** conducted a more comprehensive evaluation and examined adversarial attacks on Tor traffic classifiers, revealing that deep-learning and machine-learning models can be fragile when exposed to perturbations in encrypted flows [41].

Collectively, these works highlight both the promise and limitations of deep learning in encrypted traffic analysis. While deep models can automatically learn powerful representations, they remain sensitive to training-data variability, adversarial manipulation, and distribution shifts—factors that are critical for real-world deployment in dynamic network environments.

2.5 Censorship Resistance and Tor Blocking

One of the major motivations behind the Tor Project is to provide censorship-resistant communication. As state-level adversaries have become more capable, research has shifted toward understanding how Tor can be detected, blocked, or throttled, and how it can adapt to avoid censorship.

2.5.1 Detection and Blocking of Tor Traffic

Early research demonstrated that Tor traffic—despite being encrypted—exhibited distinct and easily recognizable TLS handshake patterns. Key findings include:

- **Winter and Lindskog** investigated Tor blocking in China and showed that the Great Firewall was able to identify Tor connections by matching characteristic TLS client fingerprints [49].

- **Farnan et al.** demonstrated that ISPs could reliably detect Tor relays through active probing techniques, allowing network operators to enumerate Tor nodes with high accuracy [14].

- **Edmundson et al.** performed a large-scale analysis of Tor censorship and reported persistent blocking signatures across multiple countries, confirming that Tor's handshake patterns were globally fingerprintable [13].

These works collectively highlight that Tor's pre-2020 TLS handshake behavior made it highly susceptible to detection unless obfuscation mechanisms were deployed.

2.5.2 Tor's Pluggable Transports

When a user enables a Pluggable Transports (PT), the Tor client sends all outgoing traffic through the selected transport module. This module transforms Tor's TLS-like traffic into an obfuscated format. On the server side, Tor bridges run the inverse module that de-obfuscates the traffic and forwards it into the Tor network. Tor supports several Pluggable Transports (PTs) designed to disguise Tor traffic and evade censorship. The most widely used PTs include:

obfs4 – Converts Tor traffic into random-looking encrypted bytes, making it resistant to deep packet inspection and active probing. It is the most commonly deployed PT for Tor bridges in censored regions.

meek – Uses domain fronting to route Tor traffic through large content delivery networks (e.g., Google, Microsoft). To censors, the traffic appears as ordinary HTTPS connections to major cloud services.

Snowflake – A modern PT that relies on volunteer WebRTC proxies. Each volunteer's browser acts as a temporary Tor relay, providing constantly changing IP addresses that are difficult to block.

FTE (Format-Transforming Encryption) – Transforms Tor traffic to match patterns of allowed protocols (such as HTTP or SSH), enabling evasion of DPI systems that rely on protocol signatures.

ScrambleSuit – Obfuscates traffic by adding randomness to packet sizes and timing patterns, while also including authentication to defend against active probing. It served as an early precursor to obfs4.

To resist censorship, Tor introduced a series of Pluggable Transports (PTs) that reshape network traffic to appear benign. Early transports such as Obfs2, Obfs3, and Obfs4 convert Tor flows into streams of seemingly random bytes, making them resistant to deep packet inspection [17]. ScrambleSuit further strengthens obfuscation

by using polymorphic handshake patterns and authentication mechanisms to defend against protocol fingerprinting and active probing [50]. Meek adopts a different strategy by using domain fronting, causing Tor traffic to blend into ordinary HTTPS requests routed through major content delivery networks (CDNs) [16]. Snowflake, Tor's latest and widely deployed PT, leverages WebRTC-based peer-to-peer connections to disguise Tor flows as ephemeral volunteer proxy traffic [24]. Although these transports are effective at bypassing active blocking and protocol identification, they offer limited protection against traffic fingerprinting, as flow-level characteristics such as packet timing, burstiness, and directionality often remain observable to sophisticated adversaries.

2.6 Darknet Datasets

The availability of high-quality datasets is critical for advancing research in Tor traffic analysis and darknet communication studies. Well-designed datasets enable researchers to systematically evaluate machine-learning algorithms, investigate the effectiveness of feature engineering strategies, and benchmark detection methods under controlled as well as real-world traffic conditions. Due to the encrypted and anonymized nature of Tor, collecting representative datasets is technically challenging and often requires specialized experimental testbeds, volunteer-based data collection, or carefully curated real-world traces. As a result, only a limited number of publicly available datasets have become widely adopted within the community.

In the current body of literature, several datasets stand out as foundational resources for Tor-versus-Non-Tor classification, darknet marketplace monitoring, and website fingerprinting research. Prominent among these are the ISCX-Tor2016, CIC-Darknet2020, Anon17, and CTU-13 datasets. Each of these collections embodies distinct design goals, traffic characteristics, and experimental environments, thereby capturing different aspects of Tor usage—from benign web browsing to botnet command-and-control behavior. Together, they form the empirical backbone of much of the state-of-the-art research in traffic analysis and encrypted communication fingerprinting.

The ISCX-Tor2016 dataset provides a controlled lab-generated collection of Tor and Non-Tor flows with detailed flow-level statistical features, making it one of the

earliest benchmarks for encrypted traffic classification. In contrast, CIC-Darknet2020 offers a large-scale, darknet-oriented dataset containing a wide diversity of Tor activities, including onion service browsing and malicious command-and-control interactions. The Anon17 dataset focuses on the behavioral and temporal properties of Tor communication, offering packet-level and flow-level captures designed specifically for anonymity and fingerprinting studies. Finally, the CTU-13 dataset, although originally built for botnet detection, includes Tor-based malware communication scenarios that provide insights into adversarial use of the Tor network.

A detailed description and critical analysis of each dataset are presented in the following subsections, highlighting their collection methodologies, traffic categories, feature sets, limitations, and applicability to modern Tor research. Collectively, these datasets illustrate both the progress made in the field and the ongoing need for clean, reproducible, and modern benchmarking resources.

ISCX-Tor-nonTor 2016 Dataset

The ISCX-Tor2016 dataset, developed by the Canadian Institute for Cybersecurity (CIC), is one of the earliest systematically collected datasets designed specifically to analyze anonymity networks—particularly Tor-based communication [30]. It provides a combination of Tor and Non-Tor traffic traces captured within a controlled laboratory testbed, where human volunteers engaged in a variety of activities such as web browsing, email exchanges, instant messaging, video streaming, and file downloads. By incorporating multiple application categories, the dataset offers a diverse representation of user behavior and serves as an important foundation for early studies on Tor traffic detection and website fingerprinting.

A key advantage of ISCX-Tor2016 is that it includes both raw packet captures (PCAPs) and flow-level statistical features generated using CICFlowMeter [18]. This dual granularity enables researchers to perform in-depth analyses ranging from packet-level behaviors to aggregated flow-level characteristics. The availability of structured flow features has made the dataset particularly attractive for machine-learning-based traffic classification, as it allows rapid experimentation without requiring researchers to perform their own flow extraction.

Despite its relevance, ISCX-Tor2016 suffers from several notable limitations that

impact its suitability for modern traffic analysis research. First, the dataset is relatively small in scale compared to recent large-scale encrypted traffic datasets, limiting its capacity to generalize across contemporary network conditions. Second, the data collection relied heavily on manual user interactions, introducing variability in session durations, browsing behaviors, and timing characteristics. Such inconsistencies can make machine learning models sensitive to user-generated artifacts rather than genuine Tor protocol behaviors. Third, many of the captured patterns no longer reflect modern Tor ecosystem behavior or today's web traffic profile. The dataset was collected prior to major developments such as the widespread adoption of HTTP/2, QUIC, modern Tor circuit management techniques, and updated Tor Browser versions.

Furthermore, the dataset does not include long-duration sessions, background system processes, or realistic multi-tab browser usage, all of which characterize modern Internet traffic. The Tor traces also lack representation of contemporary privacy tools (e.g., Snowflake, obfs4 bridges) and do not account for censorship-evasion behaviors that have become increasingly common. As a result, models trained solely on ISCX-Tor2016 tend to perform unrealistically well and may fail to generalize to real-world Tor environments.

Nevertheless, ISCX-Tor2016 remains one of the most widely used benchmark datasets in the field. Its structured format, accessibility, and historical importance have made it a standard point of comparison for evaluating new Tor traffic classification methods. The dataset's availability also continues to support reproducible research, enabling fair comparison across studies despite its inherent limitations.

CIC-Darknet2020 Dataset

The CIC-Darknet2020 dataset [1] is one of the most comprehensive darknet-focused traffic collections released by the Canadian Institute for Cybersecurity (CIC). Unlike earlier Tor datasets that primarily captured benign browsing activities, CIC-Darknet2020 was specifically designed to model realistic darknet ecosystems. It includes a diverse range of Tor-based communication patterns such as browsing onion services, interacting with darknet marketplaces, visiting illegal content portals, and communicating with malicious command-and-control (C2) infrastructures operated

over Tor hidden services. The dataset also contains benign Non-Tor traffic from everyday applications, enabling meaningful comparative studies between regular Internet usage and darknet-related activity.

A key strength of CIC-Darknet2020 is its scale and richness. The dataset provides millions of labeled network flows, each annotated with ground-truth class information across multiple categories including benign, Tor, VPN-over-Tor, and various darknet threat classes. The traffic was processed through CICFlowMeter-V4 [18], which automatically extracts a comprehensive set of bidirectional flow-level statistical features covering packet counts, byte volumes, timing metrics, inter-arrival times, header-based statistics, and several behavioral properties relevant for encrypted traffic analysis. This multi-dimensional feature set makes the dataset particularly suitable for training machine-learning classifiers aimed at cybercrime detection, anomaly identification, and darknet activity recognition.

However, despite its considerable value for security research, CIC-Darknet2020 is less appropriate for studies focusing on general Tor-versus-Non-Tor discrimination. The dataset is heavily oriented toward malicious or high-risk Tor usage scenarios and does not represent typical benign Tor behaviors such as privacy-preserving web browsing, content consumption, or general-purpose Internet access. As a result, classifiers trained on CIC-Darknet2020 may become biased toward detecting malicious patterns rather than distinguishing Tor anonymity characteristics themselves. Additionally, the dataset's mixed inclusion of Tor, VPN-over-Tor, and other layered anonymization techniques introduces behavioral variations not representative of standard Tor traffic. Consequently, while CIC-Darknet2020 is an excellent benchmark for darknet and cybercrime detection, it is not ideal as a baseline for pure Tor-versus-Non-Tor classification tasks.

Genuine Tor Traces 2023 Dataset

The Genuine Tor Traces 2023 (GTT23) dataset [26] is one of the most recent and realistic collections of Tor network measurements, capturing traffic generated by real users in 2023. Unlike earlier laboratory-produced Tor datasets, GTT23 consists entirely of naturally occurring Tor connections collected in the wild, thereby reflecting authentic user behaviour, access patterns, and temporal dynamics of modern Tor

usage.

To preserve user anonymity and comply with ethical research guidelines, the dataset does not include any sensitive information such as payload content, IP addresses, session identifiers, or personally identifiable metadata. Instead, only high-level traffic descriptors—such as packet timing, directionality, and coarse-grained connection metadata—are recorded. This design ensures strict privacy protection while still providing rich behavioural information for traffic analysis research.

