DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

AI-Powered Network Traffic Prediction

AMIN BOLAKHRIF

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

AI-Powered Network Traffic Prediction

AMIN BOLAKHRIF

Master's Programme, Machine Learning, 120 credits
Date: August 18, 2021
Academic Supervisor: Mustafa Özger
Industrial Supervisors: David Sandberg, Ezeddin Al Hakim
Examiner: Cicek Cavdar
School of Electrical Engineering and Computer Science
Host company: Ericsson
Swedish title: AI baserad prediktering av nätverkstraffik

AI-Powered Network Traffic Prediction / AI baserad prediktering av nätverkstraffik

Abstract

In this Internet and big data era, resource management has become a crucial task to ensure the quality of service for users in modern wireless networks. Accurate and rapid Internet traffic data is essential for many applications in computer networking to enable high networking performance. Such applications facilitate admission control, congestion control, anomaly detection, and bandwidth allocation. In radio networks, these mechanisms are typically handled by features such as Carrier Aggregation, Inter-Frequency Handover, and Predictive Scheduling. Since these mechanisms often take time and cost radio resources, it is desirable to only enable them for users expected to gain from them.

The problem of network traffic flow prediction is forecasting aspects of an ongoing traffic flow to mobilize networking mechanisms that ensures both user experience quality and resource management. The expected size of an active traffic flow, its expected duration, and the anticipated amount of packets within the flow are some of the aspects. Additionally, forecasting individual packet sizes and arrival times can also be beneficial. The wide-spread availability of Internet flow data allows machine learning algorithms to learn the complex relationships in network traffic and form models capable of forecasting traffic flows.

This study proposes a deep-learning-based flow prediction method, established using a residual neural network (ResNet) for regression. The proposed model architecture demonstrates the ability to accurately predict the packet count, size, and duration of flows using only the information available at the arrival of the first packet. Additionally, the proposed method manages to outperform traditional machine learning methods such as linear regression and decision trees, in addition to conventional deep neural networks. The results indicate that the proposed method is able to predict the general magnitude of flows with high accuracy, providing precise magnitude classifications.

Keywords

Deep Learning, IP Address, Machine Learning, Network Traffic Flows, Neural Networks, Predictive Model, Regression.

Sammanfattning

I denna Internet och data era har resurshantering blivit allt mer avgörande för att säkerställa tjänstekvaliteten för användare i moderna trådlösa nätverk. Noggrann och hastig Internet-trafikinformation är avgörande för många applikationer inom datanätverk för att möjliggöra hög nätverksprestanda. Sådana applikationer underlättar kontroll av behörighet, kontroller av trängsel, detektering av avvikelser och allokering av bandbredd. I radionätverk hanteras dessa mekanismer vanligtvis av funktioner som Carrier Aggregation, Inter-Frequency Handover och Predictive Scheduling. Eftersom dessa funktioner ofta tar tid och kostar resurser så är det önskvärt att nätverk endast möjliggör sådana funktioner för användare som förväntas dra nytta av dem.

Prediktering av trafikflöden i nätverk grundar sig i att förutsäga aspekter av ett pågående trafikflöde för att kunna mobilisera nätverksfunktioner som säkerställer både kvaliteten för användare samt resurshantering. Den förväntade storleken på ett aktivt trafikflöde, dess varaktighet och mängden paket inom flödet är några av dessa aspekter. Det kan dessutom vara fördelaktigt att förutsäga individuella paketstorlekar och ankomsttider. Den stora tillgången till data med nätverks-flöden gör det möjligt för maskininlärningsmetoder att lära sig de komplexa förhållandena i nätverkstrafik och därigenom formulera modeller som kan förutsäga flöden i nätverk.

Denna studie föreslår en djupinlärningsbaserad metod för att prediktera flöden i nätverk, med hjälp av ett anpassat neuralt nätverk som utnyttjar genvägar i modellens konstruktion (ResNet). Den föreslagna modell-arkitekturen visar sig nöjaktigt kunna förutsäga antalet paket, storlek och varaktighet för flöden med endast den information som är tillgänglig från det första paketet. Dessutom lyckas den föreslagna metoden att överträffa både traditionella maskininlärningsmetoder som linjär regression och beslutsträd, samt konventionella djupa neurala nätverk. Resultaten indikerar att den föreslagna metoden kan förutsäga den allmänna storleken på flödens egenskaper med hög noggrannhet, givet att IP-adresser är tillgängliga.

Nyckelord

Djupinlärning, IP Address, Maskininlärning, Nätverkstrafikflöden, Neurala Nätverk, Prediktionsmodell, Regression.

Acknowledgments

I would like to express my deepest gratitude to my supervisors David Sandberg, Ezeddin Al Hakim, and Mustafa Özger, for their patience, guidance, and support. The completion of this thesis would not have been possible without their immense encouragement and supervision. I would also like to thank Ericsson for providing me with a valuable opportunity to learn more about how to conduct research, in addition to providing me real-world data and computational resources enabling a comprehensive work. Also, I am thankful to my examiner Cicek Cavdar for the valuable feedback on this degree project.

As this thesis marks the end of my master's degree at KTH, I would like to acknowledge my family and friends who supported me throughout this academic journey and who kept me striving for excellence. Without your sacrifice, this journey would not have been possible.

Stockholm, August 2021
Amin Bolakhrif

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

ANN Artificial Neural Network

BPTT Backpropagation through time

DBN Deep Belief Network

DNS Domain Name System

FNN Feedforward Neural Network

HTTP Hypertext Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

IEFT Internet Engineering Task Force

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISP Internet Service Provider

LAN Local Area Network

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MC Monte Carlo

MdAE Median Absolute Error

MLP Multilayer Perceptron

MPI Mean Prediction Interval

MSE Mean Squared Error

MVE Mean Variance Estimation

PICP Prediction Interval Coverage Probability

ReLU Rectified Linear Unit

ResNet Residual Neural Network

RFC Request For Comment

RNN Recurrent Neural Network

SMAPE Symmetric Mean Absolute Percentage Error

SVM Support Vector Machine

SVR Support Vector Regressor

TCP Transmission Control Protocol

UDP User Datagram Protocol

URL Uniform Resource Locator

Chapter 1

Introduction

This introductory chapter aims to provide the reader with some background on the relevance of network traffic predictions, the purpose of the thesis, and its objective. The limitations of the thesis and some sustainability and ethical aspects will also be discussed in this chapter.

1.1 Background

In the past decade, the Internet and the traffic within it have remarkably grown due to the competitive nature of the technology industry that have enabled the Internet widely accessible for a significant fraction of the world's population [1]. Consequently, Internet providers now face the difficult task of providing good service quality to their users despite the growing traffic demands. Being able to utilize networking equipment efficiently has therefore become a crucial task in order to maintain service quality. Accurate network traffic predictions could become a solution that helps automate and manage resources to ensure quality of service (QoS) [2] along with energy savings by turning off processors during low traffic times [1]. By keenly provisioning bandwidth, more fair resource allocation can be achieved while also maintaining adequate network performance. Due to the dynamic nature of network traffic and the considerable risks of network congestions, it is essential to robustly model and forecast accurate network traffic flows as wrong predictions can result in severe consequences in model-based decision-making systems. Network traffic predictions can also prove valuable in network traffic planning, which has become a crucial task in the computer networking research community, as accurate traffic predictions are needed to help network administrators and engineers deal with unexpected conditions [3].

1.2 Problem

The problem of traffic flow prediction is to predict aspects of network flows, such as, the total size of the flow, the time the flow is expected to be active, the number of packets within the flow, and even the size and arrival times of individual packets. The information used to infer the target quantities possibly resides in the data available in the first packet of the flow; the data includes the two communicating devices' IP addresses, ports, and the transport-layer protocol used. The traffic prediction problem is similar to the traffic classification problem that has been studied and reviewed in [4]. However, one major disadvantage with most traffic classification algorithms is that they require manual labeling of flows, which can be very time-consuming and costly. Traffic prediction is an important problem in modern wireless networks with features that requires some facilitation procedure before any gain can be observed. Examples of such radio network features are Carrier Aggregation (where radio transmission takes place over a wider bandwidth), Inter-Frequency Handover (where mobile terminals are handed over to cells with higher capacity), and Predictive Scheduling (where individual packets are predicted and grants are sent to the terminal proactively). These features often takes time and costs radio resources, it is therefore desirable that the network only enables the feature for users that are expected to gain from it.

While the subject of network traffic prediction has been investigated through many researchers' use of time series models [2, 3, 5, 6], the problem is still left partially unsolved as the proposed time series models require an observation period of the flow before meaningful predictions regarding the flow can be generated. The fact that the majority of Internet traffic flows consist of relatively few packets, of count 10 or less [7, 8], implies that most traffic flows can not be accurately predicted due to their short sequence length.

1.3 Research Question

The formal research question of the thesis is defined as follows:

"How can a network traffic flow be accurately predicted using only information found in the first packet header?".

The packet count, total size, and duration of individual flows are the primary properties sought to be predicted. The information available in the first packet corresponds to the two communicating systems' IP addresses, the ports, and the utilized transport-layer protocol.

While Section 1.2 defines the ambition and the direction of the study, particularizing the main intention of the investigation into smaller specific objectives and definite questions allows for broader comprehension along with explicit formulations. The sub-questions that this thesis aims to clarify are

1. Can IP addresses be used to make predictions without a warm-up period?
2. Can packets with anonymized/excluded IP addresses (due to privacy concerns) be used to make predictions about the magnitude of an incoming traffic flow?
3. Can the predictor learn the uncertainty of predicted quantities?

Each of these objective questions will be approached in an appropriate setup of experiments presented in Chapter 4.

1.4 Goals

This thesis work aims to investigate if parts of the information contained in an IP packet header (source IP address, source port, destination IP address, destination port, and transport-layer protocol) can be used to accurately predict the total packet count, size, and duration of a network traffic flow, using machine learning methods.

The main research goal of this thesis is to design an accurate network traffic predictor that removes or significantly decreases the required warm-up period seen in other proposed methods [2, 3]. The warm-up period refers to an arbitrary

time interval where the predictor only observes the packets within a given flow before useful predictions are made. An additional research goal is that the proposed predictor outperforms existing baselines such as naive mean/mode predictors on the defined performance indicators described in Section 4.5.

1.5 Scope and Limitations

This thesis mainly focuses on investigating machine learning model architectures and developing possible methods to predict the total packet count, size, and duration of traffic flows, including estimated uncertainties, using only the first packet of flows. This thesis does not address decision interpretability, model integration within the networks, nor predictions for individual packets. Additionally, the thesis is limited by user privacy and sensitivity regulations to datasets that scientists at various public research institutions have carefully collected. The aimed attention of the thesis is on the possible improvement of the overall performance accuracy of proposed model architectures and training methods.

1.6 Research Methodology

The theory, experiments, and evaluations of the thesis work are based on previously published and publicly available research papers within network traffic prediction and machine learning methods. By following an empirical research method, various algorithms within the field of machine learning are tested and evaluated on publicly available real-world network traffic datasets provided by public research institutions and classified real-world datasets provided by Ericsson. Using traditional evaluation metrics, the proposed algorithms and feature processing techniques are assessed quantitatively and compared with each other and predefined baselines. Based on the quantitative evaluations, the goal of the thesis work is to find suitable machine learning algorithms that satisfy the answer to the research question and the objectives in Section 1.4.

1.7 Sustainability and Ethics

The implementation of accurate network traffic prediction models should result in more optimized and effective network infrastructures enabling network traffic planners to carefully plan and allocate resources that otherwise would have been wasted on a process that does not require them. The ability to provision resources such as bandwidth and control expensive radio features with high confidence should decrease the required operational energy. Energy efficiency is closely related to the 12^{th} goal of the 2030 Sustainable Development Goals (SDG) [9] which advocates for doing more and better with fewer resources, ensuring sustainable consumption and production.

As the methods proposed in the thesis work require personal data that may identify individual users and their private Internet activity, it is important to take the General Data Protection Regulation (GDPR) into account whenever the methods are applied to real-world cases. The operators of such applications should maintain strict policies on the entities that can view and use the data. Additionally, the collected data should be encrypted, stored securely, and available to the corresponding consumer to preserve the ethical values of the methods. The required act of monitoring users' Internet traffic should be explicitly stated in the Terms of Service (ToS) of the application and conform to the country's data protection legislation in which the application is used in. While parts of the thesis work are conducted on publically available and anonymized datasets, it is also evaluated using strictly confidential data provided by Ericsson, which can not be shared externally for reproduction purposes.

1.8 Contributions

The contributions of this thesis work are summarized below:

- The work provides a generic deep learning architecture useful for predicting the magnitude of incoming network traffic flows.
- The work proposes a set of features and a pre-processing method useful for predicting the magnitude of network traffic flows.
- The work provides an analysis of predictions made on several variations of network data.
- The work provides an in-depth analysis of the proposed deep learning architecture and its characteristics.

1.9 Outline

In Chapter 2, a general overview of computer networking and the main traditional machine learning methods involved in the work is given. Chapter 3 discusses relevant an related work at the intersection of machine learning and network traffic. The proposed model architectures, datasets, data processing methods, experimental design, and evaluation framework is discussed in great detail in Chapter 4. Analysis of data and experimental results are reported in Chapter 5. In Chapter 6, an in-depth discussion of the major results is presented. Finally, Chapter 7 discusses the main conclusions drawn from the thesis work and suggests improvements for future work.

Chapter 2

Background

The following chapter introduces the reader to the core concepts of computer networking and the background of which the thesis work will be carried out within. Additionally, the theoretical background on machine learning methods used throughout the thesis will also be explained.

2.1 Computer Networking

The broad term Internet commonly refers to the extensive network of interconnected computing devices across the world. Traditionally, these devices mainly consisted of servers, workstations, and desktop computers whose purpose was to transmit and receive e-mail messages and files such as Web pages. As technology has evolved, new devices with embedded software and sensors connect with other devices and systems to exchange data over the Internet. Additionally, today's Internet is being used for a much broader set of matters such as smartwatches, public transport systems, and even home appliances. The Internet has even been estimated to be used by 75% of the world's population via mobile devices by 2025 [10]. With the expansion of nontraditional devices accessing the Internet, user devices are modernly referred to as end systems or hosts in Internet jargon. Figure 2.1 illustrates some of the pieces that make up the Internet.

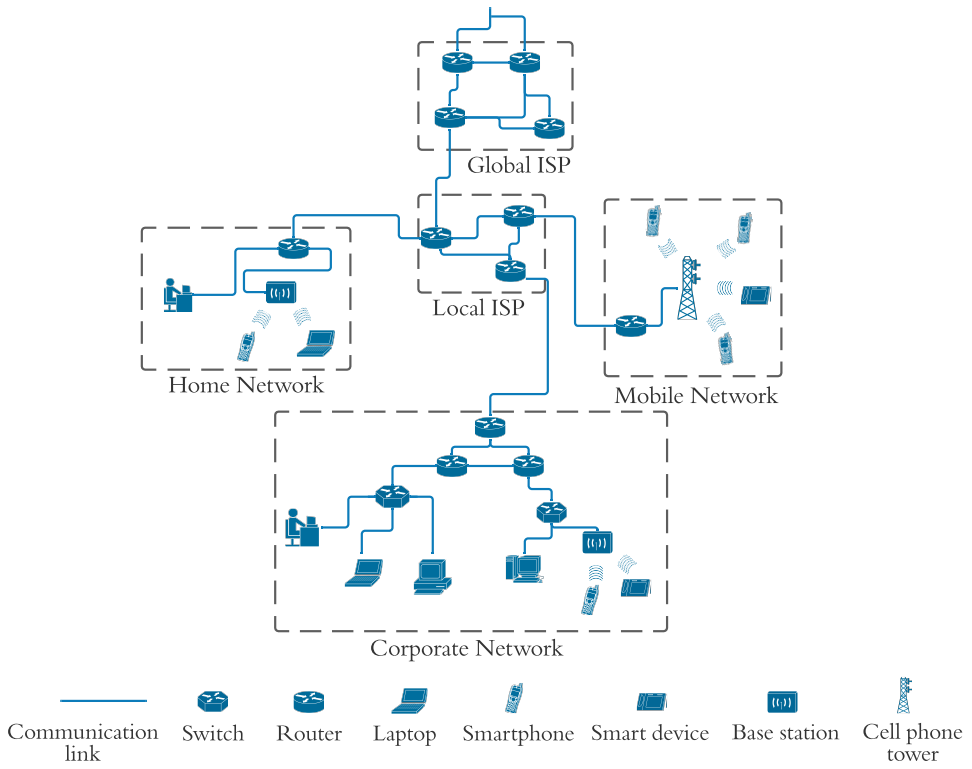


Figure 2.1 – Segments of the Internet and their relationship to each other.

2.1.1 End Systems

End systems in the context of computer networking often allude to computing devices connected to the Internet since they are located at the edge of the Internet architecture. The edge points generally encompass desktop computers, servers, and mobile devices that are also interchangeably called hosts. Their primary function is to run software applications that enable data exchange between devices and large data centers. By dividing the host class into the categories clients and servers, we can also distinguish more powerful machines that store and distribute (servers), from consumer devices (clients), making the separation clearer.

2.1.2 Internet Service Provides

The umbrella term Internet Service Providers (ISPs) refers to various systems that enable access to the Internet. Most commonly, an ISP is recognized as a

telephone company that provides Internet access in the form of WiFi or cable to residential areas, or a cellular data ISPs that provide mobile Internet access to smartphones. However, it is essential to note that there is an array of diverse ISPs, including university ISPs and corporate ISPs. The shared property of ISPs is enabling end systems to access the Internet; therefore, it is important for them to be interconnected with each other. A hierarchical architecture thus exists in the realm of ISPs where lower-tier ISPs are linked together by national tier ISPs and further connected through upper-tier ISPs that are all interconnected. ISPs work by delegating traffic with a network of packet switches and physical communication links that enable the transfer of the data. These communication links all have different transmission rates depending on their physical properties, where the fastest link is typically the renowned optic fiber. Still, there exist various other slower links, including copper wires and coaxial cables. When one system sends data to another system, the data is segmented into smaller pieces specified as packets in the field of computer networking. Each packet is joined by a header that describes the meta-information used by the receiving system to route or reassemble the packets into the original data with packet switches. These switches work by receiving and forwarding packets through the incoming and outgoing communication links, similar to a post office operation whenever a post or a letter is sent. There is a wide variety of packet switches; the most common ones are the link-layer switches and the routers. The latter is generally used in the network core, while link-layer switches often times are used in access networks. Routes and paths are interchangeable terms describing the sequence of communication links and packet switches that a packet traverses through from its origin system to the destination system.

2.1.3 Protocols

In order to enable interoperability between end systems, packet switches, and communication links, we resort to protocols to control the information flow. Two of the most prominent protocols utilized for this are the Internet Protocol (IP) and the Transmission Control Protocol. Together they form the principal protocols recognized as TCP/IP, which plays a vital role for systems to communicate with each other. The Internet standards are essential to ensure that each protocol's scope is unanimously agreed upon and constructed by the Internet Engineering Task Force (IETF). For the Web, the most notable protocol is HTTP; correspondingly, for e-mail services, SMTP is used. These protocols are collectively defined and governed by the IETF with documents that specify

the standards called requests for comments (RFCs), of which there currently exists just over 9000 [11]. Worthy to note is that other governing bodies exist that lay the standards for different network components; the Ethernet and WiFi standards, for instance, are ruled by the IEEE 802 LAN Standards Committee [12]. Protocols manage all forms of activity between two or more communicating systems on the Internet. These protocols resemble human behavior protocols (assuming they are well-mannered) and allow hardware or software components to follow a diligent chain of conduct to take actions and exchange information. In essence, a protocol states the configuration and the order in which information is exchanged between two systems in addition to the possible actions that can be taken by the recipient and/or sender system. Underneath the surface, we interact with network protocols daily. Say for instance, we want to visit a website, we type in a Uniform Resource Locator (URL) to the address bar on a web browser and press the enter key. The computing device (desktop, smartphone, tablet) thereafter sends a connection request message to the Web server represented by the URL and awaits a web server response. Eventually, the Web server returns a response to the request in the form of a connection reply message. The computing device subsequently sends another message (commonly a GET request in HTTP) to request the actual Web page (file), after which the Web server returns the file to the computing device, and we can continue our Web browsing. As previously mentioned, one example of packet switchers is routers. Routers enable activity between two or more communicating systems and also operate under protocols. More specifically, their protocols allow the packet route between the source and the destination to be determined.

2.1.4 Packet Switches

Circuit switching and packet switching are two central methods that enable information to flow through a network of communication links and switches. This thesis will mainly target packet switching and thus not delve into circuit switching. Whenever end systems interchange information, in the form of text, images, audio, or any other type of media, the source end system's information is broken down into packets that travel through both packet switches and communication links to reach the destination end system for reassembly. The transmission rate R (bits/sec) of the communication link is determined by its physical properties. Theoretically, it would take L/R seconds for a message of size L bits to travel from a source end system to the destination

end system. As packets usually contain more information than a few bits, it would be problematic for the destination end system to receive individual bits that are part of a packet. Store-and-forward transmission solves this problem and is implemented in most packet switches at the input to the communication link. Store-and-forward transmission ensures that the switch must first receive the packet's entire data content before the packet switch can transmit any bit of information to the outbound link [12]. Generally, an output queue/buffer is present inside packet switches that allow packet storage for each attached outgoing communication link, enabling the router to store parts of a packet outbound for a specific link. As a consequence of store-and-forward and buffers, delays arise in the packet switch's final transmission rate, having to wait for entire packets to arrive and from packets waiting in the output buffer in a busy link. Related to buffers are packet losses, which generally occurs whenever the finite buffer space is full, and new outbound packets enter the queue. Depending on how the system is designed for network congestion, the newly arriving packet or another packet in the output buffer will be dropped and lost [12].

2.1.5 TCP/IP Protocol Architecture

To understand the flow of Internet traffic, we must first grasp how computer networking is organized and the architecture behind it. The TCP/IP architecture is a layered architecture that simplifies a complicated system into smaller, more manageable components. This essential modular nature of the architecture adds a level of abstraction and simplifies implementational adjustments of selected layers in a black box fashion as long as the service from and to the neighboring layers remains the same. The advantage of layering is particularly important for the architecture, where other components must provide similar services after a module has been iterated on. The TCP/IP architecture consists of five layers in which the collective protocols in the various layers are termed the protocol stack, see Figure 2.2. The five layers in the architecture include the physical, link, network, transport, and application layers [12].

Application Layer

On top of the TCP/IP Protocol Architecture lies the application layer, which contains the application protocols that enable applications to communicate with each other. The previously mentioned HTTP and the SMTP protocols and various other protocols such as the domain name system (DNS) and the file

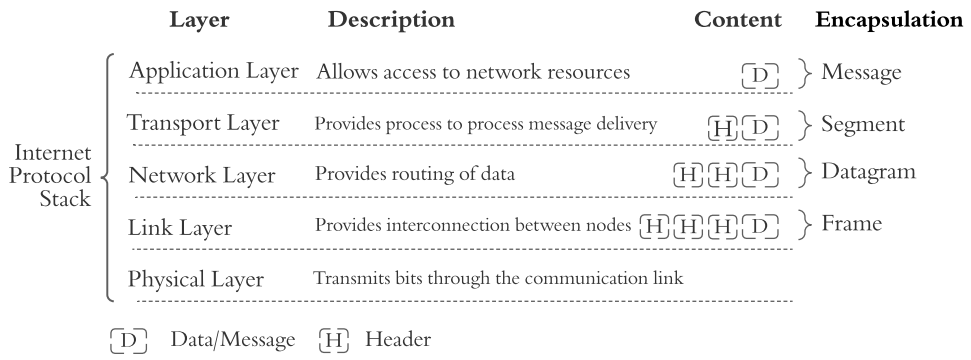


Figure 2.2 – The Internet protocol stack.

transfer protocol (FTP) reside here. In short, the HTTP protocol administers requests and transfers of Web documents. The SMTP protocol administers transfers of e-mail messages. The DNS administers the translation of domain names into numerical IP addresses, and the FTP protocol administers transfers of files between end systems. Application layer protocols commonly govern the exchange of packets of information between two end systems within an application and are thus shared in multiplex end systems.

Transport Layer

The transport layer resides beneath the application layer in the TCP/IP Protocol Architecture and retains the responsibility of transporting information from the application layer to application endpoints. Arguably, the two of the most important protocols in the transport layer are the TCP and the User Data Protocol (UDP), which are technically quite different. The UDP is a straightforward protocol that only enables the mechanism to receive or send a pack of information to another process. It has no additional measurements for verifying the delivery of the information, nor can it handle corrupted packets. Because of this, the UDP is usually considered unreliable but has the advantage of being lightweight, which is why it is used for data exchanges that are time-sensitive. The Transmission Control Protocol (TCP) is usually utilized in applications that require reliable data delivery because of its ability to verify and deliver data accurately as well as in proper sequences across the network. Contrary to UDP, TCP is a connection-oriented protocol that establishes a logical end-to-end connection between the corresponding hosts before any information is transmitted. TCP's reliability requires a large header to be

concatenated with the transmitted data, thus ensuring reliable delivery and creating connections might actually require a greater overhead than the actual transmitted information for a small amount of data. A practical example is Voice over IP (VoIP), which generally uses UDP instead of TCP because of the increased latency brought by TCP in the case of network congestion. Using UDP removes the instances where delayed VoIP packets get delivered later than they are supposed to, causing one end of the conversation to hear a short part of a word that was said a moment before in the conversation. Both TCP and UDP contain a 16-bit source port and destination port in the first word (row) of their headers to enable data delivery to the appropriate application process or network service; this is a critical element of computer networking as processes usually include multiple threads with instructions that are executed concurrently.

Network Layer

The network layer exists below the transport layer and is accountable for advancing datagrams (in the form of packets) between hosts. The network layer receives the transmitting data coupled with the destination address from the transport layer and administers the transmission of the packet to the transportation layer of the receiving host. The IP protocol resides in this layer and is central for all communicating devices; the protocol is responsible for defining the packets' structure and the action space of end systems concerning the packets. The route that a packet takes between end systems is governed by routing protocols located in the network layer. The network layer is occasionally referred to as the IP layer and can be visualized as a postal office that handles the routing of letters between addresses.

Link Layer

Below the network layer is the link layer. The link layer provides essential services that enable the network layer to route packets between end systems. Throughout the route of a packet, it usually passes through multiple hosts and routers, referred to as nodes. At each node, the transmitted information and destination source are passed down by the network layer to the link layer, which subsequently passes the information on to the next node in the route. Depending on the varying link layer protocols, the services offered by the layer may differ. Like TCP, there are protocols in the link-layer that offer reliable

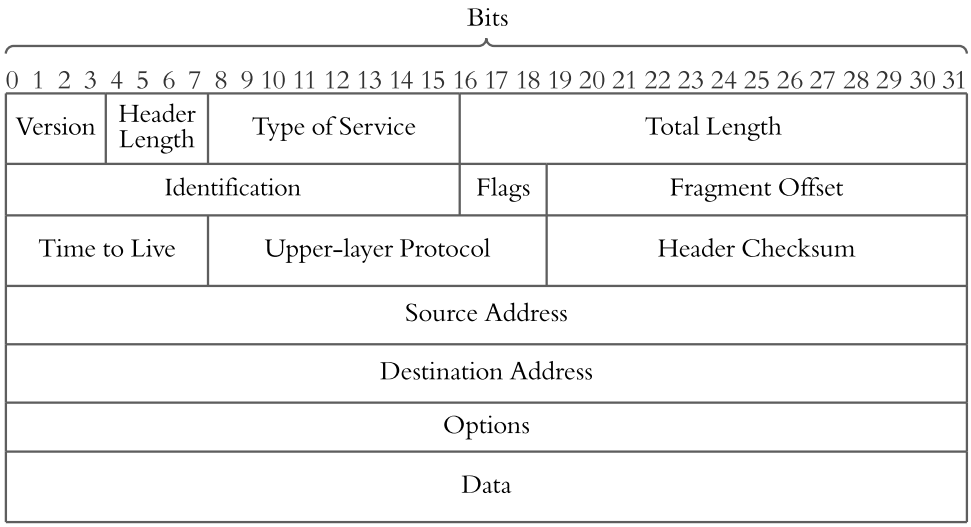


Figure 2.3 – A network-layer packet structure (IPv4).

delivery between nodes (TCP offers reliable delivery between end systems). There also exist other well-known protocols at this layer, such as WiFi and Ethernet. It is important to note that the protocols used in each node at the link-layer do not have to be identical throughout the route of a packet.

Physical Layer

At the bottom of the TCP/IP Protocol Architecture lies the physical layer. The physical layer is assigned to handle individual bits' transmission between the nodes in a networking system. Different protocols are utilized within the layer depending on the communication link's physical medium and its transmission properties. Each communication link, for example, coaxial cables or optic fibers, require its own protocol to determine how a bit is carried throughout the link. While this layer will not be handled in this thesis, it is essential to acknowledge, as it allows for a greater understanding and improved intuition of computer networking.

2.1.6 IP Addresses

Every device attached to a TCP/IP network is identified by what is called an IP address. The address generally consists of four octets that make up the 32-bit value (IPv4) that uniquely pinpoints the device's network interface. Usually, the address is represented as four values in the range of 0 to 255 separated by dots, also referred to as dotted-decimal notation, see Table 2.1.

Typically, a network of computers are connected to a gateway system linked to the Internet. The gateway system usually has two addresses. The computers connected to the gateway refer to the gateway with a separate address than external devices do, making data transfers more efficient [12]. Addresses can thus be used to specify either individual systems (unicast) or a specific group of systems (multicast). In the case of a packet sent to a multicast address, routers along the route can create copies of the same packet and forward it to the group members. In contrast to unicast and multicast addresses, a broadcast address is generally an address used to transmit information to all devices (not just a specific group) connected to a network [12]. Some IP addresses are reserved and unassignable to the majority of network devices. These addresses are typically restricted to broadcasting devices, private networks, and documentations governed by the IETF [13], and are used in the aforementioned network layer to move packets to their specified destinations.

Table 2.1 – Examples of reserved IP address ranges and their designations.

Address range	**Designation**
0.0.0.0-0.255.255.255	Local address space
10.0.0.0-10.255.255.255	Private address space
192.0.2.0-192.0.2.255	Documentation and example code
198.18.0.0-198.19.255.255	Testing and benchmark address space
255.255.255.255	Broadcast on a local hardware network

2.1.7 Network Traffic Flow

A network traffic flow is a high-level cumulative construct of the previously described computer networking building blocks; it outlines the information about packets moved between two end systems in a network at a given communication period. It is defined neatly as "*A flow is a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label as a flow. A flow could consist of all packets in a specific transport connection or a media stream.*" by RFC 3697 [14]. A network traffic flow is usually represented by a 5-tuple with elements that are included in the packet headers. The five parameters are; source IP address, source port, destination IP address, destination port, and the protocol used in the transport layer.

Table 2.2 – Example of four network traffic flows.

	Source IP	**Source Port**	**Destination IP**	**Destination Port**	**Transport Protocol**
1	192.168.121.1	67	172.16.255.186	67	UDP
2	192.168.121.2	58188	172.16.255.200	53	UDP
3	192.168.121.2	49194	23.45.23.65	443	TCP
4	192.168.121.3	50057	204.79.197.203	80	TCP

2.2 Supervised Learning

Machine learning, a sub-field of artificial intelligence, deals with learning complex relationships in data and producing mathematical models around them. Machine learning algorithms commonly fall into three categories; supervised, unsupervised, and reinforcement learning, depending on the structure of the data at hand [15]. The iterative property possessed by machine learning algorithms is a crucial aspect as models are able to independently adapt to new data, resulting in more accurate, reliable, and reusable outcomes. In recent years, the field has experienced a massive resurgence due to technological advancements that enable large scales of data to be collected and the computational power required to apply vast mathematical calculations to large datasets [16]. Supervised learning refers to algorithms that are given input data and their desired outputs in order to learn the detailed underlying mapping between the two. Linear regression, support vector machines, decision trees, and neural networks are all widely used supervised learning algorithms that can be used to solve regression problems.

