LEVERAGING A DEEP LEARNING-BASED PRIVACY COMPLIANCE FRAMEWORK FOR IOT APPLICATIONS

MSc Cyber Security Thesis

OCTOBER 4, 2024

HAMZA AHMED
The Manchester Metropolitan University

Table of Contents

Abbreviations:

IoT – Internet of Things

IIoT – Industrial Internet of Things

DL – Deep Learning

DLR – Deep Reinforcement Learning

ML – Machine Learning

AI – Artificial Intelligence

GDPR – General Data Protection Regulation

CCPA – California Consumer Privacy Act

NIST – National Institute of Standards and Technology

DNNs – Deep Neural Networks

CNNs – Convolutional Neural Networks

RNNs – Recurrent Neural Networks

LSTM – Long Short-Term Memory

MQTT – Message Queuing Telemetry Transport

DoS – Denial of Service

U2R – User-To-Root

R2L – Remote-To-Local

ENISA – European Union Agency for Cybersecurity

UI – User Interface

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Every piece of content is written by me as an MSc Cyber Security Student at The Manchester Metropolitan University apart from those parts of this project containing citations to the work of others. This project is my own unaided work.

Signed: Hamza Ahmed

October 2024

Acknowledgments

I wish to thank my personal tutor [redacted] for her guidance throughout this project.

Abstract

This thesis explores and investigates the development and implementation of a deep learning-based privacy compliance framework for Internet of Things (IoT) applications. The geometric growth in potentially sensitive data from the vast generations of data as IoT devices proliferate raises critical security and privacy concerns for individuals and organisations. Simultaneously, GDPR and CCPA which are rigorous data protection regulations have created an urgent and imperative need for automated compliance solutions.

Utilising these regulatory standards, the research harnesses deep learning methodologies, specifically Convolutional Neural Networks (CNNs) and Long-Short-Term Memory (LSTM) networks to tackle these complex challenges. Cutting-edge features are incorporated into the proposed deep-learning framework including threat characterising, anomaly detection, dynamic policy adjustment, and regulatory compliance checking/verification for IoT environments.

This study sought to employ publicly IoT datasets. In this approach, the efficacy and effectiveness in identifying potential security threats and privacy infringements are demonstrated through the framework where the implementation pipeline encompasses comprehensive data processing, model training, threat detection, compliance checking, and an intuitive user-friendly alert system.

The findings concluded that the deep-learning models achieved high accuracy in threat detection, concurrently while the integrated compliance checking functions effectively enforced privacy policies from a ‘docx’ file. In personal subjective opinion, this research thesis makes a significant contribution to the relevant field by offering a novel, automated approach to handling privacy compliance in intricate IoT landscapes, prospectively reducing organisational resource expenditure, meanwhile enhancing overall data protection measures.

Chapter 1 Introduction

1.1 Introduction

As we are moving into a digital age, the infrastructure of Internet of Things (IoT) applications is becoming more complex due to the increasingly interconnected ecosystem of networks, each bringing its unique vulnerabilities. Cyber-attacks have evolved becoming more persistent, advanced, and specifically targeted to individuals and organisations where IoT devices are implemented such as smart homes, healthcare, and industrial systems. In this case, signature-based defences, rule-based detections, and manual active monitoring have all become less effective in managing the proliferating threat landscape, all in conjunction with raising concerns about privacy compliance. Privacy regulations such as General Data Protection Regulation (GDPR) require organisations to ensure personal sensitive data is handled securely. Addressing these privacy compliance challenges concerning IoT applications/devices, overwhelmingly majority of organisations are looking for innovative solutions. At the same time, from the billions of IoT devices connected and used, vast amounts of data are generated, collected, analysed, and flow through the network unfiltered unless security controls i.e. firewalls are configured.

Enter the universe of deep learning (DL) – a subset of artificial intelligence (AI) utilised for predictive analysis, pattern recognition, and anomaly detection, with the prospective of revolutionising cybersecurity by fortifying organisations IT networks with the adoption of forward-thinking security protocols that can continuously identify, analyse and thwart potential attacks in real-time and also improving privacy protection (Alionsi, D.D. 2023).

1.2 Proposal

A novel solution presented in this report is leveraging a deep learning-based privacy compliance framework to increase security measures in place for personally identifiable/confidential information collected by IoT applications. Furthermore, DL-based frameworks can be leveraged through the automation of processes in monitoring and managing data privacy, thereby minimising resource expenditure and time investment for business entities, and boosting overall compliance with privacy regulations. A study from the paper ‘Compliance-Oriented IoT Security and Privacy Evaluation Framework’ shows the introduction to the concept of COPSEC, a framework designed and implemented to evaluate IoT devices’ compliance with privacy regulations and security guidelines. The authors found out from their research of conducting hundreds of automated experiments on a set of devices, revealing that these devices not only neglected fundamental security standards but also posed privacy vulnerabilities/risks due to their data collection operations. What can be highlighted is the concerning gap in privacy measures and IoT applications security, emphasising the critical necessity for comprehensive frameworks such as COPSEC to safeguard user data protection and adhere to regulatory compliance within IoT ecosystems (Anselmi, G. *et al.* 2023).

1.3 Motivations & Research Problems

The motivations for this research arise from several factors and its research problems. The first being the exponential market expansion of IoT applications with billions of IoT devices and the massive amounts of data they generate, making them vulnerable attributable to the growth which has outpaced the evolution of adequate privacy protection mechanisms. The second argument is the limitations of conventional privacy protection methods. They are insufficient for the heterogeneous and dynamic nature of IoT environments and grapple with managing to keep up with real-time data flow, in addition to the diverse regulatory requirements. The third motivation for conducting this research is that there are stringent data protection regulations like GDPR, the California Consumer Privacy Act (CCPA), and the National Institute of Standards and Technology (NIST) IoT Cybersecurity Guidelines. They have created an urgent requirement for automated compliance frameworks that align with shifting legal standards. Concurrently, the final motivation to undertake this project is the recent advancements in deep learning techniques. They present innovative solutions to tackle intricate challenges in data analysis and anomaly/pattern recognition. Factoring all these four points mentioned, leveraging the power of neural networks, deep learning algorithms can analyse and process large amounts of data from IoT applications, pinpoint patterns, and make intelligent decisions, applying it to be suitable for addressing privacy challenges in IoT environments.

1.4 Aims and Objectives

In regard to the research objectives, the proposed research aims to develop a novel framework that incorporates and leverages deep learning techniques, specifically a basic Long Short-Term Memory (LSTMs) because of its maintainability in privacy compliance in IoT applications and real-time threat detection. Also, Deep Neural Networks (DNNs) Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) are integrated to process and integrate data for various IoT ecosystems. The focus area of the framework will be on the following aspects: real-time anomaly and threat detection, federated learning, dynamic policy adjustment, aspects of blockchain technology, and comprehensive evaluation utilising Python (TensorFlow) when needed. Once the implementation is successful, the formulation of a privacy policy will necessitate the detected threats and uphold regulatory requirements.

1.5 Report Structure

Comprising of several chapters, this report is structured around the different stages undertaken during the initial phase, development, and outcome of the project concluded that an IoT application can be put through the process of checking if there are any threats detected from the IoT applications and if they comply with the regulatory frameworks and policies. In this introduction stage, the basis of the project thesis title is discussed, highlighting the necessity and importance of IoT applications complying with regulatory frameworks and from the proposal, the brief mention of the novel solution of the proposed product is discussed alongside my motivations and research problems for taking on this project idea.

The scope of this report centres around the need for a privacy compliance framework using deep learning for IoT applications.

Furthermore, the literature review brings up deep learning models i.e. CNNs, RNNs, LSTMs, and DNNs with a comparison of the different architectures and what I chose as the final model to train. This sections discusses the in-depth explanations of GDPR, CPPA, NIST IoT of Cybersecurity Guidelines and its challenges in implementing compliance across IoT applications. All alluding to current gaps and future directions in terms of weakness in current approaches and opportunities for improvements with privacy compliance frameworks. In each stage, references are used throughout the report, especially the literature review and related works section, thoroughly explaining various authors approaches and their proposed product. Additionally, it touches on the methodology regarding data collection and preparation (public IoT dataset), model development, framework implementation, testing and evaluation approach.

Moving onto the design stage, the systems/framework architecture is mapped out in the form of DFDs, flowcharts, pseudocode, activity diagrams, a top-down approach of an ERD diagram and also a diagram of the entire system. During the implementation stage, its starts off a discussion with data processing regarding cleaning, normalisation, and splitting of testing and training data of the IoT dataset, created in a pipeline inside of loading the data function . Then moves onto training the CNN and RNN (LSTM) models for threat detection to which an integration of the privacy compliance framework aspect kicks in the form of compliance checking functions and policy enforcement mechanism and finally, an alert system for potential threats and compliance status is enforced to let the user know if there are threats related to the IoT dataset and if it adheres to the regulatory frameworks stated. The results and discussions in the implementation of the proposed products are outlined too.

Chapter 2 Literature Review:

2.1 Deep Learning for IoT Security and Privacy in Applications

With the rise of IoT, comes a certainty the need for robust security and privacy measures significantly. Deep learning methods and techniques have surfaced as powerful tools for initiating and maintaining privacy and enhancing security in IoT applications, offering refined techniques for threat detection and policy compliance. Deep learning works by using artificial neural networks with multiple layers to learn from vast amounts of data and extract meaningful information from complex data, granting systems/machines to perform tasks that formerly required human-level intelligence. Its capacity to automatically acquire features and recognise patterns has driven significant progress across various domains, encompassing spaces like autonomous systems, scientific research, healthcare, and also natural language processing (Paluszek, M. and Thomas, S. 2020). From the academic papers and resources found and studied, there is no one solution to address my proposal. Aspects of each area of my proposal have been investigated to conduct this literature review below.

2.1.1 Deep Learning for Threat Detection

To enhance threat detection in IoT security, deep learning has been extensively applied using various architectures. The architectures focused on in this report include CNNs, RNNs using LSTMs, and also DNNs. In another scenario, a study has shown that CNNs and RNNs were compared for IoT malware detection. They both showcased the best accuracy in detecting IoT malware, with RNN being especially proficient in identifying new malware samples (99.19%), while CNN accomplished high accuracy (98.05%) with less training time (T. Salim, A. and Khammas, B.M. 2023). CNNs are marginalised used for spatial feature extraction, and RNNs are efficacious for temporal data analysis (Hemalatha, T. *et al.* 2023).