A unique advantage of GTT23 is that each trace is annotated with the destination website domain accessed through Tor. This enables fine-grained website fingerprinting experiments, where researchers attempt to infer which site a user visited based solely on encrypted traffic patterns. Because the dataset is derived from real-world Tor usage rather than scripted browsing sessions, the domain annotations offer a more realistic and diverse ground truth than previous synthetic datasets.

In terms of scale, GTT23 is significantly larger than earlier Tor datasets, containing millions of Tor circuits and connection events spanning a wide variety of websites, network conditions, and user behaviours. This substantial volume makes it particularly valuable for machine-learning research, including studies on traffic classification, deep fingerprinting, and generalization performance under real-world variability.

However, the dataset also has two key limitations. First, it contains only Tor traffic, with no corresponding Non-Tor baseline, which limits its suitability for binary Tor-versus-Non-Tor classification studies. Second, access to GTT23 is restricted and requires formal permission due to ethical and privacy considerations. Despite these constraints, GTT23 remains an essential resource for understanding modern Tor behaviour and evaluating website fingerprinting attacks under realistic conditions.

Anon17 Dataset

The Anon17 dataset [44] was introduced as part of a broader effort to analyze anonymity networks by characterizing the fundamental behaviors of Tor traffic. Unlike traditional intrusion detection or encrypted traffic datasets, which typically prioritize application diversity or attack scenarios, Anon17 was specifically designed to capture the core architectural properties of Tor's onion-routing protocol. These include Tor's use of fixed-size 512-byte cells, its multi-hop circuit construction, inherent

timing jitter, and the periodic circuit rebuilding process, all of which influence traffic patterns in measurable ways.

A key contribution of Anon17 is its dual representation of network activity through both packet-level traces and flow-level statistical summaries. The dataset encompasses several common Tor usage scenarios—most prominently web browsing, file downloads, and interactive user sessions—allowing researchers to examine how Tor's anonymity mechanisms manifest across different types of network behaviors. The packet-level captures provide fine-grained insight into timing, packet direction, and burst structure, while the flow-level statistics help model higher-level behavioral metrics such as inter-arrival time distributions, burstiness, throughput trends, and circuit-level delays.

Despite being more limited in scale and diversity compared to modern datasets, Anon17 remains widely referenced in the literature, particularly in studies related to website fingerprinting, traffic fingerprinting, and deanonymization attacks. Its emphasis on temporal, protocol-level, and behavioral characteristics makes it suitable for controlled experimentation, as it isolates Tor's architectural properties without the confounding effects of heterogeneous applications or noise-heavy real-world network traces.

However, it is important to note that Anon17 reflects the Tor ecosystem as it existed in the mid-2010s. Subsequent changes—such as improvements to congestion control, introduction of newer pluggable transports, and evolving client behaviors—are not represented. As a result, while Anon17 provides valuable foundational insights into Tor's behavior, its applicability to modern Tor traffic classification is limited unless supplemented with more recent or realistic datasets.

CTU-13 Dataset

The CTU-13 dataset [19], developed by the Stratosphere Laboratory at the Czech Technical University in Prague, is one of the most widely used benchmark datasets for botnet and malware traffic analysis. It comprises thirteen carefully constructed scenarios, each capturing a specific botnet family such as Neris, Rbot, Virut, Menti,

and others. These scenarios include various stages of botnet activity—such as infection, command-and-control (C2) communication, propagation, and data exfiltration—allowing researchers to study different behavioral patterns exhibited by compromised hosts.

Each scenario consists of three major traffic components: (1) Background traffic, reflecting routine network operations occurring within the monitored environment; (2) Benign user activities, such as web browsing or normal workstation usage; and (3) Labeled malicious flows, enabling ground-truth-based evaluation of intrusion detection and malware identification techniques. The presence of clearly annotated botnet behaviors has made CTU-13 a foundational dataset in network security research, particularly for supervised machine-learning approaches.

Although the dataset was not designed with anonymity-network research as its primary focus, several botnet families in CTU-13 rely on Tor-based command-and-control channels. These Tor-enabled scenarios illustrate how malware authors exploit onion routing to conceal C2 servers, hide communication patterns, and evade IP-based detection mechanisms. As a result, CTU-13 offers valuable examples of malicious Tor traffic, providing insight into how Tor can be misused as a covert communication layer in real attacks.

However, CTU-13 has notable limitations for general Tor-versus-Non-Tor classification research. The Tor flows present in the dataset are malicious, limited in volume, and tied to a small set of botnet behaviors, rather than representing typical or diverse Tor user activities such as browsing, streaming, or file transfer. Additionally, the dataset lacks large-scale benign Tor sessions and does not include controlled non-Tor equivalents for comparative analysis. Therefore, while CTU-13 remains an important resource for studying malware behaviors and Tor-based C2 channels, it is not suitable as a standalone dataset for evaluating general Tor traffic detection or fingerprinting models.

Chapter 3

Realistic Tor Dataset Generation

This chapter presents the methodology used to design, implement, and evaluate a fully controlled framework for generating realistic Tor and Non-Tor traffic. The methodology includes the architecture of the testbed, traffic generation procedures, the deployment of a noise-free server, packet capturing settings, and a flow-based feature extraction pipeline. This framework ensures reproducibility, comparability, and isolation of Tor and Non-Tor scenarios, providing a clean dataset suitable for machine learning analysis.

3.1 Overview of Tor Dataset Generation Framework

The primary objective of the framework is to generate realistic Tor and Non-Tor traffic for modern applications under identical and controlled conditions. To achieve this, a virtualized environment using Oracle VirtualBox was configured, consisting of two virtual machines: a Debian Workstation and a Whonix Gateway. The Debian Workstation serves as the traffic generator, while the Whonix Gateway provides Tor routing functionality.

Figure 3.1 illustrates the high-level architecture of the system, consisting of two separate network interface cards (NICs) to isolate Tor and Non-Tor traffic paths.

The dual-NIC design and controlled environment guarantee that both Tor and Non-Tor traffic are captured under identical application conditions, enabling a fair comparison for downstream machine learning tasks.

3.2 Framework Architecture

3.2.1 Debian Workstation

The Debian Workstation is responsible for generating all user activities including web browsing, video streaming, file transfers, instant messaging, VoIP calls, and video

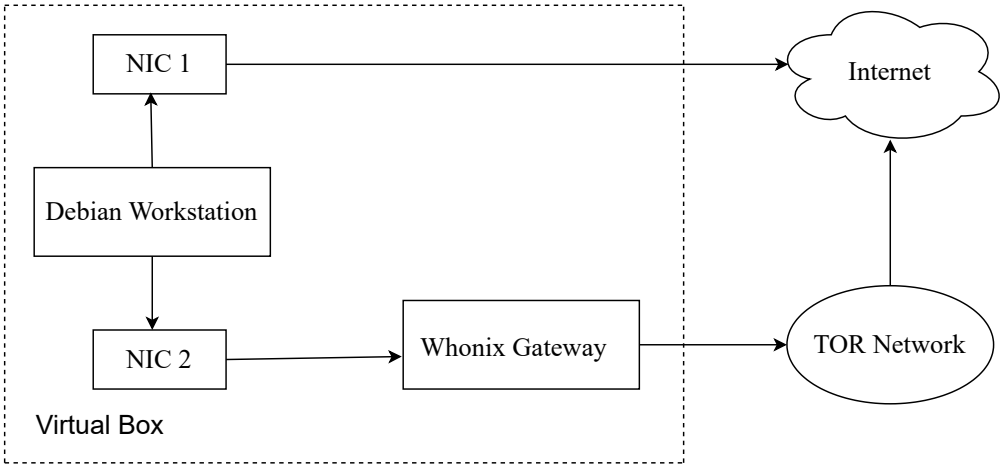

Figure 3.1: Testbed architecture used to generate realistic Tor and Non-Tor traffic.

conferencing. The system was configured with the following properties:

- Debian GNU/Linux 12 (bookworm)
- 4 GB RAM, 2 CPU cores, VirtualBox virtualization
- Google Chrome installed
- TCP-only mode under UDP blocking
- Selenium WebDriver for scripted automation

In our testbed architecture, two virtual network interfaces were configured on the Debian Workstation to ensure a clean separation between Tor-routed and direct Internet traffic. This dual-interface design enables controlled, reproducible, and interference-free data collection for both traffic categories.

NIC 1 (Non-Tor Path). NIC 1 provides the workstation with a direct, unproxied connection to the public Internet. All packets transmitted through this interface follow the standard TCP/IP routing path without any anonymization layers, proxy services, or intermediate relays. Consequently, the traffic captured on NIC 1 represents clean and unmodified Internet communication, making it the ground-truth source for the Non-Tor dataset. Because no Tor components interact with this interface, it preserves genuine timing characteristics, packet size distributions, and flow

25 behaviors associated with regular network usage. This ensures that the Non-Tor dataset reflects realistic browsing, streaming, and application activities under ordinary network conditions.

NIC 2 (Tor Path). NIC 2 is dedicated exclusively to Tor traffic. This interface connects the Debian Workstation to the Whonix Gateway, which acts as the transparent Tor router in the system. All packets routed via NIC 2 are intercepted by the Whonix Gateway and subsequently processed through Tor's multi-layered anonymization pipeline. The gateway handles Tor circuit establishment, multi-hop relay selection, encryption and decryption across the onion layers, and the forwarding of packets through guard, middle, and exit relays. As a result, all traffic observed on NIC 2 accurately represents authentic Tor communication, including its inherent characteristics such as variable latency, circuit rotation, packet batching, and routing randomness. This guarantees that the Tor dataset captures real-world Tor behavior rather than simulated or proxy-based approximations.

Design Rationale. By assigning Tor and Non-Tor traffic to distinct physical/virtual interfaces, the system achieves strict traffic isolation, eliminating risks of cross-contamination or misclassification during data capture. This separation ensures that each dataset—Tor and Non-Tor—retains its unique statistical properties, enabling accurate downstream analysis, fair comparison, and reliable machine-learning evaluation. Moreover, this architecture mirrors realistic operational behavior of Tor clients while preserving full experimental control within a virtualized testbed environment.

3.2.2 Whonix Gateway

The Whonix Gateway operates as a dedicated Tor routing appliance that isolates and anonymizes all traffic generated by the Debian Workstation. Built on a hardened, security-oriented architecture, the gateway is responsible for establishing and maintaining Tor circuits, enforcing onion routing at the network boundary, and ensuring that no packet leaves the workstation without passing through the Tor network. Specifically, the Whonix Gateway handles Tor circuit construction and rotation, manages relays, and applies strict network firewalling rules that prevent any form of direct Internet connectivity from the workstation [46]. As a result, all packets transmitted

over NIC 2—including DNS queries, TCP streams, and control traffic—are transparently proxied through Tor's multi-hop encrypted routing infrastructure, eliminating the possibility of DNS leaks, IP leaks, or proxy bypass attempts [12].

Using Whonix instead of the standalone Tor Browser provides several methodological advantages for controlled dataset generation. The Tor Browser includes browser-specific background activities such as telemetry, update checks, and circuit isolation mechanisms that introduce noise unrelated to application-level traffic. In contrast, Whonix offers a clean and reproducible environment where traffic originates solely from the researcher's chosen applications (e.g., web browsing, video streaming, file transfer), thereby maximizing experimental repeatability. Moreover, the gateway–workstation isolation model aligns with best practices in operational security by separating user applications from networking logic, improving both privacy and experimental control [47]. This architecture ensures that the captured traffic on NIC 2 represents authentic Tor-routed communication—free from browser artifacts—and is therefore suitable for reliable machine-learning analysis and Tor vs. Non-Tor classification research.