2.2.1 Linear Regression

The linear regression method is used to approximate a dependent response variable using a set of independent variables. The method attempts to find regressor coefficients β that best represents a linear relationship between the response variable Y and the regression variable X [17]. For n observations each consisting of k regression variables, the model can be displayed in matrix notation as

$$Y = \beta X + \varepsilon \tag{2.1}$$

where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{k1} \\ 1 & x_{12} & \cdots & x_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & \cdots & x_{kn} \end{bmatrix}, \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}, \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}. \tag{2.2}$$

The term β_0 is the intercept of the linear function, and ε is the residual term that is sought to be minimized. To correctly perform linear regression and yield fair results, certain assumptions have to be satisfied. First, there must exist an approximately linear relationship between the response variable and the independent variables; this can partly be observed in a scatter-plot for each independent variable. Additionally, it is assumed that the residuals are independent and normally distributed, having a constant variance and a zero mean. Lastly, it is assumed that the independent variables are not perfectly correlated with each other as it would render the moment matrix $X^T X$ non-invertible, yielding no analytical solution [17]. To analytically estimate the regression coefficients, the method of *ordinary least squares* (OLS) is usually adopted to minimize the distance between the prediction and the response [18],

$$S(\beta) = \varepsilon^T \varepsilon = (Y - \beta X)^T (Y - \beta X), \tag{2.3}$$

yielding the analytically estimated regression coefficients

$$\hat{\beta} = (X^T X)^{-1} X^T Y. \tag{2.4}$$

2.2.2 Decision Tree Regression

A common and widely used supervised machine learning method is the decision tree [19]. The decision tree can be applied to both classification tasks as well as regression and works by learned rules on how to break up the input data using conditions. In essence, a decision tree arrives at an estimate using a series of conditions on the input data that narrows the possible target prediction space until enough confidence has been achieved to produce a final prediction. When applied to a regression task, the decision tree provides quantitative outcomes. Recursive partitioning is used to construct the decision tree starting from the first parent node, also called the root node. Apart from the end nodes, also called leaf nodes, each node is accompanied by a condition used for a two-way node split depending on whether the condition is true or false. To determine the structure of the tree, an iterative process starting from the root node is conducted, splitting each node on the feature f that results in the largest Information Gain (IG) [19]. For a binary decision tree, the objective function aimed to be maximized is defined as

$$IG(D_p, f) = I(D_p) - \left(\frac{N_{left}}{N_p} I(D_{left}) + \frac{N_{right}}{N_p} I(D_{right}) \right), \tag{2.5}$$

where D_p, D_{left}, D_{right} are the datasets of the parent, left, and right node; while N_p, N_{left}, N_{right} are the number of samples in the parent, left, and right node. The impurity measure I is dependent on the specific task and differs depending on if the decision tree performs classification or regression [20]. For regression, a common impurity measure for node t is the mean squared error, defined as

$$MSE(t) = \frac{1}{N_t} \sum_{i \in D_t} \left(y^{(i)} - \frac{1}{N_t} \sum_{i \in D_t} y^{(i)} \right)^2, \tag{2.6}$$

where D_t and N_t correspond to a nodes dataset and size, and $y^{(i)}$ the target value for data point i at the specified node. One of the biggest advantages of decision trees is that they can handle both categorical and numerical variables without the need for data preparation. Additionally, they tend to be fairly robust to outliers. Decision trees tend to be prone to overfitting caused by deep trees with many nodes [20]. The problem of overfitting can though be easily fixed by defining a maximal depth for the tree, essentially pruning excessive nodes.

2.2.3 Artificial Neural Network

Artificial neural networks (ANNs) aim to approximate complex functions by combining multiple linear operations regulated by layer-wise structured neurons [21]. Mathematically, each neuron j can be connected to another neuron i by a weight w_{ji} representing their connective strength. Each neuron produces an output given inputs that correspond to the outputs of previous neurons. Each layer in a neural network is constructed by an arbitrary amount of stacked neurons with connections to the adjacent layers' neurons. Given N inputs to a neuron, its output is the inputs' activated weighted sum. The output of neuron j, given the bias term w_{0j} and an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, can be described as a function $f : \mathbb{R}^N \to \mathbb{R}$ [22], defined as

$$f(\mathbf{x}, \mathbf{w}_j) := \sigma \left(w_{0j} + \sum_{i}^{N} w_{ij} x_i \right). \tag{2.7}$$

To prevent artificial neural networks from transforming into linear regression models, the choice of activation function must be nonlinear. Nonlinear activation functions facilitate neural networks the ability to capture both the complex and nonlinear dynamics in the input data [23].

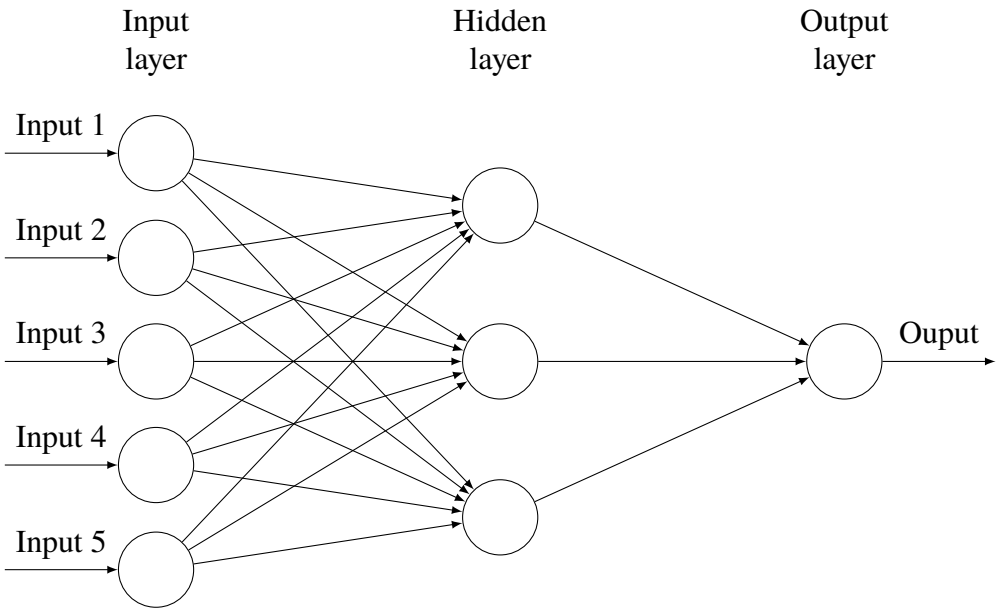


Figure 2.4 – Schematic view of an artificial neural network.

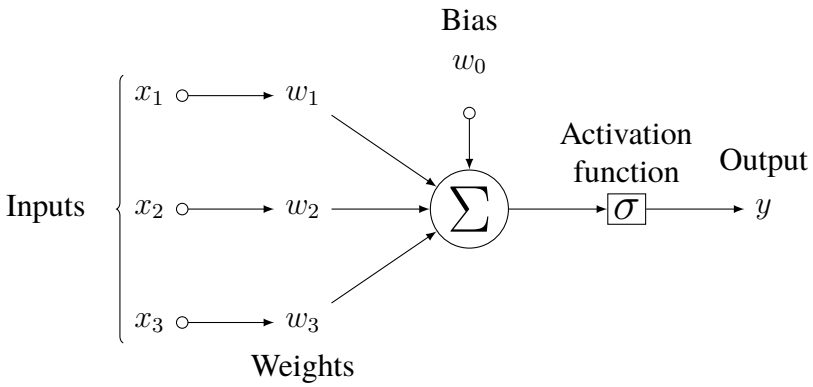


Figure 2.5 – Schematic view of an artificial neuron.

Activation Functions

Activation functions are used to map the inputs to a node to its output, determining the node's activation. They play an important role in a network's ability to capture complex relationships and help prevent imbalances in the network, such as the vanishing gradient problem [22]. Nonlinear activation functions allow networks to solve nonlinear problems, while linear activation functions result in the network, at most, approximating a linear function [21]. The choice of the activation function is usually based on the type of learning problem at hand. The most prevalent activation functions in deep learning are nonlinear functions such as the rectified linear unit (ReLU), the sigmoid function and the hyperbolic tangent function (*tanh*) [22]. Each of the functions has certain characteristics that can ultimately lead to a faster training process. The ReLU function is usually chosen as default in most deep learning architectures as it has proven to be a good general approximator; on the other hand, the sigmoid and the *tanh* functions are typically reserved for classification problems. The three mentioned activation functions are visualized in Figure 2.6.

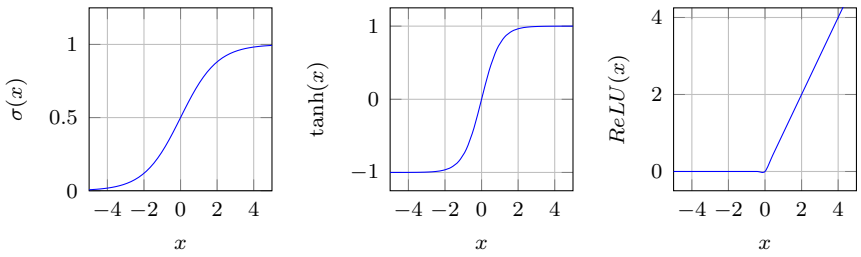


Figure 2.6 – The sigmoid, *tanh* and ReLU activation functions.

Loss Functions

In supervised learning, given M labeled data points, a neural network can generate M predictions. The prediction error of a neural network can be formulated using a differentiable loss function $\mathcal{L} : \mathbb{R}^{M \times M} \rightarrow \mathbb{R}$, which defines the distance between the predicted value $\hat{y}$ and the actual value y [22]. The choice of loss function heavily depends on the problem the model is trying to solve; regression problems utilize different loss functions than classification problems. However, it is essential to note that a loss function is required to be differentiable, as it becomes a crucial aspect whenever the network is trained using gradient descent [24]. The most common loss functions used for regression problems are the mean absolute error (MAE), and the mean squared error (MSE) [21], defined respectively as

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{2.8}$$

$$MAE = \frac{1}{n} \sum_{i=1} |y_i - \hat{y}_i|. \tag{2.9}$$

Weight Initialization

The initial choice of weights in a neural network is detrimental to the final performance of the network. If the weights are initialized correctly, the training process can be shortened, and the loss function can converge to minima [22]. With bad weight initialization, the risk of either vanishing gradients or exploding gradients becomes larger. Large initial weight values close to the value of one have the same effect as small initial weight values close to the value of zero, causing the gradients with respect to each weight to be very similar. In the training process, this leads to difficulties in minimizing the loss, as the backpropagation algorithm updates the weights similarly.

He initialization [25] was proposed with the non-linearity of the ReLU activation function in mind and to take the network size into account when initializing the weights. It provides proper initialization of weights that avoids an exponential reduction and increase of input signal magnitudes. The method initializes the weights in each layer randomly but within a range that is computed as a function of the layers' size. For each layer, uniform He initialization draws samples from the uniform distribution $\mathcal{U}(-\sqrt{\frac{6}{fan_{in}}}, \sqrt{\frac{6}{fan_{in}}})$ to produce the initial weights, where fan_{in} is the number of incoming network connections to the layer.

Backpropagation

Backpropagation is a key algorithm that allows neural networks to train and generate improved predictions given a dataset. It is the backbone of the learning process and is used to iteratively search for optimal network weights that minimize the loss function [24]. The backpropagation algorithm consists of two parts; a forward pass and a backward pass. The forward pass sends the input data to be processed by the operations defined in each network layer. Each node's output is transformed through an activation function for every layer and becomes the input for the next layer's nodes. This process is repeated until the final layer, commonly referred to as the output layer, whose output is interpreted as the models' estimation $\hat{y}$. The predefined loss function allows for mapping the estimate and the true value to a real number that generally is read as a prediction error. In the backward pass, dynamic programming and the chain rule are employed to compute the loss function gradient with respect to all network weights [21]. As the error of the output layer propagates backward in the network for each forward pass, the loss function gradient is computed with respect to each weight. The gradient descent algorithm adjusts the weights slightly so that the differentiable loss function moves towards a minimum [22].

Batch Normalization

Neural networks containing multiple hidden layers (layers between the input and output layer) are commonly called deep networks [24]. While deep networks are profoundly good at capturing the input data's underlying structure, they also entail the problem of distribution changes of the inputs as a result of the learned weights in the hidden layers. This problem is known as an internal covariate shift and has consequences that impair the training time [26]. Batch normalization was developed as mend to the problem and works by normalizing each hidden layer's inputs to have a zero mean and a unit standard deviation. The method reduces the distribution shift of the inputs caused by the hidden layers, making each layer's values more stable. Though it is not the intention of batch normalization, the method also brings about a slight regularization effect as it constrains the amount of which the weights are updated [26]. Normalizing each layer's inputs has a similar conclusive effect as normalizing the inputs of a shallow network, turning the learning problem's contours into a more rounded shape, facilitating ease for the gradient descent algorithm, effectively shortening the training time.

2.2.4 Recurrent Neural Network

A major shortcoming of traditional neural networks is their inability to recall previous information processed in the network [27]. Since the neurons in each layer have no connection, the inputs to the neurons in the next layers also become independent of each other. Recurrent neural networks (RNNs) enable a circular network structure and allow neurons in a network to be connected to themselves, usually referred to as a recurrent structure [27]. The recurrent structure enables information to persist within the network and facilitates properties that allow RNNs to better deal with and predict time series data compared to traditional neural networks [28]. The figure below displays a diagram of an RNN.

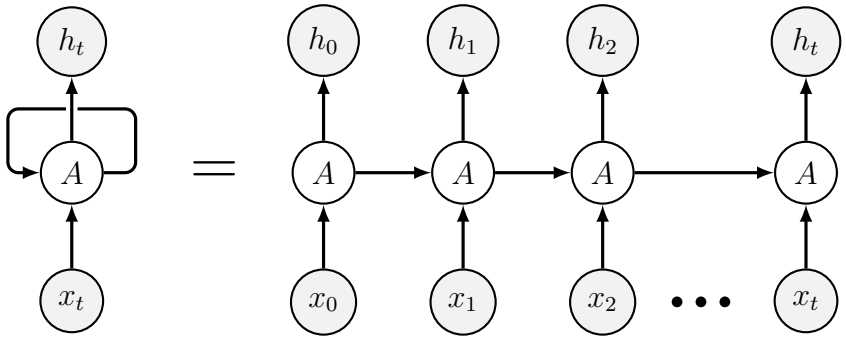


Figure 2.7 – The structure diagram of an unrolled recurrent neural network.

Both RNNs and traditional feedforward neural networks (FNN) utilize the forward pass in their prediction process. The most significant difference lies in the method of propagating the error back through the network. While FNN employs the backpropagation algorithm, RNNs utilize the backpropagation through time (BPTT) algorithm [28]. BPTT allows error propagation to consider the time dimension [27], an essential property as the output layer error is influenced by the previous moments. The algorithm works by unfolding an RNNs computational graph at each time step $1, 2, ..., T$ to capture the dependencies between the network parameters. With the network unfolded, the errors at each time step are computed and accumulated to apply the traditional backpropagation algorithm using the chain rule to calculate and store the gradients [28].

2.2.5 Long Short-Term Memory Network

Long Short-Term Memory networks (LSTMs) fall into the category of RNNs and are proven to have the capability of learning complex long-term dependencies. The method was first introduced by Hochreiter and Schmidhuber in 1997 [29], and has since been widely implemented in applications that deal with complex problems like speech recognition and machine translation [30, 31]. Much like traditional RNNs, LSTM models are built using a structured chain of repeating modules, called LSTM cells, containing four interacting layers. Instead of neurons, LSTMs employ cell states, displayed in Figure 2.8, which are time-wise sequentially connected with minor linear interactions. Three nonlinear summation units exist in each LSTM cell, referred to as gates, regulate the information in each cell state by carefully removing or adding new information. The input gate, output gate, and forget gate are composed out of matrices containing weights and a pointwise multiplication operation, activated by a sigmoid function to control the flow of information [29].

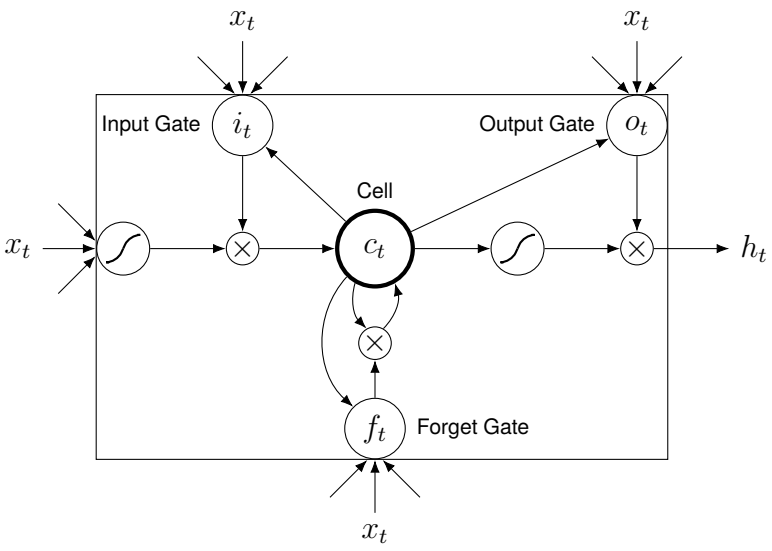


Figure 2.8 – The structure diagram of an LSTM cell.

Given a time series input data of length T, the forward pass algorithm used in a single LSTM cell is represented by,

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.10}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2.11}$$

$$c'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{2.12}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t, \tag{2.13}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.14}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{2.15}$$

Similar to a traditional RNN, LSTMs utilize the BPTT algorithm to propagate the error backwards and to update the weights W_f, W_i, W_c, W_o which correspond to each of the gates. Starting from T in the time series, the gradient of each parameter is computed at each time step and used to update network parameters by way of the gradient descent method.

2.2.6 Support Vector Machine

Given a dataset with N features, a support vector machine is an algorithm that finds a hyperplane in the N-dimensional feature space that separates the data points into their corresponding classes [32]. More specifically, it finds the hyperplane with the maximum margin distance to the different classes' data points. By choosing the hyperplane with the largest margin, the classifier also provides more confidence in its classification of future data points. The hyperplane's position and orientation are influenced by the data points that are most closest to it, also called support vectors. From the support vectors, the algorithm maximizes the hyperplane margins; therefore, removing the data points that correspond to the support vectors also changes the position of the hyperplane [32]. Commonly, the hinge loss is combined with a regularization parameter to form a cost function that balances the loss and the margin maximization. Kernel functions are used to enable a linear separation using a hyperplane by transforming the input data into a higher dimensional feature space [33].

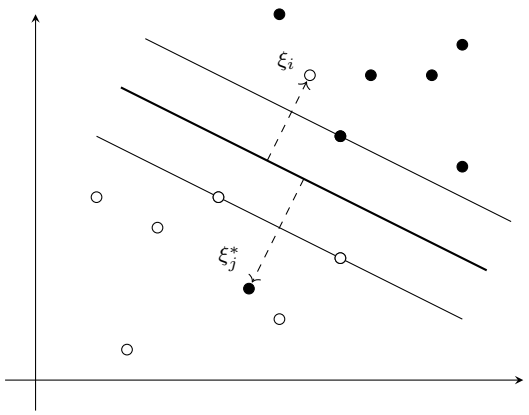


Figure 2.9 – SVM enabling linear separation with an error tolerance.

The support vector machine can also be applied to regression problems while maintaining the main features that characterize the algorithm, i.e., maximal margin. In support vector regression (SVR) [34], we use the same principles found in SVMs to formulate an optimization problem that seeks to find an approximate function,

$$f(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + b \quad \mathbf{x}, \mathbf{w} \in \mathbb{R}^N, \tag{2.16}$$

that minimizes the prediction error while enclosed by the narrowest tube possible, representing the error tolerance ϵ. Since most problems can not be perfectly modeled due to outliers, a soft-margin method is applied using the slack variables ξ, and ξ^* to regulate the number of data points outside of the error tolerance. If an observed point is above the tolerance tube, ξ_i takes the value of the positive difference between the observed point value and ϵ. Analogously, if a point is observed below the tolerance tube, ξ_i^* takes the value of the positive difference between the observed point value and ϵ, as shown in Figure 2.9. Regularization can be achieved with the tuneable slack parameter C, which adjusts the importance of minimizing the error versus the flatness of the multiobjective optimization problem.

The final optimization problem for a support vector regressor is described as,

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & y_i - (\mathbf{w}^T\mathbf{x}_i + b) \leq \epsilon + \xi_i^* \quad i = 1...N \\ & (\mathbf{w}^T\mathbf{x}_i + b) - y_i \leq \epsilon + \xi_i \quad i = 1...N \\ & \xi_i^*, \xi_i \geq 0 \quad i = 1...N. \end{aligned} \tag{2.17}$$

Chapter 3

Related Work

3.1 Network Traffic Prediction

Various researchers and developers have investigated network traffic prediction using intelligent algorithms, most of which focus on time series methods where the objective has been to predict the arrival of individual or bursts of packets to a select few destination addresses or servers. This information could then potentially be used to identify critical users whose quality-of-service is in jeopardy due to the expected traffic behavior. The data used for such methods are generally sequential in which observations have been made in successive order, often times varying by a few seconds. A common way of predicting time series is by using recurrent neural networks that have the ability to exhibit temporal dynamic behavior [27]. By using historic packet sizes and inter-arrival times, future ones can be predicted. Numerous time series models have been proposed to deal with the subject, using statistical learning [35], machine learning [5, 36, 6, 37] and hybrid schemes [3]. It is important to note that most methods that deal with traffic prediction require a warm-up period, i.e., the model needs to observe the packets during some time T, in order to make predictions on the traffic for $t > T$.

A recent study that took advantage of warm-up periods was done by Azari et al. [2], who explored both long short-term memory (LSTM) and autoregressive integrated moving average (ARIMA) to investigate cellular traffic prediction using different wisely-selected sets of features from their internal dataset. By introducing additional augmented features, such as the ratio of the received and dispatched packet count, the study demonstrated that a network, consisting of

an LSTM layer with 100 units, a fully connected layer, and a regression layer, contribute to superior future traffic predictions compared to the ARIMA model, reaching an accuracy of 78% on a busy interval (a time period containing at least one packet). Aldyani et al. [3] deal with cellular traffic prediction similarly by treating it as a time series problem. Their proposed methodology involves advanced pre-processing that combine fuzzy c-mean clustering and an exponential smoothing method to improve deep learning LSTM and adaptive network-based fuzzy inference system (ANFIS) models. The network data is split into clusters of similar characteristics using the fuzzy c-mean clustering algorithm. The clusters are subsequently fed to a weighted exponential smoothing (WES) method to obtain predictors considered as inputs for the LSTM and ANFIS time series models. The proposed pre-processing is shown to have a reducing effect on prediction errors.

While the subject of network traffic prediction has been thoroughly studied through the use of time series models, the problem is still left partially unsolved when applied to flow traffic due to the data observation window time-series models tend to utilize. The warm-up/observation periods are a hindrance for network traffic flow prediction, as the majority of network traffic flows are short and contain only a handful of packets [7, 8]. The required warm-up periods could thus correspond to the entire length of short flows. Additionally, these methods are predominantly only used to predicting the traffic to a set of destinations addresses [2, 3, 5, 6] and do so without utilizing the information embedded in the IP addresses for their predictions.

3.2 IP Address Representation

In order to learn the structural properties of the Internet, it is essential to represent IP addresses in a fashion that preserves their networking properties. Understanding how close two end systems are, is crucial to approximate both the travel time of a packet and potential atypical events. Previous attempts to use IP addresses in machine learning algorithms are mostly based on embedding techniques that project the addresses in a low-dimensional space using vector representations. The vector representations are then further used to estimate hosts' positions in networks and, as a result, can also be utilized to restore structural properties.

IP2Vec is an embedding method proposed by Ring et al. [38] builds on the ideas of a popular approach used in natural language processing, *Word2Vec* [39], to

extract information about an IP address. Their method utilizes a skip-gram neural network, consisting fully connected network with a single hidden layer to perform unsupervised learning on addresses and their previous behavior. The learned weights of the network are subsequently used as a vector representation of the IP address. Since the method relies on context information gathered from traffic flows from each address (as input data), behavioral similarities can be measured between pairs of addresses and are valuable in the case of pinpointing possible network attacks. While the proposed method performs well at identifying similarities between two addresses, it includes drawbacks because of its dependency on previous behavior. The dependency results in the representations being non-stationary, causing them to lose their performative properties over time as the behaviours evolve. Additionally, IP addresses that are not present in the training dataset have no representations in their method and are can thus not be embedded into models built on top of it.

Li et al. [40] deal with IP address embeddings differently in their deep learning-based approach DIP. By utilizing a deep neural network, the proposed method is able to discover hidden features in the input data that include IP addresses and accurately preserve structural properties in their representations, allowing for the distance between nodes to be inferred, even if they are not included in the dataset used for training. The inputs used include the two nodes IP addresses, routing information in the form of routing prefixes, and a hop count (number of routers between the two nodes on the default route); this allows DIP to accurately produce representations used to estimate the distance between any two end systems. The method works by normalizing the IPv4 addresses of the source and destination end-systems along with the routing prefixes into two 64 bit arrays. The addresses are then sequentially fed into their neural network architecture in parallel, introducing 8 new bits of the address at each neural network layer. The last neural network layers in the architecture are used to estimate the hop count between the two embedded IP addresses. DIP has been suggested to be useful as a replacement for expensive measurement systems to select load balancing servers and can also be used as a defense mechanism against attacks such as IP spoofing. Since the size and intricacy of the Internet are enormous, and the collected data used to create such embeddings often are too sparse to form a cohesive representation of scale, they usually fail to capture the full structure of unseen addresses, even with the best methods. Furthermore, traffic flow datasets are rare, and finding data of flows containing routing prefixes is even more difficult, resulting in the method being hard to implement.

Similar to DIP, Beverly et al. [41] researched latency predictions between IP addresses using a Support Vector Machine (SVM) as early as in 2006. However, instead of predicting hop counts, their method approximated the latency in milliseconds. The simple method, an SVM with a fifth-degree polynomial kernel, performed surprisingly well with an estimated accuracy within 30% of the actual value for approximately 75% of the dataset. Comparable to [40], the study also transformed IPv4 addresses into a vector of 32 elements to be used as input data for their model, each corresponding to the bits of the address.

Transforming IP addresses into their corresponding bits, as proposed in both [41] and [40], can prove beneficial as it includes all elements that would enable a model to learn structural properties that do not change with time nor depend on previous communications.

3.3 Regression Methods

Unlike traditional network traffic prediction that deals with estimating sequential chunks of packets to specific addresses, this thesis's scope is to investigate complete network traffic flows consisting of multiple packets of various sizes sent to unspecified destinations from unspecified sources. The data points in flow traffic are generally not sequential or ordered per address; an exemplary predictive model must generalize to different end systems and find flow-based relationships without any individual flow history. The objective problem is thus considered a regression problem and will be solved using a regression framework.

The traditional deep residual learning method [42] has been thoroughly investigated and used in many applications that involve inputs in the form of images. The *ResNet* architecture, as it's commonly called, has been proven to be a potent approach to extract hierarchical features from images and avoid vanishing gradients through the use of skip connections. In 2020, Chen et al. [43] proposed a deep residual model for the regression of nonlinear functions. The architecture consists of one dense block and two identity blocks stacked repeatedly before the output layer. The suggested regression method replaces the convolution layers found in the traditional *ResNet* with dense layers. Each block consists of three hidden layers with batch normalized outputs sent through a ReLU activation function, except the last layer, which receives a linear activation. The difference between the two blocks lies in the skip connection. The identity block sends the input directly through the skip

connection, while the dense block sends the input through a parallel dense layer with batch normalization before advancing it through the skip connection. In the case of regression, artificial neural networks (ANN) usually risk having the loss function getting stuck in a local minimum, yielding inferior predictions compared to other methods such as SVMs that are guaranteed to converge to the global minimum. Despite this, the proposed regression *ResNet*, with optimal width and depth, proved to outperform most traditional techniques in their experiments, which included regression of multiple nonlinear functions [43]. While the results may partly be influenced by the underlying structure of the predicted nonlinear function, the method makes for a practical and novel approach to regression problems, facilitating a large room for possible advancements that can be implemented conveniently.

One possible methodological direction for regression is to apply an ensemble learning method to the problem. By strategically combining multiple learning algorithms, better predictions have proven to be obtainable. By linking various learning methods together, locally optimal solutions can be avoided when initializing the learns differently. A weighted sum of several hypotheses generally approximates a function better than a single one [44]. For this reason, Qui et al. [44] proposed an ensemble deep learning method for regression and time series forecasting. The method takes advantage of outputs produced by multiple deep belief networks (DBNs), trained using different numbers of epochs and feeds them as inputs to a support vector regressor (SVR), with the expected prediction values of the original targets. The prediction results on the regression-based dataset *California Housing* [45] show more accurate performance by the proposed ensemble than from single structure algorithms, including stand-alone SVR, FNN, and DBN. While the choice of using DBNs to form the ensemble proves to work well on the housing dataset, it does not necessarily transfer to traffic flow data. Nevertheless, combining multiple learners with an SVR is an exciting approach as the learners can be arbitrary.

3.4 Uncertainty Estimation

Decision-making models built on real-world data have consequential traits that can potentially affect the majority of end-users. It is thus essential for such models to include estimates of how reliable each prediction is. In most machine learning applications where the final decision directly influences humans, e.g., autonomous driving, insurance, and healthcare, uncertainty measurements have a critical role in mitigating severe accidents. There exist numerous machine learning methods capable of producing uncertainty estimates. Specific models built using a Bayesian framework are able to provide uncertainties as part of their predictions [46]. Other methods require multiple trained models or numerous predictions in order to compute the variance in the predictions [47].

Nix & Weigend proposed the uncertainty estimation method Mean-Variance Estimation (MVE) in 1994 [48], which adopts neural networks to learn the variance of the target probability distribution given the prediction and the true target. Model uncertainty has also been investigated using Bootstrap methods [49] where multiple models are trained with different parameter initializations using subsets of the original dataset to estimate the variance and mean of the predictions. Yarin Gal & Zoubin Ghahramani [47] demonstrated that incorporating dropout in neural networks provides ground for interpreting the model as if it is performing variational inference. Commonly, dropout is only used during a model's training phase; however, using it during inference provides the same outcome as an ensemble of networks since the dropout involves deactivating a random subset of neurons in the network resulting in the use of different networks for each prediction. By creating a set of different predictions from the same input, a mean and variance tuple can be obtained to estimate the uncertainty. Nonetheless, the computational cost of running bootstrapping methods is significant and can thus be prohibitive if the application is latency-constrained. Section 6.2 further discusses the discrepancies found in the results between target variables.

Chapter 4

Methods

This chapter aims to present the thesis works' research paradigm and methodology used to achieve the most accurate network traffic prediction model compared to baselines and traditional machine learning algorithms described in Section 2.2. The chapter introduces the reader to the inner workings of residual neural networks for regression problems, which is the main method this thesis examines to predict network traffic better. Furthermore, the datasets used in the research experiments and the evaluation metrics used to quantify the performances of the final models are also presented in this chapter.

4.1 Research Process

The main objective of this research work is to investigate suitable machine learning algorithms that are able to capture inherent relationships within network traffic flows that facilitate accurate predictions. To evaluate whether a method is accurate and has a predictive skill, two baselines are first established using naive predictive models that predict the mean and the mode target value from the data used for training. A model that achieves better values in the defined performance indicators than the naive baselines is determined to be accurate and skillful. The best performing algorithm is determined by a quantitative analysis using well-known evaluation metrics used in supervised machine learning, i.e., MSE, MAE, MdAE, coefficient of determination, and SMAPE, all defined in Section 4.5.