Furthermore, Selvakumar, B. *et al.* (2021) proposed a deep-learning framework for anomaly detection in IoT-enabled systems. What the various authors demonstrated was the effectiveness of CNNs and RNNs in identifying unusual patterns that may reveal security breaches. Their proposed system could process large volumes of IoT data in real-time maintaining immediate detection of anomalies compared to traditional methods. Similarly, Gokdemir, A., & Calhan, A. (2022) carried out a comparative study of machine learning and deep learning approaches for anomaly detection in IoT environments. Their research concluded showing Long Short-Term Memory (LSTM) algorithm achieved a higher accuracy in anomaly detection from cyber-attacks from their experiment of predicting “duplication, interception, and modification attacks in Message Queuing Telemetry Transport (MQTT) messages using an IoT dataset with artificial intelligence techniques”.

2.1.2 Comparisons of Different Neural Network Architectures

Moving on to discuss the comparisons of architectures, from the vast majority of information on the internet, I came across an article published on LinkedIn by Joinal Ahmed. What Joinal has illustrated are diagrams of different neural network architectures useful for my proposal and it is named 'A Comparison of DNN, CNN and LSTM using TF/Keras'.

2.1.2.1 Deep Neural Networks (DNNs)

DNNs (a class of neural networks) architecture consists of multiple layers of interconnected neurons, layers are fully connected and each neuron implements the equation "y = f(Wx + b) for inputs x and output y, where f is the non-linear activation function, W is the weight matrix and b is the bias" (Ahmed, J. 2020). DNNs have a tree-like architecture enabling them to model complex relationships between inputs and outputs (Almasi, M. 2023).

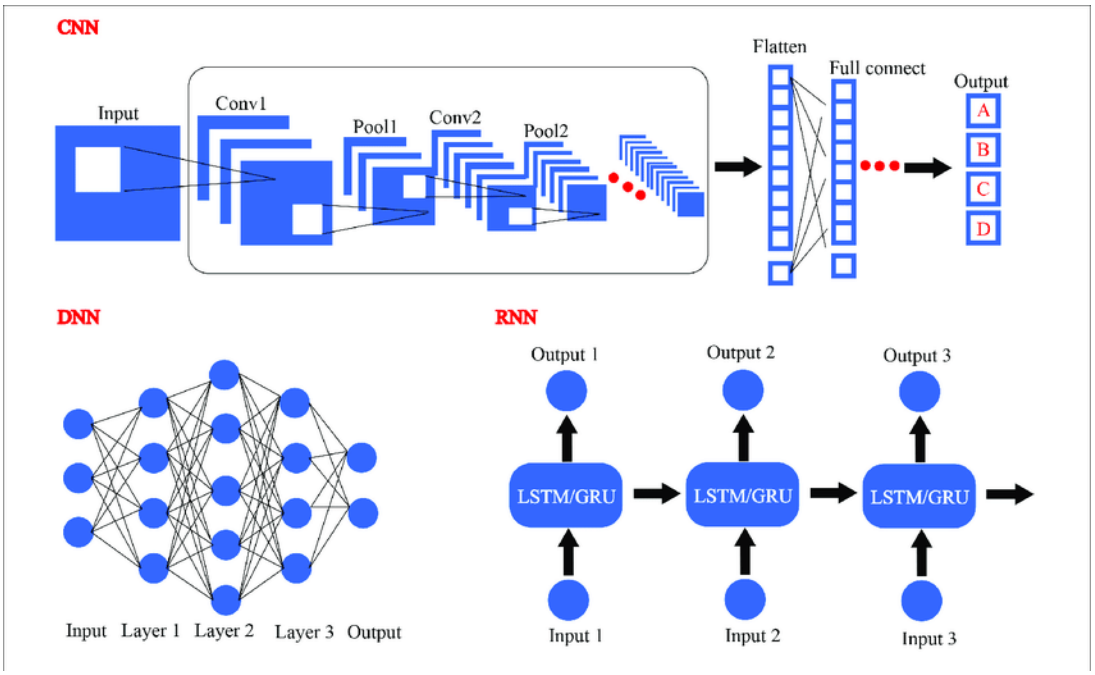


Figure 1.1: Different deep learning architectures (Ahmed, J. 2020)

From Figure 1, you can see the visualisation of architectures and the layers of interconnected neurons. It shows all three simplified architectures of deep learning. DNN characteristics include the simplest architecture out of the three presented, it is effective for various machine learning tasks and requires a large number of parameters for complex tasks.

2.1.2.2 Convolutional Neural Networks (CNNs)

From the visual representation taken from Figure 1.2, CNN have several parallel filters that can be tuned to extract various relevant features and back propagation computes the filter weights. On the left of Figure 2, the input vector is filtered by each of the convolutional layers. With a kernel, they convolve the input vector, and each convolutional layer then generates its own output vector. Diminishing the dimensionality, we can use a "pooling" layer to calculate the maximum/minimum or average number of samples.

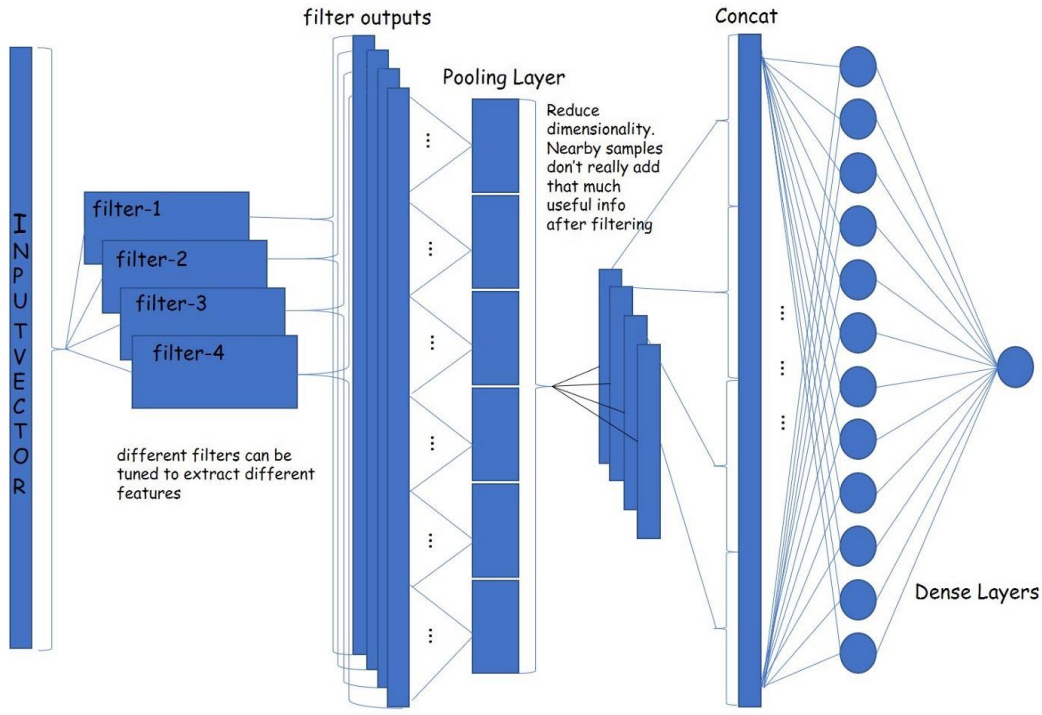


Figure 2: Convolution Neural Networks Illustration (Ahmed, J. 2020)

The characteristics I will summarise are that CNNs are effective at extracting spatial features, specialised for processing grid-like data for example images, reduces the number of parameters compared to DNNs, and are slightly slower to train than DNNs (Xiao, C. and Sun, J. 2021).

2.1.2.3 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

LSTM is a type of RNN and in sequential data, they are designed to capture prolonged dependencies. From Figure 1.3, you can see LSTM processes inputs sequentially passing "state information" between time steps unlike DNNs traditionally used today that process whole input sequences at once. In the obligations of natural language processing and time series data, LSTMs are well suited for these tasks because they can effectively learn and remember important information over long sequences. Over DNNs, the key advantage is the capability to manage long-term dependencies without necessitating an excessive number of parameters because they reuse the same set of neurons over multiple time steps, preserving pertinent information through their unique memory structure (Ahmed, J. 2020).

Characteristics summarised are that RNNs can process long sequences without increasing network sizes, can handle variable-length input sequences, LSTMs (a type of RNN) can capture long-term dependencies, are designed for sequential data aforementioned, and finally are slower to train than DNNs and CNNs.

