Østfold University College

MASTER'S THESIS

Network Traffic Behavior Analysis

A method for pattern recognition in IoT networks

Iraj Lohrasbinasab

June 2023

Master's Degree in Applied Computer Science
Faculty of Computer Science, Engineering and Economics

Network Traffic Behavior Analysis

A method for pattern recognition in IoT networks

Master's Thesis in Applied Computer Science

Iraj Lohrasbinasab

Faculty of Computer Science, Engineering and Economics
Østfold University College
Halden, Norway
June 15, 2023

Abstract

Telecommunication networks and Internet of Things (IoT), as two important sources of big data generation, have grown significantly in the last decade. The data generation process in computer networks, in addition to the characteristics arising from the network environment, is defined by three typical dominant principles of big data including Velocity, Variety, and Volume, commonly referred to as the 3V. In response to this explosive increase in the speed and volume of generated data, in parallel with the development of data processing infrastructures such as Cloud, Fog, and Edge, great steps have been taken in developing network management methods. In network management, maintaining different issues such as supporting Quality of Service (QoS), Service Level Agreements (SLAs) and Key Performance Indicators (KPIs), security, and resource efficiency calls for consistent monitoring of the network. These concerns will be addressed on the foundation laid down by Network Traffic Monitoring and Analysis (NTMA) techniques like Network Traffic Prediction (NTP), Network Traffic Classification (NTC), security and privacy managmement techniques, and fault management. However, IoT, is mostly established on the resource-constrained ad-hoc networks, which impedes applying the heavy-weight NTMA's techniques, namely that of Network Traffic Analysis (NTA), in such environments. Change Point Detection (CPD) is an appropriate statistical technique that is gaining popularity due to its capability to offer lightweight and practical solutions for NTA in IoT ad-hoc networks. By observing and evaluating network traffic streams represented in the form of time series data in a near-real-time manner, CPD can provide an useful insight into the network traffic behaviour in terms of the metric(s) of concern.
In this master's thesis, we have proposed an efficient method of CPD that paves the way to effectively utilizing NTA in IoT ad-hoc networks. The proposed method builds upon an existing CPD method named TONTA and aims to enhance its performance in various aspects. By applying some innovative improvements to the base method, we have achieved a highly effective approach that outperforms its predecessor significantly. To highlight its coherence to the original method, the proposed method has been named **"μTonta"**. The report offers a thorough theoretical overview of the topic followed by a literature review on similar applications of CPD, which is supplemented by our independent research on statistical and machine learning-based NTP approaches. In the next step, along with analyzing the base method and identifying its potential for optimization, the proposed improvements are elaborated. Furthermore, we have conducted a comparative performance

evaluation of both methods that shows the proposed method has significant advantages over the base method in several key areas, including accuracy, time complexity, and detection delay.

Keywords: Network Traffic Monitoring and Analyzing, NTMA, Network Traffic Analysis, NTA, Change Point Detection, CPD, Machine Learning, Statistical Forecasting Methods, Network Traffic Time series Data, Internet of Things, IoT

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The IoT has changed the way we interact with our daily devices and the world around us [1]. From the technical lens, it offers an overlay network on top of existing network infrastructures, from cellular networks to Near-Field Communication (NFC) [2]. The major part of the IoT is established on so-called "ad-hoc" networks. Ad-hoc network is a general term referring to some types of networks constructed on-the-fly by a group of devices without the need for a pre-existing infrastructure, which are widely used in IoT networking. Monitoring and analyzing network traffic is vital to improve the performance of networking. Network management is an ongoing task, aiming to provide and ensure optimal performance of the network in various aspects such as availability, stability, security, efficiency, etc. To achieve these objectives, network management benefits from NTMA's capabilities. NTMA refers to a wide range of methods and techniques that provide a comprehensive and accurate perspective of the network, ultimately enhancing network management efficiency. To this end, NTMA relies mostly on analysing massive network traffic data streams in form of time series, and are therefore considered heavy-weight tasks. Generally, the common approaches to NTMA can be classified into one of two groups: machine-learning-based approaches and statistical approaches [3]. NTMA is an overarching concept that includes various aspects of network management and analysis, like NTP and NTA. NTP deals with predicting network traffic behavior, which involves forecasting future network traffic patterns considering historical data, while NTA puts the focus on gaining insights into the behavior and characteristics of network traffic through analyzing traffic data [4].
The unprecedented growth of IoT devices [2] increasingly demands for further efficient and effective techniques for NTA to monitor and manage IoT networks. However, some structural limitations in IoT ad-hoc networks impede the effective leveraging of NTA techniques in such environments. These challenges typically stem from the characteristics and structural limitations of ad-hoc networks, including limited resources, dynamic topology, and frequent changes in network traffic patterns. To deal with these challenges, some specific lightweight NTA solutions capable of operating efficiently in ad-hoc IoT networks are needed. In response to this need, some statistical techniques such as CPD are effective due to their lightweight nature and ability to detect changes in network traffic patterns. CPD techniques aim to identify abrupt changes in the statistical characteristics of a time series that can provide a profile of fluctuations in network traffic behavior.
In this Master's thesis, we propose a novel statistical-based approach for network traffic

behavior analysis in ad-hoc IoT networks. Our proposed method, called µTonta, builds upon the strengths of a base method called Trend-based Online Network Traffic Analysis in ad-hoc IoT networks (TONTA), which was introduced in 2020. The proposed method introduces several new features to improve TONTA's performance and makes significant contributions to the field of NTA for ad-hoc IoT networks. The thesis commences with a comprehensive review on the primary concepts in computer networks, a general IoT framework, the key features and concepts of network traffic data analytics, as well as review some related works on the topic of CPD. Meanwhile, coinciding with this thesis, a comprehensive survey on network traffic prediction was conducted and is accessible in Appendix A. We then introduce the base method, called "TONTA", which focuses on detecting change points in network traffic behavior using end-to-end delay data. TONTA is a lightweight, modular, and efficient approach that is well-suited for online traffic analysis of ad-hoc networks. However, there is some capacities for improving its performance measures, such as detection efficiency, time complexity, and speed. To address these challenges, we propose µTonta, a statistical-based method that enhances TONTA's performance by reducing time complexity and detection's delay along with enhancing the accuracy of change point detection in stream analysis for ad-hoc IoT networks.
Along with the proposed approach, this thesis also provides an in-depth outline of the adopted development and evaluation methodologies. Moreover, a comparative performance evaluation between the proposed method and the base approach, exhibiting the improvements achieved, has been conducted. This thesis is wrapped up by discussing the implications of our findings and ideas for future studies on applicable NTA techniques for ad-hoc IoT networks.

1.2 The Overall Direction of This Thesis

This thesis aims to tackle the issue of lightweight and agile solutions for network traffic analysis in ad-hoc networks. Given the current deficiencies in this area, the focus is on developing a statistical-based technique that can detect the behavior of ad-hoc IoT networks in real-time. To this end, we follow these steps:

- First, various definitions and contents related to IoT, ad-hoc networks, NTMA, etc are presented comprehensively to provide a concrete overview of what this thesis has improved.
- A comprehensive literature review is provided to outline the ML-based and statistical-based techniques for network behavior analysis challenges and future directions.
- A new model is proposed for network behavior analysis in such networks based on a method called Trend-based Online Network Traffic Analysis in ad-hoc IoT networks (TONTA) introduced 2021 [3].To highlight the relation between the proposed framework and the TONTA model, we have named it as"μTonta".
- The performance of the proposed method is evaluated in comparison with the TONTA.

1.3 Research Questions

The research aims to answer the following questions:

RQ 1: What are the current approaches to predicting network traffic , and how effective are they?

RQ 2: What are the challenges with utilizing NTA in ad-hoc IoT networks? And which approaches are more suitable for analyzing network traffic on ad-hoc devices in terms of accuracy and computational cost?

RQ 3: How can a method be designed to recognize the patterns of network behavior in ad-hoc IoT networks, and how does it compare to existing methods in terms of accuracy and computational cost?

1.4 Method

To answer the above questions, in this study, we, first, review the existing literature and tried to investigate various types of NTP techniques focusing on NTA. In addition, we review how the methods can be fitted into the ad-hoc IoT networks environment. In other words, we need to investigate which types of NTP methods can handle the characteristics of such networking environments. Then a new method is proposed to address the challenges of NTP in these types of networks. The performance of the proposed method is evaluated to show how the proposed method can achieve NTP goals in ad-hoc IoT networks. Last but not least, future works are considered as a road-map for doing research in this area.

1.5 Deliverables

Among the achievements of this master's thesis project, the following can be mentioned:

- *literature review:* A comprehensive review of the subject's literature was provided in a published survey (B). In this paper, the main approaches in NTP, including ML-based and statistical-based methods, are introduced and the features and common applications of each are explained. In addition, some of the latest research conducted in this field are introduced and categorized based on the type of approach adopted.
- *The proposed model:* A statistical-based model for network traffic behavior analysis in ad-hoc networks, called "μTonta", built upon the strengths of a base model called TONTA introduced in 2020. The proposed model introduces several new features to improve TONTA's performance and makes significant contributions to the subject.
- *Master's thesis report:* The report commences with a discussion of the theoretical aspects and components involved in NTP for IoT networks, and then the process of developing a proposed model through identifying the improvement opportunities in the based model is elaborated. The remaining sections of the report focus on a comparative performance evaluation between the proposed model and the base model by replicating the experiments and settings used in the source model.

1.6 Report Outline

The rest of the this thesis is summarized as follows: In Chapter 2 we first introduce the basic concepts and discuss the available types of techniques for applying NTMA tasks

in IoT networks. Chapter 3 is dedicated to introducing and analyzing the base method, followed by the elaboration of the adopted solutions for enhancing it, which have resulted in the proposed method. In Chapter 4, we have evaluated the performance of the proposed method in comparison to the base technique. then, The results obtained in relation to the research questions raised have been summarized and presented in Chapter 5. And finally, Chapter 6 is dedicated to wrapping up and conclusion of this research. Moreover, a supplementary research on the topic of NTP approaches, in which I, as the main author, have made contributions, is provided in Appendix A. in Appendix B some pieces of codes used in the main algorithms and procedures has been provided.

Chapter 2

Background and Theory

Understanding the concepts and phenomena that underlie computer networks is essential to any discourse in the domain of NTMA. However, due to the diversity of these notions and how they are intertwined with several distinct sub-domains, we have to narrow the focus to most relevant topics. To this end, we first introduce some elements of computer networks, including basic structures and features; then, the general NTMA framework, including its applications and components, is described; and finally, this chapter will be wrapped up with a brief look at IoT systems and involving issues with leveraging NTMA techniques in a specific type of IoT network called "ad-hoc."

2.1 Computer networks

Since the late 1950s, computer networks have come a long way. They have evolved to include a variety of network architectures, such as LAN, MAN, and WAN networks, underpinned by technologies like cable-based, optical, and wireless, as well as communication protocols. Over the last two decades, there has been a fundamental change in network technology and management in response to the exponential increase in data generation over networks [5]. Most computer networks are based on the packet-mode transmission model, in which data is formatted and sent in the form of packets with a standard structure, such as that shown in Figure 2.1. This structure enables efficient utilization of the communication media's bandwidth. However, computer networks vary widely in terms of other characteristics such as topology and scale. Common topologies are Bus, Star, Ring, Mesh, Fully Connected, and Tree networks (Figure 2.2).
The capabilities provided by technologies such as sensors and short-range communications have also led to the formation of a range of networks, which can be classified, based on their spatial range covered, into Nano Scale, Body Area Network (BAN), Personal Area Network (PAN), Local Area Network (LAN), Campus Area Network (CAN), Metropolitan Area Network (MAN), and Wide Area Network (WAN) (Figure 2.3).

2.1.1 Internet of Things (IoT)

The term "Internet of Things" describes a network of physical objects, so-called "things," capable of data generation and communication with computing infrastructures using sensors, software, and connectivity where The processing infrastructure consists of local resources and servers located at different hops away from end-nodes. IoT things are able to gather

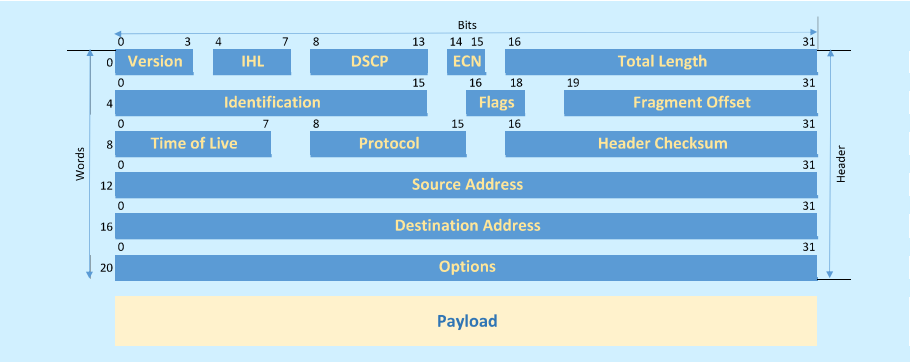


Figure 2.1: IPv4 Packet Structure

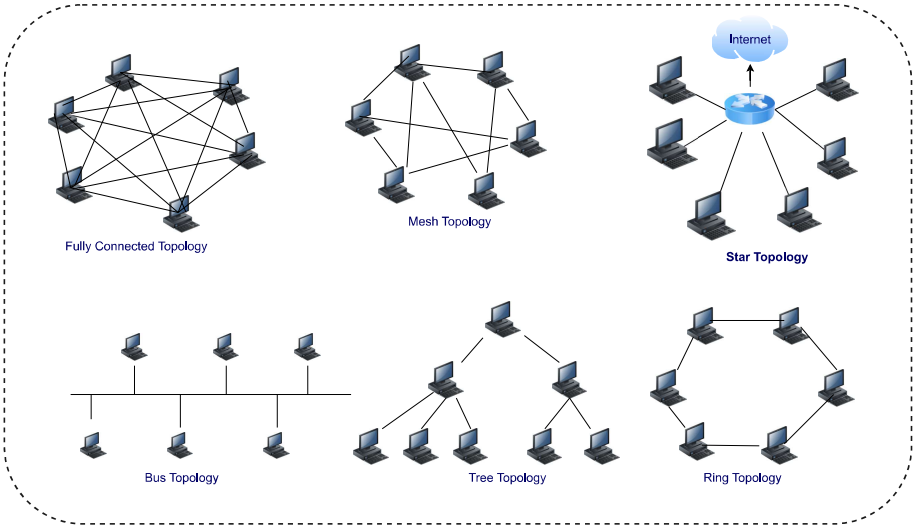


Figure 2.2: The most popular basic network topologies

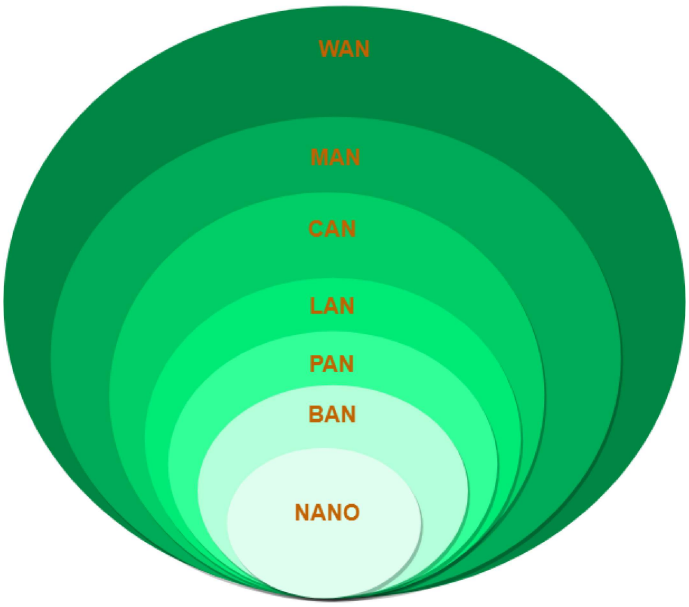


Figure 2.3: The typography of networks based on their spatial scope.

data from the environment in which they are housed and to exchange data over the Internet to allow automated decision-making and control. An IoT system's ultimate objective is to enable the seamless integration of physical or virtual objects, data, and processes over the internet, enabling real-time monitoring, analysis, and control of diverse systems and processes. Enabling wiser decision-making, increasing efficiency, cutting expenses, and improving user experiences are the objectives [6]. Figure (2.4) shows an overall schema of IoT networks.
The main stages that make up the data flow throughout IoT architecture are as bellow:

- *Sensing:* The collecting of data from sensors or other input devices, such as cameras, RFID readers, temperature sensors, motion sensors, and etc. is the first stage in the IoT data flow. These sensors gather information about a range of environmental elements, including temperature, humidity, light, sound, motion, and so on, depending on the sensors are used.

- *Processing:* The raw data gathered by the sensors is processed in the second stage. This processing may be carried out locally, on the device, or remotely, on a server hosted in the cloud. The data is cleaned, refined, and converted into a structured format for analysis.

- *Analysis:* To derive valuable insights and spot trends, the processed data are analyzed in the third stage. Several methods, including machine learning, artificial intelligence, and statistical analysis, can be used to conduct this analysis.

- *Communication:* The analyzed data are then sent to other IoT networked devices or applications in the fourth stage. This communication may take place through cellular networks, Wi-Fi, Bluetooth, or other communication protocols.

- *Action:* Adopting an action or decision based on the data analysis constitutes the last phase. Either an automatic process or human activity can start this procedure.

IoT Architecture

The design and deployment of hardware and software elements that provide the effective transmission of data flow throughout the IoT network are referred to as IoT architecture. IoT architecture typically consists of five layers:

1. *Devices and Sensors Layer:* The initial layer of the Internet of Things architecture is made up of devices and sensors, which gather environmental data and transmit it to the subsequent layer [6].

2. *Network Layer:* In this layer, the data is sent to the cloud or fog infrastructure over a network. Wi-Fi, ZigBee, Bluetooth, and other wireless and wired technologies are included at the network layer [7].

3. *Fog/Edge Computing Layer:* This layer consists of computing infrastructures made up of devices like routers, smart switches, smart phones, etc. that offer processing power near to the devices or sensors. This layer aids in lowering latency and enhancing the functionality of IoT applications [8].

4. *Cloud Computing Layer:* The fourth tier of the IoT architecture is the cloud computing layer, where the gathered data is processed, analyzed, and stored. To manage the enormous volume of data created by IoT devices, cloud computing offers the required infrastructure and services including storage, data analytics, and machine learning [9].

5. *Application Layer:* This layer facilitates the interaction with IoT system through user-facing applications With the ability of controlling data and devices.

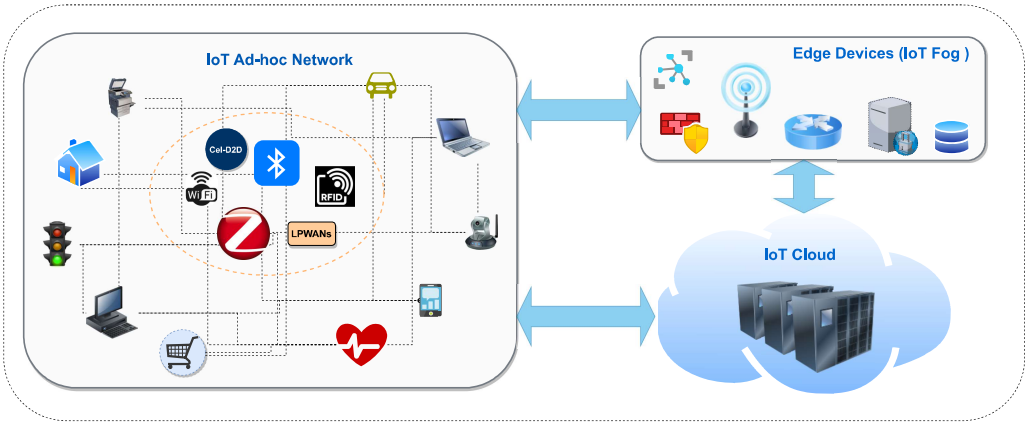


Figure 2.4: IoT Network schema

IoT Computing Infrastructures

This feature of IoT comprises all resource-handling tasks for computing, data processing, and storage retrieving in a real-time manner. Mainly, it consists of three types including Cloud, Fog, and Edge computing infrastructure paradigms.A schematic of IoT computing infrastructure is illustrated in Figure 2.6[10].

- **Cloud Computing:** The Cloud is a centralized infrastructure accessible via the Internet for data storage and retrieval, processing, and analysis that relieves users from the constraints associated with hardware and software resources by providing the various resources required by a distributed system such as the IoT in the form of instant services (Figure 2.5). Cloud services are generally provided in three modes including, Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Networking equipment, servers, infrastructure for data storage and retrieval, a hardware abstraction layer for virtualizing hardware, and software resources are the core elements of the Cloud. The key features of flexibility and scalability of the Cloud infrastructure provide the possibility of dealing with IoT big-data and real-time processing, which are the main components of the IoT's environment [11][12].

- **Fog Computing:** As a supplemental infrastructure for the Cloud, fog provides the capabilities for processing, storage, networking, and managing data on the part of the network that is near end nodes (for example, IoT devices and sensors). In general, efficiency increases with the proximity of the processing resources to the end nodes. By conducting some of the processing on the route between the end nodes and the Cloud, Fog seeks to decrease the workload and the number of tasks forwarded to faraway

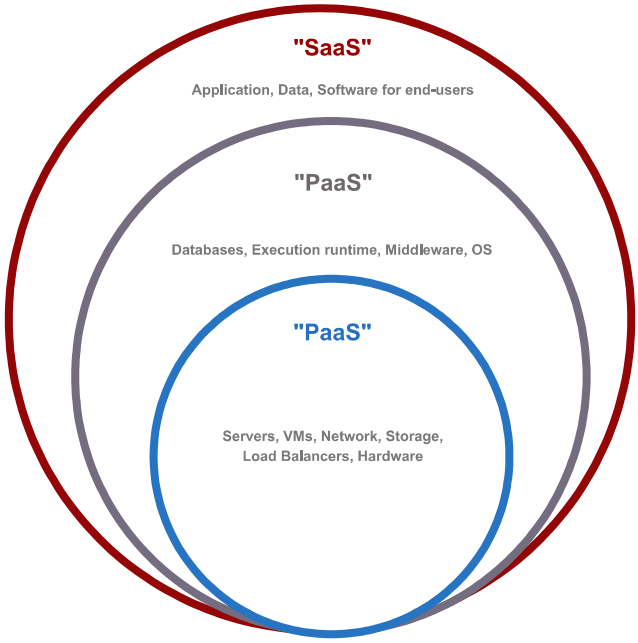


Figure 2.5: The most common Cloud services[12]

Clouds [13]. Fog computing, according to the Open Fog Consortium's definition is: "a horizontal system-level architecture that distributes computing, storage, control and networking functions closer to the users along a Cloud-to-thing continuum." In contrast to a vertical platform, which encourages isolated applications, the "horizontal" platform used in fog computing allows computing functions to be dispersed throughout many platforms and sectors [14].

- **Edge Computing:** Edge computing refers to available resources at the network's edge, not more than one hop away from IoT endpoints. These processing resources include a variety of devices with computing capabilities, such as routers, switches, access points, smartphones, etc. Although fog computing and edge computing both take place on the edge of the network, there are significant distinctions between them [15]. Fog is a complementary structure to support integration and continuity within the continuum of endpoints to the Cloud, in the core of the network, while edge computing deals with the resources located at the edge of the network as an isolated system and independent from the Cloud [13].

Ad-hoc IoT Networks

The IoT network infrastructure typically comprises two different kinds of structures: (1) Infrastructure-dependent: This is the typical design of traditional computer networks, which consists of fixed elements with specialised functions like routers, switches, etc. (2) infrastructure-independent or "ad-hoc networks" that refers to a range of wireless networks organized in a mesh topology with fully connected nodes. These networks can be classified as Wireless Sensor Network (WSN), Mobile Ad-hoc NETwork (MANET), Wireless Mesh Networks (WMN), and other types, depending on their specific purpose [16]. In ad-hoc networks as a main part of the IoT's network platform, the stated roles like routers and

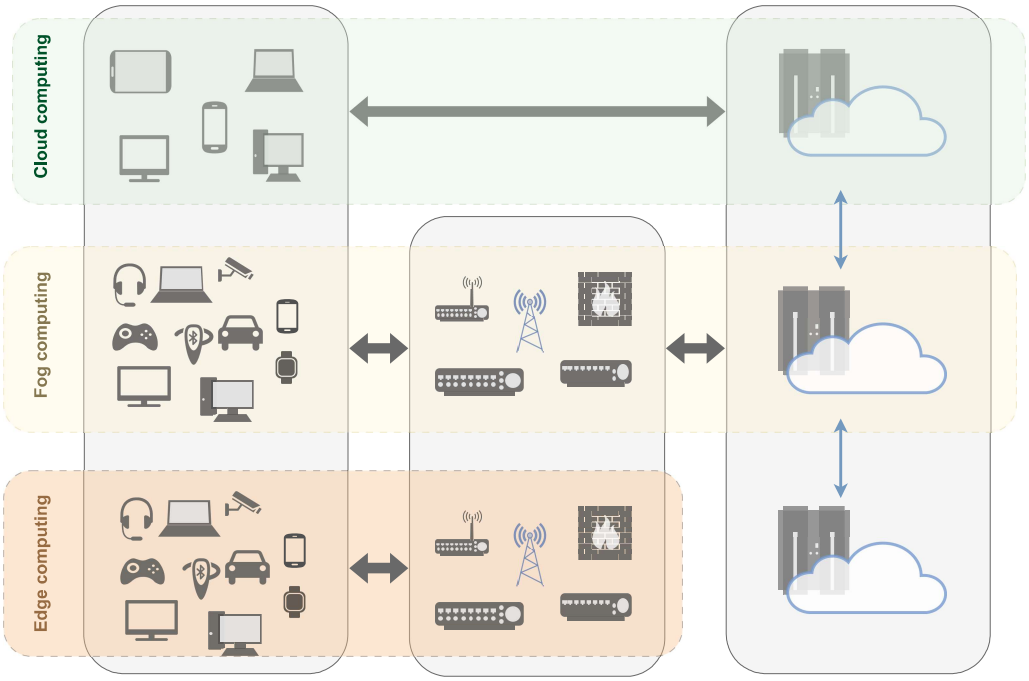


Figure 2.6: The IoT computing infrastructure including Edge, Fog, and Cloud

switches are carried out by some end nodes with appropriate processing and networking abilities. In temporary cooperation with each other, these components are responsible for the tasks assigned. Because these nodes are chosen based on the extra resources that are available at the time, their selection is not fixed and is subject to change at any time [17]. Such networks are under the general title of "ad-hoc networks," form a significant part of the IoT network infrastructure. In ad-hoc networks, nodes use various protocols to communicate with each other namely, Wi-Fi and Wi-Fi Direct, Bluetooth and BLE(Bluetooth Low Energy), Zigbee, 4G/LTE and 5G Cellular, long range(LoRa), Lightweight M2M (LWM2M), etc [18].

2.2 Network Traffic Data

Network traffic data can be assumed in a hierarchy in terms of granularity consisted of connection-level information, at the highest level, flow-related/time-dependent data in the middle, and finally, the most often used data in NTMA applications, namely packet-level data, is at the lowest level [[19]]. The packet-level data structure conforms to various network protocols such as TCP, UDP, and ICMP. A wide range of packet-based networks and the enormous number of various linked devices utilize packets to send and receive information [20].

2.2.1 Network Traffic Data Acquisition

In general, network traffic data collection techniques can be divided into three categories, Packet-based, Flow-based , and Connection-based , according to the granularity of these data. Packet-based data acquisition techniques involve collecting data at the packet level, which includes elements such as time stamp, source and destination IPs, source

and destination ports, and length [19], [21]. This type of data is essential for analyzing network traffic [22]. On the other hand, Flow-based techniques focus on analyzing groups of packets with similar attributes [23], such as source and destination, protocol types, and more. Statistical data about flows is analyzed based on classification types, including size, count, duration, direction, and rate [22]. At the highest resolution, connection-based data acquisition techniques provide even deeper insight into the nature of network traffic between two specific endpoints. This type of analysis involves extracting and inferring metadata flagged by other data collection methods and includes information such as connection duration, connection count, and connection type (TCP, UDP, or ICMP) [20].

Packet Inspection Methods

Capturing network traffic data in the form of packets underpins most of data extraction methods in network traffic data analytics. However, one of these approaches' drawbacks is packet loss, which is especially noticeable when dealing with enormous volums of network data. Furthermore, when it comes to data capture via high-speed lines, packet capturing performance deteriorates significantly. Generally, there are two main methods used for monitoring network traffic at the packet-level namely Deep Packet Inspection (DPI), and Shallow Packet Inspection (SPI). DPI has a broad range of usage in network security context where it is often known as stateful packet inspection or dynamic packet inspection, referring to the process of analysing the packet's header and payload data entirely. DPI usually is carried out at firewalls in order to detect and block malicious content through monitoring, control and filtering transmitting content [24]. Further, some kinds of buffer overflow attacks can be prevented by using this technique. Among the other most popular use-cases of DPI, one can mention to serving targeted advertising, lawful interception, and policy enforcement [25]. Despite the significant capabilities and advantages of DPI in the aforementioned use-cases, it is infeasible to leverage this technique for pace-attached scenarios of NTMA applications, and at resource-constraint devices. The SPI technique, on the contrary, is an alternative way that just examines the header data. This approach may be implemented by employing software and devices capable of sniffing packet header data, including source/destination IP, time stamp, etc. [19]. Some other deterrents to the utilising DPI approach in NTMA techniques include [25]:

- Accessing and analysing packet text information may violate users' privacy.
- Since the DPI method relies on processing all packet data, it takes more time and requires more resources than the SPI method, which suffices with little header data.
- In network scenarios that involve encrypting network traffic and using a VPN, DPI cannot be used due to the lack of access to packet content.

The above limitations have led most of the probes used in NTMA to exploit the SPI method. Packet analysis may be applied in a variety of ways, depending on the capabilities and processing capacity of the monitoring node [26]. On routers and switches, packet analyzer software is used to capture, store, and analyse packet data. TCPdump and Wireshark are two examples of such tools. These programs, among a range of profitable capabilities for monitoring and managing the network, also provide the possibility of extracting the delay data. In addition to common tools, packet sniffing can be performed directly with the help of the Libpcap library, a frequently used package that underlies the most packet analysing

products. The latter method is especially useful for use on poor-source devices. The direct packet capturing method could be applied in different ways; either in the system core, in the user space (i.e., programming environment), or using specific hardware [26], [27]. In this context, a broadly prevalent protocol is the Network Time Protocol, which propagates the time through a hierarchical structure from a precisely tuned clock server to lower-level nodes for synchronization.

2.3 Network Performance Measurement

Collecting network traffic data is one of the essential tasks that must be addressed in the initial steps of an NTMA procedure. The significance of this issue becomes more apparent, particularly in monitoring techniques based on active or passive probes [21]. Probes are one of the most efficient and common ways to gain insight into network status in terms of end-to-end, QoS, and QoE performance. Active and passive probes are two different strategies in this area. In the method based on active probes, by synthesizing a simulated network traffic flow and sending it within the network, we aim to evaluate the observed feedback and measure the performance parameters (e.g., one way delay) regarding the sent testing flow. Passive methods, on the other hand, focus on monitoring the whole actual network traffic on a given link. This structural difference leads to different perspectives on the state of the network [[22]]. Several aspects affect the aims and approaches used in collecting network traffic data through network management and the usage of NTMA, including network type, type of NTMA application (e.g., NTP, traffic classifications, etc.), policies and strategies adopted in network management, and so on [[28]].

2.3.1 Common Network Traffic Metrics

Given the area of interest of this research, some of the most widely used metrics in evaluating network performance in relation to QoS and Quality of Experience (QoE) are [29]:

- *Throughput:* Network throughput indicates the actual amount of data packets transmitted successfully via a network channel within a certain time slot [26]. It reflects the dropped packets' amount implicitly. Throughput is usually measured and shown in bits per second (bps), but bytes per second (Bps) or packets per second (pps) are also common [30].

- *Bandwidth:* Bandwidth refers to the accessible (or allowed) maximum data transfer capacity of a network channel at a given time. Since handling data streams beyond bandwidth capacity will trigger some proactive mechanisms at links, as well as increase the rate of packet loss and resultant problems, maintaining the data rate to this level, and avoiding exceeding it, plays a key role in guaranteeing optimal network performance [27], [30].

- *Jitter:* The condition of packet delay variation is known as jitter. In some applications requiring a steady transmission rate (e.g., video and voice traffic), exceeded jitter may disrupt communication entirely [28], [30].

- *Packet loss (loss rate):* One of the causes degrading the quality of end-user services is the loss or drop of data packets on the way from source to destination [30]. Given

the high speed and huge number of data packets being transmitted via networks, some packet loss is unavoidable, while TCP enables for packet monitoring and retransmission. However, beyond the defined thresholds, the high incidence of this issue hinders data requests being fulfilled at a committed time, disrupting consumer services [28], [30].

- *Delay:* One of the key functional features of the network used in predicting network traffic is delay (or latency). This feature, while simple and low-cost data extraction, effectively reflects the state of network performance to address issues like QoS and SLA maintenance [23], [28], [31].

2.3.2 One-Way-Delay (OWD)

In its most basic definition, this feature is the time between the transmission of a packet from the source sender and its arrival time at the receiver device . In practice, despite its simplicity, this definition is not without flaws. Due to the complexity of networks, this concept is tied to factors such as the scale and topology of the network, density of network nodes, distance from source to destination, protocols used, and etc [32]. Furthermore, OWD is strongly intertwined with clock synchronisation and most often, it is performed by utilising special hardware. Otherwise, relying on operating system-level or software-level techniques calls for clock synchronisation to an adequate degree between nodes that should make data packets timestamped [29]. The packet's OWD between two nodes over the network is the absolute time from the instant of sending the first bit of packet by the source device to the instant when the last bit of packet is received at the destination node. Devices may be servers, user terminals, routers, switches, etc., connected through a wired or wireless physical medium. As will be described later, different types of delay contribute to the OWD [21]. In most applications, given the asynchronous network traffic in round-trip routes, the delay related to one side of the packet communication from origin to destination is more interpretable and provides more useful information, in comparison to the overall delay of the round-trip data packet route (Round-Trip-Time) [26]. Typically, depending on the scale of the network (LAN, WAN, etc.), each data packet would has to traverse a route consisting of several nodes, such as switches and routers. As explained below, passing through each of these nodes in the route will impose some time delay on the packet transmission, which we denote by nodal delay (d_{nodal}). The total network latency between the source and the node receiving the packet results from the accumulation of the nodal delays created by the previous nodes [33].

- **Nodal delay:** The delays that a data packet encounters in a node can be divided into four major components that, when added together, constitute nodal delay [21], as shown in Equation (2.1).

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop} \tag{2.1}$$

  Equation (2.1) divides the nodal delay into a set of distinct delays based on their type and triggers making it easier to measure and analyse them. These components are outlined in more detail below [34]:

  1. *Processing delay* (d_{proc})*:* Processing delay is the time a node spends processing packet information, including error checking, reading a packet's header, and searching for a link to the next node by destination address. This kind of delay is usually negligible in compared to other types of latencies.

2. *Transmission delay (d_{trans}):* The time that it takes a data packet on the outgoing link to be entirely transmitted through the communication medium. This time depends on the transmission rate as well as the packet length.
3. *Queue delay (d_{queue}):* The queue delay refers to the time a data packet would have to spend in the queue on the outgoing link waiting to be sent. Queue delay is closely related to transmission delay (d_{trans}) and the queue's length (l_{queue}). This relationship can be roughly formulated as follows:

$$d_{queue} = d_{trans} * l_{queue} \tag{2.2}$$

   In equation (2.2), l_{queue} indicates the average queue length. l_{queue}, in turn, depends on the load factor. The latter feature is equal to the ratio of the actual transmission rate attempted by the link to its maximum transmission rate. The remarkable thing about the load factor is that by exceeding its value of 1, the queue length increases indefinitely, leading to queue overflow, which is one of the main causes of packet loss.
4. *Propagation delay (d_{prop}):* The amount of time it takes for a signal change to traverse the distance between two adjacent nodes via the available communication medium. The propagation speed is close to the speed of light where radio broadcast is used as a communication medium, whereas it drops to approximately 60% and 80% of the light speed for wired and fibre-optic networks, respectively.

- **Comparison of delay components:** Among the types of delays mentioned, processing latency is negligible, while the magnitude and effectiveness of other delays vary based on the scale of the network and traffic load on the sending links. Based on this, some general circumstances are foreseeable [33]. For example, in a LAN network, for light-loaded links, the transmission delay (d_{trans}) would be more significant than for other types of delays, whereas for high-loaded links, the magnitude of the queue delay overrides the other ones. On the other hand, this order for a WAN network, where the distance between two consecutive nodes may be hundreds of kilometers, changes in favor of propagation delay [34].

2.3.3 OWD Extraction Using Packet Sniffing Technique

In this technique, a sniffing tool is used to capure packets and examine their timestamps [30]. A timestamp is a snapshot of the current moment in time which typically would be associated with a certain "packet" or "event" by a network device. A packet timestamp is a value that indicates when a packet was sent through a network device [33]. In this context, two terms are distinguishable:

1. *The ingress timestamp:* This indicates the instant when the device receives the first bit of the packet
2. *The egress timestamp:* This indicates the instant when the first bit of the packet is transmitted out of the device

Depending on the monitoring setting and purpose, one can leverage timestamps in two approaches to measure the residence delay at a certain node, or calculate the overall latency between sender and receiver (i.e, OWD) A device can compute the resident time of a packet

by measuring both the ingress and egress timestamps, which reveals how long the packet has been on the current network device [35].
In the proposed method, the OWD parameter is considered as the target feature that can be extracted from the packet's time stamp data. The OWD can be calculated by catching the timestamp from the arrival packet header (i.e., the egress timestamp at the previous node) and comparing it to the time of receiving the packet (i.e., the ingress timestamp) on the node under observation. Although, as is discussed in section 2.3.2, there are different types of delay in packet base transmition, owd is the feature among others which effectively satisfys the demand for representative data reflecting the current state of the route traversed by a packet from the source to destination node. It also allows us to disregard the effect of some uncertain and constantly fluctuating factors like response time, and differentiation of the transmission and acknowledge routes as is seen in the overall delay. One-way delay (OWD) is strongly intertwined with clock synchronization [33]. most often, it is performed buy utilising special hardware. unless, relying on operating system-level or software-level techniques calls for clock synchronisation to an adequate degree [29].

2.4 NTMA

The NTMA refers to a diverse set of techniques that, through continuous monitoring of network traffic at various granularity levels (e.g., packet, flow, and application), provide a deep and applicable view of the network's status and performance, and user behaviour. This insight is critical to support network management tasks such as maintaining QoS and QoE, security, network resource allocation optimization, among others. Some of the most common areas for applying NTMA techniques are:

- To provide insight into the network's status and performance
- To detect suspicious network security issues
- To forecast traffic patterns and take preventative measures to maintain QoS and QoE
- To improve the network's resource allocation efficiency

NTMA techniques, in terms of traffic monitoring approach, can be divided into: 1) active and 2) passive. In active approach, a synthesized probe traffic is injected into the network at specified time intervals to evaluate its status, in real-time, in terms of network performance criteria such as latency, packet loss ratio, and jitter. Given providing a real-time insight to network performance, active methods are the primary approaches to determine whether the actual state of the provided services conforms to the expected level within the SLA framework [36].
In contrast to active approaches, passive methods analyse past data of real network traffic in offline mode. These techniques may be used to manage and plan for increased performance and productivity. Their ability to precisely monitor network traffic, especially in post-event scenarios like fault tolerance and diagnostics, as well as evaluate and monitor the QoE, is remarkable in this context.
In network management, maintaining different issues such as supporting QoS, security and resource efficiency call for consistent monitoring of the network. These concerns will be addressed on the foundation laid down by NTMA.Generally, NTMA comprises various sub-fields; some of their most prominent and widely used ones, including NTP, NTC, fault

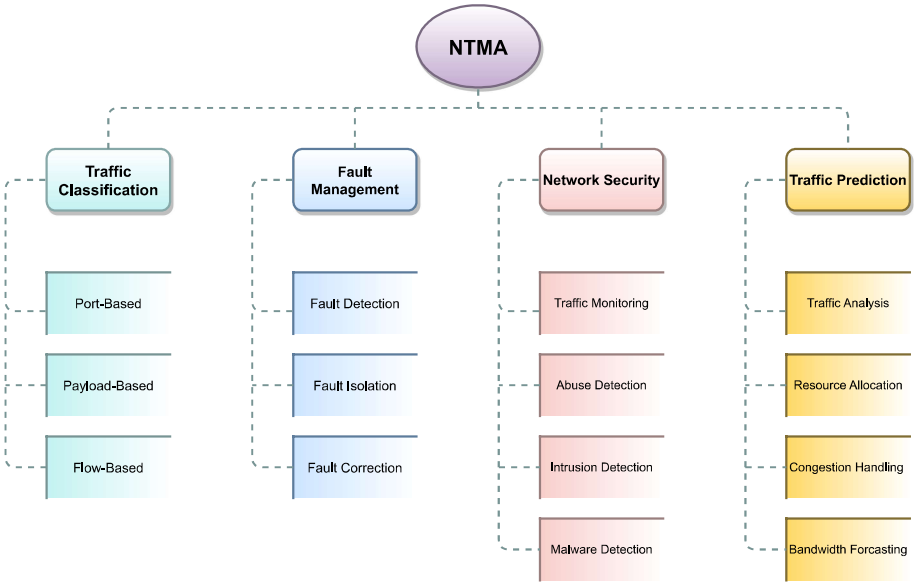


Figure 2.7: NTMA sub-fields

management, and network security, as well as the sub-techniques used by each, are shown in the Figure 2.7. Moreover, each of these managerial tasks, on the other hand, has its own specific goal, requirements, methodologies, and a set of criteria and determinants that must be set up within unique situations based on the nature of the problem at hand [23].