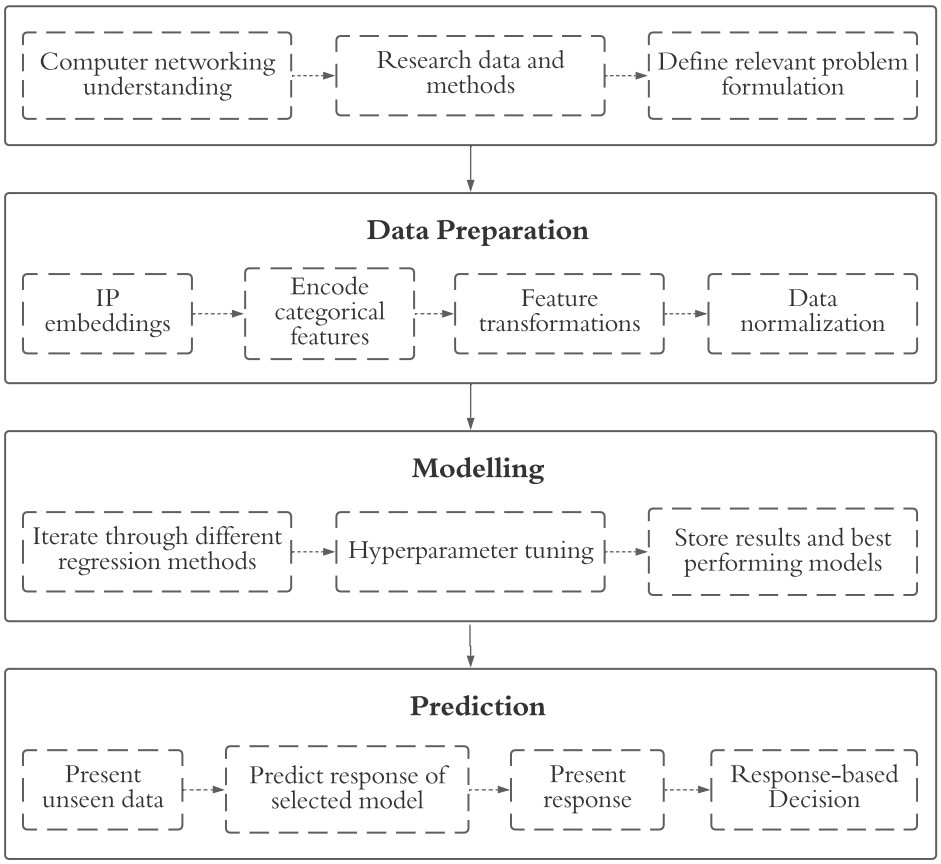


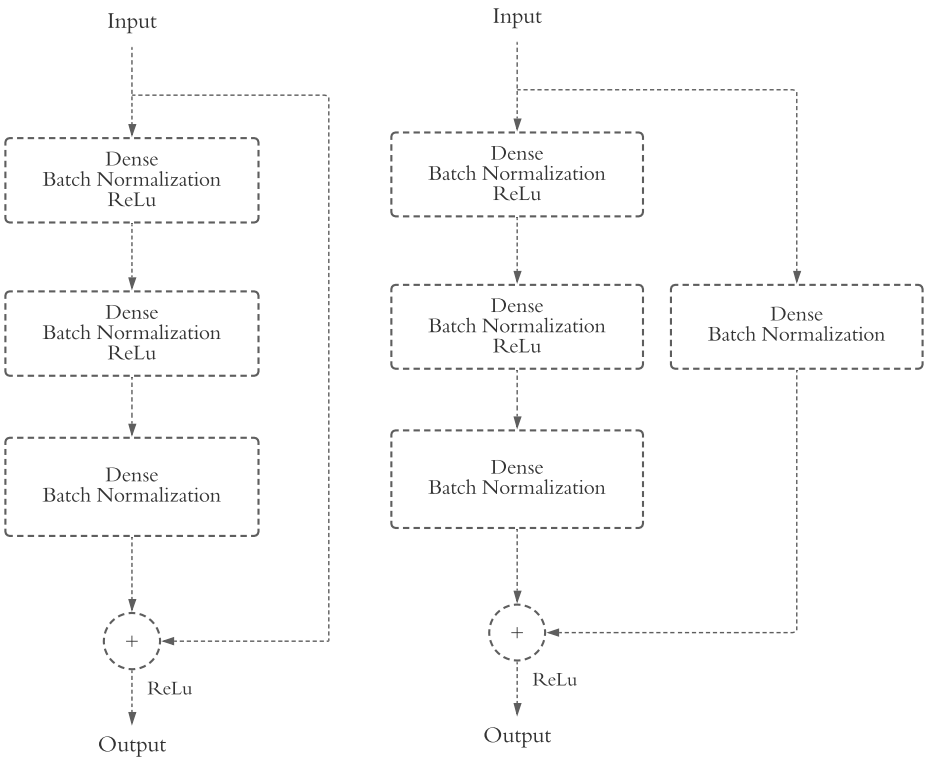
Figure 4.1 – Research Paradigm.

The size, duration, and packet count of a network traffic flow are predicted using a residual neural network specifically designed for regression problems [43], with its performance quantitatively compared to baselines and traditional algorithms such as linear regression, decision tree regression, and other neural network architectures.

Furthermore, an exploratory approach is used to investigate the large hyper-parameter space for each supervised learning algorithm to administer optimal conditions that facilitate legitimate comparisons. This exploratory methodology is achieved using the grid search technique that trains and evaluates each method using different sets of hyper-parameters. In consideration of user-privacy regulations and ethical codes, this research also explores different ways IP addresses can be handled to preserve privacy and their final impact on the model performances. Lastly, the uncertainty of the residual neural network is estimated using the Monte Carlo dropout method to understand the characteristics of the model better.

4.2 Residual Neural Network

The residual neural network architecture, first introduced [42] for image recognition, demonstrated that residual shortcut connections offer vital properties to the training process of deep networks. The shortcut connections were demonstrated to have a diminishing effect on the vanishing gradient problem that occurs when networks become very deep, without the need for additional network parameters or added computational complexity. Residual shortcuts enable networks to become deeper without running the risk of weights not being updated, potentially allowing for more complex relationships to be captured by the network.



(a) Identity Block (b) Dense Block

Figure 4.2 – Block structure of the regression ResNet architecture.

Typically, the *ResNet* architecture consists of two blocks that are stacked repeatedly between the input layer and the output layer. For image recognition, the two blocks utilize unique residual shortcuts, classified as identity shortcuts

and convolution shortcuts. In the case of regression, [43] proposes replacing the convolution shortcut with a dense layer as shown in Figure 4.2. Additionally, the regression *ResNet* replaces all convolutional layers and pooling layers in the blocks used for image recognition with fully connected layers, resulting in each block consisting of three dense layers with batch normalization. By keeping the units in each dense layer constant, the shortcut in the identity block only requires a tensor addition operation, as shown in Figure 4.2(a). The batch normalization layers are used to prevent internal covariate shift of the input data and facilitate a faster training process. The ReLU activation function is used in the first two layers of each block as well as in the final summation of the output, allowing the network to overcome the vanishing gradient problem [22].

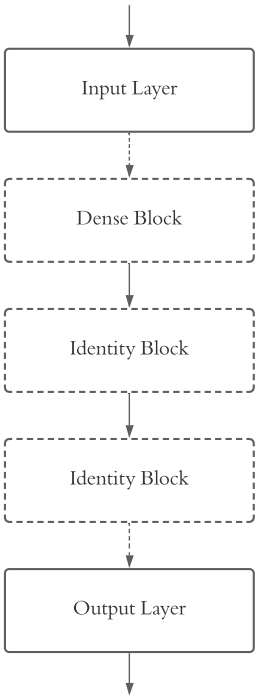


Figure 4.3 – The ResNet architecture structure.

The complete regression architecture is structured by stacking a dense block followed by two identity blocks repeatedly until a satisfactory depth has been reached, as shown in Figure 4.6.1. This pattern is chosen to keep the architecture generalizable; when the input and output shapes of two identity blocks are different, a dense block is utilized to reshape the dimension of the processed data. For network traffic prediction, the network's width, that is, the number of units in each dense layer, is set to a constant and has to be large enough to approximate complex mappings.

4.3 Uncertainty Estimation

Since machine learning models trained on real-world data produce results that have a vast influence on end-users, it is essential to include estimations of how reliable each produced prediction is. One solution is a prediction interval, a probabilistic estimate of an interval that a specific prediction is anticipated to fall within. Typically, the interval for a point prediction is defined as,

$$\hat{y} \pm z\sigma_y \tag{4.1}$$

where $\hat{y}$ denotes the prediction made by the model, σ_y the standard deviation, and z the multiplier that depends on the coverage probability. Traditionally, the coverage probability is set to 90%, 95%, or 99%.

As non-uniform distributions and noisy data introduce challenges for learning algorithms, the data noise variance is usually approximated using a model trained on a held-out validation set. The data variance is then added to the point prediction variance before a wider prediction interval is computed that accounts for both model and data uncertainty. The total variance for each estimation is thus given by

$$\sigma_y^2 = \sigma_{model}^2 + \sigma_{data}^2, \tag{4.2}$$

where σ_{model}^2 corresponds to the variance produced by the predictor, and σ_{data}^2 corresponds to the inherent data variance. Since the data variance is unknown, it has to be estimated. Traditionally, it is estimated as the residual variance obtained from an independent validation set. However, since the distributions for flow properties are wide and account for values that differ by many orders of magnitude, such variance is bound to be overestimated and dominate the total prediction variance [50]. This thesis will therefore only investigate the uncertainty estimation of the model, using $\sigma_y^2 = \sigma_{model}^2$ as opposed to Equation 4.2.

In order to estimate the model variance σ_{model}^2, the study employs the Monte Carlo dropout method. The Monte Carlo dropout method can be applied to any neural network and works by stochastically deactivating a fraction of the node outputs within the network according to a Bernoulli distribution [47]. The stochastic deactivation of neuron causes a regularizing effect while training, as an approximation of a probabilistic deep Gaussian process is formed [47]. By activating dropout during inference, different predictions will be produced in what is called a stochastic forward pass. In essence, each point

prediction in a stochastic forward pass is generated by a slightly different network, constituting what can be compared to an ensemble learning method. The different predictions generated from the same input data can then be used to estimate the model uncertainty.

The final point prediction of the Monte Carlo dropout method after the stochastic forward pass is repeated N times per data point is defined as,

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i, \tag{4.3}$$

where the estimated model variance is defined as

$$\sigma^2_{model} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - \bar{\hat{y}})^2. \tag{4.4}$$

Using the estimated model mean value and variance, a $(1 - \alpha)\%$ prediction interval is computed from Equation 4.1 for each data point. The Prediction Interval Coverage Probability (PICP) is used to evaluate the prediction intervals, which define the percentage of instances for which the target value is contained within the produced interval. On average, the PICP should be close to $(1-\alpha)\%$, as it is a quantization of validity [51]. For this thesis, α is set to 0.05, providing 95% prediction intervals. Another quantity of interest is the Mean Prediction Interval (MPI), quantifying the average prediction interval length. The MPI is regarded as a representation of optimality, where a narrow average interval is generally more desirable.

4.4 Data

A total of four Internet traffic flow datasets are used in the experimental phase of this thesis work, all of which are listed together with their shortened name, description, and individual data point count in Table 4.1.

The first dataset, DS-1, was collected at the University of Cauca by J.S. Rojas [8] using the packet capture software CICFlowMeter [52] at different hours between April to June of 2019. It contains the packet header information of flows as well as the total accumulated packet count, duration, and flow size. DS-1 also includes labels for each flow that classify their consumed network

service (e.g., Youtube, Google, and Facebook). However, in order to keep the study uniform and to prevent the final method from being dependent on any data generated after a given flow has ended, only features that are available in the packet headers are used as input features. The dataset consists of evenly distributed flows utilizing both the TCP and the UDP transport-layer protocols.

DS-2 was designed by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [53] and consists of both real normal activities and synthetic attacks. The synthetic flows that simulate attacks were removed from the dataset before preprocessing, as this study aims to investigate traditional network traffic flows. From the total 2,540,038 flows in the dataset, 321,283 attack flows are discarded, leaving 2,218,755 legitimate data points.

The ISCX-IDS-2012 intrusion detection dataset (DS-3) was simulated during the span of a week in June of 2010 by researchers at the University of New Brunswick [54]. The data contains simulated user behavior from abstracted user profiles at the institutions using programmed agents to execute their behavior in real-time. The attack scenarios incorporated in the dataset to mirror real-world instances of malicious behavior were discarded for the purpose of the thesis work, leaving a total of 3,595,630 legitimate flows.

Additionally, a private and processed dataset (DS-4) is provided by Ericsson, consisting of real-world cellular packet header logs collected during 2017 and 2020. The packet header logs consisted of approximately 100 million packet headers which were subsequently processed into 1,131,728 individual flows using a timeout limit of 10 seconds between each packet header. Because of privacy regulations, the dataset was anonymized by shuffling the first two octets of each unique destination IP address and removing the source IP address. Additionally, the International Mobile Subscriber Identifier (IMSI) number of the cellular device that can specifically be used to identify a mobile subscriber was also discarded. With the exception of DS-4, the datasets listed in Table 4.1 are publicly available for research purposes.

Table 4.1 – Datasets used for experiments and evaluations.

Dataset	Description	Data Count
DS-1	Flow data from the University of Cauca	2 704 839
DS-2	Flow data from the UNSW-NB15 network dataset	2 218 755
DS-3	Flow data from the ISCX-IDS dataset	3 595 630
DS-4	Packet-header data provided by Ericsson	1 131 728

The datasets listed in Table 4.1 all contain a multitude of fields that describe certain characteristics of each individual flows. However, to create a robust and universal traffic prediction model, it is important that the information within the fields is not manufactured by post-flow algorithms. Thus, only the packet header information fields are pulled; see Table 4.2. The total flow packet count, size, and duration are treated as target variables. The target variables are to be approximated by models that process the explanatory variables, i.e., source IP address, source port number, destination IP address, destination port number, transport layer protocol, and the size of the first packet in the flow.

Since DS-4 is heavily anonymized, information about the source device is not available. Additionally, important to note is that only DS-4 contains the true, measured size of the first packet. DS-1, DS-2, and DS-3 all approximate the first packet size as the mean packet size within the actual flow. While the actual true size of the first packet is preferable, such approximation should not theoretically be far off from the true size as studies have shown that the discrete distribution of packet sizes within flows are narrow and are usually dominated by a single packet-size [55, 56].

Table 4.2 – Fields pulled from the datasets.

Field	Explanation	Available in
`src_ip`	Source IP Address	DS-1, DS-2, DS-3
`src_port`	Source Port Number	DS-1, DS-2, DS-3
`dst_ip`	Destination IP Address	DS-1, DS-2, DS-3, DS-4
`dst_port`	Destination Port Number	DS-1, DS-2, DS-3, DS-4
`proto`	Transport Layer Protocol	DS-1, DS-2, DS-3, DS-4
`first_pkt`	First Packet Size (bytes)	DS-1, DS-2, DS-3, DS-4
`pktTotalCount`	Total Packet Count	DS-1, DS-2, DS-3, DS-4
`octetTotalCount`	Total Flow Size (bytes)	DS-1, DS-2, DS-3, DS-4
`flowDuration`	Total Flow Duration (s)	DS-1, DS-2, DS-3, DS-4

4.4.1 Data Analysis

A brief data analysis on the different datasets is conducted in order to understand the final model behavior and its results. Intuitively, the total accumulated size of a single flow should be closely correlated to the number of packets within the flow. In order to verify this, the Pearson correlation coefficient was investigated between the target variables in each dataset. The coefficient r describes the normalized covariance of two variables, resulting in a value between -1 and 1, in which 1 would indicate a perfect correlation and -1 a perfect inverse correlation. The Pearson correlation coefficient between n pairs of variables x

and y is defined as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{4.5}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{4.6}$$

and

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{4.7}$$

The Pearson correlation coefficient between the target variables in each dataset is displayed in Table 4.3 and indicates that the size and packet count of flows are universally closely correlated. However, in contrast to what might have been intuitively expected, the duration of individual flows seems to be linearly correlated to neither the packet count nor the size of the flow. This characteristic has also been observed in [57, 58], where the authors reasoned that user and application behavior are the primary influencers of a flow's duration. In the case of user behavior, a network user is mainly influenced by their link speed whenever faced with a choice that is associated with a transfer size [59], an example of this might be a user choosing to abort the download of a large file. Thus, a flow's size and duration need to be treated as independent dimensions.

Table 4.3 – Pearson correlation coefficient between the target variables for each dataset.

		Packet Count	**Flow Size**	**Flow Duration**
DS-1	Packet Count	1.000	0.922	0.050
	Flow Size	0.922	1.000	0.041
	Flow Duration	0.050	0.041	1.000
DS-2	Packet Count	1.000	0.954	0.067
	Flow Size	0.954	1.000	0.067
	Flow Duration	0.067	0.067	1.000
DS-3	Packet Count	1.000	0.767	0.012
	Flow Size	0.767	1.000	0.009
	Flow Duration	0.012	0.009	1.000
DS-4	Packet Count	1.000	0.950	0.379
	Flow Size	0.950	1.000	0.226
	Flow Duration	0.379	0.226	1.000

The target variable distributions for each dataset are shown in Figures 4.4, 4.5, and 4.6, binned into orders of magnitudes (powers of ten). From the figures, it is clear that the majority of flows in each dataset are substantially short, containing only a few packets that are transported in less than one second. DS-1 and DS-4 share very close similarities and contain significantly smaller flows than DS-3. While DS-3 is dominated by flows containing 10 to 99 packets, it bears a unique property of encompassing much larger accumulated flow sizes than the other datasets. This property indicates that the flows in DS-3 have larger individual packets than the flows in DS-1, DS-2, and DS-4.

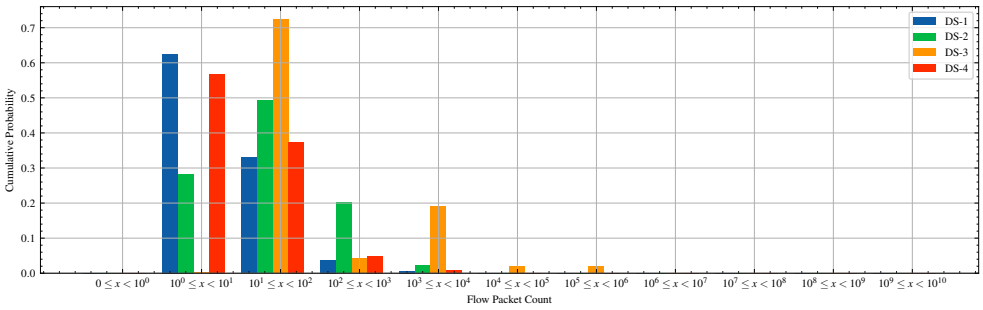


Figure 4.4 – Distribution of *flow packet count* for each of the datasets.

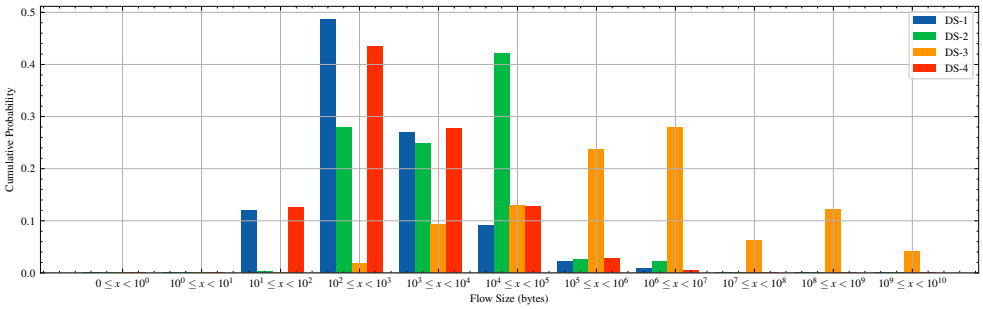


Figure 4.5 – Distribution of *flow size* in bytes, for each of the datasets.

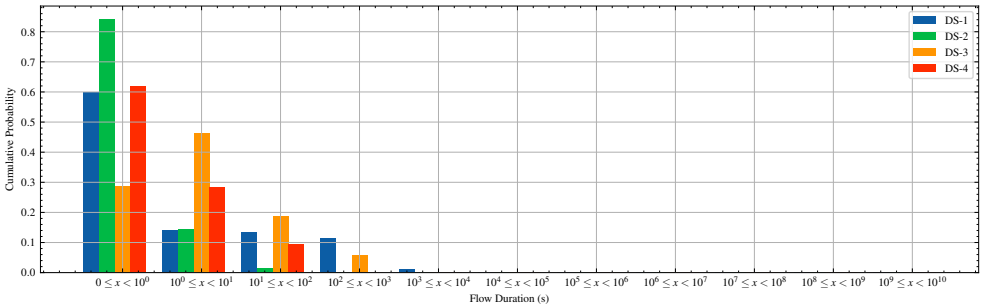


Figure 4.6 – Distribution of *flow duration* in seconds, for each of the datasets.

4.4.2 Data Preparation

All IP addresses available in the datasets are represented by individual bits features by transforming the two IP address fields for source and destination systems into a total of 64 additional fields. Each IP address is thus represented as a vector $\mathbf{x}_{\text{IP}} \in \{0,1\}^{32}$. To maintain the relation between ports and their inherent structure, the ports used during the flow by the source and destination systems are also transformed into their corresponding bit-representation as $\mathbf{x}_{\text{Port}} \in \{0,1\}^{16}$. Additionally, the field containing the transport-layer protocol is one-hot encoded using the IANA Assigned Internet Protocol Numbers standards [60].

Since the target variable distributions are non-negative, partially right-skewed, and vary by many orders of magnitude, a logarithmic transformation is applied to avoid large error gradient values that can potentially unstabilize the learning process as a result of altering the weight values substantially for each iteration.

Furthermore, all dataset features, including the first packet size and the target variables, were re-scaled using normalization to eliminate the scale disparities between the features. Normalizing the features help neural networks to reach smaller generalization errors as the input sensitivity is stabilized. The normalization processes utilize the largest and the smallest value in each variable to ensure that the normalized variable is mapped to $[0,1]$. Normalization was achieved using a transformation defined as,

$$y_i' = \frac{\ln(y_i) - \ln(y_{min})}{\ln(y_{max}) - \ln(y_{min})} \tag{4.8}$$

where y_{max} and y_{min} correspond to each dataset feature's maximum and minimum value.

During inference, it is necessary that the predictions are inversely transformed in order to acquire the true value; this encompasses reversing the preprocessing method of the target variables as shown by

$$\hat{y}_i = \exp\{y_i' \cdot (\ln(y_{max}) - \ln(y_{min})) + \ln(y_{min})\}. \tag{4.9}$$

Finally, the data is split into a training, validation, and testing set. The validation set was used to obtain optimal hyperparameters for each model, while the training and testing sets were used to produce the final evaluations of the models. The fractions used for the split were 0.6 for training, 0.2 for validation, and 0.2 for testing. An exception was made for the experiments that utilized linear regression models, in which the data is split into fractions of 0.8 for training and 0.2 for testing.

4.5 Performance Indicators

As all of the developed models share the property of predicting numerical values, it is crucial to use metrics specifically designed for such in order to evaluate the performance of the proposed methods. For numerical value predictions, it is thus only fitting to use regression metrics that involve calculating errors and summarizing them into a score describing their predictive capabilities.

The different methods in this work are tested and compared to each other using five different performance indicators. The indicators include Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (MdAE), Symmetric Mean Absolute Percentage Error (SMAPE), and the Coefficient of Determination (R Squared).

4.5.1 Mean Squared Error

The Mean Squared Error (MSE) metric is computed as the average squared difference between a model's predicted values and the expected target values. Since the metric squares the errors, it effectively turns negative errors into positive values. The MSE metric is proportionally affected by each prediction error, making it prone to outliers as a few large errors result in a large effect on the final metric score.

The metric is defined by

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.10}$$

where y_i is the target value, $\hat{y}_i$ the predicted value, and n the number of data points predicted. If a model predicts all the data points to exactly match the target values, it yields a perfect MSE of 0.0. It is important to note that the performance indicator relies heavily on the specific dataset used for modeling and is thus not suitable for making comparisons to other studies that use different data.

4.5.2 Mean Absolute Error

Dissimilarly to MSE, the Mean Absolute Error (MAE) yields a measurement that has the same unit as the predicted target value. It measures the average absolute distance between a model's predicted values and the expected target value. A perfect MAE value of 0.0 implies that every single prediction matched the expected target value exactly and may suggest that the problem is trivial.

The metric is defined by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.11}$$

where y_i is the target value, $\hat{y}_i$ the predicted value, and n the number of data points predicted. While the MSE tends to magnify the error value due to the square operation, the MAE does not, as it increases linearly. For this reason, the metric does not penalize large outlier errors made by the predictor the same way as MSE. Despite this, the MSE metric is still vulnerable to become skewed by substantial outlier values. A similar but more robust measure is the Median Absolute Error (MdAE) which assesses the median deviation of the predicted values in relation to the measured values. The MdAE can be used jointly with the MAE to enable greater comprehension of a model's behaviour. The metric is defined by

$$MdAE = \text{median}(|y_i - \hat{y}_i|) \tag{4.12}$$

where y_i is the target value and $\hat{y}_i$ the predicted value.

4.5.3 Coefficient of Determination

The Coefficient of Determination, or R^2 value, is a statistical measure used to quantify the degree of linear correlation between the predicted value and the expected target value. The metric provides an idea of how much a model is able to explain the variability of its predictions. An R^2 score of 0.0 suggests that the model is unable to explain the variability around the mean of the response, while a score of 1.0 indicates that the model is able to explain all the variability.

The metric is defined by

$$R^2 = 1 - \frac{SS_{tot}}{SS_{res}} \tag{4.13}$$

where,

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{4.14}$$

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.15}$$

given the target value y_i, the predicted value $\hat{y}_i$, the mean target value $\hat{y}$ and the amount of data points predicted n.

Generally, the larger R^2 value a model has, the better it fits the observations. However, it is important to know that low values do not necessarily correspond to bad fits. Some processes can inherently have larger amounts of unexplained data variation and thus have a lower upper-bound R^2 score compared to other processes [61].

4.5.4 Symmetric Mean Absolute Error

While the Mean Absolute Percentage Error (MAPE) is one of the standard metric used in statistics to measure the relative prediction error in statistics, defined by

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{4.16}$$

it also brings on drawbacks that lead to severe consequences. One of the major shortcomings of MAPE is its heavy emphasis on penalizing negative errors and its inability to handle zero values. The Symmetric Mean Absolute Error (SMAPE) can partially overcome this issue while also provide better protection against outliers [62]. The measure is defined by,

$$SMAPE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} (y_i + \hat{y}_i)}. \tag{4.17}$$

4.5.5 Prediction Ratio

Provided that the data values describing an Internet traffic flow are strictly positive, a relative accuracy of a models' predictions can be obtained based on the ratio between the predicted value and the measured value. This measure facilitates symmetric and unbiased properties making it easy to analyze statistically. By drawing the cumulative distribution of the relative prediction ratio for an entire dataset, it can also be used to interpret how much of the dataset has been predicted accurately within a predefined percental error margin. The measure is defined by,

$$r_i = \frac{\hat{y}_i}{y_i}. \tag{4.18}$$

While the MSE, MAE, and R^2 provide values that enable model evaluation and comparisons, it can be hard to gather an intuitive understanding of their predictive skill solely based on table data. A visualization of the cumulative ratio distribution for a specific target variable facilitates an additional evaluation method that is both easy to understand and comprehensive.

By capturing the cumulative distribution of error ratios across the dataset, the performance of different models become easily comparable as the visualization indicates how much of the dataset is wrongfully predicted by a specific amount.

4.6 Experimental Setup

The main objective of the experiments is to design a supervised machine learning model that predicts the size, duration, and packet count of a network traffic flow, which accounts for the majority of variation within the given dataset. The proposed regression ResNet architecture's results are evaluated against both baselines and traditional supervised machine learning algorithms to investigate its advantages and disadvantages. Since the key element of the ResNet is its skip connections, allowing the network to be deeper without facing the vanishing gradient problem, an identical architecture with the same width and depth but without any skip connections is also studied in order to understand the impact of the skip connections further. This model is referred to as Artificial Neural Network without skip connections (ANN w/o shortcuts) throughout the experimental phase. The machine learning algorithms explored in the experiments are listed in Table 4.4.

Table 4.4 – Regression models used in the experiments.

Regression Technique	**Description**
Baseline Mean	Dataset mean value for target variable.
Baseline Mode	Dataset mode value for target variable.
Linear Regression	Analytically solved linear regression.
Decision Tree Regression	Decision tree regression.
MLP	Multilayer perceptron.
ANN w/o shortcuts	Artificial neural network without skip connections.
ResNet	Residual neural network.

The two naive baselines are established on the mean and mode value of the target variables in each training dataset, thus predicting constant values regardless of the inputs. Naturally, the baselines predict low values as the majority of the flows are small. The errors produced by the baselines thus increase proportionally with larger flows.

The weights for the linear regression model are solved analytically through matrix multiplication, hence requiring no validation set nor hyperparameter optimization. In contrast, the decision tree regression model requires a limit of the maximum depth of the tree to prevent overfitting. The decision tree models used during for experiments use the MSE to measure the quality of each split.

The multilayer perceptron model is defined as a shallow neural network with one hidden layer. The multilayer perceptron aims to serve as a point of reference whenever examining the potential performance improvements of the deeper networks. The width of the hidden layer is treated as a hyperparameter and is optimized for each dataset.

The number of block-stacks in the ResNet architecture (see Figure 4.2) is referred to as the depth of the model. The depth of ResNet is also treated as a hyperparameter together with the constant width of the layers within the blocks.

The MLP, ANN w/o shortcuts, and the ResNet model all utilize the ReLU activation function and use the mean square error (MSE) as a loss function. To prevent the neural networks from overfitting, early stopping is applied during training which observes the validation loss and terminates the training if the neural network has not improved in the last predefined number of epochs. The early stopping is set to 20 epochs due to the large size of the datasets, resulting in a slow incremental change in the validation loss between epochs.

Adaptive Moment Estimation Optimizer

The weights of the neural network models are all updated and optimized using the Adaptive Moment Estimation (Adam) optimization. The Adam optimizer is a stochastic gradient descent method that appropriates estimations of the first- and second-order moments of the loss gradient, allowing networks to capture more information and give fair weights to non-frequent features. At iteration t of the training process, the weights of a neural network $w^{(t)}$ are updated by Adam using the process defined in Equation 4.19. Specifically, the exponential moving averages of the gradient $m_w^{(t)}$ and the squared gradient $v_w^{(t)}$ are updated by the algorithm with the use of hyperparameters that control the exponential decay of the moving averages. The moving averages are estimates of the mean and the uncentered variance of the gradient [63].

$$
\begin{aligned}
m_w^{(t+1)} &\leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)} \\
v_w^{(t+1)} &\leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)(\nabla_w L^{(t)})^2 \\
\hat{m}_w &= \frac{m_w^{(t+1)}}{1-\beta_1^{t+1}} \\
\hat{v}_w &= \frac{v_w^{(t+1)}}{1-\beta_2^{t+1}} \\
w^{(t+1)} &\leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}
\end{aligned}
\tag{4.19}
$$

In the algorithm above, $L^{(t)}$ is the loss function at iteration t, η the learning rate, and the forgetting factors β_1 and β_2 determine how much of the moments of the gradients are neglected each iteration. With β_1^t and β_2^t we denote β_1 and β_2 to the power t. To prevent division by zero, ϵ is set to a small scalar. The Adam optimizer allows the learning rate for each parameter to be adaptive by storing and exponentially decaying the moments of the gradient. Conceptually, the method is described as a ball with momentum rolling down the slope of the loss gradient, but responds to friction, thus preferring flat minima in the error surface [64].