3.2.3 Packet Capture Configuration and Data Management

Accurate and reliable packet capture is essential for building a high-quality dataset. In this work, all traffic was captured at the Debian Workstation using Wireshark and command-line tools such as `tshark`. Capturing at the client side provided full visibility into both Tor and Non-Tor flows, while avoiding potential packet loss or reordering that might occur at intermediate network devices.

Separate capture sessions were performed for each experiment and for each interface. For example, during a web browsing experiment, one capture file recorded Non-Tor traffic on NIC 1, and a separate file recorded Tor traffic on NIC 2. This separation simplifies subsequent labeling and prevents ambiguity about whether a given flow belongs to the Tor or Non-Tor class. Capture filters were kept minimal (typically capturing all IP traffic) to avoid accidentally excluding relevant packets, especially control traffic and retransmissions that might carry important timing information.

Packet captures were stored in the standard `pcapng` or `pcap` formats and organized into a directory hierarchy based on application category (web browsing, video

streaming, file transfer, instant messaging, VoIP, and video conferencing), routing mode (Tor or Non-Tor), and session index. Each one-hour experiment produced a pair of capture files (one for Tor, one for Non-Tor), which were then processed by the flow-extraction script. This naming and organization scheme facilitated traceability between raw captures, extracted flow-level CSV files, and the final combined datasets used for machine-learning experiments.

The use of a one-hour collection window for each experiment also plays an important role in the reliability and performance of the proposed detection methods. Longer capture durations allow flows to accumulate a sufficient number of packets, producing more stable and representative statistical features. Metrics such as average packet length, inter-arrival time variation, jitter, entropy, and flow duration can fluctuate significantly in very short captures due to temporary bursts, idle periods, or protocol start-up behavior. These short-term artifacts do not accurately reflect the true behavior of Tor or Non-Tor traffic and may degrade classifier performance. In contrast, extended one-hour sessions smooth out these fluctuations and capture the full behavioral characteristics introduced by Tor's multi-hop routing, batching, and congestion-control mechanisms. This richer and more consistent feature space enhances class separability and contributes directly to the strong performance observed in the machine-learning models.

To ensure data integrity and reproducibility, all capture files and derived CSV datasets were versioned and backed up to a separate storage device. Basic sanity checks were performed after each experiment, including verifying the duration of the capture, the number of packets recorded, and the absence of truncated files. When anomalies were detected (e.g., an experiment interrupted before one hour, or an unusually low packet count due to network issues), the experiment was repeated from the latest virtual machine snapshot.

The flow-extraction pipeline recorded metadata such as the source capture file, interface, and application scenario for each generated CSV. This metadata was used to construct higher-level aggregated datasets per application and to maintain a clear mapping between experimental conditions and the flows observed by the machine-learning models. The resulting data management workflow ensures that every result presented in this thesis can be traced back to a specific set of raw packet captures

and experimental configurations.

3.2.4 Automated Traffic Generation Using Selenium WebDriver

To ensure consistent, reproducible, and realistic traffic patterns across all experimental scenarios, an automated browser-based traffic generation framework was developed using Selenium WebDriver [43]. Automation was essential for controlling user interactions, eliminating human variability, and producing synchronized Tor and Non-Tor traffic traces.

In our project, Selenium (version: 4.35.0) was configured to launch Google Chrome on the Debian Workstation and execute predefined interaction sequences such as opening webpages, waiting for random intervals, and starting video playback. For video streaming experiments, Selenium automatically selected the desired resolution (240p, 480p, or 1080p) and initiated playback on both YouTube and the controlled server. Similarly, for file-transfer tests, Selenium triggered downloads at fixed or randomly chosen file sizes, with randomized waiting periods between subsequent requests.

The automation scripts were executed identically under Tor and Non-Tor conditions to guarantee fair comparison. Each script followed the same timing logic, random delays, and user-behavior patterns, ensuring that any differences in the captured network traffic were attributable solely to the underlying routing mechanism rather than variations in user behavior.

Overall, Selenium-based automation provided a robust, scalable, and repeatable framework for generating realistic network traffic, forming the foundation for the controlled Tor vs. Non-Tor dataset used in this research.

3.3 UDP Blocking

Although UDP is a widely used transport protocol for modern web applications, the Tor network does not natively support UDP forwarding. All Tor traffic is encapsulated within TCP streams, and any protocol requiring UDP—such as QUIC, WebRTC, or certain VoIP and video-conferencing features—cannot be transmitted through Tor's onion-routing architecture. Prior work highlights that this protocol mismatch introduces substantial behavioral differences between Tor and Non-Tor traffic [12].

In typical networking environments, many applications opportunistically select UDP-based protocols when available. For instance, platforms such as YouTube, Microsoft Teams, and other real-time communication systems preferentially rely on QUIC (UDP-based HTTP/3) or WebRTC. These protocols significantly alter packet timing, burstiness, retransmission behavior, and congestion-control dynamics. Consequently, if Non-Tor traffic employs UDP while Tor traffic is restricted to TCP, the resulting datasets differ not due to Tor's anonymity mechanisms, but solely due to transport-protocol selection. This introduces artificial bias and may cause machine-learning classifiers to achieve unrealistically high accuracy by learning protocol-level differences instead of genuine Tor characteristics.

To ensure strict comparability between Tor and Non-Tor datasets, all UDP communication was explicitly disabled on the client workstation. The Uncomplicated Firewall (`ufw`) was configured to deny all inbound and outbound UDP packets, and supplementary `iptables` rules were deployed to drop residual UDP flows, including QUIC handshake packets and WebRTC negotiation messages. Once UDP was blocked, all applications were forced to fall back to TCP-based communication, even for video streaming, VoIP, and interactive services.

This configuration ensured that:

- Both Tor and Non-Tor sessions used an identical transport-layer protocol (TCP);
- Differences between the two datasets arose from Tor's multi-hop routing, circuit construction, congestion behavior, and layered encryption, rather than protocol-level discrepancies;
- QUIC- or WebRTC-specific statistical artifacts were eliminated, preventing classifiers from learning trivial protocol fingerprints instead of anonymization-related patterns.

By aligning the transport-layer behavior across both environments, this controlled design strengthens experimental validity and ensures that any observed distinctions are attributable to Tor's anonymity mechanisms rather than disparities in protocol stacks.

In summary, UDP needed to be blocked to ensure a fair and scientifically valid

comparison between Tor and Non-Tor traffic. Because Tor supports only TCP, allowing UDP would cause Non-Tor applications to switch to UDP-based protocols such as QUIC and WebRTC, creating a fundamental protocol mismatch. In such a scenario, machine-learning models could trivially distinguish Tor from Non-Tor traffic based solely on transport-layer differences rather than meaningful behavioral characteristics. By forcing both environments to operate exclusively over TCP, the dataset isolates differences that arise from Tor's multi-hop routing, circuit behavior, and encryption design—ensuring that any classification performance reflects true Tor-related patterns rather than protocol-level artifacts.

3.4 Noise-Free Server

A custom noise-free server was deployed on Amazon Web Services (AWS) to generate controlled traffic free from unpredictable noise such as advertisements, auto-play content, recommendations, and analytics. The server was provisioned using Terraform for consistency and repeatability.

3.4.1 AWS Configuration

The server was deployed with the following characteristics:

- Ubuntu Server 24.04.3 LTS
- Apache HTTP Server 2.4.58
- PHP 8.3 runtime
- Hosted in `us-east-1` region
- Firewall rules restricted to ports 22, 80, and 443

3.4.2 Customized Data

The server hosted three core components:

1. **Web pages:** Web browsing was simulated using clean.php, a minimal PHP script that returned raw ASCII output of a configurable size defined by URL

parameters. Each request produced text of random characters, allowing control of page size while avoiding dynamic rendering effects. This approach provided a simple yet flexible setup for evaluating network traffic under controlled conditions.

2. **Video files:** The `/videos` directory contained MP4 files encoded at three resolutions: 240p, 480p, and 1080p, each accompanied by a corresponding thumbnail image. A lightweight `video.php` script was developed to embed these files into an HTML5 `<video>` element, providing a minimal streaming interface similar to modern video platforms. This configuration preserved realistic transfer patterns while maintaining a controlled, reproducible, and resource-efficient environment for generating video streaming traffic.

3. **File transfer:** File transfer experiments utilized several static data files (random 10MB.dat through random 100MB.dat), which were generated using the Linux `/dev/urandom` interface. This approach produced high-entropy binary data, ensuring that the files did not contain any compressible patterns or repetitive structures. Each file was created by reading a fixed number of bytes from `/dev/urandom` and writing them directly to disk, for example:

   ```
   dd if=/dev/urandom of=random_100MB.dat bs=1M count=100
   ```

   All files were then served through Apache without compression or caching enabled, ensuring that the server delivered the raw binary content exactly as stored on disk. This configuration provided consistent, reproducible, and noise- free transfer behavior, making the resulting network traces suitable for flow-level feature extraction and machine learning analysis.

   By eliminating external noise, the traffic becomes highly reproducible and ideal for evaluating Tor vs. Non-Tor classification.

3.5 Applications under Investigation

To generate a realistic and diverse dataset, we considered six categories of network activities that broadly represent everyday Internet usage patterns: web

browsing, video streaming, file transfer, instant messaging, voice over IP (VoIP), and video conferencing. These categories were carefully selected to cover a wide range of traffic behaviors, packet characteristics, and temporal dynamics commonly seen in modern applications. For each category, the traffic generation experiment was executed twice first through the Tor network and then through a direct connection to the Internet ensuring a controlled comparison between anonymized and non-anonymized environments. All traffic was captured at the packet level using Wireshark [51], allowing us to collect detailed, fine-grained traces for feature extraction and subsequent analysis.

3.5.1 Web Browsing

Web browsing traffic was generated by automating user interactions using Selenium WebDriver, a widely used browser automation framework. This approach allowed us to reproduce consistent, human-like browsing behavior without manual intervention, ensuring that both Tor and Non-Tor sessions followed identical browsing patterns. Two distinct browsing modes were designed to capture both controlled and uncontrolled page delivery environments:

- **Noise-free Server Pages:** We hosted a custom lightweight PHP script on a distraction-free web server capable of generating web pages of arbitrary sizes. For each request, the script produced raw ASCII output whose size was randomly selected between 0.5 MB and 5 MB, controlled strictly through URL parameters. This setup enabled us to create deterministic page sizes without dynamic content, caching effects, or external network noise, leading to clean and comparable traffic profiles.
- **Wikipedia Pages:** To complement the controlled environment, we included browsing sessions to publicly accessible Wikipedia pages. Since these pages contain mixed content (text, images, CSS, JavaScript, and additional embedded elements), the exact page size could not be controlled. This provided natural variability and helped capture real-world browsing behavior within the dataset.

Each browsing experiment lasted one full hour, during which Selenium automatically switched between pages every 5 and 15 seconds. This random interval was carefully chosen to mimic natural user behavior long enough for pages to load completely but short enough to represent typical exploration patterns seen in normal browsing. Importantly, we maintained a predefined, fixed sequence of URLs for both Tor and Non-Tor sessions. The same sequence of web pages was visited in both sessions, first under Tor and then under direct Internet connectivity enabling a fair, one-to-one comparison of how Tor alters traffic characteristics for the same underlying user activity.