2.4.1 NTMA Framework

The more networks grow in terms of scale, complexity, and volume of traffic data, the more advanced and efficient approaches and techniques will be needed to establish the network's availability, stability, and efficiency. In most studies, the general framework offered for NTMA is, more or less, the same. As it will be discussed, this framework may be formulated into the following main steps.
(i :) At the highest level, i.e., the first step, the main purpose of the NTMA application and related functional goals are defined. As stated in the introduction, NTMA techniques could be used for NTP, NTC, security oriented tasks like anomaly detection, or other purposes. The main goal(s) must be defined clearly and precisely while determining certain sub-goals as prerequisites for obtaining the primary goals is usually required, as well. For instance, if the primary purpose of employing NTMA is to predict network traffic, then determining breakpoints in network traffic data will be one of the sub-goals to serve the primary goal.
(ii :) The second step involves collecting network traffic data using active and passive methods. Given the different views obtained through each of these monitoring methods, combining them together can lead to a more comprehensive and accurate view of the status and performance of the network. (iii :) The data must next be preprocessed and cleaned. This step is critical, especially in applications like NTP, because refining the data and removing duplicate and superfluous elements like re-sent packets and repeated ACKs can greatly improve the ML model's performance.
(iv :) Network traffic data contains a range of parameters, some of which are crucial

depending on the kind of NTMA applications, while others have no meaningful and decisive association with the usage context. Based on this, applying feature selection as a subsequent stage of data preprocessing constitutes an important part of the NTMA's procedure. Although algorithms created for this purpose undertake most of the feature selection work automatically, a significant portion of the task is left to data analysis experts to be accomplished manually, relying on their domain knowledge.
(v :) Performing the aforementioned steps on the raw network traffic data enables data engineers to extract useful data related to the primary goal for which NTMA is performing through in-depth data analysis. For this purpose, various methods are available that can be classified into statistical, ML-based, and big-data methods [23]. Each of these methods focuses on specific aspects of data analysis and serves different purposes. One of the most important tasks is to identify and select the most appropriate and relevant solution, considering the nature and inherent requirements of each NTMA application in a way that ensures accuracy, usefulness, and reproducible results.
Identifying the pattern of data changes, and predicting the trend of these changes, is the dominant aspect and the main purpose of most NTMA applications in the field of network management [37].

2.4.2 Feature Selection in NTMA

The raw data gathered using data collection methods allows for the compilation of a wide range of metrics that may be used to analyze the status of the network from various perspectives. Choosing the appropriate metrics is a critical step in network monitoring and management [28][38] that entails consideration of several factors of the subject matter being monitored [31]. However, some general principles to consider while selecting the appropriate metrics are:

- The existence of a meaningful relationship between the metric in question and the goal set in the NTMA application (e.g., QoS and SLA status assessment, security, or resource management) [23], [39]
- Availability of the considered metric within the requirements and limitations of the technique adopted in the NTMA process.
- Replicability of the generation process and utilisation of metric data [28], [39].
- Ease of use in terms of data collection, processing, and analysis methods.
- Cost-effectiveness in terms of the resources required for measurement, storage, and processing [31], [39].

2.4.3 Challenges With Network Traffic Data Analytics

In comparison to traditional big-data analytics, what distinguishes the NTMA process is the set of some environmental factors, intrinsic needs, and limitations that make network traffic data generation encounter certain unique challenges. The following are some of the factors that influence network management data [23], [36]:

- *Heterogeneity:* Due to the great variety of connected devices on the network, the data produced or consumed forms a vast range in various aspects, such as data structure, data volume, data production rate, communication protocols, etc. This is especially the case in IoT networks, where a wide range of nodes, such as smartphones, sensors, vehicles, and other IoT devices, generate or consume data in different ways.

- *Data generation and streaming rate:* One of the distinguishing and inherent aspects of the network traffic data generation process is the high rate of data production and transmission in the form of continuous and bulky streams. This feature is especially noticeable in scenarios such as media streaming, online gaming, and P2P-based applications. These specifications face network management with persistent challenges in terms of resource management and the efficiency of adopted techniques for data collection, storing, and analysis, especially in areas that serve QoS and QoE.

- *Strong temporal-spatial correlation:* The increasing prevalence of smartphones, which typically would host a bunch of different apps, as well as the pervasiveness of multimedia platforms and social networks, are among the factors that have extremely amplified the penetration of the Internet into various areas of daily life. Along with this progress, the pattern of network users' activity has also evolved from simple data transfer based on more or less rigid and predictable scenarios to complex and fluid behaviours with high dependence on temporal and geographical factors. Thanks to achievements in communication technologies, especially in IoT and cellular networks, as well as improvements in multi-hop processing infrastructures, we are witnessing ever increasing applications whose distinctive feature is mobility. In this sense, smartphones with a plethora of data-generating apps and autonomous vehicles (AV) are both well-known examples. As a result of these shifts in user behaviour patterns, network traffic data frequently exhibits significant temporal-spatial correlation.

- *Security and privacy:* The widespread use of network traffic encryption techniques and security protocols such as Hypertext Transfer Protocol Secure (HTTPS) as well as the use of VPNs in Internet communications has significantly improved security and users' privacy. Nevertheless, that puts the use of NTMA in a conflicting position due to restricting access to network packet information and source-destination information.

- *Noisy data:* The environment of generating and sending network traffic data at its various stages, including generation, sending, routing, and receiving, is prone to interruptions and errors. Noise detection and removal from network traffic data is a fixed and important component of a communication system's tasks. In this context, packet loss and jitter are both common occurrences.

2.4.4 Challenges Of Applying NTMA Tasks In Ad-hoc Networks

The capacity of NTMA techniques to support network performance management has gained considerable interest from both industry and academia. However, as we see in the case of the IoT, along with the consistent growth of computer networks in all directions, including prevalence, scale, and complexity, as well as the emergence of new network technologies and paradigms, managing and supporting network performance has become even more challenging [23].
The IoT network is one of the main sources of big-data generation, where massive data

streams are constantly produced by a huge number of interconnected components that are capturing and transmitting their operating environment data over the Internet; However, it is not only the tremendous volume of data that has made IoT the focal arena for innovation and research in NTMA techniques; characteristics such as diversity of data generation patterns, heterogeneity of data sources, speed of data production, and severe temporal and spatial correlation of data have also made management of this type of networks more complex and challenging than conventional computer networks and calls for further improvement [3].

2.5 Network Traffic Prediction

NTP is a general term for the process of predicting changes in a network performance metric based on observed changes in its previous values and current state. The aim of traffic prediction is to forecast traffic performance metrics, such as bandwidth, packet loss, and delay, up to a specific horizon. An extended horizon shows traffic patterns in subsequent stages and provides sufficient time to take appropriate action [40]. However, this process can be pursued in different ways, both in terms of the final purpose and the way it is realized. In predicting network traffic fluctuations, the desired horizon for prediction plays a key role in choosing the NTP method. Depending on the intended application, forecasting can be focused on a long-term or short-term horizon. Long-term forecasting usually covers ranges between milliseconds and minutes, while short-term forecasting focuses on predicting values of the investigated parameter in time frames less than milliseconds. In applications such as resource management, it is usually necessary to recognize the pattern of changes for the regarded metric by using long-term forecasting techniques such as Trend Change Detection (TCD) and CPD. On the other hand, short-term forecasting is focused on the one-step-ahead horizon to predict instantaneous fluctuations, as is common in applications like congestion detection and dynamic bandwidth allocation [41]. In an NTP process, several factors are concerned in choosing the appropriate method and model to proceed the task. The final purpose of performing NTP (the NTP application in request), the forecast horizon (short/long-term), characteristics related to network traffic data such as dataset size, data feeding type (batch or stream), attributes of concern, as well as other characteristics such as available processing resources, are among these factors. However, in a broad sense, the existing approaches could be categorised into statistical analysis models and supervised machine learning models [42].

2.5.1 Change Point Detection

One of the most popular approaches in NTMA, which uses statistical analytic techniques to uncover the pattern of data changes in a time series, is pattern recognition. Different performance criteria may be sought in network monitoring. One may get an overview of the present and future status of the network in terms of the desired measure by carefully selecting a metric that properly represents the intended performance criterion, monitoring its changes over time, and analysing the pattern of these changes. For instance, measures like throughput, jitter, and latency can offer a precise and useful perspective of network traffic behaviour. These data are produced as time series data streams that accumulate temporary datasets and exhibit a certain pattern that may be identified by pattern recognition techniques. CPD is one of the widely used techniques in pattern recognition

that divides the time series dataset into separate segments according to the trend of the data points [43]. Implementing CPD may be done in one of two ways:

- *Offline:* The full dataset is processed at once in this scenario. This approach is suitable for situations that call for long-term forecasting.
- *Online:* In this case, the data is received as an intermittent or regular stream of data, and the purpose of monitoring this data stream is to detect and interpret the meaningful changes in the data at the moment in order to take an appropriate reaction to the detected trend immediately [44].

Typically, the process of analysing network traffic behaviour through CPD includes two tasks. First, the points in the dataset where the dominant trend changes (i.e., CPs) are identified, and then the direction and intensity of the dominant trend between two successive CPs are determined. In online CPD applications, the data stream is received on-the-fly, and according to the necessity of immediate response to trend changes, CP detection is done in near-real-time. In addition, trends are classified according to their direction, intensity, and duration, and in the form of descriptive terms such as "increasing," "decreasing," and "level-off." The lightweight of CPD techniques and their lack of training and retraining are the most important advantages of these methods compared to ML-based techniques [3].

2.6 Big-data Analytics

From a data analysis perspective, big-data refers to a situation where, extracting information from data is under the pressure of factors such as volume, velocity, variety, and veracity. Applying NTMA techniques to high-speed, complicated and dynamic networks like IoT and mobile is one of the main areas, among others, where the big-data phenomenon is present and influences extensively. Modern networks have been increasingly dominated by the relatively new phenomena of big-data, which stems from the fundamental characteristics of network management data, such as volume, velocity, variety, and veracity [8], [32]. Network traffic data refers to all the data that reflects the network's state and situation. So, the circumstance will not appear surprising when we consider that the main source of such data is the tremendous volume of packets flowing, in the form of high-speed streams, between the huge number of linked devices and end-points. This puts NTMA at the forefront of big-data consumers and makes it one of the most important areas of research in big-data analytics.
In general, common data processing techniques in Big-data analytics can be divided into two categories [22]:

- Mathematical and statistical methods based on time series analysis
- Data mining approaches, and methods based on machine learning algorithms (for example, SL for predicting network traffic in applications such as congestion handling.)

The collection and conversion of raw data into useful information in each big-data analytics process , per se, requires a series of complex steps. However, when it comes to NTMA, the process faces additional challenges arising from concerns such as accuracy, data generation rate, and real-time big-data processing. Aside from these concerns, the rapidly changing

nature of networks and the exponential growth in the number of connected devices, as well as the emergence of new paradigms like vehicular ad-hoc networks highlights the need for more efficient big-data analytics methodologies in terms of processing load, processing speed, and accuracy, to name a few. Beyond conventional big-data analytics, given the need for real-time or near real-time processing in most time sensitive IoT applications, IoT data calls for fast and streaming data analytics, which is not possible to be accomplished in the Cloud. This circumstance highlights the necessity of lightweight and small-scale data analytics working at the edge layer or on resource constrained IoT devices. In the following, the two primary approaches in dealing with big-data analysis problems in network environments i.e., statistical and ML-based approaches will be discussed. However, before delving into statistical methods, it is crucial to grasp the time series concept and it's general attributes, which is widely employed in statistical techniques [39].

2.6.1 Time Sries Data; Definition and Attributes

The term time series refers to any phenomenon that is observed consecutively over time and at certain intervals. The mentioned time interval can be any measurable time unit such as milliseconds, minutes, hours, weeks, years, etc.[45] In each time series at least one of the following patterns can be observed that reflect the behaviour of data during the time period:

- *Trend:* A trend is a pattern that demonstrates how the relative values of a time series rise or fall over an extended period of time. The slope of the time series change graph that represents the trend would be either rising or decreasing.
- *Seasonality:* Seasonal pattern occurs when the time series fluctuations are dependent on a fixed frequency. In other words, seasonality refers to similar patterns that repeat in constant time periods.
- *Cyclic Pattern:* Cyclic pattern refers to those fluctuations in the time series that are not bound to a fixed period of time. In general, if the fluctuations in the time series are of a constant frequency, they indicate a cyclical pattern; if the repetition frequency of fluctuations is an unchanged time unit, they display a seasonal pattern.
- *Noise:* The remaining after decomposing a time series into trend, seasonal, and cyclic would be a random fluctuation without any predictable behaviour. Figure 2.8 shows an instance of such data graphs.
- *Stationarity:*The concept of stationarity refers to time series whose statistical characteristics (mean, variance, etc.) do not depend on time. In a more precise sense, "statistical behavior" of a stationary time serie doesn't change over the time which could be considered as a constant probability distribution in time. Simply put, for any randomly-selected pair moments of a stationary time series, that segment reflects the general behavior of the process. Based on this, stationarity in a time series can be effectively expressed in the form of the relationship between the γ coefficient and the autocovariance between any pair of arbitrary points of the time series, as shown in Equation 2.3.

$$\gamma_y(k) = Cov(y_t, y_{t+k}) \tag{2.3}$$

  Where, k is the number of time lags. The two takeaways from this equation are: (1)

the expected value of the time series is independent of time. (2) the autocovariance function defined as $Cov(y_t, y_{t+k})$ is only a function of k and is unaffected by time. As a result, a strong and slowly fading ACF will also point out departures from stationarity when analysing the pattern of a time series process. a sample of a stationary data and its ACF graph in presented in figures 2.8 and 2.9.

Accordingly, time series with a trend or seasonality would be non-stationary. On the contrary, a time series without trend and seasonality, so-called "white noise," is considered stationary. Stationarity in a time series means unpredictability, especially over long periods. Therefore, to approximate the pattern of changes in a stationary time series, statistical estimation techniques come into play, focusing on providing an estimate of the future values of the time series with the least possible error. Figure 2.10 demonstrates an example of such process.

- *Differencing:* Differencing process could be thought of as substituting a time series with its own modified one in order to get rid of persistent patterns that dominate the data. as a matter of fact, by performing a proper-order differencing on a non-stationary time series, it would yield a stationary time series. In this process, a time series of values is differentiated into a time series of changes in values over time. The method used to achieve this most frequently is to compute pairwise differences between two adjacent time points, where the value of the differenced series at a given time t is equal to the value at time t minus the value at time $t-1$. Although, from a theoretical aspect, there is no limitation to performing more levels of differencing, in practice, differencing of order one or two is the most often used [46]. To make it easier to describe the differencing process and also to apply linear algebra to it, the concept of 'Backshift Value' with the symbol B is used (in some sources, the concept of Lag Value symbolized by L is used instead) [45].

  Assuming that:

  $$By_t = y_{t-1}$$

  then,

  $$B^d y_t = y_{t-d}$$

  And based on this, the values of L variable y can be shown as follows:

  $$y\prime_t = y_t - y_{t-1} = (1 - B)y_t$$

  Which can be generalized as:

  $$y_t^d = (1 - B)^d y_t$$

  that is, the d_{th}-order difference of y [45].

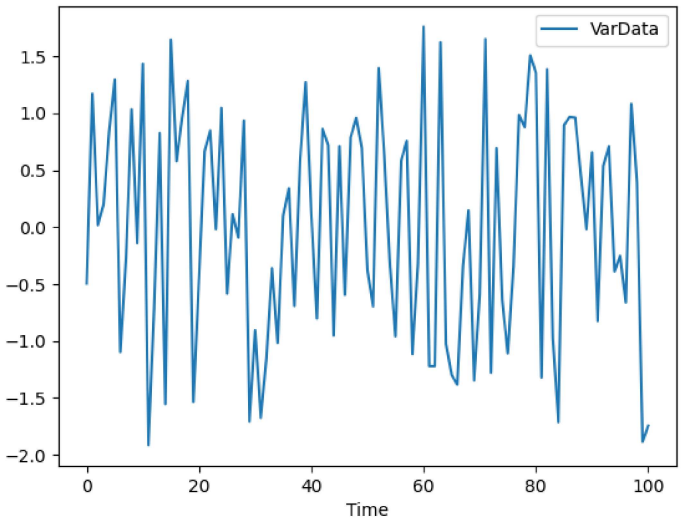


Figure 2.8: A white noise graph shows stationarity

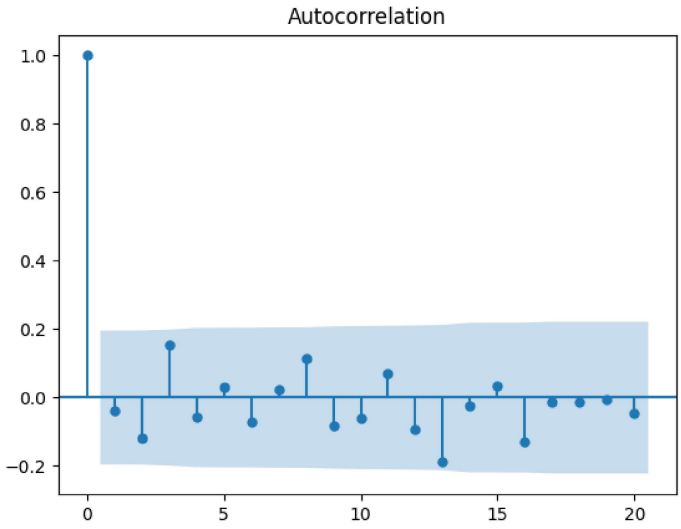


Figure 2.9: The sample autocorrelation function for 20 first observations of the data in Figure 2.8 with a boundary around ±2

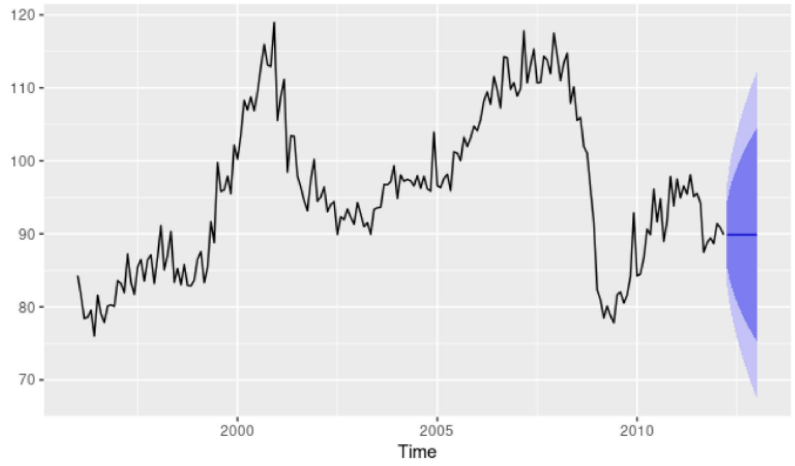


Figure 2.10: The blue areas indicate the confidence intervals for the approximated next value, established by a statistical estimation methods (image adapted from [45])

2.6.2 Statistical Analysis and Forecasting Techniques

Statistical models have good efficiency in the linear analysis of time series data. These methods focus on the correlation between the previous values of a time series variable (lagged data) and its future values, so that by presenting the model of this aouto-correlation, they provide the possibility of predicting future values. When discussing a time series forecasting, it is assumed that each one consists of both a deterministic and a random component. This assumption is based on the idea that the unpredictable behavior of a time series is due to random noise that is completely independent of time. Although this assumption is generally correct and forms the basis of many prediction techniques like exponential smoothing, there are situations where it may not hold true in practice. based on this, even in time series with a high degree of stationarity, serial dependencies between succeeding observations may be recognised. This situation has laid down the foundation for the invention of methods that, in the most effective way, exploit the serial dependence feature of observations for prediction. The most pervasive of these methods is a series of forecasting models grouped under a general class named ARIMA [47]. In this section we discuss ARIMA models and the simple linear regression model.

Auto-Regressive Models (AR)

An auto-regression model leverages the concept of auto-correlation to forecast the next value in a time series. Simply put, it is based on the fact that the value of a variable at any time (t) can be represented as a linear combination of its previous values plus an additional value called error. The term auto-correlation emphasises the fact that the model represents the regression between the variable and its lagged values. Say y as the desired variable and p as the order of the model (i.e., the number of lagged values considered in this linear combination), and the amount of error created at time t is shown by ε_t, an auto-regressive model can be expressed as Equation 2.4:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t \tag{2.4}$$

Normally, the application of AR models is limited to stationary data and up to order 2, and the following constraints are applied to the model parameters:

- for $p = 1 : -1 < \phi < 1$
- for $p = 2 : -1 < \phi_2 < 1, (\phi_1 + \phi_2) < 1, (\phi_2 - \phi_1) < 1$

[45]

Moving Average Model (MA)

Moving Average model is another popular linear forecasting model which in comparison with AR model, instead of using the previous values of the desired variable, uses previous consecutive prediction errors (i.e., 'ε') in a linear combination as shown in Equation 2.5:

$$Y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \tag{2.5}$$

the model is referred to as MA(q), where ε is white noise and q is the order of the MA model. Moreover, the equation highlights the fact that each value of Y_t can be assumed as a weighted moving average of forecast errors belonging to a given number of recent lags. Since the MA model deals with an abstracted parameter i.e. ε, rather than the relevant variable itself, it could be considered somehow a pseudo-regression process.

Auto Regression Integrated Moving Average (ARIMA)

The ARIMA model, which is frequently used in time series forecasting for the near-term, is a generic class of statistical models constructed by integrating AR and MA models [46]. Since the fundamental premise of this model is that the time series are stationary, it is required to apply differing to make our time series stationary in lack of stationarity. In this situation, it may be necessary to combine differing with non-linear transformations like logging and deflating throughout the time series transformation process.
A time series is stationary if its statistics remain the same across time. In this scenario, there is no permanent or long-term pattern to the time series, and its range of variations will be a fixed value centered around the mean. This means that the time series' short-term patterns at arbitrary time intervals, are identical. The key implication of such a condition is that a time series' random variable's autocorrelation, which means its correlation with the deviation from the mean in previous moments, remains unchanged over time. Such a random variable can be thought of as a combination of noise and signal, while the signal may be any regular and predictable pattern (e.g., sinusoidal oscillation, frequent sign shifts, or gradual/accelerated mean reversion), potentially displaying certain seasonal behaviors as well. Based on this, an ARIMA model can be considered as a filter that decomposes the time series into noise and signal in order to enable prediction by extending the signal into the future [48]. The ARIMA model uses a linear equation to predict stationary time series, in which the predictors are a linear combination of random variable lags and forecast errors associated to those lags. The model in its full form can be written as Equation 2.6:

$$y\prime_t = c + (\phi_1 y\prime_{t-1} + \cdots + \phi_p y\prime_{t-p}) + (\theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}) + \varepsilon_t \tag{2.6}$$

Some spesific modes of ARIMA(p,d,q)	
ARIMA(0,0,0)	White noise
ARIMA(0,1,0) when constant = 0	Random walk
ARIMA(0,1,0) when constant ≠0	Random walk with drift
ARIMA(p,0,0)	Autoregression (i.e., AR(p))
ARIMA(0,0,q)	Moving average (i.e., MA(q))

Figure 2.11: The ARIMA (p, d, q) specific modes

where $y\prime_t$ is the differenced time series considering that more level of differencing is possible, and "ε" is the error term. This equation is notated by ARIAMA(p, d, q), where:

- p is the order of the autoregressive part;
- d is the degree of first differencing involved
- q is the order of the moving average part.

Equation 2.7 formulates the ARIMA model in another form using backshift operator (B).

$$\Phi_p\left(B\right)\left(1-B\right)^d\left(Y_t\right)=\Theta_q\left(B\right)\varepsilon_t, \tag{2.7}$$

where :
$\Phi_p\left(B\right)=1-\varphi_1 B-\cdots-\varphi_p B^p$ and $\Theta_q\left(B\right)=1+\theta_1 B+\cdots+\theta_q B^q$,
$\left(1-B\right)^d=\sum_{j=0}^{\infty}\left(\begin{array}{c} d \\ j \end{array}\right)\left(-1\right)^j B^j$ with $\left(\begin{array}{c} d \\ j \end{array}\right)\left(-1\right)=\frac{\Gamma(-d+j)}{\Gamma(-d)\Gamma(j+1)}$ and $\varepsilon_t\sim\ N\left(0,\sigma^2\right)$
as the error terms [49].
In addition to the extant components of the conventional AutoRegressive Integrated Moving Average (ARIMA), the Seasonal-ARIMA (SARIMA) [50] model also includes the frequency component (known as *seasonality* and shown by (S) [51]. An SARIMA model would conduct prediction based on a linear combination of past observations and their related errors. As its name implies, the seasonality factor plays a key role in the structure and performance of this model. The SARIMA process often would be shown as models in the form of $SARIMA(p,d,q)\times(P,D,Q)S$. For given time series $\{Y_t\}$ with seasonality length (S), the SARIMA process is indicated by the Eq. (2.8), while the differenced series $w_t=(1-B)^d(1-B^S)^D y_t$ is a stationary Autoregression Moving Average (ARMA) process and, d and D are non-negative integer values.

$$\phi_p(B)\Phi_P(B^S)w_t=\theta_q(B)\Theta(B^S)\epsilon_t \quad ,t=1,2,\ldots,n \tag{2.8}$$

Where:

- n is the number of observations up to time t and the backshift operator B is defined as: $B^a W_t=W_{t-a}$.
- ϵ_t is defined as Identically and Independently Distributed (i.i.d) samples with a zero mean and variance σ^2, which for all $K\neq 0$ we have: $Cov(\epsilon_t,\epsilon_{t-k})=0$

- The non-seasonal components are:
  - AR: $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$
  - MA: $\theta(B) = 1 - \theta_1 B - \cdots - \theta_q B^q$
- The seasonal components are:
  - Seasonal AR: $\Phi(B^S) = 1 - \Phi_1 B^S - \cdots - \Phi_P B^{PS}$
  - Seasonal MA: $\Theta(B^S) = 1 - \Theta_1 B^S - \cdots - \Theta_Q B^{QS}$

As an another ARIMA's extension, Fractional AutoRegressive Integrated Moving Average (FARIMA) is a generalization of the ARMA model customized to support those applications like NTP in which, besides short-term dependencies, there are considerable linear long-term dependencies between the observations. Unlike the ordinary ARIMA process, the difference parameter (d) in FARIMA model could take non-integer values [52]. The general FARIMA process is expressed in Eq. (2.9) where B is the backshift operator.

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right) (1 - B)^d y_t = \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \varepsilon_t \tag{2.9}$$

[47][53].

GARCH

One of the other standard statistical models widely employed in time series problems is that of Generalized Autoregressive Conditional Heteroskedasticity (GARCH) [54]. This model is an extension of the Autoregressive Conditional Heteroskedasticity (ARCH) model innovated by Engle in 1982 to estimate the target variables' volatility [55]. The main goal is to model the changes in variance of target variables whose part of the total variance is conditioned on lagged values of target variance and model's residuals. To this end, the concept of Conditional Variance (also referred to as Conditional Volatility) plays a key role. Considering $\{\epsilon_t\}$ as a real-valued discrete-time stochastic process, as:

$$\epsilon_t = \sigma_t w_t$$

where w_t is discrete white noise given i.i.d ($\mu = 0, \sigma^2 = 1$), the GARCH(p,q) process is then denoted by Eq. (2.10).

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \tag{2.10}$$

Where α_i and β_j are model's parameters meanwhile, to avoid negative variance the following constraints are imposed [56],[57]:

$$\{p \geq 0, q > 0, \alpha_0 > 0, \alpha_i \geq 0, i = 1, \ldots, q, \beta_j \geq 0, j = 1, \ldots, p\}$$

in addition to combined approaches based on integrating different models with ARIMA, some hybrid models based on the ARIMA's foundation have been proposed. For instance, Fuzzy-AutoRegressive Integrated Moving Average (Fuzzy-ARIMA) is an invented method which fuzzifies ARIMA's parameters using the fuzzy regression method [58]. In terms of acronyms, as seen sporadically in some references, Fuzzy-ARIMA is also referred to as FARIMA; which should not be confused with the Fractional-ARIMA model. Meanwhile, The latter is recorded as ARFIMA in some sources as well [59].

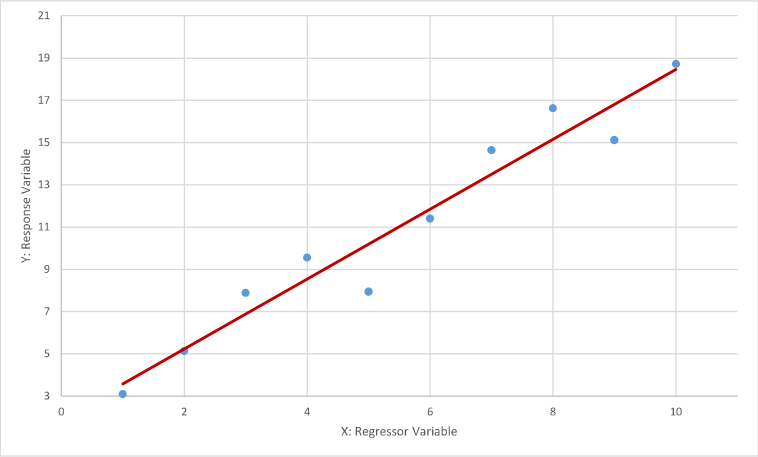


Figure 2.12: A typical regression model

Linear Regression Analysis

Regression analysis is a statistical method for analysing and modelling the relation between one or more predictor variable(s), (also known as regressor variable,) and an outcome variable (also known as response). The model developed via this inquiry is used to forecast the future values of the response variable based on the values taken by the predictor variables. The general Equation 2.11 does formulate the link between one predictor variable and its associated response variable.

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{2.11}$$

where, y is the response variable, x is the predictor variable, β_1 and β_0 are the regression parameters, and ε indicates the error value. As can be seen in Figure 2.12, the linear regression relationship actually gives the equation of a straight line where the slope of the line (i.e. β_1) is the amount of change in the mean of the dependent variable (y) per unit change in the predictor variable x. The term ε in the regression equation reflects the deviation of the actual data from the predicted value (points on the regressor line) and is usually referred to as statistical error.The error term ε assumed to be independent and identically distributed (i,i,d) variables sampled from a normal distribution with zero mean and constant standard deviation. The distribution function of such variable is generally represented by the symbol $N(0, \sigma^2)$)[47].

2.6.3 Machine Learning Based Approaches

Another group of widely used methods and tools in data analysis can be grouped under the title ”machine learning-based methods.” ML is a sub-branch of AI that extracts patterns from data using learning algorithms and makes predictions and decisions based on those patterns. Overall, ML works well for that kind of problem in which there is an adequate

amount of relevant data. Basically, the problems in which ML could be involved are categorized into the following fundamental tasks:

- *Clustering:* Clustering refers to the inclusion process of the objects examined in a set of distinct groups based on the similarity of their characteristics. The basic condition in this grouping is that the similarity between members of a group must be greater than their similarity with those of other groups. A well known example of use of clustering in the networking context is Elephant/Mice flows detection.
- *Classification and Regression:* Classification task refers to predict and mapping data points to a finite set of discrete quantities or nominal values while, estimating a continuous-value feature is what a regression task entails. To name a few, some most common classification algorithms are Linear classifiers, decision trees, support vector machines, and random forest, whereas the frequent regression methods are linear regression, logistic regression, and polynomial regression.
- *Rule Extraction:* Another sort of problems that benefits ML is the rule extraction task. Rule extraction is focused on identifying the dominant statistical correlations among data, which makes it fundamentally different than other types of tasks.

To address such problems, ML techniques are designed and used in the framework of various scenarios and settings to uncover the concealed patterns within data. An overview to some of those learning scenarios and settings, which are more relevant to NTMA context, are discussed in the following.

Learning Paradigms

ML techniques and models have been developed in the framework of approaches that we call Learning Paradigms. Obviously, these approaches are adopted according to the on-hand problem, in order to satisfy the essential criteria of it. In general, four main types of LP can be listed as: (*i*) Supervised, (*ii*) Unsupervised, (*iii*) Semi-supervised, and (*iv*) Reinforcement, which are briefly introduced.

- **Supervised:** An SL algorithm's goal is to build a model from the existing data. It gets a feature vector as input and gives the predicted value for the label of that vector as an output.[60] In SL, data is a collection of labelled samples in the form of $\{(x_i, y_i)\}_{i=1}^{N}$. In fact, such a data set can be thought of as a matrix with each column denoting an attribute or dimension of the data sample, and rows that represent the feature vectors (X_i) [61]. The label Y_i, a variable that relies on the feature vector, can belong to a finite set of classes, including nominal values, real numbers, and even more intricate structures like matrices, vectors, and so on. The sample's class is indicated by the label's value.[62]
- **Unsupervised:** in unsupervised learning, machine learning algorithms are used for analysing and clustering unlabelled data sets. These algorithms are referred to as "unsupervised" since they identify hidden patterns in data without the assistance of an individual. Clustering, association, and dimensionality reduction are the three basic tasks that unsupervised learning models are applied for[61].
- **Semi-Supervised:** In real-world scenarios, we frequently encounter situations in which a sizable portion of the data is unlabelled for a variety of reasons. In problems with

the nature of SL, the pragmatic solution to analyse and model such data is to use the SSL approach. At first glance, it may seem that unlabelled data is just a source of uncertainty and that only labelled data is useful in the learning process. However, some crucial information on the problem can be discovered in this portion of the data. The probability distribution of the labelled data, as a valuable additional information for the learning algorithm, is more accurately depicted the greater number of these data points there are. [62], [60],[61]

- **Reinforcement:** Reinforcement Learning (RL) encompasses some specific type of machine learning algorithms where the machine is constantly interacting with its environment, allowing it to perceive the state of that environment as a vector of features. A RL algorithm's objective is to learn a policy. A policy is a function that takes the feature vector of a state as input and produces the best course of action to take in that state (similar to the model in supervised learning) [61]. If the action maximises the expected average reward, it is the best course of action. In every state, some actions can be performed by the machine. Each action leads to a different reward and has the potential to alter the environment around the machine. In order to build a state-action pair table during the learning process, the algorithm randomly explores the state-action pairs within a given environment. Once the algorithm has learned the information, it then uses the state-action pair rewards to select the best action for each state that will lead to a particular goal state [60]. Figure 2.13 illustrates the general schema of the RL process.

Model Training Settings

Having the suitable and, if feasible, unbiased representative data is critical for developing an efficient model for a problem [42]. Because of the variety of data in various networking applications, as well as the alternation of data over time, it is critical to choose an appropriate approach for data collecting for ML methods to train their models. The pattern of data generation and transmission in networks is truly diverse and fluid. This situation, as a set of environmental characteristics of a network ecosystem, imposes its own requirements and limitations on a ML-based solution. Therefore, it is not possible to create and train a universal model for all types of networks, and inevitably, the collection and injection of training data into the model will be affected by the characteristics of the host network ecosystem [63].
Data collection is often carried out either offline or online. The former approach uses all of the data to train the model at once, and then the model is deployed to apply to operational data. However, in the online method, training would begin concurrently with deployment in the operational environment, and model would update itself by incoming input datapoints received in sequential order [64], [65]. Those models that depend on learning from lagged data points (i.e., supervised, semi-supervised, and unsupervised) often read the training data in the form of batch feeding, while RL-based techniques such as State–action–reward–state–action (SARSA) [66] and incremental learning techniques [67] are mainly designed to update the model gradually by incoming data points.
Depending on the context and model training setting, training the model is accomplished in one of two ways: batch, or incremental (also known as streaming). In a batch scenario, the dataset is divided into three subsets: training, test, and validation. Separating a part of the dataset as validation data helps to choose the suitable model, while the training and test

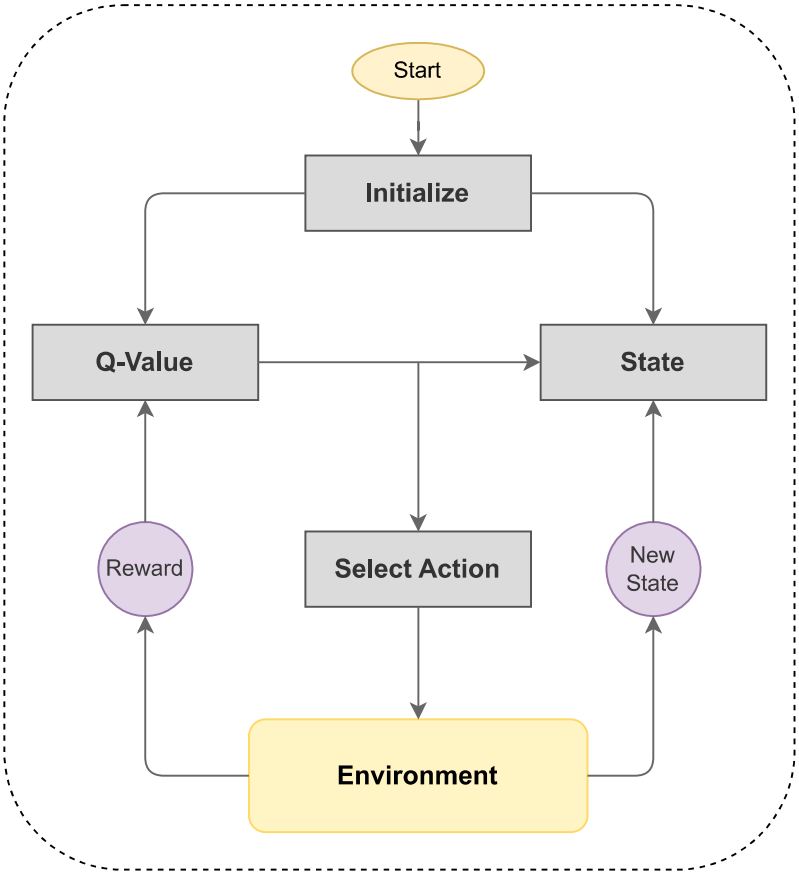


Figure 2.13: a general Reinforcement Learning process

sets are leveraged to adjust the model parameters (e.g., the weight of connections between neurons in a neural network (NN)) and evaluating the model's performance, respectively. The incremental technique, on the other hand, streams input to the training model for a variety of reasons (for example, enormous volume and impossibility of loading at once, or gradual production of data) [64], [65]. Furthermore, in dynamic situations, particularly online applications, the ML model must be constantly retrained. Given the significant computational cost of starting from scratch in such circumstances, employing only recent data for training is a cost-effective option provided by the incremental method while also being helpful in solving the drifting problem [68]. Figure 2.14 shows different types of collecting data to train ML models in networking.

Machine Learning Algorithms Classification

Machine Learning Algorithms Classification: A variety of algorithms and techniques fall under the umbrella of machine learning, with each one tailored specifically to satisfy the requirements of a certain set of problems or criteria. ML algorithms can be categorized based on how they process and use data to build models, and their internal structure in terms of transparency and the number of inferring layers. Based on this, different criteria can be considered to distinguish ML models, including, model-based vs. instance-based learning. The former refers to developing the model and adjusting its parameters using training data, while the latter indicates modeling the whole dataset based on data samples. Moreover, ML models can be divided into two main types known as Shallow Learning and Deep Learning. Shallow Learning involves methods that learn their parameters directly from training data in a single step, while Deep Learning models use a multi-layer network in their structure called Artificial Neural Networks (ANN), where each layer learns its parameters from the output of the preceding levels. The number of layers in ANN determines the model's depth. On the other hand, from the application view, ML models can also be categorized according to the NTMA problems they are applied in. Among these applications we cane refer to security, performance, and fault management. In [20], some popular algorithms along with the common areas in which they are leveraged are listed as shown in the table 2.1.

Deep Neural Network algorithms

Deep Learning (DL) is a sub-field of ML which in its core benefits a multi-layer structure called Artificial Neural networks (ANN). Inspired by the structure and function of interconnected neurons in the human brain, ANN mimics some of their basic mechanisms using a multi-layered network of computing nodes. Figure 2.15 depicts the overall anatomy of a neuron. Each neuron in a certain layer is linked to all or some of the neurons in the next layer, and the outputs from neurons in the preceding layer are processed as an input feature vector. The internal mechanism of a neuron can be represented by an equation as:

$$y = \sum(w_i * x_i) + b$$

this processing unit is designed in such a manner that the relevance of each component of the input vector can be decided by assigning weights to them. To put it simply, the basic premise of the work done throughout the model training process is altering the weight coefficients plus the amount of bias offset in a way that leads to the best prediction or classification in the output layer.