The hyperparameters used in all experiments are listed in Table 4.5. Additional unique hyperparameters for each dataset are optimized through a grid search on the validation set and listed in Chapter 5.

Table 4.5 – Hyperparameters used for models through all experiments.

Model	**Hyperparameter**	**Value**	**Description**
Decision Tree Regression	criterion	MSE	Quality of a split measure.
	min. sample split	2	Minimum number of samples required to split an internal node
MLP, ANN w/o shortcuts, ResNet	loss function	MSE	Function minimized by the optimizer.
	activation	ReLU	Activation function for each hidden layer.
	batch size	8192	Amount of data points for forward pass iteration.
	early stopping	20	Maximum amounts of epochs without performance increase.
	η	0.001	Adam optimizer factor.
	β_1	0.9	Adam optimizer factor.
	β_2	0.999	Adam optimizer factor.
	ϵ	10^{-7}	Prevent division by zero during training.

4.6.1 Experiment I: Hyperparameter Optimization

For each dataset, the maximum depth of the decision tree, width of each hidden layer within the MLP and ResNet model, and depth of the ResNet model (number of dense-identity-identity block stacks visualized in Figure) is optimized using the validation set through grid search. The ANN model inherits the architecture (the same depth and width) of the best ResNet model except for the shortcut connections. The width of the hidden layers in the MLP and ResNet model is set to be constant throughout the network and is chosen to be a multiple of eight. The grid search range for the width and depth of the ResNet is $\{128, 256, 512\}$, and $\{1, 2, ..., 9, 10\}$, while the range for the MLP width is set to $\{32, 64, 128, 256, 512\}$. The maximum depth for the decision tree regression is chosen from the set $\{1, 2, ..., 19, 20\}$. The neural network models are simulated a total of 5 times using the validation set. The hyperparameters of the best average performing model are considered to be optimal.

4.6.2 Experiment II: Flow Predictions & Evaluations

Using all the available pre-processed input features in the datasets (except for source IP and port in DS-4), the regression methods are trained to predict the transformed target variables. The standardized predictions are evaluated using the MAE, MSE, and the R^2 value for each target variable.

To acquire the true values of the predicted target variables, the predicted values are inversely transformed using Equation 4.9 and evaluated using SMAPE, MdAE, and MAE metrics. Since the MAE is susceptible to large outlier errors, the MdAE is used to put the discrepancy caused by the outliers in perspective. The cumulative distribution of each target variable in the test set for the models is then computed and compared to the true values to investigate how well the models are at capturing the intrinsic flow distributions.

Since a flow with a single packet has a duration of zero seconds, most metrics used to estimate the relative error fall short due to the division by zero that occurs. To understand the relative error better, the cumulative distribution of the ratio between the predicted and the measured target variable is investigated for each model and displayed in a figure. The difference in the cumulative probability between the ratio $1 - x$ and $1 + x$ shows how much of the target

set is predicted correctly within a $100 \cdot x\%$ error margin, enabling an intuitive way of understanding the relative error.

By grouping the predicted target variable into classes representing orders of magnitude (powers of ten), the predictions can be further simplified in use-cases where the overall magnitude of the flow is more important than precise flow property values. The predicted and measured variables for the test set in each dataset are consequently organized into their respective order of magnitude (class) and evaluated using classification accuracy. The classification accuracy indicates how well each model estimates the general size of individual flow properties. Naturally, the naive baselines display high accuracies as the majority of flows in the datasets are concentrated within small ranges. A machine learning model that outperforms the naive baseline classification accuracy thus demonstrates the capability of identifying and distinguishing small flows from large flows.

Furthermore, the absolute errors in the test set target predictions made by the ResNet are computed and displayed for each order of magnitude (powers of ten) class. This permits a deeper understanding of the ResNet properties for different sizes of flows. The absolute errors for each class demonstrate the network's capability to handle and predict different flow sizes and indicate if the architecture has difficulties predicting flow properties within certain intervals. The absolute error is displayed for each class using a box and whisker plot in where the box represents the interquartile range, the whiskers represent the minimum and the maximum absolute error, and the line dividing the box represents the median absolute error.

4.6.3 Experiment III: Exclusion of IP Addresses

The third experiment is done on DS-1, DS-2, and DS-3, which involves training the models listed in Table 4.4 without using the source and destination IP address. The ports used to carry out the traffic flow will, however, be included. Since the source IP and ports are excluded in DS-4 due to user anonymization, the dataset is not used for the particular experiment. The purpose of the IP address exclusion is to examine if embedded information in the IP addresses helps the models improve their predictions of flow properties. The results obtained through the exclusion of IP addresses are then used to review whether the proposed models can generalize to arbitrary network traffic flows without learning the specific behaviors of individual IP addresses. The regression

methods are trained to predict the transformed target variables using the pre-processed input features (excluding IP addresses). The normalized predictions are evaluated using the MAE, MSE, and the R^2 value for each target variable. The tangible values of the predicted target variables are acquired through re-transforming the predictions using Equation 4.9, and further evaluated using SMAPE, MdAE, and MAE metrics.

4.6.4 Experiment IV: Monte Carlo Dropout

The fourth experiment is done on all datasets and involves applying 0.2 dropout after each ReLU activation in the optimal ResNet architecture from Section 4.6.1, which stochastically deactivates 20% of the layer nodes during forward passes. Using the available pre-processed input features for each dataset, the optimal ResNet architecture is re-trained to predict the transformed target variables. At inference, with the dropout activated, each data point in the training and testing set is predicted a total of 100 times. The means of the standardized predictions are evaluated using the MAE, MSE, and the R^2 value for each target variable. To acquire the true values of the predicted mean target variables, the predicted values are inversely transformed using Equation 4.9 and evaluated using the SMAPE, MdAE, and MAE metrics.

Using the computed model variance described in Section 4.3, 95% prediction intervals are calculated for each target variable value using Equation 4.1. To evaluate the prediction intervals, the average ratio between the interval length and mean value is computed in a quantity called Mean Prediction Interval (MPI), where a narrow ratio is more desirable. Finally, for each target variable, the fraction of train and test set that is correctly situated in the prediction intervals, referred to as the Prediction Interval Coverage Probability (PICP), is measured [51].

Chapter 5

Results

This chapter aims to present the results obtained from the experimental setup described in Section 4.6 for each of the datasets presented and analyzed in Section 4.4. The main objective is to evaluate the performance of the proposed Residual Neural Network architecture and compare it to traditional machine learning methods and baselines when predicting the packet count, duration, and size of Internet traffic flows. The performance of the Residual Neural Network architecture is then further examined by excluding the available IP addresses from the input space (to investigate generalization properties) and by implementing a Monte Carlo Dropout extension to produce uncertainty estimations. The results are consequently discussed in detail in Chapter 6.

5.1 DS-1: Unicauca Network Flows Dataset

The following section aims to present the results obtained from the experimental setup described in Section 4.6 for each of the regression methods presented in Table 4.4 using the Unicauca Network Flows dataset [8], referred to as DS-1.

5.1.1 Hyperparameter Optimization

The optimal hyperparameters found for each of the regression methods described in Table 4.4 are displayed in Table 5.1. The optimal hyperparameters were obtained through a grid search simulated a total of 5 times for each variation, using the validation set.

Table 5.1 – Optimal hyperparameters found for regression methods in DS-1.

Regression Technique	**Name of Hyperparameters**	**Range**	**Optimal Value**
Linear Regression	NA	NA	NA
Decision Tree Regression	Maximum Depth;	$1, 2, ..., 19, 20$;	11
MLP	Width; Depth;	$32, 64, 128, 256, 512$; 1;	128; 1;
ANN w/o Shortcuts	Width; Depth;	NA NA	128; 2;
ResNet	Width; Depth;	$128, 256, 512$; $1, 2, ..., 9, 10$;	128; 2;

5.1.2 Predictions & Evaluations

Normalized Prediction Results

Tables 5.2, 5.3, and 5.4 exhibit the MAE, MSE, and R^2 obtained by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows for the training and the testing set. The ResNet architecture best predicted the three target variables with similar performance metric values for the target variables corresponding to the total packet count and size of flows. The linear regression model displays the worst performance out of the proposed machine learning methods, and is unable to minimize the prediction errors as efficiently as the neural network models. Evidently, the ResNet's use of shortcut connections promote better performance as the errors are comparably smaller compared to the identical model without

shortcut connections. The decision tree manages to rival the deep neural network without any shortcut connections and performs better than the MLP model, suggesting that higher complexity models do not correspond to more accurate predictions.

Table 5.2 – Results of predicting *total packet count* using transformed targets for DS-1.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.810	0.999	0.000	0.811	1.003	0.000
Baseline Mode	1.050	2.101	-1.103	1.052	2.109	-1.104
Linear Regression	0.464	0.573	0.426	0.465	0.575	0.426
Decision Tree Regression	0.232	0.218	0.782	0.235	0.225	0.775
MLP	0.269	0.258	0.742	0.271	0.262	0.739
ANN w/o Shortcuts	0.210	0.176	0.824	0.226	0.220	0.781
ResNet	**0.194**	**0.141**	**0.859**	**0.213**	**0.191**	**0.810**

Table 5.3 – Results of predicting *total flow size* in bytes, using transformed targets for DS-1.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.847	0.999	0.000	0.848	1.003	0.000
Baseline Mode	1.360	2.849	-1.851	1.363	2.859	-1.851
Linear Regression	0.485	0.550	0.450	0.485	0.551	0.450
Decision Tree Regression	0.237	0.189	0.811	0.240	0.194	0.806
MLP	0.272	0.222	0.778	0.274	0.225	0.776
ANN w/o Shortcuts	0.215	0.149	0.850	0.229	0.186	0.814
ResNet	**0.198**	**0.120**	**0.880**	**0.215**	**0.162**	**0.839**

Table 5.4 – Results of predicting *total flow duration* in seconds, using transformed targets for DS-1.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.839	1.000	0.000	0.839	0.999	0.000
Baseline Mode	0.722	1.521	-0.521	0.723	1.522	-0.523
Linear Regression	0.598	0.683	0.317	0.597	0.681	0.318
Decision Tree Regression	0.406	0.431	0.569	0.407	0.435	0.565
MLP	0.441	0.441	0.559	0.443	0.444	0.555
ANN w/o Shortcuts	0.363	0.353	0.647	0.379	0.390	0.610
ResNet	**0.319**	**0.294**	**0.706**	**0.344**	**0.347**	**0.653**

Unnormalized Prediction Results

Tables 5.5, 5.6, and 5.7 exhibit the SMAPE, MdAE, and MAE obtained by applying the inverse transformation (Equation 4.9) to the predictions made by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows for the training and the testing set. The ResNet architecture best predicted the three target variables with similar performance metric values for the target variables corresponding to the total packet count and size of flows. A disparity is however observable in the model's capability of predicting the duration of flows, indicating that the variable is harder to infer and does not necessarily have a strong relationship to the input features. The median absolute error is observed to be orders of magnitude smaller than the mean absolute error for all the target variables, indicating that the error distribution is skewed to the right, a consequence of the relatively large errors obtained when predicting the properties of large flows.

Table 5.5 – Results of predicting *total packet count* using inversely transformed targets for DS-1.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.514	5.102	87.949	0.514	5.102	86.040
Baseline Mode	0.581	2.000	88.549	0.581	2.000	86.651
Linear Regression	0.289	2.604	84.409	0.289	2.616	82.479
Decision Tree Regression	0.151	0.626	71.648	0.153	0.634	73.281
MLP	0.181	1.034	78.460	0.182	1.048	76.552
ANN w/o Shortcuts	0.143	0.641	73.237	0.149	0.664	74.364
ResNet	**0.135**	**0.655**	**68.723**	**0.144**	**0.687**	**70.479**

Table 5.6 – Results of predicting *total flow size* in bytes, using inversely transformed targets for DS-1.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.661	738.609	$1.08{\cdot}10^{5}$	0.661	743.391	$1.05{\cdot}10^{5}$
Baseline Mode	0.748	272.000	$1.08{\cdot}10^{5}$	0.749	274.000	$1.05{\cdot}10^{5}$
Linear Regression	0.384	527.174	$6.56{\cdot}10^{8}$	0.385	533.187	$1.77{\cdot}10^{7}$
Decision Tree Regression	0.214	105.248	$9.44{\cdot}10^{4}$	0.215	105.757	$9.37{\cdot}10^{4}$
MLP	0.246	137.626	$1.03{\cdot}10^{5}$	0.247	139.297	$9.99{\cdot}10^{4}$
ANN w/o Shortcuts	0.202	101.314	$9.60{\cdot}10^{4}$	0.209	104.981	$9.52{\cdot}10^{4}$
ResNet	**0.191**	**95.802**	$\mathbf{8.93{\cdot}10^{4}}$	**0.200**	**101.066**	$\mathbf{9.01{\cdot}10^{4}}$

Table 5.7 – Results of predicting *total flow duration* in seconds, using inversely transformed targets for DS-1.

	Train			Test		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.838	3.275	54.276	0.837	3.275	54.294
Baseline Mode	0.937	0.175	53.609	0.936	0.177	53.635
Linear Regression	0.792	3.677	52.798	0.792	3.703	52.800
Decision Tree Regression	0.707	0.556	46.010	0.707	0.584	46.296
MLP	0.720	1.373	47.119	0.720	1.390	47.368
ANN w/o Shortcuts	0.681	0.537	43.310	0.686	0.563	44.404
ResNet	**0.650**	**0.497**	**39.540**	**0.658**	**0.522**	**41.394**

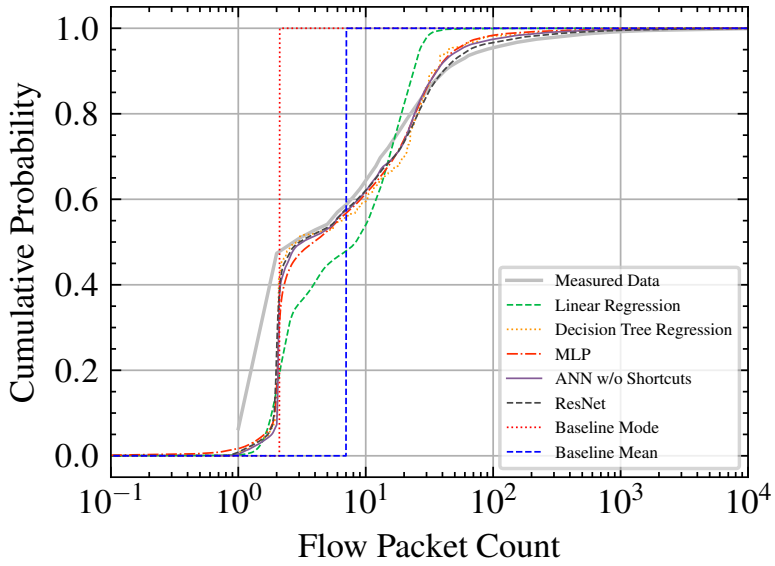


Figure 5.1 – Cumulative distribution of the measured and the predicted packet counts within the flows of the DS-1 test set.

Cumulative Distribution of Predictions

Figures 5.1, 5.2, and 5.3, display the cumulative distribution of the predicted test set target variables for the regression methods listed in Table 4.4. The proposed neural network and decision tree methods demonstrate a good ability to capture the test set distribution for the packet-count and flow size variables, compared to the baselines and the linear regression model. In contrast, a significant disparity can be observed between the cumulative distribution of the predicted flow durations (see Figure 5.3) and the true distribution. The

dissimilarity indicates that the duration of a flow is tough for the proposed methods to predict accurately. Because the flow duration target is a continuous variable, the most frequent value in the test set is zero due to flows containing a single packet. The cumulative distribution for the mode baseline thus stretches infinitely to the left in Figure 5.3 due to the logarithmic x-axis.

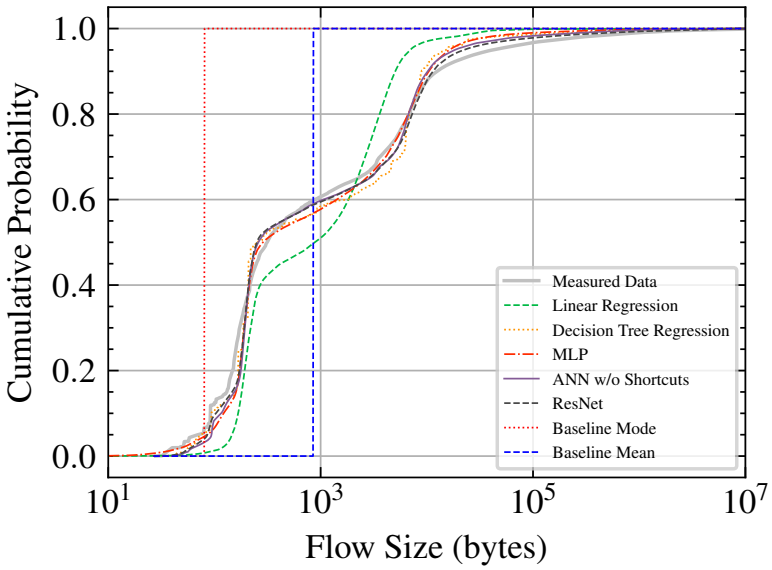


Figure 5.2 – Cumulative distribution of the measured and the predicted flow size within the DS-1 test set.

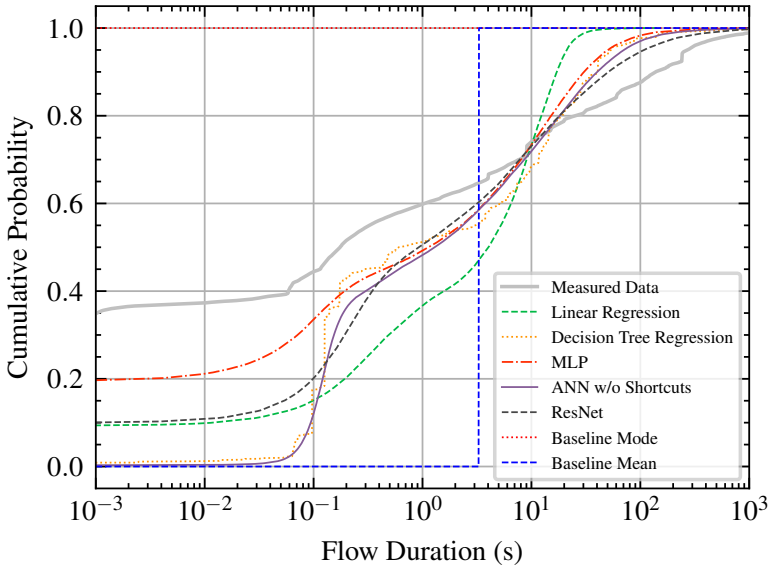


Figure 5.3 – Cumulative distribution of the measured and the predicted flow duration within the DS-1 test set.

Cumulative Distributions of Prediction Ratios

Figures 5.4, 5.5, and 5.6, display the cumulative distribution of the prediction ratios obtained from the test set target variable predictions yielded by the regression methods listed in Table 4.4. The difference in the cumulative probability between the ratio $1 - x$ and $1 + x$ shows how much of the target set is predicted correctly within a $100 \cdot x\%$ error margin, enabling an intuitive way of understanding the relative error. The proposed neural network and decision tree methods demonstrate the ability to predict approximately $\sim 60\%$ of the packet-count and $\sim 40\%$ of the flow size variables within 20% error margin. In contrast, both baselines display capturing only a small fraction of the test set within the same error margin. While the linear regression method manages to outperform both baselines, it fails to rival the performance of the remaining machine learning methods. A significant performance disparity is observed in the test set flow duration prediction ratios (see Figure 5.6), in where the machine learning methods - except for linear regression - manage to predict $\sim 20\%$ of the test set duration variables within a 50% error margin. The dissimilarity indicates that the duration of a flow is particularly hard for the proposed methods to predict accurately. Because the flow duration target is a continuous variable, the most frequent value in the test set is zero due to flows containing a single packet. The cumulative ratio distribution for the flow duration mode baseline thus remains at 0 in the x-axis, making it not appear in Figure 5.6.

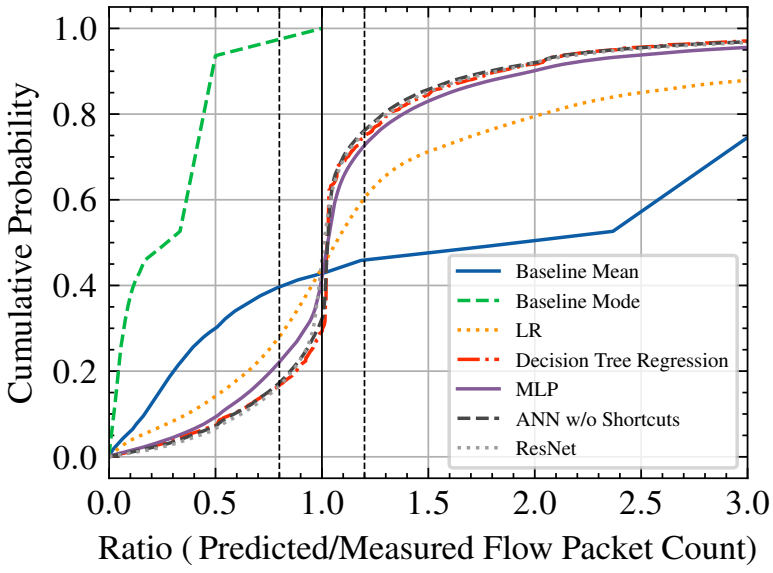


Figure 5.4 – Cumulative distribution of the ratio between the measured and predicted packet count within flows of the DS-1 test set.

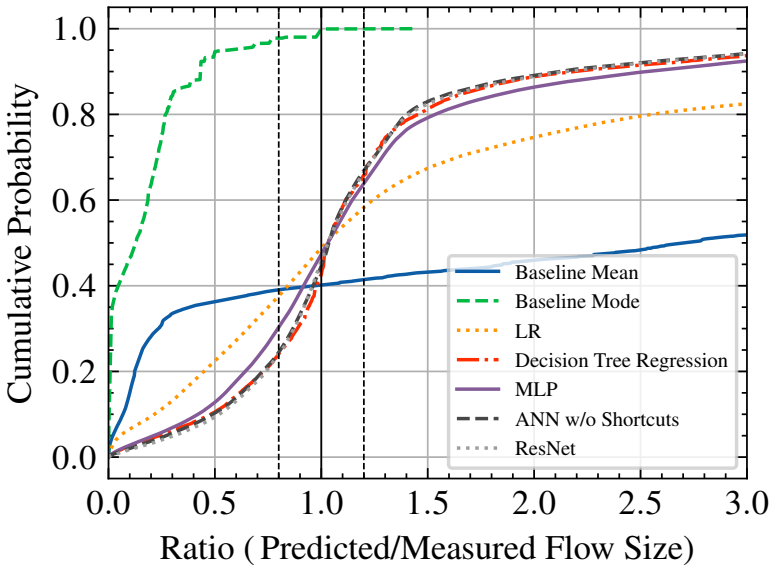


Figure 5.5 – Cumulative distribution of the ratio between the measured and predicted flow size within the DS-1 test set.

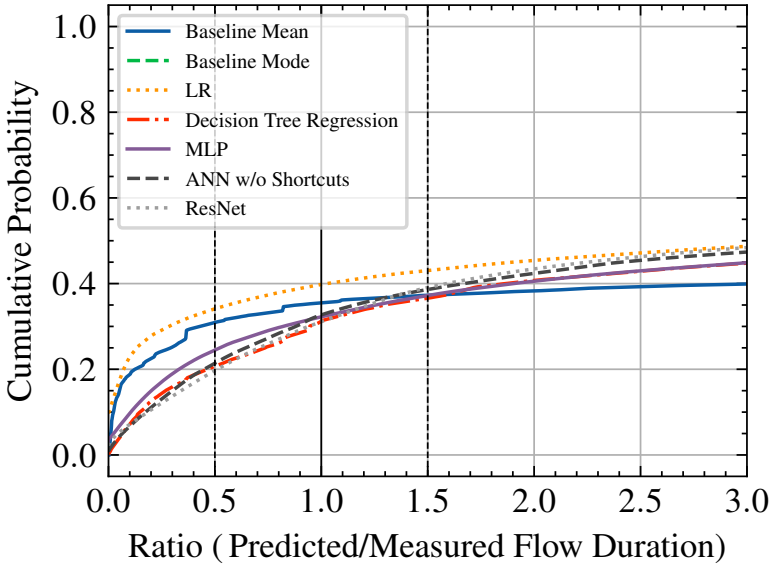


Figure 5.6 – Cumulative distribution of the ratio between the measured and predicted flow duration within the DS-1 test set.

Symmetric Mean Absolute Percentage Errors

Figure 5.7 displays the symmetric mean absolute percentage errors obtained by the different regression methods listed in Table 4.4 on the test set target variables. The proposed machine learning methods exhibit smaller errors compared to the baselines. The ResNet architecture achieves the lowest SMAPE metric value for all three of the target variables, reducing the SMAPE with $\sim 35\%$ for packet counts, $\sim 40\%$ for total flow size, and $\sim 15\%$ for flow durations, compared to the baselines. The small SMAPE decrease in the flow duration target variable indicates that the variable is particularly hard for the proposed method to predict accurately, as shown by previous results.

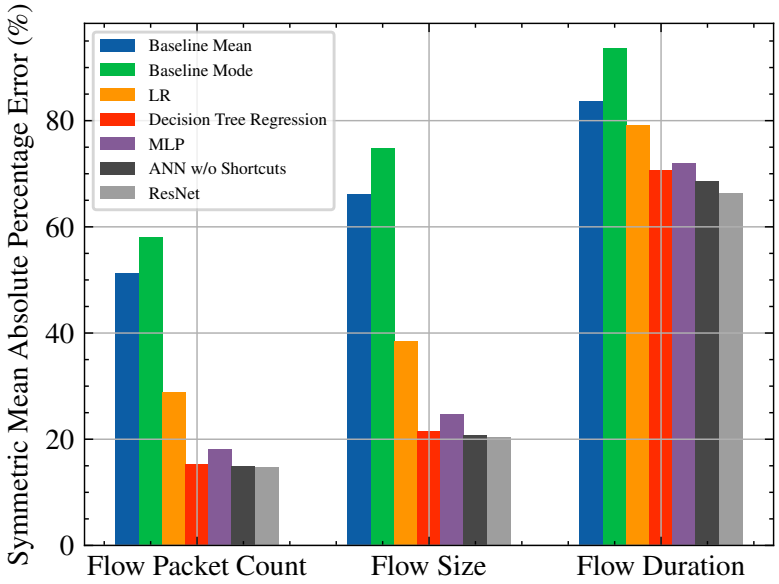


Figure 5.7 – The symmetric mean absolute percentage error within the DS-1 test set for the different regression techniques and target variables.

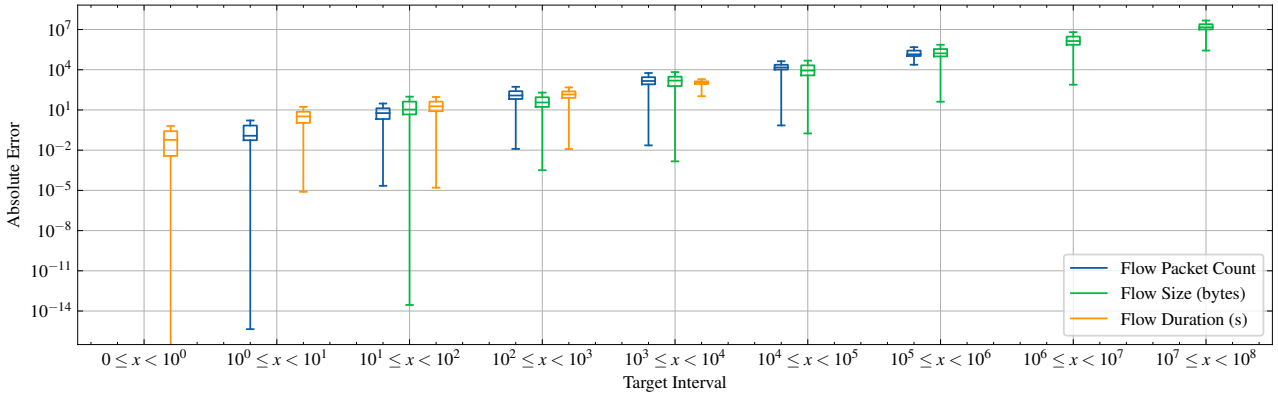


Figure 5.8 – Quartile and median visualization of the absolute error of the ResNet test set predictions for different target intervals in DS-1.

ResNet Absolute Error for Target Intervals

The absolute error produced by the ResNet test predictions for each of the target variables, binned into powers of ten, is displayed in Figure 5.8. The corresponding distribution of the test set is exhibited in Figure 5.9. For each interval, the absolute error is displayed using a box and whisker plot where the box represents the interquartile range. The whiskers represent the minimum and the maximum absolute error, and the line dividing the box represents the median absolute error. The median absolute errors for each target interval have the common behavior of being at most smaller than the order of magnitude of the target interval, except for the smallest target interval.

The trend of long lower whiskers indicates that the ResNet architecture can predict targets with relatively low absolute errors and rarely wrongfully predicts target variables with errors outside of the actual order of magnitude as indicated by the short top whiskers, marking the maximum error. The target interval distribution of the test set demonstrates that the ResNet architecture can capture complex relationships in flows of large size and produce similar results as for smaller flows despite the lack of data quantity.

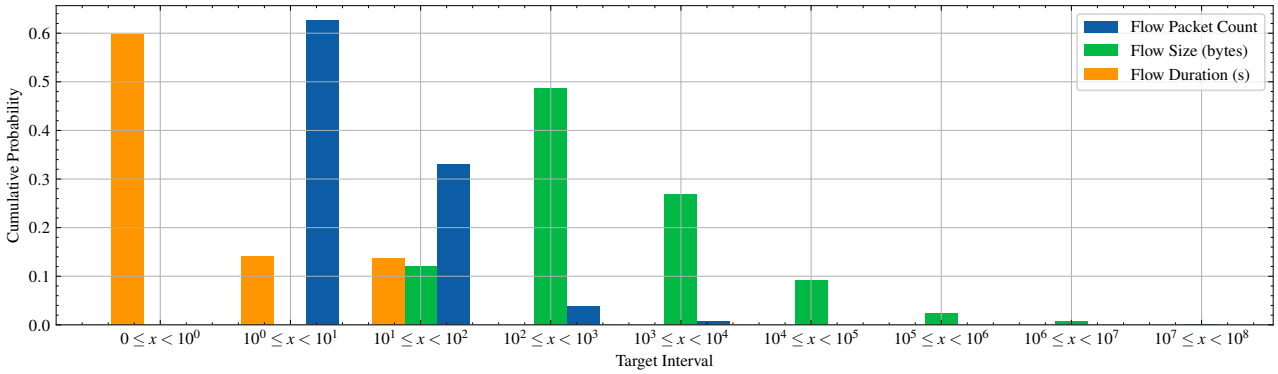


Figure 5.9 – DS-1 test set target distribution grouped into the corresponding order of magnitude.

Classification Accuracies

By grouping the predicted target variable into classes representing orders of magnitude (powers of ten), the predictions made by the methods in Table 4.4 are further simplified to investigate the overall magnitudes of the flow predictions rather than the precise property values. The predicted and measured variables in the test set are organized into their respective class and evaluated using classification accuracy. The corresponding classification accuracy for each target variable is displayed in Figure 5.10. Naturally, the naive mean baseline exhibits high accuracies - especially for the flow duration target variable - as the majority of flows in the dataset have a duration of less than one second (see Figure 5.9). The decision tree regression model outperforms the MLP for target variables *packet count* and *flow size*. The ResNet model is able to outperform all machine learning models with a slight margin for packet count and flow size and a larger margin for the flow duration. The order of magnitude of the packet count and flow size can evidently be predicted with $\sim 90\%$ and $\sim 80\%$ accuracy. Furthermore, the flow duration accuracy is increased with $\sim 20\%$ by the ResNet model compared to the mean and mode baseline.