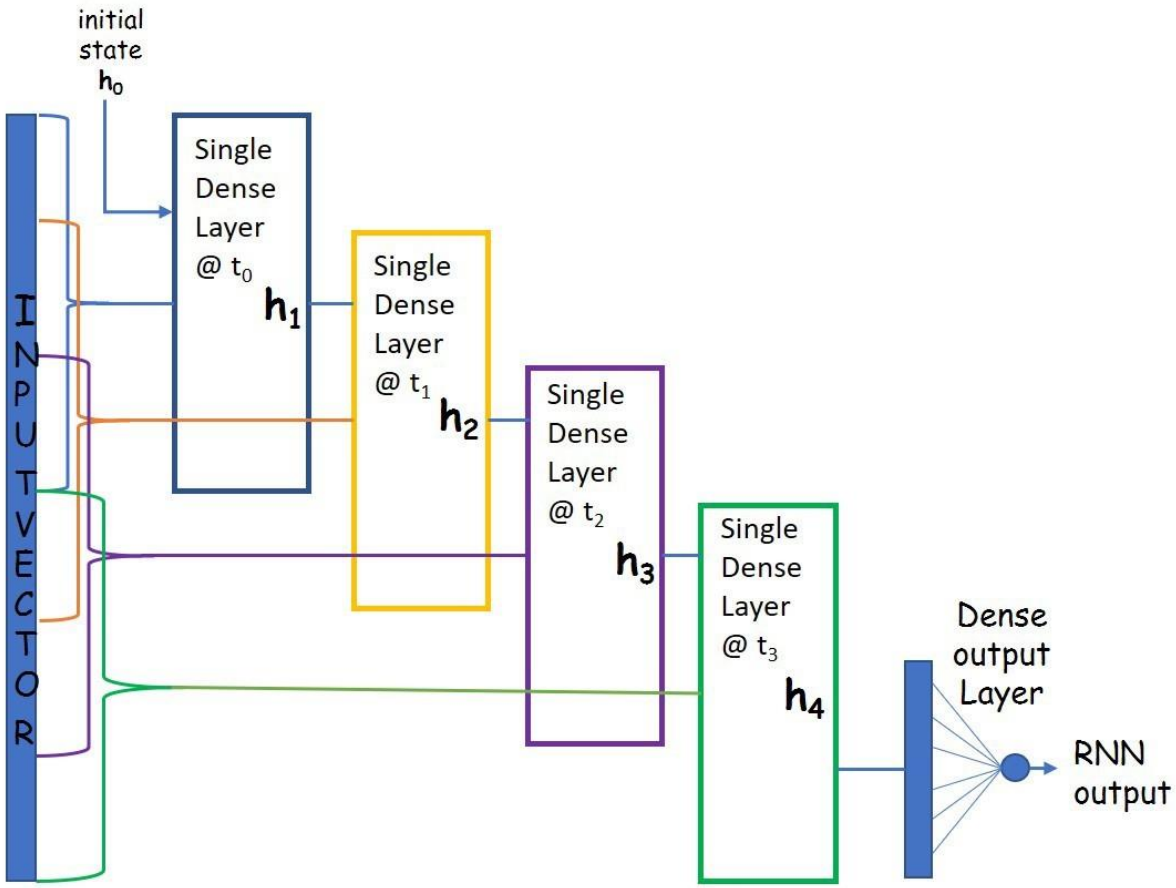


Figure 3: Recurrent Neural Networks Illustration (Ahmed, J. 2020)

Moreover, 5 aspects regarding comparisons are noted down between each architecture from (Ahmed, J. 2020) which I have taken into account when implementing my framework:

1. Training Speed
    a. CNNs are marginally slower than DNNs.
    b. RNNs are the slowest among the three architectures.
    c. DNNs are relatively the fastest to train.
2. Sequence Handling
    a. Without expanding network size, LSTMs are able to handle big sequences.
    b. To manage extended sequences, CNNs, and DNNs require large architectures.
3. Parameter Efficiency
    a. CNNs are the most efficient out of all three architectures, only needing fewer parameters than DNNs for corresponding duties/tasks.
    b. LSTMs use less parameters than DNNs but more than CNNs
4. Specialisation
    a. DNNs are multi-functional however they require large architectures for complex tasks.
    b. RNNs/LSTMs are optimal for sequential data and capturing temporal dependencies.
5. Time Series Prediction Performance

a. When all three architectures trained on the same number of input samples (IoT datasets), they performed similarly.
b. Based on past predictions, DNNs exhibited good performance in the generation phase.
c. CNNs performed adequately with less parameters however displayed a few limitations in long-term signal generation.
d. LSTMs could handle bigger IoT datasets without increasing the network size.

Touching on different deep learning architectures, (Otoum, Y., Liu, D., & Nayak, A. 2022) implemented a deep learning-based instruction detection framework (DL-IDS) that is specifically utilised for securing IoT networks, demonstrating the successfulness of RNNs and CNNs in detecting/identifying anomalies and intrusions such as denial of service (DoS), user-to-root (U2R) attack, remote-to-local (R2L) attack and probe attack.

Thinking about what has been discussed regarding which architectures are recommended and given the requirements of the framework, my proposed product will include LSTMs and CNNs to train as models as they are recommended due to their ability to capture long-term dependencies and their efficiency in handling sequential data, both critical for maintaining privacy compliance in IoT applications and real-time threat detection. DNNs require large architectures to solve complex tasks making it invaluable.

2.2 Deep Learning for Policy Adjustment and Framework Integration

Assimilating adaptive policy frameworks into deep learning-based threat detection offers significant prospects but challenges for IoT security. With this integration, there comes a challenge due to resource constraints, heterogeneity, and its dynamic landscape. The rapidly evolving attack vendors necessitate ongoing model updates and policy optimisation as stated by (Hemalatha, T. *et al.* 2023).

The authors of 'Real-time security threat detection in IOT devices using machine learning algorithms' proposed a novel security algorithm which leverages ML to mitigate but first detect security threats in real-time, utilising "multi-layer perception model trained on a diverse dataset of IoT device behaviors", achieving a 92% accuracy in identifying positive results of accurately distinguishing security threats (Raju Ch and Dr. A.V. Krishnaprasad 2023).

From the papers preceding analysis, IoT applications often have insufficient computational memory and power, posing difficulties in executing complex deep-learning models for real-time policy adjustments. In addition to this, the diverse characteristics and nature of IoT applications and protocols obscure the development in creating cohesive/unified policy frameworks. Every application in the nature of IoT environments has different capabilities, security requirements, and communication protocols, raising concerns about how it's possible

to implement a one-size-fits-all solution when developing a deep learning-based privacy compliance framework for IoT applications.

Addressing the issues raised, Popoola, S.I. *et al.* (2022) demonstrated the effectiveness of federated deep learning for zero-day botnet attack detection in IoT-edge devices. Concerning the dynamic threat landscape, federated learning can train models across multiple devices, devoid of sharing raw data to preserve privacy but at the same time guarantee the model benefits from diverse IoT data sources. My implementation will deal with updating security policies for a specific IoT application without compromising data privacy but does not enforce FL as a single dataset will be used.

2.2.1 Adaptive Policy Mechanisms Using Deep Learning and Frameworks

Transitioning into the facet of leveraging deep learning for adaptive policy management in IoT applications, several researchers have proposed frameworks in this domain but more is discussed in the related works section. As deep learning frameworks continuously analyse IoT network traffic, environmental factors, and IoT device behaviour, this approach allows systems to respond rapidly to emerging threats and developing conditions enabling a flexible and robust security framework. They can both work hand-in-hand to enhance the security and privacy of IoT applications against evolving cyber threats.

Deep learning models conjoin with policy engines and feedback loops for example, typical architectures like CNNs, RNNs, or hybrid models for policy optimisation and threat detection collaborate with components that translate deep learning outputs into actionable security policies. Feeding this information back into the learning process, mechanisms like a feedback loop evaluate the effectiveness of policy changes. Examples and implementation include (Kumar, R. and Tripathi, R. 2021) who proposed a ‘deep blockchain-based trustworthy privacy-preserving secured framework (DBTP2SF)’ for Industrial IoT (IIoT) environment using an “IIoT-based realistic dataset, namely ToN-IoT” (Kumar, R. and Tripathi, R. 2021). The framework incorporates reinforcement learning techniques to constantly adapt security policies based on device behaviour patterns, advancing threat landscapes, and alterations in network topology. (Popoola, S.I. *et al.* 2022) and (Aljuhani, A. *et al.* 2024) have similar proposals where they discuss deep learning for threat detection and policy refinement, collaboration of learning across IoT applications, privacy-preserving policy updates without centralised sensitive data, and adaption to contemporary attack patterns without comprising individual application privacy. What has been mentioned in this section allows the framework developed in this thesis project to ensure transparency in policy changes while leveraging deep learning for adaptive security measures.

2.3 Privacy Compliance Frameworks

Within IoT applications, privacy compliance frameworks play a pivotal job in addressing the privacy challenges of what IoT applications can function throughout their device lifetime. Stemming from the exponential influx in data generation from IoT devices, it raises catastrophic security and privacy challenges. Encryption, self-protection, identity management, and policy enforcement are proposed to safeguard confidential data in IoT ecosystems. For example, the notable PACTA framework (an IoT data privacy regulation compliance scheme) that leverages Trusted Execution Environments (TEEs) and blockchain ensures compliance with data privacy regulations, efficiently managing both static and dynamic data owner consent and compliance analysis (Echenim, K.U. and Joshi, K.P. 2023). In this report, only 3 privacy compliance frameworks (GDPR, CCPA, and NIST IoT Cybersecurity Guidelines) are discussed and compared.

2.3.1 Regulatory Requirements

2.3.1.1 General Data Protection Regulation (GDPR)

Implemented in 2018, the European Union's legislation GDPR replacing the Data Protection Act 1998, is now recognised as the global benchmark for data protection and privacy regulations to protect EU citizens' data regardless of the entity's location. There are other existing regulatory frameworks such as the 'US Health Insurance Portability and Accountability Act (HIPPA), the United States (US) National Institute of Standards and Technology Interagency Act (NISTIR), and GDPR aforementioned, all aiming to address the security concerns within this context of IoT data handling practices and compliance throughout the data lifecycle stages. Encompassing the forgoing statements, the IoT-Reg ontology incorporates GDPR and HIPPA, providing a comprehensive approach to IoT data handling and emphasising risk management (Zhang, Y. *et al.* 2024).

Furthermore, the key GDPR requirements for any IoT application are its 'right to access, rectify and erase personal data', explicit user consent for processing and data collection, 'data protection and implementation of privacy by design and default', and data breach notification within 72 hours (Zhang, Y. *et al.* 2024). Recent studies from the related works section have investigated the application of GDPR principles in the environment of IoT. (Zhang *et al* 2023) launched a BERT-based experimental assessment of privacy policies' compliance with GDPR underscoring the difficulties in automated compliance verification and its adherence. Similarly, (Rahat *et al* 2022) introduced a deep learning-based framework, evaluating privacy policies' compliance with GDPR to tell us that the potential of AI can be used in automating compliance checks.

2.3.1.2 California Consumer Privacy Act (CCPA)

CCPA (effective since January 1st, 2020) is a privacy law legislated to enhance and mandate privacy protections including IoT applications, and the rights of California USA residents, impacting numerous aspects of processing and data collection by organisations mainly. The enforcement of CCPA gives California residents the right to know what personal information about them is being collected, the reasons for its use, and to whom it is being disclosed, in addition to allowing consumers to delete and opt-out of the sale of their personal information.