3.5.2 Video Streaming

Video streaming traffic was generated to capture both controlled and real-world streaming behaviors, as these applications produce rich temporal and volumetric patterns that are highly useful for traffic classification. To achieve this, we used two streaming sources: a fully controlled, noise-free server and the public YouTube platform.

- **Noise-free Server:** The configuration of the Noise-free server is described in the next section. For the streaming experiments, the server hosted six original video clips, each approximately ten minutes long and recorded at 1080p resolution. These clips were transcoded into three different resolutions 240p, 480p, and 1080p to represent low, medium, and high video quality settings. Because the server introduced no compression, caching, adaptive bitrate selection, or third-party network effects, the resulting traffic traces reflect clean, reproducible video flows suitable for controlled analysis.
- **YouTube:** To complement the controlled environment, the same six video clips were uploaded to YouTube and streamed directly from this platform. YouTube's adaptive streaming mechanisms and buffering behavior introduce natural variability into the traffic, allowing us to capture realistic streaming patterns that users encounter in everyday scenarios.

Each streaming session lasted one hour and involved automated playback of the video clips. Using Selenium automation, videos were played sequentially, with the viewing duration for each clip randomized between 1 and 5 minutes. This randomness was designed to emulate natural user behavior, such as switching videos or watching short segments.

To ensure consistency and comparability across captures, we maintained a fixed playback sequence. The exact same set of video clips, in the same order, was streamed twice: first through the Tor network and then through direct Internet connectivity. This design ensured that any differences observed in the resulting traffic were attributable to the underlying network conditions rather than variations in content or playback behavior.

3.5.3 File Transfer

File transfer traffic was generated to capture the behavior of bulk data downloads under both Tor and direct Internet connectivity. Unlike interactive browsing or streaming scenarios, file transfers typically produce long-lived, high-volume flows with consistent throughput patterns, making this category essential for understanding how Tor affects sustained data movement.

To ensure a controlled and reproducible environment, all file downloads were served from our Noise-free server. This approach was selected because commercial cloud services such as OneDrive, Google Drive, and Dropbox restrict continuous or sequential downloads, which limits the ability to produce consistent traffic captures. The server hosted three types of files - 10 MB, 50 MB, and 100 MB—each generated using `/dev/urandom` to create incompressible binary data. This approach eliminated the influence of caching, compression, or content optimization, thereby producing clean and deterministic traffic profiles.

Each file transfer session lasted one hour. During this period, files were downloaded repeatedly, with the inter-download gap randomized between 5 and 15 seconds. This setup emulated realistic user behavior where files downloaded in intervals rather than continuously. The selection of file size for each request followed one of two strategies: a fixed-size mode, where only one file size was

downloaded throughout the session, and a randomized mode, where the file size was selected uniformly at random among the three available options.

As with the other traffic categories, a fixed download sequence was maintained to ensure consistency across Tor and Non-Tor sessions. The exact same ordering of requested files was followed in both environments first under Tor and then under direct Internet connectivity allowing a direct comparison of throughput, latency, and flow-level characteristics under the two network conditions.

3.5.4 Instant messaging (IM)

Instant messaging (IM) traffic was captured using Microsoft Teams to represent real-time, text-based communication. Unlike the automated scenarios used for web browsing, video streaming, and certain file transfer activities, messaging behavior is inherently irregular and user-driven. Therefore, the IM traffic in our dataset was generated manually to more accurately reflect natural communication patterns.

During each one-hour session, short text messages were exchanged at unpredictable intervals, simulating realistic human conversation. This included actions such as typing brief messages, responding to incoming messages, waiting between replies, and occasionally sending multiple messages in quick succession. These variations introduced temporal randomness and burstiness that are typical in real-world IM usage but difficult to reproduce accurately through automation.

Both Tor and Non-Tor sessions followed the same interaction style to ensure comparability. By capturing manually generated messages under identical behavioral patterns, we were able to observe how Tor influences the timing, packet count, and flow structure of interactive, low-volume communication.

3.5.5 Voice over IP (VoIP)

VoIP traffic was generated using Microsoft Teams to capture the characteristics of real-time, low-latency audio communication. VoIP applications are particularly sensitive to delay, jitter, and packet loss, making them an important traffic category for understanding how Tor influences interactive communication.

For each experiment, a continuous one-hour audio call was conducted. During the session, normal conversational behavior was simulated by speaking intermittently, pausing naturally, and occasionally introducing short bursts of speech. This produced a mixture of steady-state audio transmission and silent periods, reflecting typical VoIP usage patterns.

Microsoft Teams uses TCP-based communication for call control, media transport, and connection management, resulting in distinct flow structures that differ from bulk-transfer or browsing traffic.

The same one-hour audio interaction was performed twice: first over the Tor network and then over direct Internet connectivity. By maintaining consistent speaking patterns and session duration, we ensured that the observed differences in traffic behavior stemmed from the underlying network conditions rather than variations in user activity.

3.5.6 Video Conferencing (VC)

Video conferencing traffic was captured using Microsoft Teams to represent real-time, bidirectional audio–video communication. Unlike single-modal activities such as VoIP or text messaging, video conferencing simultaneously transmits video frames, audio streams, screen updates, and control packets, resulting in complex and highly dynamic flow patterns. This makes it an essential category for evaluating how Tor affects high-bandwidth, delay-sensitive applications.

For each experiment, a continuous one-hour Microsoft Teams meeting was conducted. The session included live two-way communication, during which the camera and microphone remained enabled for the full duration. Natural interaction behaviors—such as speaking intermittently, moving in front of the

camera, and performing small gestures—were intentionally included to introduce realistic variability in video encoding rates and audio activity. These actions generated fluctuating bitrates, periodic bursts of packets, and mixed traffic types typical of real-world conferencing applications.

Microsoft Teams employs adaptive encoding mechanisms that adjust video quality based on network conditions. As a result, the captured traffic contains variations in frame size, transmission rate, and audio-video synchronization patterns. These dynamics allow for an accurate assessment of how Tor's routing and latency influence multimedia performance.

The video conferencing experiment was performed twice: once over the Tor network and once through direct Internet connectivity. Both sessions followed the same interaction style and duration, ensuring that any differences observed in the traffic characteristics were attributable to the underlying network environment rather than variations in user behavior.

3.6 Flow-Based Feature Extraction

In our research, we implemented a custom flow-based feature extraction script using Scapy [6]. The script processes raw packet capture (pcap) files and aggregates individual packets into bidirectional flows based on the standard 5-tuple definition: *Source IP address, Destination IP address, Source Port, Destination Port, and Transport Protocol*. This approach provides a structured representation of communication sessions and enables the computation of meaningful statistical descriptors for machine learning.

To ensure robust and temporally consistent segmentation, each flow is terminated under one of two conditions: (i) when no packets are observed for an idle timeout period of 120 seconds, or (ii) when a TCP connection teardown is detected through the presence of a `FIN` or `RST` flag. These termination rules prevent excessively long flows and ensure correspondence with natural communication boundaries. Once a flow is closed, all computed statistics are immediately written to a CSV file for preprocessing and later use in machine-learning pipelines. Table 3.1 presents the flow statistics for the video streaming and file

Table 3.1: Flow Statistics for Video Streaming and File Transfer

Category	**Resolution**	**Tor Flows**	**Non-Tor Flows**
YouTube	240p	1237	1064
	480p	901	1247
	1080p	973	1131
	Random	916	1150
Noise-free Server (Video Streaming)	240p	265	439
	480p	261	391
	1080p	285	420
	Random	436	477
Noise-free Server (File)	240p	680	677
	480p	351	315
	1080p	215	131
	Random	286	293

transfer categories, while Table 3.2 summarizes flow counts for all high-level application categories.

In total, we extracted 48 statistical features capturing multiple dimensions of network behavior. Table 3.3 presents the extracted flow-level statistical features along with their descriptions. These features collectively represent a comprehensive characterization of each bidirectional flow, enabling the models to learn subtle behavioral differences between Tor and Non-Tor traffic. Specifically, the feature set covers identification attributes (such as IP addresses, ports, and protocol), timing-related metrics that capture session duration and temporal dynamics, and multiple measures of traffic volume including packet counts and byte totals. Directional statistics are also included to differentiate flow behavior in the forward and backward directions, reflecting real-world asymmetry in client–server communication.

In addition, packet length features and their directional counterparts describe variability and distribution of packet sizes, while inter-arrival time (IAT) metrics quantify temporal spacing between successive packets—an important indicator of application-level behavior and congestion patterns. Directional IAT features further refine this characterization by isolating client-originated and

Table 3.2: Flow Statistics for other Applications

Category	**Application**	**Tor Flows**	**Non-Tor Flows**
Web Browsing	Wikipedia	522	864
	Noise-Free Server	1024	1252
Instant Messaging	MS Teams	2,912	2,740
VoIP	MS Teams	1,654	1,703
Video Conference	MS Teams	3,227	3,451

server-originated timing patterns. Finally, TCP flag counters provide insights into connection control events, retransmissions, and protocol signaling behavior. Together, these 48 features form a rich representation of encrypted network flows, allowing machine learning models to detect Tor traffic without relying on payload or deep packet inspection.

3.6.1 Impact of Flow Definition on Classification Performance

The definition of a flow, particularly the idle timeout interval, influences how packets are grouped into flows and how statistical features are computed. Short timeouts fragment Tor sessions into many small flows, reducing the availability of long-term temporal characteristics such as burstiness, inter-arrival patterns, and jitter variability. Conversely, excessively long timeouts may merge packets from multiple Tor circuits, which typically rotate every 10 minutes, mixing distinct behaviors into a single flow.

This thesis adopts a 120-second idle timeout value, which balances granularity and stability. It ensures that each flow retains meaningful temporal structure while avoiding unintended blending across different Tor circuits. Experimental results indicate that this flow definition leads to consistent and accurate classification performance across all application categories.

3.7 Realism of the Generated Dataset

The dataset developed in this thesis provides a more realistic and representative view of modern Tor and Non-Tor traffic compared to existing datasets such as

Table 3.3: Extracted Flow-Level Statistical Features

Category	Feature	Description
Identification	SrcIP, DstIP	IP addresses of the sender and receiver used to identify endpoints of the flow.
	SrcPort, DstPort	Source and destination port numbers indicating application-level interaction.
	Protocol	Transport-layer protocol (TCP/UDP) used for flow construction.
Timing	Flow Start Time	Timestamp of the first packet observed in the flow.
	Flow End Time	Timestamp of the last packet in the flow.
	Duration	Total lifetime of the flow (End Time – Start Time).
Traffic Volume	Total Packets	Total number of packets (forward + backward) in the flow.
	Total Bytes	Total number of bytes transferred in both directions.
	Packets/s	Packet rate across the entire flow duration.
	Bytes/s	Byte transfer rate averaged over flow duration.
Directional Traffic Volume	Fwd/Bwd Packet Count	Number of packets sent in the forward and backward directions.
	Fwd/Bwd Byte Count	Total bytes sent in forward and backward directions.
Packet Length Statistics	Min, Max, Mean, Std of Packet Lengths	Statistical summary of packet sizes across the entire flow.
Directional Packet Length	Fwd/Bwd Min, Max, Mean, Std	Packet length statistics computed separately for each direction.
Inter-Arrival Time (IAT)	Min, Max, Mean, Std of IATs	Time gap statistics between consecutive packets in the flow.
Directional IAT	Fwd/Bwd Min, Max, Mean, Std	Inter-arrival time metrics computed separately per direction.
TCP Flags	FIN, SYN, RST, PSH, ACK, URG, ECE, CWR	Counters tracking the occurrence of each TCP control flag within the flow.