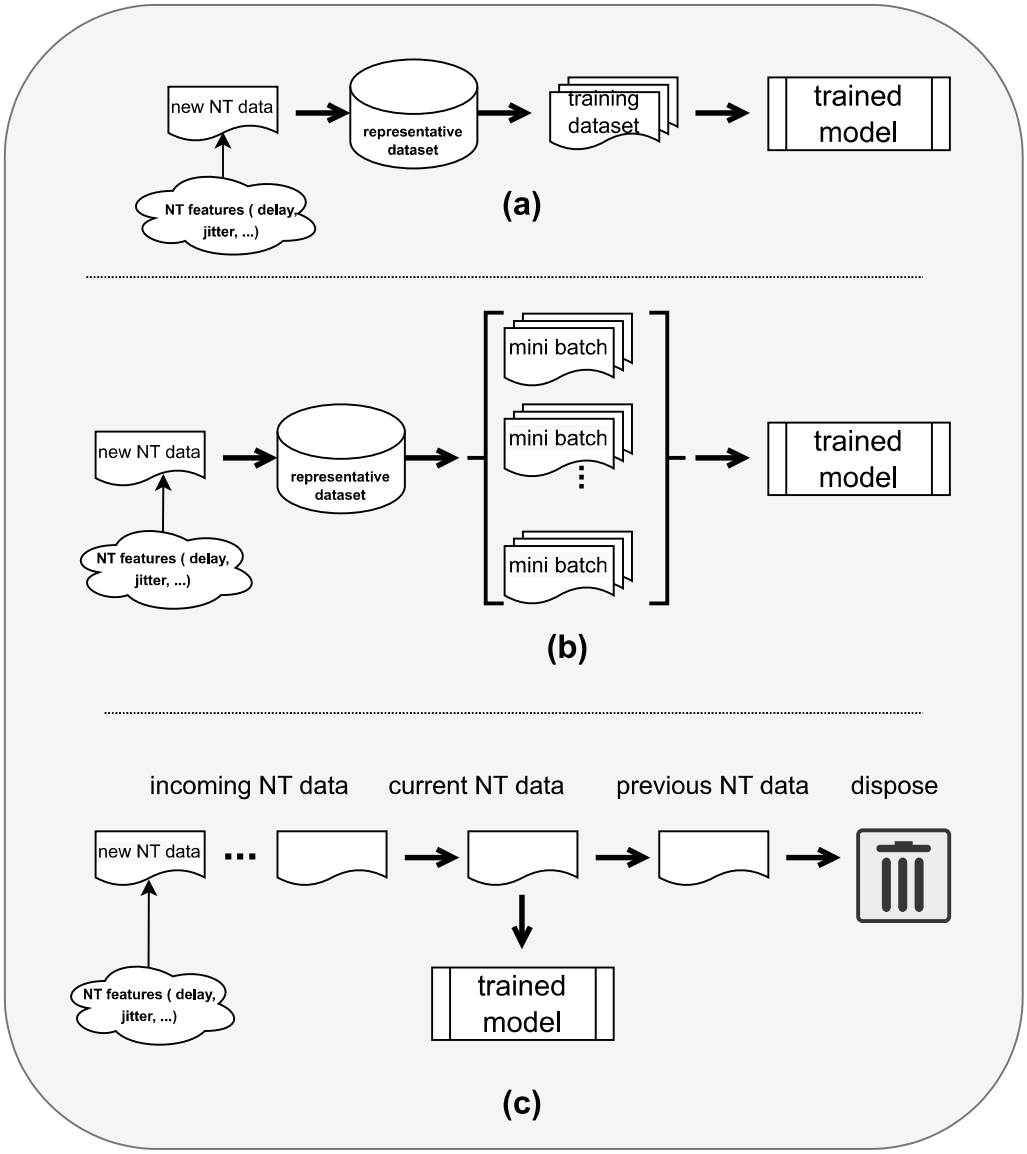


Figure 2.14: Data collection settings, *(a)* Batch Offline *(b)* Batch incremental *(c)* Online incremental (streaming)

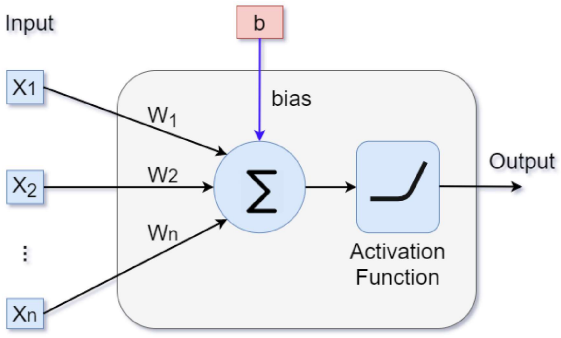


Figure 2.15: Typical structure of a Neuron

Management area	Task	ML techniques
Performance	Network Traffic Prediction	(Ensemble) NN, BN, SVM,
	QoE-QoS correlation	DT, BN, SVM, Q-learning
	Adaptive resource allocation	Q-Learning, Deep
	Adaptive service configuration	Q-Learning
Security	Misuse detection	NN, DT, BN, SVM
	Anomaly detection	(Ensemble) NN, DNN, k-NN, k-means, (Ensemble) DT, Ensemble BN, SVM
Fault management	Fault detection	NN, k-NN, k-Means, DT, BN, SVM
	Fault localization	NN, k-NN, k-Means, DT
	Automated mitigation	BN, SVM

Table 2.1: popular ML models in networking

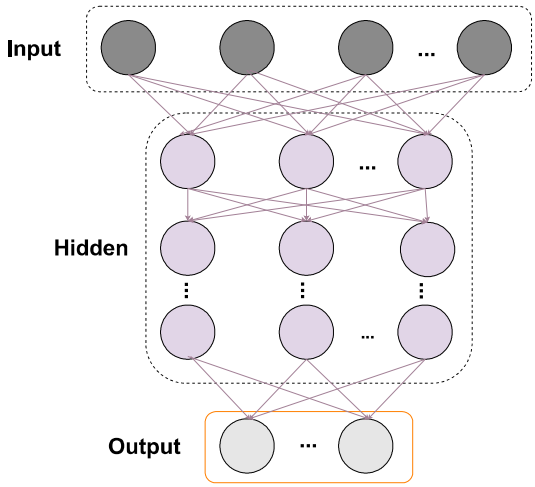


Figure 2.16: The the schematic of Deep neural network

The inputs and outputs of ANN are known as visible layers. The input layer is where the deep learning model ingests the data points in the form of feature vectors. The final prediction or classification generation takes place in the output layer, based on the inputs received from the hidden layers. Between the input and output layers, there is a set of successive layers called "hidden layers." One of the key parameters in describing an ANN model is "depth," which indicates the number of hidden layers in the model. ANNs with a depth greater than two are known as Deep Neural Networks (DNN)"deep neural networks." Figure 2.16 shows the general scheme of a DNN. In this structure, each layer takes its inputs from the previous layer and, after applying weighting, projects the result as an output to refine or optimise the prediction (or classification). This mechanism is called 'forward propagation.'
Back propagation is another mechanism that is frequently utilized in ANNs. This mechanism is a repetitive cycle of trial and error that calculates and evaluates the error created in the output during model training using algorithms such as gradient descending and allows the weights and biases in the neurons to be modified in some way, resulting in the closest prediction or classification. In conjunction, FP and BP enable prediction and refinement of the prediction, accordingly. In recent years, NTMA, like many other disciplines of DL use, has benefited from the unique characteristics of DL in various application like as NTC and NTP. to name a few, FFNN, RNN, CNN, and LSTM, among lots of existing DL algorithms, are the most prevalent ones in NTMA context. In general, deep algorithms offer two major advantages over traditional ML methods:

- Deep learning has eliminated the feature engineering step through analysing and choosing desirable and effective features during the learning process, which increases performance compared to manual feature engineering.

- Deep models are inherently involved with large amounts of input data, which enhances the model's efficiency in terms of accuracy and loss.

However, some drawbacks with DNNs have limited the general use of them. For instance, in cases of poor-resource environment, few training data, or frequent changes in data patterns Utilization of these algorithms would face some pitfalls. the long training time and the vast computing resources required to proceed with the training task, in parallel with the fact that DNN models inherently show their best performance on large amounts of data, could be mentioned as limitations in this respect.

Model Selection

Given the variety and multiplicity of ML algorithms and techniques, choosing the proper model is one of the most important steps in the data analysis process. Since it is practically impossible to achieve the optimal model through a trial-and-error process, including some criteria could focus attention on a narrow number of potential models. Among these requirements are:

- *Explicability:* The visibility and traceability of the path taken by a model to achieve its predictions would place it in one of two categories: opaque models, sometimes called as "black-boxe" (e.g., NN and ensemble models), or explainable models (e.g., KNN and DT).

- *In-memory Vs. out-of-memory:* The nature and characteristics of the problem's environment dictate whether the dataset has to be loaded into RAM all at once or gradually and in chunks, as with incremental learning.

- *Number of features and data points :* The size of the dataset and the number of characteristics in each instance are important considerations in model selection. While certain models, such as SVM, work well with limited features, NN and gradient boosting methods can handle large amounts of data with many features.

- *Training Speed:* The type of application of the algorithm, the conditions of the problem, and the limitations of the host ecosystem determine the conventional and acceptable time for learning and constructing the model. Generally, simple models such as SVM , DT , Logistic Regression , and Linear Regression are very fast in this respect, while nonlinear and complex models such as NN are notorious for their long training time.

- *Other considerable Factors:* Among other noteworthy factors, we can mention whether the features are categorical or numerical, linear or non-linear in the data, and the time complexity of the algorithm.

2.7 Related Works

A comprehensive survey (Appendix A) on NTMA and common NTP methods has been published in which I, as the main author have made contribution [69]. In this research, along with the theoretical aspects of the subject, we have reviewed and categorized the existing research on the subject conducted in recent years. The evaluated works have been assessed in terms of the insight and methodologies used, including ML-based and statistical methods as well as integrated and heuristic approaches. The survey indicates the wide application of neural network techniques as the dominant techniques in research that deals with the topic with an ML-based approach, and on the other hand, the central role of the ARIMA family algorithms in the statistical approach is underlined. In this section to wrap up that evaluation, we discuss some instances of works accomplished by focusing on the topic of CPD in different applications of NTMA.
The application of CPD covers a wide range of purposes. There are many different use cases for CPD applications. Examples of such scenarios include handling processing load fluctuations on network resources, monitoring topological changes, tracking odd behavioural changes of network traffic brought on by cyberattacks, and improving IoT applications in smart homes, etc.
Anomaly detection in network security is one of the typical uses of CPD, as was highlighted in the discussion's backdrop. In this context, the accomplished research by Segura et al. in [70] and [71] could be referred to as an instance. These two research exhibit a great affinity with the subject of this thesis in terms of dealing with the issue in a CPD framework along with focusing on resource limitations in SDN networks and the lightness of the proposed solution as an essential feature. In [70] the authors have established a lightweight DDOS attack detection method tailored to the constrained SDN environments using the CPD technique. Their adopted approach relies on the non-parametric Cumulative Sum (CUSUM) test for the detection of changes in the concerned metrics as well as an indicator called MACD to identify the direction of those changes by determining the moving average of

convergence/divergence in the data sequence. They have extended their proposed method in [71] by considering both centralized and distributed modes. Depending on the setting in which this technique is used, metrics specific to each mod are considered. In the centralized mode, the metrics of interest are the control packet overhead and the data packet delivery rate of the network, while in the distributed mode, the system deals with the local metrics of the hosting node. The proposal is a non-parametric statistical approach, that benefits from a CUSUM-based CPD algorithm at its core to monitor the metrics in question, and alert an attack at the abrupt change of the mean value of metrics.
Song et al. in [72] addressed network performance measurement at the application level using a probabilistic Bayesian change point-based estimation method in the context of QoE maintenance in cellular networks through congestion handling and bandwidth estimation. In this work, the available bandwidth estimation is thought of as a change point detection problem focused on observing the transport block size, over the aggregated data sequences from the response time to prob signals, as a congestion indicator.
Network topology monitoring is important from different aspects, especially network security and spotting abnormal changes brought on by cyberattacks. In this domain, dynamic observation of network topology is another area of application for the CPD technique, addressed by Liu et al. in [73]. The introduced method in this research is aiming at discovering the minor changes in the network topology at certain intervals. Their approach is based on statistical process control in which the network topology as a dynamic state is reflected by a graph, constructed through measuring the topological metrics of the network over time. A cumulative sum method (CUSUM) is employed to discover trendy changes in topological parameters.
Aminikhanghahi et al. in [43] have addressed the issue of activity recognition in the context of IoT health systems in smart homes, employing a CPD-based model to segment time series data received from the behaviour-driven sensor network in a health-monitoring system. Their proposed method focuses on recognizing time points of change in the pattern of data generation, which reflect activity transitions and improve the performance of this process compared to traditional window-based methods. In the same vein, in ref[44] and [74] they have presented an online CPD method by density ratio technique called SEP. It is an unsupervised non-parametric method that examines and detects change points in multi-dimensional data using probability density as the primary criterion. In their proposed method, by comparing the probability density function derived from two data subsequences before and after a given time point, their difference is measured in terms of probability density parameters, and if it exceeds a certain limit, it is considered a change in trend.

Chapter 3

The Proposed Method

3.1 Introduction

Using ML-based techniques for performing network management tasks in ad-hoc IoT networks is faced with certain structural obstacles including heavy weighted process and abundance of unlabeled data. As an alternative option, for some applications such as NTP, It seems that statistical methods are more suited to the constraints of ad hoc networks. In general, IoT network infrastructure can be classified into two categories: (i) infrastructure-dependent, referring to established networks with a fixed framework. In this framework, certain hardware and software infrastructures, such as routers and smart switches, are in charge of networking tasks. (ii) infrastructure-independent or "ad-hoc". Ad-hoc networks form the major part of the IoT network infrastructure. Despite structural distinctions between ad-hoc and conventional networks, the need for monitoring and analyzing traffic data in such networks, as in infrastructure-dependent networks, persists. NTMA tasks are increasingly relying on ML-based techniques, which, despite their efficiency and accuracy, are considered heavy tasks in terms of required processing resources. In infrastructure-dependent networks, resource-rich devices such as routers and switches are well able to perform such processes, while in ad-hoc networks, since the nodes involved are typically low-resource devices, effective implementation of NTMA techniques faces serious challenges. Two prominent cases of the challenges of applying ML-based methods in ad-hoc networks are:

- *Overhead processes of training and retraining:* The need for training and periodic retraining in response to the continuous changes in the network's environment and data is one of the barriers of these methods in IoT ad-hoc networks. The application of ML in the network has significant differences from most other applications, from the sense of how to train and retrain the model. Due to the special features of the network ecosystem, the ML model used in such an environment, in addition to training, needs frequent and occasional retraining. This situation is caused by some factors, including intermittent and rapid changes in network traffic behaviour, continuous changes in network topology, the increase and decrease in the number of nodes, etc., all of which go hand in hand and create an extremely dynamic environment. Training and retraining of ML models, in addition to imposing a significant processing load on the hosting devices and degrading network performance, are very time-consuming processes. This is especially important for IoT networks, which are highly time-sensitive, and is one of the main obstacles to using ML techniques in ad-hoc networks.

- *Abundant unlabelled data:* Most of the tasks of NTMA, such as network traffic classification and network traffic prediction, are formulated as Supervised Learning (SL) problems. SL-ML models need labelled data to operate at their optimum, whereas real-world network data are either largely unlabelled or their labels cannot be retrieved on time. Deep learning techniques have been extensively embraced as a solution to this issue because of the remarkable efficiency they have demonstrated in semi-supervised learning environments; nonetheless, these models are notorious in the sense of required resources and lengthy training times.

In response to the demand for lightweight methods for effective monitoring and analysis of ad-hoc network traffic, the investigation of methods compatible with the structural limitations of such networks is the main goal of this research. In this regard, and intending to offer an efficient method for online pattern recognition of the changes in network traffic data, statistical methods appear to be a suitable solution. The inherent advantage of statistical methods for dealing with time series data, combined with the capacity to collect and process network traffic characteristics such as delay in the form of continuous streams of time series data, as well as considering no need for training in this methods, all together makes them a favourable alternative to traditional methods. As one of the popular statistical pattern recognition techniques, CPD refers to the process of identifying time points at which a time series data set exhibits an abrupt shift in its behavior. The CPD procedure in the NTMA context seeks to identify the time intervals at which a major trend shift in network traffic data happened [44]. Compared to other ML-based pattern recognition methods, the CPD process based on statistical techniques is a very lightweight alternative that does not require training or retraining, either.
In this research we concentrate on enhancing a CPD technique called TONTA. The ultimate goal of this technique is to detect the occurrence of changes in the trend of network traffic data flow while also recognizing the type of current trend in a near-real-time manner. We can mention TONTA's significant overlap with the topic at hand, as well as the existing capacity to develop and improve its performance, as grounds for selecting it as the research basis. Before diving into explaining our proposed solution, it is necessary to get acquainted with the structure and workflow of TONTA.

3.2 Fundamental Model

Network traffic can be measured and analyzed from various aspects by focusing on specific features of interest in the network traffic. However, the feature selection task is extensively depended on the type of intended NTMA application. Based on the ultimate goal of applying a specific NTMA technique (e.g., pattern recognition), one or more characteristics of network traffic, such as jitter, throughput, or delay, become highlighted.
One of the factors involved in choosing TONTA as the base for our proposed solution is its focus on the OWD variable in order to expose the changes in network traffic behavior. This technique, within a time series dataset consisting of end-to-end delay's data extracted on-the-fly from network traffic by the shallow packet inspection method, tries to detect change points with as little delay as possible. In addition to detecting the points of traffic state change, classifying the dominant trend in the form of ascending, descending, or horizontal types is another functional feature of TONTA. Moreover, other characteristics such as light weight, a simple and modular design, and good efficiency make this model a worthy choice for online traffic analysis of ad-hoc networks. However, in most of the

main steps of the TONTA procedure, there seems promising potential for developing and improving performance measures such as detection efficiency (percentage of False Positive, True Positive, and False Negative detections), time complexity, and speed.

3.2.1 Model Explanations

The overall work-flow of TONTA is shown by an activity diagram in Figure 3.1. The depicted diagram indicates the main tasks, elaborated as follows:

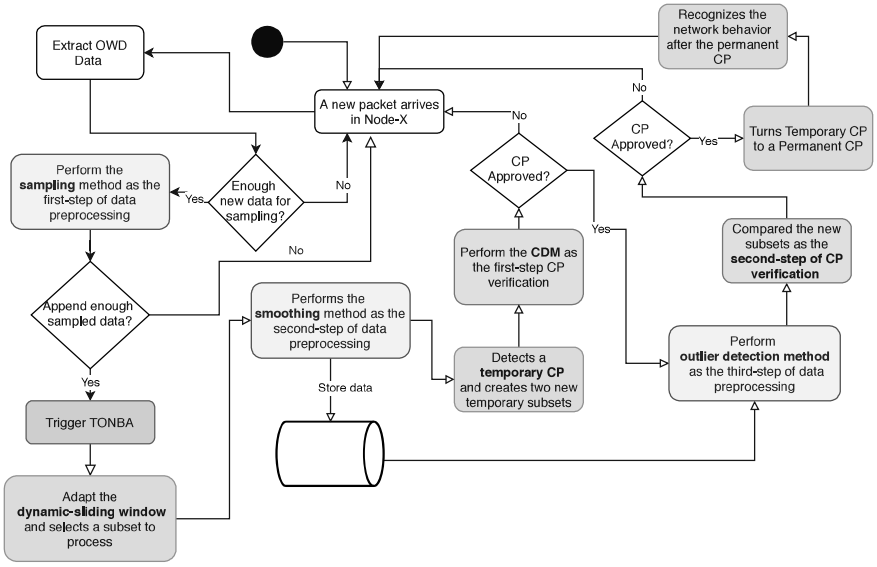


Figure 3.1: The activity diagram in TONTA (adopted from [3])

1. TONTA's procedure commence with sampling. The OWD raw data extracted by Shallow Packet Inspection (SPI) technique are digested through a sampling method in which each sampled data point(d) is calculated through the mean of a defined number(s) of raw data points (r) (Equation 3.1). The sampled data is added to the end of the time-series dataset. TONTA transforms every fifteen raw data points into one sampled data point (e.i., $s = 50$)

$$d = \frac{\sum_{f=1}^{s} r_f}{s} \tag{3.1}$$

   Before each trigger, TONTA waits for enough fresh sampled data points, so that the next stage is only reached if sufficient sampled data has been added to the time-series dataset *(Interval_Thr)*

2. To choose an appropriate amount of data points from the end of the time-series collection, TONTA adjusts the size of a dynamic sliding window.

3. In the second stage of data pre-processing, to enhance data quality and remove the effects of various packet generator distribution functions, the sampled date are smoothed.

4. TONTA splits the subset into two new temporary subsets termed ($Sec1$ and $Sec2$) after identifying a temporaryChange Point (CP)in each chosen subset using an unique CPD approach.

5. In order to verify the temporary CP, a special Curve Detection Method (CDM) is used in the initial stage. As demonstrated in Figure 3.2, curves are intermediate data points that translate two continuous dominant trends, but they do not represent a distinctive trend. The CDM approach aids in avoiding choosing any curve data point or repeating CPs as CPs.

6. As the final stage in the pre-processing of the data, an outlier identification approach is used to take outliers from the $Sec1$ and $Sec2$ subsets in order to make the subsets ready for comparison.

7. In order to assess the temporary CP's important level based on the strength of their trends, two outlier-free subsets are compared as the second phase of the verification process. The temporaryCPbecomes a permanentCPif the significance level of the temporaryCPexceeds a threshold (Imp T hr), in the absence of which TONTA reverts to the first stage and waits for fresh OWD data to be obtained from the network.

8. If the temporaryCPbecomes a permanent CP, the existing network traffic behaviour is used to gauge $Sec2$ intensity.

9. To prevent further processing, $Sec1$ and all preceding data points are eliminated from the dataset by choosing the permanent CP.

3.2.2 Technical Explanation of The Fundamental Model

In order to reveal the changes in network traffic behavior, TONTA divides the problem into two primary tasks to accomplish: First, to recognize the borders of a trend shift by detecting the change points, and second, to identify the dominant trend in the intervals between two consecutive change areas [3]. To this end, TONTA follows five main steps, which are described in the rest of this section:

Sampling Procedure

In general, the sampling method used in TONTA responds to the need to reduce redundant or similar OWD raw data appearing in subsequent triggers, as well as to avoid unnecessary processes that can significantly increase overall resource consumption. in this vein, TONTA attempts to optimize the amount of data that has to be processed in the initial stage of pre-processing. To increase the temporal complexity of the algorithm, the volume of OWD raw data is first reduced using a sampling approach. Where r represents OWD raw data and s is the fixed sample rate, sampling is performed using Equation 3.1. In addition, the *Interval_Thr* parameter is added to increase temporal complexity and prevent needless data processing. In order to collect enough data before processing, as seen in the Algorithm **??**, TONTA is triggered every *Interval_Thr* of d fresh sampled data points.
Setting the optimal size of *Interval_Thr* is one of the challenging tasks. It is subject to the trade-off between, on the one hand, decreasing the TONTA triggers by a large *Interval_Thr* and, on the other, adding more delay in identifying CPs as well as reducing accuracy by concealing the CPs in numerous new data points.

Dynamic-sliding Window

To determine CPs in an online manner, data points needs to be analyzed gradually on-the-fly. As explained in Step 2, TONTA, in each trigger, processes a subset selected from the time-series dataset by a dynamic-running window. The subset should not be very big to consume a lot of processing resources. In addition, it should not be very small resulting in reducing the accuracy. To achieve low time complexity and high accuracy, the dynamic-sliding window is proposed which runs through the time-series dataset from the end and select subsets in every trigger to process. The size of the window is highly dynamic, adapted before each trigger, based on different factors like last permanent CP, data density, and *Interval_Thr*.
The dynamic sliding window has two thresholds, *Max_Thr* and *Min_Thr*, which display the maximum and least number of data points that the window has picked, respectively. *Max_Thr* restricts the size of the window to increase time complexity and resource use. In order to determine the pattern of network traffic, TONTA requires a minimum quantity of data points. To offer sufficient data points for processing, *Min_Thr* is proposed. The dynamic-sliding window's pseudo-code, which is used to choose a subset of the time-series dataset, is listed in Algorithm **??**. The variable "j" is used to eliminate unnecessary data points when the sliding window contains more than *Max_Thr* data points. The size of the dynamic sliding window is maintained between *Min_Thr* and *Max_Thr* to reduce the time complexity. The default value of j is set to 1.5, and the window's subset consists of p data points.

Smoothing Procedure

A temporaryCPin the chosen subset should be chosen at this phase. After adding enough fresh sampled data points, the subset is smoothed to eliminate jitters and abrupt changes in trend. It also helps to convert datasets produced by various distribution function types into a dataset format that TONTA can handle, as described in step 3. Equation 3.2, where d_i is the i_{th} data point of the subset and m is a parameter to vary the size of the frame, is used to smooth data based on a floating frame with a frame size of $2 * m + 1$. The subset is smoother when m is greater, although extreme smoothing may mask the CPs.

$$X_i = median[d_{i-m}, d_{i-m+1}, \cdots, d_{i+m-1}, d_{i+m}] \qquad for\ i = 2, 3, \cdots, p-1 \tag{3.2}$$

Where $X_1 = d_1$ and $X_p = d_p$.
TONTA uses the first order difference of the smoothed data obtained from Equation 3.3, instead of the original subset to determine a temporaryCPfor each trigger.

The Steps To Locating A TemporaryCPIn The Sub-dataset

At this stage, TONTA deals with the task of locating a temporaryCPin the sub-dataset, as a simple linear regression problem. In this method, to speak roughly, a value is predicted for each data point using the standard deviation and then, this predicted value is compared with its actual value so that the observed error reflects the position of that data point in relation to the prevailing trend before it.

$$T_i = X_i - X_{i-1} \qquad for\ i = 2, 3, \cdots, p \tag{3.3}$$

Algorithm 1: Dynamic-sliding window generator

Input: Data point *d_point* extracted by SPI
Input: Interval threshold $Interval\_Thr$
Input: Sampling rate s
Input: Minimum window size Min_Thr
Input: Maximum window size Max_Thr

```
InputDS ← ∅;
RawData ← ∅;
while stream is being monitored do
    for i = 1 to Interval_Thr do
        RawData ← s number of new d_points extracted by SPI;
        sample ← Sampling(RawData);
        InputDS[i + size(InputDS)] ← sample;
    end
    if size(InputDS) < Min_Thr then
        continue;
    end
    if size(InputDS) > Max_Thr then
        InputDS ← InputDS[Interval_Thr × j : size(InputDS)];
    end
    if Min_Thr < size(InputDS) < Max_Thr then
        InputDS ← Smoothing(InputDS);
        CP_empirical ← TONTA(InputDS);
        if CP_empirical is a Permanent CP then
            InputDS[1 : CP_empirical] ← ∅;
        end
        else
            continue;
        end
    end
end
```

The most recent sampled data points are given priority using a unique exponential weighting mechanism. The weighting method is useful to spot the recent trend changes that are more important, as the current pattern of the network traffic. Each data point in the subset gains or loses weight depending on the weighting strategy. The weight assigned to each data point is determined using Equation 3.4.

$$\theta_{ij} = \alpha_j(1-\alpha_j)^{p-i} \qquad for\ i = 1, 2, 3, \cdots, p \tag{3.4}$$

To assign various weights to data points, Equation 3.5 calculates the value of α as $0 < \alpha_j < 1$. The most recent selected data points are more significant than older ones when α is getting close to 1. In contrast, all data points appear to have equal importance when α is close to 0. The smoothed data points must be weighted ranging from low to high, to determine the error rates in order to detect the temporary CP.

$$\alpha_j = \alpha_{j-1} + \frac{1}{\varepsilon} \qquad for\ j = 2, 3, \cdots, \varepsilon \qquad where \quad \alpha_1 = \frac{1}{\varepsilon} \tag{3.5}$$

ε indicates how many times the data points have been compared using various weights. In addition to using the weighting model, Equation 3.6 also calculates a finite second order difference.

$$\Delta_{ij} = \theta_{ij}(T_i - T_{i-1}) \qquad for \quad i = 1, 2, 3, \cdots, p \quad and \quad j = 1, 2, 3, \cdots, \varepsilon \tag{3.6}$$

The finite order difference effect is eliminated by Equation 3.7, which also determines the standard deviation error for each weighted data point and totals the error rates for each θ

$$e_{\theta_{ij}} = \sum_{i=1}^{p} (\Delta_{ij} - T_i)^2 \qquad for \quad i = 1, 2, 3, \cdots, p \quad and \quad j = 1, 2, 3, \cdots, \varepsilon \tag{3.7}$$

Each $e_{\theta_{ij}}$ that has the lowest prediction error is taken into account as the temporary CP's position in the subset. For instance, let say the minimum value of $e_{\theta_{ij}}$ is occurred when $\alpha_j = 0.81$ and $p = 500$ then, the position of related temporaryCPwill be as: $0.81{*}500 = 405$ i.e. the 405_{th} data point in the subset.
Several factors are considered when determining the optimal value for ε, including the volume of network traffic, the processing power of the node hosting TONTA, and the sensitivity of network traffic to find precise CPs. Notably, the subset is weighted from the end, and the finalCPis chosen if there are multiple dominant trends in the subset.

Curve Detection Method

In the context of theNTA, dealing with curves is subject to some complexities. Since each trend progressively gives way to the following one, a curve is formed by the data between these two trends. The data of a network traffic flow that connects two successive trends but is not itself a trend is referred to as the "curve."To better clarify the notion of the curve, a part of a flow diagram containing the trends and the curves between them is shown in Figure 3.2. Thick black lines represent the trends, and the curves between them are indicated by orange lines.
Ensuring that the identified temporaryCPdoes not belong to a subset of the curve data is one of the important steps in theCPverification task. For this purpose, a parameter called *Curve_Thr* has been introduced. Two requirements (C1 and C2) must be satisfied, as

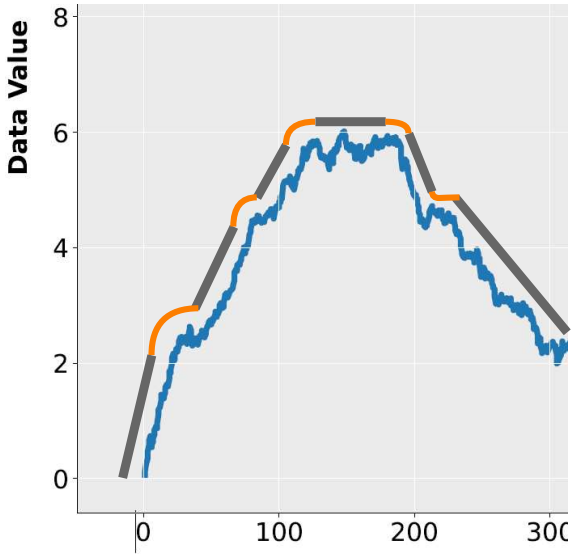


Figure 3.2: an example of curves and trends in a sub-section of the dataset

given below, in order to approve that the temporaryCPis not in a curve. $Pos(y)$ indicates the location of data point y in the time-series dataset, whereas $Len(x)$ gives the total number of data points for X. The latest detected permanentCPis displayed by CP_{last}. the temporaryCPdivides the dataset into two parts: the part before it (that is, from the beginning of the dataset to CP) is called *Sec1*, and the part after it (that is, fromCPto the end of the dataset) is $Sec2$.

- ***C1 :*** $\boldsymbol{(Len(}Sec1\boldsymbol{) > Curve\_Thr)\&\&(Len(}Sec2\boldsymbol{) > Curve\_Thr)}$
- ***C2 :*** $\boldsymbol{Pos(}CP_{Temporary}\boldsymbol{) > Pos(}CP_{last}\boldsymbol{)}$

Condition C1 provides a distinction between trends and curves based on their length, while condition C2 guarantees that the identified permanentCPwill be located after the last one.

Verifying The Temporary CP

After locating the temporary CP, TONTA must decide if it is important enough to become a permanentCPor to be rejected as a false alarm. Notably, the suggested technique always establishes a temporaryCPfor each trigger, but depending on how much the network traffic changes, thisCPmight be erroneous or negligible. As a result, *Sec1* and $Sec2$ must be contrasted to determine the significance of the change. Before comparison, since TONTA is extremely sensitive to outliers, *Sec1* and $Sec2$ must be free of any outlier datapoints. An outlier identification technique based on assigning scores to data points is suggested as a solution to the problem. In order to identify outliers and delete them, the allocated score is compared with a threshold. Equation (8) determines the slope of *Sec1* and $Sec2$ when $y = mx + b$, where m is the line's slope and b is its intercept. Assuming that the line's slope is known, the first order difference for x and y (i.e., x and y) can be used to show the intensity of fluctuations in *Sec1* and $Sec2$. Equation 3.9 is used to get the score for each data point based on detrending the subset. To identify whether each data point is

an outlier or not, Equation 3.10 is employed in the last step. Before comparing *Sec1* and *Sec*2, if the answer to Equation 3.10 for a data point is 1, it will be detected as outlier and will be deleted from the subset.

$$\text{Slope } = m = \frac{\sum_{i=1}^{x} (x_i - x)(y_i - y')}{\sum_{i=1}^{x} (x_i - x')^2} \tag{3.8}$$

$$Scores = |d - y| \tag{3.9}$$

$$f(x) = \begin{cases} 0 & \forall x \mid \text{Score}(x) \leq \text{Avg}(Scores) + \sqrt{\text{Var}(Scores)} \\ 1 & \forall x \mid \text{Score}(x) > \text{Avg}(Scores) + \sqrt{\text{Var}(Scores)} \end{cases} \tag{3.10}$$

$$C_{Sec1} = \frac{1}{\text{Len (Sec1)}} \sum_{i=2}^{\text{Pos}(CP_{\text{temporary}})} (X_i - X_{i-1}) \tag{3.11}$$

$$C_{Sec2} = \frac{1}{\text{Len (Sec2)}} \sum_{i=\text{Pos}(CP_{\text{temporary}+1})}^{p} (X_i - X_{i-1}) \tag{3.12}$$

Diff displays the magnitude of difference between *Sec1* and *Sec*2 subsets in terms of trend's slope. TheCPparameter, which can take one of the two values 0 or 1, is the TONTA module's outcome at the end of each trigger. The value 1 means converting the detectedCPinto a permanent CP, while the value 0 indicates that the discoveredCPhas not yet satisfied the requirements to be regarded as a permanent CP. Additionally, TONTA benefits from a parameter called Imp_Thr to determine the sensitivity to the difference between the trends of the two subsets, *Sec1* and *Sec*2. The parameter Imp_Thr is initiated as the sensitivity threshold of TONTA for implying aCPas the permanent CP, and is used to determine whether the detectedCPshould be accepted as a permanentCPor ignored. For example, $Imp\_Thr = 0.5$ can be interpreted as meaning that the trend difference on both sides of the temporaryCPmust be at least 50% in order to consider that point as a permanent CP. The notation C_{Sec2} represents the recent predominant behaviour of the time-series subset, which can be a negative or positive value, indicating a decrease or increase in packet flow's delay, respectively. Additionally, shifting the C_{Sec2} parameter toward zero demonstrates the flow delay's stability.

$$Diff = ABS\left(C_{Sec1} - C_{Sec2}\right) \tag{3.13}$$

$$CP = \begin{cases} 1 & \text{if: Diff } \geq ABS\left(C_{Sec1} * Imp\_Thr\right) \\ 0 & \text{if: Diff } < ABS\left(C_{Sec1} * Imp\_Thr\right) \end{cases} \tag{3.14}$$

After the position of the new permanent change point is specified, all data points before thisCPare removed from the data set since they will not be processed anymore. Besides, in the next trigger, by adding $Interval\_Thr$ new data points to the existing data, that is *Sec*2, the size of the new data set may exceed the Max_Thr threshold (i.e., $len(Sec2) + Interval\_Thr > Max\_Thr$). In such a case, to fit the data set's size to Max_Thr, some data points from the beginning of the subset will be removed.

3.3 Proposed Method

Although TONTA has shown acceptable efficiency in providing a lightweight and agile technique with appropriate accuracy, it still has the potential for improvement in some respects. In this vein, identifying the hindering factors and challenges that this technique is faced with is the first step in presenting any optimization. These challenges are mainly seen in the following aspects:

- *Time complexity challenges:* As mentioned in section 4.3 of the TONTA paper, the time complexity of their proposed method is $O(Max\_Thr+(Max\_Thr*\varepsilon)$.Although Max_Thr is the main variable to increase the time complexity of TONTA, it gets a variable that is under control. The second important variable is ε which has a fixed value equals to 100 in their proposed method. Considering the procedure of TONTA, ε has a high opportunity to be optimized resulting in reducing the time complexity of the proposed method considerably.

- *Accuracy challenges:* As the performance evaluation section of TONTA shows, it has high accuracy compared to the RuLSIF as a benchmark dataset, but still the number of False positive alarms and False negative alarms are high which is not negligible. In addition, considering that the method is online, the number of true alarms are not convince able

- *Latency challenges:* One of the main challenges of TONTA is the latency as the methods needs to wait for a while to discover the change point. Although by considering that the method is online, a reasonable latency can be understandable, in some cases, the latency is too high. As an example, considering that we have more than one stream that can affect each other negatively, recognizing the change point lately can ruin the performance of the system, especially for the other streams.

3.3.1 Our Improvements

To address the aforementioned challenges, our study has directed its attention towards the identified weaknesses within the TONTA's methodology. It is our belief that by eliminating these recognized vulnerabilities, a resolution to the challenges at hand may be achieved. Based on this, the upgrades that we have made in different parts of the base method can be classified as follows:

Accuracy Oriented Changes

In general, one of the main performance criteria of a CPD model is their accuracy, which can be measured by the number of FP, FN, and TA (TP) detections. In order to improve the accuracy of the proposed model, some adjustments were made in the sampling and smoothing procedures, which have led to a notable reduction in FP and FN diagnoses as well as a rise in detected True-Alarm (TA)s. The earliest processing steps in TONTA's workflow are sampling and smoothing, which are aimed at improving the overall performance of the model. These deterministic steps are designed to filter out raw data and reduce its volume, as well as remove any noise or unimportant data. In order to further improve the performance of the sampling and smoothing steps, some changes have been made to the proposed solution. When it comes to network environments, and particularly IoT networks,

raw data can be full of repetitive values, outliers, and minor change points. To address this issue, it is necessary to use sampling and smoothing methods to reduce the volume of raw data, remove duplicate and similar data, eliminate outliers, and focus on major change points. To investigate the effects of these methods on the system's performance, some changes have been made to the sampling and smoothing techniques. Since Tonta's method will be used as a cornerstone to compare and evaluate the performance of the proposed method, the sampling and smoothing techniques have been set up in a way that produces structurally similar datasets, allowing for replicating the same experiments.

1. **Sampling:** In terms of the sampling process, due to the high speed of data generation in the network, it is essential to apply appropriate filters to reduce the large volume of raw input data. The sampling process is a tradeoff between accuracy and time complexity. The sampling rate can be shown as $1/s$, where s is the number of raw data points taken in every sampling execution. Increasing the sampling rate means increasing the amount of raw data entered into the system, which puts more load on the system. Meanwhile, it results in concealing the real and major CPs among redundant minor or false CPs by raising False-Positive (FP) diagnoses. Conversely, a very small sampling rate causes the removal of a larger number of potentially important raw data in early pre-processing steps and raises the False-Negative (FN) detections.
   To address this issue, we have decided to replace the sampling method used in Tonta with a two-stage sampling process formulated in Equation (3.15). In the first round of sampling, the mean of every 10 raw data points is calculated, and in the second round, the mean of every five consecutive means calculated in the previous round would be passed to the system as an input. This method adds further variability to the input data to mimic the original pattern in the raw data more precisely in addition to to reduce the loss of important data points.

$$d = \frac{\sum_{i=1}^{n} \left(\frac{\sum_{j=1}^{s} r_j}{s} \right)_i}{n} \quad where \;\; s = 10 \;, \;\; and \;\; n = 5 \tag{3.15}$$

2. **Smoothing:** Smoothing is a statistical technique that is commonly used to remove noise and outliers from data. Its primary goal is to identify underlying patterns and trends in the data by reducing the influence of random fluctuations. Particularly, when data belongs to noisy and dynamic ecosystems, smoothing can be an effective way to remove outliers and help to identify main trends.
   Choosing the appropriate smoothing algorithm for a given dataset is crucial to the success of the technique. Smoothing algorithms vary widely including simple moving averages, kernel smoothing, and regression techniques, to name a few. The selection of the most suitable smoothing algorithm depends on the nature of the data and the research question being investigated. Common applications of smoothing include image processing, finance, and machine learning. For instance, in image processing, smoothing can help reduce the noise in an image while preserving its essential details. In finance, smoothing can be useful for identifying trends in stock prices and removing short-term fluctuations. In machine learning, smoothing can help to regularize models and prevent overfitting.
   In the context of IoT networks, packet streams are often exposed to high levels of noise and changes in the pattern of packet generation due to variations in parameters such as transmission rate and jitter. Therefore, smoothing can be an essential part of data

preparation, helping to reduce the impact of outliers, limit the effects of distribution function variation, and eliminate short-term fluctuations.
The smoothing process is another part of TONTA's workflow that has the enhancement capacity to improve the functionality of the model by increasing the true alarms (True-Positive (TP)s). In Tonta, a smoothing method based on the median of every three adjacent data points is used to remove jitters and reduce the effect of variation of packet-generating distribution functions. It also fades out the short-term fluctuations from data. (Equation 3.2). After examining different smoothing methods, we chose the Geometric Mean (GM) algorithm because of its ability to remove outliers caused by noise.
The geometric mean filter could be useful for analyzing the traffic's data in IoT ad-hoc networks prone to ambient noise. In comparison with other algorithms such as the median filter, the geometric mean filter can be a reasonable option considering the specific characteristics of the IoT ad-hoc network data. Although, the median filter is effective at removing impulse noise (sudden spikes in the data) , it may not be as effective at reducing random noise. GM process is formulated in Equation (3.16)

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n} \tag{3.16}$$

where : $\prod$ =the geometric mean, n = number of values, and x_i = values to average

Time Complexity Oriented Changes

The time complexity of TONTA is expressed as Equation 3.17, which shows its direct relationship with parameters ε and Max_Thr. Although, according to this equation, reducing the value of ε results in a decrease in the time complexity of the algorithm, the side effects of such a change on other performance metrics should be considered. As shown in Table 3.1, the effect of different parameters on time complexity and accuracy is not the same. For example, while the time complexity of the algorithm is mainly influenced by ε and Max_Thr, without significant effect by Min_Thr, the model's accuracy is affected equally by both ε and Min_Thr. Thus, along with enhancing the time complexity of the method, lowering the value of ε inevitably results in an overall decline in accuracy.

$$Time\ Complexity:\ O(Max\_Thr + (Max\_Thr * \varepsilon)) \tag{3.17}$$

Considering the equivalence of ε and Min_Thr in relation to accuracy, compensating the side effects of reducing ε on accuracy by increasing the value of Min_Thr is the idea that we have explored. Overall, By replacing ε with $(X * (\varepsilon))$ in Equation 3.17, the time complexity can be formulated as:

$$O(Max\_Thr + Max\_Thr * (X * \varepsilon)),$$

which causes a significant reduction in complexity if X is a percentage less than 100. Taking the default scenario as the base in which $\varepsilon = 100$ and $Min\_Thr = 100$, as much as the X factor decreases the ε's value, it adds to Min_Thr concurrently. We examined different values of X to come up with the balance point of ε and Min_Thr, among which two cases $X = 50\%$ and $X = 75\%$ caught our attention. Thus, by applying X to the

default settings, we will have:

$$\begin{cases} \text{for } X = 50\% : \quad \varepsilon = 50, \quad and \quad M = 150 \\ \\ \text{for } X = 75\% : \quad \varepsilon = 75, \quad and \quad M = 125 \end{cases}$$

Although taking $X = 50\%$ yields more reduction compared to $X = 75\%$, evaluating the results in terms of accuracy leads us to prefer the latter case. The graph for each of these two cases is shown in Figures 4.21 and 4.20 respectively.