Regarding my proposed product, due to the geographical locations where IoT applications are being used/tested against the deep learning-based privacy compliance framework, the CCPA law compliance will be applied to this framework if the location of its users and IoT devices is set in California, with the addition of checking if the flow of traffic is compliant or not or if they do not leverage standard communication protocols for example.

A recent study shows that 6 out of 10 consumer IoT devices testing against two security guidelines "(GP-TM-50 and ENISA's - GP-OP-04)" shown on Table 1 of its journal (Anselmi, G. *et al.* 2023) violates GP-TM-50 with unused ports during idle which is non-compliant with the guideline. The same 6 devices are non-compliant with GP-Op-04 because they use non-standard protocols when idle. Even one of the IoT applications they tested sends firmware update requests with a plant text MAC address which is dangerous and violates GDPR (Anselmi, G. *et al.* 2023).

2.3.1.3 National Institute of Standards and Technologies (NIST) IoT Cybersecurity Guidelines

NIST is an American federal agency and its IoT Cybersecurity guidelines offers a comprehensive framework, making sure the privacy and security of IoT systems are applied. The mentioned guidelines aim to support the creation and implementation of secure IoT devices/systems providing recommended approaches and methods for developers, users, and manufacturers. Key components include device & network security, data protection, access control, monitoring and detection, incident response and to conclude, privacy controls.

The guidelines in organisations and the context of the proposed product detects privacy threats and adjusts privacy policies to maintain compliance with regulations like CCPA and GDPR. Furthermore, the framework leveraged/implemented ensures that IoT applications follow NIST's security guidelines such as firmware integrity checks or secure boot processes. Moreover, the deep learning model used in the framework is designed to detect anomalies in data flows. What's made sure is data at rest and in transit is encrypted and safely stored, complying with NIST's data protection guidelines. Further reasons NIST guidelines are integrated into the proposed framework is that the implementation of NIST's network security practices, integration of NIST's monitoring and detection techniques, and lastly ensuring compliance with NIST's privacy controls are all utilised, etc.

2.4 Comparison of Requirements across Different Regulations

Serial Number	Regulatory Frameworks	Insight	Citation
1	General Data Protection Regulation (GDPR)	GDPR mandates explicit consent for data processing, rectification, erase data, grants rights to access data, and sanctions breach notification. It is emphasised the importance of data protection being integrated into systems and applications by design and default	(Zhang *et al.* 2024)
2	California Consumer Privacy Act (CCPA)	For residents of California, CCPA enhances privacy rights facilitating user control over data, transparency, and meticulous data handling practices.	(Anselmi *et al.* 2023)
3	NIST IoT Cybersecurity Guidelines	Provides a framework for securing IoT systems, concentrating on device security, incident response, access control, and data protection too.	(Popoola *et al.* 2022)

Table 1: A Comparison of Different Regulatory Frameworks

The proposed deep learning-based privacy compliance framework for IoT applications integrates these regulations by leveraging DL for decentralised data processing, ensuring GDPR compliance, having a robust IoT application dataset and network security for NIST IoT Cybersecurity Guidelines, and finally dynamic policy adjustment and threat detection for CCPA adherence.

2.5 Challenges in Implementing Compliance across various IoT applications

2.5.1 The heterogeneity of IoT devices and the data formats needed

Due to the heterogeneity of data formats and devices over the years, implementing compliance over a set of diverse IoT applications faces a catastrophic challenge because they are expected to reach 41 billion by 2027 (Rizvi, S., Campbell, S. and Alden, K. 2020). The components of various IoT devices range from simple sensors to complex systems, each with unique capabilities, data structures, and protocols. With these IoT devices, comes the vast amounts of data generated in multiple formats like XML, JSON, mostly CSV, and other

proprietary formats. That means the complication of data integration and compliance efforts (Kibria, M.G. *et al.* 2017). There is an insufficiency of fundamental security measures resulting numerous challenges in terms of compliance with GDRP and CCPA regulations etc. A study by Palo Alto Networks concluded that "57% of IoT devices are vulnerable to medium or high-severity attacks" making them low-hanging fruit for attackers to recon and exploit (*2020 unit 42 IOT threat report* 2024).

2.5.2 Challenges in Implementing Compliance across various IoT applications

The significant challenges in implementing compliance across a variety of IoT applications result from interoperability issues, resource constraints, scalability, data security and privacy, data management, user privacy and consent, cross-border data flow, and maintaining adherence to regulatory compliance. Cyber-attacks are a major compliance challenge for example in healthcare, IoT devices such as IoT-based smart drug delivery systems must comply with regulations that govern patient data protection and medical devices (Raikar, A.S. *et al.* 2023). The necessity to protect this data from cyber-attacks such as unauthorised access is a major compliance challenge. In the proposed systems presented in this thesis, the deployment/development of the 'deep learning-based privacy compliance framework' needs to be robust and for this to try and tailor to unique characteristics of IoT applications across different sectors.

2.6 How Privacy Compliance Frameworks Fit into Deep Learning

2.6.1 Challenged in Integration and Interpretation of Deep Learning outputs for Compliance purposes

Interpreting and integrating deep learning outputs of IoT applications for compliance purposes comes with a certain struggle. The proposed product in this thesis leverages deep learning techniques like LSTM with RNN and CNN to enhance privacy compliance and threat detection. Mentioning the heterogeneity of IoT devices and deep learning, complicate integration between the two. Interpreting deep learning output requires translating very complex model predictions into practical compliance measures. DL offers a solution to all of the opaque nature of these models and integration and ensures compliance with regulatory requirements (GDPR, CCPA, etc) because it preserves privacy while training global models enabling data processing already mentioning this aspect. However, there is no insurance of consistent policy enforcement across devices. Further challenges include a strain on computational resources and hindering real-time compliance.

2.6.2 Novel Approaches to Automated Compliance Checking Using AI

Many of these novel approaches and researcher's proposed products/systems are detailed in my related works section. Therefore, the recent advancements in AI have surpassed previous

research and allowed for new novel approaches and implementation for automating compliance checking.

Not just FL allowing decentralised training of models across multiple devices without needing to share raw data which makes sure the compliance frameworks like the proposed product can adapt to the dynamic nature of IoT environments without compromising data privacy, but reinforcement learning for dynamic policy adjustment so that it can continue to update and optimise security policies. The domains of automated compliance checking that have been explored are AI systems, Building Information Modelling (BIM) models, and ethical AI frameworks, all leveraging deep learning models and natural language processing to automate compliance assessments detect errors, and certify adherence to standards and regulations.

2.7 Current Gaps and Future Directions

2.7.1 Identified Weaknesses in Current Approaches

There are two main weaknesses around the current approach of this project. The first is not enough rule-based compliance systems in IoT environments and the second is the lack of real-time adaption to evolving threats and regulations. IoT ecosystems are expanding rapidly because of IoT devices with varying capabilities and protocols and the addition of this heterogeneity complicates the development and implementation of universal rule-based compliance systems as they often rely on signature-based defences and manual monitoring. Existing privacy compliance frameworks such as 'IoTPrivComp' and 'COPSEC' neglect fundamental security standards and pose privacy risks due to their data collection. IoT applications will always be vulnerable to new attack vectors and non-compliant with ever-changing privacy laws.

2.7.1.1 Opportunities for Improvement

Addressing these weaknesses, DL especially LSTM will be leveraged for privacy-preserving model training so it can allow IoT devices to collaboratively learn from datasets without compromising individual data privacy. In this case, as mentioned before, one IoT application dataset is used in this implementation. No programming code regarding the implementation of a privacy compliance framework using deep learning by authors/researchers in this domain has been included in their work which points to the reason for the complexity of this scenario. As this project undertaken is novel, the ability to combine deep learning with IoT applications and have the aspect of regulatory frameworks integrated makes it innovative.

2.7.2 Proposed Research Direction and My Contribution

This innovative approach of developing a novel privacy compliance framework combining deep learning threat detection with dynamic policy adjustment is my contribution. First, the public IoT dataset is handled by splitting the training (trains the model) and testing (evaluates the model) data and then looking out for device sensor/traffic logs/user activity logs. For the training deep learning models, the data will need cleaning and reformatting to handle missing

values and removing noise. Labelling the data based on whether it represents compliance with regulations or privacy threats will be required. The proposed framework leverages basic deep learning models especially LSTM to detect privacy threats in potentially real-time with the addition of CNNs and RNNs to process the sequential data in Python. Also, a dynamic policy adjustment to update privacy policies (written document) based on detected threats and changing regulatory environments will be implemented ensuring continuous compliance. The evaluation stage will test the privacy compliance framework and if it aligns with regulatory standards such as GDPR, CCPA, and NIST IoT Cybersecurity Guidelines.

Combining all the elements stated, the expected outcome is a robust and scalable framework that could be developed/deployed in diverse IoT environments, proving compliance enforcement and privacy threat detection.

2.8 Related Works

In this area of expertise, there have been a number of projects, products, and research initiatives around deep learning, privacy compliance frameworks, and IoT applications. These areas are often explored separately, but there are notable work being done integrating them together. However, in all research papers, no implementation of code/programming languages is added into their report but only the results of their implementation. This makes it much harder to design and implement the proposed system.

The authors of 'Exploring deep federated learning for the Internet of Things: A GDPR-compliant architecture' presented a GDPR-compliant architecture using deep federated learning for IoT applications. GDPR compliance was ensured by decentralising data processing/keeping data localised on devices while training a global model collaboratively. This method aided in retaining compliance with GDPR reducing centralisation and data transfer (Abbas, Z. *et al.* 2024). The authors (S. Chandra Sekaran *et al.* 2024) presented a "comprehensive framework for Privacy Assurance in Autonomous IoT Systems", also known as 'PAIS' aligning with evolving data protection regulations globally which is paramount for them as they have stated this in their research. Their deep intelligent framework using nonlinear analysis and topological methods seeks to detect and mitigate privacy threats, through analysis of complex data patterns in autonomous IoT systems.