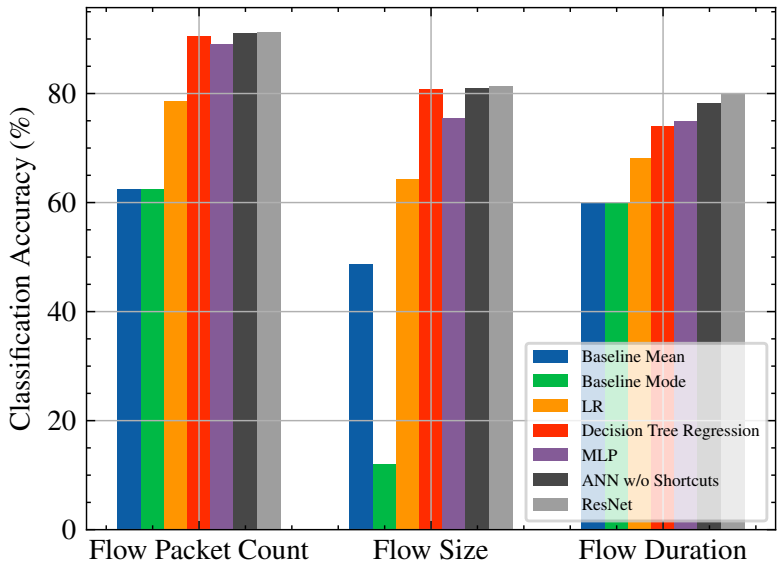


Figure 5.10 – The predicted order of magnitude accuracy within the DS-1 test set for the different regression techniques and target variables.

Temporal Conditions

The training and inference phases were timed and registered to cover the temporal aspect of the main research question. In a practical setting, the regression methods presented in Table 4.4 might need to be retrained as more data is collected. Having insight into the time it takes to train a model or infer a data point is thus beneficial. Table 5.8 displays the number of trainable parameters, stopping epoch, total training time, and the test set prediction time for each of the proposed machine learning models. The results in Table 5.8 were obtained using a machine equipped with an Nvidia Tesla V100 graphics processing unit (GPU), an Intel Xeon 2.00GHz central processing unit (CPU), and 32 GB of random-access memory (RAM). The number of parameters displayed for the decision tree regression techniques refers to its number of leaves. NA in Table 5.8 stands for Not Applicable.

The ANN without shortcuts has the same depth and width as the optimal residual model (ResNet) but with slightly fewer parameters due to the single dense layer utilized in the dense block shortcut (see Figure 4.2(b)) being removed. The total amount of training epochs required to train the ResNet using early stopping with 20 epoch patience extended that of the ANN without shortcuts with 42 epochs. The optimal residual model is trained for approximately 9 minutes through 123 epochs, a significant increase in training time compared to the linear regression and decision tree. As the proposed ResNet architecture is a much more complex model compared to a linear regressor, it is bound to require more computational resources to train. The need for computational resources is not only tied to the training process but also to the inference phase, as each data point prediction requires a complete forward pass through the entire neural network.

It can be observed from Table 5.8 that the neural network predictions required computational time two orders of magnitudes larger than the linear regression and decision tree predictions. The test set used at the prediction phase consisted of 540,967 data points, suggesting that an individual data point requires on average $5.306 \cdot 10^{-5}$ seconds to be inferred by the ResNet. The median and mean packet inter-arrival time of flows in DS-1 were measured to be 0.039 and 3.789 seconds. The measured packet inter-arrival times indicate that large neural networks can infer most flows within the time interval between the first and second packet in a flow. Model predictions made before the arrival of the second packet can thus save approximately $3.789 \cdot n$ seconds on average, compared to methods requiring warm-up periods (where n corresponds to the number of packets needed to be observed before a prediction can be made).

Table 5.8 – Temporal properties and parameter count for the different regression techniques used on DS-1.

Regression Technique	Parameters	Stopping Epoch	Training Time	Prediction Time (s)
Linear Regression	100	NA	00:00:08	0.239
Decision Tree	1 934	NA	00:00:28	0.297
MLP	13 315	213	00:04:19	10.489
ANN w/o Shortcuts	303 235	81	00:06:17	28.406
ResNet	333 699	123	00:09:18	28.703

5.1.3 ResNet Monte Carlo Dropout

The Monte Carlo Dropout method was applied to the best ResNet architecture with 0.2 dropout following every ReLU activation, as explained in Section 4.6.4. A total of 100 forward passes were conducted for each data point in the training and testing set. Table 5.9 displays the MAE, MSE, and R^2 metric values computed from the normalized mean target predictions. Table 5.10 displays the SMAPE, MdAE, and MAE metrics computed from the inversely transformed mean target predictions (using Equation 4.9). Similarly to Section 5.1.2 the Monte Carlo Dropouot ResNet demonstrate the ability to predict packet count and flow size variables with comparable relative errors seen in Table 5.9 and the SMAPE metric in Table 5.10. The large disparity in performance between the flow duration and the other variables seemingly persists when applying the Monte Carlo Dropout method to the proposed ResNet architecture, mostly visible in the large SMAPE value difference in Table 5.10. Overall, a slight increase in all performance metric values can be observed when compared to the values obtained in Section 5.1.2 from the same architecture without any dropout.

Table 5.9 – Mean MC Dropout ResNet results using transformed targets for DS-1.

	Train			**Test**		
Target Variable	MAE	MSE	R^2	MAE	MSE	R^2
Packet Count	0.179	0.111	0.889	0.210	0.185	0.815
Flow Size	0.186	0.098	0.902	0.212	0.157	0.843
Flow Duration	0.288	0.239	0.761	0.333	0.331	0.669

Table 5.10 – Mean MC Dropout ResNet results using inversely transformed targets for DS-1.

	Train			**Test**		
Target Variable	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Packet Count	0.129	0.585	63.006	0.143	0.615	68.782
Flow Size *(bytes)*	0.184	86.38	83199.163	0.198	92.015	90454.687
Flow Duration *(s)*	0.639	0.404	37.242	0.655	0.43	40.67

Table 5.11 displays the properties of the 95% prediction intervals produced by the stochastic forward passes during inference, in terms of Mean Prediction Interval (MPI) and Prediction Interval Coverage Probability (PICP) metrics defined in Section 4.3. For the target variables packet count and flow size, the MPI is relatively narrow and indicates good optimality. However, the fact that the PICP values do not reach 95% undermines the validity of the intervals [51]. The duration target variable intervals display neither good optimality nor validity, as the coverage is well below 95% and the mean intervals are extensively large. The large mean intervals for flow durations are most likely due to the model's inability to predict small values accurately, which dominate the data distribution.

Table 5.11 – MC Dropout ResNet model uncertainty results for DS-1.

	Train		**Test**	
Target Variable	MPI	PICP	MPI	PICP
Packet Count	$\pm$33.029%	75.120%	$\pm$34.491%	74.194%
Flow Size *(bytes)*	$\pm$50.500%	61.614%	$\pm$52.620%	60.828%
Flow Duration *(s)*	$\pm$162.548%	73.180%	$\pm$163.104%	72.165%

5.1.4 IP Address Exclusion

Tables 5.12 and 5.13 exhibit the performance metrics of the test set predictions made by the ResNet architecture after training with and without the source and destination IP addresses as inputs. Table 5.12 describes the metrics computed from the normalized targets, while Table 5.13 displays the metrics computed from the inversely transformed targets (using Equation 4.9). The ResNet architecture exhibits an evident decrease in performance when IP addresses are excluded as inputs, especially when tasked to infer flow durations. The SMAPE metric increases by approximately 6% for the packet count and the flow size variables whenever the IP addresses are excluded, see Table 5.13. The flow duration variable exhibits a larger increase in the SMAPE metric whenever IPs are excluded. The decrease in performance indicates that the IP addresses contain valuable information that helps the proposed method to infer the properties of a flow more competently. In particular, it is evident that the flow duration predictions are influenced by IP addresses more than the rest of the variables, suggesting that the duration of a flow is largely influenced by user and application-specific characteristics.

Table 5.12 – ResNet results using transformed test set targets for DS-1 with and without IP addresses as inputs.

	with IPs			**without IPs**		
Target Variable	MAE	MSE	R^2	MAE	MSE	R^2
Packet Count	0.213	0.191	0.810	0.329	0.384	0.617
Flow Size	0.215	0.162	0.839	0.320	0.334	0.667
Flow Duration	0.344	0.347	0.653	0.53	0.637	0.363

Table 5.13 – ResNet results using inversely transformed test set targets for DS-1 with and without IP addresses as inputs.

	with IPs			**without IPs**		
Target Variable	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Packet Count	0.144	0.687	70.479	0.204	1.396	117.076
Flow Size *(bytes)*	0.200	101.066	91042.398	0.267	190.052	167007.752
Flow Duration *(s)*	0.658	0.522	41.394	0.747	2.329	51.407

5.2 DS-2: UNSW-NB15 Network Dataset

The following section aims to present the results obtained from the experimental setup described in Section 4.6 for each of the regression methods presented in Table 4.4 using the UNSW-NB15 Network dataset [53], referred to as DS-2. The results obtained from this dataset differ significantly from the rest of the datasets. The reasons for the difference is explained in Section 6.3.

5.2.1 Hyperparameter Optimization

The optimal hyperparameters found for each of the regression methods described in Table 4.4 are displayed in Table 5.14. The optimal hyperparameters were obtained through a grid search simulated a total of 5 times for each variation, using the validation set.

Table 5.14 – Optimal hyperparameters found for regression methods in DS-2.

Regression Technique	Name of Hyperparameters	Range	Optimal Value
Linear Regression	NA	NA	NA
Decision Tree Regression	Maximum Depth;	$1, 2, ..., 19, 20$;	9
MLP	Width; Depth;	$32, 64, 128, 256, 512$; 1;	128; 1;
ANN w/o Shortcuts	Width; Depth;	NA NA	512; 2;
ResNet	Width; Depth;	$128, 256, 512$; $1, 2, ..., 9, 10$;	512; 2;

5.2.2 Predictions & Evaluations

Normalized Prediction Results

Tables 5.15, 5.16, and 5.17 exhibit the MAE, MSE, and R^2 obtained by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows using the training and testing set. Out of the proposed machine learning methods, the decision tree model is able to best predicted the three target variables with similar performance metric values for the target variables corresponding to the total packet count

and size of flows. The decision tree manages to outperform the neural network architectures on all metrics for the test set while the ResNet architecture exhibits the best results on the training set, possibly implying a slight case of overfitting despite the use of early stopping. Section 6.3 further discusses the discrepancies found in the results between datasets.

Table 5.15 – Results of predicting *total packet count* using transformed targets for DS-2.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.829	1.000	0.000	0.829	1.001	0.000
Baseline Mode	2.052	5.210	-4.211	2.051	5.209	-4.206
Linear Regression	0.377	0.296	0.704	0.377	0.296	0.704
Decision Tree Regression	**0.038**	0.015	0.985	**0.039**	**0.016**	**0.984**
MLP	0.071	0.020	0.980	0.071	0.021	0.979
ANN w/o Shortcuts	0.082	0.031	0.969	0.096	0.045	0.955
ResNet	0.040	**0.006**	**0.994**	0.055	0.017	0.983

Table 5.16 – Results of predicting *total flow size* in bytes, using transformed targets for DS-2.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.830	1.000	0.000	0.830	1.001	0.000
Baseline Mode	2.336	6.459	-5.461	2.336	6.460	-5.450
Linear Regression	0.358	0.284	0.715	0.357	0.284	0.716
Decision Tree Regression	0.046	0.019	0.981	**0.047**	**0.020**	**0.980**
MLP	0.080	0.025	0.975	0.081	0.025	0.975
ANN w/o Shortcuts	0.080	0.027	0.973	0.098	0.044	0.956
ResNet	**0.035**	**0.007**	**0.993**	0.054	0.022	0.978

Table 5.17 – Results of predicting *total flow duration* in seconds, using transformed targets for DS-2.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.635	0.999	0.000	0.636	1.003	0.000
Baseline Mode	0.490	1.240	-0.240	0.490	1.243	-0.239
Linear Regression	0.476	0.704	0.296	0.476	0.704	0.298
Decision Tree Regression	0.237	0.301	0.699	**0.239**	**0.307**	**0.694**
MLP	0.265	0.335	0.666	0.267	0.344	0.653
ANN w/o Shortcuts	0.197	0.211	0.790	0.278	0.397	0.599
ResNet	**0.128**	**0.180**	**0.821**	0.262	0.399	0.597

Unnormalized Prediction Results

Tables 5.18, 5.19, and 5.20 exhibit the SMAPE, MdAE, and MAE obtained by applying the inverse transformation (Equation 4.9) to the predictions made by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows using the training and the testing set.

A disparity can be observed between the evaluation metric values for the three target variables, in which the ResNet and the decision tree model minimize different error metrics. The symmetric mean absolute percentage error for the test set is distinctly always best reduced by the decision tree model. The evaluation values obtained by the machine learning models for the flow duration are observably worse than for the remaining target variables, expressing that the flow duration is hard to infer and does not necessarily have a strong relationship to the input features, compared to a flows packet count and size. Section 6.2 further discusses the discrepancies found in the results between target variables.

Table 5.18 – Results of predicting *total packet count* using inversely transformed targets for DS-2.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.488	31.043	82.142	0.488	31.043	82.225
Baseline Mode	0.871	35.000	95.406	0.870	35.000	95.484
Linear Regression	0.246	23.423	63.427	0.246	23.461	63.525
Decision Tree Regression	**0.026**	**0.715**	9.412	**0.027**	**0.715**	**9.536**
MLP	0.051	2.143	12.998	0.051	2.151	13.057
ANN w/o Shortcuts	0.058	1.484	16.725	0.066	1.723	19.190
ResNet	0.030	0.998	**7.317**	0.040	1.143	10.254

Table 5.19 – Results of predicting *total flow size* in bytes, using inversely transformed targets for DS-2.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.601	6094.729	49696.144	0.601	6094.729	49863.145
Baseline Mode	0.930	9202.000	51903.990	0.930	9202.000	52072.328
Linear Regression	0.305	5430.765	45007.229	0.305	5437.453	45170.160
Decision Tree Regression	0.044	**103.784**	7944.436	**0.045**	**103.482**	8063.185
MLP	0.080	726.027	10070.642	0.080	729.362	10174.589
ANN w/o Shortcuts	0.079	385.578	11751.865	0.094	433.389	13192.715
ResNet	**0.035**	306.537	**5744.547**	0.052	357.096	**7523.535**

Table 5.20 – Results of predicting *total flow duration* in seconds, using inversely transformed targets for DS-2.

	Train			Test		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.750	0.275	0.828	0.750	0.275	0.819
Baseline Mode	0.994	0.026	0.741	0.993	0.026	0.732
Linear Regression	0.700	0.141	0.699	0.700	0.141	0.690
Decision Tree Regression	**0.365**	0.038	0.507	**0.366**	0.038	0.500
MLP	0.551	0.042	0.536	0.552	0.043	0.520
ANN w/o Shortcuts	0.528	0.039	0.408	0.558	0.041	0.522
ResNet	0.383	**0.021**	**0.307**	0.460	**0.026**	**0.492**

Cumulative Distribution of Predictions

Figures 5.11, 5.12, and 5.13, display the cumulative distribution of the predicted test set target variables for the regression methods listed in Table 4.4. The proposed neural network and decision tree methods demonstrate a good ability to capture the test set distribution for the packet-count and flow size variables, compared to the baselines and the linear regression model. In contrast, a significant disparity can be observed between the cumulative distribution of the predicted flow durations (see Figure 5.13) and the true distribution.

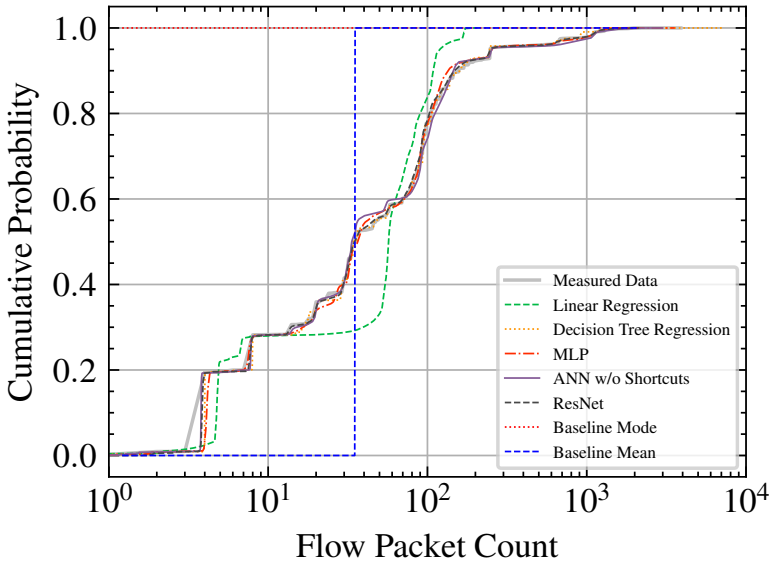


Figure 5.11 – Cumulative distribution of the measured and the predicted packet counts within the flows of the DS-2 test set.

The decision tree model best captures the step-wise nature of the true distribution. The cumulative distribution for the mode baseline stretches infinitely to the left in Figure 5.13 due to the logarithmic x-axis and the mode flow duration being zero.

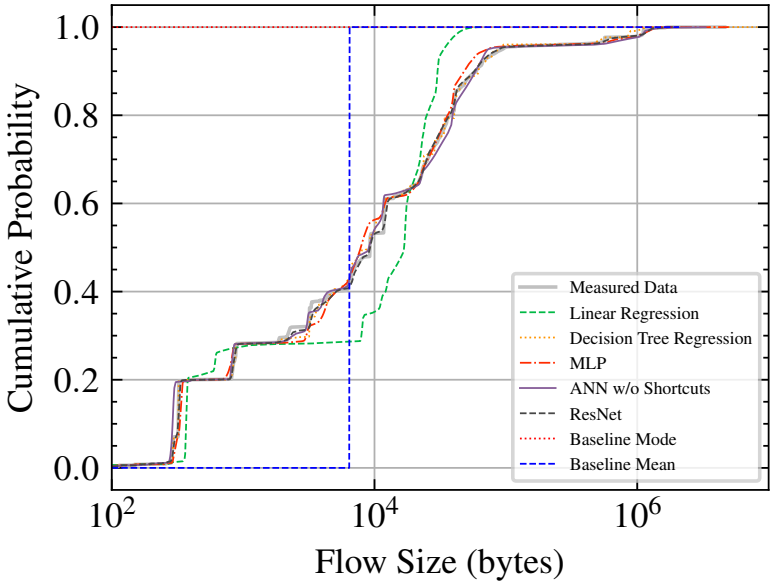


Figure 5.12 – Cumulative distribution of the measured and the predicted flow size within the DS-2 test set.

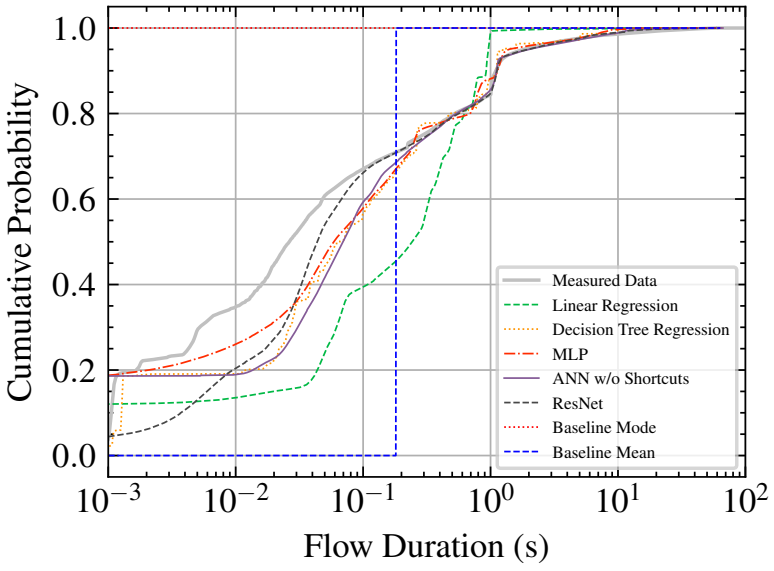


Figure 5.13 – Cumulative distribution of the measured and the predicted flow duration within the DS-2 test set.

Cumulative Distributions of Prediction Ratios

Figures 5.14, 5.15, and 5.16, display the cumulative distribution of the prediction ratios obtained from the test set target variable predictions yielded by the regression methods listed in Table 4.4. The difference in the cumulative probability between the ratio $1 - x$ and $1 + x$ shows how much of the target set is predicted correctly within a $100 \cdot x\%$ error margin, enabling an intuitive way of understanding the relative error. The proposed neural network and decision tree methods demonstrate the ability to predict approximately $\sim 90\%$ of the packet-count and total size variables within a 20% error margin. In contrast, both baselines display capturing only a small fraction of the test set within the same error margin. While the linear regression method manages to outperform both baselines, it fails to rival the performance of the remaining machine learning methods. A significant performance disparity is observed in the test set flow duration prediction ratios (see Figure 5.16), in where the best performing model, the decision tree, manages to predict $\sim 30\%$ of the test set duration variables within a 50% error margin. The discrepancy in performance indicates that the duration of a flow is particularly hard for the proposed methods to predict accurately. Since the flow duration target is a continuous variable and the most frequent value in the test set is zero due to flows containing a single packet, the cumulative ratio distribution for the mode baseline remains as a vertical line at 0 in the x-axis, making it not appear in Figure 5.16.

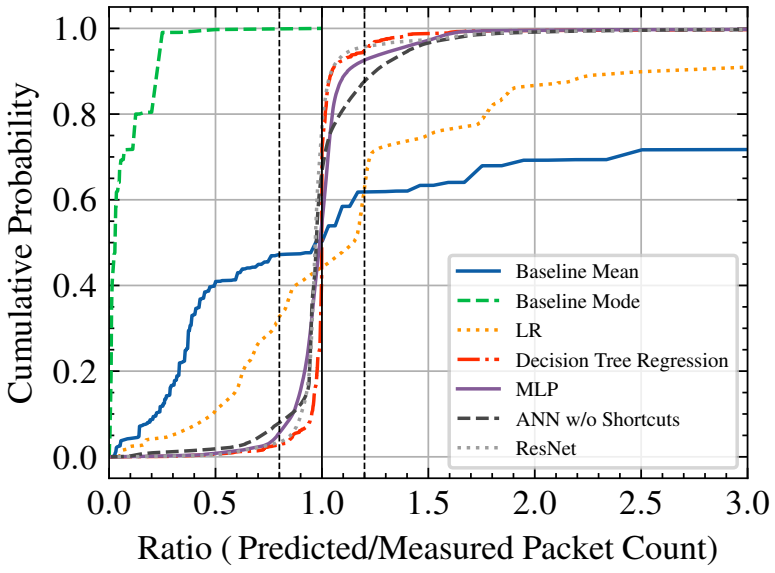


Figure 5.14 – Cumulative distribution of the ratio between the measured and predicted packet count within flows of the DS-2 test set.

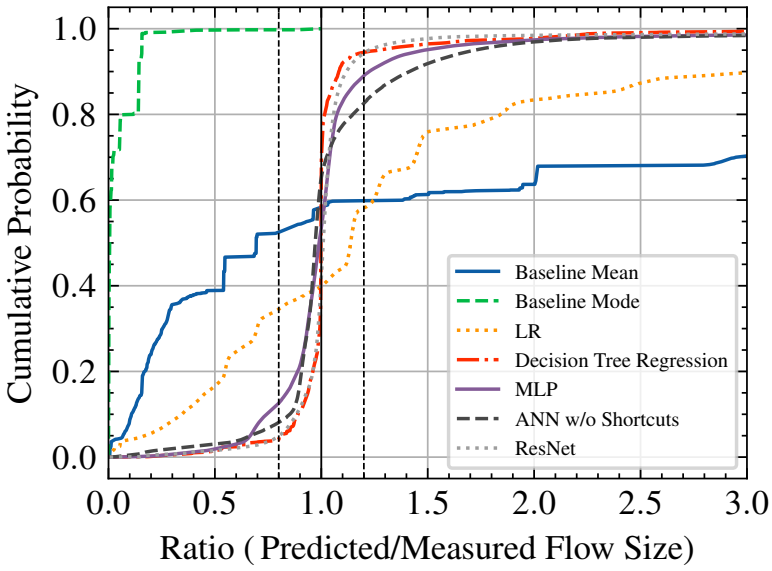


Figure 5.15 – Cumulative distribution of the ratio between the measured and predicted flow size within the DS-2 test set.

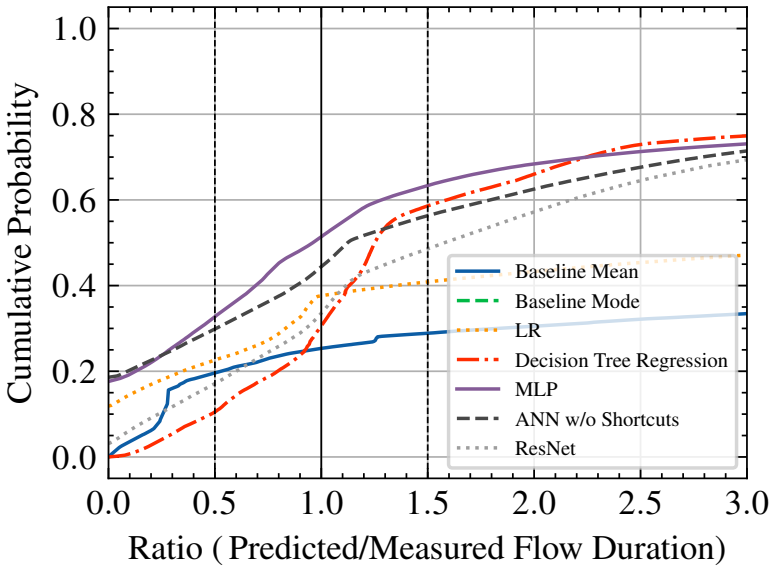


Figure 5.16 – Cumulative distribution of the ratio between the measured and predicted flow duration within the DS-2 test set.

Symmetric Mean Absolute Percentage Errors

Figure 5.17 displays the symmetric mean absolute percentage errors obtained by the different regression methods listed in Table 4.4 on the test set target variables. The proposed machine learning methods exhibit distinctly smaller errors compared to the baselines. The decision tree model achieves the lowest SMAPE metric value for all three of the target variables, reducing the SMAPE with $\sim 45\%$ for packet counts, $\sim 50\%$ for total flow size, and $\sim 40\%$ for flow durations, compared to the mean baseline. The ResNet architecture achieves similar performance except for the flow duration target, in which the decision tree outperforms the ResNet model with an additional $\sim 10\%$ SMAPE decrease.

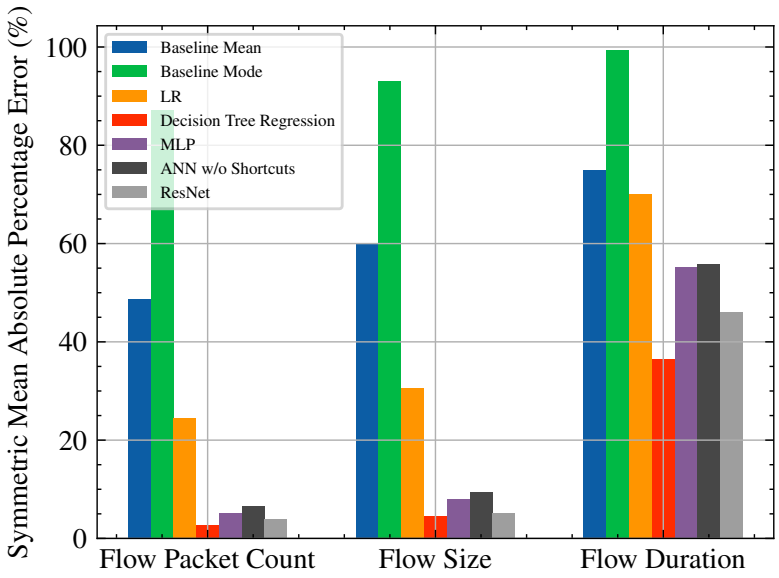


Figure 5.17 – The symmetric mean absolute percentage error within the DS-2 test set for the different regression techniques and target variables.

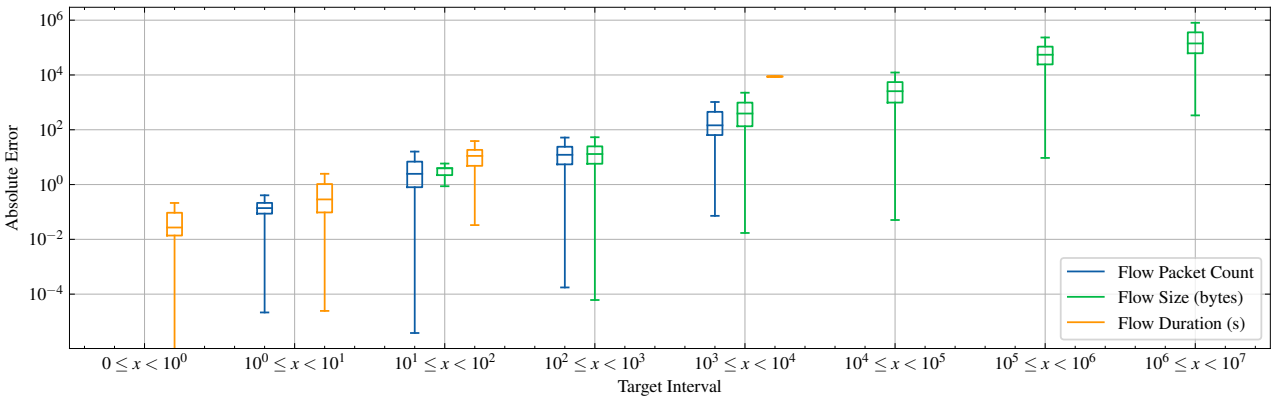


Figure 5.18 – Quartile and median visualization of the absolute error of the ResNet test set predictions for different target intervals in DS-2.

ResNet Absolute Error for Target Intervals

The absolute error produced by the ResNet test predictions for each of the target variables, binned into powers of ten, is displayed in Figure 5.18. The corresponding distribution of the test set is exhibited in Figure 5.19. For each interval, the absolute error is displayed using a box and whisker plot where the box represents the interquartile range. The whiskers represent the minimum and the maximum absolute error, and the line dividing the box represents the median absolute error. The maximum absolute errors for each target interval have the common behavior of being at most smaller than the order of magnitude of the target interval. The trend of long lower whiskers indicates that the ResNet architecture can predict targets with relatively low absolute errors and rarely wrongfully predicts target variables with errors outside of the actual order of magnitude as indicated by the short top whiskers. The target interval distribution of the test set demonstrates that the ResNet architecture can capture complex relationships in flows of large size and produce similar results as for smaller flows despite the lack of data quantity.

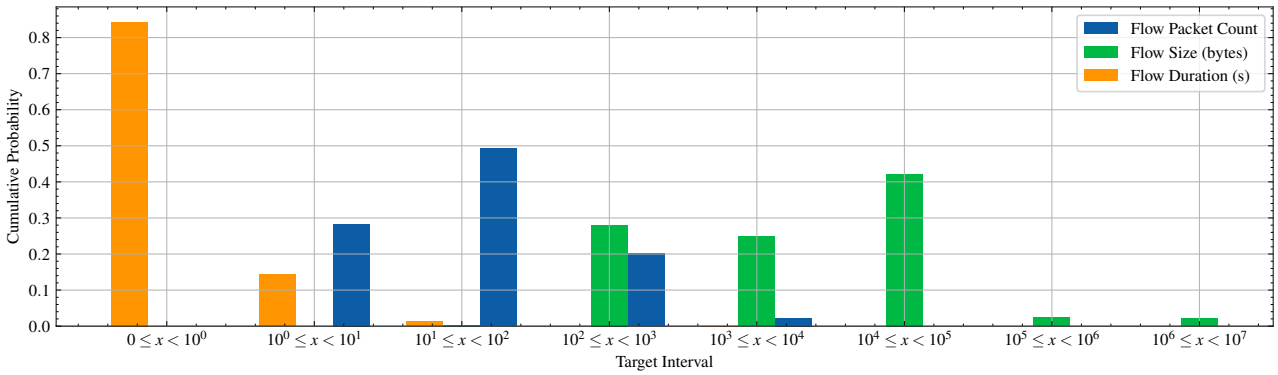


Figure 5.19 – DS-2 test set target distribution grouped into the corresponding order of magnitude.