Parameter	Accuracy	Time complexity
ϵ	Strong	Strong
Max_Thr	Very Low	Strong
Min_Thr	Strong	Low
Imp_Thr	Strong	Very Low
Interval_Thr	Strong	Low

Table 3.1: The key parameters and how they effect accuracy and time complexity

Latency Oriented Changes

To address the issue of delayed detection of TAs after each CP, we devised a modification to TONTA's algorithm. This improvement enables the system to reassess the potential CPs against the subsequent close datapoints using flexible parameters. As is observed in the base method, large "$Interval\_Thr$"s can increase average delay ofCPdiscovery, but it can also reduce the number of system triggers. The delay is calculated based on the time of identifying aCPafter occurring. Furthermore, a large $Interval\_Thr$ may decrease the accuracy of TONTA, because in some cases, CPs will be overwhelmed by the redundant new data points. In addition, Table 3.1 indicates that this parameter has a high impact on the delay of findingCPafter happening. based on this, we have introduced a new algorithm to change $Interval\_Thr$ dynamically rather than having a fix value. As shown in Algorithm 2 the value of the parameter is flexible and in each trigger, it should be re-calculated. Although there is a default value for that, based on the possibility of recognizing a newCPin the current sliding window, the value will be changed to give more chance to the system to recognize a CP. The possibility of recognizing a newCPin the current window depends on the lenght of $Min\_Thr – size(C_{Sec2})$ which reflects the distance of the point in question from a real CP. If the proximity is too near, then there will be a new opportunity that system re-evaluate the datapoint once additional data are received. To avoid getting stuck in small intervals, the value of the parameter is increased exponentially till reaching the default value of $Interval\_Thr$ or finding a CP. Algorithm 2 shows the method of calculating the value of the parameter for each round.

Algorithm 2: A function which Calculates $Interval\_Thr$ dynamically

Input: Pre_Calc
Input: $CP\_recognized$
Input: $Interval\_Thr\_Default = 50$
Input: Min_Thr
Input: C_{Sec2}
Input: Imp_Thr
Input: Dif

```
1  X ← 0;
2  Y ← 0;
3  Z ← 0;
4  if CP_recognized then
5  |  return Min_Thr − size(C_Sec2);
6  end
7  else
8  |  if Pre_Calc ≥ Interval_Thr_Default then
9  |  |  X ←|Imp_Thr − Dif|;
10 |  |  if X < 0.03 then
11 |  |  |  Y ← 3;
12 |  |  end
13 |  |  else
14 |  |  |  Y ←Interval_Thr_Default ×X;
15 |  |  end
16 |  |  return int(Y × ln(Y));
17 |  end
18 |  else
19 |  |  Z ←Pre_Calc ×ln(Pre_Calc);
20 |  |  return int(Z);
21 |  end
22 end
```

Chapter 4

Evaluation

To evaluate the performance of the improve version of TONTA, same scenarios are evaluated that helps to understand the improvements better. In this case, scenarios that are evaluated in the TONTA paper are exactly applied again with the new proposed method and the results are compared to show the effectiveness of the the changes.

4.1 Dataset

In order to evaluate the proposed method, the same dataset is used to show the advantages of the proposed method. As mentioned in the TONTA paper, the dataset includes various scenarios that network behaviors can follow. To clarify, the packet generator's distribution functions performed at the source node for a period of 1000 sec, as they are used by TONTA, are provided in Table 4.1. During this period, 99,800 packets have been generated as raw data, which, by performing sampling method based on the rate 50 (i.e., $s = 50$), has resulted in 1914 sampled data points.

Start time	Distribution function	Parameter (s)	Traffic behavior	
			(Trend)	(Intensity)
0	Exponential	0.00980392	Uptrend	High
100	Exponential	0.00990099	Side-Way	Medium
200	Normal	0.01030927 0.001030927	Downtrend	High
300	Constant	0.01	Side-Way	High
400	Poisson	0.0095238	Uptrend	Very High
500	Exponential	0.01010101	Downtrend	Low
600	Exponential	0.00970873	Uptrend	Medium
700	Normal	0.0095238 0.00095238	Uptrend	Medium
800	Constant	0.01	Side-Way	High
900	Poisson	0.01010101	Side-Way	Low

Table 4.1: Packet generators

4.2 Evaluation Policies and Rules

To set the policies and rules for interpreting and evaluating the results of the proposed method, we follow the principles adopted in TONTA. These principles stem from the inherent traits of the network environment. For an online CPD method, the ideal performance would be the immediate recognition, at the destination node, of emerging CPs and the new data trend as soon as a change happens in the distribution function of the packet generation process at the source node. However, this is often impractical due to the effect of curves as well as delay in the intermediate nodes between the source and destination nodes (see Section 2.3.2). Therefore, in evaluating model's results, we need to determine a time interval around the real change points in which the detected CPs would be taken into account as true positive detection. It also enables the model to recognize the new trend of data based on adequate data points after the CP. Based on this, since the input data is simulated in such a way to show a change in the distribution function of packet generating every 100 seconds, a time range of 30 seconds before and after each real change point has been considered as the acceptable distance for calculating the recognized CPs as true detection. The value of 30 seconds for the time range has been adopted in accordance with TONTA to synchronize the results of the two compared methods.

4.3 Evaluation Scenarios

By generating the same dataset used in TONTA, we had the opportunity to replicate the conducted experiments of that approach as a base to compare the performance of the proposed method. In this framework, the results obtained by two methods in each of these scenarios are discussed in the rest of this chapter. Different scenarios are considered, mainly to evaluate the parameters of the proposed method. In this case, a default scenario is used as same as TONA, to use as benchmark and develop other scenarios by changing the values of parameters. All parameter settings of the investigated scenarios are listed in Table 4.2 along with the links to corresponding graphs that exhibit the obtained results using these parameter values.

4.3.1 Default Scenario

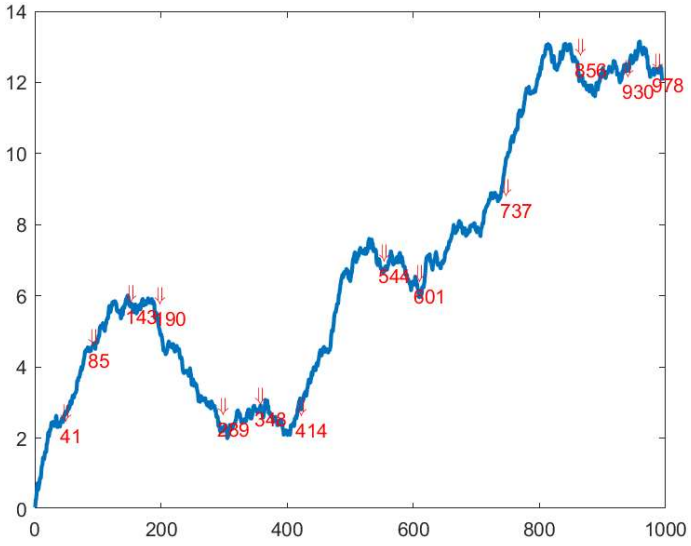
Figure 4.1 illustrates the comparison between TONTA and μTonta in terms of the obtained results for default settings where, *(Interval_Thr=50, Min_Thr=100, Max_Thr=300, Imp_Thr=0.5, ε=100)*. To more clarify, the results obtained by both methods on these settings are categorized in Table 4.3. As can be seen in two graphs, despite the displacement of most of the CPs compared to the TONTA method, the positions of the CPs are still within proximity of their corresponding spots in TONTA. Overall, cutting down on false alarms (FPs) is one of μTonta's key benefits, which is visible in the mentioned graph, where there are no false alarms between points 300 and 400 in addition to eliminating three false alarms at points 143, 544, and 856. Considering the lower false alarms detected by TONTA as one of it's main advantages over RuLSIF, used as the benchmark model, our proposed method has further enhanced this advantage by reducing the number of FP detections even more. Although the distance between the recognized change points by μTonta and the real time of the changes is a little bit higher than TONTA, the number of true alarms has increased, meanwhile, false alarms shows a significant reduction.

Table 4.2: Detailed settings of investigated scenarios
(to redirect to a figure click on its number)

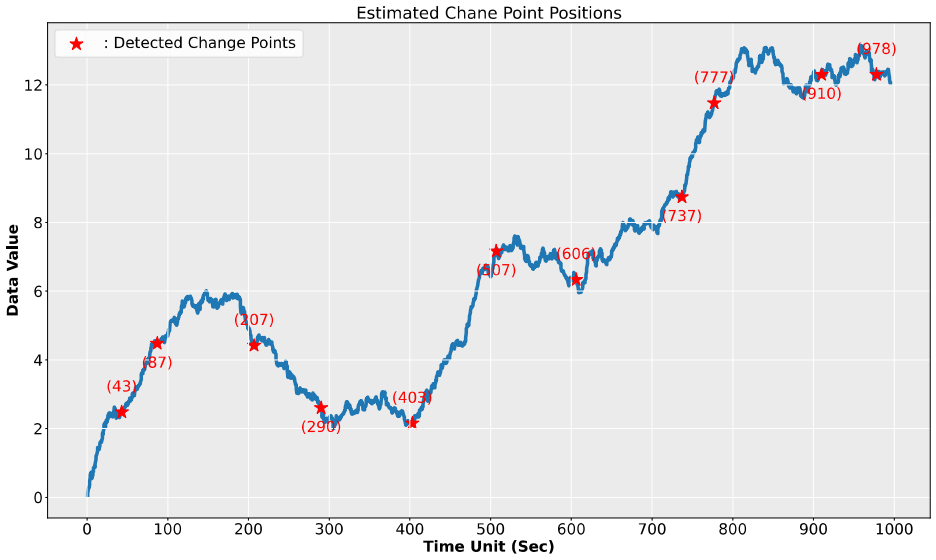
Scenario	Interval_Thr	Min_Thr	Max_Thr	Imp_Thr	ε	Figure
1	50	100	300	0.5	25	4.11
2	50	100	300	0.5	50	4.12
3	50	100	300	0.5	75	4.13
4	50	100	150	0.5	100	4.14
5	50	100	600	0.5	100	4.15
6	50	300	600	0.5	100	4.16
7	50	100	300	0.2	100	4.17
8	50	100	300	0.8	100	4.18
9	50	100	300	1.0	100	4.19
10 (default)	50	100	300	0.5	100	4.1
11	10	100	300	0.5	100	4.2
12	20	100	300	0.5	100	4.3
13	30	100	300	0.5	100	4.4
14	40	100	300	0.5	100	4.5
15	60	100	300	0.5	100	4.6
16	70	100	300	0.5	100	4.7
17	80	100	300	0.5	100	4.8
18	90	100	300	0.5	100	4.9
19	100	100	300	0.5	100	4.10
20	50	125	300	0.5	75	4.20
21	50	150	300	0.5	50	4.21

	Total CPs	FP	FN	True Alarms (TP)
µTonta	11	2	1	9
TONTA	13	6	3	7

Table 4.3: The comparison of results on the default scenario, in μTonta and TONTA



(a)



(b)

Figure 4.1: the obtained results in default settings, *(a)* by TONTA, and *(b)* by μTonta

4.3.2 Evaluating The Effects of *Interval_Thr*

Table 4.4 contains the results obtained by μTonta for different values of *Interval_Thr*. Further, the comparative data extracted from Table 4.4 and it's equivalent in TONTA are collected in Table 4.5. These data are illustrated, row by row, in the following Figures from 4.2 to 4.10, and 4.1 (that represents the outcome in default setting). As shown in Table 4.5, μTonta has always recognized lower number of CPs. Although in case of lower *Interval_Thr*, this can be considered as an advantage, in high values of *Interval_Thr*, this causes inefficiencies as the number of True alarms reduced significantly; However, the number of False positive alarms by μTonta shows a significant reduction that helps to reduce triggering the whole method falsely. In overall, μTonta has detected more true alarms than TONTA, so that it demonstrates a 25% improvement on average, with only a few exceptions. Another notable difference is that while TONTA has the best results for true alarms in the case of *Interval_Thr*=10, but the best results of μTonta belong to *Interval_Thr*= 40 and 50. In this case, the time complexity of μTonta is better than TONTA as lower *Interval_Thr* causes very higher time complexity. In case of false positive alarms, μTonta has better results in most cases that help to reduce triggering the whole method falsely.

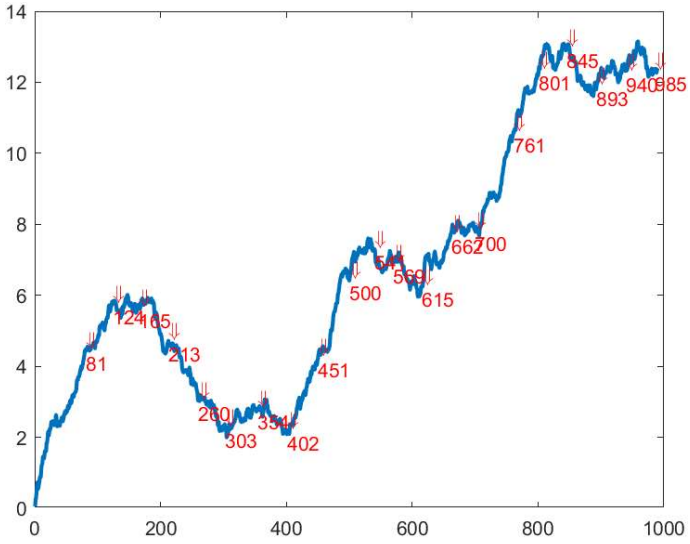
Interval_Thr	Total detected CPs	Position of CPs	False positive	False negative	True positive (true alarms)
10	16	94, 131, 203, 255, 313, 350, 398, 446, 538, 609, 660, 704, 803, 843, 941, 992	8	2	8
20	12	94, 157, 212, 303, 331, 457, 506, 580, 637, 705, 909, 960	5	3	7
30	12	103, 157, 201, 294, 405, 453, 512, 565, 700, 791, 900, 969	4	2	8
40	18	39, 101, 143, 159, 200, 302, 363, 402, 504, 560, 588, 645, 733, 808, 866, 903, 955, 981	9	1	9
50	11	43, 87, 207, 290, 403, 507, 606, 737, 777, 910, 978	2	1	9
60	13	133, 210, 254, 320, 354, 410, 453, 517, 620, 688, 747, 815, 935	6	3	7
70	7	104, 160, 315, 408, 613, 708, 835	2	5	5
80	9	102, 301, 411, 449, 513, 593, 704, 867, 906	2	3	7
90	7	103, 202, 300, 702, 867, 938, 991	2	5	5
100	9	105, 193, 308, 394, 565, 636, 938, 960, 995	4	5	5

Table 4.4: The effects of changing the *Interval_Thr* on the results of μTonta.

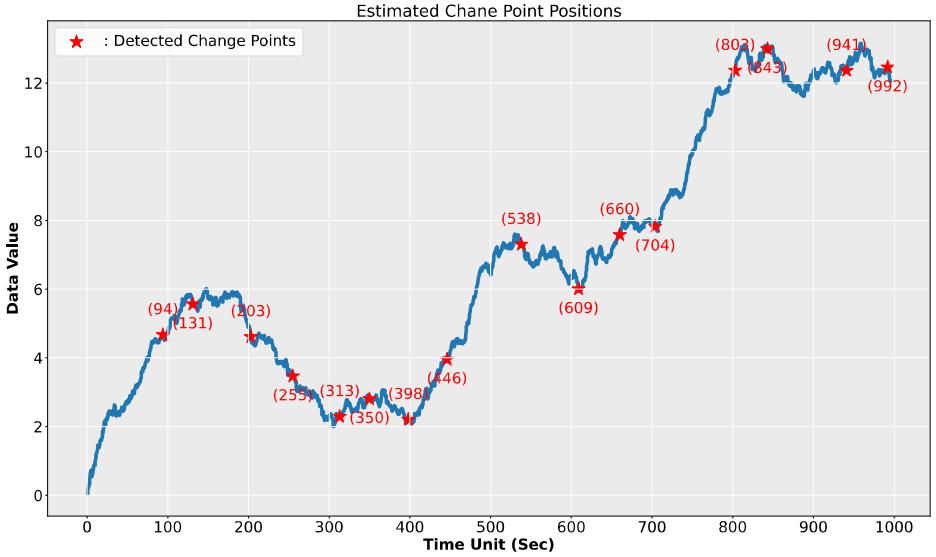
Interval_Thr	Total detected CPs		False positive		False negative		True alarms	
	µTonta	TONTA	µTonta	TONTA	µTonta	TONTA	µTonta	TONTA
10	16	21	8	11	2	0	8	10
20	12	18	5	9	3	1	7	9
30	12	15	4	7	2	2	8	8
40	18	13	9	7	1	4	9	6
50	11	13	2	6	1	3	9	7
60	13	10	6	6	3	5	7	4
70	7	7	2	3	5	6	5	4
80	9	6	2	1	3	5	7	5
90	7	5	2	1	5	6	5	4
100	9	5	4	2	5	7	5	3

Table 4.5: Comparing the results obtained from μTonta with the same from TONTA in terms of *Interval_Thr* affection.

Figure 4.2 illustrates the results by two methods, where *Interval_Thr* in the default scenario has been changed to 10. Comparing the two graphs indicates that while TONTA has reached the maximum number of true alarms, our method has better performance in terms of false alarms. As can be seen, in this case, µTonta has been able to create a 27% improvement in terms of FPs, but at the same time, the number of TAs detected by it also shows a 20% decrease.



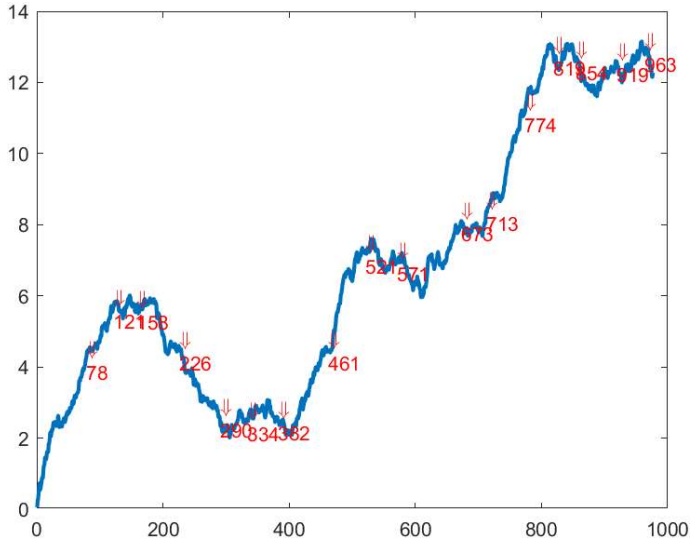
(a)



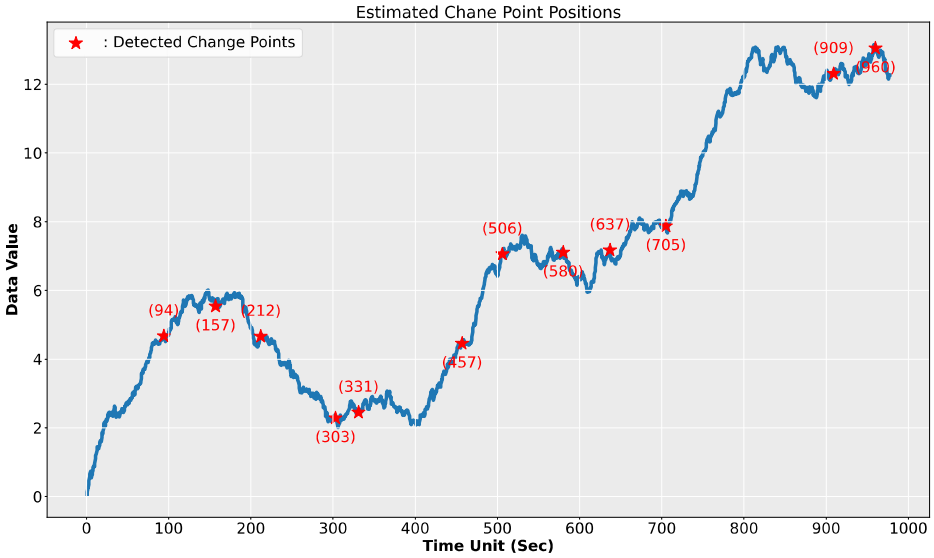
(b)

Figure 4.2: the obtained results for *(Interval_Thr=10, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

Figure 4.3 illustrates the results by two methods, where *Interval_Thr* in the default scenario has been changed to 20. Comparing the two graphs indicates that TONTA has still better performance in terms of TAs, while the proposed method has performed significantly in reducing FPs. As can be seen, in this case, µTonta has achieved a 44% improvement in terms of FPs, but at the same time, a 22% decline has occurred as well.



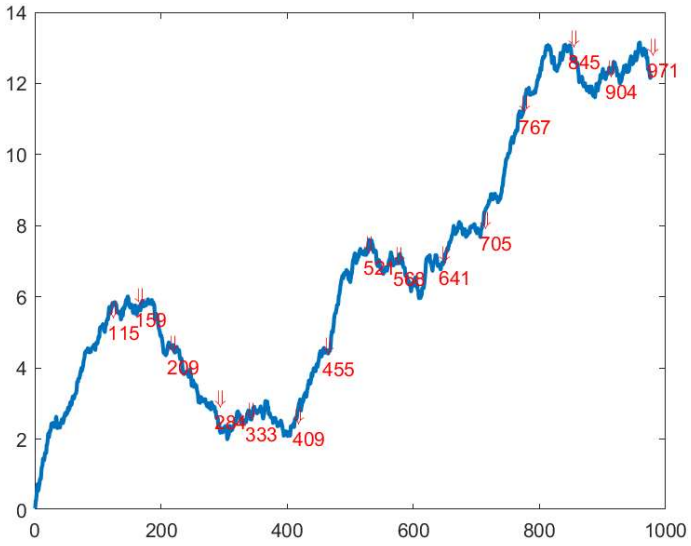
(a)



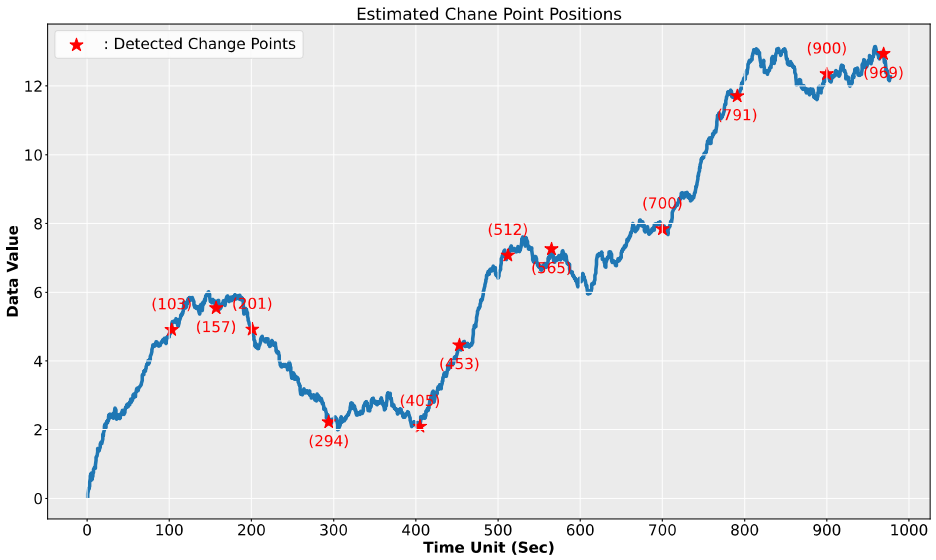
(b)

Figure 4.3: the obtained results for *(Interval_Thr=20, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

The graphs in Figure 4.4 demonstrate the results by two methods, where *Interval_Thr* in the default scenario has been changed to 30. As can be seen, in this case, the improvement in FPs by μTonta is considerable, while its performance about TAs is also as good as TONTA where both have detected 8 TAs.



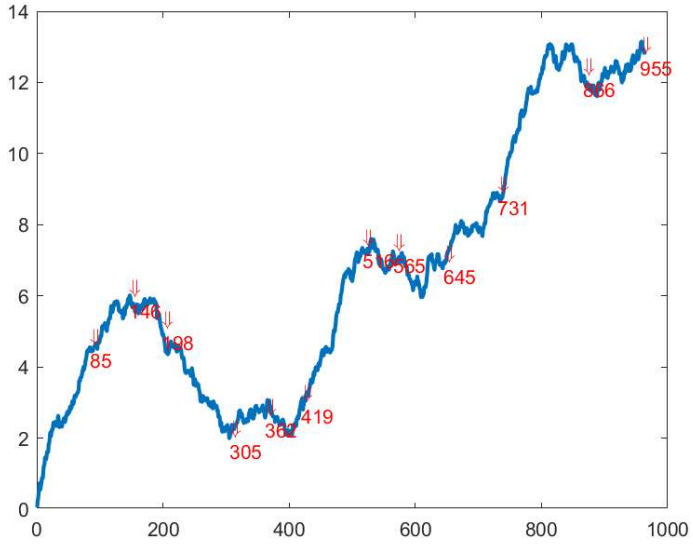
(a)



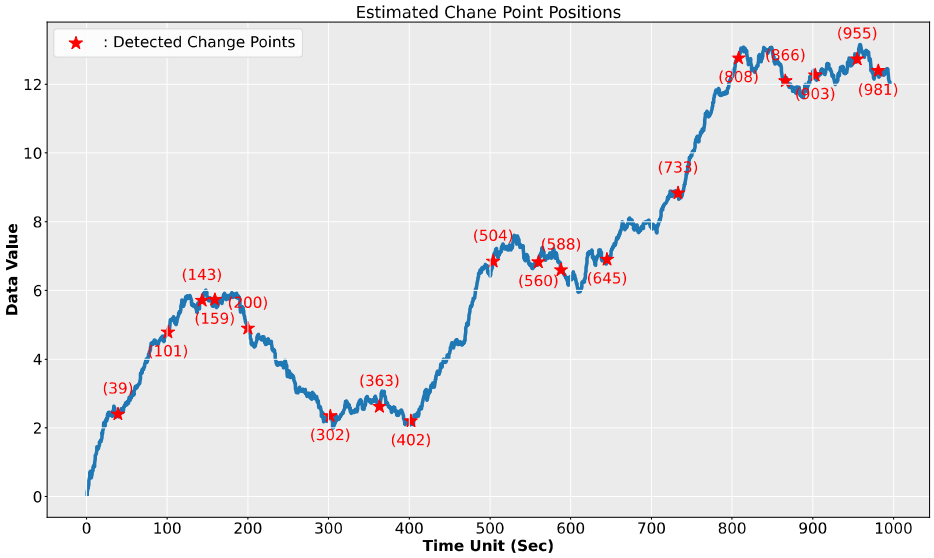
(b)

Figure 4.4: the obtained results for *(Interval_Thr=30, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

The obtained results by the two methods are shown in graphs of Figure 4.5, where *Interval_Thr* in the default scenario has been changed to 40. As can be seen, in this case, μTonta has achieved its best performance across all cases, in terms of TAs, by a 50% increase in relation to TONTA. However, the number of FPs announced by μTonta is 2 more than that of TONTA, which is considered as a 20% decline in this aspect.



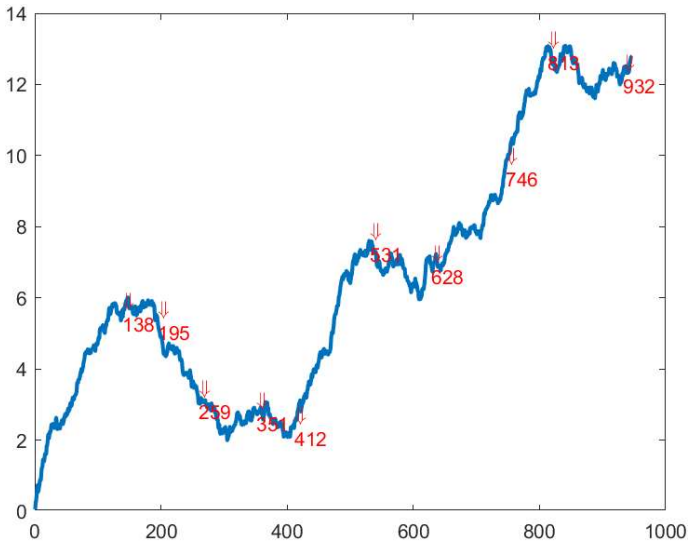
(a)



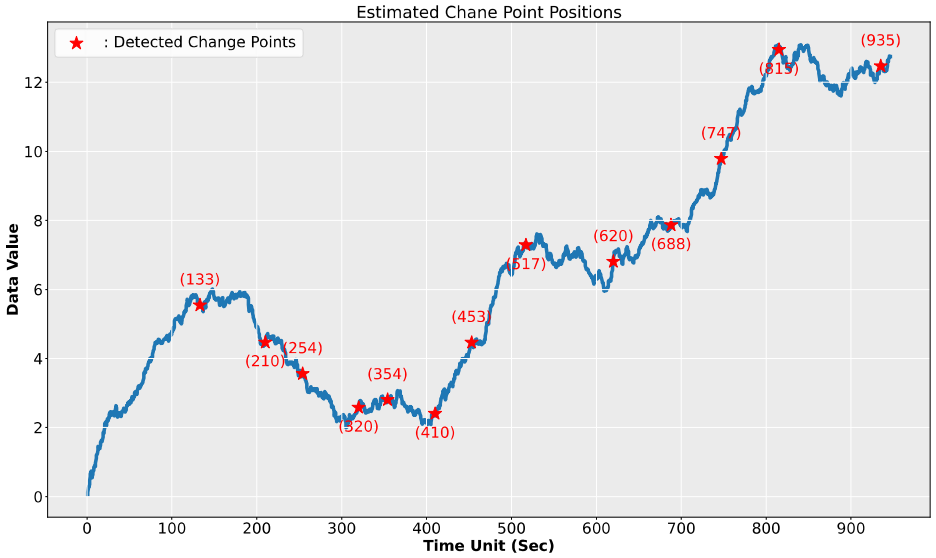
(b)

Figure 4.5: the obtained results for *(Interval_Thr=40, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

Figure 4.6 illustrates the results by two methods, where *Interval_Thr* in the default scenario has been changed to 60. Comparing the two graphs shows an equal outcome on FPs, whereas μTonta has managed to create a 75% improvement in TAs, in this case.



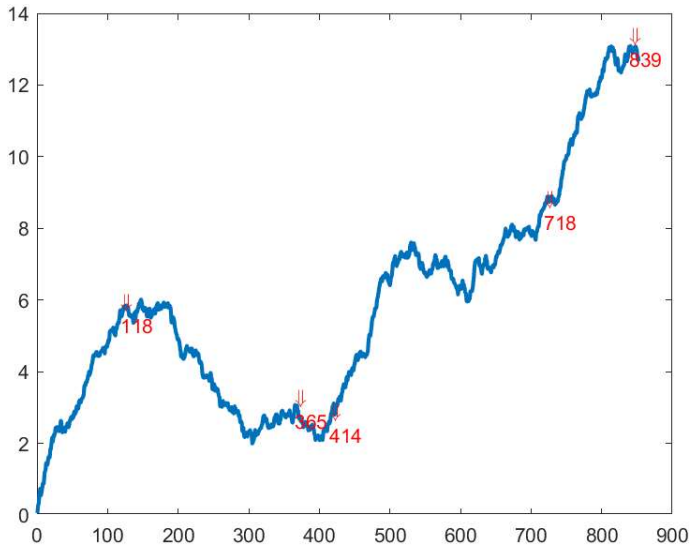
(a)



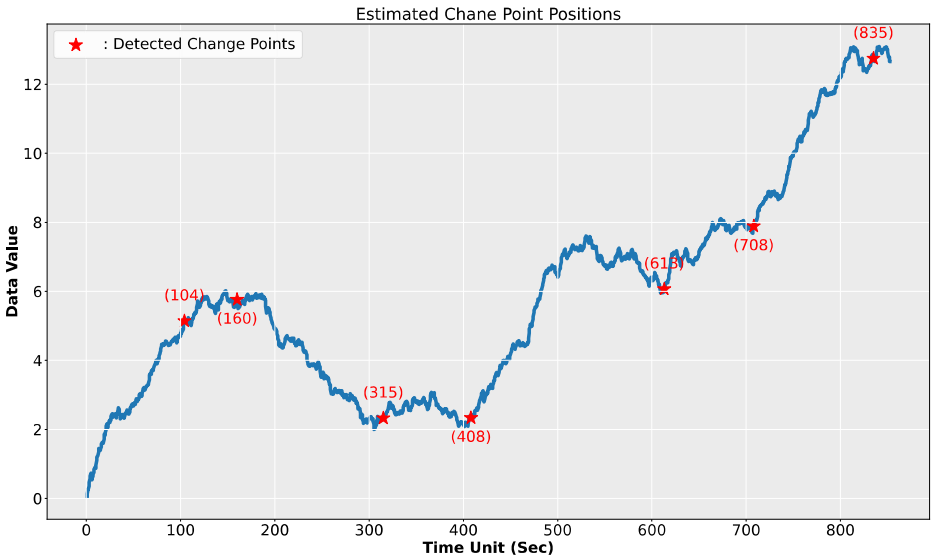
(b)

Figure 4.6: the obtained results for *(Interval_Thr=60, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

The obtained results by the two methods are shown in graphs of Figure 4.7 where *Interval_Thr* in the default scenario, has been changed to 70. Overall, the illustrated graphs indicate weak performance by both methods in this case. However, μTonta has outperformed TONTA slightly at this settings.



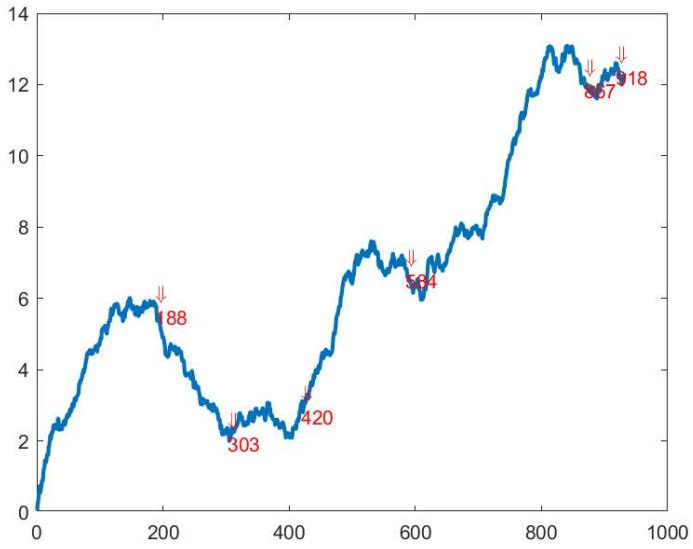
(a)



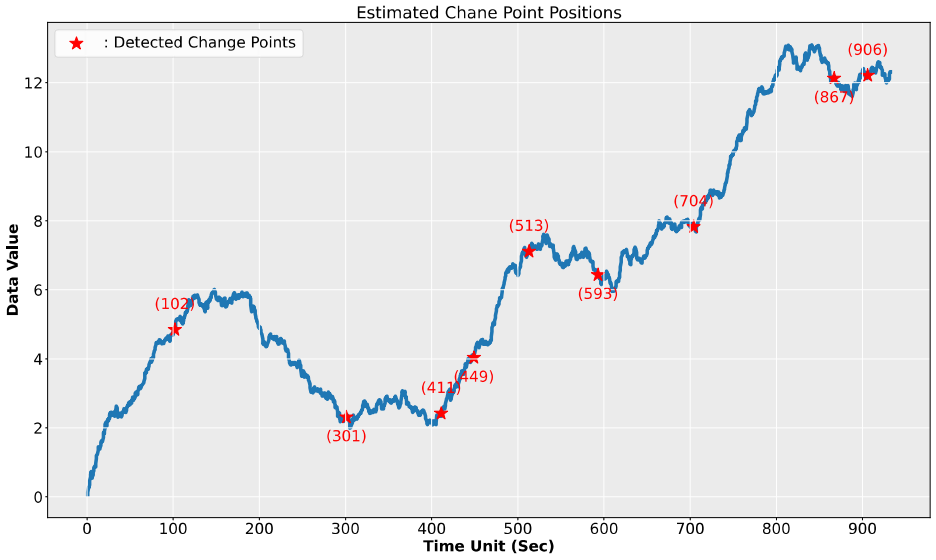
(b)

Figure 4.7: the obtained results for *(Interval_Thr=70, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

The graphs illustrated in Figure 4.8 represent the outcomes by the two methods, where *Interval_Thr* in the default scenario has been changed to 80. In this scenario, TONTA has delivered its best performance across all cases in terms of FPs. Nonetheless, its overall performance has been degraded due to the limited number of detected TAs. As can be seen, μTonta has managed to improve the TAs detection with a negligible increase in FPs.



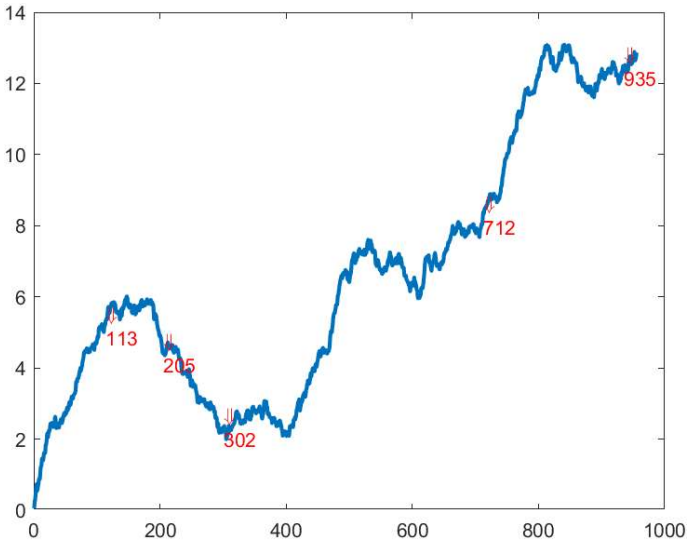
(a)



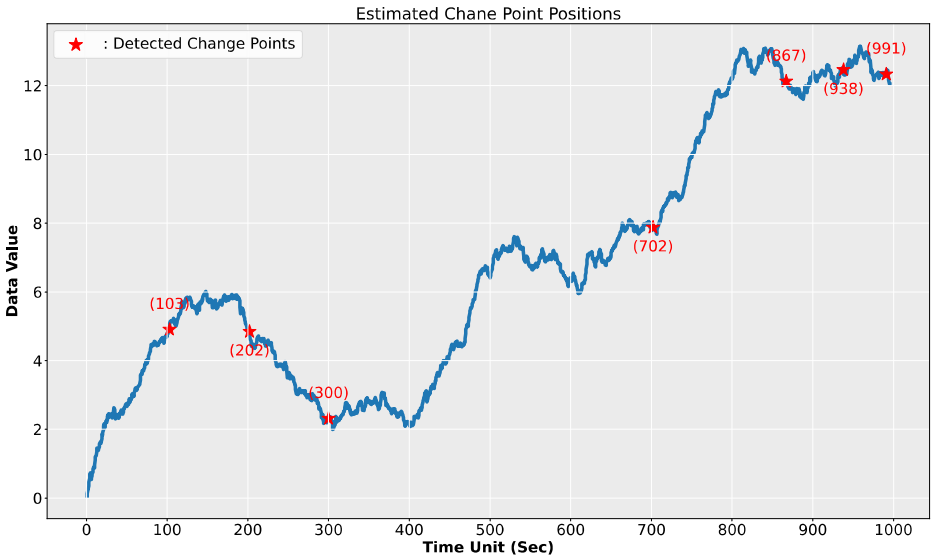
(b)

Figure 4.8: the obtained results for *(Interval_Thr=80, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

Figure 4.9 shows the graphs representing the obtained results by the two methods, where *Interval_Thr* in the default scenario has been changed to 90. As can be observed, the performance of both methods is still close to that of the previous scenario with a few detected TAs.