Lane, N.D. *et al.* utilised two common deep learning algorithms used in this report, CNNs and DNNs, to build a model processing sensor data employed in smartphones and wearables only by using a triad of distinct IoT hardware. Das, R. *et al.* employed deep learning techniques, specifically LTSM networks, authenticating devices by detecting characteristic hardware imperfections in low-power components, bolstering security in IoT ecosystems. Furthermore, a novel IoT threat detection system was leveraged using LTSM networks by Das, R *et al* which incorporated 'OpCode analysis' and feature extraction, etc. While the proposed product 'TF-IDF' was tested for feature selection, the results from the study indicated its inefficiency with real-time processing and large datasets hinders its real-world

utility for IoT security applications in contrast to the LSTM-based approach (Das, R. *et al.* 2018).

The authors Tama, BA, and Rhee, KH brought together a DNN for IoT attack classifications. For the performance evaluation, the authors utilised subsampling and cross-validation that initialised parameters through a 'time-intensive grid method (Tama, BA and Rhee, KH. 2017). A learning-based deep-Q-network by Mohamed Shakeel, P. *et al.* presented a framework to maintain the privacy and security of IoT applications in IoT healthcare environments. On the other hand, a bidirectional LSTM-RNN method was used for botnet detection in IoT applications due to the complexity and ineffective implementation of an IDS (McDermott, C.D., Majdani, F. and Petrovski, A.V. 2018).

Shurrab, M. *et al,* proposed a deep learning-based framework to address cold start and sensor bias issues of low cost IoT-enabled monitoring applications/sensors (LCS). Their framework improved data reliability, which is crucial for maintaining privacy compliance and accurate threat detection within environmental factors (Shurrab, M. *et al.* 2024). Okafor, N.U. and their fellow authors presented a similar case where they proposed a data fusion and machine learning approach to enhance the data quality of low-cost IoT sensors with the same outcome of privacy-complaint data handling (Okafor, N.U., Alghorani, Y. and Delaney, D.T. 2020).

Furthermore, Garg, S *et al* proposed a deep reinforcement learning (DLR) framework to improve overall network performance, security posture, robust privacy and security compliance in next-generation IoT networks as the framework can dynamically evolve defence mechanisms, adapt to changing conditions and protect against emerging vulnerabilities (Garg, S. *et al.* 2023). Moreover, the authors conducting this research of 'Detecting Internet of Things attacks using distributed deep learning' presented a cloud-based distributed deep learning framework for detecting cyber threats like phishing and botnet attacks. They utilised CNNs and RNNs and had models trained on the UNSW-NB15 dataset which showed high accuracy in identifying privacy threats, and contributed to privacy compliance efforts detecting potential attacks (Janardhana, D.R., Manu, A.P. and Pavan Kumar, V. 2023). Abdel-Basset and associated authors followed the same technique of exploring the use of CNNs and RNNs for their proposed product for anomaly detection and privacy threat identification in IoT systems (Abdel-Basset, M., Moustafa, N. and Hawash, H. 2022).

Regarding this concluded section of related works, (Rahat, T.A., Long, M. and Tian, Y. 2022) and Zhang, L. *et al.* deployed BERT (deep learning) models to automate GDPR compliance verification of privacy policies. Deep learning privacy-compliant frameworks such as 'DeepCog' (Schiliro, F. *et al.* 2023), 'DBTP2SF' (Kumar, R. and Tripathi, R. 2021), and 'IoTPrivComp' (Ahmad, J., Li, F. and Luo, B. 2022) within the context of smart cities and industrial settings where IoT applications exist, integrate deep learning with blockchain technology to magnify the security and privacy of its various IoT applications. With (Garg, S. *et al.* 2023) proposed framework using DFL for adaptive security in next-generation IoT networks, these projects undertaken are complemented by domain-specific/advanced

methodologies such as rule-based models (Chakraborty *et al.,* 2023) and ECC-based mechanisms (Sivaranjani, R., Rao, P.M. and Saraswathi, P. 2021) for detecting novel threats.

All of these relevant research methodologies and products presented by authors in this domain have one thing in common, which is making sure their deep learning-based framework complies with the evolving regulatory standards such as GDPR, CCPA, and NIST IoT Cybersecurity Guidelines while significantly improving the security in a range of IoT applications and environments.

A table comparison of each product/ research idea from the related works can be seen in Appendix A with their strengths and weaknesses (their proposed strengths and weaknesses too) associated and what will be included/implemented in the proposed product.

Acting as a guide for my product production design and implementation, 3 research papers come to mind, (Abbas, Z. *et al.* 2024) GDPR-compliant architecture using federated learning, (S. Chandra Sekaran *et al.* 2024) PAIS framework to detect and mitigate privacy threats, and also (Janardhana, D.R., Manu, A.P. and Pavan Kumar, V. 2023) 'Detecting privacy attacks in IOT network using Deep Learning Models' employing RNNs to detect cyber-attacks, trained on UNSW-NB15 IoT dataset.

Chapter 3 Design

3.1 Requirements

There are 3 requirements in this report:

1. Technical Requirements
    a. Data Handling – Managing the large volumes of data generated from a public IoT dataset. This is coming from the CSV file storing all information.
    b. Real-Time Processing – Data flow management and dynamic policy compliance checks adapting to the dynamic nature of IoT environments.
    c. Deep Learning Models – Anomaly detection and privacy threat identification using Deep Learning i.e. LSTMs, CNNs.
2. Regulatory Compliance
    a. GDPR – Making sure the IoT application complies with GDPR by addressing the right to access, rectify, and erase personal data, user consent, privacy by design, etc.
    b. CCPA – Ensuring compliance with CCPA allowing for California residents with similar processes and procedures like GDPR.
    c. NIST IoT Cybersecurity Guidelines – Ensuring robust security and privacy practices for the IoT application, the proposed framework must align with NIST guidelines.
3. Integration and Deployment
    a. Framework Integration – Ensuring the proposed framework supports integration and dynamic policy adjustments based on detected threats.
    b. Testing and Evaluation once Framework Integration works – Ensures effective privacy management and regulatory compliance.

3.2 Methodology

The following steps make up the methodology:

1. Data Collection and Preparation
    a. Collecting and preprocessing the data from the public IoT application dataset (mentioned in the implementation phase), labelling data for compliance or privacy threats including testing and training the data in Google Colab.
2. Model Development
    a. Develop and train basic RNNs (LSTMs) and CNNs using Python (TensorFlow) for anomaly detection and pattern recognition tasks.
3. Framework Implementation
    a. Proposed framework implemented detecting privacy threats and dynamically adjusting policies based on regulatory requirements and threats
4. Testing and Evaluation
    a. Ensuring the framework complies with GDPR, CCPA, and NIST Guidelines during the testing phase
5. Deployment

a. Once the framework is successful, deployment occurs continuously, updating it to adapt to new regulations and threats.

The design phase will consider the following components. The first is dataset collection and preparations, model development, privacy compliance framework implementation, policy formation, testing, and evaluation.