Classification Accuracies

By grouping the predicted target variable into classes representing orders of magnitude (powers of ten), the predictions made by the methods in Table 4.4 are further simplified to investigate the overall magnitudes of the flow predictions rather than the precise property values. The predicted and measured variables in the test set are organized into their respective class and evaluated using classification accuracy. The corresponding classification accuracy for each target variable and method is displayed in Figure 5.20. Naturally, the naive mean baseline exhibits high accuracies - especially for the flow duration target variable - as the majority of flows in the dataset have a duration of less than one second and are concentrated within small size ranges. The ResNet and decision tree models outperform the remaining machine learning models for all target variables with similar performances. The order of magnitude of the packet count, flow size, and flow duration can evidently be predicted with over 95% accuracy.

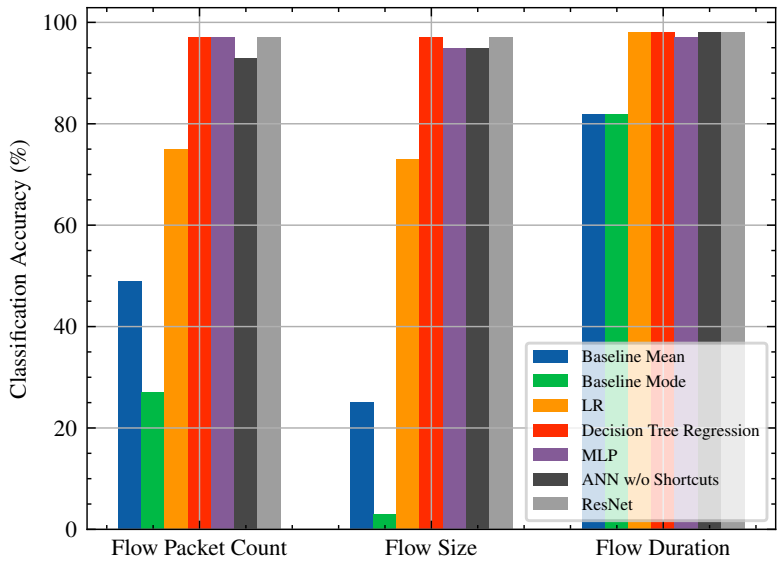


Figure 5.20 – The predicted order of magnitude accuracy within the DS-2 test set for the different regression techniques and target variables.

Temporal Conditions

The training and inference phases were timed and registered to cover the temporal aspect of the main research question. In a practical setting, the regression methods presented in Table 4.4 might need to be retrained as more data is collected. Having insight into the time it takes to train a model or infer a data point is thus beneficial. Table 5.21 displays the number of trainable parameters, stopping epoch, total training time, and the test set prediction time for each of the proposed machine learning models. The results in Table 5.21 were obtained using a machine equipped with an Nvidia Tesla V100 graphics processing unit (GPU), an Intel Xeon 2.00GHz central processing unit (CPU), and 32 GB of random-access memory (RAM). The number of parameters displayed for the decision tree regression techniques refers to its number of leaves. NA in Table 5.21 stands for Not Applicable.

The ANN without shortcuts has the same depth and width as the optimal residual model (ResNet) but with slightly fewer parameters due to the single dense layer utilized in the dense block shortcut (see Figure 4.2(b)) being removed. The total amount of training epochs required to train the ResNet using early stopping with 20 epoch patience extended that of the ANN without shortcuts with 57 epochs. The optimal residual model is trained for approximately 32 minutes through 127 epochs, a significant increase in training time compared to the linear regression and decision tree. As the proposed ResNet architecture is a much more complex model compared to the decision tree, it is bound to require more computational resources to train. The need for computational resources is not only tied to the training process but also to the inference phase, as each data point prediction requires a complete forward pass through the entire neural network.

It can be observed from Table 5.21 that the neural network predictions required computational time two orders of magnitudes larger than the linear regression and decision tree predictions. The test set used at the prediction phase consisted of 443,751 data points, suggesting that an individual data point requires on average $6.091 \cdot 10^{-5}$ seconds to be inferred by the ResNet. The median and mean packet inter-arrival time of flows in DS-2 were measured to be 0.003 and 0.004 seconds. The measured packet inter-arrival times indicate that large neural networks can infer most flows within the time interval between the first and second packet in a flow. Model predictions made before the arrival of the second packet can thus save approximately $0.003 \cdot n$ seconds on average, compared to methods requiring warm-up periods (where n corresponds to the number of packets needed to be observed before a prediction can be made).

Table 5.21 – Temporal properties and parameter count for the different regression techniques used on DS-2.

Regression Technique	Parameters	Stopping Epoch	Training Time	Prediction Time (s)
Linear Regression	100	NA	00:00:09	0.246
Decision Tree	351	NA	00:00:16	0.249
MLP	13 315	205	00:03:50	8.728
ANN w/o Shortcuts	4 558 851	70	00:16:27	25.001
ResNet	4 880 899	127	00:32:10	27.029

5.2.3 ResNet Monte Carlo Dropout

The Monte Carlo Dropout method was applied to the best ResNet architecture with 0.2 dropout following every ReLU activation, as explained in Section 4.6.4. A total of 100 forward passes were conducted for each data point in the training and testing set. Table 5.22 displays the MAE, MSE, and R^2 metric values computed from the normalized mean target predictions. Table 5.23 displays the SMAPE, MdAE, and MAE metrics computed from the inversely transformed mean target predictions (using Equation 4.9). Similarly to Section 5.2.2 the Monte Carlo Dropouot ResNet demonstrate the ability to predict packet count and flow size variables with comparable relative errors seen in Table 5.22 and the SMAPE metric in Table 5.23. The large disparity in performance between the flow duration and the other variables seemingly persists when applying the Monte Carlo Dropout method to the proposed ResNet architecture, mostly visible in the large SMAPE value difference in Table 5.23. Overall, a slight decrease in all performance metric values can be observed when compared to the values obtained in Section 5.2.2 from the same architecture without any dropout, indicating underfitting caused by the relatively large 0.2 dropout value.

Table 5.22 – Mean MC Dropout ResNet results using transformed targets for DS-2.

	Train			**Test**		
Target Variable	MAE	MSE	R^2	MAE	MSE	R^2
Packet Count	0.078	0.019	0.981	0.081	0.023	0.977
Flow Size	0.085	0.024	0.976	0.088	0.027	0.973
Flow Duration	0.222	0.232	0.769	0.254	0.325	0.672

Table 5.23 – Mean MC Dropout ResNet results using inversely transformed targets for DS-2.

	Train			**Test**		
Target Variable	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Packet Count	0.056	2.034	16.469	0.057	2.069	16.783
Flow Size *(bytes)*	0.086	516.51	12125.085	0.088	527.378	12190.75
Flow Duration *(s)*	0.492	0.040	0.437	0.501	0.040	0.489

Table 5.24 displays the properties of the 95% prediction intervals produced by the stochastic forward passes during inference, in terms of Mean Prediction Interval (MPI) and Prediction Interval Coverage Probability (PICP) metrics defined in Section 4.3. For the target variables packet count and flow size, the MPI is relatively narrow and indicates good optimality. However, the fact that the PICP values do not reach 95% undermines the validity of the intervals [51]. The duration target variable intervals display neither good optimality nor validity, as the coverage is well below 95% and the mean intervals are extensively large. The large mean intervals for flow durations are most likely due to the model's inability to predict small values accurately, especially those smaller than the value of one. Predicting the duration of a flow as two seconds when the actual value is one second causes large deviations that skew the intervals.

Table 5.24 – MC Dropout ResNet model uncertainty results for DS-2.

	Train		**Test**	
Target Variable	MPI	PICP	MPI	PICP
Packet Count	$\pm$21.754%	88.952%	$\pm$22.080%	89.012%
Flow Size *(bytes)*	$\pm$31.050%	88.998%	$\pm$31.469%	89.081%
Flow Duration *(s)*	$\pm$100.257%	61.583%	$\pm$100.564%	61.291%

5.2.4 IP Address Exclusion

Tables 5.25 and 5.26 exhibit the performance metrics of the test set predictions made by the ResNet architecture after training with and without the source and destination IP addresses as inputs. Table 5.25 describes the metrics computed from the normalized targets, while Table 5.26 displays the metrics computed from the inversely transformed targets (using Equation 4.9). The ResNet architecture exhibits an evident decrease in performance when IP addresses are excluded as inputs, especially when tasked to infer flow durations. The SMAPE metric increases by approximately 1% and 3.5% for the packet count and the flow size variables whenever the IP addresses are excluded (see Table 5.26). The flow duration variable exhibits a larger increase in the SMAPE metric whenever IPs are excluded. The decrease in performance indicates that the IP addresses contain valuable information that helps the proposed method to infer the properties of a flow more competently. In particular, it is evident that the flow duration predictions are influenced by IP addresses more than the rest of the variables, suggesting that the duration of a flow is largely influenced by user and application-specific characteristics.

Table 5.25 – ResNet results using transformed test set targets for DS-2 with and without IP addresses as inputs.

	with IPs			**without IPs**		
Target Variable	MAE	MSE	R^2	MAE	MSE	R^2
Packet Count	0.055	0.017	0.983	0.067	0.019	0.981
Flow Size	0.054	0.022	0.978	0.086	0.024	0.976
Flow Duration	0.262	0.399	0.597	0.267	0.455	0.471

Table 5.26 – ResNet results using inversely transformed test set targets for DS-2 with and without IP addresses as inputs.

	with IPs			**without IPs**		
Target Variable	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Packet Count	0.040	1.143	10.254	0.048	1.958	14.572
Flow Size *(bytes)*	0.052	357.096	7523.535	0.087	662.845	12604.348
Flow Duration *(s)*	0.460	0.026	0.492	0.550	0.045	0.572

5.3 DS-3: ISCX-IDS Dataset

The following section aims to present the results obtained from the experimental setup described in Section 4.6 for each of the regression methods presented in Table 4.4 using the ISCX-IDS dataset [54], referred to as DS-3.

5.3.1 Hyperparameter Optimization

The optimal hyperparameters found for each of the regression methods described in Table 4.4 are displayed in Table 5.27. The optimal hyperparameters were obtained through a grid search simulated a total of 5 times for each variation, using the validation set.

Table 5.27 – Optimal hyperparameters found for regression methods in DS-3.

Regression Technique	**Name of Hyperparameters**	**Range**	**Optimal Value**
Linear Regression	NA	NA	NA
Decision Tree Regression	Maximum Depth;	$1, 2, ..., 19, 20$;	11
MLP	Width; Depth;	$32, 64, 128, 256, 512$; 1;	128; 1;
ANN w/o Shortcuts	Width; Depth;	NA NA	128; 2;
ResNet	Width; Depth;	$128, 256, 512$; $1, 2, ..., 9, 10$;	128; 2;

5.3.2 Predictions & Evaluations

Normalized Prediction Results

Tables 5.28, 5.29, and 5.30 exhibit the MAE, MSE, and R^2 obtained by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows for the training and the testing set. The results observed are very similar to the results obtained for DS-1. The ResNet model is able to best predict the three target variables with similar performance metric values for the target variables corresponding to the total packet count and size of flows. The linear regression model displays the worst performance out of the proposed machine learning methods, and is unable to minimize the prediction errors as efficiently as the neural network models. Evidently, the ResNet's use of shortcut connections promote better performance as the errors are comparably smaller compared to the identical model without

shortcut connections. The decision tree manages to rival the deep neural network without any shortcut connections and performs better than the MLP model, suggesting that higher complexity models do not correspond to more accurate predictions. Section 6.2 further discusses the discrepancies found in the results between target variables.

Table 5.28 – Results of predicting *total packet count* using transformed targets for DS-3.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.769	1.000	0.000	0.769	0.999	0.000
Baseline Mode	0.853	1.560	-0.560	0.854	1.562	-0.563
Linear Regression	0.640	0.735	0.265	0.640	0.735	0.265
Decision Tree Regression	0.303	0.315	0.686	0.304	0.316	0.684
MLP	0.276	0.210	0.790	0.278	0.213	0.787
ANN w/o Shortcuts	0.255	0.290	0.710	0.262	0.302	0.697
ResNet	**0.196**	**0.147**	**0.853**	**0.211**	**0.174**	**0.825**

Table 5.29 – Results of predicting *total flow size* in bytes, using transformed targets for DS-3.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.804	1.000	0.000	0.803	0.999	0.000
Baseline Mode	1.348	2.550	-1.550	1.350	2.555	-1.559
Linear Regression	0.577	0.650	0.350	0.577	0.650	0.350
Decision Tree Regression	0.285	0.217	0.783	0.286	0.218	0.782
MLP	0.288	0.192	0.808	0.290	0.195	0.805
ANN w/o Shortcuts	0.276	0.286	0.714	0.282	0.297	0.703
ResNet	**0.214**	**0.140**	**0.860**	**0.229**	**0.164**	**0.836**

Table 5.30 – Results of predicting *total flow duration* in seconds, using transformed targets for DS-3.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.800	1.001	0.000	0.799	0.996	0.000
Baseline Mode	0.932	1.870	-0.868	0.932	1.865	-0.872
Linear Regression	0.675	0.793	0.208	0.675	0.792	0.205
Decision Tree Regression	0.492	0.493	0.507	0.493	0.495	0.503
MLP	0.441	0.412	0.589	0.443	0.416	0.583
ANN w/o Shortcuts	0.403	0.357	0.643	0.412	0.376	0.623
ResNet	**0.349**	**0.269**	**0.732**	**0.371**	**0.314**	**0.685**

Unnormalized Prediction Results

Tables 5.31, 5.32, and 5.33 exhibit the SMAPE, MdAE, and MAE obtained by applying the inverse transformation (Equation 4.9) to the predictions made by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows using the training and the testing set. The ResNet architecture best predicted the three target variables with similar performance metric values for the target variables corresponding to the total packet count and size of flows. A disparity is observable in the architecture's capability of predicting the duration of flows, indicating that the variable is harder to infer and does not necessarily have a strong relationship to the input features. Additionally, the median absolute error is observed to be orders of magnitude smaller than the mean absolute error for all the target variables as the target distributions are heavy-tailed which skews the metrics.

Table 5.31 – Results of predicting *total packet count* using inversely transformed targets for DS-3.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.387	7.768	40.700	0.386	7.768	41.835
Baseline Mode	0.542	7.000	43.412	0.543	7.000	44.572
Linear Regression	0.321	5.924	39.487	0.321	5.937	40.631
Decision Tree Regression	0.170	1.453	30.762	0.171	1.456	31.523
MLP	0.172	1.545	535.030	0.173	1.559	65.620
ANN w/o Shortcuts	0.145	1.001	28.855	0.148	1.020	30.504
ResNet	**0.121**	**0.879**	**23.904**	**0.128**	**0.901**	**26.667**

Table 5.32 – Results of predicting *total flow size* in bytes, using inversely transformed targets for DS-3.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.561	$2.03{\cdot}10^{3}$	$3.39{\cdot}10^{4}$	0.561	$2.03{\cdot}10^{3}$	$3.47{\cdot}10^{4}$
Baseline Mode	0.690	$1.40{\cdot}10^{3}$	$3.43{\cdot}10^{4}$	0.691	$1.41{\cdot}10^{3}$	$3.51{\cdot}10^{4}$
Linear Regression	0.470	$2.30{\cdot}10^{3}$	$3.34{\cdot}10^{4}$	0.470	$2.31{\cdot}10^{3}$	$3.42{\cdot}10^{4}$
Decision Tree Regression	0.260	$6.45{\cdot}10^{2}$	$2.68{\cdot}10^{4}$	0.261	$6.57{\cdot}10^{2}$	$2.74{\cdot}10^{4}$
MLP	0.250	$5.92{\cdot}10^{2}$	$1.80{\cdot}10^{8}$	0.252	$6.01{\cdot}10^{2}$	$1.10{\cdot}10^{7}$
ANN w/o Shortcuts	0.217	$4.60{\cdot}10^{2}$	$2.45{\cdot}10^{4}$	0.221	$4.71{\cdot}10^{2}$	$2.58{\cdot}10^{4}$
ResNet	**0.187**	$\mathbf{3.84{\cdot}10^{2}}$	$\mathbf{2.09{\cdot}10^{4}}$	**0.196**	$\mathbf{4.00{\cdot}10^{2}}$	$\mathbf{2.31{\cdot}10^{4}}$

Table 5.33 – Results of predicting *total flow duration* in seconds, using inversely transformed targets for DS-3.

	Train			Test		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.659	3.395	55.817	0.658	3.395	54.001
Baseline Mode	0.714	1.000	55.930	0.715	1.000	54.116
Linear Regression	0.627	2.822	54.721	0.626	2.823	52.921
Decision Tree Regression	0.572	1.465	35.681	0.572	1.468	34.606
MLP	0.548	1.172	37.049	0.547	1.174	35.685
ANN w/o Shortcuts	0.525	1.077	32.682	0.528	1.088	31.952
ResNet	**0.509**	**0.926**	**32.031**	**0.518**	**0.953**	**31.806**

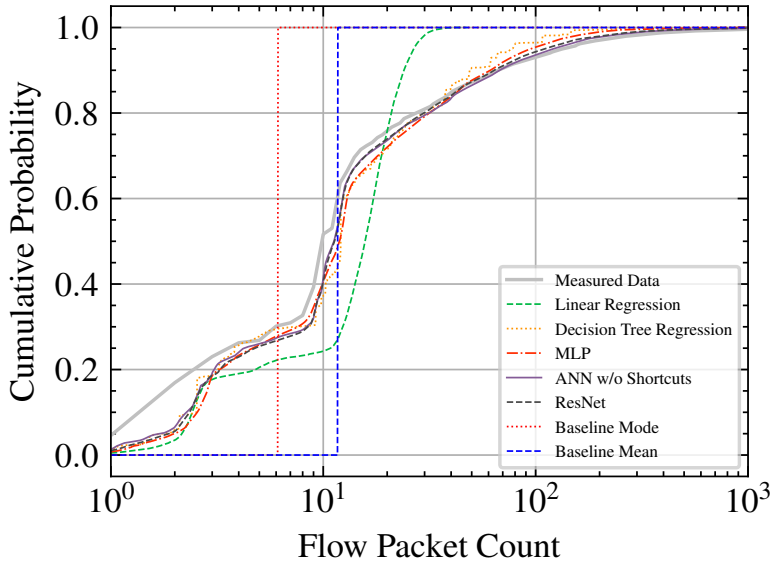


Figure 5.21 – Cumulative distribution of the measured and the predicted packet counts within the flows of the DS-3 test set.

Cumulative Distribution of Predictions

Figures 5.21, 5.22, and 5.23, display the cumulative distribution of the predicted test set target variables for the regression methods listed in Table 4.4. The proposed neural network and decision tree methods demonstrate a good ability to capture the test set distribution for the packet-count and flow size variables, compared to the baselines and the linear regression model. In contrast, a significant disparity can be observed between the cumulative distribution of the predicted flow durations (see Figure 5.23) and the true distribution.

The dissimilarity indicates that the duration of a flow is particularly hard for the proposed methods to predict accurately. Because the flow duration target is a continuous variable, the most frequent value in the test set is zero due to flows containing a single packet. The cumulative distribution for the mode baseline thus stretches infinitely to the left in Figure 5.23 due to the logarithmic x-axis.

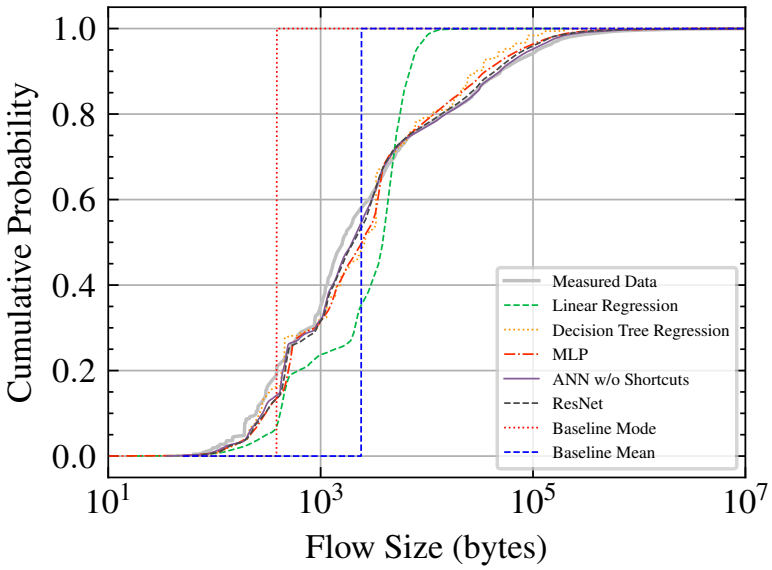


Figure 5.22 – Cumulative distribution of the measured and the predicted flow size within the DS-3 test set.

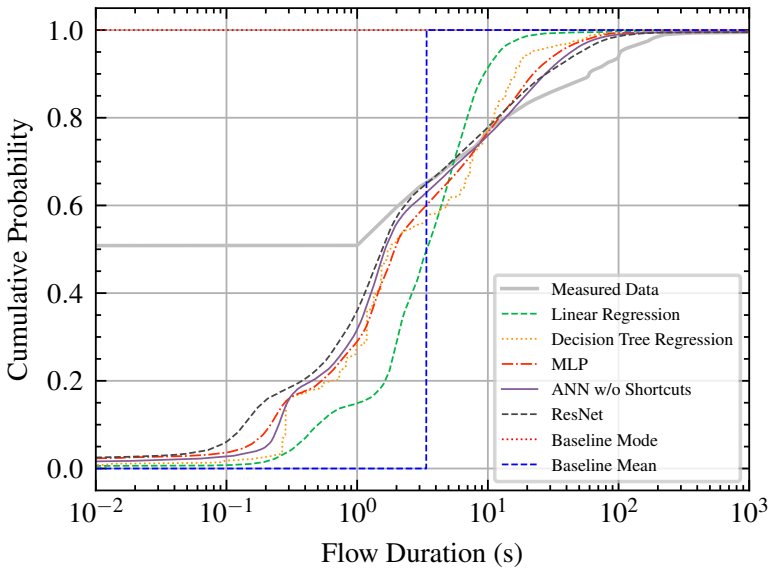


Figure 5.23 – Cumulative distribution of the measured and the predicted flow duration within the DS-3 test set.

Cumulative Distributions of Prediction Ratios

Figures 5.24, 5.25, and 5.26, display the cumulative distribution of the prediction ratios obtained from the test set target variable predictions yielded by the regression methods listed in Table 4.4. The difference in the cumulative probability between the ratio $1-x$ and $1+x$ shows how much of the target set is predicted correctly within a $100 \cdot x\%$ error margin, enabling an intuitive way of understanding the relative error. The error margins are displayed by the dashed vertical lines in the figures. The proposed neural network methods demonstrate the ability to predict approximately $\sim 60\%$, and $\sim 50\%$ of the packet-count and total size variables within a 20% error margin. In contrast, both baselines display capturing only a small fraction of the test set within the same error margin. While the linear regression method manages to outperform both baselines, it fails to rival the performance of the remaining machine learning methods. A significant performance disparity is observed in the test set flow duration prediction ratios (see Figure 5.26), in where the machine learning methods (except for linear regression) manage to predict $\sim 35\%$ of the test set duration variables within a 50% error margin. The dissimilarity indicates that the duration of a flow is particularly hard for the proposed methods to predict accurately. Because the flow duration target is a continuous variable, the most frequent value in the test set is zero due to flows containing a single packet. The cumulative ratio distribution for the flow duration mode baseline thus remains at 0 in the x-axis, making it not appear in Figure 5.26.

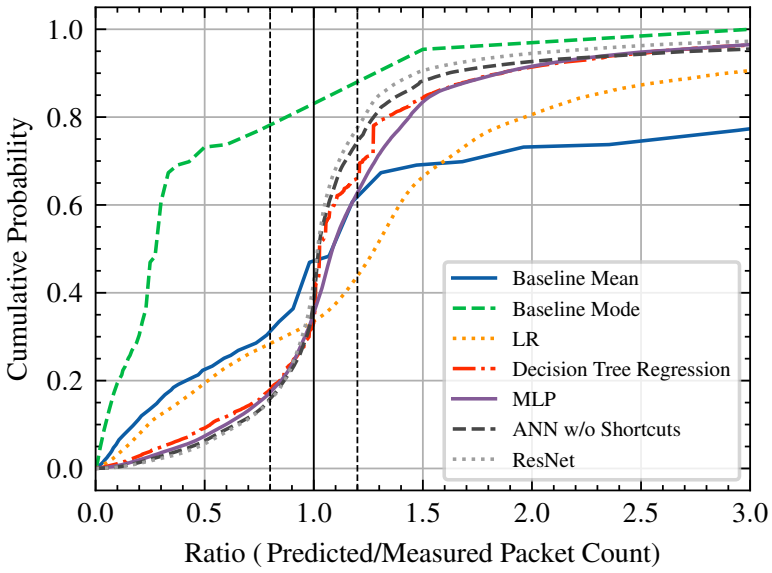


Figure 5.24 – Cumulative distribution of the ratio between the measured and predicted packet count within flows of the DS-3 test set.

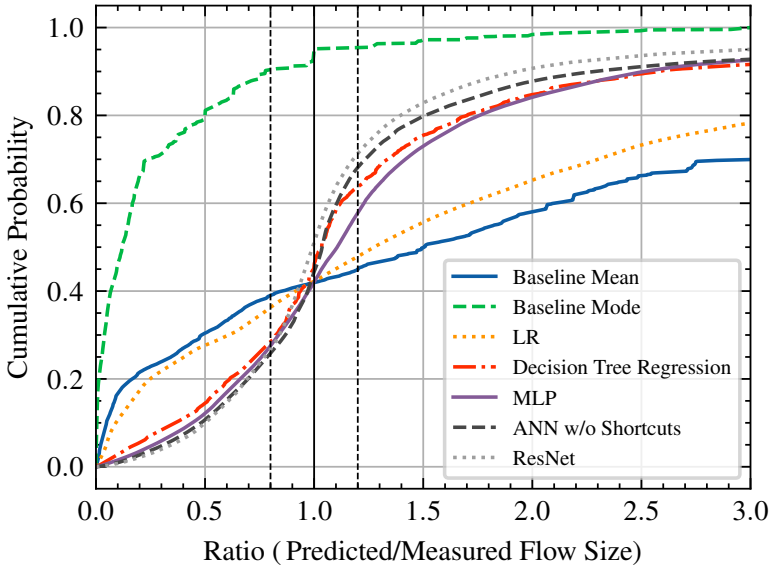


Figure 5.25 – Cumulative distribution of the ratio between the measured and predicted flow size within the DS-3 test set.

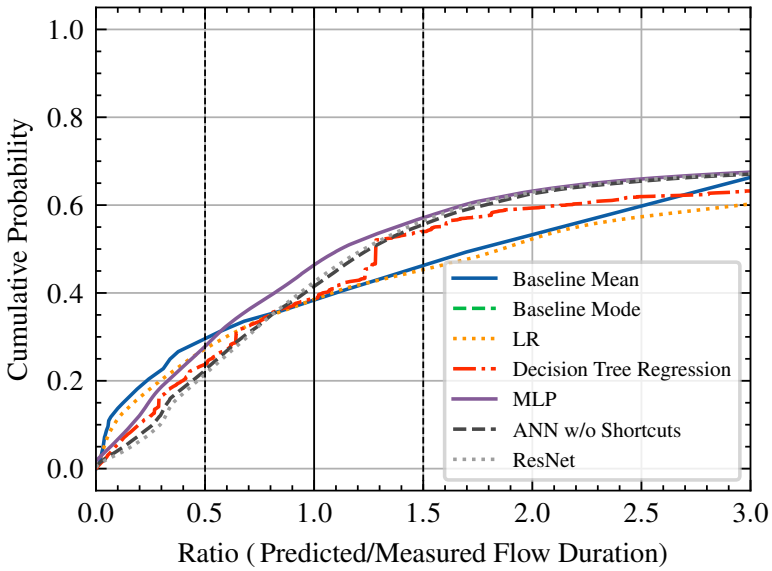


Figure 5.26 – Cumulative distribution of the ratio between the measured and predicted flow duration within the DS-3 test set.

Symmetric Mean Absolute Percentage Errors

Figure 5.27 displays the symmetric mean absolute percentage errors obtained by the different regression methods listed in Table 4.4 on the test set target variables. The proposed machine learning methods exhibit smaller errors compared to the baselines. The ResNet architecture achieves the lowest SMAPE metric value for all three of the target variables, reducing the SMAPE with $\sim 25\%$ for packet counts, $\sim 35\%$ for total flow size, and $\sim 15\%$ for flow durations, compared to the mean baseline. The small SMAPE decrease in the flow duration target variable indicates that the variable is particularly hard for the proposed method to predict accurately, as shown by the results of previous datasets.

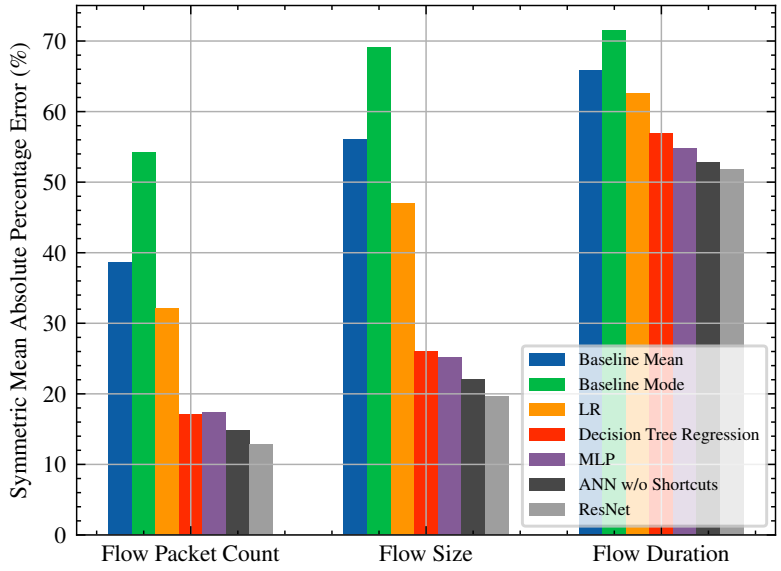


Figure 5.27 – The symmetric mean absolute percentage error within the DS-3 test set for the different regression techniques and target variables.