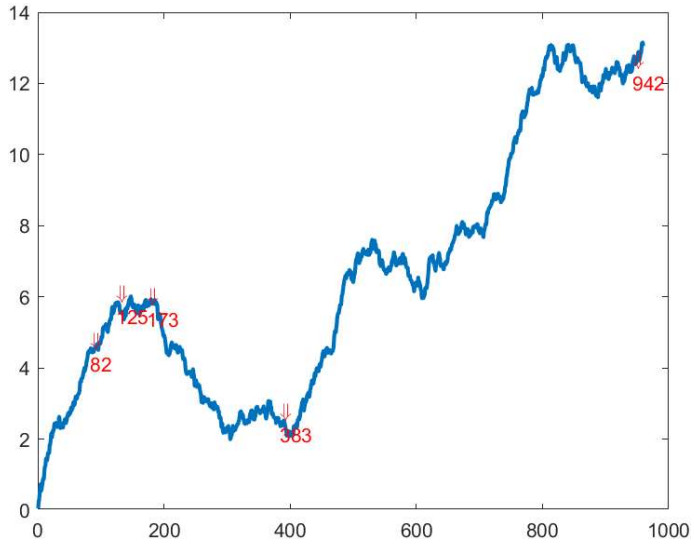
(a)



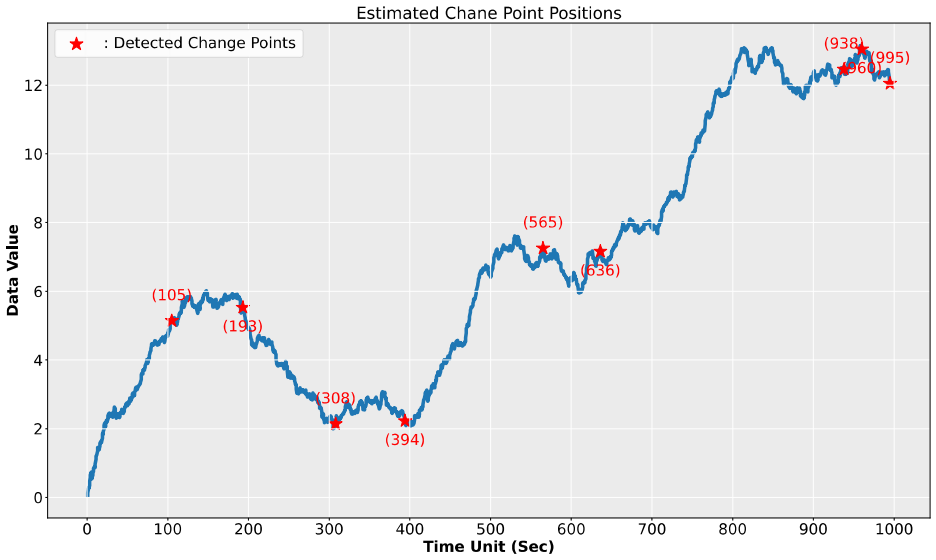
(b)

Figure 4.9: the obtained results for *(Interval_Thr=90, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100)*, *(a)* by TONTA, and *(b)* by μTonta

Figure 4.10 illustrates the results by two methods, where *Interval_Thr* in the default scenario has been changed to 100, which is the highest value of that parameter examined. Like the three previous cases, in this scenario as well, both methods have had their worst performance in terms of the total number of detections and TAs.



(a)



(b)

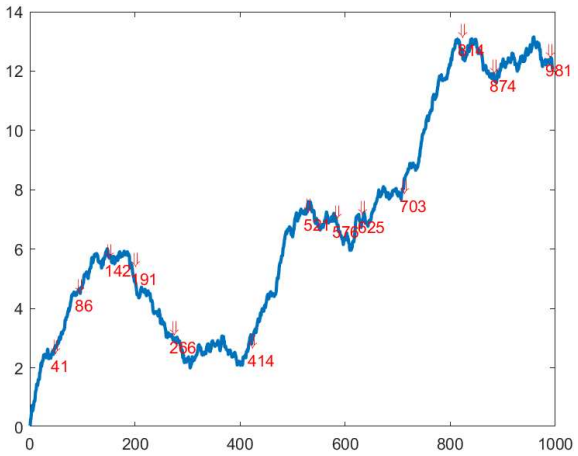
Figure 4.10: the obtained results for *(Interval_Thr=100, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

4.3.3 Evaluating the effects of $\mathcal{E}$

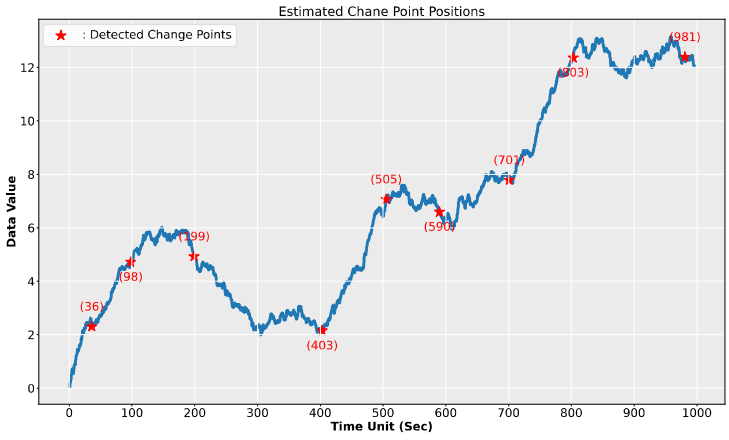
Comparing the impact of changing the ε in μTonta and TONTA are shown in Table 4.6, extracted from Figures 4.11, 4.12, and 4.13. As can be seen, in terms of the number of FPs, μTonta has performed better, while in terms of the number of true alarms and FNs, TONTA still has the advantage. In this regard, relatively speaking, the μTonta model has shown its best performance for $\varepsilon = 50$, where FPs have been halved without degrading in the other aspects. This information are useful to the phase of hyper-parameter tuning.

ε	False positive		False negative		True alarms	
	µTonta	TONTA	µTonta	TONTA	µTonta	TONTA
25	3	3	4	3	6	7
50	3	6	1	1	9	9
75	2	7	2	1	8	9

Table 4.6: Comparing the effects of changing the ε in μTonta and TONTA.

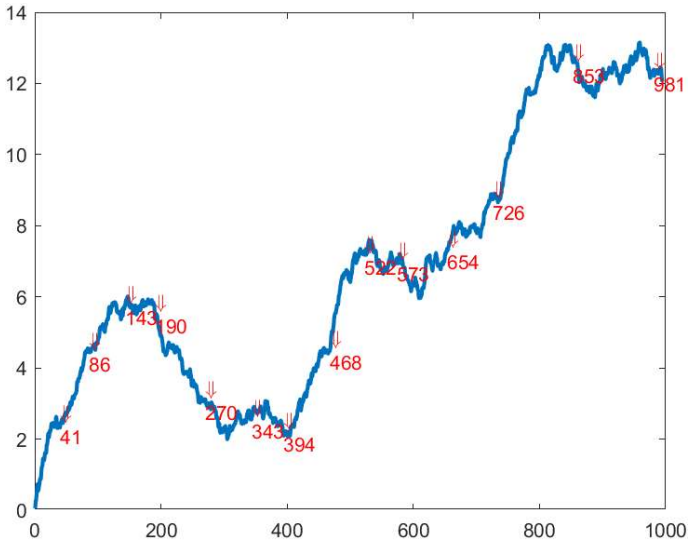


(a)

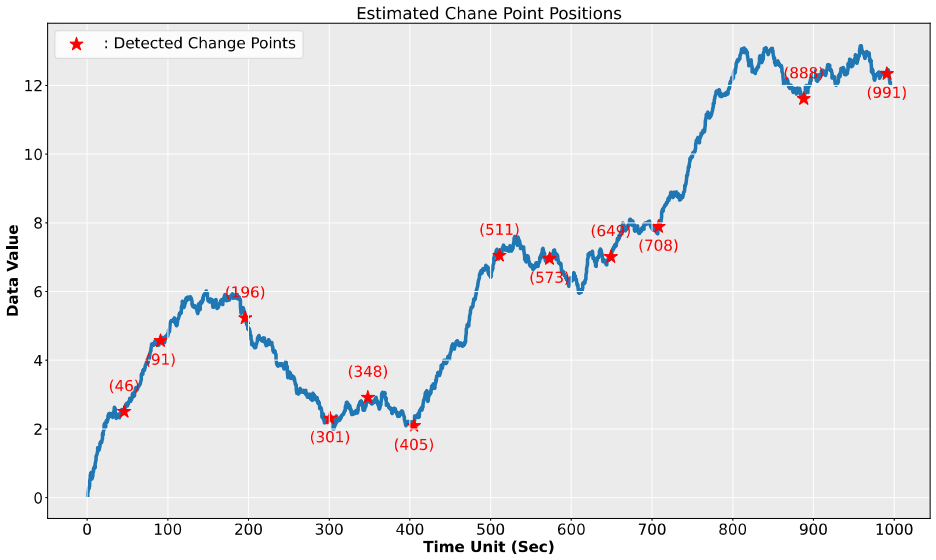


(b)

Figure 4.11: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=300, Imp_Thr=0.5, ε=25)*, *(a)* by TONTA, and *(b)* by μTonta

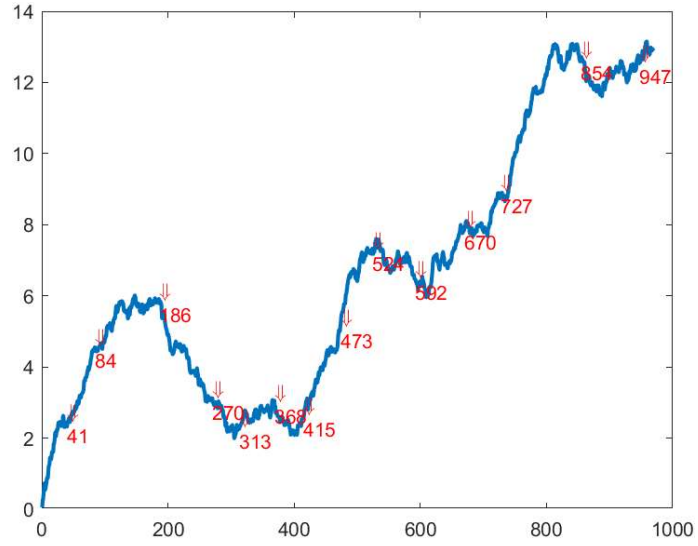


(a)

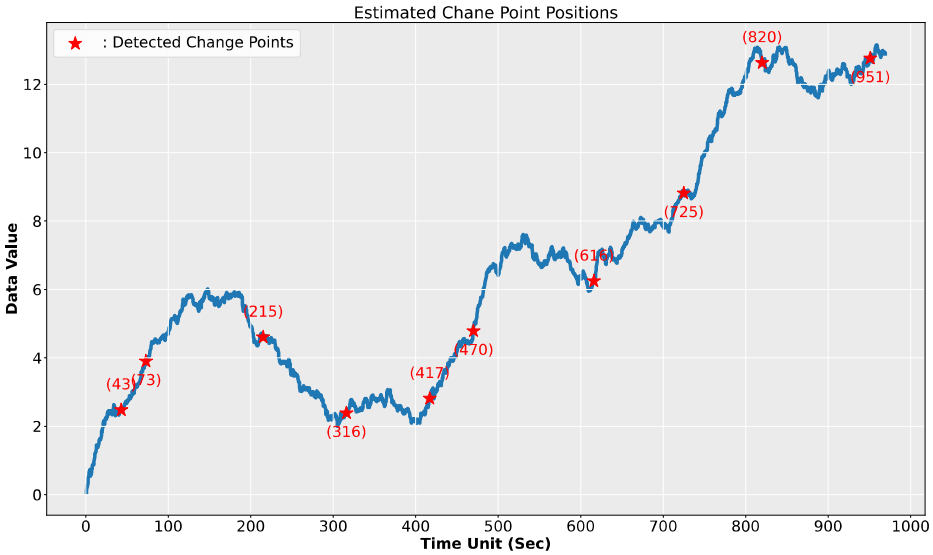


(b)

Figure 4.12: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=50), (a)* by TONTA, and *(b)* by μTonta



(a)



(b)

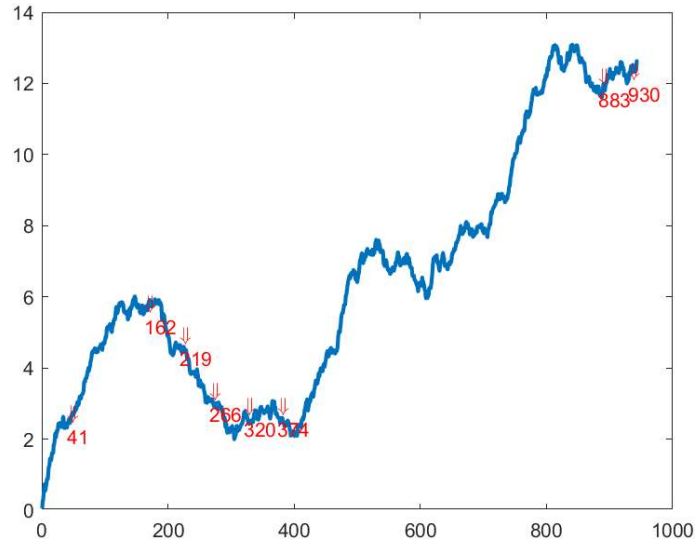
Figure 4.13: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=300 Imp_Thr=0.5, ε=75), (a)* by TONTA, and *(b)* by μTonta

4.3.4 Evaluating The Effects of *Min_Thr, and Max_Thr*

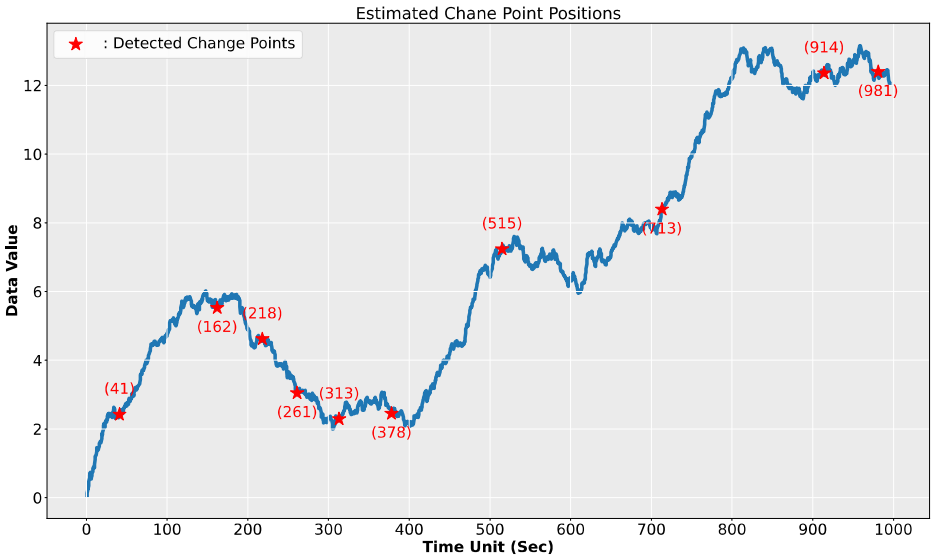
The results of evaluating μTonta and TONTA in terms of two parameters Min_Thr and Max_Thr is shown in Table 4.7 extracted from Figures 4.14, 4.15, and 4.16. In the first scenario shown by 4.14 we have evaluated the values of parameters that are pretty close to each other, which led to the small size of the window. As shown in the figure, for $Min\_Thr = 100$ and $Max\_Thr = 150$, the results are so disappointing, where TONTA could not detect a lot of true alarms and the number of false alarms is so high. Although μTonta detects more true alarms and the number of false alarms is lower than TONTA, still the results are no satisfactory which shows that the values of the parameters close to each other can negatively affect the performance of the proposed method. In the second scenario (Figure 4.15) the values of the parameters are far from each other and the size of the windows is so big in the highest case as $Min\_Thr = 100$ and $Max\_Thr = 600$. In this scenario, the results of TONTA is more satisfactory than the previous scenario, but also the results of the μTonta improved significantly where it was able to detect nine positive alarms with a negligible rise in FPs, marking it as one of its best results in terms of true alarms. It seems that the high distance between the values of the parameters can help the proposed method detect more true alarms, given a more flexible scope of data available to compare and more effectively discover extreme points. Although, in the case of false alarms, the results are still challenging. In the last scenario to evaluate the parameters, we consider high values for the parameters in addition to a high minimum size of the window. In this case, $Min\_Thr = 300$ and $Max\_Thr = 600$ and the results are shown in 4.16. As shown in the figure, both TONTA and μTonta have missed most of the true alarms. The main reason is that the minimum size of the window is high resulting in concealing the true alarms in a bunch of data points. In other words, the high minimum window size reduces the performance of the method significantly because the minimum size of the data points is so high and the method is unable to detect the CP in large subsets of datapoints. Based on evaluating the proposed method in these three scenarios, the optimal value for the parameters is offered by the second scenario, where the size of the window is big but the minimum size is small; however, compared to the default values, none of the evaluated scenarios has yielded better results.

Min_Thr	Max_Thr	FP		FN		True Alarm	
		µTonta	TONTA	µTonta	TONTA	µTonta	TONTA
100	150	3	4	3	6	7	4
100	600	4	6	1	3	9	7
300	600	5	2	6	6	4	4

Table 4.7: Comparing the effects of changing Min_Thr and Max_Thr in μTonta and TONTA

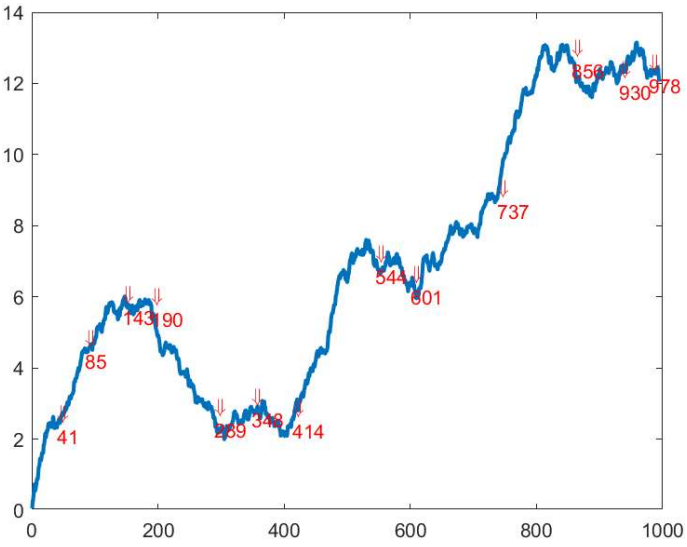


(a)

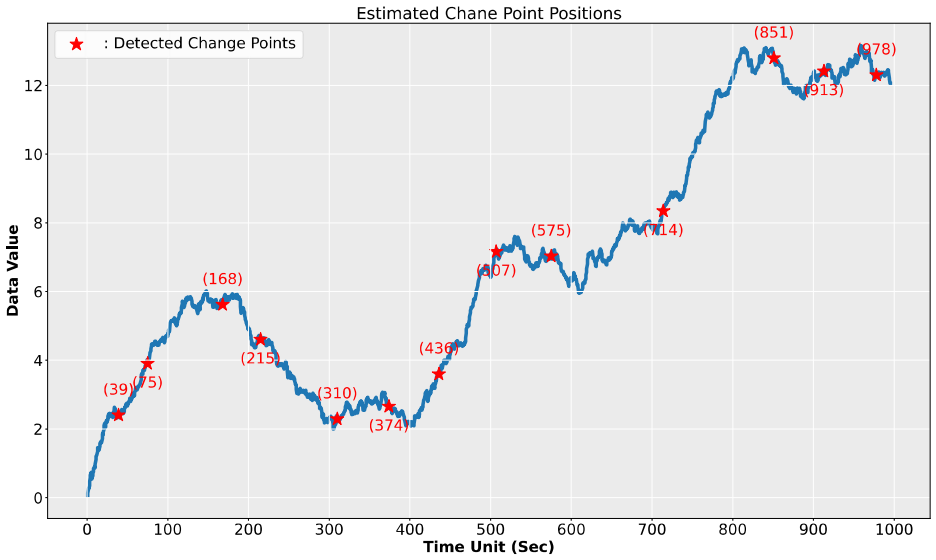


(b)

Figure 4.14: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=150 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

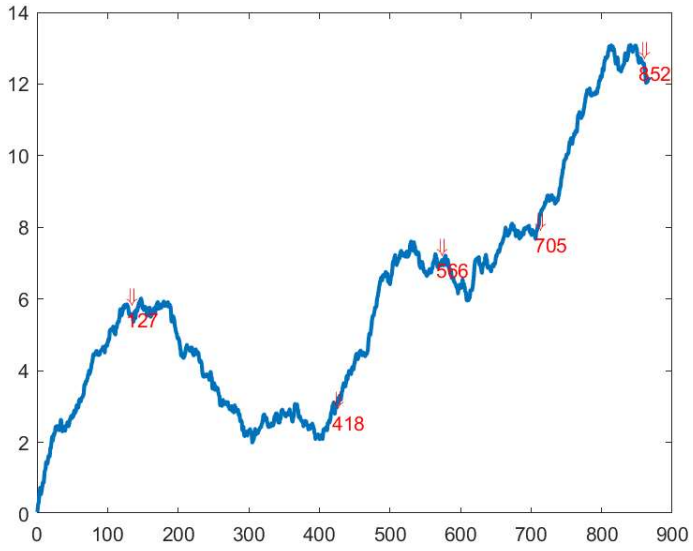


(a)

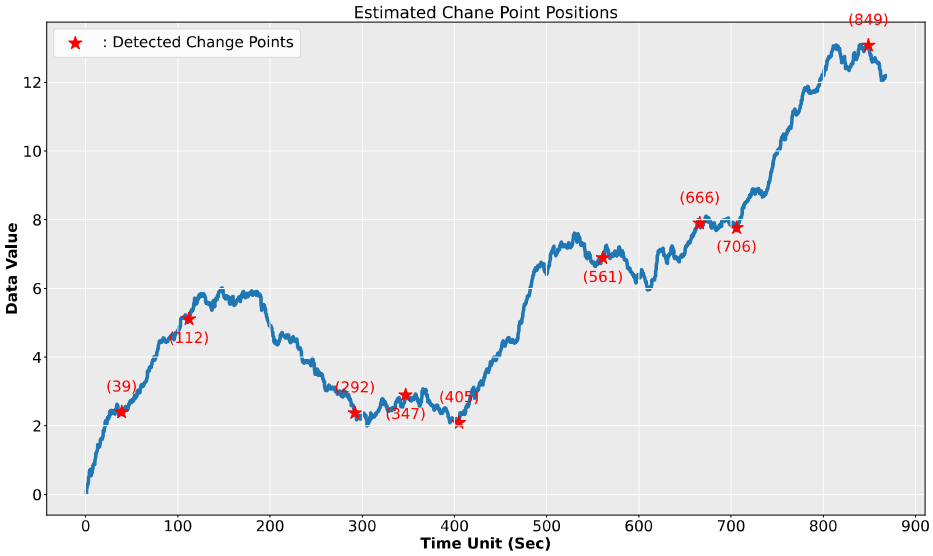


(b)

Figure 4.15: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=600 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta



(a)



(b)

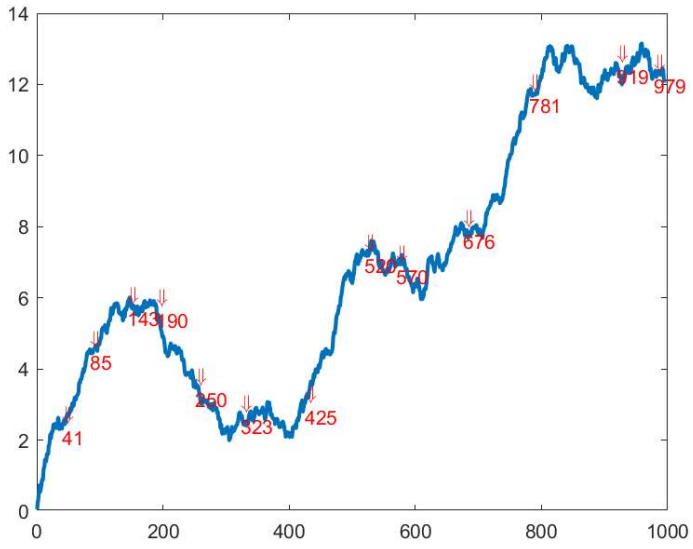
Figure 4.16: the obtained results for *(Interval_Thr=50, Min_Thr=300, Max_Thr=600 Imp_Thr=0.5, ε=100), (a)* by TONTA, and *(b)* by μTonta

4.3.5 Evaluating the effects of *Imp_Thr*

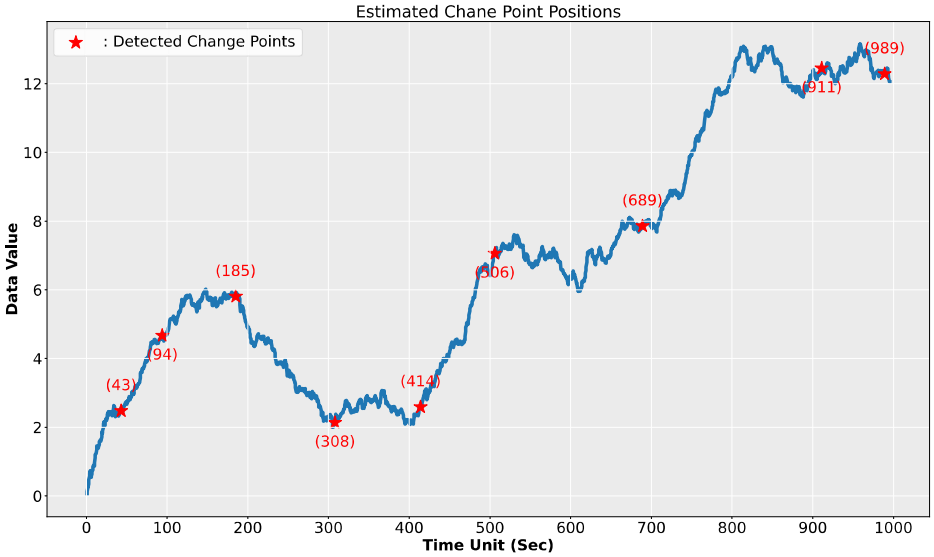
The results of evaluating μTonta and TONTA in terms of the parameter Imp_Thr is shown in Table 4.8 extracted from Figures 4.17, 4.18, and 4.19. Among the three investigated cases, in the first scenario, the Imp_Thr parameter has the lowest value of 0.2. Nonetheless, in this setting, while the μTonta model has successfully decreased the false positive detections of the TONTA, this achievement has come at the cost of a decline in the number of true alarms. Comparing the results of TONTA and μTonta for large values of Imp_Thr, i.e., 0.8 and 1, shows that adopting large values for this parameter visibly degrades the performance of both TONTA and μTonta methods. However, this degrading effect appears in each of the two methods in a different way, i.e., a drastic decrease of TAs in TONTA and a significant increase of FPs in μTonta. Another noteworthy point is that, unlike TONTA, μTonta is sensitive to changes in the Imp_Thr parameter in terms of the number of FPs. In general, it seems that smaller values of the desired parameter lead to better performance in both models.

Imp_Thr	FP		FN		True Alarm	
	µTonta	TONTA	µTonta	TONTA	µTonta	TONTA
0.2	1	3	2	0	8	10
0.8	4	3	2	4	8	6
1	5	3	1	4	9	6

Table 4.8: Comparing the effects of changing Imp_Thr in μTonta and TONTA

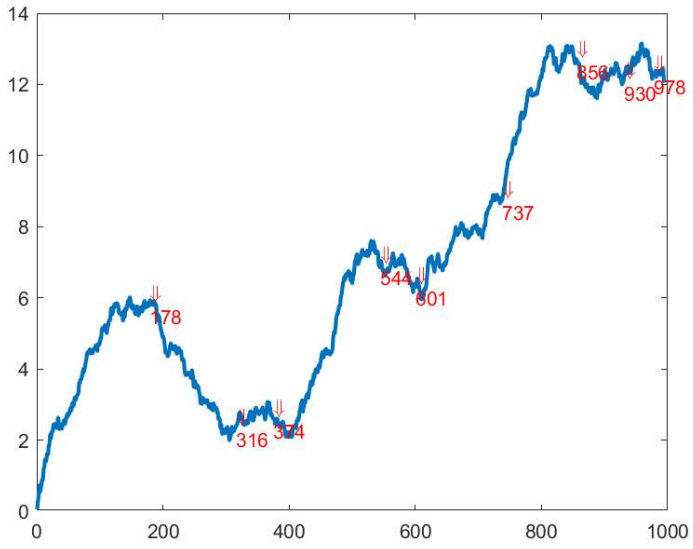


(a)

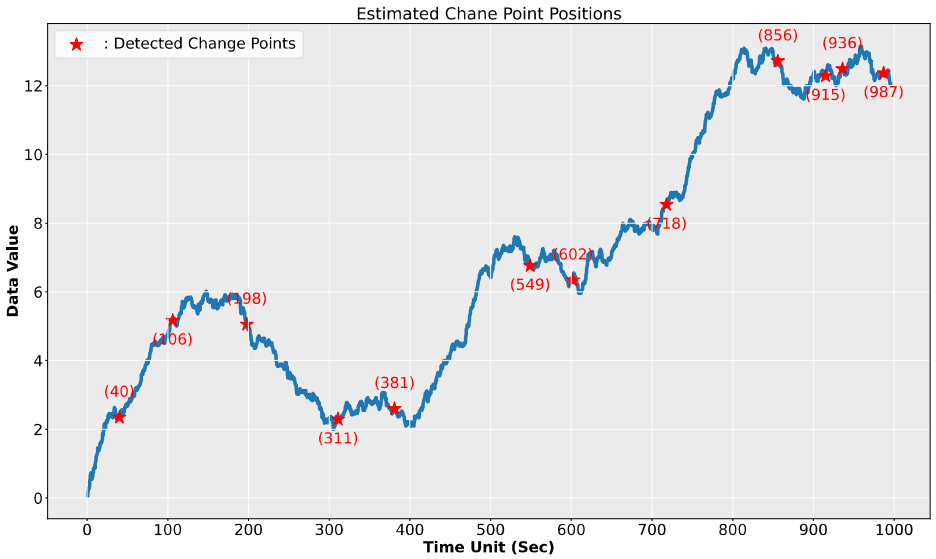


(b)

Figure 4.17: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=300 Imp_Thr=0.2, ε=100), (a)* by TONTA, and *(b)* by μTonta

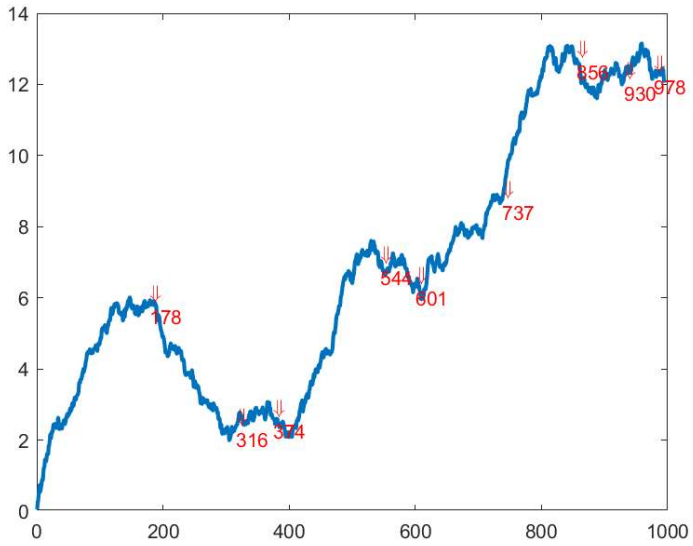


(a)

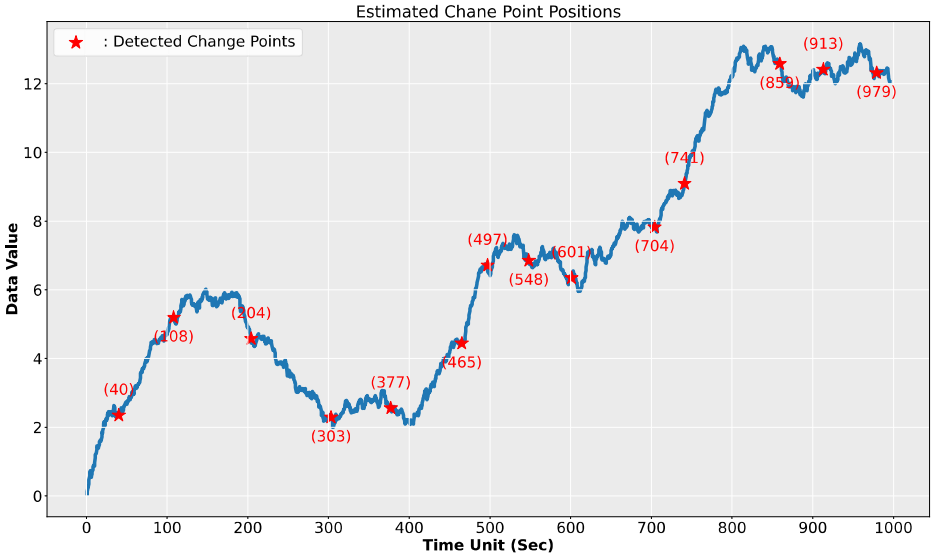


(b)

Figure 4.18: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=300 Imp_Thr=0.8, ε=100), (a)* by TONTA, and *(b)* by μTonta



(a)



(b)

Figure 4.19: the obtained results for *(Interval_Thr=50, Min_Thr=100, Max_Thr=300 Imp_Thr=1, ε=100), (a)* by TONTA, and *(b)* by μTonta

4.3.6 Evaluating The Trade-off Between ε and Min_Thr

As was discussed in section 3.3.1, in order to reduce the time complexity through adjusting the trade-off between lowering ε and increasing Min_Thr, two scenarios was examined. In the first scenario, we compared the outcomes of $\varepsilon = 50$ and $Min\_Thr = 150$ (illustrated in Figure 4.21) with $\varepsilon = 50$ and $Min\_Thr = 100$. In the second scenario, we evaluated the results of $\varepsilon = 75$ and $Min\_Thr = 125$ (shown in Figure 4.20) against

$\varepsilon = 75$ and $Min\_Thr = 100$. Table 4.9 provides a comparative assessment of these scenarios and their corresponding settings, both by μTonta. This evaluation indicates that in parallel with the enhancement achieved on time complexity, the method's accuracy is maintained as well, specifically in the second scenario (i.e., $\varepsilon = 75$ and $Min\_Thr = 125$).

	(Min_Thr=100 , ε=50)	*(Min_Thr=150 , ε=50)*	*(Min_Thr=100 , ε=75)*	*(Min_Thr=125 , ε=75)*
FP	3	4	2	3
TA	9	8	8	9

Table 4.9: Comparing the effects of changing ε in balance with Min_Thr

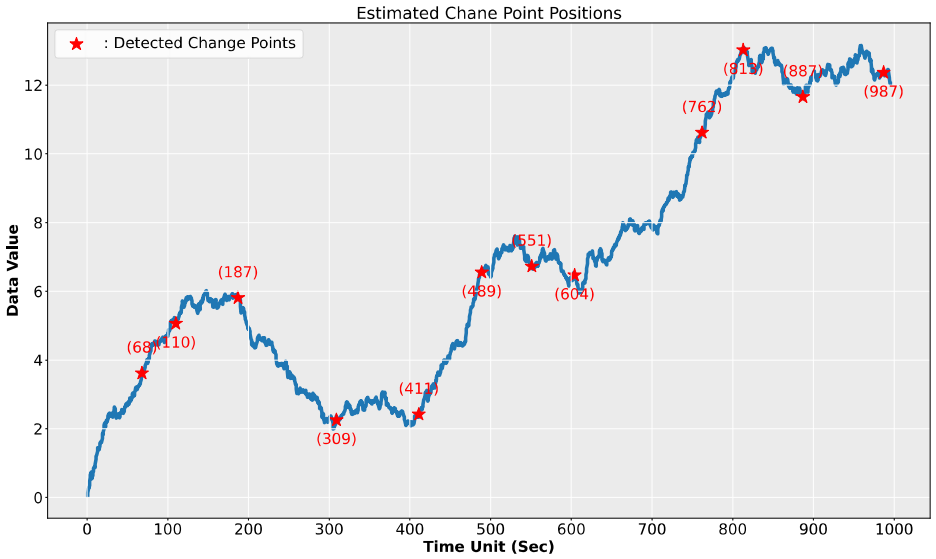


Figure 4.20: the obtained results for *(Interval_Thr=50, Min_Thr=125, Max_Thr=300 Imp_Thr=0.5, ε=75)*

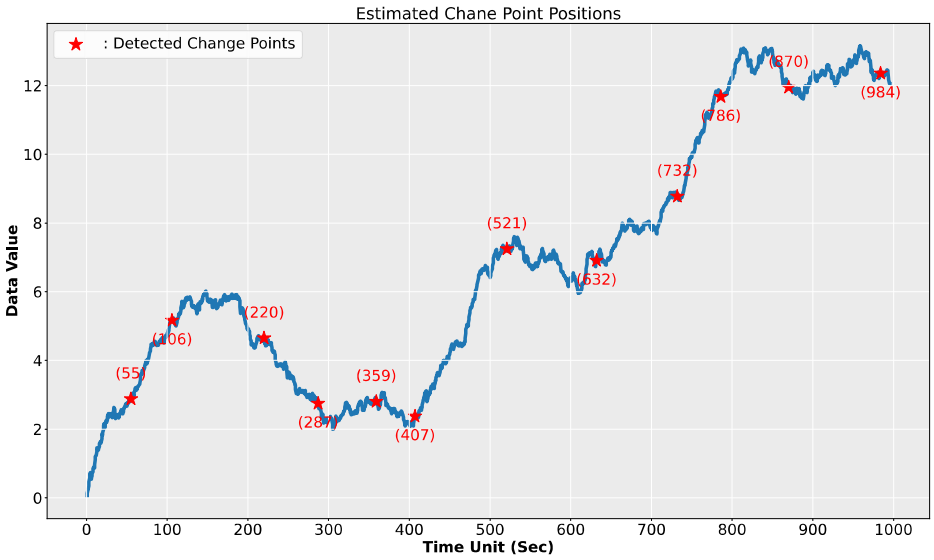


Figure 4.21: the obtained results for *(Interval_Thr=50, Min_Thr=150, Max_Thr=300 Imp_Thr=0.5, ε=50)*

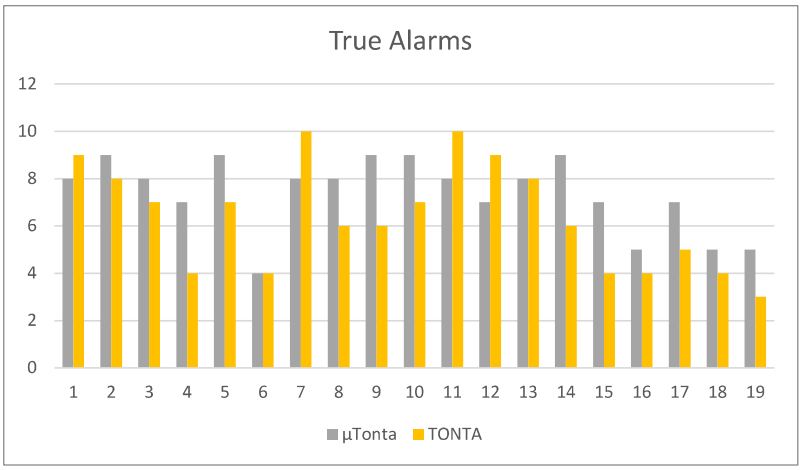
4.4 Evaluating The Overall Performance

Overall, the conducted evaluations in this chapter evidence that the μTonta model outperforms the TONTA method in various aspects. However, with appropriate parameter adjustments, it is possible to enhance the performance of μTonta even more.
As mentioned in section 4.3, all scenarios discussed in TONTA were reproduced by μTonta and compared in terms of FPs and TAs. Considering that to decrease FPs and increase TAs were among the main objectives of the μTonta method, a comparison of two methods in terms of these objectives has provided in Figuer 4.22. Further, Table 4.10 provides some useful information on how μTonta performs in various scenarios, including the percentage change of TAs and FPs and the resultant of them in relation to TONTA. This table helps to gain insights into how the considered parameters influence μTonta's performance as well as outlines the settings in which μTonta has had the best and worst results.
The results show that the proposed method has reduced the number of false alarms for about 2% and improve recognizing the number of true alarms for about 24% across all scenarios. However, these measures have been adversely affected by certain boundary and non-applicable parameter settings in some scenarios like 6, 17, 18, and 19. On the other hand, the method has achieved its best results at scenarios 4 and 10 (the default scenario), where the resultant of TAs and FPs shows an improvement of 100% and 95%, respectively.
In addition, as discussed in Section 4.3.6, the time complexity of the proposed method can be significantly reduced by appropriately adjusting parameters ε and Min_Thr. This optimization improves the model's efficiency, making it more applicable to run on poor-resource IoT devices.
Last but not least, since the method is online, recognizing change points closer to the time that they happen is a big advantage. Figure 4.10 shows the comparison between the two methods in terms of TA delay. Although calculating the total average of delay by μTonta method across all scenarios shows a 41% reduction, a closer look at the graph reveals that

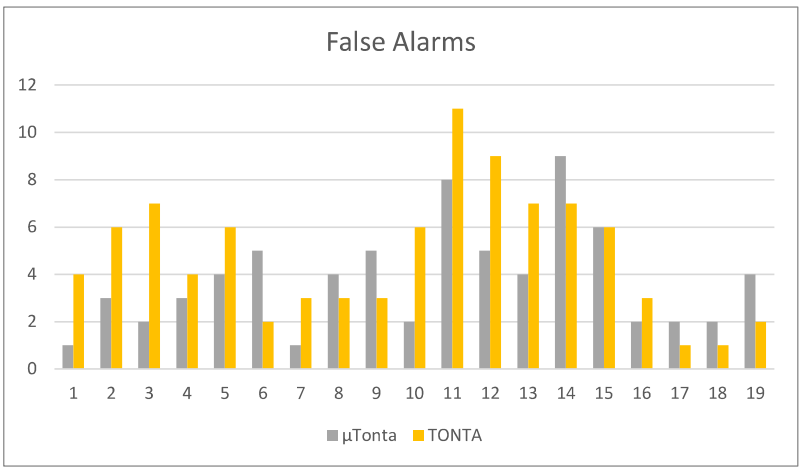
this average is skewed by a few exceptional cases. In fact, for the majority of scenarios we see a noteworthy enhancement in this aspect.