ISCX-Tor2016, CIC-Darknet2020, Anon17, GTT23, and CTU-13. Several key characteristics contribute to the improved realism of the proposed dataset:

(a) **Modern and Diverse Internet Applications**
Most existing Tor datasets focus on limited or outdated traffic types, such as simple web browsing, darknet activity. In contrast, this dataset includes six widely used and contemporary applications: dynamic web browsing (Wikipedia and a controlled noise-free server), YouTube streaming at multiple resolutions (240p–1080p), server-hosted video playback, Microsoft Teams instant messaging, Teams audio calls, and Teams video conferencing. These applications reflect current Internet behaviors and create traffic patterns that older datasets cannot capture.

(b) **Controlled and Noise-Free Testbed Environment**
The dataset was collected using a dual-NIC Debian workstation and a dedicated Whonix Gateway, ensuring strict separation between Tor and Non-Tor traffic. This eliminates background system noise such as software updates, telemetry, or DNS leaks. All captured traffic is therefore clean, accurate, and directly tied to the intended activities, unlike many older datasets with uncontrolled noise.

(c) **Paired Tor and Non-Tor Sessions for Fair Comparison**
Each experiment was executed twice—once over Tor and once over the direct Internet—using the exact same browsing sequence, video playback timeline, and messaging pattern. This provides a fair one-to-one comparison between Tor and Non-Tor. Existing datasets rarely use mirrored experiments, which makes them less comparable and sometimes biased.

(d) **Long, Continuous One-Hour Capture Sessions**
All six applications were recorded for uninterrupted one-hour sessions. This enables the dataset to capture Tor circuit rotations, congestion variation, traffic burstiness, and multi-phase application behavior. Previous datasets often include short, synthetic, or incomplete sessions, limiting their representativeness.

(e) **Modern Tor Environment and Real Routing Behavior**

The dataset uses the latest Tor implementation and a Whonix-based routing configuration, including real relay selection, multi-hop routing, and periodic circuit rotation. Many older datasets rely on outdated Tor versions or simplified routing models, reducing their relevance to present-day Tor behavior.

(f) **Balanced, Well-Labeled, and Reproducible Data**
All flows are carefully labeled, version-controlled, and traceable back to their raw PCAP files. Automation ensures consistent behavior across runs. Older datasets often contain mislabeled flows or uncontrolled user activity, reducing reliability.

Taken together, these characteristics make the proposed dataset substantially more realistic, diverse, and representative of modern Tor and Non-Tor traffic, providing a stronger foundation for accurate machine-learning analysis.

3.8 Ethical and Privacy Considerations

All experiments conducted in this thesis were designed with privacy and ethical considerations in mind. First, the dataset contains only traffic generated within a controlled laboratory environment using a small set of virtual machines and cloud servers under the researcher's control. No traffic from uninvolved third-party users or external clients was captured or analyzed. When public services such as Wikipedia, YouTube, or Microsoft Teams were used, the interactions were limited to test accounts and benign activities that do not involve sensitive personal information.

Second, the feature extraction pipeline operates exclusively on flow-level statistics and does not inspect or store packet payloads. As a result, the resulting dataset does not expose the content of web pages, messages, audio, or video streams. Instead, it contains only aggregate timing, size, and count-based metrics that are sufficient for machine-learning analysis but substantially reduce privacy risks.

Third, the goal of this research is to understand whether Tor traffic can be distinguished from Non-Tor traffic under controlled conditions, not to deanonymize individual users or infer specific websites visited through Tor. Any potential misuse of such techniques is beyond the intended scope of this thesis. By releasing the dataset and methodology primarily for academic research, the intention is to support the development of transparent, well-understood detection methods and to inform the design of stronger privacy-preserving mechanisms.

Finally, should the dataset be shared with the wider research community, appropriate documentation and licensing terms will be provided to ensure responsible use. These terms will emphasize that the data must not be combined with external user-identifying information or deployed in ways that violate legal or ethical framework.

Chapter 4

Machine Learning-based Tor Traffic Detection

After generating the realistic Tor dataset, we carried out a comprehensive analysis to understand whether machine learning models can classify Tor and Non-Tor traffic successfully. Specifically, we leveraged six machine learning models to detect Tor traffic in the dataset. The models adopted in our research include Decision Tree (DT), Random Forest (RF), XGBoost (XGB), Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN).

4.1 Data Pre-Processing

In this study, both the Tor and Non-Tor flow datasets were initially imported and subjected to a systematic preprocessing pipeline to ensure fairness, reproducibility, and the removal of environment-specific biases. As a first step, all host-specific identifiers were removed—including `SrcIP`, `DstIP`, `SrcPort`, `DstPort`, and `Proto`—because these attributes may inadvertently encode information about the testbed rather than genuine traffic characteristics. Additionally, temporal identifiers such as `Flow Start Time` and `Flow End Time` were discarded to prevent the models from exploiting absolute timestamps tied to the collection environment rather than intrinsic flow behavior.

After removing these fields, a binary class label was assigned to each flow instance, where Tor flows were labeled as 1 and Non-Tor flows as 0. To address potential class imbalance, both datasets were trimmed to contain an equal number of samples. This balancing step ensured that each class contributed equally during training and prevented the models from being biased toward the majority class.

Following class balancing, the Tor and Non-Tor flows were merged into a single dataset and randomly shuffled to eliminate any ordering bias introduced during the data collection process. Columns containing only zero values—typically originating from unused or non-applicable statistical fields—were automatically detected and removed, thereby reducing dimensionality and improving computational efficiency without sacrificing meaningful information.

To standardize feature scales, all remaining numerical attributes were normalized using the `StandardScaler`, which transforms each feature to have zero mean and unit variance. This normalization step is essential for ensuring that features with larger numeric ranges do not disproportionately influence distance-based or gradient-based learning algorithms.

The final cleaned, balanced, and normalized dataset served as the unified input for all machine-learning models evaluated in this research. These preprocessing steps ensured that the predictions generated by the classifiers were driven by genuine behavioral patterns in the traffic rather than contextual artifacts of the experimental setup.

To evaluate the effectiveness of machine-learning approaches for Tor traffic detection, our dataset was divided into training and testing sets using an 80/20 stratified split to preserve class balance. Three traditional models—Decision Tree, Random Forest, and XGBoost—were trained using scikit-learn and XGBoost libraries [38, 10]. These tree-based models are widely used for encrypted-traffic analysis due to their robustness on tabular data and ability to capture nonlinear feature interactions [40]. The Random Forest and XGBoost models also provide feature-importance scores, enabling interpretation of which flow-level statistics contribute most to Tor detection.

In addition to traditional models, three deep-learning architectures were trained using Keras/TensorFlow [11, 2]: a Multilayer Perceptron (MLP), a one-dimensional Convolutional Neural Network (CNN), and a Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN). The MLP consisted of several dense layers with batch normalization and dropout, while the CNN applied one-dimensional convolutions to capture local patterns among flow features. The

RNN model leveraged LSTM units to learn temporal dependencies in the feature representation. All neural networks were trained using the Adam optimizer with binary cross-entropy loss and early stopping to prevent overfitting.

For each model, evaluation metrics including accuracy, precision, recall, and F1-score were computed using the test set. This comparative analysis across six models—Decision Tree, Random Forest, XGBoost, MLP, CNN, and RNN—provides a comprehensive understanding of how traditional and deep-learning techniques perform on the proposed Tor vs. Non-Tor flow dataset.

4.2 Hyperparameter Settings

To ensure reproducibility, this section lists all hyperparameters used for the six classifiers evaluated in this study. The dataset was split using an 80/20 train-test split with stratified sampling to preserve the Tor and Non-Tor class ratio.

Decision Tree

The Decision Tree classifier was configured with the following parameters:

- Maximum depth: 10
- Criterion: Gini impurity
- Random state: 42

Random Forest

The Random Forest classifier was configured with the following parameters:

- Number of trees: 300
- Maximum depth: 20
- Class weighting: Balanced to address any dataset imbalance
- Random state: 42

XGBoost

The XGBoost classifier was configured with the following parameters:

- Number of trees: 300
- Maximum depth: 20
- Learning rate: 0.01
- Evaluation metric: Logarithmic Loss
- Random state: 42

Multilayer Perceptron (MLP)

The MLP configured with the following parameters:

- Hidden Layers: 3 (128, 256, 128 nodes)
- Activation Function: ReLU
- Regularization: Dropout layers with a rate of 0.3
- Optimization: Adam Optimizer

Convolutional Neural Network (CNN)

The CNN configured with the following parameters:

- Convolutional Layers: 2 (64 and 128 filters, kernel size = 1)
- Pooling: Max pooling with a pool size of 1
- Fully Connected Layers: 1 with 64 nodes
- Dropout Rate: 0.3

Recurrent Neural Network (RNN)

The RNN configured with the following parameters:

- LSTM Layer: 1 with 128 nodes

- Dropout: 0.3
- Fully Connected Layers: 1 with 256 nodes
- Optimization: Adam Optimizer

4.3 Experimental Results

The models were trained separately for different applications (i.e., web browsing, video streaming, file transfer, instant messaging, voice over IP, and video conferencing) and their accuracies were compared across sub-categories such as webpage type, video resolution, and file size. The detailed experimental results in terms of classification accuracy are presented as follows.

4.3.1 Web Browsing

Figure 4.1 presents the classification accuracy results for the web browsing scenario. The Wikipedia dataset achieved notably higher accuracies across all six models compared to the noise-free server dataset. For Wikipedia, traditional machine-learning models such as Decision Tree (94%), Random Forest (97%), and XGBoost (93%) demonstrated outstanding performance, while deep learning models including MLP (91%), CNN (90%), and RNN (90%) also performed strongly.

In contrast, the noise-free server dataset produced moderately lower accuracies across all models. Among these, XGBoost achieved the highest accuracy at 81%, followed by Random Forest (78%) and Decision Tree (78%). Deep learning models showed slightly reduced performance as well, with RNN (76%), MLP (75%), and CNN (72%) trailing behind. This performance reduction is primarily attributed to the uniform and repetitive traffic patterns produced by the controlled server, which provide fewer discriminative variations between Tor and Non-Tor flows.