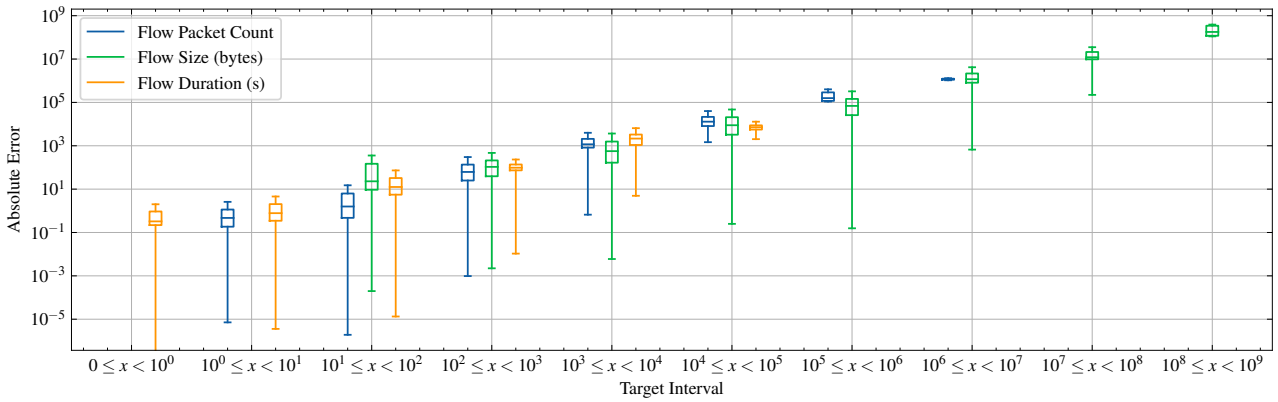


Figure 5.28 – Quartile and median visualization of the absolute error of the ResNet test set predictions for different target intervals in DS-3.

ResNet Absolute Error for Target Intervals

The absolute error produced by the ResNet test predictions for each of the target variables, binned into powers of ten, is displayed in Figure 5.28. The corresponding distribution of the test set is exhibited in Figure 5.29. For each interval, the absolute error is displayed using a box and whisker plot where the box represents the interquartile range. The whiskers represent the minimum and the maximum absolute error, and the line dividing the box represents the median absolute error. The median absolute errors for each target interval have the common behavior of being at most in the same order of magnitude of the target interval, except for the smallest target interval.

The trend of long lower whiskers indicates that the ResNet architecture can predict targets with relatively low absolute errors and rarely wrongfully predicts target variables with errors outside of the actual order of magnitude as indicated by the short top whiskers, marking the maximum error. The target interval distribution of the test set demonstrates that the ResNet architecture can capture complex relationships in flows that are larger than 10^6 bytes and produce similar results as for smaller flows despite the lack of data quantity.

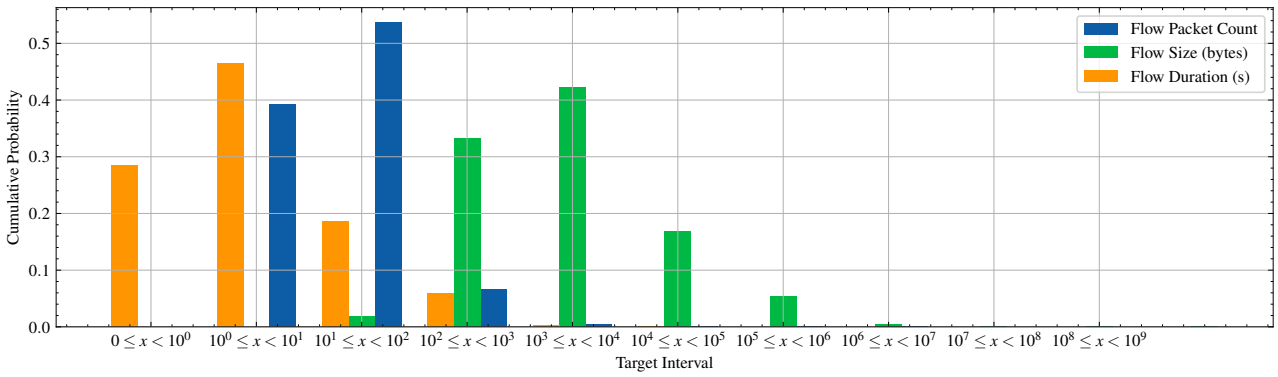


Figure 5.29 – DS-3 test set target distribution grouped into the corresponding order of magnitude.

Classification Accuracies

By grouping the predicted target variable into classes representing orders of magnitude (powers of ten), the predictions made by the methods in Table 4.4 are further simplified to investigate the overall magnitudes of the flow predictions rather than the precise property values. The predicted and measured variables in the test set are organized into their respective class and evaluated using classification accuracy. The corresponding classification accuracy for each target variable is displayed in Figure 5.30. Naturally, the naive mean baseline exhibits similar accuracies as the percental occurrence of the most dominating target interval in the test set, seen in Figure 5.29. The best performing machine learning method is the proposed ResNet model, which is able to outperform the mean baseline by approximately 35% for all target variables.

Evidently, for the DS-3 test set, the ResNet model is capable of predicting the order of magnitude of both flow size and duration with $\sim 85\%$ accuracy and packet count with $\sim 90\%$ accuracy.

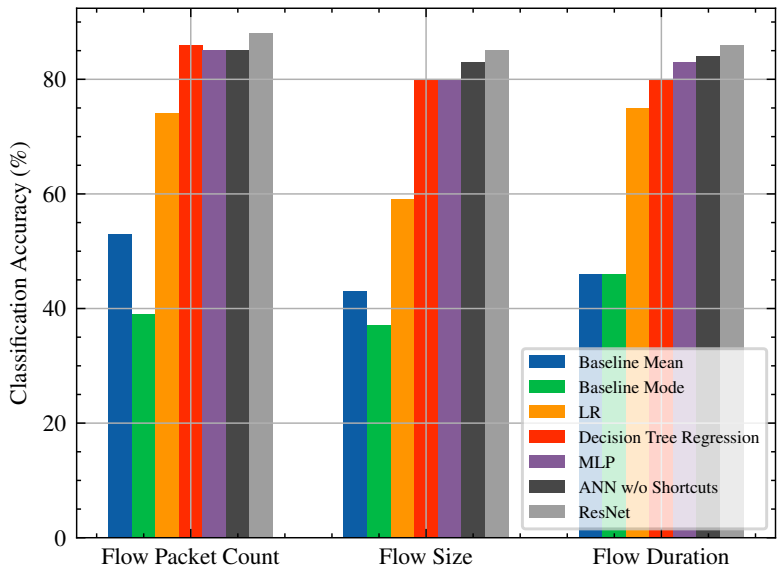


Figure 5.30 – The predicted order of magnitude accuracy within the DS-3 test set for the different regression techniques and target variables.

Temporal Conditions

The training and inference phases were timed and registered to cover the temporal aspect of the main research question. In a practical setting, the regression methods presented in Table 4.4 might need to be retrained as more data is collected. Having insight into the time it takes to train a model or infer a data point is thus beneficial. Table 5.34 displays the number of trainable parameters, stopping epoch, total training time, and the test set prediction time for each of the proposed machine learning models. The results in Table 5.34 were obtained using a machine equipped with an Nvidia Tesla V100 graphics processing unit (GPU), an Intel Xeon 2.00GHz central processing unit (CPU), and 32 GB of random-access memory (RAM). The number of parameters displayed for the decision tree regression techniques refers to its number of leaves. NA in Table 5.34 stands for Not Applicable.

The ANN without shortcuts has the same depth and width as the optimal residual model (ResNet) but with slightly fewer parameters due to the single dense layer utilized in the dense block shortcut (see Figure 4.2(b)) being removed. The total amount of training epochs required to train the ANN without shortcuts using early stopping with 20 epoch patience extended that of the ResNet with 23 epochs. The optimal residual model is trained for approximately 8.5 minutes through 86 epochs, a significant increase in training time compared to the linear regression and decision tree. As the proposed ResNet architecture is a much more complex model compared to the decision tree, it is bound to require more computational resources to train. The need for computational resources is not only tied to the training process but also to the inference phase, as each data point prediction requires a complete forward pass through the entire neural network.

It can be observed from Table 5.34 that the neural network predictions required computational time two orders of magnitudes larger than the linear regression and decision tree predictions. The test set used at the prediction phase consisted of 719,126 data points, suggesting that an individual data point requires on average $5.419 \cdot 10^{-5}$ seconds to be inferred by the ResNet. The median and mean packet inter-arrival time of flows in DS-3 were measured to be 0.025 and 2.492 seconds. The measured packet inter-arrival times indicate that large neural networks can infer most flows within the time interval between the first and second packet in a flow. Model predictions made before the arrival of the second packet can thus save approximately $2.492 \cdot n$ seconds on average, compared to methods requiring warm-up periods (where n corresponds to the number of packets needed to be observed before a prediction can be made).

Table 5.34 – Temporal properties and parameter count for the different regression techniques used on DS-3.

Regression Technique	Parameters	Stopping Epoch	Training Time	Prediction Time (s)
Linear Regression	100	NA	00:00:15	0.356
Decision Tree	1 698	NA	00:00:40	0.403
MLP	13 315	327	00:08:29	13.943
ANN w/o Shortcuts	303 235	109	00:10:52	38.998
ResNet	333 699	86	00:08:38	38.989

5.3.3 ResNet Monte Carlo Dropout

The Monte Carlo Dropout method was applied to the best ResNet architecture with 0.2 dropout following every ReLU activation, as explained in Section 4.6.4. A total of 100 forward passes were conducted for each data point in the training and testing set. Table 5.35 displays the MAE, MSE, and R^2 metric values computed from the normalized mean target predictions. Table 5.36 displays the SMAPE, MdAE, and MAE metrics computed from the inversely transformed mean target predictions (using Equation 4.9). Similarly to Section 5.3.2 the Monte Carlo Dropouot ResNet demonstrate the ability to predict packet count and flow size variables with comparable relative errors seen in Table 5.35 and the SMAPE metric in Table 5.36. The large disparity in performance between the flow duration and the other variables seemingly persists when applying the Monte Carlo Dropout method to the proposed ResNet architecture, mostly visible in the large SMAPE value difference in Table 5.36. Overall, a slight decrease in all performance metric values can be observed when compared to the values obtained in Section 5.3.2 from the same architecture without any dropout, indicating underfitting caused by the relatively large 0.2 dropout value.

Table 5.35 – Mean MC Dropout ResNet results using transformed targets for DS-3.

	Train			**Test**		
Target Variable	MAE	MSE	R^2	MAE	MSE	R^2
Packet Count	0.147	0.074	0.926	0.168	0.106	0.894
Flow Size	0.165	0.070	0.930	0.185	0.098	0.902
Flow Duration	0.305	0.203	0.797	0.335	0.257	0.743

Table 5.36 – Mean MC Dropout ResNet results using inversely transformed targets for DS-3.

	Train			**Test**		
Target Variable	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Packet Count	0.097	0.647	18.738	0.107	0.671	22.39
Flow Size *(bytes)*	0.154	289.086	16020.95	0.167	306.321	19138.168
Flow Duration *(s)*	0.487	0.839	30.046	0.500	0.875	29.947

Table 5.37 displays the properties of the 95% prediction intervals produced by the stochastic forward passes during inference, in terms of Mean Prediction Interval (MPI) and Prediction Interval Coverage Probability (PICP) metrics defined in Section 4.3. For the target variables packet count and flow size, the MPI is relatively narrow and indicates good optimality. However, the fact that the PICP values do not reach 95% undermines the validity of the intervals [51]. The duration target variable intervals display neither good optimality nor validity, as the coverage is well below 95% and the mean intervals are extensively large. The large mean intervals for flow durations are most likely due to the model's inability to predict small values accurately, especially those smaller than the value of one. Predicting the duration of a flow as two seconds when the actual value is one second causes large deviations that skew the intervals.

Table 5.37 – MC Dropout ResNet model uncertainty results for DS-3.

	Train		**Test**	
Target Variable	MPI	PICP	MPI	PICP
Packet Count	$\pm$27.354%	72.487%	$\pm$28.392%	71.155%
Flow Size *(bytes)*	$\pm$45.551%	64.699%	$\pm$47.032%	63.361%
Flow Duration *(s)*	$\pm$75.616%	45.885%	$\pm$77.266%	44.768%

5.3.4 IP Address Exclusion

Tables 5.38 and 5.39 exhibit the performance metrics of the test set predictions made by the ResNet architecture after training with and without the source and destination IP addresses as inputs. Table 5.38 describes the metrics computed from the normalized targets, while Table 5.39 displays the metrics computed from the inversely transformed targets (using Equation 4.9). The ResNet architecture exhibits an evident decrease in performance when IP addresses are excluded as inputs, especially when tasked to infer flow durations. The SMAPE metric increases by approximately 9% and 14% for the packet count and the flow size variables whenever the IP addresses are excluded (see Table 5.39). The flow duration variable exhibits a similar increase in the SMAPE metric whenever IPs are excluded. The decrease in performance indicates that the IP addresses contain valuable information that helps the proposed method to infer the properties of a flow more competently.

Table 5.38 – ResNet results using transformed test set targets for DS-3 with and without IP addresses as inputs.

	with IPs			**without IPs**		
Target Variable	MAE	MSE	R^2	MAE	MSE	R^2
Packet Count	0.211	0.174	0.825	0.376	0.357	0.643
Flow Size	0.229	0.164	0.836	0.412	0.356	0.644
Flow Duration	0.371	0.314	0.685	0.605	0.653	0.346

Table 5.39 – ResNet results using inversely transformed test set targets for DS-3 with and without IP addresses as inputs.

	with IPs			**without IPs**		
Target Variable	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Packet Count	0.128	0.901	26.667	0.222	3.554	36.324
Flow Size *(bytes)*	0.196	400.246	23142.941	0.335	1443.945	31252.305
Flow Duration *(s)*	0.518	0.953	31.806	0.612	2.314	35.881

5.4 DS-4: Ericsson Dataset

The following aims to present the results obtained from the experimental setup described in Section 4.6 for each of the regression methods presented in Table 4.4 using the Ericsson dataset (referred to as DS-4), where the source IP addresses are removed and the two first octets of each unique destination IP address are scrambled.

5.4.1 Hyperparameter Optimization

The optimal hyperparameters found for each of the regression methods described in Table 4.4 are displayed in Table 5.40. The optimal hyperparameters were obtained through a grid search simulated a total of 5 times for each variation, using the validation set.

Table 5.40 – Optimal hyperparameters found for regression methods in DS-4.

Regression Technique	**Name of Hyperparameters**	**Range**	**Optimal Value**
Linear Regression	NA	NA	NA
Decision Tree Regression	Maximum Depth;	$1, 2, ..., 19, 20$;	13
MLP	Width; Depth;	$32, 64, 128, 256, 512$; 1;	256; 1;
ANN w/o Shortcuts	Width; Depth;	NA NA	512; 4;
ResNet	Width; Depth;	$128, 256, 512$; $1, 2, ..., 9, 10$;	512; 4;

5.4.2 Predictions & Evaluations

Normalized Prediction Results

Tables 5.41, 5.42, and 5.43 exhibit the MAE, MSE, and R^2 obtained by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows for the training and the testing set. The ResNet architecture best predicted the three target variables with similar performance metric values for the target variables corresponding to the total packet count and size of flows. The linear regression model displays the worst performance out of the proposed machine learning methods, and is unable to minimize the prediction errors as efficiently as the neural network

models. Evidently, the ResNet's use of shortcut connections promote better performance as the errors are comparably smaller compared to the identical model without shortcut connections. The decision tree manages to rival the deep neural network without any shortcut connections and performs better than the MLP model, suggesting that higher complexity models do not correspond to more accurate predictions.

Table 5.41 – Results of predicting *total packet count* using transformed targets for DS-4.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.787	1.001	0.000	0.786	0.997	0.000
Baseline Mode	1.207	2.457	-1.455	1.208	2.456	-1.463
Linear Regression	0.726	0.890	0.110	0.725	0.886	0.111
Decision Tree Regression	0.479	0.490	0.510	0.481	0.494	0.505
MLP	0.492	0.476	0.524	0.496	0.485	0.514
ANN w/o Shortcuts	0.441	0.399	0.601	0.460	0.443	0.555
ResNet	**0.418**	**0.369**	**0.632**	**0.444**	**0.421**	**0.577**

Table 5.42 – Results of predicting *total flow size* in bytes, using transformed targets for DS-4.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.828	1.001	0.000	0.827	0.998	0.000
Baseline Mode	1.363	2.848	-1.847	1.364	2.849	-1.855
Linear Regression	0.747	0.860	0.140	0.747	0.858	0.140
Decision Tree Regression	0.431	0.399	0.601	0.432	0.401	0.598
MLP	0.448	0.395	0.605	0.452	0.402	0.597
ANN w/o Shortcuts	0.394	0.326	0.674	0.412	0.361	0.638
ResNet	**0.373**	**0.296**	**0.704**	**0.395**	**0.337**	**0.662**

Table 5.43 – Results of predicting *total flow duration* in seconds, using transformed targets for DS-4.

	Train			**Test**		
Regression Technique	MAE	MSE	R^2	MAE	MSE	R^2
Baseline Mean	0.823	1.000	0.000	0.823	1.000	0.000
Baseline Mode	0.803	1.645	-0.645	0.805	1.649	-0.648
Linear Regression	0.793	0.963	0.037	0.793	0.963	0.037
Decision Tree Regression	0.697	0.843	0.157	0.701	0.854	0.147
MLP	0.707	0.851	0.149	0.710	0.860	0.140
ANN w/o Shortcuts	0.661	0.775	0.225	0.680	0.821	0.179
ResNet	**0.634**	**0.746**	**0.254**	**0.666**	**0.821**	**0.180**

Unnormalized Prediction Results

Tables 5.44, 5.45, and 5.46 exhibit the SMAPE, MdAE, and MAE obtained by applying the inverse transformation (Equation 4.9) to the predictions made by the proposed regression methods post-training when given the objective of inferring the total packet count, size, and duration of flows using the training and the testing set. The ResNet architecture best predicted the three target variables with similar SMAPE metric values for the target variables corresponding to the total packet count and size of flows. A more considerable SMAPE value is observed to be caused by all machine learning methods' predictions of the flow duration target variable. The large flow duration SMAPE values indicate that the variable is harder to infer and does not necessarily strongly correlate to the input features. Additionally, as observed in the previous dataset experiments, the median absolute error is observed to be orders of magnitude smaller than the mean absolute error for all the target variables as the target distributions are heavy-tailed which skews the metrics. Section 6.2 further discusses the discrepancies found in the results between target variables.

Table 5.44 – Results of predicting *total packet count* using inversely transformed targets for DS-4.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.485	7.273	85.169	0.484	7.273	84.152
Baseline Mode	0.645	6.000	86.963	0.646	6.000	85.964
Linear Regression	0.451	7.253	84.430	0.451	7.279	83.415
Decision Tree Regression	0.318	3.366	78.654	0.319	3.401	78.497
MLP	0.332	3.973	79.071	0.334	3.985	78.275
ANN w/o Shortcuts	0.302	3.199	72.922	0.311	3.322	**76.895**
ResNet	**0.290**	**2.966**	**70.153**	**0.302**	**3.108**	77.690

Table 5.45 – Results of predicting *total flow size* in bytes, using inversely transformed targets for DS-4.

	Train			**Test**		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.627	955.579	67599.675	0.626	955.579	65874.156
Baseline Mode	0.735	572.000	67738.798	0.736	578.500	66015.423
Linear Regression	0.577	1016.118	67454.446	0.577	1021.202	65720.881
Decision Tree Regression	0.366	403.236	64808.540	0.368	408.737	63369.246
MLP	0.386	483.493	64774.666	0.388	487.789	63080.311
ANN w/o Shortcuts	0.348	376.800	60761.822	0.358	394.200	**62114.747**
ResNet	**0.333**	**334.819**	**58652.558**	**0.346**	**355.700**	62626.499

Table 5.46 – Results of predicting *total flow duration* in seconds, using inversely transformed targets for DS-4.

	Train			Test		
Regression Technique	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Baseline Mean	0.698	1.185	3.815	0.697	1.185	3.798
Baseline Mode	0.880	0.314	3.727	0.881	0.316	3.712
Linear Regression	0.686	1.099	3.753	0.685	1.099	3.736
Decision Tree Regression	0.646	0.939	3.501	0.647	0.940	3.539
MLP	0.655	0.975	3.556	0.655	0.976	3.548
ANN w/o Shortcuts	0.632	0.950	3.376	0.638	0.953	**3.463**
ResNet	**0.614**	**0.870**	**3.287**	**0.625**	**0.897**	3.482

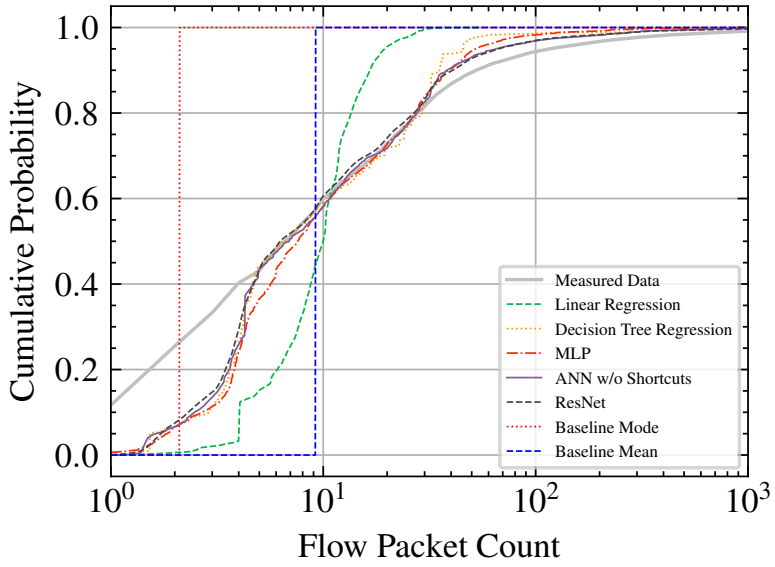


Figure 5.31 – Cumulative distribution of the measured and the predicted packet counts within the flows of the DS-4 test set.

Cumulative Distribution of Predictions

Figures 5.31, 5.32, and 5.33, display the cumulative distribution of the predicted test set target variables for the regression methods listed in Table 4.4. The proposed neural network and decision tree methods demonstrate a good ability to capture the test set distribution of the packet-count and flow size variables compared to the baselines and the linear regression model, especially for flows larger than 400 bytes and 5 packets. In contrast, a significant disparity can be observed between the cumulative distribution of the predicted flow durations (see Figure 5.33) and the true distribution.

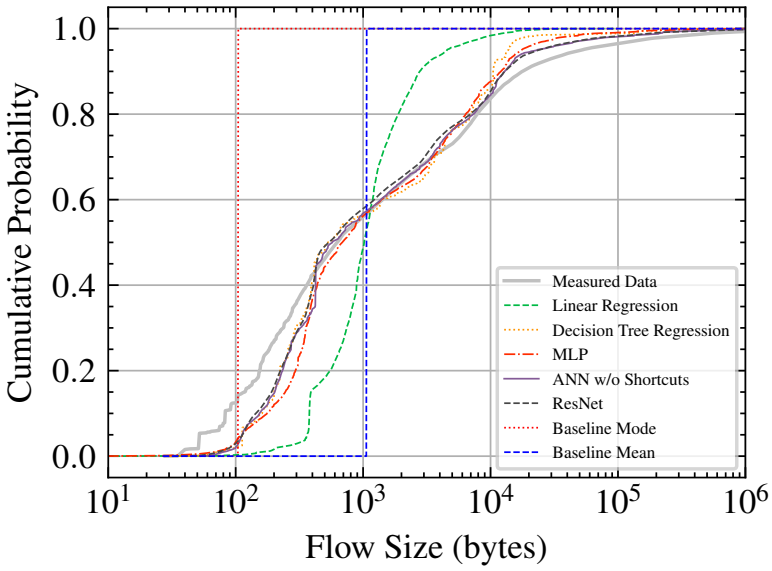


Figure 5.32 – Cumulative distribution of the measured and the predicted flow size within the DS-4 test set.

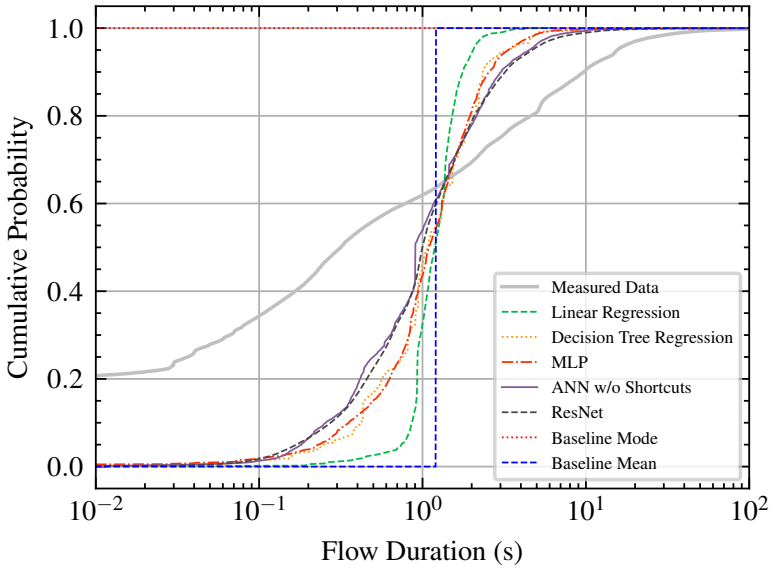


Figure 5.33 – Cumulative distribution of the measured and the predicted flow duration within the DS-4 test set.

Cumulative Distributions of Prediction Ratios

Figures 5.34, 5.35, and 5.36, display the cumulative distribution of the prediction ratios obtained from the test set target variable and predictions yielded by the regression methods listed in Table 4.4. The proposed decision tree and neural network methods demonstrate the ability to predict approximately $\sim$ 25%, and $\sim$ 15% of the packet-count and total size variables within a 20% error margin. In contrast, both baselines display capturing only a small fraction of the test set within the same error margin. The linear regression method displays similar behavior to the mean baseline and fails to rival the performance of the remaining machine learning methods. A significant performance disparity is observed in the test set flow duration prediction ratios (see Figure 5.36), in where the machine learning methods manage to predict $\sim$ 10% of the test set duration variables within a 50% error margin. The dissimilarity indicates that the duration of a flow is particularly hard for the proposed methods to predict accurately. Because the flow duration target is a continuous variable, the most frequent value in the test set is zero due to flows containing a single packet. The cumulative ratio distribution for the flow duration mode baseline thus remains at 0 in the x-axis, making it not appear in Figure 5.36.

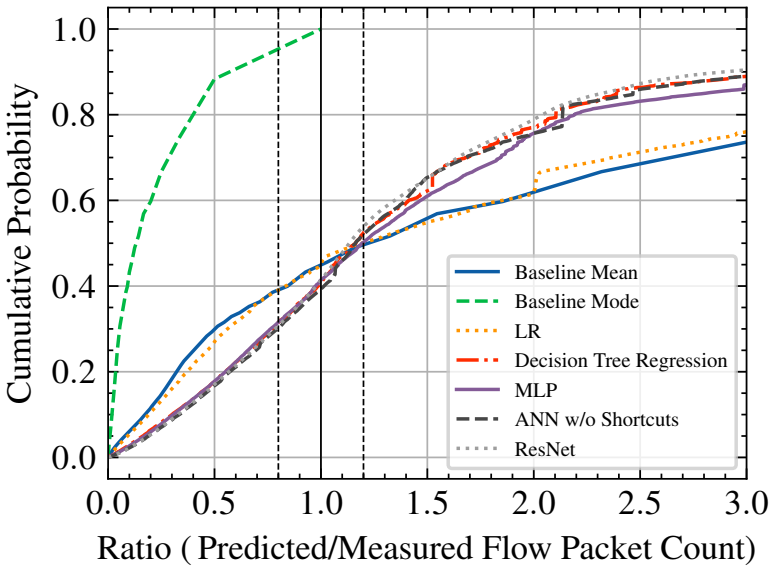


Figure 5.34 – Cumulative distribution of the ratio between the measured and predicted packet count within flows of the DS-4 test set.

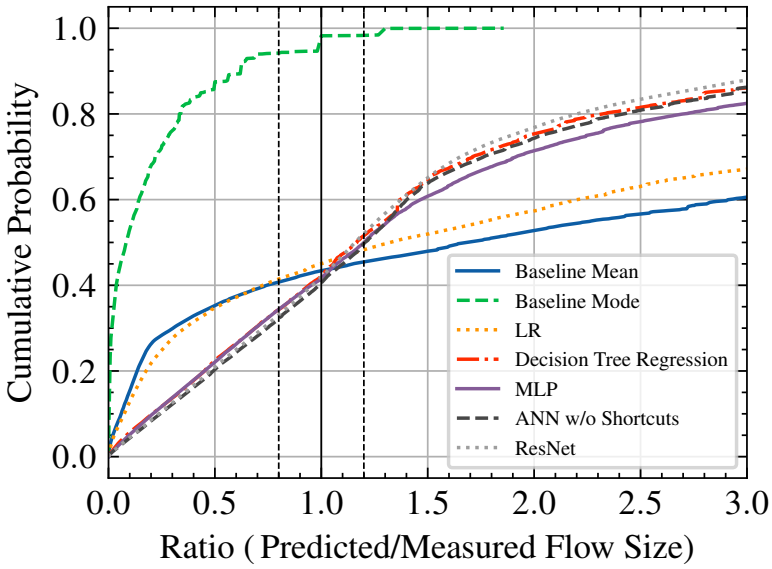


Figure 5.35 – Cumulative distribution of the ratio between the measured and predicted flow size within the DS-4 test set.

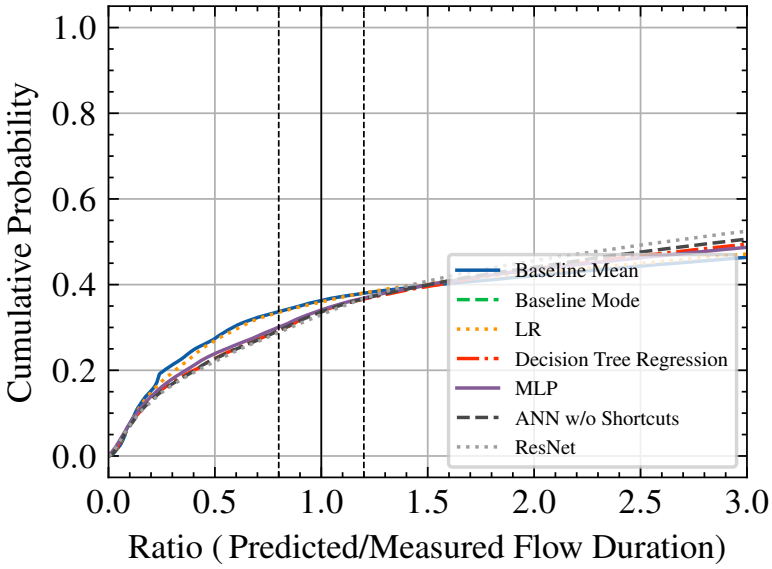


Figure 5.36 – Cumulative distribution of the ratio between the measured and predicted flow duration within the DS-4 test set.

Symmetric Mean Absolute Percentage Errors

Figure 5.37 displays the symmetric mean absolute percentage errors obtained by the different regression methods listed in Table 4.4 on the test set target variables. The proposed machine learning methods exhibit smaller errors compared to the baselines. The ResNet architecture achieves the lowest SMAPE metric value for all three of the target variables, reducing the SMAPE with $\sim 20\%$ for packet counts, $\sim 25\%$ for total flow size, and $\sim 5\%$ for flow durations, compared to the mean baseline. The small SMAPE decrease in the flow duration target variable indicates that the variable is particularly hard for the proposed method to predict accurately, as shown by the results of previous datasets.

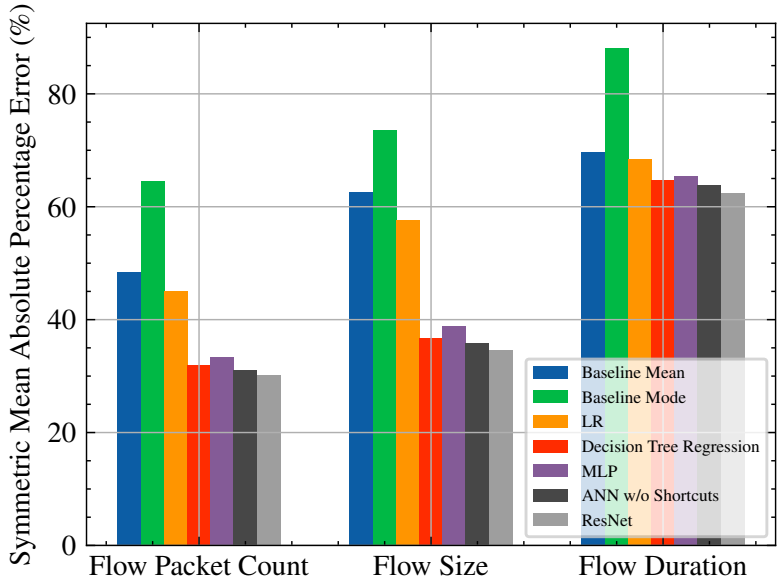


Figure 5.37 – The symmetric mean absolute percentage error within the DS-4 test set for the different regression techniques and target variables.