3.3 Diagram Design

3.3.1 Flowchart of Proposed Framework:

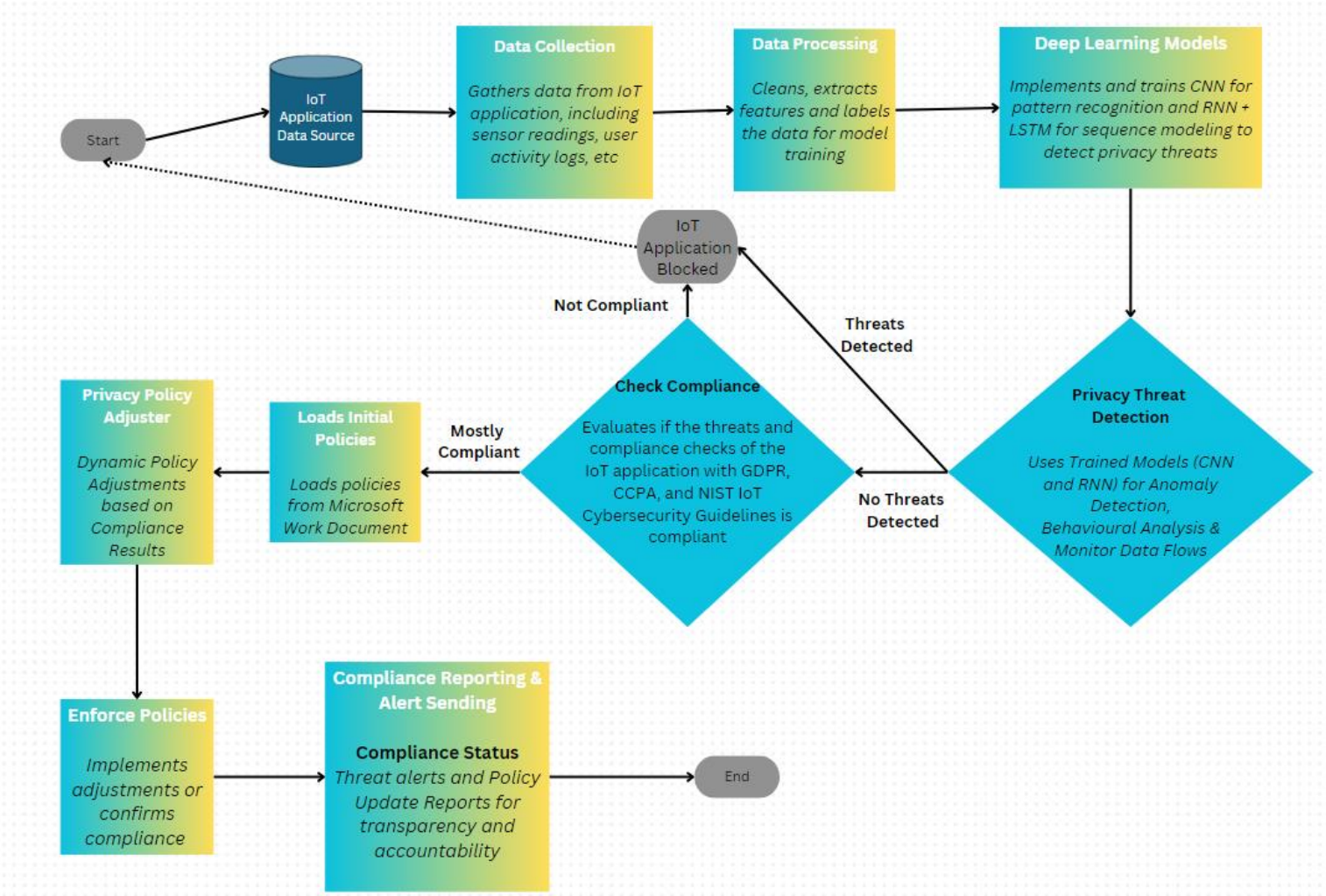


Figure 4: Flowchart of Proposed Framework

3.3.2 Pseudocode of Proposed Framework:

```
# Pseudocode for a 'Deep Learning-Based Privacy Compliance Framework for an IoT Application'
# Dataset/CSV file can be data analysed in Microsoft Excel for any help before loading
# into program
# The code will be split up into 7 sections
# 1). Data Collection and Pre-Processing 2). Model Training With Tensorflow
# 3). Threat Detection                   4). Compliance Checking
# 5). Policy Adjustment                  6). Real-time Policy Enforcement and Monitoring
# 7). An Alert Procedure/Mechanism


Importing the Neccesary Libraries
Import tensorflow as tf  <---Example


#Step 1): Data Collection and Pre-Processing
Function i.e. 'def' load_and preprocess_data(file_path):
    Loading the Dataset from (file_path)
    # Pre-processing stage of handling missing values and normalisation
    handle missing values
    Seperate features (X) and target (y)
    # Next stage is splitting the data
    Split data into training and test sections
    Normalise data
    Reshape data for CNN and RNN models
    # Final command in this Function
    Return X_train, X_test, y_train, y_test
```

```
#Step 2): Model Training with Tensorflow, Two Functions for training CNN and RNN
#using TensorFlow's Keras API
Function train_cnn_model(X_train, X_test, y_train, y_test):
    Create CNN Model
    Compile CNN Model
    Train CNN Model
    Return cnn_model

Function train_rnn_model(X_train, X_test, y_train, y_test)
    Create RNN Model
    Compile RNN Model
    Train RNN Model
    Return rnn_model

#Step 3). Threat Detection
Function detect_threats(cnn_model, rnn_model, X_test):
    Get predictions from cnn_model
    Get predictions from rnn_model
    Calculate the final predictions
    Return final_predictions

#Step 4). Compliance Checking (agaisnt GDPR, CCPA, and NIST IoT...)
Function check_compliance(threats, threshold)
#Threshold is important as it is used to verify if IoT application is compliant
    checks if threats are below the Threshold
    Return compliance results

```

Figure 5: Pseudocode of Proposed Framework

```
Function regulatory_compliance_check(threats, threshold):
    Get compliance results from check_compliance
    Get GDPR compliance
    Get CCPA compliance
    Get NIST compliance
    Calculate overall compliance
    Return overall compliance

Function evaluate_against_gdpr(threats):
    Return GDPR compliance check result

Function evaluate_against_ccpa(threats):
    Return CCPA compliance check result

Function evaluate_against_nist(threats):
    Return NIST compliance check result

#Step 5). Policy Adjustment (I am usuing a Microsoft Word Doc)
Function load_policies(doc_path):
    Load policies from doc_path
    Return policies

Function save_policies(doc_path, policies):
    Save policies to doc_path

Function adjust_policies(compliance_results, policies):
    Create adjusted policies
    Save adjusted policies
    Return adjusted policies


#Step 6: Real-time Policy Enforcement and Monitoring
Function enforce_policies(policies):
    For each policy:
        If adjustment needed:
            Enforce policy adjustment
        Else:
            Indicate no adjustment needed

#Step 7: A Simple Alert Procedure/Mechanism
Function send_alert(message):
    print alert message/statement

# The final part of the code
# Main function to tie every step together
Function main():
    set file paths (dataset, policy docx) + regulatory frameworks
    Load and preporcess data
    Train models
    Detect threats
    Check compliance_results
    Load the initial policies (docx)
    Adjust policies based on compliance results
    Enforce policies
    Send alerts/alarms if adjustements are needed

# Runing the main Function
if __name__ == "__main__":
    main()
```