# Scenario	μTonta		TONTA		TAs growth	FPs growth	Resultant
	TA	FP	TA	FP			
1	8	1	9	4	-11.11%	-75.00%	63.89%
2	9	3	8	6	12.50%	-50.00%	62.50%
3	8	2	7	7	14.29%	-71.43%	85.71%
4	7	3	4	4	75.00%	-25.00%	100.00%
5	9	4	7	6	28.57%	-33.33%	61.90%
6	4	5	4	2	0.00%	150.00%	-150.00%
7	8	1	10	3	-20.00%	-66.67%	46.67%
8	8	4	6	3	33.33%	33.33%	0.00%
9	9	5	6	3	50.00%	66.67%	-16.67%
10 (default)	9	2	7	6	28.57%	-66.67%	95.24%
11	8	8	10	11	-20.00%	-27.27%	7.27%
12	7	5	9	9	-22.22%	-44.44%	22.22%
13	8	4	8	7	0.00%	-42.86%	42.86%
14	9	9	6	7	50.00%	28.57%	21.43%
15	7	6	4	6	75.00%	0.00%	75.00%
16	5	2	4	3	25.00%	-33.33%	58.33%
17	7	2	5	1	40.00%	100.00%	-60.00%
18	5	2	4	1	25.00%	100.00%	-75.00%
19	5	4	3	2	66.67%	100.00%	-33.33%
Overall average:					23.72%	2.24%	

Table 4.10: Comparison of μTonta and TONTA in terms of the growth percentage of the obtained results



(a) in terms of the true alarms



(b) in terms of the false alarms

Figure 4.22: Comparison of μTonta and TONTA, based on their results over the 19 examined scenarios

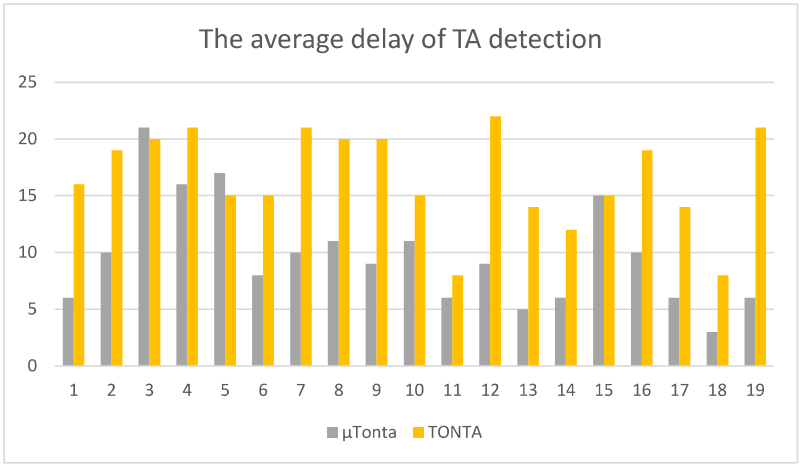


Figure 4.23: Comparing the delay of two methods in detecting true alarms over all examined scenarios

Chapter 5

Discussion

The increasingly expansion of IoT in the recent years has gained a lot of attention from the community as different methods and techniques are introduced aiming to improve the performance of networking in IoT. As an important topic, monitoring the performance of networking in IoT is investigated significantly by the community.
NTMA is an umbrella term for various tasks and techniques such as NTA, NTP, NTC, and so on, serving the network management goals like QoS/QoE, SLA, security, and fault detection among others. Each of these tasks are relying on some specific techniques that benefit from big data analytics techniques and tools. Network traffic behavior analysis and pattern recognition can be thought of as sub-techniques of NTP which by providing knowledge about the current state and the pattern of fluctuations in traffic data enable a prediction model to predict the future changes of data. The focus of this master's thesis is on the use of Pattern Recognition, one of the key NTA techniques, in this framework, we raised three research questions that served as the central themes of our study.

RQ 1: *What are the current approaches to predicting network traffic , and how effective are they?*
in order to investigate RQ1, we conducted a comprehensive review on the topic (presented in Appendix A). Based on the outcomes from that research, the common approaches to NTP and its sub-field i.e. NTA can be categorized mainly into two groups: ML-based and statistical methods. However, some heuristic and combined methods can also be mentioned. While ML-based approaches leverage machine learning and deep learning algorithms to predict the future changes in network traffic data by learning the pattern of historical data, statistical approaches use statistical modeling techniques to analyze the statistical properties of the data and make predictions based on those properties. Statistical methods in particular, demonstrate significant capabilities for dealing with time series data. On the other hand, ML-based approaches offer flexibility and can handle complex relationships. Nonetheless, they are strictly depended on train an retraining from the new emerging data , which demands extensive processing resources. In this sense, they are categorized as heavy processes.

RQ 2: *What are the challenges with utilizing NTA in ad-hoc IoT networks? And which approaches are more suitable for analyzing network traffic on ad-hoc devices in terms of accuracy and computational cost?*
In Chapter 2, along with other fundamentals and features of NTMA, we discussed the

challenges associated with applying NTA techniques in ad-hoc IoT networks, namely that of dynamic environments, network traffic encryption, limited processing resources, and time-dependent applications. These complexities make it difficult to effectively utilize ML-based solutions in ad-hoc IoT networks. In comparison with ML-based techniques, statistical methods benefit from the two key advantages of being lightweight and requiring no training. In addition, these methods have special capabilities for analyzing time series data, particularly, when cracking on metrics such as delay, jitter, and throughput, which are produced in the form of continuous streams of time series data. To wrap up, Despite the good reputation of ML methods and especially deep learning techniques in dealing with complex problems and working with big data, the generic resource limitation of ad-hoc devices , and extensive time-dependence of IoT applications are considerable barriers to the effective leveraging of such techniques. Hence, in many circumstances, statistical techniques can be thought of as acceptable alternatives which reduce the computational cost considerably.

RQ 3: *How can a method be designed to recognize the patterns of network behavior in ad-hoc IoT networks, and how does it compare to existing methods in terms of accuracy and computational cost?*
In response to last research question, which constitutes the core of this thesis project, it was shown that Any solution in the topic's realm should focus on two thoughts: (1) a methodology devoid of the complexity and weight of ML-based approaches, and (2) selecting suitable metrics in terms of direct accessibility and simplicity without the need for extra processing. By these two principles in mind, we have opted for a statistical approach as a more practical alternative of heavy ML-based techniques. Statistical methods are well-suited to this field due to their lack of training requirements and their specific capabilities to deal with time series data. Among the various available techniques, CPD is a popular technique for analyzing network traffic behavior that, while being lightweight, can effectively satisfy a significant portion of the current network behavior analysis needs. In this vein, we introduced a new method (as an extension of a method called TONTA which is introduced in 2020 [3]) entitled μTonta which is mainly focused on improving the performance of TONTA and solves the challenges mentioned in the cornerstone research. In chapter 3, The base method has been explained and analysed along with explaining our proposed improvements to it. To show the effect of adopted enhancements, in chapter 4 we replicated all the scenarios examined in TONTA. Based on this, then, a through comparison between the proposed method and its base method was conducted.
The results indicate that μTonta has reduced the number of false alarms and improved the recognition of true alarms across the majority of scenarios. However, there were some scenarios where certain parameter settings adversely affected the performance. On the other hand, μTonta achieved its best results in more practical scenarios, demonstrating significant improvements in TAs and FPs.
Considering the typical characteristics of highly dynamic environments like IoT networks Applying NTA in online manner, calls for light weight and agile techniques that can deal with challenges like insufficient historical data and temporal dependency. In this sense, the proposed method offers noteworthy enhancements in terms of time complexity and less delay in detecting true alarms. As shown in the evaluation chapter, Adjusting the parameters of μTonta can significantly reduce the time complexity, making it more efficient and suitable for resource-constrained IoT devices. Moreover, thanks to the alterations in

the change point detection algorithm, we managed to considerably decrease the time gap between the actual occurrence of a change point and its detection by the model (i.e. TAs). Although the proposed method still suffers from false alarms, the improved version has higher performance which is so valuable for various applications, especially applications that are sensitive to triggering network reconfiguration frequently. Last but not least, the higher performance of the proposed method compare to TONTA can allow the users to run it on weaker devices with lower over head. Finally, it should be considered that both TONTA and μTONTA have not been implemented in real world environments and they are limited to simulated experiments. This deficiency opens up a new chapter for future research in this field, with the aim of developing and improving the proposed method in the real-world environment and using real data.

Chapter 6

Conclusion

The expansion of IoT has sparked significant interest in optimizing networking performance through various methods and techniques. All these solutions are gathered under the title "Network Traffic Monitoring and Analysis (NTMA)" by focus on different tasks and goals within network management frame. One of the most important tasks in that frame is Network Traffic Analysis (NTA), which provides applicable insight into the state and behavior of network traffic data. In this master's thesis, within NTA paradigm, I have focused on a promising technique for pattern recognition in ad-hoc IoT networks. According to the raised research questions as the main pivots of this thesis the obtained results can be summarized as follow:
The common approaches and techniques adopted under NTMA umbrella, mainly, take place in one of statistical or ML-based approaches. While the former extracts knowledge from evaluating the statistical parameters of network traffic data, the latter learns from historical data. In parallel with the desirable accuracy, ML-based techniques typically need significant processing resources. On the other hand, IoT is extremely relayed on some types of networks constructed on-the-fly by a group of end nodes without the fixed topology and dedicated devices for control and routing. These devices typically are limited-resource which obstacles running heavy ML processes on them.
In promising to develop a lightweight model applicable for online pattern recognition in ad-hoc IoT networks, a cutting-edge change point detection technique called "TONTA" was selected as the cornerstone of the proposed method. TONTA is mainly designed for evaluating the behavior of streams on poor-resource IoT devices. The new method called "μTonta" aimed to improve TONTA's challenges by reducing time complexity and detection delay while enhancing accuracy. The evaluation shows that μTonta has achieved significant enhancements in compare to its precedent approach regarding the defined objectives.
Considering the enhancements in performance obtained with the applied improvements, we assess the proposed method as promising for smoothing network traffic management in IoT networks. As an open area to the future research, implementation of the proposed method in real-world environments using the real data is considered. Further, the proposed approach may also be extended to different kinds of networks and can serve as a foundation for developing more advanced NTA techniques. It is our aspiration that this thesis has contributed towards the growing body of research on NTA for ad-hoc IoT networks and provides valuable insights into the challenges and opportunities in this domain.

Bibliography

[1] A. Heidari, N. J. Navimipour, and M. Unal, "Applications of ml/dl in the management of smart cities and societies based on new trends in information technologies: A systematic literature review," *Sustainable Cities and Society*, vol. 85, p. 104 089, 2022, ISSN: 2210-6707. DOI: https://doi.org/10.1016/j.scs.2022.104089. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670722004061.

[2] S. Siddiqui, S. Hameed, S. A. Shah, *et al.*, "Toward software-defined networking-based iot frameworks: A systematic literature review, taxonomy, open challenges and prospects," *IEEE Access*, vol. 10, pp. 70 850–70 901, 2022. DOI: 10.1109/ACCESS.2022.3188311.

[3] A. Shahraki, A. Taherkordi, and Ø. Haugen, "Tonta: Trend-based online network traffic analysis in ad-hoc iot networks," *Computer Networks*, vol. 194, p. 108 125, 2021.

[4] P. Casas, A. D'Alconzo, T. Zseby, and M. Mellia, "Big-DAMA: Big data analytics for network traffic monitoring and analysis," in *LANCOMM 2016 - Proceedings of the 2016 ACM SIGCOMM Workshop on Fostering Latin-American Research in Data Communication Networks, Part of SIGCOMM 2016*, 2016, ISBN: 9781450344265. DOI: 10.1145/2940116.2940117.

[5] B. M. Leiner, V. G. Cerf, D. D. Clark, *et al.*, "A brief history of the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 5, pp. 22–31, Oct. 2009, ISSN: 0146-4833. DOI: 10.1145/1629607.1629613. [Online]. Available: https://doi.org/10.1145/1629607.1629613.

[6] P. P. Ray, "A survey on internet of things architectures," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, pp. 291–319, 3 Jul. 2018, ISSN: 1319-1578. DOI: 10.1016/J.JKSUCI.2016.10.003.

[7] S. Li, L. D. Xu, and S. Zhao, "The internet of things: A survey," *Information Systems Frontiers*, vol. 17, pp. 243–259, 2 Apr. 2015, ISSN: 15729419. DOI: 10.1007/S10796-014-9492-7.

[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys and Tutorials*, vol. 17, pp. 2347–2376, 4 Oct. 2015, ISSN: 1553877X. DOI: 10.1109/COMST.2015.2444095.

[9] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, pp. 1645–1660, 7 Sep. 2013, ISSN: 0167-739X. DOI: 10.1016/J.FUTURE.2013.01.010.

[10] S. Svorobej, P. T. Endo, M. Bendechache, *et al.*, "Simulating fog and edge computing scenarios: An overview and research challenges," *Future Internet 2019, Vol. 11, Page 55*, vol. 11, p. 55, 3 Feb. 2019, ISSN: 1999-5903. DOI: 10.3390/FI11030055. [Online]. Available: https://www.mdpi.com/1999-5903/11/3/55/htm%20https://www.mdpi.com/1999-5903/11/3/55.

[11] R. Hasan, M. M. Hossain, and R. Khan, "Aura: An iot based cloud infrastructure for localized mobile computation outsourcing," *Proceedings - 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2015*, pp. 183–188, Jun. 2015. DOI: 10.1109/MOBILECLOUD.2015.37.

[12] D. Sikeridis, I. Papapanagiotou, B. P. Rimal, and M. Devetsikiotis, "A comparative taxonomy and survey of public cloud infrastructure vendors," Oct. 2017. DOI: 10.48550/arxiv.1710.01476. [Online]. Available: https://arxiv.org/abs/1710.01476v2.

[13] S. Yang, "IoT Stream Processing and Analytics in the Fog," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 21–27, 2017, ISSN: 01636804. DOI: 10.1109/MCOM.2017.1600840. arXiv: 1705.05988.

[14] A. Yousefpour, C. Fung, T. Nguyen, *et al.*, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[15] F. Jalali, O. J. Smith, T. Lynar, and F. Suits, "Cognitive IoT gateways: Automatic task sharing and switching between cloud and Edge/Fog computing," in *SIGCOMM Posters and Demos 2017 - Proceedings of the 2017 SIGCOMM Posters and Demos, Part of SIGCOMM 2017*, Association for Computing Machinery, Inc, Aug. 2017, pp. 121–123, ISBN: 9781450350570. DOI: 10.1145/3123878.3132008.

[16] V. Ramasamy, "Recent advances in ad-hoc networks," in *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2017, pp. 132–132. DOI: 10.1109/ICRITO.2017.8342413.

[17] J. R. Parvin, "An overview of wireless mesh networks," *Wireless Mesh Networks-Security, Architectures and Protocols*, 2019.

[18] T. Salman and R. Jain, "Networking protocols and standards for internet of things," in *Internet of Things and Data Analytics Handbook*. John Wiley & Sons, Ltd, 2017, ch. 13, pp. 215–238, ISBN: 9781119173601. DOI: https://doi.org/10.1002/9781119173601.ch13.

[19] A. Ghosh and A. Senthilrajan, "Research on Packet Inspection Techniques," *International Journal of Scientific & Technology Research*, vol. 8, pp. 2068–2073, 2019.

[20] P. Mallory. "Network traffic analysis for ir: Data collection and monitoring - infosec resources." (Dec. 2019), [Online]. Available: https://resources.infosecinstitute.com/topic/network-traffic-analysis-for-ir-data-collection-and-monitoring/.

[21] J. Kögel, "One-way delay measurement based on flow data: Quantification and compensation of errors by exporter profiling," *International Conference on Information Networking 2011, ICOIN 2011*, pp. 25–30, 2011. DOI: 10.1109/ICOIN.2011.5723108.

[22] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, and P. Casas, "A survey on big data for network traffic monitoring and analysis," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 800–813, Sep. 2019, ISSN: 19324537. DOI: 10.1109/TNSM.2019.2933358.

[23] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (ntma): A survey," *Computer Communications*, 2021.

[24] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1988–2014, Apr. 2019, ISSN: 1553877X. DOI: 10.1109/COMST.2018.2883147.

[25] C. Brook. "What is deep packet inspection? how it works, use cases for dpi, and more | digital guardian." (Dec. 2018), [Online]. Available: https://digitalguardian.com/blog/what-deep-packet-inspection-how-it-works-use-cases-dpi-and-more.

[26] D. H. Hagos, P. E. Engelstad, A. Yazidi, and O. Kure, "A machine learning approach to TCP state monitoring from passive measurements," in *IFIP Wireless Days*, vol. 2018-April, IEEE Computer Society, May 2018, pp. 164–171, ISBN: 9781538656327. DOI: 10.1109/WD.2018.8361713.

[27] S. Niccolini, M. Molina, F. Raspall, and S. Tartarelli, "Design and implementation of a one way delay passive measurement system," *IEEE Symposium Record on Network Operations and Management Symposium*, pp. 469–482, 2004. DOI: 10.1109/NOMS.2004.1317734.

[28] M. Abbasi, A. Shahraki, M. J. Piran, and A. Taherkordi, "Deep reinforcement learning for qos provisioning at the mac layer: A survey," *Engineering Applications of Artificial Intelligence*, vol. 102, p. 104 234, 2021.

[29] H. Elbouanani, C. Barakat, W. Dabbous, and T. Turletti, "Passive delay measurement for fidelity monitoring of distributed network emulation," *Computer Communications*, vol. 195, pp. 40–48, 2022.

[30] O. TG, O. DO, U. CC, and F. TA, "Qos performance metrics for analyzing wireless network usability," 2019.

[31] H.-J. Lee, M.-S. Kim, J. W. Hong, and G.-H. Lee, "Qos parameters to network performance metrics mapping for sla monitoring," *KNOM Review*, vol. 5, no. 2, pp. 42–53, 2002.

[32] S. Qaiyum, I. A. Aziz, and J. B. Jaafar, "Analysis of big data and quality-of-experience in high-density wireless network," in *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 2016, pp. 287–292. DOI: 10.1109/ICCOINS.2016.7783229.

[33] "One-way delay measurement techniques," Accedian Networks Inc., Jan. 2012. [Online]. Available: https://accedian.com/wp-content/uploads/2015/05/One-WayDelayMeasurementTechniques-AccedianWhitePaper.pdf.

[34] G. Shute. "Network delays and losses." (2016), [Online]. Available: https://www.d.umn.edu/~gshute/net/delays-losses.xhtml.

[35] T. Mizrahi, "Zen and the art of network timestamping white paper," Dec. 2021.

[36] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[37] J. Liu, F. Liu, and N. Ansari, "Monitoring and analyzing big traffic data of a large-scale cellular network with hadoop," *IEEE Network*, vol. 28, pp. 32–39, 4 2014, ISSN: 08908044. DOI: 10.1109/MNET.2014.6863129.

[38] A. Hanemann, A. Liakopoulos, M. Molina, and D. M. Swany, "A study on network performance metrics and their composition," *Campus-Wide Information Systems*, vol. 23, I. Melve, Ed., pp. 268–282, 4 Jan. 2006, ISSN: 1065-0741. DOI: 10.1108/10650740610704135. [Online]. Available: https://doi.org/10.1108/10650740610704135.

[39] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019, ISSN: 1553877X. DOI: 10.1109/COMST.2019.2904897. arXiv: 1803.04311.

[40] A. Bayati, K. K. Nguyen, and M. Cheriet, "Multiple-Step-Ahead Traffic Prediction in High-Speed Networks," *IEEE Communications Letters*, vol. 22, no. 12, pp. 2447–2450, Dec. 2018, ISSN: 15582558. DOI: 10.1109/LCOMM.2018.2875747.

[41] M. Joshi and T. H. Hadi, "A Review of Network Traffic Analysis and Prediction Techniques," Jul. 2015. arXiv: 1507.05722. [Online]. Available: http://arxiv.org/abs/1507.05722.

[42] R. Boutaba, M. A. Salahuddin, N. Limam, *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Dec. 2018, ISSN: 18690238. DOI: 10.1186/s13174-018-0087-2. [Online]. Available: https://doi.org/10.1186/s13174-018-0087-2.

[43] S. Aminikhanghahi and D. J. Cook, "Enhancing activity recognition using cpd-based activity segmentation," *Pervasive and Mobile Computing*, vol. 53, pp. 75–89, Feb. 2019, ISSN: 1574-1192. DOI: 10.1016/J.PMCJ.2019.01.004.

[44] S. Aminikhanghahi, T. Wang, and D. J. Cook, "Real-time change point detection with application to smart home time series data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, pp. 1010–1023, 5 May 2019, ISSN: 15582191. DOI: 10.1109/TKDE.2018.2850347.

[45] R. Hyndman and G. Athanasopoulos. "Forecasting: Principles and practice (3rd ed)." (2021), [Online]. Available: https://otexts.com/fpp3/.

[46] A. Nielsen, *Practical time series analysis: Prediction with statistics and machine learning*. O'Reilly Media, 2019.

[47] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*. Somerset, UNITED STATES: John Wiley & Sons, Incorporated, 2015, ISBN: 9781118745229. [Online]. Available: http://ebookcentral.proquest.com/lib/hiof-ebooks/detail.action?docID=1895570.

[48] R. Nau. "Statistical forecasting: Notes on regression and time series analysis." (Aug. 2020), [Online]. Available: https://people.duke.edu/~rnau/411home.htm.

[49] C. Katris and S. Daskalaki, "Dynamic bandwidth allocation for video traffic using farima-based forecasting models," *Journal of Network and Systems Management 2018 27:1*, vol. 27, pp. 39–65, 1 Apr. 2018, ISSN: 1573-7705. DOI: 10.1007/S10922-018-9456-1. [Online]. Available: https://link.springer.com/article/10.1007/s10922-018-9456-1.

[50] N. Dengen *et al.*, "Comparison of sarima, narx and bpnn models in forecasting time series data of network traffic," in *2016 2nd International Conference on Science in Information Technology (ICSITech)*, IEEE, 2016, pp. 264–269.

[51] W. C. Hong, "Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting," *Neural Computing and Applications*, vol. 21, no. 3, pp. 583–593, Apr. 2012, ISSN: 09410643.

[52] A. Biernacki, "Improving quality of adaptive video by traffic prediction with (F)ARIMA models," *Journal of Communications and Networks*, vol. 19, no. 5, pp. 521–530, Oct. 2017, ISSN: 12292370. DOI: 10.1109/JCN.2017.000083.

[53] A. Nielsen, *Practical time series analysis: prediction with statistics and machine learning.* " O'Reilly Media, Inc.", 2019.

[54] T. Andrysiak, L. Saganowski, M. Maszewski, and A. Marchewka, "Detection of network attacks using hybrid arima-garch model," *Advances in Intelligent Systems and Computing*, vol. 582, pp. 1–12, 2018.

[55] R. Engle, "Garch 101: The use of arch/garch models in applied econometrics," *Journal of Economic Perspectives*, vol. 15, pp. 157–168, 4 2001, ISSN: 0895-3309. DOI: 10.1257/JEP.15.4.157.

[56] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986, ISSN: 03044076. DOI: 10.1016/0304-4076(86)90063-1.

[57] H. Chen, J. Zhang, Y. Tao, and F. Tan, "Asymmetric garch type models for asymmetric volatility characteristics analysis and wind power forecasting," *Protection and Control of Modern Power Systems 2019 4:1*, vol. 4, pp. 1–11, 1 Dec. 2019, ISSN: 2367-0983. DOI: 10.1186/S41601-019-0146-0. [Online]. Available: https://pcmp.springeropen.com/articles/10.1186/s41601-019-0146-0.

[58] H. Mehdi, Z. Pooranian, and P. G. Vinueza Naranjo, "Cloud traffic prediction based on fuzzy ARIMA model with low dependence on historical data," *Transactions on Emerging Telecommunications Technologies*, Sep. 2019, ISSN: 2161-3915. DOI: 10.1002/ett.3731. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3731.

[59] Z. Hajirahimi and M. Khashei, "Hybrid structures in time series modeling and forecasting: A review," *Engineering Applications of Artificial Intelligence*, vol. 86, pp. 83–106, Nov. 2019, ISSN: 09521976. DOI: 10.1016/j.engappai.2019.08.018.

[60] M. T. Jones. "Models for machine learning." (Dec. 2017), [Online]. Available: https://developer.ibm.com/articles/cc-models-machine-learning/#reinforcement-learning.

[61] A. Burkov, *The hundred-page machine learning book*. Andriy Burkov Quebec City, QC, Canada, 2019, vol. 1.

[62] J. Delua. "Supervised vs. unsupervised learning: What's the difference? | ibm." (Mar. 2021), [Online]. Available: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.

[63] S. Ayoubi, N. Limam, M. A. Salahuddin, *et al.*, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.

[64] M. Kulin, T. Kazaz, I. Moerman, and E. de Poorter, "A survey on Machine Learning-based Performance Improvement of Wireless Networks: PHY, MAC and Network layer," Jan. 2020. arXiv: 2001.04561. [Online]. Available: http://arxiv.org/abs/2001.04561.

[65] S. Aroussi and A. Mellouk, "Survey on machine learning-based QoE-QoS correlation models," in *2014 International Conference on Computing, Management and Telecommunications, ComManTel 2014*, IEEE, 2014, pp. 200–204, ISBN: 9781479929030. DOI: 10.1109/ComManTel.2014.6825604.

[66] Z.-x. Xu, L. Cao, X.-l. Chen, C.-x. Li, Y.-l. Zhang, and J. Lai, "Deep reinforcement learning with sarsa and q-learning: A hybrid approach," *IEICE TRANSACTIONS on Information and Systems*, vol. 101, no. 9, pp. 2315–2322, 2018.

[67] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European symposium on artificial neural networks (ESANN)*, ESANN, 2016.

[68] S. Ayoubi, N. Limam, M. A. Salahuddin, *et al.*, "Machine Learning for Cognitive Network Management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, Jan. 2018, ISSN: 01636804. DOI: 10.1109/MCOM.2018.1700560.

[69] I. Lohrasbinasab, A. Shahraki, A. Taherkordi, and A. D. Jurcut, "From statistical- to machine learning-based network traffic prediction," *Transactions on Emerging Telecommunications Technologies*, vol. 33, e4394, 4 Apr. 2022, ISSN: 2161-3915. DOI: 10.1002/ETT.4394. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/ett.4394%20https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4394%20https://onlinelibrary.wiley.com/doi/10.1002/ett.4394.

[70] G. A. Segura, S. Skaperas, A. Chorti, L. Mamatas, and C. B. Margi, "Denial of service attacks detection in software-defined wireless sensor networks," *2020 IEEE International Conference on Communications Workshops, ICC Workshops 2020 - Proceedings*, Jun. 2020. DOI: 10.1109/ICCWORKSHOPS49005.2020.9145136.

[71] G. A. Segura, A. Chorti, and C. B. Margi, "Centralized and distributed intrusion detection for resource-constrained wireless sdn networks," *IEEE Internet of Things Journal*, vol. 9, pp. 7746–7758, 10 May 2022, ISSN: 23274662. DOI: 10.1109/JIOT.2021.3114270.

[72] L. Song, E. Halepovic, A. Mohammed, and A. Striegel, "Cup: Cellular ultra-light probe-based available bandwidth estimation," *2021 IEEE/ACM 29th International Symposium on Quality of Service, IWQOS 2021*, Jun. 2021. DOI: 10.1109/IWQOS52092.2021.9521270.

[73] Y. Liu, L. Liu, and Y. Yan, "Network topology change detection based on statistical process control," *ACM International Conference Proceeding Series*, pp. 145–151, Jul. 2020. DOI: 10.1145/3409501.3409532. [Online]. Available: https://dl.acm.org/doi/10.1145/3409501.3409532.

[74] C. Culman, S. Aminikhanghahi, and D. J. Cook, "Easing power consumption of wearable activity monitoring with change point detection," *Sensors 2020, Vol. 20, Page 310*, vol. 20, p. 310, 1 Jan. 2020, ISSN: 1424-8220. DOI: 10.3390/S20010310. [Online]. Available: https://www.mdpi.com/1424-8220/20/1/310/htm%20https://www.mdpi.com/1424-8220/20/1/310.

Glossary

ANN

Artificial Neural networks. 32, 35

ARCH

Autoregressive Conditional Heteroskedasticity. 27

ARIMA

AutoRegressive Integrated Moving Average. 26, 27

ARMA

Autoregression Moving Average. 26, 27

CP

Change Point. 42, 43, 45–47, 51, 54, 57, 70

CPD

Change Point Detection. i, 1, 2, 19, 20, 36, 37, 40, 42, 48

DL

Deep Learning. 32, 35

DNN

Deep Neural Networks. 35

FARIMA

Fractional AutoRegressive Integrated Moving Average. 27

FN

False-Negative. 49, 67

FP

False-Positive. 49, 54, 67, 70, 74

Fuzzy-ARIMA

Fuzzy-AutoRegressive Integrated Moving Average. 27

GARCH

Generalized Autoregressive Conditional Heteroskedasticity. 27

GM

Geometric Mean. 50

i.i.d

Identically and Independently Distributed. 26, 27

IoT

Internet of Things. i, v, 1–5, 7–10, 18–21, 36, 37, 39, 48–50, 84, 87

KPI

Key Performance Indicator. i

ML

Machine Learning. vii, 2, 3, 16, 17, 20, 28–30, 32, 34–36, 39, 40

NTA

Network Traffic Analysis. i, 1–3, 45, 87

NTC

Network Traffic Classification. i, 15, 16, 35

NTMA

Network Traffic Monitoring and Analysis. i, v, 1–3, 5, 11, 15–20, 29, 35, 36, 39, 40

NTP

Network Traffic Prediction. i, 1, 3, 15, 16, 19, 27, 35, 36, 39

OWD

One-way delay. 15, 40–42

QoE

Quality of Experience. 12, 15, 18, 37

QoS

Quality of Service. i, 12, 13, 15, 17, 18

RL

Reinforcement Learning. 30

SARIMA

Seasonal-ARIMA. 26

SL

Supervised Learning. 40

SLA

Service Level Agreement. i, 13, 15, 17

SPI

Shallow Packet Inspection. 41

TA

True-Alarm. 48, 74

TCD

Trend Change Detection. 19

TONTA

Trend-based Online Network Traffic Analysis in ad-hoc IoT networks. 2, 3

TP

True-Positive. 50

Appendix A

(A published research in connection to this thesis)

Received: 22 July 2021 | Accepted: 6 September 2021

DOI: 10.1002/ett.4394

SURVEY PAPER

WILEY

From statistical- to machine learning-based network traffic prediction

Iraj Lohrasbinasab[1] | **Amin Shahraki**[2,3] | **Amir Taherkordi**[2] | **Anca Delia Jurcut**[3]

[1]Faculty of Computer Sciences, Østfold University College, Halden, Norway

[2]Department of Informatics, University of Oslo, Oslo, Norway

[3]School of Computer Science, University College Dublin, Dublin, Ireland

Correspondence
Amin Shahraki, School of Computer Science, University College Dublin, Dublin, Ireland.
Email:am.shahraki@ieee.org

Abstract

Nowadays, due to the exponential and continuous expansion of new paradigms such as Internet of Things (IoT), Internet of Vehicles (IoV) and 6G, the world is witnessing a tremendous and sharp increase of network traffic. In such large-scale, heterogeneous, and complex networks, the volume of transferred data, as *big data*, is considered a challenge causing different networking inefficiencies. To overcome these challenges, various techniques are introduced to monitor the performance of networks, called Network Traffic Monitoring and Analysis (NTMA). Network Traffic Prediction (NTP) is a significant subfield of NTMA which is mainly focused on predicting the future of network load and its behavior. NTP techniques can generally be realized in two ways, that is, statistical- and Machine Learning (ML)-based. In this paper, we provide a study on existing NTP techniques through reviewing, investigating, and classifying the recent relevant works conducted in this field. Additionally, we discuss the challenges and future directions of NTP showing that how ML and statistical techniques can be used to solve challenges of NTP.

1 | INTRODUCTION

During the last decades, new networking paradigms, for example, Wireless Sensor Networks (WSNs), Internet of Things (IoT), Internet of Vehicles (IoV) and 6th generation of cellular networks (6G)[1] have been emerging to establish the network infrastructures for real-world applications, for example, smart cities, crisis management, smart roads, etc..[2] Thanks to miniaturization of digital equipment, today's networks include thousands of connected User Equipments (UEs) (known as end node devices) that can generate and/or consume data. IoT as a new emerging networking paradigm, provides an overlay network on top of other network infrastructures, from Near-Field Communication (NFC) to cellular networks to connect a virtually unlimited number of UEs.[3] It is expected that in 2025, the number of connected IoT devices will increase to 75 billion as predicted by Cisco.[4,5] While managing such numerous devices is a challenging issue, other characteristics of networks, for example, heterogeneity and mobility, can cause inefficiency in networking. Network heterogeneity is not solely due to the diversity of device types, as it can also be related to some other factors such as the volume of data generated by each connected UEs, the required services, and the diversity of network connections. Given these characteristics of new networking paradigms, the volume of data generated may be very huge which has given rise to the Big Data era.[6]

To provide an efficient network infrastructure to transfer and manage such a huge volume of data, different techniques are introduced, mainly to prevent various network faults and inefficiencies, support Quality of Service (QoS) and

provide security. Apart from the type of the network infrastructure, the QoS depends on the performance of route of data, and even the ability of routing devices to analyze real-time network situation to make dynamic networking adjustments and allocation decisions. Network Traffic Monitoring and Analyzing (NTMA) techniques are mainly introduced to monitor the performance of networking by providing information to analyze the network and offer solutions to address the challenges without human intervention.[7] There are four main subfields in the NTMA including[8] (i) Network Traffic Prediction (NTP),[9] (ii) Network Traffic Classification (NTC),[10] (iii) Fault Management, and (iv) Network Security. Among all these subfields, NTP focuses on analyzing the network load and prediction of the network traffic to avoid faults and inefficiencies in networking. In this study, we focus on NTP, as one of the most critical solutions to address various networking challenges, for example, resource provisioning, congestion control, resource discovery, network behavior analysis, and anomaly detection.[6]

Different techniques are introduced to perform NTP, but generally, existing solutions can be divided into two types, that is, Machine Learning (ML)-based techniques and statistical-based techniques. As NTP can be designed based on both types, we first review the most relevant techniques and then investigate the proposed solutions for each type. The contributions of this study include:

- Investigating the existing NTP techniques and available solutions to predict the network behavior.
- Classifying the NTP techniques based on statistical-, ML-based, and hybrid techniques.
- Providing a concrete future direction based on real-world applications compared to the state of the art techniques, models, and frameworks.
- Proposing a schema to integrate statistical-based techniques and ML-based techniques to improve the performance of NTP techniques.

The rest of the paper is summarized as follows. In Section 2, we first introduce the basic concepts and discuss the available types of techniques for the NTP. In Section 3, we survey and analyze existing solutions and provide a classification of them. In Section 4, we discuss the challenges and future directions, and finally, in Section 5, we conclude this study.

2 | BASIC CONCEPTS

This section briefly explains some key concepts of NTP and existing types of solutions. We first introduce the NTP concept and then continue this section by discussing and comparing the statistical- and ML-based techniques which are further reviewed in Section 3.

2.1 | Network traffic prediction

The continuous and exponential growth in scale, speed, and heterogeneity of network traffic streams, as seen in networking platforms such as IoT and 5G, as well as emerging networking technologies like Software-Defined Network (SDN), Network Function Virtualization (NFV), and Fog computing has proven the desperate need to NTMA solutions.[8,11] On the foundation laid down by Big Data Analytics, and using techniques such as Traffic Prediction, Traffic Classification, Anomaly Detection, and Fault Detection, NTMA is engaged in tackling a range of substantial problems. To name a few, many aspects of network management including fault management, configuration management, accounting management, security management, and performance management achieve effectiveness by utilizing the NTMA capabilities.[12]

At the core of the mentioned techniques, sensing the network status in a real-time manner and analyzing the trend of changes based on the data extracted from the network traffic plays a key role. Maintaining user satisfaction at the desired level (known as Quality of Experience [QoE]) is one of the main goals of performance management which can be achieved by continuously monitoring the status of the networking functions. Considering the inherent relationships between user-oriented QoE and network-oriented QoS parameters, the performance management objectives could be achieved through monitoring and measuring QoS parameters such as throughput, delay, jitter, or loss rate.[13]

The network traffic data can be extracted from the packet inspection process by one of the Deep Packet Inspection (DPI) or Shallow Packet Inspection (SPI) methods.[12] While the first is based on reading and analyzing the full packet contents that include application headers and payload, the second method examines only headers of the

network and/or transport layers in those packets selected based on the adopted sampling strategies.[7] Nevertheless, apart from some specific applications of DPI, in cases such as filtering or troubleshooting—given the lack of feasibility of applying this method due to privacy challenges and imposing significant computational and memory overheads on the network—the real-time Traffic-Monitoring-Centric tasks would rely on SPI for extracting the required data from packet streams.[12,14]

NTP as a subtechnique of NTMA is used to determine the status of the network, identify changes and predict the network traffic behavior in a foreseeable future. Generally, the results of NTP techniques can be used in a wide range of applications, for example, QoS provisioning, fault detection, and security attack detection. The problem of predicting future network traffic volume is traditionally formulated in the form of a Time Series Forecasting (TSF) or rarely a spatiotemporal problem aimed at constructing a regression model capable of estimating future traffic volume by extracting the existing dependencies between historical and future data.[15,16] Typically, low computational overhead, simplicity, and the limited number of features can be referred to as advantages of TSF approaches.[17] On the other hand, due to new demands and exigencies stemmed from the ever-increasing rise in scale, speed, and heterogeneity of networks, non-TSF approaches are becoming more and more propounded.[18] These methods typically leverage flow and packet header data to estimate future incoming flows instead of traffic volume.[19]

Basically, popular NTP solutions are divided into statistical analysis methods and ML-based models.[20,21] In the following, we shed light on some differentiation aspects of common network prediction's methods and techniques.

2.2 | Statistical techniques for NTP

Statistical techniques are mainly based on analyzing the data patterns without having any prior knowledge (without training). Most of such techniques compare the current situation of the data, that is, the pattern of the data with the last identified pattern to recognize important changes. Linear statistical-based models extract patterns from the historical data as well as predict future points in time-series according to the lagged data. The well-known members of this category are Autoregression Moving Average (ARMA) and AutoRegressive Integrated Moving Average (ARIMA), as well as variants of the latter.[22] ARIMA, also known as Box-Jenkins model, is a prevalent paradigm among statistical models for time-series prediction. Both of ARMA and ARIMA models are emerged from the convergence of autoregressive (AR) model which involves with lagged values of observations, and moving average (MA) model which takes lagged errors. Nevertheless, The distinction between them is in their approach to the notion of stationarity in time-series. While ARMA assumes the time-series is stationary, ARIMA would provide the stationarity of data through differencing process which might be applied multiple times, until establishment of stationarity. In "ARIMA" the I (stands for "Integrated") refers to this procedure. ARIMA model is denoted as ARIMA(p, d, q), where p indicates the order of the AR part, d shows the involved differencing degree, and q is the order of the moving average part.[23,24] As concisely described below, ARMA and ARIMA can be formulated, respectively, in Equations (2) and (3). Assuming a time series data X_t consisting of real numbers (x_t) and an integer index (t), an ARMA (p,q) model is given by Equation (1) where α_i are the parameters of the (AR) model, θ_i represents the (MA) model's parameters, and ε_t are error terms assumed to be independent and identically distributed (i,i,d) variables sampled from a normal distribution with zero mean.

$$X_t - \alpha_1 X_{t-1} - \cdots - \alpha_p X_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}. \tag{1}$$

Interchangeably, it can be shown as below where L is the lag operator:

$$\left(1 - \sum_{i=1}^{p} \alpha_i L^i\right) X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t. \tag{2}$$

Interested readers are referred to Reference 23. Assuming the ARIMA model as the generalization of ARMA, the following formula defines an ARIMA(p,d,q) process given drift $\left(\frac{\delta}{1-\sum \varphi_i}\right)$:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right) (1-L)^d X_t = \delta + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t. \tag{3}$$

Diverse variations of ARIMA are proposed according to various applications and time-series; Among them, it is worth mentioning "Seasonal-ARIMA (SARIMA)"[25] and *"Fractional AutoRegressive Integrated Moving Average (FARIMA)"*.[26] The former is often used in NTP considering its compatibility with the nature of changes in networks which usually obeys certain time patterns. A FARIMA forecasting model is an extension of the ARIMA (p, d, q) model in which the fractional parameter d can take real values rather than just integers, and is given by the equation:

$$\boldsymbol{\Phi}_p(L)(1-L)^d(Y_t) = \Theta_q(L)\varepsilon_t, \tag{4}$$

where L is the lag operator, $\boldsymbol{\Phi}_p(L) = 1 - \varphi_1 L - \cdots - \varphi_p L^p$ and $\Theta_q(L) = 1 + \theta_1 L + \cdots + \theta_q L^q$, $(1-L)^d = \sum_{j=0}^{\infty}\binom{d}{j}(-1)^j L^j$ with $\binom{d}{j}(-1) = \frac{\Gamma(-d+j)}{\Gamma(-d)\Gamma(j+1)}$ and $\varepsilon_t \sim N\left(0, \sigma^2\right)$ as the error terms.[27]

In addition to the extant components of the conventional ARIMA, the SARIMA[28] model also includes the frequency component (known as *seasonality* and shown by (S).[29]

An SARIMA model would conduct prediction based on a linear combination of past observations and their related errors. As its name implies, the seasonality factor plays a key role in the structure and performance of this model. The SARIMA process often would be shown as models in the form of $SARIMA(p,d,q)\times(P,D,Q)S$. For given time series $\{X_t\}$ with seasonality length (S), the SARIMA process is indicated by the Equation (5), while the differenced series $w_t = (1-B)^d(1-B^S)^D X_t$ is a stationary ARMA process and, d and D are nonnegative integer values.

$$\phi_p(B)\Phi_P(B^S)w_t = \theta_q(B)\Theta(B^S)\epsilon_t, t = 1, 2, \ldots, n, \tag{5}$$

where:

- n is the number of observations up to time t and the backshift operator B is defined as: $B^a W_t = W_{t-a}$.
- ϵ_t is defined as i.i.d samples with a zero mean and variance σ^2, which for all $K \neq 0$ we have: $\mathrm{Cov}(\epsilon_t, \epsilon_{t-k}) = 0$
- The nonseasonal components are: $\begin{cases} \bullet\ \mathrm{AR}: \phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p \\ \bullet\ \mathrm{MA}: \theta(B) = 1 - \theta_1 B - \ldots - \theta_q B^q \end{cases}$
- The seasonal components are: $\begin{cases} \bullet\ \mathrm{SeasonalAR}: \Phi(B^S) = 1 - \Phi_1 B^S - \ldots - \Phi_P B^{PS} \\ \bullet\ \mathrm{SeasonalMA}: \Theta(B^S) = 1 - \Theta_1 B^S - \ldots - \Theta_Q B^{QS} \end{cases}$

As an another ARIMA's extensions, FARIMA is a generalization of the ARMA model customized to support those applications like NTP in which, besides short-term dependencies, there are considerable linear long-term dependencies between the observations. Unlike the ordinary ARIMA process, the difference parameter (d) in FARIMA model could take noninteger values.[30] The general FARIMA process is expressed in Equation (6) where B is the backshift operator.

$$\left(1 - \sum_{i=1}^{p}\phi_i B^i\right)(1-B)^d X_t = \left(1 + \sum_{i=1}^{q}\theta_i B^i\right)\varepsilon_t. \tag{6}$$

Generally, the family of ARIMA models is drawn on the assumption of time-series data stationarity; while, in dynamic environments such as IoT, network traffic with severe and intermittent fluctuations could lead to degrading model performance. Nonetheless, by using some transformations other than differentiation (eg, logarithms) to decrease nonstationarity in input data, this deficiency can be overcome to some extent.[23,31]

One of the other standard statistical models widely employed in time-series problems is that of Generalized Autoregressive Conditional Heteroskedasticity (GARCH).[32] This model is an extension of the Autoregressive Conditional Heteroskedasticity (ARCH) model innovated by Engle in 1982 to estimate the target variables' volatility.[33] The main goal is to model the changes in variance of target variables whose part of the total variance is conditioned on lagged values of target variance and model's residuals. To this end, the concept of Conditional Variance (also referred to as Conditional Volatility) plays a key role. Considering $\{\epsilon_t\}$ as a real-valued discrete-time stochastic process, as:

$$\epsilon_t = \sigma_t w_t,$$

where w_t is discrete white noise given i.i.d ($\mu = 0, \sigma^2 = 1$), the GARCH(p,q) process is then denoted by Equation (7).