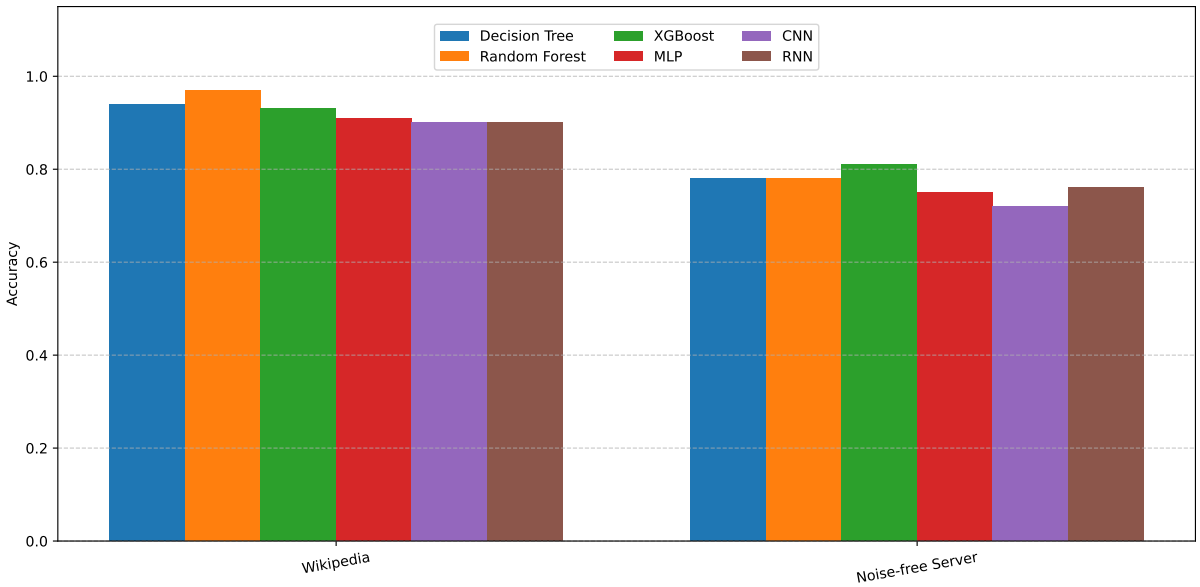

Figure 4.1: Web Browsing

4.3.2 Video Streaming

For video streaming, we considered two categories of datasets: YouTube streaming and noise-free server video streaming, each analyzed at four resolutions (240p, 480p, 1080p, and random). Across all datasets, traditional machine learning algorithms consistently outperformed deep learning-based models.

In detail, Figure 4.2 presents the classification accuracy results for video streaming via YouTube. The classification accuracies remained high across all resolutions. The Decision Tree (DT), Random Forest (RF), and XGBoost (XGB) models achieved the best overall performance at 240p, reaching 92% accuracy. At higher resolutions (480p–1080p), these models maintained accuracies between 86–90%, demonstrating strong generalization across varying video quality levels. Deep learning models such as RNN (80–87%) and MLP (79–87%) showed moderate performance, while CNN consistently performed the weakest, achieving only 66–69% accuracy, indicating its limitations in modeling the sequential temporal patterns inherent in flow-level data.

Fig. 4 shows the classification accuracy results for video streaming via the noise-free server. A similar pattern was observed, although accuracies were slightly lower at certain resolutions due to the uniform and controlled nature of the

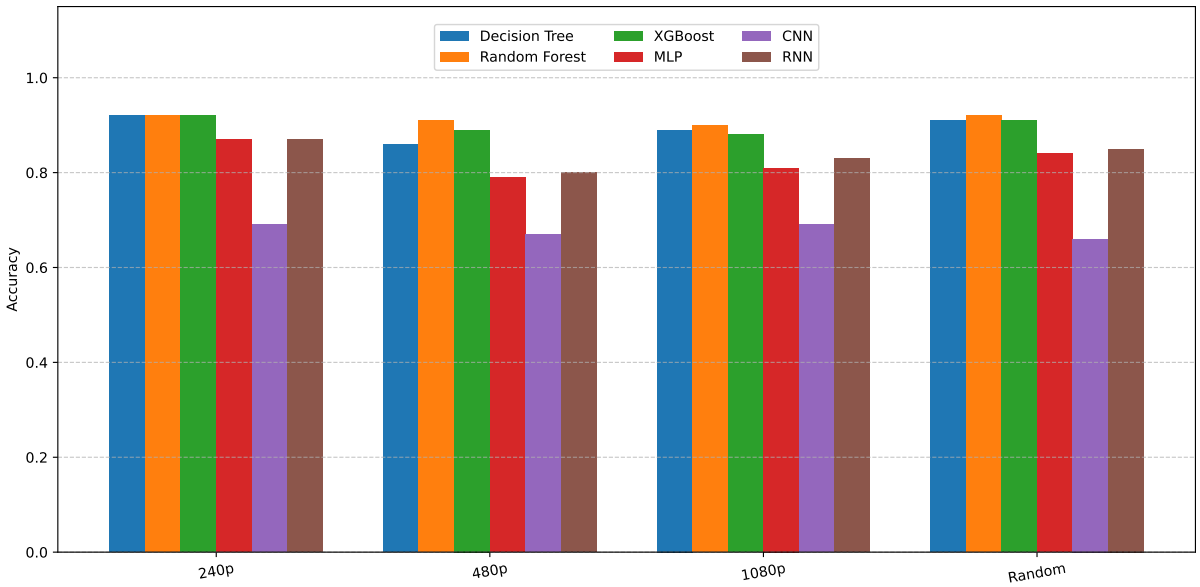

Figure 4.2: Video Streaming via Youtube

traffic. The highest accuracy was obtained for the random resolution, where Decision Tree (92%), Random Forest (92%), and XGBoost (91%) demonstrated excellent classification performance. At specific resolutions such as 480p and 1080p, traditional models achieved accuracies ranging from 84–87%, while deep learning models attained 72–80%. Again, CNN exhibited the weakest performance (67–71%), highlighting that convolution-based architectures are less effective for sequential flow data compared to tree-based or recurrent models.

Overall, ensemble-based algorithms—particularly Random Forest and XGBoost—consistently outperformed all other models across all datasets and all resolutions.

4.3.3 File Transfer

Figure 4.4 presents the classification accuracy results for file transfer. In this experiment, we analyzed four datasets corresponding to different file sizes: 10 MB, 50 MB, 100 MB, and a random combination of these sizes. Across all cases, traditional machine learning models consistently achieved higher accuracy compared to deep learning approaches, highlighting their robustness in modeling

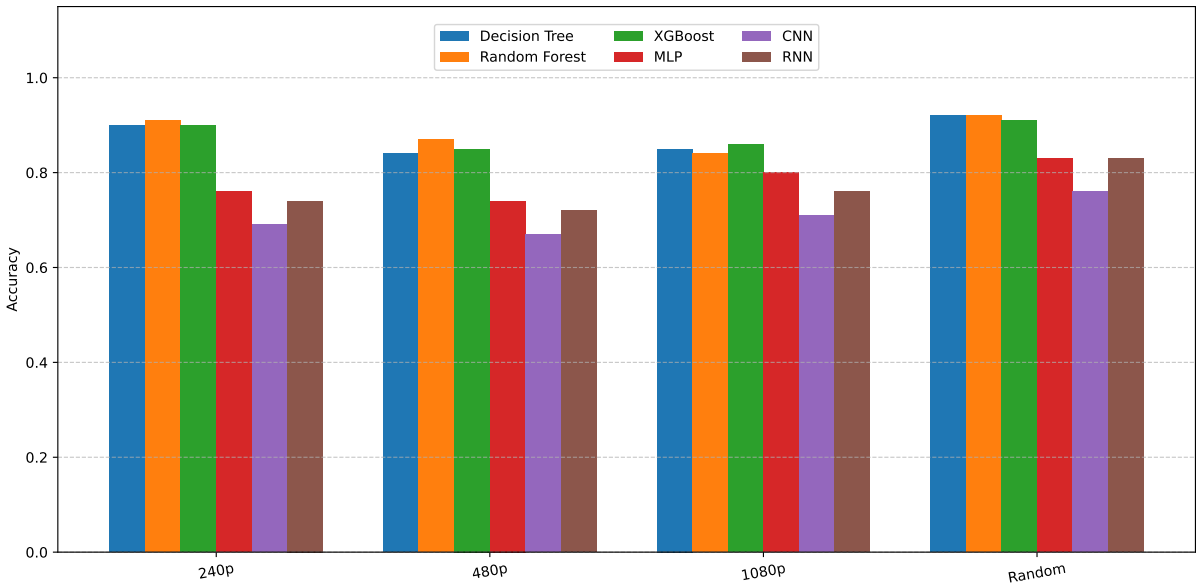

Figure 4.3: Video Streaming via Noise-free Server

flow-level statistical characteristics.

Specifically, for the 10 MB file transfer dataset, the highest accuracy was achieved by XGBoost (83%), followed closely by Decision Tree (82%) and Random Forest (82%). Deep learning models such as RNN (79%) and MLP (80%) exhibited moderate performance. When the file size increased to 50 MB, performance decreased slightly, with Random Forest (79%) and XGBoost (79%) maintaining the strongest accuracy, whereas deep models such as CNN (61%) and RNN (63%) performed notably worse.

For the 100 MB file transfers, traditional models again achieved the highest accuracy, with Decision Tree, Random Forest, and XGBoost each reaching 85%. In contrast, deep learning models—including MLP (77%), CNN (56%), and RNN (73%)—performed significantly lower, indicating that larger, longer-duration flows reduce temporal variability, making it more difficult for neural-network-based models to generalize. The random file transfer dataset produced stable results, with Decision Tree (87%), XGBoost (87%), and Random Forest (86%) yielding the strongest performance.

Overall, XGBoost consistently delivered the most reliable classification results across all file sizes, closely followed by Random Forest and Decision Tree. Deep

learning models underperformed in this scenario due to the homogeneous nature of file transfer traffic, which contains fewer temporal fluctuations compared to interactive applications. These findings indicate that tree-based ensemble models are particularly effective for identifying Tor vs. Non-Tor traffic during bulk data transfer activities.

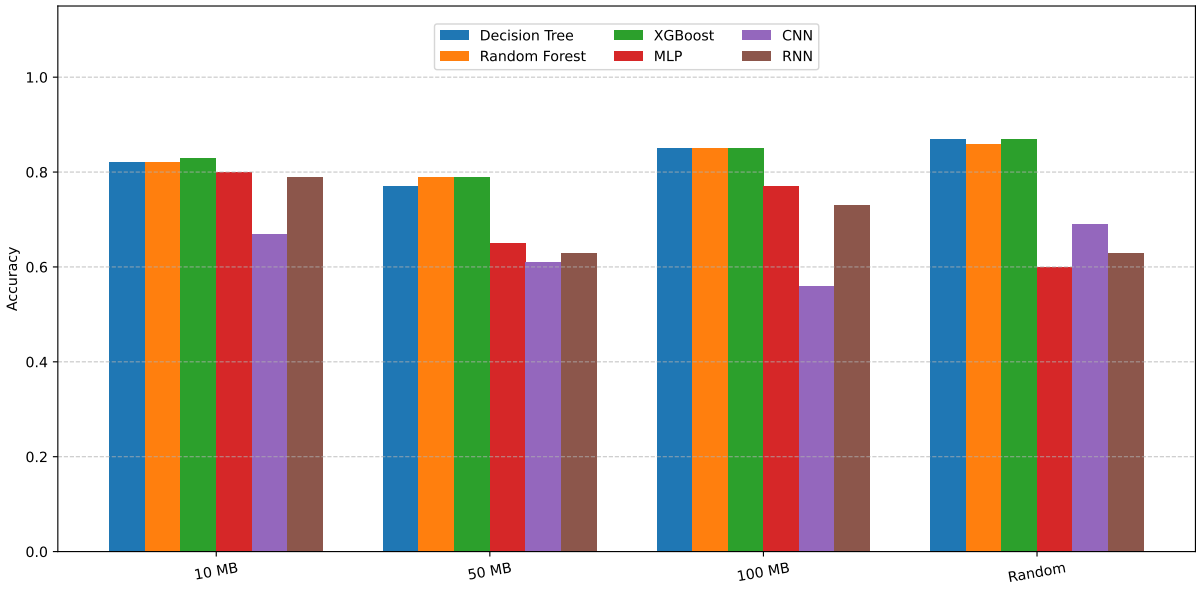

Figure 4.4: File Transfer via Noise-free Server

4.3.4 Instant Messaging, Voice over IP, and Video Conference

Figure 4.5 presents the classification accuracy results for instant messaging (IM), voice over IP (VoIP), and video conference (VC). Across all three applications, traditional machine learning models consistently outperformed deep learning architectures, demonstrating their effectiveness in capturing flow-level statistical features from interactive network sessions.