3.3.3 Entity Relationship Diagram

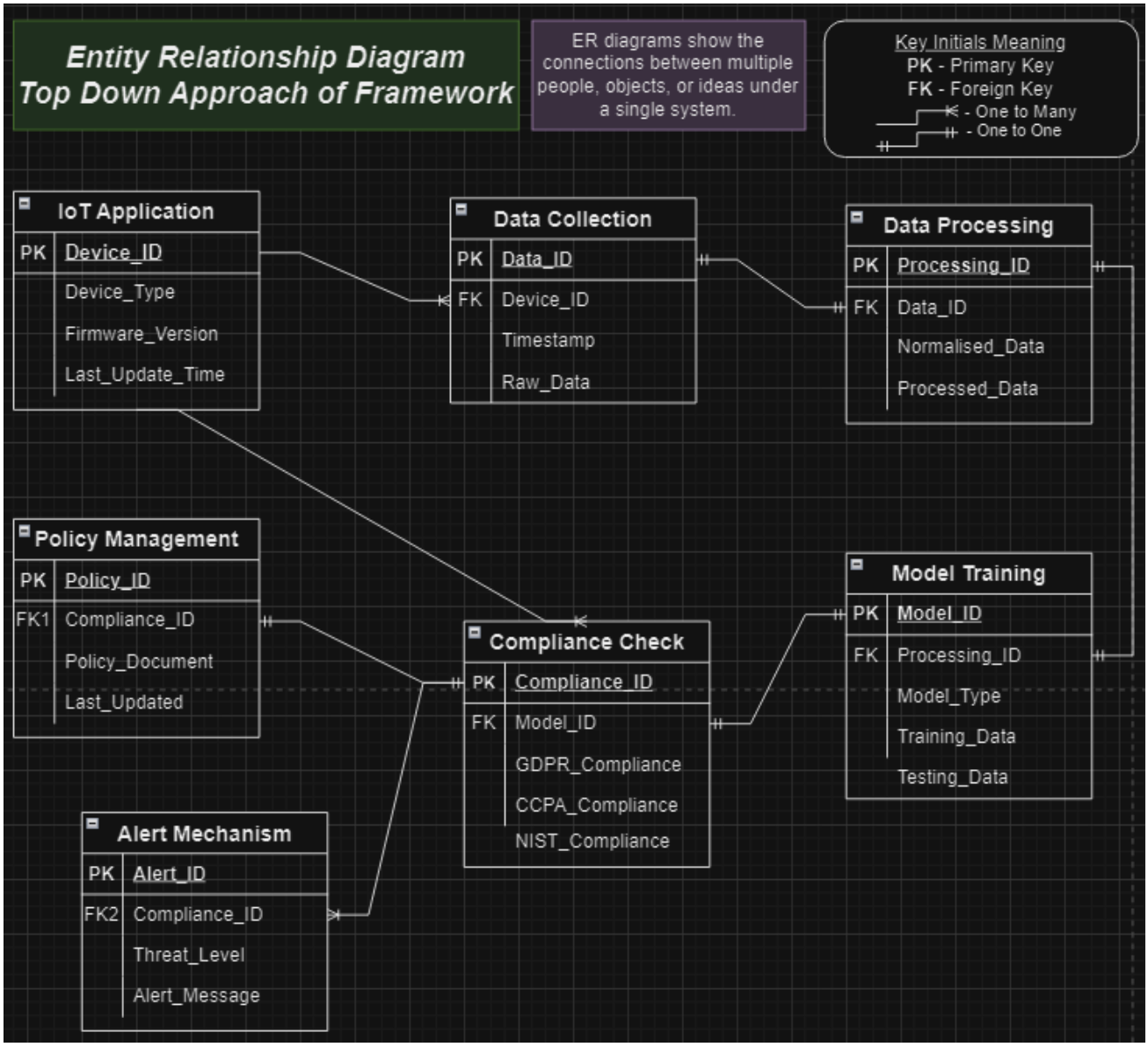


Figure 6: Entity Relationship Diagram

3.3.4 Activity Diagram

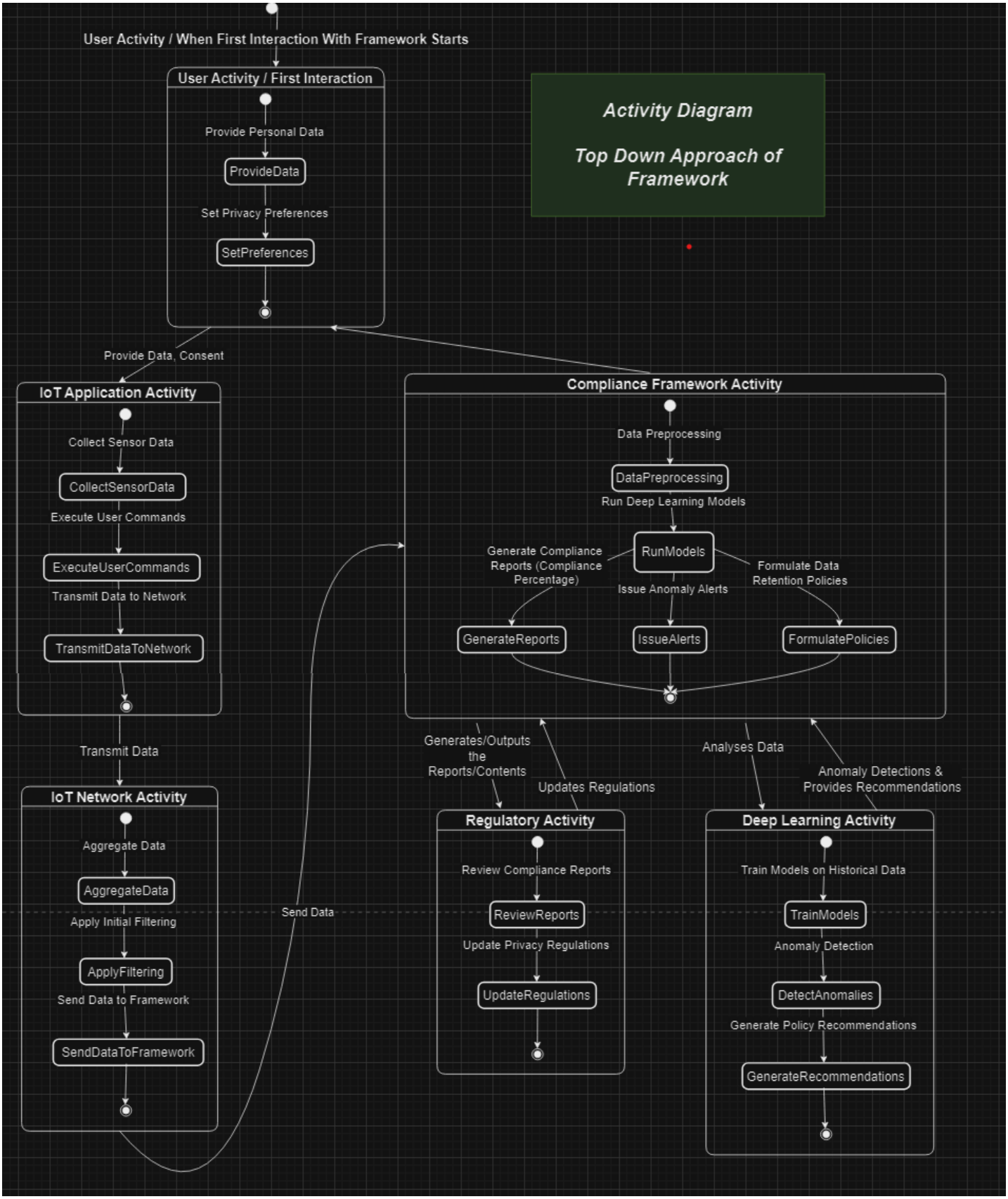


Figure 7: Activity Diagram

3.3.5 Visualisation of Entire Framework:

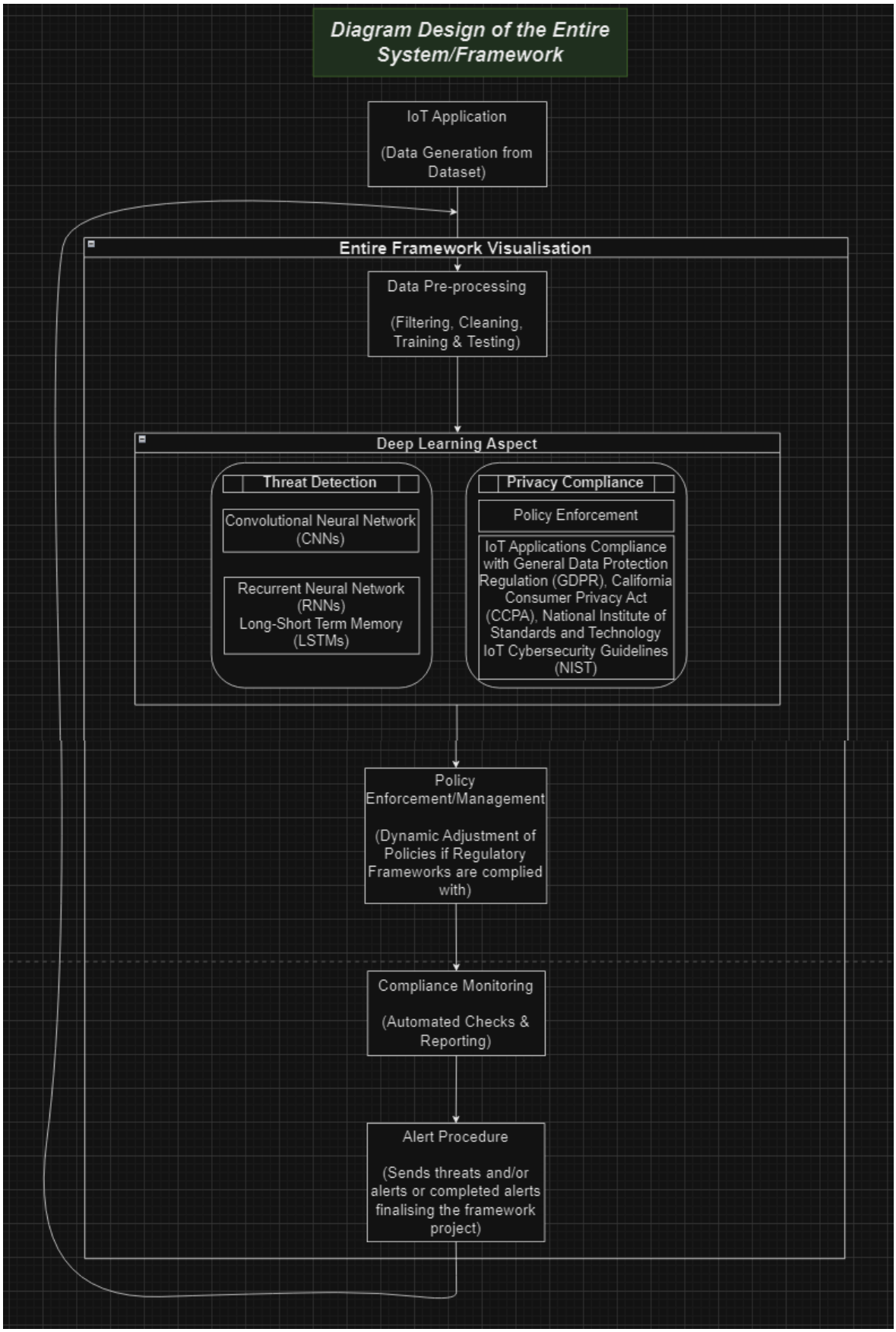


Figure 8: Visualisation of Entire Framework

Chapter 4: Implementation/Proposed Solution

Introduction

Using the requirements identified in Chapters 2 and 3 from the literature review and design stage, a suitable prototype of a Python program has been implemented and developed. This chapter summarises the implementation methods and technical processes used to fulfil the project requirements and how they have been executed.

The entire implementation of the Python programs, including all the neural network aspects was developed via Google Colab using many libraries such as Keras and TensorFlow with my Google Drive imported/integrated as seen in Figure 11 and Figure 9. The entire framework including the neural networks was trained on Huawei Matebook D15 AMD Ryzen 5 4500U with Radeon Graphics, (2.38 GHz), 6 Core(s) with 8GB RAM on a 64-bit operating system, x64-base processor.

The GitHub repository containing links to framework implementation and a ReadMe file with product description be found here: https://github.com/HamzaAhmed78629/MSc-Thesis-Proposed-Product.git

Chapter 4 Implementation code can be found here:
Deep_Learning_Framework_Implementation.ipynb

Chapter 5 Testing/Evaluation Python Code before ultimate completion can be found here:
Initial_Testing_Stage.ipynb

The final complete implementation of the Proposed Product i.e. Deep Learning-Based Privacy Compliance Framework from the Chapter 5 Testing/Evaluation can be found here:
Implementation_Completed_&_Testing_Another_IoT_Dataset.ipynb

4.1 Proposed Solution Implementation Structure

The proposed framework integrates two crucial components: threat detection utilising CNN and LSTM deep learning models and regulatory compliance checking based on standards (GDPR, CCPA, NIST IoT Cybersecurity Guidelines) as a baseline. Several key steps of the framework involve data collection and preprocessing, model development and training, compliance checking, dynamic policy adjustment, policy enforcement, and an alert mechanism that simply mentions if the IoT application can continue to process if compliant against regulatory frameworks, or if there are threats detected and either an adjustment in policies should be administered or further action is required.

4.2 Data Collection and Preprocessing

4.2.1 Dataset Description and Location

Firstly, I wanted to use a public dataset that is freely available to anyone and from a trusted website/source. The website Kaggle; a data science competition platform and online community for data scientists and machine learning practitioners under Google LLC provides

public datasets from various sources of devices including IoT devices with a Creative Commons license. The question is the CSV file 'HomeC.csv' which is available via https://www.kaggle.com/datasets/taranvee/smart-home-dataset-with-weather-information/data contains the readings of data for a length of 1 minute of 365 days from house appliances in kilowatts (kW). In addition, data was taken from a smart meter, as well as the weather conditions of that particular region. Having around 49,000 rows of data the following features it contains are 'Timestamp' and 'Transaction ID', application usage e.g. TV, refrigerator, microwave, energy consumption metrics, and weather information such as temperature, cloud cover, and humidity, etc.

Due to the time-series data from IoT devices in a smart home environment being genuine instead of a dummy dataset, I chose this for its relevance to IoT applications and its rich feature set, enabling detailed analysis for both privacy compliance evaluation and threat detection.

4.2.2 Data Loading, Data Transformation and Feature Selection

Once downloaded, this CSV file could then be uploaded to my Google account's OneDrive where other files can be added e.g. the 'Policies.docx' and 'AdjustedPolicies.docx' as seen in figure 9. The policy element will be discussed at the stage of the 'Dynamic Policy Adjustment' section of this implementation.