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2, \tag{7}$$

where α_i and β_j are model's parameters meanwhile, to avoid negative variance the following constraints are imposed:[34,35]

$$\left\{p \geq 0, q > 0, \alpha_0 > 0, \alpha_i \geq 0, i = 1, \ldots, q, \beta_j \geq 0, j = 1, \ldots, p\right\}.$$

Moreover, parallel to applying jointly with some nonlinear approaches (eg, ARIMA), some hybrid models based on the ARIMA's foundation have been proposed. For instance, Fuzzy-AutoRegressive Integrated Moving Average (Fuzzy-ARIMA) is an invented method which fuzzifies ARIMA's parameters using the fuzzy regression method.[36] In terms of acronyms, as seen sporadically in some references, Fuzzy-ARIMA is also referred to as FARIMA; which should not be confused with the Fractional-ARIMA model. Meanwhile, The latter is recorded as ARFIMA in some sources as well.[37]

2.3 | ML techniques for NTP

In general, the logical framework of problems solvable by ML techniques can be formulated in four broad categories, namely, classification and regression, clustering, and rule extraction.[19] There are four ML paradigms in the same vein corresponding to the nature of the problem at hand, namely Supervised Learning (SL), Unsupervised, Semi-supervised, and Reinforcement Learning,[38] respectively. Each of these paradigms has its own different effects on proceeding data collection, ground truth creation, and feature engineering. Most of ML methods used in NTP are subtechniques of SL as the models need to be trained by historical data. SL uses labeled data (historical data) to build up the models employed in classification and regression problems, where predicting the outcomes in the form of discrete or continuous quantities is intended. In many real-world problems, access to labeled data is subject to constraints. In networking, most of data gathered from a network is unlabeled or semi-labeled.[39] In the lack of sufficient knowledge or the abundance of missing labels, the Semi-Supervised Learning (SSL) paradigm as a particular variant of SL exploiting different techniques such as Active Learning[40] can be leveraged.[6]

One of the critical aspects in ML is choosing the proper model from the mass of available algorithms and techniques. Different factors can be applied, for example, goals of applications, pros and cons of the operating environment regarding the deployments and applications of ML models, the learning method (ie, supervised or unsupervised), how to access data, etc.[41] Some of the most widely used models are presented in the following:

- *Neural Networks:* Artificial Neural networks (ANNs) are among the most potent and widely used ML techniques.[42] Thanks to the Activate Function, ANNs can learn complex nonlinear dependencies among numerous variables, thus they are generally known as Universal Function Approximators.[42] The general architecture of ANNs is a directed graph[43] consisting of input and output layers, which are connected via the so-called hidden layer, which itself could be consist of one or more layers. Input values reach the output layer by applying transformations through the hidden layers. The number of these layers is also referred to as the *depth* of the model. Based on the ANNs' depth, the idiom "Deep Neural Network" denotes ANNs constructed of two or more hidden layers, opposed to "Shallow Neural Network" referring to the traditional baseline ANNs.[42] Due to their flexible structure, Deep Neural Networks (DNN) have gained striking popularity in time-series prediction. In this context, Recurrent Neural Network (RNN), by allowing inputs to be recycled in hidden layers around the recurrent connections has been a leaping advance. Different RNN-based architectures can be defined as per adopted activation function and how the neurons connect to each other, namely that of Fully Recurrent Neural Network (FRNN), Bidirectional Neural Networks (BNN), stochastic neural networks, and the well-known paradigm Long short-term memory (LSTM).[44] LSTM as an extension of RNN is proposed to resolve the vulnerability of normal RNNs against the gradient exploding/vanishing problem caused by long-term dependencies.[45-47] LSTM with some innovations in its architecture including a triple gate mechanism to control inputs to cells, and a feedback loop for data retention, can learn long-term dependencies and remove invalid inputs that cause perturbation in cell's outputs.[6,48] In practice, an implemented LSTM model usually consists of a set of blocks where each block contains several LSTM cells.

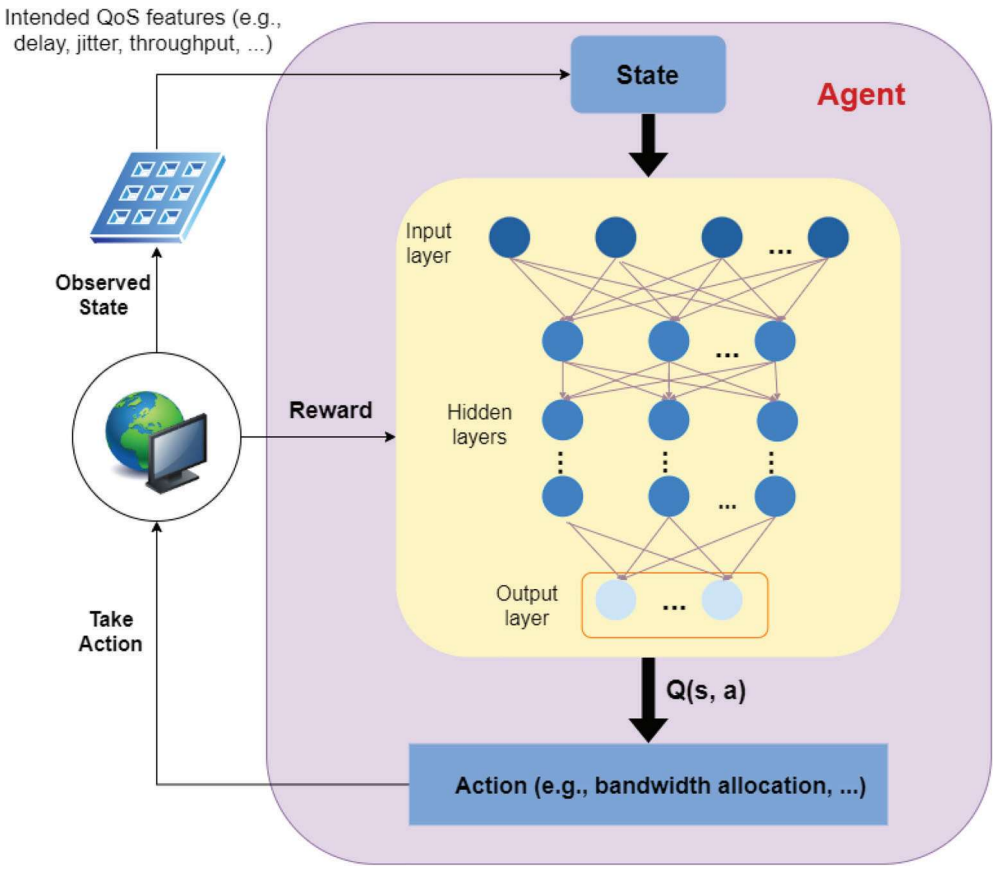


FIGURE 1 A schema that how deep reinforcement learning interacts with the network

Despite the unavoidable capabilities of deep learning-based methods, the slow training process of these models is a significant problem in their application in dynamic environments. Moreover, the lack of transparency in the learning process of these models is another limitation[49,50]

- *Reinforcement Learning:* The Q-Learning algorithm, along with Deep Reinforcement Learning (DRL) (which is actually a combination of Q-Learning and DNN), is one of two algorithms representing the Reinforcement Learning (RL) method.[8,12] As elaborated by Watkins,[51] the Q-Learning algorithm provides learner agents that can act optimally in Markovian environments relying on their knowledge stemmed from experiencing actions' consequences, without the need to mapping the environment.[51] It relies on a function called Q-function to learn the table containing all available state-action pairs and their long-term rewards.[52] In NTP, RL and its variants have a good potential to interact with the network to learn the network behavior and predict the behavior of the network in the future. As mentioned, RL can be integrated with DNN which can help to improve the performance of RL techniques. Figure 1 shows a schema of interacting DRL and the network regarding NTP.

Although the number of ML techniques used in NTP is not limited to the list above, DNN and RL are the most important ML techniques based on our literature review explained in Section 3.

2.3.1 | Data collection

Creating an efficient model for a problem is highly dependent on the availability of appropriate and unbiased representative data.[19] Due to the variety of data in different applications of networking as well as alternation of data over time, it is essential to adopt a suitable method for data collection for ML methods to train their models.

In networking, traffic data can be extracted through the packet inspection process by the DPI and SPI methods.[12] While the former is based on reading and, if necessary, analyzing the full packet contents that include application

headers and payload, the latter method examines only headers of the network and/or transport layers in those packets selected based on the target sampling strategies.[7] Nevertheless, apart from some specific applications of DPI (eg, filtering or troubleshooting), since this method imposes significant computational and memory overheads on the network, the real-time traffic-monitoring-oriented tasks would rely on SPI for extracting the required data from packet streams.[12,14]

Data collection is performed typically in offline or online manners. In the offline method, data are entirely used for training the model at once, and then, the model is deployed and used for operational data analysis. In the online method, throughout a continuous process, model training would launch in conjunction with deployment in the operating environment, and model knowledge is updated with new input data, which is received in a sequential order.[13,53] As SL, Unsupervised Learning (UL) and SSL are generally used in the offline learning setting, some of RL techniques, for example, State-action-reward-state-action (SARSA)[54] and incremental learning techniques[55] are mainly designed to update the model gradually based on new data (2). Training the model using collected data can be achieved in one of the batch or incremental (also known as streaming) ways depending on the situation and learning settings. In a batch setting, the collected data are divided into three subsets, including training, test, and validation (which in some cases is also called development set). The validation set is used when selecting the appropriate model, and its architecture is part of the process. Otherwise, it would not be needed. Determining the optimal values for the model parameters (eg, the weight of connections between neurons in a neural network [NN]) and evaluating the model's performance would be accomplished, respectively, using the training and test sets. On the other hand, in the incremental method, the data is streamed to the training model for various reasons (eg, large volume and infeasibility of loading at once, or gradual generation of data).[13,53] Moreover, in dynamic environments, especially online applications, the ML model must be continuously retrained. In such cases, to solve the concept drift problem, given the high computational cost of doing it from scratch, using only the new data for training is an efficient solution by incremental approaches.[12] Figure 2 shows different types of collecting data to train ML models in networking.

2.3.2 | Feature engineering

As one of the ML's pillars, feature engineering includes feature selection and feature extraction in addition to data cleansing and data preprocessing. Feature selection refers to selecting the effective discriminator features and removing the irrelevant and redundant features; whereas feature extraction involves extracting new extended features from existing ones. These procedures both would lead to diminishing data dimensionality and computational overhead and, consequently, increasing the model's efficiency and accuracy.[56]

In networking, features can be classified based on the granularity in three levels: packet-level, flow-level, and connection-level.[57] The finest level of granularity is packet-level features where the packet-related statistical data such as mean, root mean square (RMS), variance, as well as time-series information are extracted or derived from collected packets.[19] The independence of these features from the sampling method adopted to collect data is their key advantage.

Features such as mean flow duration, mean of packets per flow, and average flow size in bytes are observable at the flow-level. At the highest level of granularity, there are connection-level features that would be extracted from the transport layer. Throughput and advertised window size in TCP connection headers are examples of these features. Despite the high-quality information provided by connection-level features, the imposition of excessive computational overhead and high distortion in facing with sampling and routing asymmetries are among their drawbacks. Feature extraction is often performed using techniques such as Principal component analysis (PCA), entropy, and Fourier transform.[19] Table 1 shows a summary of some popular network features.

3 | REVIEW OF EXISTING WORK

Using ML techniques and statistical-based techniques in NTP is a well-established research area. Historically, one of the earliest works in applying ML in NTP belongs to Yu and Chen, carried out in 1993. They used multilayer perceptron (MLP) NN (MLP-NN) motivated by enhancing the accuracy over traditional AR methods.[19] Since then, many researchers have lent themselves to improve the ML-based solutions for predicting network behavior as accurately and timely as possible. However, in this survey we focus on the most important recent works in the field. To review the literature, we have used the method shown in Figure 3 to refine the related literature. The NTP-related literature is classified as below.

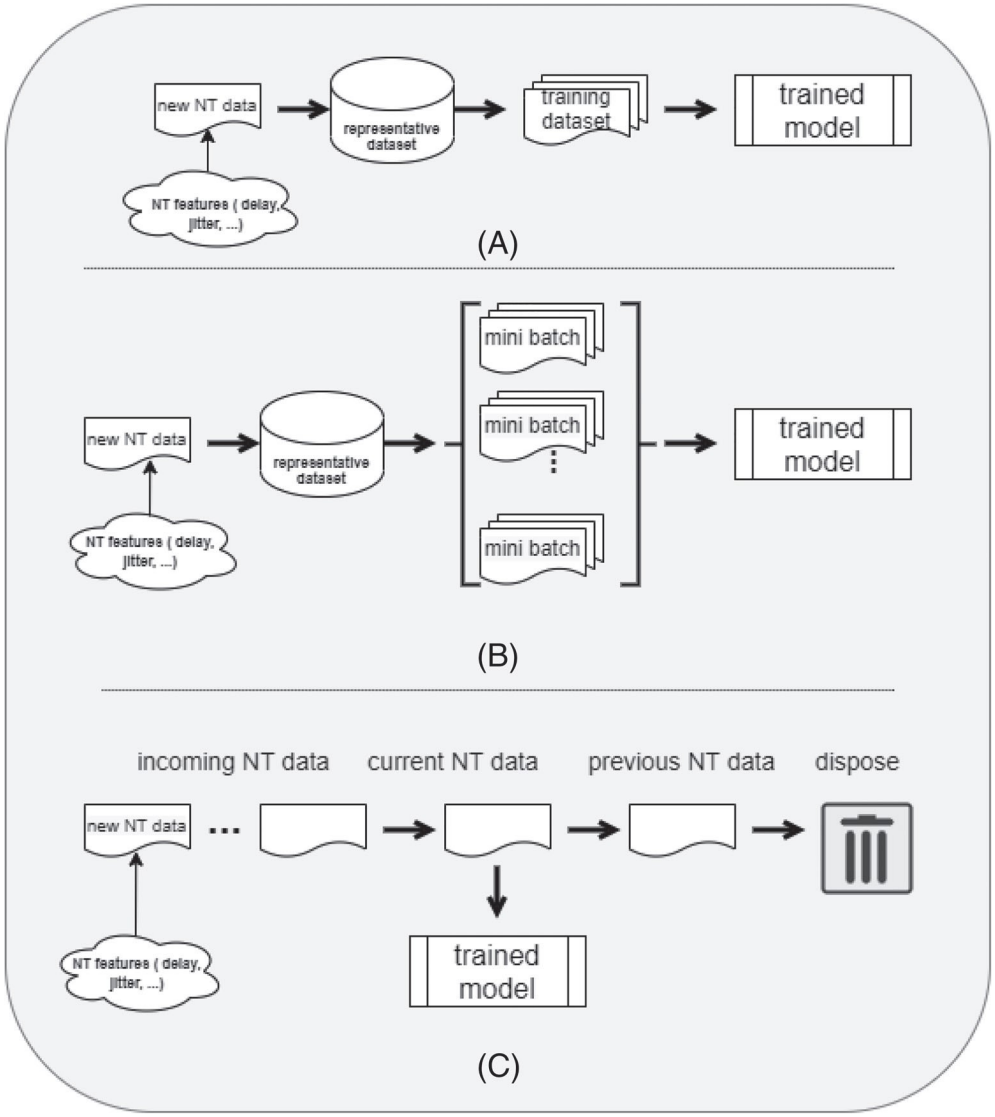


FIGURE 2 Data collection settings. (A) Batch Offline; (B) Batch incremental; (C) Online incremental (streaming)

TABLE 1 More popular features used in machine learning for networking

Feature granularity	Description and examples
Packet-level:	Source IP address
	Destination IP address
	Source host number
	Destination host number
	Frame length
	Frame number
	Packet inter-arrival time
Content-based:	HTTP protocol used to submit data from client to server
Time-based:	Number of frames received by a unique destination in the last T seconds from the same source
	Number of frames received by a unique source in the last *T* seconds from the same destination
Connection-based:	The number of packets flowing from source to destination
	The number of packets flowing from destination to source

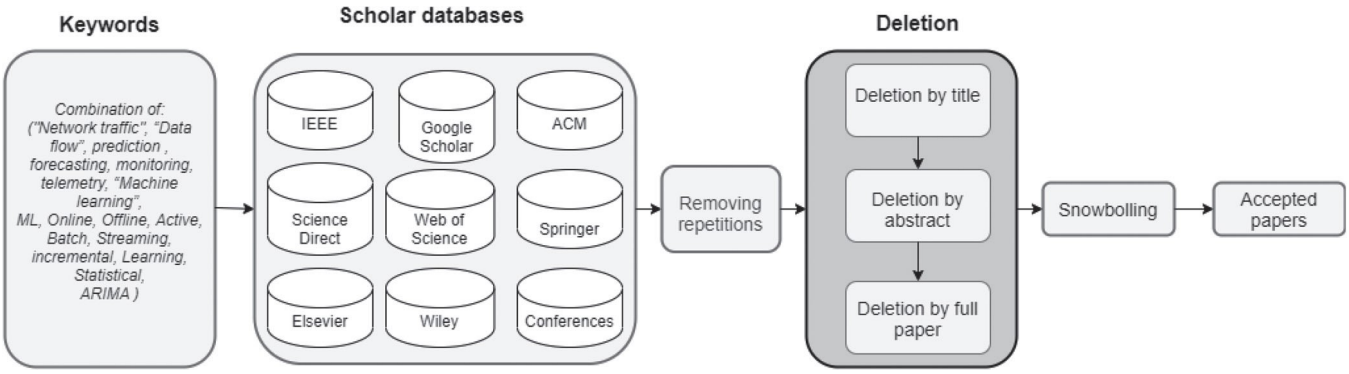


FIGURE 3 The method adopted to refine the literature[58]

3.1 | Pure ML-based NTP solutions

Evaluation of performance and efficiency of ML-based methods is a significant part of the literature. LSTM is one of the most widely used techniques in this field.

The survey by Alawe et al[59] aims at Access and Mobility Management Function (AMF) in the 5G network. To this end, two approaches, Feedforward neural network (FFNN) and LSTM are examined. The authors propose to use ML to forecast the arrival time of requests from User Equipment (UE), and consequently, the scale-out/in process. This avoids the rejection of requests and keeping the attach duration (how long do UE are connected to network resources) low. The results indicate that LSTM outperforms FFNN in terms of prediction accuracy. The dataset is classified into 10 different classes, based on the load and the number of AMFs needed. The first technique used for predicting the class of load of the next period is FFNN. The second technique tested for predicting the average load of the upcoming period of time is LSTM. Both networks are trained with the 60% dataset and then asked to predict the remaining 40%.The results indicate that LSTM outperforms FFNN in terms of prediction accuracy.

Trinh et al[46] have dedicated their work to present a network traffic prediction model in the LTE environment, using the LSTM algorithm. The LSTM network consisting of multiple unified LSTM units is applied to the raw mobile traffic data collected directly from the Physical Downlink Control CHannel (PDCCH) of LTE. By assuming NTP as a supervised multivariate problem, the proposed model aims at minimizing the prediction error with respect to the information extracted from the PDCCH. This data gathering methodology and the multistep structure adopted for the predictive network are emphasized as dedicated research aspects. According to comparison results, the composed model outperforms ARIMA and FFNN models.

Pruning internal connections between NN neurons to diminish computational cost constitutes the idea underlying the research by Hua et al.[47] Based on this, a heuristic architecture with sparse neural connections, called the Random Connectivity Long Short-Term Memory (RCLSTM) is introduced, in which the complete (one-to-one) connection between neural network neurons, as is in conventional LSTM, has given way to a random pattern of links between them. The simulated model consists of a three-layer stack RCLSTM.

Wang et al[60] puts forward a model that aims to improve cellular traffic prediction accuracy with limited real data and support data privacy. The model is called (ctGAN- S2S) and consists of a cellular traffic generative adversarial network and a sequence-to-sequence neural network, in which learning is fed by an arbitrary length of time-window from the historical time-series data of cellular traffic. The augmentation model generates close-to-real cellular traffic data; thus, eliminating the need for original data supports data protection and data privacy.

The main contribution of Vinayakumar et al[20] is to compare some nonlinear prediction methods, namely that of FFNN, RNN, identity recurrent neural network (IRNN), gated recurrent unit (GRU), and LSTM (generally referred to as RNN) in terms of performance in Traffic Matrix (TM) prediction in large networks under different experiments. The arranged experiments in test scenarios help to identify the optimal network parameters and network structure of RNN. The TM is passed by using the sliding window approach. The obtained results indicate the superiority of LSTM over other models; meanwhile, GRU has relatively imposed less computational cost.

In considerable number of researches, Gaussian Process Regression (GPR) is the undergrounding leveraged technique. The conducted research in Bayati et al[61] is one among those who exploit GPR in this context. The proposed solution is

based on the Direct (or Parallel) strategy for multiple-step-ahead traffic prediction in which the entire process is divided into H distinct models that are trained concurrently to conduct H-step-ahead time-series predicting. Each time-step forecasting is fed by the gained prediction in the previous time-step, as one of the input features. Subsequently, the prediction error at a time-step (or the uncertainty in the feature vector) is propagated through the forecasts at the next time-step. In order to tackle the error propagation, the paradigm has been investigated, indicating that the desired performance of a multiple-step-ahead prediction is strictly depended on, and could be influenced by the classification of data at higher levels.

Ensuring the QoS in the network in a way that simultaneously provides resource efficiency (bandwidth) requires an accurate real-time estimation of the network's future behavior. To this end, an online bandwidth allocation method based on GPR has been proposed by Kim and Hwang.[62] The theorem that a stationary increase in the size of a given process would lead to large-buffer asymptotics of the queue length process forms the basis of the proposed model for deriving the proper bandwidth.

A combined solution to deal with complex network data flows is presented by Wang et al.[63] The proposed solution consists of a preprocessing step and a 2-fold prediction process using LSTM and GPR models. In the initial phase, the data flow's dominant periodic features are extracted through Fourier analysis, and then the LSTM model is applied to the remaining small components. A complementary step adopting GPR is launched, estimating the residual components to improve the prediction's accuracy.

Poupart et al[64] elaborate utilizing ML techniques to approximate each network flow's size at its start, focused on detecting elephant flows independent of source application or end host. The proposed estimation method benefits the metadata exploited from the first few packets of a flow. In this regard, the authors explain three ML techniques, including Gaussian processes, Gaussian Mixture Model with Bayesian Moment Matching, and neural networks to predict the flow size based on existing historical data and online streaming data.

Mikaeil[65] proposes a method for near-time prediction of primary user (PU) channel state availability (ie, spectrum occupancy) in Cognitive Radio Networks (CRN) using Bayesian online learning (BOL). Given that the nature of the PUs channel state availability can also be considered a dual-states switching time series, the captured time-series representing the Pus channel state (ie, PU idle or PU occupied) are fed as an observations sequence into the BOL prediction algorithm.

The primary use of cardinality estimation algorithms in computer networks is counting the number of distinct flows. Cohen and Nezri[66] have investigated the application of flow cardinality estimation algorithms in SDN environments. their focus was on common deficiencies faced by sampling methods, especially in adapting to changes in flow size distribution. Further, they have introduced and elucidated a framework which benefits from online ML, and to achieve the best performance and accuracy, three popular linear regression ML algorithms, namely that of Stochastic Gradient Descent (SGD), Recursive Least Squares (RLS), and Passive-Aggressive (PA), have been examined.

Cui et al[67] discuss the Stochastic Online Learning (SOL) technique as a tailored model for that of environments like Mobile Edge Computing (MEC), which can experience time-varying, stochastic traffic arrivals without the Markov property. Among the multitude of articles, this paper is one of the few that addresses online learning from network behavior. Unlike the majority of ML methods that rely on learning from the training data, SOL learns from network changes by using SGD. This approach is aimed at establishing a trade-off between learning accuracy and learning time and increasing network throughput.

Zhang et al[68] present an incremental deep computation model for wireless big data feature learning in IoT. The model is constructed by stacking several incremental tensor auto-encoders (ITAE). To handle new arrival wireless samples, two types of ITAEs, based on the learning strategy, are developed, namely the parameter-based incremental learning algorithm (PI-TAE) and the structure-based incremental learning algorithm (SI-TAE). In facing new arrival samples, the proposed model only needs to load the new samples to the memory to update the parameters and structure, respectively, by PI-TAE and SI-TAE. This mechanism underlies this model's capability to deal with wireless big data for feature learning in real-time.

The cellular link bandwidth prediction in Long-Term Evolution (LTE) networks has been investigated by Yue et al.[69] The authors have approached the problem by analyzing the correlations between various lower-layer information and the link bandwidth. Thereupon, they propose a framework using a Random Forest-based prediction model, which through an offline data feeding process, and exploiting the intrinsic capabilities of Random Forest, identifies the most important features, and uses them to predict link bandwidth in real time.

Zhang et al[70] have introduced an approach using Convolutional Neural Network (CNN) to collectively model spatial and temporal dependence for cell traffic prediction. The key features of the proposed approach are considering the traffic

data as images, as well as utilizing a parametric matrix-based fusion method to estimate influence degrees of the spatial and temporal dependence.

In a study by Pfulb et al,[71] the problem of estimating the expected bit-rate of network flows based on their metadata has become a three-tier classification problem using a fully connected DNN with the ReLU activation function. Their approach embraces three stages including data collection, data preparation, and data processing. In this approach, DNN training is treated as a streaming problem by dividing the dataset into identical data blocks. The model is trained and tested in a semi-streaming fashion applied on all blocks one by one such that all intended preparations being performed block-wise.

To improve the self-management and active adjustment capabilities of base stations in wireless networks, by Li et al[72] the temporal and spatial correlation of traffic data is addressed. To this end, they have composed a deep network-based framework of network traffic prediction using CNN and LSTM, constituted of three main units by which the spatio-temporal correlation of wireless network traffic data can be captured effectively.

The first part of Table 2 shows the list of the reviewed literature in this category.

3.2 | Pure statistical-based NTP solutions

The following articles mainly deal with the problem using statistical techniques such as ARIMA family and their innovative combinations.

The conducted research by Tran et al[73] is a study on different varieties of exponential smoothing method namely single, double, Holt-Winters No Seasonal, and Holt-Winters Additive Seasonal (that are categorized as simple exponential smoothing methods [S-ESM]), and Error, Trend, Seasonal exponential smoothing method (ETS-ESM) in terms of the effect of their smoothing factors on accuracy of short-term NTP, as well as their suitability for dealing with voice and data over the cellular network. The results show that HWMS outperform other methods.

Rafsanjani et al[22] proposed a method called *QARIMA* as an extension of ARIMA to predict the queues in packet forwarders in networks. Their proposed method determine the last trend of the behavior of the queue and make a linear regression model to predict. They showed that their proposed method can predict the delay of packets with a high accuracy. Shahraki et al[74] integrated and extended the ideas proposed in References 75 and 76 to propose an online network behavior analysis technique for IoT networks. The proposed technique analyses the data gathered from IoT networks to determine and predict important changes that can cause inefficiencies.

Mehdi et al[36] have incorporated fuzzy regression and ARIMA models (Fuzzy-ARIMA) in promising to take both methods' advantages. Besides, to perform real-time predictions based on historical data, they have adopted a sliding window technique called SOFA ,which reduces the effect of input data fluctuations over time.

In promising to allocate bandwidth efficiently in SDNs, Bouzidi et al,[77] introduce a heuristic rules placement algorithm in which an Online-Learning module, designated upon the linear regression, is utilized for the network delay prediction. The overall framework of the proposed solution encompasses the formulation of the flow rules placement as an Integer Linear Program (ILP) targeting to minimize the total network delay, And finally, to solve the defined ILP problem through an excogitated algorithm, that leads to reducing the time complexity and enhancing the estimation accuracy.

The second part of Table 2 shows the list of the reviewed literature in this category.

3.3 | Hybrid NTP solutions

In addition to the pure statistical- and ML- based NTP techniques, there are some proposed models that use both of these approaches simultaneously called Hybrid solutions.

Xu et al[50] have presented an extension of Reference 78, proposing an architecture for traffic prediction in the Cloud radio access network (C-RAN) with the distributed remote radio heads (RRHs) and the centralized Baseband Units BBUs pool bearing lots of parallel Baseband Unit (BBU)s. In such an architecture, an alternating direction method of multipliers (ADMM) and the cross-validation empowered Gaussian Process (GP) framework is performed in which, the parallel BBUs' contribute in training process and the local predictions would be incorporated altogether via cross-validation to create the final prediction. retaining the trade-off between accuracy and time consumption, and scalability are the two focal claims in this method. Bouzidi et al,[79] have composed an extension to Reference 77 with a different experimental

TABLE 2 Comparing existing works on Network Traffic Prediction (NTP)

References	Network	Purpose	Tr.	Approach	Description
Pure ML-based NTP solutions					
59	5G	Access and mobility management	f / b	LSTM and FFNN	Comparison of LSTM and FFNN in predicting network traffic state.
46	LTE	Optimal resource allocation	f / b	LSTM	To Predict network traffic by a multi-step LSTM model, using the raw data collected from physical channel
47	Wireless Network	Reducing the computing cost of LSTM for NTP	f / b	LSTM	Computational-cost-reduction by means of reconstructing LSTM network in form of a random sparse graph
60	5G	Data privacy protection-aware and accurate prediction with few real data	f / s	LSTM	A NTP model called ctGAN-S2S, based on data augmentation.
20	Cellular Network	Traffic matrix estimation	f / s	LSTM, FFNN, RNN, IRNN, GRU	Analyzing the performance of various RNNs in forecasting TM
61	High-speed networks	Addressing the error propagation	f / b	GPR	Multiple-step-ahead traffic prediction by using a GPR framework which models the traffic at different timescales
62	Internet	QoS, and resource efficiency	n / s	GPR	Online bandwidth allocation method based on GPR
63	Cellular Network	Improving the prediction performance	f / b	LSTM, GP	Two LSTM and GPR modules jointly are adopted to improve the prediction performance
64	Cellular Network	Flow size estimation	f / b	GPR, NN	Estimating the flow size and separating elephant and mice flows using online ML techniques based on historical data as well as online streaming data under hybrid learning settings.
65	CRN	Predicting the near future of spectrum occupancy	n / s	BOL	Modeling the PU channel state indicator sequence as a time series, of which changes are predicted in advance by the proposed BOL model
66	SDN- and NFV-based networks	Cardinality estimation in the field of network measurement	n / b	SGD , RLS, and PA	A sampling-based framework for adaptive cardinality estimation utilizing sampling and online learning
67	Mobile edge network	Minimizing the time-averaged operational cost of MEC	n / s	SGD	A SGD -based faramework for online learning from the changes of the network without needing training data
68	IoT	Real-time feature learning	n / s	ITAE	An incremental deep computation model consisting of two incremental tensor auto-encoders, that is, parameter-based and structure-based incremental learning algorithms
69	LTE	Real-time bandwidth prediction	f / b	Random Forest	A ML framework (called LinkForecast) is developed that utilizes both past throughput and lower-layer information to predict link bandwidth in real time.
70	Cellular network	Modeling the nonlinear dynamics of wireless traffic	f / b	CNN	Treating traffic data as images, and applying the influence degrees of lagged spatial and temporal data to the prediction

(Continues)

TABLE 2 (Continued)

References	Network	Purpose	Tr.	Approach	Description
72	Cellular network	Spatiotemporal modeling and prediction of network traffic	f / s	CNN, LSTM	A compound architecture to extract and applying the spatiotemporal correlation of network data using CNN multi-layer stack, LSTM -based temporal information processing unit, and an attention module
71	Cellular network	Flow size estimation	f / b	DNN	Altering the regression problem of estimating flows bit-rate values to a three-level classifying task, under a semi-streaming fashion setting.
Pure statistical-based NTP solutions					
73	Cellular network	investigating the smoothing factor impact	f / b	S-ESM and ETS-ESM methods	Different varieties of exponential smoothing methods are investigated based on their smoothing factor impact on NTP accuracy.
22	TCP/IP	QoS-aware routing	f / s	ARIMA	The delay of packets is predicted based on the last trend extracted from the queue's behavior on the router
74	Ad hoc IoT networks	Trend change detection	n / s	Online data analysis by dynamic-sliding window	A statistical light-weight Trend Change Detection (TCD) method in an online manner
75	Ad hoc IoT networks	Trend change detection	f / s	Data analysis	Recognizing majore trend's change points in a Poisson distribution dataset through a recursive matrix-based model
76	IoT	Contextual outliers detection	f / s	Data analysis	Fragmenting time-series dataset into segments based on trend change points to discover the contextual outliers
36	Cloud network	Reducing dpendence on historical data	f / s	Fuzzy-ARIMA	The statistical Fuzzy-ARIMA model conducts short-term traffic prediction in real time, based on the historical data blocks provided by a sliding window
77	SDN-based network	QoS-aware routing	n / s	Linear regression	Applying linear regression-based online learning for dynamically predict the latency, in order to updating the flow rules in network devices
Hybrid NTP Solutions					
50	C-RAN	Large-scale NTP in a cost-efficient manner	f / s	GP and ADMM	A C-RAN-based architecture on which, an ADMM and cross-validation empowered scalable GP framework could be performed.
79	SDN-based networks	QoS-aware routing	n / s	LSTM and DRL	Forecasting the future network traffic and detecting the optimal route, dynamically by using a DRL agent.
80	IIoT	Short-term traffic prediction in Real-time	f / s	RL, MCL,Q-learning, and KL	Real-time prediction of short-term time-varying features of network traffic via a RL approach, as well as adopting a dictionary learning algorithm to reduce the complexity of the proposed model
81	Cellular network	forecasting network-wide traffic behavior	f / b	CNN, LSTM, and DWT	Temporal decomposition of the original TM via a wavelet transform, as well as extracting spatial patterns and long-term temporal features correspondingly by CNN, and an input sequence aware LSTM.

(Continues)

TABLE 2 (Continued)

References	Network	Purpose	Tr.	Approach	Description
82	Cellular network	To predict cell-station traffic volumes	f / b	K-means clustering, Wavelet Tramsform, and RNN (Elman NN)	Decomposing the clustered network traffic flows into high and low frequencies to facilitate the prediction process.
83	Cellular network	To improve network traffic prediction by using sequence mining	f / b	LSTM , ANFIS, and FCM	Accelerating the performance of prediction models by using preprocessing and clustering the training data
Comparative works					
78	Cellular networks	Comparison in terms of complexity and accuracy	f / s	GP, SARIMA, and sinusoid superposition	In a 2-fold process, the kernel is designed based on captured traffic patterns. As well as leveraging the Toeplitz structure of the covariance matrix, the computational complexity of hyperparameter learning is decreased.
84	Cellular network	Long- and short-term traffic forecasting	f / b	Auto regression, NN, and GP	Traffic pattern forecasting, based on big data, ML, and network KPIs
85	Wireless network	Decreasing the training overhead without dilution of accuracy	f / b	LSTM , SARIMA	Studying the impact of the training-related factors on accuracy and overhead through camparison of two approaches
43	LTE-A	Comparison of approaches in terms of accuracy and learning time	f / b	LSTM, ARIMA, FFNN	The survey highlights outperforming of proposed LSTM model to FFNN and ARIMA in terms of prediction performance and training time efficiency
86	Wireless network	Real-time NTP	f / b	LV, MA, DES, AR, ARMA, ANN, and Wavelet	A comprehensive comparison, and analyzing three main classes of predictors from various aspects
87	Cellular network	Forecasting TM in large networks	f / s	LSTM and ARIMA	Model parameters optimization to forecast large scale TMs
27	Internet	Comparing various usage schemes of models	f / b	FARIMA, FARIMA/GARCH, and NN	Comparison between distict schemes of applying composed approaches to dynamic bandwidth allocation task
88	Internet	Comparing various usage schemes of models	f / b	FARIMA, NN	FARIMA and MLP models are used in a framework to compare different application methods of traffic prediction, including individual, hybrid, and selective schemes

Abbreviations: ADMM, alternating direction method of multipliers; AR, autoregressive; ARMA, autoregression moving average; ARIMA, Autoregressive integrated moving average; BOL, Bayesian online learning; CNN, convolutional neural network; CRN, cognitive radio networks; DES, double exponential smoothing; DRL, deep reinforcement learning; DNN, Deep Neural Networks; DWT, discrete wavelet transform; ETS-ESM, error, trend, seasonal exponential smoothing method; FARIMA, fractional autoregressive integrated moving average; FCM, fuzzy-C-means; FFNN, feedforward neural network; GARCH, generalized autoregressive conditional heteroskedasticity; GPR, Gaussian process regression; GRU, gated recurrent unit; IoT, Internet of Things; IRNN, identity recurrent neural network; ITAE, incremental tensor auto-encoders; KL, Kullback-Leibler; LSTM, long short-term memory; LV, last value; MV, moving average; NFV, network function virtualization; PA, passive-aggressive; QoS, quality of service; RL, reinforcement learning; RLS, recursive least squares; RNN, recurrent neural network; S-ESM, simple exponential smoothing methods; SARIMA, Seasonal-ARIMA; SDN, software-defined network; SGD, stochastic gradient descent; Tr., Training: Online *(n)* or Offline *(f)* / Batch *(b)* or Stream *(s)*

setting in terms of the adopted learning algorithm, where the Linear Regression model has given way to an LSTM-based model.

Nie et al[80] have modeled the NTP problem in the Intelligent Internet of Things (IIoT) ecosystem as a Markov decision process. In promising to extract short-term time-varying features of network traffic in real-time, and aiming at minimizing the training data size, it proposes a RL-based approach consisting of Monte-Carlo learning (MCL), Q-learning, and Kullback-Leibler (KL) divergence. Moreover, to deal with degrading impact of the vast state space of

Monte-Carlo-Q-learning in IIoT ecosystems, a greedy adaptive dictionary learning algorithm is proposed that reduces the computational complexity. To discover future network-wide traffic behavior, Zhao et al[81] introduce a TM prediction method, coined WSTNet, a complementary combination of CNN and LSTM, and utilizing Discrete Wavelet Transform (DWT) as a feature engineering tool. The method is comprised of three phases. First, at the preprocessing step using DWT, the original TM series is decomposed into multilevel time-frequency subseries at various timescales; second, to draw out the spatial patterns of traffic flows between endpoints, CNN without pooling is leveraged. Finally, the LSTM with self-attention technique is adopted to extract the TM series' long-term temporal dependencies.

Zang et al[82] have composed K-means clustering, Wavelet decomposition, and Elman neural network (ENN) in a framework to predict cell-station traffic volumes by using the spatial-temporal information of cellular traffic flow. After clustering the multiple BS traffic flows, the integrated time series are decomposed into high and low frequencies through Wavelet transform. This creates new subdivisions of data with higher stability and more tractable features facilitating the prediction process.

The main idea of Aldhyani et al[83] is to make the adopted ML models robust by using specific techniques. It proposes a 2-fold process to improve the performance of LSTM and Adaptive neuro fuzzy inference system (ANFIS) models in predicting network traffic. In the first step, the data is preprocessed by using the weighted exponential smoothing model then, the historical data is clustered by non-crisp Fuzzy-C-Means (FCM). The presented results show that identifying and classifying existing patterns of data can improve the performance of LSTM and ANFIS models in forecasting the network traffic behavior.