For instant messaging, the highest accuracy was achieved by Decision Tree (96%) and XGBoost (96%), followed closely by Random Forest (95%). Deep learning models such as RNN (86%) and MLP (88%) also performed competitively, whereas CNN (68%) showed substantially lower performance, suggesting that spatial convolutional filters are less suitable for short, bursty message exchanges.

In the case of voice over IP, which involves continuous packet transmission, classification accuracy remained high. Decision Tree (87%), Random Forest (84%), and XGBoost (87%) provided the strongest results. RNN (82%) and MLP (81%) demonstrated moderate performance, while CNN (77%) again lagged behind, indicating limited ability to capture temporal dynamics in voice-based sessions.

For video conference traffic, performance was similarly strong. Random Forest (93%), Decision Tree (90%), and XGBoost (90%) achieved the highest accuracies. Deep models such as MLP (88%) and RNN (82%) also performed reasonably well, whereas CNN (69%) exhibited the lowest performance due to the complex and irregular packet patterns characteristic of real-time video transmission.

Overall, Decision Tree, Random Forest, and XGBoost models demonstrated consistently superior performance across all applications, confirming the robustness of tree-based approaches for encrypted Tor traffic classification. Among deep learning models, RNN delivered the best results due to its ability to learn temporal dependencies, while CNN showed significant limitations for flow-level analysis.

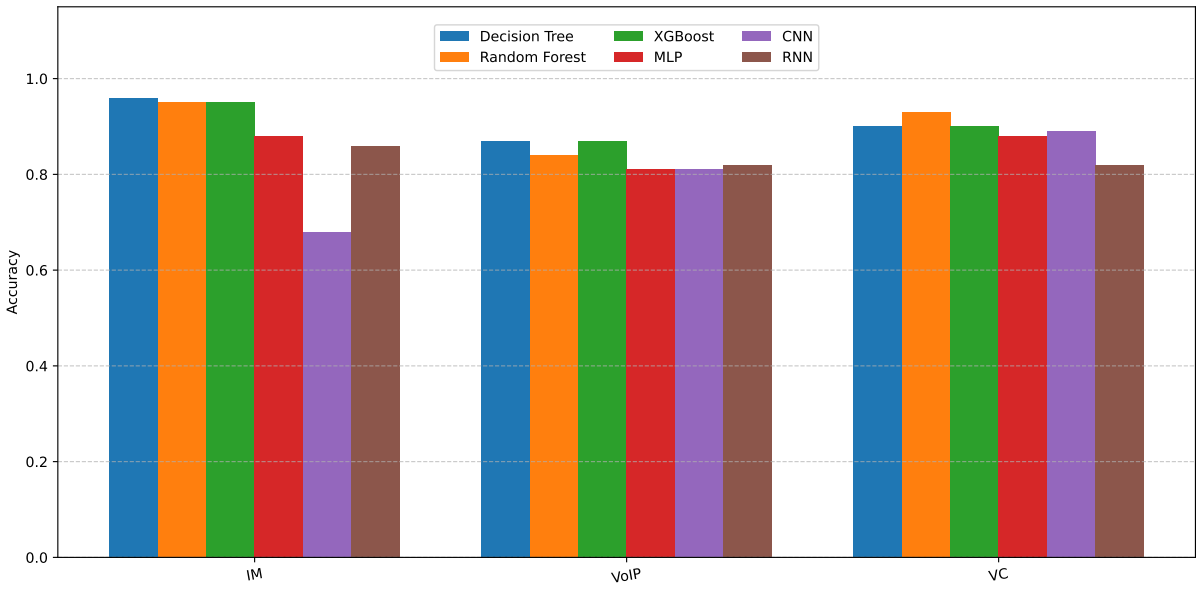

Figure 4.5: IM, VoIP, and VC

4.4 Discussion and Summary

The experimental results presented in this chapter demonstrate that traditional tree-based models—Decision Tree, Random Forest, and XGBoost—consistently outperform deep-learning architectures across all application categories and datasets. This pattern holds for web browsing, video streaming, file transfer, instant messaging, VoIP, and video conferencing, indicating that flow-level Tor detection is primarily a tabular learning problem where ensemble trees are particularly effective.

Several important trends emerge when comparing performance across applications. First, traffic scenarios with inherently rich and diverse behavior—such as Wikipedia browsing or YouTube streaming—yield the highest classification accuracies. In these cases, the natural variability in packet sizes, timing patterns, flow durations, and burstiness provides ample discriminative information for the models. Conversely, highly controlled or homogeneous traffic, such as fixed-size file transfers or noise-free videos encoded at a single resolution, yields slightly lower performance. This effect is especially noticeable in deep-learning models, which depend more heavily on temporal variation and struggle when flows offer limited structural diversity.

At this point, it is important to clarify the role of packet size in Tor detection. Although packet-size variation contributes to traffic diversity, packet size alone is not a reliable indicator of Tor activity. Tor internally uses fixed-size 512-byte cells, but these cells are transmitted inside TCP packets whose observed lengths vary significantly. Modern Internet traffic—especially QUIC-based applications, multimedia streaming, and interactive messaging—also exhibits wide packet-length distributions. Therefore, models do not rely on any simple packet-size fingerprint. Instead, the discriminative power comes from packet-size statistics such as mean, variance, interquartile range, and entropy. These metrics capture subtle behavioral differences introduced by Tor's batching, scheduling, and congestion-control mechanisms and help explain why packet-length features appear among the top contributors in the feature-importance analysis.

Second, the results highlight the sensitivity of deep-learning architectures to

the structure and representation of the input data. While MLP and RNN models achieve competitive performance in some scenarios, CNN-based approaches consistently lag behind. This suggests that simply applying convolutional layers over aggregated flow features is not sufficient to capture the sequential dynamics of network traffic. More sophisticated sequence modeling, alternative feature encodings, or raw packet-level inputs may be required to fully leverage deep learning in this domain.

Third, the similarity of results across Tor and Non-Tor datasets for different applications underscores the importance of a controlled experimental framework. Because the same user activities were performed under both routing modes, the models are compelled to focus on subtle behavioral differences introduced by Tor rather than on unrelated application-level artifacts. This provides stronger evidence that the classification performance observed in this thesis is due to genuine Tor-induced characteristics and not to confounding factors such as page content or file selection.

In summary, the findings of this chapter confirm that flow-level machine-learning models can reliably distinguish Tor from Non-Tor traffic in a realistic, controlled environment. At the same time, we highlight open challenges related to model robustness, generalization beyond the experimental testbed, and the impact of obfuscation or adversarial modification. These issues motivate the future work directions discussed in the final chapter of this thesis.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this research, we discussed a realistic dataset generation and flow-based framework for Tor traffic detection using both machine learning and deep learning techniques. A fully controlled testbed was developed inside Oracle VirtualBox, integrating a Debian Workstation and a Whonix Gateway to enable isolated and noise-free collection of Tor and Non-Tor network traffic. Automated traffic generation was performed using Selenium to emulate real-world user interactions across diverse applications, including web browsing, video streaming, file transfers, and real-time communication scenarios such as VoIP, video conferencing, and instant messaging.

The captured packet traces were transformed into flow-level datasets using a custom Scapy-based feature extraction pipeline that generated 48 statistical features per flow, reflecting temporal, directional, and volumetric characteristics of the traffic. Experimental results across multiple application categories demonstrated that traditional machine learning models—particularly Random Forest, XGBoost, and Decision Tree consistently outperformed deep learning architectures such as CNN and MLP. The strong performance of tree-based models highlights their robustness, interpretability, and suitability for encrypted traffic classification tasks.

Overall, our findings confirm that anonymized Tor traffic can be effectively distinguished from regular Internet traffic using only flow-level statistical features and supervised learning models, without requiring any packet payload inspection. The dataset and framework developed in this study serve as a reliable foundation for future research on encrypted traffic analysis, obfuscation techniques, and counter-classification strategies.

5.2 Future Work

Although the current dataset includes one-hour traffic sessions for each application scenario, future work will extend the data collection period to multiple hours or even multiple days. Longer and more diverse capture windows will introduce greater temporal variability, including diurnal usage patterns, background system processes, periodic congestion, and unpredictable network fluctuations. Incorporating these effects is essential for evaluating whether the trained classifiers remain stable when confronted with realistic, long-term network behaviors. Such extended data collection will also enable a deeper examination of temporal drift—that is, how statistical properties of Tor and Non-Tor traffic evolve over time—and whether models require periodic retraining or adaptive calibration to sustain high accuracy in operational deployments.

In addition, future research will explore advanced traffic obfuscation and adversarial manipulation strategies designed to stress-test and potentially evade state-of-the-art Tor classifiers. While this thesis conducted initial experiments on feature-targeted packet injection, more systematic evaluations are needed. Promising directions include (i) controlled timing perturbation to distort inter-arrival time signatures, (ii) dummy packet insertion to alter burstiness and volume-related features, (iii) flow-level morphing to reshape packet-length, jitter, and entropy distributions, and (iv) adversarial learning approaches that generate synthetic Tor flows crafted to mislead machine learning models. These investigations will offer deeper insight into which statistical features are most susceptible to manipulation and how classifiers can be reinforced to remain resilient against adaptive evasion techniques.

Future work will also evaluate next-generation deep learning architectures specifically designed for temporal sequence modeling. While this thesis examined traditional feedforward networks and recurrent neural networks, several powerful architectures remain unexplored. Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), 1D CNN–LSTM hybrids, and transformer-based models- such as Time-Series Transformers and Temporal Convolutional Transformers provide strong capabilities for capturing long-range dependencies

and subtle timing dynamics inherent in encrypted traffic. These architectures will be systematically analyzed to determine whether they offer measurable performance improvements, particularly under adversarial conditions or when trained on longer-duration traces. Furthermore, attention mechanisms may provide additional interpretability by highlighting critical traffic segments or feature patterns that influence classification decisions.

Another important direction for future research is the use of explainable machine learning (XML) techniques to better understand how classifiers distinguish Tor and Non-Tor traffic. While this thesis evaluates a wide range of models and demonstrates high classification accuracy, it does not yet analyze why the models make their predictions. Explainability tools such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) can provide insights into which flow-based features—such as traffic volume. directional traffic volume, packet length statistics, inter arrival time behavior,TCP flag patterns, and other statistical properties contribute most strongly to model decisions.

Applying explainability techniques would help validate whether models rely on stable, application-independent characteristics or on dataset-specific artifacts. It would also allow researchers to identify potentially biased features and gain deeper understanding of Tor's underlying behavioral patterns. Incorporating explainability into future work would therefore enhance transparency, improve trustworthiness, and support the development of more robust Tor detection systems.

Collectively, these extensions will strengthen the practical relevance of the proposed framework and contribute to the development of more robust, generalizable, and privacy-conscious mechanisms for Tor traffic detection. By evaluating both natural temporal drift and deliberate adversarial manipulation, future work will help ensure that encrypted traffic classification systems remain effective while maintaining resilience and respecting user privacy in increasingly adversarial network environments.