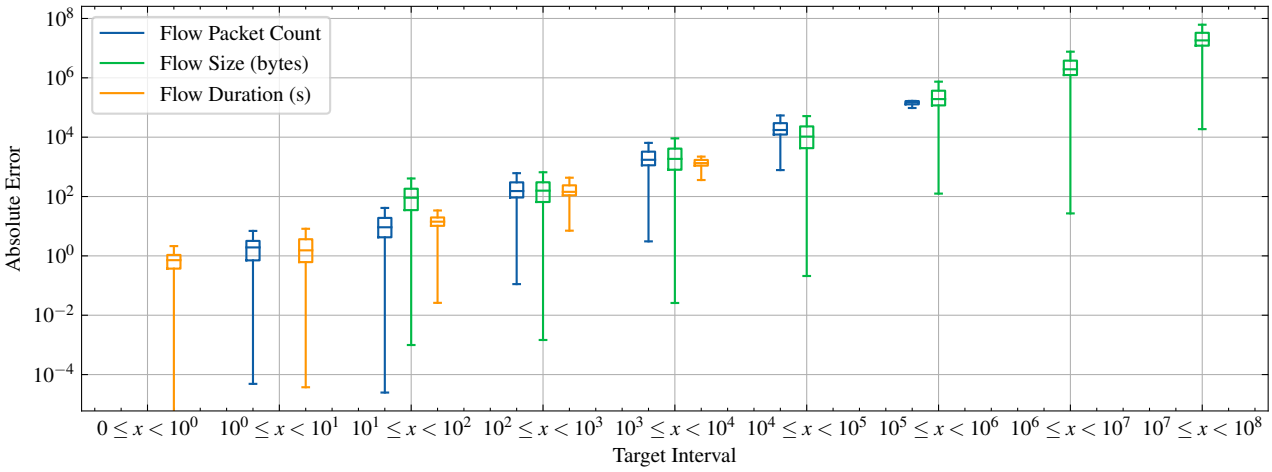


Figure 5.38 – Quartile and median visualization of the absolute error of the ResNet test set predictions for different target intervals in DS-4.

ResNet Absolute Error for Target Intervals

The absolute error produced by the ResNet test predictions for each of the target variables, binned into powers of ten, is displayed in Figure 5.38. The corresponding distribution of the test set is exhibited in Figure 5.39. For each interval, the absolute error is displayed using a box and whisker plot where the box represents the interquartile range. The whiskers represent the minimum and the maximum absolute error, and the line dividing the box represents the median absolute error. The median absolute errors for each target interval have the common behavior of being at most in the same order of magnitude of the target interval, except for the smallest target interval.

The trend of long lower whiskers indicates that the ResNet architecture can predict targets with relatively low absolute errors and rarely wrongfully predicts target variables with errors outside of the actual order of magnitude as indicated by the short top whiskers, marking the maximum error. The target interval distribution of the test set demonstrates that the ResNet architecture can capture complex relationships in flows that are larger than 10^5 bytes and produce similar results as for smaller flows despite the lack of data quantity.

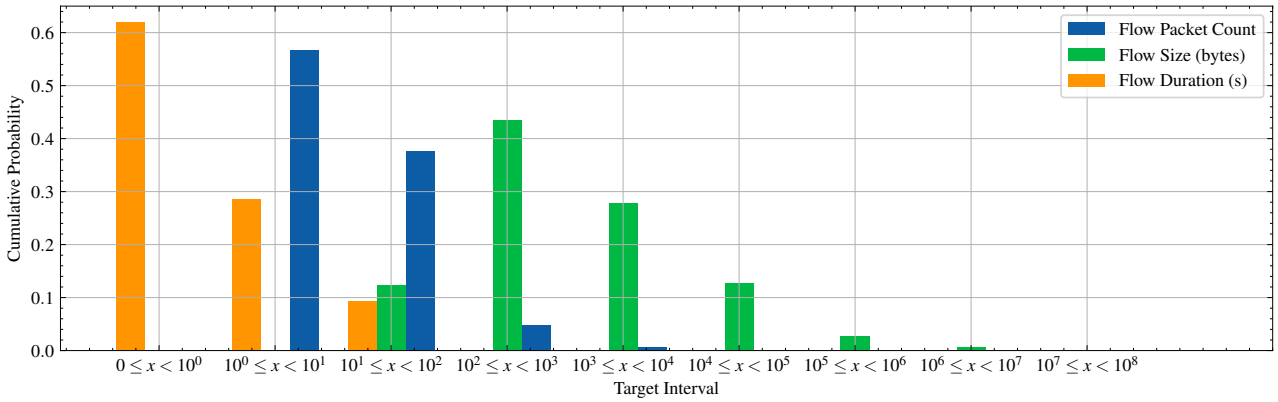


Figure 5.39 – DS-4 test set target distribution grouped into the corresponding order of magnitude.

Classification Accuracies

By grouping the predicted target variable into classes representing orders of magnitude (powers of ten), the predictions made by the methods in Table 4.4 are further simplified to investigate the overall magnitudes of the flow predictions rather than the precise property values. The predicted and measured variables in the test set are organized into their respective class and evaluated using classification accuracy. The corresponding classification accuracy for each target variable is displayed in Figure 5.40. Naturally, the naive mean baseline exhibits similar accuracies as the percental occurrence of the most dominating target interval in the test set, seen in Figure 5.39. The best performing machine learning method is the proposed ResNet model, which is able to outperform the mean baseline with approximately 20% for packet counts, $\sim 35\%$ for flow size, and $\sim 25\%$ for flow duration, compared to the mean baseline.

Evidently, for the DS-4 test set, the ResNet model is capable of predicting the order of magnitude of for packet counts with $\sim 80\%$ accuracy, flow size with $\sim 65\%$ accuracy, and flow duration with $\sim 88\%$ accuracy.

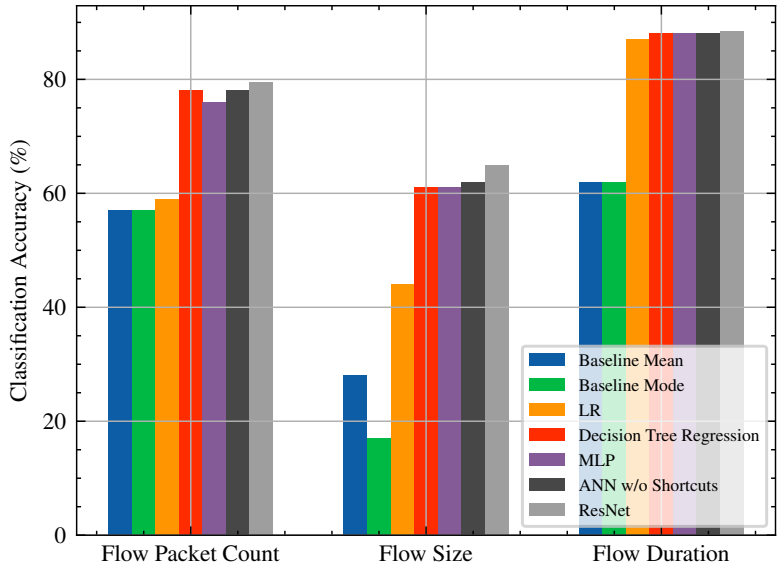


Figure 5.40 – The predicted order of magnitude accuracy within the DS-4 test set for the different regression techniques and target variables.

Temporal Conditions

The training and inference phases were timed and registered to cover the temporal aspect of the main research question. In a practical setting, the regression methods presented in Table 4.4 might need to be retrained as more data is collected. Having insight into the time it takes to train a model or infer a data point is thus beneficial. Table 5.47 displays the number of trainable parameters, stopping epoch, total training time, and the test set prediction time for each of the proposed machine learning models. The results in Table 5.47 were obtained using a machine equipped with an Nvidia Tesla V100 graphics processing unit (GPU), an Intel Xeon 2.00GHz central processing unit (CPU), and 32 GB of random-access memory (RAM). The number of parameters displayed for the decision tree regression techniques refers to its number of leaves. NA in Table 5.47 stands for Not Applicable.

The ANN without shortcuts has the same depth and width as the optimal residual model (ResNet) but with slightly fewer parameters due to the single dense layer utilized in the dense block shortcut (see Figure 4.2(b)) being removed. The total amount of training epochs required to train the ANN without shortcuts using early stopping with 20 epoch patience extended that of the ResNet with 10 epochs. The optimal residual model is trained for approximately 28 minutes through 113 epochs, a significant increase in training time compared to the linear regression and decision tree. As the proposed ResNet architecture is a much more complex model compared to the decision tree, it is bound to require more computational resources to train. The need for computational resources is not only tied to the training process but also to the inference phase, as each data point prediction requires a complete forward pass through the entire neural network.

It can be observed from Table 5.47 that the neural network predictions required computational time two orders of magnitudes larger than the linear regression and decision tree predictions. The test set used at the prediction phase consisted of 226,345 data points, suggesting that an individual data point requires on average $9.689 \cdot 10^{-5}$ seconds to be inferred by the ResNet. The median and mean packet inter-arrival time of flows in DS-4 were measured to be 0.041 and 0.239 seconds. The measured packet inter-arrival times indicate that large neural networks can infer most flows within the time interval between the first and second packet in a flow. Model predictions made before the arrival of the second packet can thus save approximately $0.239 \cdot n$ seconds on average, compared to methods requiring warm-up periods (where n corresponds to the number of packets needed to be observed before a prediction can be made).

Table 5.47 – Temporal properties and parameter count for the different regression techniques used on DS-4.

Regression Technique	Parameters	Stopping Epoch	Training Time	Prediction Time (s)
Linear Regression	68	NA	00:00:03	0.054
Decision Tree	4 877	NA	00:00:08	0.062
MLP	14 851	365	00:04:13	3.904
ANN w/o Shortcuts	9 296 387	123	00:27:36	19.864
ResNet	10 120 707	113	00:28:03	21.930

5.4.3 ResNet Monte Carlo Dropout

The Monte Carlo Dropout method was applied to the best ResNet architecture with 0.2 dropout following every ReLU activation, as explained in Section 4.6.4. A total of 100 forward passes were conducted for each data point in the training and testing set. Table 5.48 displays the MAE, MSE, and R^2 metric values computed from the normalized mean target predictions. Table 5.49 displays the SMAPE, MdAE, and MAE metrics computed from the inversely transformed mean target predictions (using Equation 4.9). Similarly to Section 5.4.2 the Monte Carlo Dropouot ResNet demonstrate the ability to predict packet count and flow size variables with comparable relative errors seen in Table 5.48 and the SMAPE metric in Table 5.49. The large disparity in performance between the flow duration and the other variables seemingly persists when applying the Monte Carlo Dropout method to the proposed ResNet architecture, mostly visible in the large SMAPE value difference in Table 5.49. Overall, a slight increase in all performance metric values can be observed when compared to the values obtained in Section 5.4.2 from the same architecture without any dropout.

Table 5.48 – Mean MC Dropout ResNet results using transformed targets for DS-4.

	Train			**Test**		
Target Variable	MAE	MSE	R^2	MAE	MSE	R^2
Packet Count	0.417	0.369	0.631	0.436	0.409	0.589
Flow Size	0.369	0.295	0.705	0.386	0.326	0.674
Flow Duration	0.637	0.750	0.250	0.659	0.802	0.199

Table 5.49 – Mean MC Dropout ResNet results using inversely transformed targets for DS-4.

	Train			**Test**		
Target Variable	SMAPE	MdAE	MAE	SMAPE	MdAE	MAE
Packet Count	0.289	2.897	70.706	0.298	3.019	75.593
Flow Size *(bytes)*	0.329	318.923	59792.981	0.339	334.767	61186.008
Flow Duration *(s)*	0.618	0.891	3.276	0.625	0.905	3.422

Table 5.50 displays the properties of the 95% prediction intervals produced by the stochastic forward passes during inference, in terms of Mean Prediction Interval (MPI) and Prediction Interval Coverage Probability (PICP) metrics defined in Section 4.3. For the target variables packet count and flow size, the MPI is relatively narrow and indicates good optimality. However, the fact that the PICP values do not reach 95% undermines the validity of the intervals [51].

The duration target variable intervals display neither good optimality nor validity, as the coverage is well below 95% and the mean intervals are extensively large. The large mean intervals for flow durations are most likely due to the model's inability to predict small values accurately, especially those smaller than the value of one. Predicting the duration of a flow as two seconds when the actual value is one second causes large deviations that distort the intervals.

Table 5.50 – MC Dropout ResNet model uncertainty results for DS-4.

	Train		**Test**	
Target Variable	MPI	PICP	MPI	PICP
Packet Count	$\pm$26.239%	29.179%	$\pm$27.282%	28.501%
Flow Size *(bytes)*	$\pm$39.002%	33.310%	$\pm$40.563%	32.680%
Flow Duration *(s)*	$\pm$38.901%	19.960%	$\pm$39.708%	19.354%

Chapter 6

Discussion

This chapter aims to discuss the results obtained from the experimental setup in Chapter 5 for each of the datasets. The main objective is to understand the performance of the proposed Residual Neural Network architecture and compare the results obtained from the different datasets against the traditional machine learning methods and baselines.

6.1 Overall ResNet Performance

The proposed Residual Neural Network (ResNet) for regression proved to outperform the baselines and the traditional machine learning methods in all datasets, except for DS-2, in which a decision tree model rivaled its performance. In general, the linear regression model performed the worst out of the proposed machine learning methods, indicating more complex relationships between the variables that a linear function can not capture. The absolute errors yielded from the ResNet architecture for powers of ten target intervals all behaved similarly. The median absolute errors for each target interval had the collective behavior of being at most in the same order of magnitude of the target interval, except for the most minor target interval. The ResNet architecture predicted targets with relatively low absolute errors and rarely predicted target variables with errors outside of the actual order of magnitude. While these errors are still relatively large, they are small enough to enable magnitude classification (powers of ten) of incoming flows with approximately $80 - 95\%$ accuracy (for DS-1, DS-2, and DS-3).

The median errors have the shared behavior of being considerably smaller than the mean errors for all target variables in every dataset. This is because the target distributions are skewed to the right, with a tail that includes values exceptionally greater than the "typical" values, which heavily skews the error distribution.

From the experiments, it is observed that the proposed neural network benefits significantly by being both deeper and including shortcut connections, as indicated by the results produced by the MLP model as well as the ANN without shortcuts model. The residual shortcut connections offered important value to the training process of the networks by enabling the networks to become deeper, allowing for more complex relationships to be captured without the drawbacks of the vanishing gradient problem, as pointed out by the improved results. The shortcut connections were thus very beneficial as they did not require any additional computational complexity.

6.2 Differences Between Target Variables

The evaluation metric values for the predicted packet count and flow size variables were similar within each dataset; this property is mainly due to the high correlation between the two variables. In contrast, the evaluation metric values from the predicted flow durations were all mutually low for each dataset. Table 4.3 shows that the bytes and packets in the flows are highly independent of the flow duration. The results obtained in the experiments indicate that the information available at the arrival of the first packet within a flow contains adequate data to estimate the packet count and flow size. A study conducted in 2004 [7] showed that the duration of a flow is primarily correlated to the type of application that initiated the flow. Flow durations for P2P file-sharing applications are typically short in relation to their size, while flow durations of web traffic are generally longer and stochastic due to user behaviors and interactions. Instant messaging applications typically have long mean flow durations as the applications have periodic interactions with the central servers to acquire updated friend lists and messages. Thus, in order to more accurately predict the duration of an incoming flow, application-specific information is needed as inputs for machine learning models, as IP addresses, ports, and transport layer protocols are seemingly insufficient.

6.3 Differences Between Datasets

DS-1 and DS-3 exhibited approximately the same test set evaluation metric values despite having different target distributions (see Section 4.4.1), with similar SMAPE and classification accuracies for the different targets. Although DS-1 and DS-2 displayed almost identical target distributions, their results were different, as the proposed ResNet was able to capture almost all of the data variance within DS-2 (as indicated by the large R^2 values) for the packet-count and flow size targets, but not within DS-1. Overall, DS-2 exhibited the best performance given the proposed machine learning methods and could be correctly classified with over 95% accuracy for all three target variables. The reason for this may lie within the process of how the data and its noise were generated by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). The step-wise nature of the cumulative distribution of the test set target variables for DS-2 indicate that the entire dataset is fairly discrete and thus exhibits synthetic characteristics not found in the other datasets.

The processed dataset DS-4, provided by Ericsson, consisted of real-world cellular packet header logs collected during 2017 and 2020. Because of privacy regulations, the dataset was anonymized by shuffling the first two octets of each unique destination IP address and removing the source IP address. An interesting observation is that the target distributions of DS-4 are reasonably similar to the distributions of DS-1, and that the evaluation metric results obtained for DS-4 closely resemble the results obtained for DS-1 when IP addresses were excluded from the input space. The anonymization of DS-4 is thus most likely the reasons as to why it performed the worst out of all datasets in the experiments.

6.4 Exclusion of IP Addresses

Overall, for all datasets, the ResNet architecture exhibits an evident decrease in performance when IP addresses are excluded as inputs. The decrease in performance indicates that the IP addresses used in a flow contain valuable information that helps the proposed method to infer the properties of a flow more competently. In particular, it is evident that the flow duration predictions are influenced by IP addresses more than the rest of the variables, suggesting that the duration of a flow is largely influenced by user and application-specific characteristics most prominently found in the addresses. Despite the decrease in performance, the architecture is still able to perform better than the baselines with relatively low errors and can thus still be helpful to indicate the magnitude of incoming flows, especially if IP addresses are unavailable due to privacy reasons.

6.5 Monte Carlo Dropout

By applying the Monte Carlo dropout method to the proposed ResNet, its evaluation metrics could be improved in all datasets except DS-2, with a slight margin. In practice, the performance gained by using the Monte Carlo dropout method must be carefully justified as the extra computational power and time it takes to simulate a forward pass a multitude of times for a single data point is most likely going to out-weight the benefits for the majority of applications. Additionally, the 95% prediction intervals produced by the Monte Carlo dropout method do not, on average, enclose 95% of the realized observations, weakening the validity of the intervals. The mean prediction interval (MPI) for the packet-count and the flow size were all within the range of approximately $\pm 30\%$, which could be considered an appropriate interval range, had the Prediction Interval Coverage Probability (PICP) covered close to 95% of the sets. The MPI for flow durations were all exceptionally high, indicating that the model is notably uncertain about the specific target variable. The sizeable average prediction intervals for flow durations are most likely due to the fact that the duration distribution is heavily concentrated within the small ranges as seen in 4.4.1. Thus, predicting a flow to be two seconds for instance, when the actual value is one second, causes large deviations that are then used to compute the intervals.

Chapter 7

Conclusions

The following chapter aims to summarize the results obtained from the experimental setup and possible directions for future work.

7.1 Summary

To summarize, the objective of the thesis was to design and evaluate a solution for network traffic flow predictions given the information available at the time of the first packet arrival.

A proposed ResNet architecture was trained and compared to traditional machine learning models using four separate datasets containing 1-3 million data points, each representing a wide variety of WiFi and cellular network flows.

The results of this study indicate that using the proposed ResNet architecture (without the need of a warm-up period) given the first packet in a flow:

- the order of magnitude (powers of ten) of the packet count can, on average, be predicted with 85% accuracy.
- the order of magnitude (powers of ten) of the flow size in bytes can, on average, be predicted with 80% accuracy.
- the order of magnitude (powers of ten) of the flow duration in seconds can, on average, be predicted with 85% accuracy.

The mean packet inter-arrival time for flows within all four datasets indicate that the proposed ResNet architecture can predict most flows in the time interval between the first and second packets in a flow, on average saving $1.638 \cdot n$ seconds compared to any method utilizing warm-up periods, where n corresponds to the number of packets needed to be observed before a prediction can be made.

It is thus shown that features gathered from the first packet in a flow can be used to make meaningful predictions about the magnitude of incoming flows without a warm-up period. Results obtained from experiments where the IP addresses were excluded from the input space indicate that the predictor is still able to outperform the baselines without the information embedded within the IP addresses. Though, including IP addresses significantly improves the models' capability of predicting incoming flows accurately. The predictor is thus able to generalize to flows in which sensitive user information has been anonymized (due to privacy regulations) and perform better than the mean baselines for the target variables, with results that indicate the magnitude of the flow sufficiently. Given that the prediction interval coverage probability of the model uncertainty did not, on average, enclose the realized observations in 95% of all cases, it is concluded that the predictor can not learn the uncertainty of the predicted quantities adequately using only model uncertainty.

7.2 Future work

There are several directions that the work presented in this thesis can be extended to achieve more precise results. The experiments in this thesis only included the first packet size within each flow as an input to the methods. One extension to the present methods is to include the first N packet sizes instead. Another possible direction is to investigate whether the individual flow packets can be sequentially predicted. A contextual LSTM [65] could be used where the ResNet outputs (flow size, number of packets, and flow duration) are introduced to the time series model as auxiliary inputs. The auxiliary inputs could then be used to learn the initial hidden states for the LSTM model to predict quantities about individual packets within flows.

The prediction intervals produced by our method only include the variance caused by the model. Including an approximation of the data variance should, in theory, result in better interval validity but worse optimality, as the intervals will become wider and contain more target values. However, approximating

data variance is difficult for network traffic predictions as the target distributions are wide and skewed, which implies that most approximations will overestimate the noise level and lead to non-optimal prediction intervals. A novel and more precise approximation of the data variance is thus going to be needed.

References

[1] M. Iqbal, M. Zahid, D. Habib, and L. John, "Efficient prediction of network traffic for real-time applications," *Journal of Computer Networks and Communications*, pp. 1–11, 2019. doi: 10.1155/2019/4067135

[2] A. Azari, P. Papapetrou, S. Denic, and G. Peters, *Cellular Traffic Prediction and Classification: A Comparative Evaluation of LSTM and ARIMA*, 2019, pp. 129–144. ISBN 978-3-030-33777-3

[3] T. Aldhyani, M. Alrasheed, A. Alqarni, Y. Alzahrani, and A. Bamhdi, "Intelligent hybrid model to enhance time series models for predicting network traffic," *IEEE Access*, 2020. doi: 10.1109/ACCESS.2020.3009169

[4] D. Harkut and Z. Shaikh, "An overview of network traffic classification methods," *Int. J. Recent Innovation Trends Comput. Commun*, vol. 3, pp. 482–488, 2015.

[5] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using lstm networks," in *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018. doi: 10.1109/PIMRC.2018.8581000 pp. 1827–1832.

[6] M. Aliyari Shoorehdeli, M. Teshnehlab, and A. Sedigh, "Training anfis as an identifier with intelligent hybrid stable learning algorithm based on particle swarm optimization and extended kalman filter," *Fuzzy Sets and Systems*, vol. 160, pp. 922–948, 2009. doi: 10.1016/j.fss.2008.09.011

[7] M. sup Kim, Y. J. Won, H. jo Lee, J. W. Hong, and R. Boutaba, "Flow-based characteristic analysis of internet application traffic," in *in Proc. E2EMON*, 2004, pp. 62–67.

[8] Juan Sebastián Rojas. (2019) Labeled network traffic flows - 141 applications. Universidad Del Cauca. [Online]. Available: https://www.kaggle.com/jsrojas/labeled-network-traffic-flows-114-applications

[9] United Nations (U.N). (2015) Transforming our world: the 2030 agenda for sustainable development. [Online]. Available: https://EconPapers.repec.org/RePEc:ess:wpaper:id:7559

[10] J. Clement. (2019) Mobile internet usage worldwide - Statistics & Facts. Statista Research Department. [Online]. Available: https://www.statista.com/topics/779/mobile-internet/

[11] Internet Engineering Task Force (IETF). (2021) Request for Comments (RFC) Index. [Online]. Available: https://tools.ietf.org/rfc/index

[12] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 7th ed. Pearson, 2016. ISBN 978-0-13-359414-0

[13] Internet Engineering Task Force (IETF). (2020) IANA IPv4 Special-Purpose Address Registry. [Online]. Available: https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

[14] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering, "IPv6 Flow Label Specification," IETF, RFC 3697, 2004. [Online]. Available: http://tools.ietf.org/rfc/rfc3697.txt

[15] M. Kubat, *An Introduction to Machine Learning*, 1st ed. Springer Publishing Company, Incorporated, 2015. ISBN 3319200097

[16] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 1310–1315.

[17] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis (4th ed.)*. Wiley & Sons, 2006. ISBN 0471754951

[18] S. Weisberg, *Applied Linear Regression*, 3rd ed. Hoboken NJ: Wiley, 2005. ISBN 9780471704096. [Online]. Available: http://www.stat.umn.edu/alr

[19] L. Rokach and O. Maimon, *Data Mining With Decision Trees: Theory and Applications*, 2nd ed. World Scientific Publishing Co., Inc., 2014. ISBN 9789814590075

[20] M. Hjorth-Jensen, "Data analysis and machine learning: From decision trees to forests and all that." Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University, 2019.

[21] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, "Artificial neural network architectures and training processes," in *Artificial neural networks*. Springer, 2017, pp. 21–28.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016. ISBN 0262035618

[23] K. Hinkelmann. (2021) Neural networks. University of Applied Sciences Northwestern Switzerland. [Online]. Available: http://didattica.cs.unicam.it/lib/exe/fetch.php?media\=didattica:magistrale:kebi:ay_1718:ke-11_neural_networks.pdf

[24] B. Mehlig, "Machine learning with neural networks," vol. ArXiv abs/1901.05639. Department of Physics, University of Gothenburg, 2019.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision (ICCV)*, 2015. doi: 10.1109/ICCV.2015.123 pp. 1026–1034.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 37. PMLR, 2015, pp. 448–456.

[27] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *ArXiv*, vol. abs/1808.03314, 2018.

[28] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," *ArXiv*, vol. abs/1912.05911, 2019.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, p. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[30] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *IEEE workshop on automatic speech recognition and understanding*, 2013, pp. 273–278.

[31] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," vol. ArXiv abs/1610.10099, 2017. [Online]. Available: https://arxiv.org/abs/1610.10099

[32] W. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, pp. 1565–1567, 2007. doi: 10.1038/nbt1206-1565

[33] X.-h. Tan, W.-h. Bi, X.-l. Hou, and W. Wang, "Reliability analysis using radial basis function networks and support vector machines," *Computers and Geotechnics*, vol. 38, no. 2, pp. 178–186, 2011.

[34] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik *et al.*, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.

[35] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, "Spatio-temporal wireless traffic prediction with recurrent neural network," *IEEE Wireless Communications Letters*, vol. 7, no. 4, pp. 554–557, 2018. doi: 10.1109/LWC.2018.2795605

[36] W. Yuankai, H. Tan, B. Ran, and Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding," *Transportation Research Part C: Emerging Technologies*, vol. 90, 2018. doi: 10.1016/j.trc.2018.03.001

[37] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE Conference on Computer Communications*, 2017. doi: 10.1109/INFOCOM.2017.8057090 pp. 1–9.

[38] M. Ring, A. Dallmann, D. Landes, and A. Hotho, "Ip2vec: Learning similarities between ip addresses," in *IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017. doi: 10.1109/ICDMW.2017.93 pp. 657–666.

[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[40] M. Li, C. Lumezanu, B. Zong, and H. Chen, "Deep learning ip network representations," in *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, ser.

Big-DAMA '18. Association for Computing Machinery, 2018. ISBN 9781450359047 p. 33–39.

[41] R. Beverly, K. Sollins, and A. Berger, "Svm learning of ip address structure for latency prediction," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*. Association for Computing Machinery, 2006. ISBN 159593569X p. 299–304.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. doi: 10.1109/CVPR.2016.90 pp. 770–778.

[43] D. Chen, F. Hu, G. Nian, and T. Yang, "Deep residual learning for nonlinear regression," *Entropy*, vol. 22, p. 193, 2020. doi: 10.3390/e22020193

[44] X. Qiu, L. Zhang, Y. Ren, P. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," 2014. doi: 10.1109/CIEL.2014.7015739

[45] K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997. [Online]. Available: https://EconPapers.repec.org/RePEc:eee:stapro:v:33:y:1997:i:3:p:291-297

[46] B. Das, D. Science, T. O. (Australia), Electronics, and S. R. L. (Australia)., *Representing uncertainties using Bayesian networks / Balaram Das*. DSTO Electronics and Surveillance Research Laboratory Salisbury, S. Aust, 1999. [Online]. Available: http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-0918.pdf

[47] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *ArXiv*, vol. abs/1506.02142, 2016.

[48] D. Nix and A. Weigend, "Estimating the mean and variance of the target probability distribution," *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 1, pp. 55–60 vol.1, 1994.

[49] T. Heskes, "Practical confidence and prediction intervals," in *Advances in Neural Information Processing Systems 9*. MIT press, 1997, pp. 176–182.

[50] E. Haglund, "Estimating prediction intervals with machine learning and monte carlo methods in online advertising," Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2020.

[51] D. Pevec and I. Kononenko, "Prediction intervals in supervised learning for model evaluation and discrimination," *Applied Intelligence*, vol. 42, 2015. doi: 10.1007/s10489-014-0632-z

[52] A. Habibi Lashkari. (2018) Cicflowmeter: Ethernet traffic bi-flow generator and analyzer for anomaly detection. University of New Brunswick. [Online]. Available: https://github.com/ISCX/CICFlowMeter

[53] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," pp. 1–6, 2015. doi: 10.1109/MilCIS.2015.7348942

[54] E. D. Source, "Intrusion detection evaluation dataset (iscxids2012) (06/11/2010 to 06/17/2010)," 2018. [Online]. Available: https://www.impactcybertrust.org/dataset_view?idDataset=916

[55] F. Liu, X. Wu, W. Li, and X. Liu, "The packet size distribution patterns of the typical internet applications," pp. 325–332, 2012. doi: 10.1109/ICNIDC.2012.6418769

[56] Cooperative Association for Internet Data Analysis (CAIDA), "Packet length distributions," 2020. [Online]. Available: https://www.caida.org/research/traffic-analysis/AIX/plen_hist/

[57] K. Lan and J. Heidemann, "A measurement study of correlations of internet flow characteristics," *Comput. Netw.*, vol. 50, no. 1, p. 46–62, 2006.

[58] M.-S. Kim, Y. J. Won, and J. W. Hong, "Characteristic analysis of internet traffic from the perspective of flows," *Comput. Commun.*, vol. 29, no. 10, p. 1639–1652, 2006. doi: 10.1016/j.comcom.2005.07.015

[59] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the characteristics and origins of internet flow rates," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, p. 309–322, 2002. doi: 10.1145/964725.633055. [Online]. Available: https://doi.org/10.1145/964725.633055

[60] Internet Engineering Task Force (IETF). (2021) IANA Assigned Internet Protocol Numbers. [Online]. Available: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

[61] M. S. Lewis-Beck and A. Skalaban, "The r-squared: Some straight talk," *Political Analysis*, vol. 2, pp. 153–171, 1990.

[62] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," *Journal of the Operational Research Society*, vol. 66, pp. 1352–1362, 2015. doi: 10.1057/jors.2014.103

[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations Conference Track Proceedings*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[64] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 6629–6640.

[65] S. Wenke and J. Fleming, "Contextual recurrent neural networks," *ArXiv*, vol. abs/1902.03455, 2019.

TRITA-EECS-EX-2021:651