The third part of Table 2 shows the list of the reviewed literature in this category.

3.4 | Comparative works

Finding a good prediction solution by comparing the characteristics, shortcomings, and strengths of statistical- and ML-based methods has formed one of the areas of interest in researches related to this discipline.

Xu et al[78] have established a wireless traffic prediction model by applying the GP method based on real 4G traffic data. Observation of network traffic to capture periodic trend and dynamic deviations of data, as well as leveraging the Toeplitz structure in the covariance matrix to reduce the computational complexity of hyperparameter learning are the two pillars of the proposed method. The proposed model claims a significant reduction in computational complexity, and high accuracy.

In order to highlight the magnitude of the impact of network Key Performance Indicators (KPIs) on prediction accuracy, a comparative research by Le et al[84] has been conducted comprising two different settings. The first one addresses the time-series traffic forecasting mainly drawn on exploiting the traffic's historical traces, and the second involves with traffic's KPIs prediction through analyzing the relationship between KPIs and future patterns of network traffic. In this analysis, the two main criteria considered are Mutual Information (MI) and Relative Mutual Information (RMI). The performance of three algorithms, including GP, ANN, and AR, has been examined in both settings, reflecting a better performance and accuracy from the GP.

The research by Soheil et al,[85] beyond a solely theoretical basis, addresses some practical aspects of training and deployment of ML models for predicting real-world network data streams in telemetry systems, issues that challenge applying an ML model and achieving the expected accuracy. To this end, various training-related aspects, including the volume, freshness, and selection of training data, have been examined to show their impact on the accuracy and overhead (and thus feasibility) of both adopted models, namely LSTM and SARIMA. Further, utilizing separate models for different segments of a data stream is explored as well. Drawn on the achieved results, the article concludes that any network modeling often needs to be customized based on its target application for a specific data stream from a particular network.

Jaffry[43] introduces a LSTM-based model for traffic prediction in the Long Term Evolution-Advanced (LTE-A) network. It has been compared with similar paradigms based on ARIMA and FFNN in terms of performance and accuracy. The results of this study indicate the superiority of LSTM over FFNN and ARIMA. In addition, the efficiency of LSTM in working with small amounts of training data is another advantage mentioned for this model.

A comprehensive comparison between three predictor classes, focusing on considering some initiative criteria, is proceeded by Faisal et al.[86] The compared techniques are Last Value (LV), Windowed MA, Double Exponential Smoothing (DES), AR, ARMA, ANN-based predictors, and wavelet-based predictors. Besides considering the accuracy, as seen in other similar studies, this study also investigates overhead in terms of both computation cost and power consumption. Further, employing an initiative synthesized metric called Error Energy Score (EE-Score), accuracy and energy consumption

have been combined into a single global performance score for comparing predictors. Among the article's conclusions, this point is noteworthy that contextual conditions (ie, network characteristics) determine the proper predictor.

Azzouni and Pujolle[87] have suggested an LSTM framework for predicting TM in large networks. The potency of RNNs for sequence modeling tasks stems from their architectural characteristics. Having a cyclic connection over time enables them to store the activation from each time step in an internal state indicator, which forms a temporal memory. To establish continuous data feeding and learning required for real-time prediction of the current traffic vector Xt, a sliding window technique has been adopted which provides a fixed number of previous time-slots to learn of from. In Reference 27 by Chriskatris and Daskalaki, under different application strategies, including single, hybrid, selective, and combined, three models namely FARIMA, FARIMA/GARCH, and NN and their compositions have been applied to some schemes of dynamic bandwidth allocation for video transmission through network. The comparison outcomes highlight the predictive capability and cost effectiveness of a hybrid model consisted of FARIMA/GARCH, and NN.

In Reference 88 by Chriskatris and Daskalaki, individual, hybrid, and selective usage schemes of models for composing prediction approaches are examined. To this end, FARIMA and MLP models and their hybridization are used, while in the selective scheme, swapping between FARIMA and NN models is proceeded based on the White NN test considering the nonlinearity of traffic data. Outcomes indicate the outperforming of the hybris method.

The last part of Table 2 shows the list of reviewed literature in this category by comparing them in different aspects, for example, network type, purpose and approach.

4 | CHALLENGES OF EXISTING SOLUTIONS IN NTP

Given the inherent complexity of working with big data, especially in network environments, there are many challenging issues that need further investigation. In this section, some open challenges in the field of NTP will be highlighted.

- **Lack of specialized theoretical framework:** The ML methods used in NTP address prediction needs in the general context of time-series-based problem solving. However, in practice, the impact of numerous environmental factors, such as topology, speed, heterogeneity, etc., on the performance of ML models are very significant. In this regard, establishing a dedicated and promising theoretical framework for the application of ML in computer networks is one of the basic needs in this area.
- **Ground truth:** ML techniques are extremely correlated to what is known as Ground-Truth (GT). GT reflects the abundance of labeled data as well as the validity of the labels assigned to the observations. In the main applications of ML, regression, and classification, the GT extracted from the training data is the basis for comparison and inference to estimate the new data labels and, in data categorization, it is used to evaluate the accuracy of model performance. One of the main challenges involved, especially in dynamic environments such as network traffic, is the scarcity of labeled data or delays in accessing labeled data; a situation in which the semi-supervised strategy becomes relevant. PCA techniques, labeling manually or synthetically, or Active-Learning techniques are among the measures that could be taken to overcome the problem.
- **Representative datasets:** Shortage of public and representative datasets to train ML models is a fundamental challenge network-related prediction scenarios.[89] Due to the data-driven nature of ML solutions, lack of proper representative data or poor quality can lead to reduced accuracy. On the other hand, gaining access to required representative data is often subject of strict constrains.[12]
- **Accuracy versus speed:** Since increasing accuracy of ML techniques may lead to higher computational load in some cases as well as time, any improvement in accuracy will come at the cost of slowing down the training of the ML models and vice versa. Finding the equilibrium point between these two parameters and using hybrid methods for NTP while the model is being (re)-trained is recommended in this regard.[12]
- **Retraining:** In dynamic environments, due to constant changes, model retraining is essential to maintain validity. Unlike static environments with a limited specific dataset, allowing model retraining at any time and retraining dynamic models involve complex technical problems. Even if available in terms of access to previous data, the task will not be feasible with conventional static methods due to high computational cost. In online environments such as computer networks, as the acceleration of change increases, the barriers to model retraining become more complex. One common solution is to use some incremental methods for retraining in which the model is only updated with new observations.

- **The *Theory of Networks* challenge**: Introduced by David Meyer in 2017,[12] this concept explains that the trained models by public datasets cannot be used in real-world applications. In this case, each ML model should be trained separately for each network in a real-world application. Considering this challenge, ML methods with very high training time are not appropriate to use in NTP models as the network can not be left unattended for a long period of time.
- **Deployment issues:** In modern prevalent frameworks for software development and support, such as DevOps and DevSecOps, the software life cycle in its traditional form has been fundamentally transformed and influenced by concepts such as CI/CD, delivery flow, workflow, etc. These frameworks would impose specific rules, automations, and restrictions to the operating environment regarding testing and deployment of software updates and changes.[90] As such, deploying online-generic NTP models that during test phase need to access to real-world network data is an important challenge. In such environments, the criteria for evaluation of new updates and deployment licensing is subject to passing a comprehensive test set to ensure that each component and the whole end-to-end workflow is not degraded through new deployments. Due to the dynamic nature of live streaming data and the data-driven structure of online-generic NTP approaches, gaining accurate and reliable results through testing, if not impossible, but is notoriously difficult.[91] Although the classical AutoML accelerates the development of ML solutions, it does not address those issues concerned in ML systems' deployment. Auto-Adaptive Machine Learning (AAML) is a notion composed by[91] to respond to this problem. It proposes an architecture for ML systems lying in the bed of continual learning concept, aiming to address the aforementioned challenge by continuous update of deployed models.

5 | CONCLUSION

Network Traffic Prediction is considered as an important solution to address the various challenges in networking, for example, resource provisioning, congestion detection, and fault tolerance. The field of NTP is still in infancy as different models and techniques are used to predict the network traffic depending on network's type and purpose. Although ML techniques are widely used for NTP, there are some challenges that should be addressed to improve their efficiency in case of NTP, for example high computational cost, training, and retraining difficulties, Intrinsic large volatility of data, etc. On the other hand, statistical-based techniques are used traditionally for network traffic prediction; however they suffer from various challenges, for example, the excessive volume of needed historical data and their inherent constraints to deal with nonstationary time-series. In this paper, we have surveyed both ML- and statistical-based techniques for NTP. Our study shows that most existing solutions are not efficient in real-world applications due to the special characteristics of network data processing in modern networking technologies such as Theory of Networks, retraining challenge and ground truth.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

REFERENCES

1. Shahraki A, Abbasi M, de Piran M, Chen M, Cui S. A comprehensive survey on 6g networks: applications, core services, enabling technologies, and future challenges; 2021. arXiv preprint arXiv:2101.12475.
2. Shahraki A, Taherkordi A, Haugen Ø, Eliassen F. A survey and future directions on clustering: from WSNs to IoT and modern networking paradigms. *IEEE Trans Netw Serv Manag*. 2020;18(2):2242-2274.
3. Shahraki A, Kaffash DK, Haugen O. A review on the effects of IoT and smart cities technologies on urbanism. Proceedings of the 2018 South-Eastern European Design Automation, Computer Engineering, Computer Networks and Society Media Conference (SEEDA_CECNSM); 2018:1-8.
4. Horwitz L. The future of IoT miniguide: the burgeoning IoT market continues Technical report. CISCO, San Jose, CA; 2019.
5. Shahraki A, Geitle M, Haugen Ø. A comparative node evaluation model for highly heterogeneous massive-scale Internet of Things-mist networks. *Trans Emerg Telecommun Technol*. 2020;31(12):e3924.
6. Mohammadi M, Al-Fuqaha A, Sorour S, Guizani M. Deep learning for IoT big data and streaming analytics: a survey. *IEEE Commun Surv Tutor*. 2018;20(4):2923-2960.
7. D'Alconzo A, Drago I, Morichetta A, Mellia M, Casas P. A survey on big data for network traffic monitoring and analysis. *IEEE Trans Netw Serv Manag*. 2019;16(3):800-813. doi:10.1109/TNSM.2019.2933358
8. Abbasi M, Shahraki A, Taherkordi A. Deep learning for network traffic monitoring and analysis (NTMA): a survey. *Comput Commun*. 2021;170:19-41.
9. Kong F, Li J, Jiang B, Zhang T, Song H. Big data-driven machine learning-enabled traffic flow prediction. *Trans Emerg Telecommun Technol*. 2019;30(9):e3482.

10. Shahraki A, Abbasi M, Taherkordi A, Kaosar M. Internet traffic classification using an ensemble of deep convolutional neural networks. Proceedings of the 4th FlexNets Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility; 2021:38-43; IEEE.
11. Abbasi M, Shahraki A, Piran MJ, Taherkordi A. Deep reinforcement learning for QoS provisioning at the MAC layer: a survey. *Eng Appl Artif Intell*. 2021;102:104234.
12. Ayoubi S, Limam N, Salahuddin MA, et al. Machine learning for cognitive network management. *IEEE Commun Mag*. 2018;56(1):158-165. doi:10.1109/MCOM.2018.1700560
13. Aroussi S, Mellouk A. Survey on machine learning-based QoE-QoS correlation models. Proceedings of the 2014 International Conference on Computing, Management and Telecommunications, ComManTel 2014:200-204; IEEE.
14. Bui N, Cesana M, Hosseini SA, Liao Q, Malanchini I, Widmer J. A survey of anticipatory mobile networking: context-based classification, prediction methodologies, and optimization techniques. *IEEE Commun Surv Tutor*. 2017;19(3):1790-1821.
15. Jiang M, Wu C-M, Zhang M, Hu D-m. Research on the comparison of time series models for network traffic prediction. *Acta Electron Sin*. 2009;37(11):2353-2358.
16. Oliveira TP, Barbar JS, Soares AS. Computer network traffic prediction: a comparison between traditional and deep learning neural networks. *Int J Big Data Intell*. 2016;3(1):28-37.
17. Yin X, Wu G, Wei J, Shen Y, Qi H, Yin B. A comprehensive survey on traffic prediction; 2020. arXiv preprint arXiv:2004.08555.
18. Malik S, Ullah I, Kim DH, Lee KT. Heuristic and statistical prediction algorithms survey for smart environments. *J Inf Process Syst*. 2020;16(5):1196-1213.
19. Boutaba R, Salahuddin MA, Limam N, et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *J Internet Serv Appl*. 2018;9(1):16. doi:10.1186/s13174-018-0087-2
20. Vinayakumar R, Soman KP, Poornachandran P. Applying deep learning approaches for network traffic prediction. Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI); 2017:2353-2358; IEEE.
21. Nikravesh AY, Ajila SA, Lung CH, Ding W. Mobile network traffic prediction using MLP, MLPWD, and SVM. Proceedings of the 2016 IEEE International Congress on Big Data (BigData Congress); 2016:402-409; IEEE.
22. Rafsanjani MK, Rezaei A, Shahraki A, Saeid AB. QARIMA: a new approach to prediction in queue theory. *Appl Math Comput*. 2014;244:514-525.
23. Montgomery DC, Jennings CL, Kulahci M. *Introduction to Time Series Analysis and Forecasting*. Hoboken, NJ: John Wiley & Sons; 2015.
24. Krishnaswamy N, Kiran M, Singh K, Mohammed B. Data-driven learning to predict wan network traffic. Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics; 2020:11-18; ACM, New York, NY.
25. Medhn S, Seifu B, Salem A, Hailemariam D. Mobile data traffic forecasting in UMTS networks based on SARIMA model: the case of Addis Ababa, Ethiopia. 2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017; 2017:285-290. 10.1109/AFRCON.2017.8095496
26. AsSadhan B, Zeb K, Al-Muhtadi J, Alshebeili S. Anomaly detection based on LRD behavior analysis of decomposed control and data planes network traffic using SOSS and FARIMA models. *IEEE Access*. 2017;5:13501-13519. doi:10.1109/ACCESS.2017.2689001
27. Katris C, Daskalaki S. Dynamic bandwidth allocation for video traffic using FARIMA-based forecasting models. *J Netw Syst Manag*. 2018;27(1):39-65. doi:10.1007/S10922-018-9456-1
28. Dengen N. Comparison of SARIMA, NARX and BPNN models in forecasting time series data of network traffic. Proceedings of the 2016 2nd International Conference on Science in Information Technology (ICSITech); 2016:264-269.
29. Hong WC. Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting. *Neural Comput Appl*. 2012;21(3):583-593.
30. Biernacki A. Improving quality of adaptive video by traffic prediction with (F)ARIMA models. *J Commun Netw*. 2017;19(5):521-530. doi:10.1109/JCN.2017.000083
31. Nielsen A. *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. Sebastopol, CA: O'Reilly Media, Inc.; 2019.
32. Andrysiak T, Saganowski L, Maszewski M, Marchewka A. Detection of network attacks using hybrid ARIMA-GARCH model. *Adv Intell Syst Comput*. 2018;582:1-12.
33. Engle R. GARCH 101: the use of ARCH/GARCH models in applied econometrics. *J Econ Perspect*. 2001;15:157-168. doi:10.1257/JEP.15.4.157
34. Bollerslev T. Generalized autoregressive conditional heteroskedasticity. *J Econ*. 1986;31(3):307-327. doi:10.1016/0304-4076(86)90063-1
35. Chen H, Zhang J, Tao Y, Tan F. Asymmetric GARCH type models for asymmetric volatility characteristics analysis and wind power forecasting. *Protect Control Modern Power Syst*. 2019;4:1-11. doi:10.1186/S41601-019-0146-0
36. Mehdi H, Pooranian Z, Vinueza Naranjo PG. Cloud traffic prediction based on fuzzy ARIMA model with low dependence on historical data. *Trans Emerg Telecommun Technol*. 2019;e3731. doi:10.1002/ett.3731
37. Hajirahimi Z, Khashei M. Hybrid structures in time series modeling and forecasting: a review. *Eng Appl Artif Intell*. 2019;86:83-106. doi:10.1016/j.engappai.2019.08.018
38. Le Duc T, Leiva RG, Casari P, Östberg PO. Machine learning methods for reliable resource provisioning in edge-cloud computing: a survey. *ACM Comput Surv*. 2019;52(5):1–39. doi:10.1145/3341145
39. Alsheikh MA, Niyato D, Lin S, Tan H-P, Han Z. Mobile big data analytics using deep learning and apache spark. *IEEE Netw*. 2016;30(3):22-29.
40. Shahraki A, Abbasi M, Taherkordi A, Jurcut AD. Active learning for network traffic classification: a technical study. *IEEE Trans Cognit Commun Netw*. 2021.

41. Yousefpour A, Fung C, Nguyen T, et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J Syst Archit*. 2019;98:289-330.
42. Park J, Samarakoon S, Bennis M, Debbah M. Wireless network intelligence at the edge. *Proc IEEE*. 2019;107(11):2204-2239. doi:10.1109/JPROC.2019.2941458
43. Jaffry S. Cellular traffic prediction with recurrent neural network; 2020. arXiv preprint arXiv:2003.02807.
44. Chen M, Challita U, Saad W, Yin C, Debbah M. Artificial neural networks-based machine learning for wireless networks: a tutorial. *IEEE Commun Surv Tutor*. 2019;21(4):3039-3071. doi:10.1109/COMST.2019.2926625
45. Zhang X, You J. A gated dilated causal convolution based encoder-decoder for network traffic forecasting. *IEEE Access*. 2020;8:6087-6097. doi:10.1109/ACCESS.2019.2963449
46. Trinh HD, Giupponi L, Dini P. Mobile traffic prediction from raw data using LSTM networks. Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC; 2018:1827-1832.
47. Hua Y, Zhao Z, Li R, Chen X, Liu Z, Zhang H. Deep learning with long short-term memory for time series prediction. *IEEE Commun Mag*. 2019;57(6):114-119. doi:10.1109/MCOM.2019.1800155
48. He Q, Moayyedi A, Dan G, Koudouridis GP, Tengkvist P. A meta-learning scheme for adaptive short-term network traffic prediction. *IEEE J Select Areas Commun*. 2020;38(10):2271–2283. doi:10.1109/JSAC.2020.3000408
49. Wen J, Sheng M, Li J, Huang K. Assisting intelligent wireless networks with traffic prediction: exploring and exploiting predictive causality in wireless traffic. *IEEE Commun Mag*. 2020;58(6):26-31. doi:10.1109/MCOM.001.1900211
50. Xu Y, Yin F, Xu W, Lin J, Cui S. Wireless traffic prediction with scalable Gaussian process: framework, algorithms, and verification. *IEEE J Select Areas Commun*. 2019;37(6):1291-1306. doi:10.1109/JSAC.2019.2904330
51. Watkins Christopher JCH, Dayan P. Technical note: Q-learning. *Mach Learn*. 1992;8(3):279-292. doi:10.1023/A:1022676722315
52. Xie J, Yu FR, Huang T, et al. A survey of machine learning techniques applied to software defined networking (SDN): research issues and challenges. *IEEE Commun Surv Tutor*. 2018;21(1):393-430.
53. Kulin M, Kazaz T, De Poorter E, Moerman I. A survey on machine learning-based performance improvement of wireless networks: PHY, MAC and network layer. *Electronics*. 2021;10(3):318.
54. Xu Z-X, Cao L, Chen X-L, Li C-X, Zhang Y-L, Lai J. Deep reinforcement learning with sarsa and q-learning: a hybrid approach. *IEICE Trans Inf Syst*. 2018;101(9):2315-2322.
55. Gepperth A, Hammer B. Incremental learning algorithms and applications. Proceedings of the European symposium on artificial neural networks (ESANN), ESANN; 2016.
56. Kuhn M, Johnson K. *Applied Predictive Modeling*. New York, NY: Springer; 2013.
57. Narudin FA, Feizollah A, Anuar NB, Gani A. Evaluation of machine learning classifiers for mobile malware detection. *Soft Comput*. 2016;20(1):343-357.
58. Abbasi M, Shahraki A, Barzegar HR, Pahl C. Synchronization techniques in "device to device-and vehicle to vehicle-enabled" cellular networks: a survey. *Comput Electr Eng*. 2021;90:106955.
59. Alawe I, Ksentini A, Hadjadj-Aoul Y, Bertin P. Improving traffic forecasting for 5G core network scalability: a machine learning approach. *IEEE Netw*. 2018;32(6):42-49. doi:10.1109/MNET.2018.1800104
60. Wang Z, Hu J, Min G, Zhao Z, Wang J. Data augmentation based cellular traffic prediction in edge computing enabled smart city. *IEEE Trans Ind Inform*. 2020;17(6):4179–4187. doi:10.1109/tii.2020.3009159
61. Bayati A, Nguyen KK, Cheriet M. Multiple-step-ahead traffic prediction in high-speed networks. *IEEE Commun Lett*. 2018;22(12):2447-2450. doi:10.1109/LCOMM.2018.2875747
62. Kim J, Hwang G. Prediction based efficient online bandwidth allocation method. *IEEE Commun Lett*. 2019;23(12):2330-2334. doi:10.1109/LCOMM.2019.2947895
63. Wang W, Zhou C, He H, Wu W, Zhuang W, Shen XS. Cellular traffic load prediction with LSTM and Gaussian process regression. Proceedings of the IEEE International Conference on Communications; 2002; IEEE Institute of Electrical and Electronics Engineers Inc.
64. Poupart P, Chen Z, Jaini P, et al. Online flow size prediction for improved network routing. Proceedings of the International Conference on Network Protocols, ICNP; 2016; IEEE.
65. Mikaeil AM. Bayesian online learning-based spectrum occupancy prediction in cognitive radio networks. *ITU J ICT Discov*. 2017;(1):1–6.
66. Cohen R, Nezri Y. Cardinality estimation in a virtualized network device using online machine learning. *IEEE/ACM Trans Netw*. 2019;27(5):2098-2110. doi:10.1109/TNET.2019.2940705
67. Cui Q, Gong Z, Ni W, et al. Stochastic online learning for mobile edge computing: learning from changes. *IEEE Commun Mag*. 2019;57(3):63-69. doi:10.1109/MCOM.2019.1800644
68. Zhang Q, Yang LT, Chen Z, Li P. Incremental deep computation model for wireless big data feature learning. *IEEE Trans Big Data*. 2019;6(2):248-257. doi:10.1109/tbdata.2019.2903092
69. Yue C, Jin R, Suh K, Qin Y, Wang B, Wei W. LinkForecast: cellular link bandwidth prediction in LTE networks. *IEEE Trans Mob Comput*. 2018;17(7):1582-1594. doi:10.1109/TMC.2017.2756937
70. Zhang C, Zhang H, Yuan D, Zhang M. Citywide cellular traffic prediction based on densely connected convolutional neural networks. *IEEE Commun Lett*. 2018;22(8):1656-1659. doi:10.1109/LCOMM.2018.2841832
71. Pfülb B, Hardegen C, Gepperth A, Rieger S. A study of deep learning for network traffic data forecasting. Proceedings of the International Conference on Artificial Neural Networks; 2019:497-512; Springer, New York, NY.
72. Li M, Wang Y, Wang Z, Zheng H. A deep learning method based on an attention mechanism for wireless network traffic prediction. *Ad Hoc Netw*. 2020;107:102258. doi:10.1016/J.ADHOC.2020.102258

73. Tran QT, Hao L, Trinh QK. Cellular network traffic prediction using exponential smoothing methods. *J Inf Commun Technol*. 2019;18(1):1-18. doi:10.32890/jict2019.18.1.1
74. Shahraki A, Taherkordi A, Haugen Ø. TONTA: trend-based online network traffic analysis in ad-hoc IoT networks. *Comput Netw*. 2021;194:108125.
75. Shahraki A, Taherzadeh H, Haugen Ø. Last significant trend change detection method for offline Poisson distribution datasets. Proceedings of the 2017 International Symposium on Networks, Computers and Communications (ISNCC); 2017:1-7; IEEE.
76. Shahraki A, Haugen Ø. An outlier detection method to improve gathered datasets for network behavior analysis in IoT. *Journal of Communications*. 2019;14(6):455-462.
77. Bouzidi EH, Luong DH, Outtagarts A, Hebbar A, Langar R. Online-based learning for predictive network latency in software-defined networks. Proceedings of the 2018 IEEE Global Communications Conference, GLOBECOM 2018 – Proceedings; 2018. IEEEInstitute of Electrical and Electronics Engineers Inc.
78. Xu Y, Xu W, Yin F, Lin J, Cui S. High-accuracy wireless traffic prediction: A GP-based machine learning approach. Proceedings of the 2017 IEEE Global Communications Conference, GLOBECOM 2017 - Proceedings; 2017:1-6; IEEE.
79. Bouzidi EH, Outtagarts A, Langar R. Deep reinforcement learning application for network latency management in software defined networks. Proceedings of the 2019 IEEE Global Communications Conference, GLOBECOM 2019 – Proceedings; 2019; IEEE.
80. Nie L, Ning Z, Obaidat M, et al. A reinforcement learning-based network traffic prediction mechanism in intelligent Internet of Things. *IEEE Trans Ind Inform*. 2020;17(3):2169-2180. doi:10.1109/tii.2020.3004232
81. Zhao JJ, Qu H, Zhao JJ, Jiang D. Spatiotemporal traffic matrix prediction: a deep learning approach with wavelet multiscale analysis. *Trans Emerg Telecommun Technol*. 2019;30(12):e3640. doi:10.1002/ett.3640
82. Zang Y, Ni F, Feng Z, Cui S, Ding Z. Wavelet transform processing for cellular traffic prediction in machine learning networks. Proceedings of the 2015 IEEE China Summit and International Conference on Signal and Information Processing, China SIP 2015 – Proceedings; 2015:458-462IEEE Institute of Electrical and Electronics Engineers Inc.
83. Aldhyani THH, Alrasheedi M, Alqarni AA, Alzahrani MY, Bamhdi AM. Intelligent hybrid model to enhance time series models for predicting network traffic. *IEEE Access*. 2020;8:130431-130451. doi:10.1109/access.2020.3009169
84. Le LV, Sinh D, Tung LP, Lin BSP. A practical model for traffic forecasting based on big data, machine-learning, and network KPIs. Proceedings of the CCNC 2018 - 2018 15th IEEE Annual Consumer Communications and Networking Conference; 2018:1-4.
85. Soheil J, Hammoudeh Z, Durairajan R, Lowd D, Rejaie R, Willinger W. On the practicality of learning models for network telemetry. Proceedings of Network Traffic Measurement and Analysis ConferenceNational Science Foundation; 2020.
86. Faisal IM, Zahid M, Habib D, et al. Efficient prediction of network traffic for real-time applications. *J Comput Netw Commun*. 2019;2019:4067135. doi:10.1155/2019/4067135
87. Azzouni Abdelhadi, Pujolle Guy. A long short-term memory recurrent neural network framework for network traffic matrix prediction. arXiv preprint arXiv:1705.05690. 2017;.
88. Katris C, Daskalaki S. Comparing forecasting approaches for internet traffic. *Expert Syst Appl*. 2015;42(21):8172-8183. doi:10.1016/J.ESWA.2015.06.029
89. Shahraki A, Abbasi M, Haugen Ø. Boosting algorithms for network intrusion detection: a comparative evaluation of real AdaBoost, gentle AdaBoost and modest AdaBoost. *Eng Appl Artif Intell*. 2020;94:103770.
90. Lohrasbinasab I, Acharya PB, Colomo-Palacios R. BizDevOps: a multivocal literature review. lecture notes in computer science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); 2020:12254; LNCS:698-713. 10.1007/978-3-030-58817-5_50
91. Diethe T, Borchert T, Thereska E, Balle B, Lawrence N. Continual learning in practice; 2019. arXiv preprint arXiv:1903.05202.

How to cite this article: Lohrasbinasab I, Shahraki A, Taherkordi A, Delia Jurcut A. From statistical- to machine learning-based network traffic prediction. *Trans Emerging Tel Tech*. 2021;e4394. doi: 10.1002/ett.4394

Appendix B

(The main parts of the implementation codes)

B.1 The sampling procedure

```
    if(samplingcounter==samplingvalue)
         Input=mean(rawdataarray);
    if  any(ismember(round(rawdata(2),0),outliertimes))
    % && (round(rawdata(2),0)>rawdata(2))
        if mod((round(rawdata(2),0)),2)==0
            tempinput=Input;
    Input=Input*1.1;
    if abs(Input-tempinput)<5
    Input=tempinput+5;
    end
        else
             tempinput=Input;
                Input= Input*0.9;

            if abs(Input-tempinput)<5
                Input=tempinput-5;
            end


            if Input<0
                Input=0;
            end
        end

    end
    time=rawdata(2);
    rawdataaaray=[];
    samplingcounter=1;
    opcallerscript

    elseif (samplingcounter-1<samplingvalue);


        rawdataarray(samplingcounter)=rawdata(1)*multiplyval;

        samplingcounter=samplingcounter+1;
    end
```

.

B.2 Smoothing

```
    %Hample filter Code
    function [YY, I, Y0, LB, UB, ADX, NO] = hampel(X, Y, DX, T, varargin)
    % HAMPEL    Hampel Filter.
    %   HAMPEL(X,Y,DX,T,varargin) returns the Hampel filtered values of the
    %   elements in Y. It was developed to detect outliers in a time series,
    %   but it can also be used as an alternative to the standard median
    %   filter.
    %
    %   References:
    %   Chapters 1.4.2, 3.2.2 and 4.3.4 in Mining Imperfect Data: Dealing with
    %   Contamination and Incomplete Records by Ronald K. Pearson.
    %
    %
```

```
14      %% Error Checking
15      % Check for correct number of input arguments
16      if nargin < 2
17          error('Not enough input arguments.');
18      end
19
20      % Check that the number of elements in X match those of Y.
21      if ~isequal(numel(X), numel(Y))
22          error('Inputs X and Y must have the same number of elements.');
23      end
24
25      % Check that X is either a row or column vector
26      if size(X, 1) == 1
27          X   = X';
28          % Change to column vector[YY,I,Y0,LB,UB] = hampel(X,Y);
29      elseif size(X, 2) == 1
30      else
31          error('Input X must be either a row or column vector.')
32      end
33
34      % Check that Y is either a row or column vector
35      if size(Y, 1) == 1
36          Y   = Y';
37          % Change to column vector
38      elseif size(Y, 2) == 1
39      else
40          error('Input Y must be either a row or column vector.')
41      end
42
43      % Sort X
44      SortX   = sort(X);
45
46      % Check that DX is of type scalar
47      if exist('DX', 'var')
48          if ~isscalar(DX)
49              error('DX must be a scalar.');
50          elseif DX < 0
51              error('DX must be larger than zero.');
52          end
53      else
54          DX  = 3*median(SortX(2:end) - SortX(1:end-1));
55      end
56
57      % Check that T is of type scalar
58      if exist('T', 'var')
59          if ~isscalar(T)
60              error('T must be a scalar.');
61          end
62      else
63          T   = 3;
64      end
65
66      % Check optional input
67      if isempty(varargin)
68          Option  = 'standard';
69      elseif numel(varargin) < 2
70          error('Optional input must also contain threshold value.');
71      else
```

```
72          % varargin{1}
73          if ischar(varargin{1})
74              Option      = varargin{1};
75          else
76              error('PropertyName must be of type char.');
77          end
78          % varargin{2}
79          if isscalar(varargin{2})
80              Threshold   = varargin{2};
81          else
82              error('PropertyValue value must be a scalar.');
83          end
84      end
85
86      % Check that DX,T does not contain NaN values
87      if any(isnan(DX) | isnan(T))
88          error('Inputs DX and T must not contain NaN values.');
89      end
90
91      % Check that varargin does not contain NaN values
92      CheckNaN    = cellfun(@isnan, varargin, 'UniformOutput', 0);
93      if any(cellfun(@any, CheckNaN))
94          error('Optional inputs must not contain NaN values.');
95      end
96
97      % Detect/Ignore NaN values in X and Y
98      IdxNaN  = isnan(X) | isnan(Y);
99      X       = X(~IdxNaN);
100     Y       = Y(~IdxNaN);
101
102     %% Calculation
103     % Preallocation
104     YY  = Y;
105     I   = false(size(Y));
106     S0  = NaN(size(YY));
107     Y0  = S0;
108     ADX = repmat(DX, size(Y));
109
110     if numel(X) > 1
111         switch lower(Option)
112             case 'standard'
113                 for i = 1:numel(Y)
114                     % Calculate Local Nominal Data Reference value
115                     % and Local Scale of Natural Variation
116                     [Y0(i), S0(i)]  = localwindow(X, Y, DX, i);
117                 end
118             case 'adaptive'
119                 % Preallocate
120                 Y0Tmp   = S0;
121                 S0Tmp   = S0;
122                 DXTmp   = (1:numel(S0))'*DX;
123                 % Integer variation of Window Half Size
124
125                 % Calculate Initial Guess of Optimal Parameters Y0, S0, ADX
126                 for i = 1:numel(Y)
127                     % Setup/Reset temporary counter etc.
128                     j       = 1;
129                     S0Rel   = inf;
```

```
130                     while S0Rel > Threshold
131                         % Calculate Local Nominal Data Reference value
132                         % and Local Scale of Natural Variation using DXTmp window
133                         [Y0Tmp(j), S0Tmp(j)]    = localwindow(X, Y, DXTmp(j), i);
134 
135                         % Calculate percent difference relative to previous value
136                         if j > 1
137                             S0Rel=abs((S0Tmp(j-1)- S0Tmp(j))/(S0Tmp(j-1) +S0Tmp(j))/2);
138                         end
139 
140                         % Iterate counter
141                         j   = j + 1;
142                     end
143                     Y0(i)   = Y0Tmp(j - 2);
144                     % Local Nominal Data Reference value
145                     S0(i)   = S0Tmp(j - 2);
146                     % Local Scale of Natural Variation
147                     ADX(i)  = DXTmp(j - 2)/DX;
148                     % Local Adapted Window size relative to DX
149                 end
150 
151                 % Gaussian smoothing of relevant parameters
152                 DX  = 2*median(SortX(2:end) - SortX(1:end-1));
153                 ADX = smgauss(X, ADX, DX);
154                 S0  = smgauss(X, S0, DX);
155                 Y0  = smgauss(X, Y0, DX);
156             otherwise
157                 error('Unknown option ''%s''.', varargin{1});
158         end
159     end
160 
161     %% Prepare Output
162     UB      = Y0 + T*S0;            % Save information about local scale
163     LB      = Y0 - T*S0;            % Save information about local scale
164     Idx     = abs(Y - Y0) > T*S0;   % Index of possible outlier
165     YY(Idx) = Y0(Idx);              % Replace outliers with local median value
166     I(Idx)  = true;                 % Set Outlier detection
167     NO      = sum(I);               % Output number of detected outliers
168 
169     % Reinsert NaN values detected at error checking stage
170     if any(IdxNaN)
171         [YY, I, Y0, LB, UB, ADX]    = rescale(IdxNaN, YY, I, Y0, LB, UB, ADX);
172     end
173 
174     %% Built-in functions
175         function [Y0, S0] = localwindow(X, Y, DX, i)
176             % Index relevant to Local Window
177             Idx = X(i) - DX <= X & X <= X(i) + DX;
178 
179             % Calculate Local Nominal Data Reference Value
180             Y0  = median(Y(Idx));
181 
182             % Calculate Local Scale of Natural Variation
183             S0  = 1.4826*median(abs(Y(Idx) - Y0));
184         end
185 
186         function M = median(YM)
187             % Isolate relevant values in Y
```

```
            YM   = sort(YM);
            NYM = numel(YM);

            % Calculate median
            if mod(NYM,2)    % Uneven
                M    = YM((NYM + 1)/2);
            else             % Even
                M    = (YM(NYM/2)+YM(NYM/2+1))/2;
            end
        end

        function G = smgauss(X, V, DX)
            % Prepare Xj and Xk
            Xj   = repmat(X', numel(X), 1);
            Xk   = repmat(X, 1, numel(X));

            % Calculate Gaussian weight
            Wjk = exp(-((Xj - Xk)/(2*DX)).^2);

            % Calculate Gaussian Filter
            G    = Wjk*V./sum(Wjk,1)';
        end

        function varargout = rescale(IdxNaN, varargin)
            % Output Rescaled Elements
            varargout     = cell(nargout, 1);
            for k = 1:nargout
                Element      = varargin{k};

                if islogical(Element)
                    ScaledElement    = false(size(IdxNaN));
                elseif isnumeric(Element)
                    ScaledElement    = NaN(size(IdxNaN));
                end

                ScaledElement(~IdxNaN)   = Element;
                varargout(k)             = {ScaledElement};
            end
        end
    end
```

B.3 Creating the sliding window

```
finalbreak=0;
breakpoint=1;

result(1)=-1;
result(2)=0.0;
result(3)=0.0;
if (max(size(inputarray))>maximum_data)
inputarray=inputarray(round(interval*1.5):length(inputarray));
end
if (intervalcounter<=interval)

    intervalcounter=intervalcounter+1;
        inputarray=[inputarray;Input(1)];
```

```
14          alldata=[alldata;Input(1)];
15          allimp=[allimp;0];
16          intervalarray=[intervalarray;0];
17          timedata=[timedata;time];
18    else
19    if (max(size(inputarray))>=minimum_data)
20
21
22          result(2)=max(size(inputarray));
23         [meanbefore,meanafter,remaindata,removedata,breakarray]=
24         arimafinal1(inputarray,plotnum,alphnum);
25        % [meanbefore,meanafter]=meancalc(remaindata,removedata);
26
27
28
29
30         difference=abs(meanbefore-meanafter);
31        [fuzzyoutput]=evalfis([meanbefore meanafter difference],fismat);
32
33        if (length(remaindata)>(interval/3))
34    %
35     %     behaviour=fuzzyoutput(2);
36      % importance=fuzzyoutput(1);
37          behaviour=meanafter;
38          importance=difference;
39        else
40            behaviour=0;
41            importance=0;
42
43        end
44
45       if (behaviour==0 && importance==0)
46       %If the remainder is too small
47      inputarray=inputarray(7:length(inputarray));
48      disp('the remain data is so low');
49     % inputarray=inputarray(length(inputarray)- inimum_data+interval:length(inputarray));
50       intervalcounter=1;
51       else % If the remaining data was enough
52            inputarraytemp=inputarray;
53            if (importance>=imp_thr)
54               allimp(length(alldata)-length(remaindata))=importance;
55               intervalarray(length(alldata)-length(remaindata))=length(remaindata);
56            end
57               intervalnum=intervalnum+1;
58            if (imp_thr<importance) % If the observed change was very significant
59              finalbreaktemp=length(alldata)-length(remaindata);
60             if ((finalbreaktemp>finalbreak)&& finalbreak+(interval*3) <finalbreaktemp )
61             % If the found point was located after the previous point
62                 finalbreak=finalbreaktemp;
63                 breakpoint=breakarray1(end);
64                 breakarray1=[breakarray1;finalbreaktemp];
65
66
67                inputarray=remaindata;
68          %inputarray=dataappender(inputarray,remaindata,removedata,interval,minimum_data);
69
70          color='r';
71               else
```

```
                color='b';
             end

            else

           color='g';

            end
              plotname=strcat('Fig_',int2str(plotnum));
        plotnum=plotnum+1;
     allfname=   savefigure(inputarraytemp,removedata,remaindata,plotname,alldata,
outputmodel,behaviour,importance,breakpoint,color,allimp,imp_thr,timedata,
intervalarray,minimum_data,maximum_data,interval,simname,meanbefore,
meanafter,alphnum,finalbreak);
       result(1)=length(removedata);
       result(3)=length(remaindata);
           intervalcounter=1;
       end
    else
        intervalcounter=1;
          inputarray=[inputarray;Input(1)];
           alldata=[alldata;Input(1)];
            allimp=[allimp;0];
            intervalarray=[intervalarray;0];
            timedata=[timedata;time];
    end
    end
```

.

B.4 Connecting windows in a row.

```
    function [ inputarray ] = dataappender(
    alldata,remaindata,removedata,interval,minimum )
    x=minimum-(length(remaindata)+interval);
    if (x>0)
    %inputarray=[removedata(length(removedata)-x+1:length(removedata));remaindata];
    inputarray=[alldata(length(alldata)-length(remaindata)-x:length(alldata))];

    else
    inputarray=remaindata;
    end
    end
```

.

B.5 Identifying the change point

```
    function [location]=breakfinder(ar1,ar2)
    location=-1;
    for (i=1:max(size(ar1)))

    if (ar1(i)==ar2(1))
        for (j=2:max(size(ar2)))
            if (ar2(j)~=ar1((i+j-1)))
                location=0;
```

```
            else
                location=i+2;
            end


        end

    end
    end
    end
```

.

B.6 To return the break position

```
    function [ databeforeshekast,datafinal,breakposition ] = shekastkham( data,alshek)

     dindex=[1:length(data)];

    ddata=data(dindex);

    nogh_shek=round(length(ddata)*alshek);

    if(nogh_shek==0)
        datafinal=ddata;
        databeforeshekast=[];
        breakposition=[];
        return;
    elseif(nogh_shek==length(data))
    datafinal=[];
    databeforeshekast=ddata;
    breakposition=[];
    return;
    end

    databeforeshekast=ddata(1:nogh_shek-1);
    datafinal=ddata(nogh_shek:length(ddata));
    % if (nogh_shek==1 )
    %     breakposition=[-1;ddata(nogh_shek:nogh_shek+1)];
    % elseif (nogh_shek==max(size(ddata)))
    %     breakposition=[ddata(nogh_shek-1:nogh_shek);-1];
    % else
    % breakposition=ddata(nogh_shek-1:nogh_shek+1);
    breakposition=0.0;

    end
```

.