Bibliography

[1] CIC-Darknet2020 Dataset. https://www.unb.ca/cic/datasets/darknet2020.html. Canadian Institute for Cybersecurity, Accessed: 2025-11-20.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283. USENIX Association, 2016.

[3] Riyad Alshammari and A. Nur Zincir-Heywood. Machine learning based encrypted traffic classification: Identifying SSH and Skype. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.

[4] Riyad Alshammari and A. Nur Zincir-Heywood. Identification of voip encrypted traffic using a machine learning approach. *Journal of King Saud University - Computer and Information Sciences*, 27:77–92, 2015.

[5] Ibrahim A. Alwhbi, Cliff C. Zou, and Reem N. Alharbi. Encrypted network traffic analysis and classification utilizing machine learning. *Sensors (Basel)*, 24(11):3509, 2024.

[6] Philippe Biondi and the Scapy Community. Scapy: the python-based interactive packet manipulation program & library. GitHub repository, 2025. Accessed: December 15, 2025.

[7] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[8] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole, 1984.

[9] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 1981.

[10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[11] François Chollet et al. Keras. https://keras.io, 2015.

[12] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, 2004. USENIX Association.

[13] Anne Edmundson et al. Internet censorship in 2016: A review. In *FOCI*, 2016.

[14] Oliver Farnan et al. Active probing of tor: An empirical study. In *PETS*, 2016.

[15] Manuel Fernández-Delgado, Eva Cernadas, Sergio Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(90):3133–3181, 2014.

[16] David Fifield et al. Meek: Censorship circumvention using domain fronting. In *FOCI Workshop*, 2014.

[17] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015:46 – 64, 2015.

[18] Canadian Institute for Cybersecurity. Cicflowmeter: Network traffic flow generator. https://github.com/ahlashkari/CICFlowMeter, 2025. Accessed: 2025-11-21.

[19] Sebastián García, Martin Grill, Jan Stiborek, and Alejandro Zunino. An empirical comparison of botnet detection methods. *Comput. Secur.*, 45:100–123, 2014.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural Computation*, volume 9, pages 1735–1780, 1997.

[22] Elike Hodo, Noureddine Abouzakhar, Panagiotis I. Ijomah, Andrew Tachtatzis, and Xavier Bellekens. Machine learning approach for detection of nontor traffic. In *International Conference on Future Networks and Distributed Systems*, 2017.

[23] Nicholas Hopper. How much anonymity does network-level obfuscation provide? *ACM CCS*, 2010.

[24] Amir Houmansadr, Rob Jansen, et al. Snowflake: A webrtc-based censorship circumvention system. In *Network and Distributed System Security Symposium (NDSS)*, 2020.

[25] Rob Jansen et al. Shadow: Running tor in a box. In *NDSS*, 2014.

[26] Rob Jansen, Ryan Wails, and Aaron Johnson. A measurement of genuine tor traces for realistic website fingerprinting. *ArXiv*, abs/2404.07892, 2024.

[27] Aaron Johnson, Colin Wacek, and Paul Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *ACM CCS*, 2013.

[28] Thomas Karagiannis et al. Blinc: Multilevel traffic classification in the dark. In *ACM SIGCOMM*, 2005.

[29] Hyunchul Kim, KC Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, New York, NY, USA, 2008. Association for Computing Machinery.

[30] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Mamun, and Ali A. Ghorbani. Characterization of tor traffic using time based features. *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*, 2017.

[31] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, 24(3):1999–2012, 2020.

[32] Navid Malekghaini, Elham Akbari, Mohammad A. Salahuddin, Noura Limam, Raouf Boutaba, Bertrand Mathieu, Stéphane Moteau, and Stéphane Tuffin. Deep learning for encrypted traffic classification in the face of data drift: An empirical study. *Computer Networks*, 225:109648, 2023.

[33] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the tor network. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[34] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[35] Andrew Moore and Denis Zuev. Internet traffic classification using bayesian analysis. In *ACM SIGMETRICS*, 2005.

[36] M. H. Pathmaperuma et al. Application of deep learning on the characterization of tor traffic. *Journal of Internet Services and Information Security*, 2021.

[37] M. H. Pathmaperuma et al. Deep learning for encrypted traffic classification and intrusion detection: A review. *Sensors*, 22, 2022.

[38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[39] Shahbaz Rezaei and Xin Liu. Deep learning for encrypted traffic classification: An overview. *arXiv preprint arXiv:1810.07906*, 2018.

[40] Shahbaz Rezaei and Xin Liu. Deep learning for encrypted traffic classification: An overview. *IEEE Communications Surveys & Tutorials*, 22(3):1725–1771, 2020.

[41] N. Rust-Nguyen et al. Darknet traffic classification and adversarial attacks using machine learning. *Computers & Security*, 2023.

[42] Md Rafiqul Islam Sadik. Tor dataset generation. https://www.kaggle.com/datasets/rafiqulislamsadik/tor-dataset-generation, 2024. Accessed: 2025-11-20.

[43] Selenium Project. Selenium webdriver. https://www.selenium.dev/, 2025. Accessed: 2025-11-21.

[44] Khalid Shahbar and Ayse Nur Zincir-Heywood. Anon 17 : Network traffic dataset of anonymity services. 2017.

[45] The Tor Project. Tor project: Anonymity online. https://www.torproject.org/, 2025. Accessed: 2025-11-21.

[46] Whonix Project. Whonix design documentation. https://www.whonix.org/wiki/Design, 2023. Accessed: 2025-11-24.

[47] Whonix Project. Whonix documentation: Security architecture. https://www.whonix.org/wiki/About, 2023. Accessed: 2025-11-24.

[48] Nigel Williams et al. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. In *ACM SIGCOMM*, 2006.

[49] Philipp Winter and Stefan Lindskog. Detecting and circumventing the great firewall of china. In *USENIX FOCI*, 2012.

[50] Philipp Winter and Stefan Lindskog. Scramblesuit: A polymorph network protocol to circumvent censorship. In *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society (WPES)*, 2013.

[51] Wireshark Foundation. Wireshark. https://www.wireshark.org/, 2025. Official website. Accessed: 2025-11-21.

Appendix A

Detailed Tor Traffic Detection Results

Table A.1: Web Browsing (Noise-Free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.78	.78	.78	.78
Random Forest	.78	.79	.78	.78
XGBoost	.81	.82	.81	.81
Multilayer Perceptron	.75	.75	.75	.75
CNN	.72	.72	.72	.72
RNN	.76	.76	.76	.76

Table A.2: Web Browsing (Wikipedia)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.94	.94	.94	.94
Random Forest	.97	.97	.97	.97
XGBoost	.93	.94	.93	.93
Multilayer Perceptron	.91	.91	.91	.91
CNN	.90	.90	.90	.90
RNN	.90	.90	.90	.90

Table A.3: Video Streaming 240P (YouTube)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.92	.92	.92	.92
Random Forest	.92	.92	.92	.92
XGBoost	.92	.92	.92	.92
Multilayer Perceptron	.87	.87	.87	.87
CNN	.69	.69	.69	.69
RNN	.87	.87	.87	.87

Table A.4: Video Streaming 480P (YouTube)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.86	.88	.86	.86
Random Forest	.91	.91	.91	.91
XGBoost	.89	.90	.89	.89
Multilayer Perceptron	.79	.79	.79	.79
CNN	.67	.67	.67	.67
RNN	.80	.80	.80	.80

Table A.5: Video Streaming 1080P (YouTube)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.89	.90	.89	.89
Random Forest	.90	.91	.90	.90
XGBoost	.88	.89	.88	.88
Multilayer Perceptron	.81	.81	.81	.81
CNN	.69	.69	.69	.69
RNN	.83	.83	.83	.83

Table A.6: Video Streaming Random (YouTube)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.91	.91	.91	.91
Random Forest	.92	.92	.92	.92
XGBoost	.91	.91	.91	.91
Multilayer Perceptron	.84	.84	.84	.84
CNN	.66	.66	.66	.66
RNN	.85	.85	.85	.85

Table A.7: Video Streaming 240p (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.90	.90	.90	.90
Random Forest	.91	.91	.91	.91
XGBoost	.90	.90	.90	.90
Multilayer Perceptron	.76	.76	.76	.76
CNN	.69	.69	.69	.69
RNN	.74	.74	.74	.74

Table A.8: Video Streaming 480p (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.84	.84	.84	.84
Random Forest	.87	.87	.87	.87
XGBoost	.85	.85	.85	.85
Multilayer Perceptron	.74	.74	.74	.74
CNN	.68	.68	.67	.67
RNN	.72	.72	.72	.72

Table A.9: Video Streaming 1080p (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.85	.86	.85	.85
Random Forest	.84	.85	.84	.84
XGBoost	.86	.87	.86	.86
Multilayer Perceptron	.80	.80	.80	.80
CNN	.71	.71	.71	.71
RNN	.76	.76	.76	.76

Table A.10: Video Streaming Random (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.92	.92	.92	.92
Random Forest	.92	.92	.92	.92
XGBoost	.91	.91	.91	.91
Multilayer Perceptron	.83	.83	.83	.83
CNN	.76	.76	.76	.76
RNN	.83	.83	.83	.83

Table A.11: File Transfer 10MB (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.82	.84	.82	.82
Random Forest	.82	.83	.82	.81
XGBoost	.83	.85	.83	.83
Multilayer Perceptron	.80	.80	.80	.80
CNN	.67	.67	.67	.67
RNN	.79	.79	.79	.79

Table A.12: File Transfer 50MB (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.77	.80	.77	.76
Random Forest	.79	.81	.79	.78
XGBoost	.79	.83	.79	.79
Multilayer Perceptron	.65	.65	.65	.65
CNN	.61	.61	.61	.61
RNN	.63	.63	.63	.63

Table A.13: File Transfer 100MB (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.85	.86	.85	.85
Random Forest	.85	.86	.85	.85
XGBoost	.85	.86	.85	.85
Multilayer Perceptron	.77	.77	.77	.77
CNN	.56	.56	.56	.56
RNN	.73	.73	.73	.73

Table A.14: File Transfer Random (Noise-free Server)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.87	.87	.87	.87
Random Forest	.86	.86	.86	.86
XGBoost	.87	.87	.87	.87
Multilayer Perceptron	.60	.60	.60	.60
CNN	.69	.69	.69	.69
RNN	.63	.63	.63	.63

Table A.15: Instant Messaging (IM)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.96	.97	.96	.96
Random Forest	.95	.95	.95	.95
XGBoost	.96	.97	.96	.96
Multilayer Perceptron	.88	.88	.88	.88
CNN	.68	.69	.69	.69
RNN	.86	.85	.85	.85

Table A.16: VoIP

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.87	.89	.87	.87
Random Forest	.84	.87	.84	.83
XGBoost	.87	.88	.87	.87
Multilayer Perceptron	.81	.81	.81	.81
CNN	.77	.81	.81	.81
RNN	.82	.82	.82	.82

Table A.17: Video Conferencing (VC)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	.90	.90	.90	.90
Random Forest	.93	.93	.93	.93
XGBoost	.90	.90	.90	.90
Multilayer Perceptron	.88	.88	.88	.88
CNN	.69	.89	.89	.89
RNN	.82	.82	.82	.82