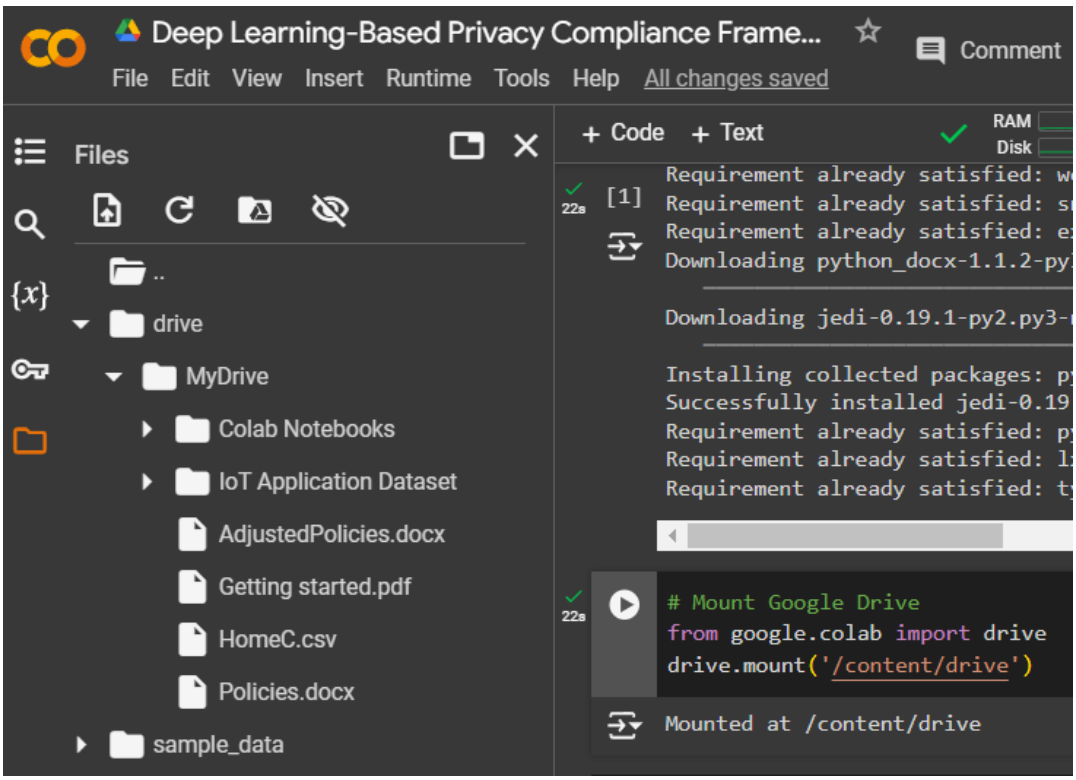


Figure 9: Mounting my Google Drive for Integration

Going onto the implementation of the proposed solution, certain required libraries will be needed using the '!pip' command. From Figure 10, the several Python packages commonly used in data science, machine learning, document processing, and interactive widgets are included such as 'pandas' for analysis and data manipulation, offering data structures like DataFrames, 'scikit-learn' for machine learning tasks, and TensorFlow for deep learning model development. It also utilised 'python-docx' for effective automation/integrating Word document creation and 'ipywidgets' that adds interactive elements in the form of buttons. The second command makes sure if there are any issues with the 'python-docx' library during the initial installation, it will be installed correctly the second time.

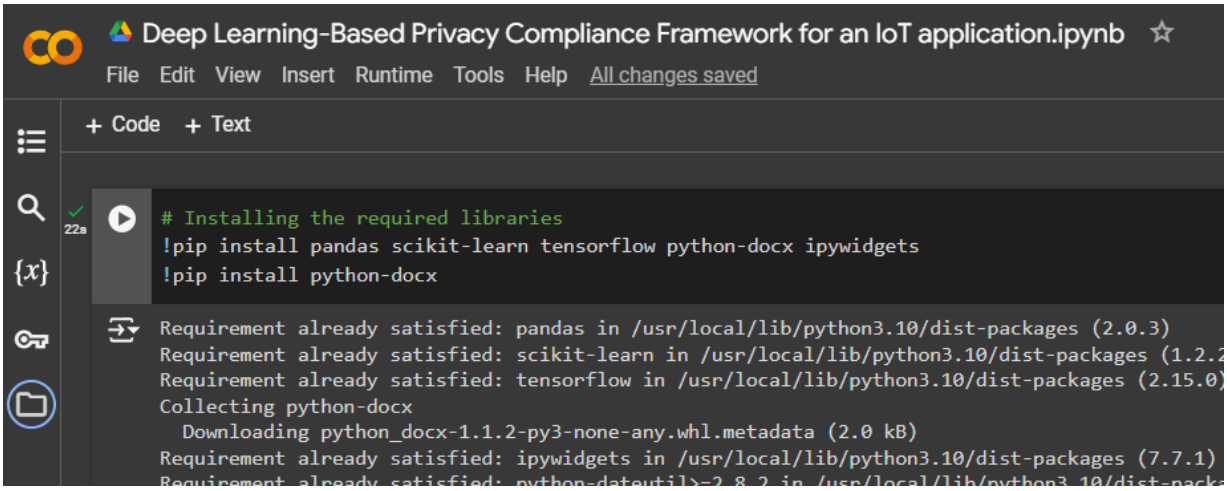


Figure 10: Installation of Required Libraries

From Figure 11, essential libraries are imported to build the deep learning-based privacy compliance framework. Pandas and NumPy handle the numerical operations and data manipulation while TensorFlow helps with the implementation of CNN and LSTM models. For the dataset preparation used for model training, data preprocessing tools like 'train_test_split, 'StandardScaler, and 'OneHotEncoder' from 'sklearn', otherwise known as 'Scikit-learn' which is a useful library for machine learning and includes various classifications, regression, and clustering algorithms, etc. 'Conv1D, 'Dense', and 'LSTM' with other various layers and sequel models define the neural networks. Additionally 'ipywidgets' creates an interactive user interface for example, buttons have been created as seen in Figure 24. 'python-docx' manages the policy documents, 'matplotlib' and 'seaborn' handle visualisation. All of these tools collectively provide efficiency in data processing, model building, and interaction from users for compliance checking.

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Conv1D, Flatten, MaxPooling1D, Dropout
from docx import Document
pd.set_option("display.max_columns", None)
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
pd.set_option('display.max_rows', None)
from datetime import datetime
import ipywidgets as widgets
from IPython.display import display, clear_output
import random
import warnings
```

Figure 11: Essential Libraries Importation

Moreover, the IoT dataset was loaded into this Python environment using pandas. The following code was used to allow and load the dataset from my Google Drive, as seen in Figure 9 and also in Figure 10. The research started through a thorough examination of the dataset to inspect for missing values and inconsistencies. Ensuring data integrity and no incomplete data points affect model performance, incomplete/missing values were addressed using pandas library's '.dropna()' function. Additionally, the dataset's structure was optimised by rejecting the final entry so that the program aligns the data length for processing.

Moving further, the time feature, which was originally represented as a Unix timestamp would need to be converted to a dataframe format to allow time-series analysis. Under this, the data was indexed by the time column. When looking at the output contents of this code, the transformation facilitated alignment of time-based features and for easier manipulation when detecting anomalies and patterns in the time-series data. This is seen in Figure 12.

```
# Loading data from IoT Dataset
data = pd.read_csv("/content/drive/MyDrive/HomeC.csv")

# Handling missing values
data = data.dropna()
data = data[:-1]

# Converting timestamp to datetime and set as index
data['time'] = pd.to_datetime(data['time'], unit='s')
data['time'] = pd.DatetimeIndex(pd.date_range('2016-01-01 05:00', periods=len(data), freq='min'))
data = data.set_index('time')

# Separating features and target
X = data.drop(columns=['use [kW]'])  # Features
Y = data['use [kW]']  # Target
```

Figure 12: Dataset Loading, Dataframe Conversion and Feature Selection

4.2.3 Data Preprocessing

From Figure 13, the dataset's diverse composition needed a multi-faceted preprocessing strategy. Including both categorical and numerical data, tailored techniques were implemented and applied for each data type. The icon and summary, representing weather conditions that are part of the categorical features underwent binary encoding to transform them into a machine-readable format and were one-hot encoded, while the numeric features were standardised using a StandardScaler. Combining the encoding process and scaling, a 'ColumnTransformer' was used. After the application of the 'ColumnTransformer', 'MinMaxScaler' was added for the data to be further scaled and to ensure all features were normalised between 0 and 1. To end this part of data preprocessing, the now pre-processed dataset was examined for any potential issues such as infinity values or 'NaN', ensuring clean coherent input data for CNN and LSTM model training.

```
# Converting 'cloudCover' to numeric
X['cloudCover'] = pd.to_numeric(X['cloudCover'], errors='coerce')

# Identifying numeric and categorical columns
numeric_features = X.select_dtypes(include=['float64', 'int64']).columns
categorical_features = ['icon', 'summary']

# Preprocessor: Scaling numeric data and encoding categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
    ])

# Applying preprocessing to the dataset
X_preprocessed = preprocessor.fit_transform(X)

# Additional Scaling
scaler = MinMaxScaler()
X_preprocessed = scaler.fit_transform(X_preprocessed)
print("Data is Loaded and preprocessed successfully")

# Checking for NaN or infinity in the preprocessed data
if np.isnan(X_preprocessed).sum() > 0 or np.isinf(X_preprocessed).sum() > 0:
    print("Warning: NaN or infinity values found in the preprocessed data.")
else:
    print("No NaN or infinity values found in the preprocessed data.")
```

Figure 13: Inside of Pipeline i.e. Data Processing

4.3 Splitting Data into Training and Testing Sets

4.3.1 Train-Test Split & Data Shaping for CNN and LSTM

Progressing forward, now that the pipeline has been created, the preprocessed dataset was split into training and testing sets using the 'train_test_split' function from scikit-learn. A 70-30 split was the way to go for this framework as having 70% of the data used for training the models and 30% for testing, offering a good balance between having adequate training data and reducing overfitting risks. Also, the 'random_state' parameter was set to 42, which made certain of the reproducibility of the results. Data needed to be reshaped due to the outcome of both the CNN and LSTM models' expected 3D input. This step essentially includes data to reshape to fit the input requirements of CNN and RNN (especially LSTM) models. Next step is to train the CCN and LSTM model to predict privacy threats in the test data and average the predictions getting the final threat detection

```
# Loading the dataset
global data, X_preprocessed, Y, X_train, X_test, y_train, y_test
```

Figure 14: Defining Variables Associated

```
X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, Y, test_size=0.3, random_state=42)
print(f"Train-test split done successfully")

# Reshaping data for CNN and LSTM
X_train_reshaped = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_reshaped = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
```

Figure 15: Data Splitting of Dataset

4.4 Model Development and Training

4.4.1 Convolutional Neural Network (CNN) Model

CNN was implemented by using the TensorFlow Keras Library. The reason for this was its ability to efficiently acquire and extract spatial features from grid-like data e.g. time series data from the IoT device used. As mentioned in the literature review, its architecture is structured to pinpoint patterns that could identify possible attack vectors and also to process sequential data. Originally set as 10 as for the epoch time to allow more time to train the model, it was brought down to 2 epochs using the training data, with validation on the testing data. The several key layers of CNNs architecture seen in Figure 16 are:

- Dense Layers: Its fully connected layers consist of 'Dense; layer with ReLU activation and 64 units. This is ensued by a final 'Dense' layer using a sigmoid activation function for binary classification. All assessing whether a threat is present or not.
- MaxPooling Layer: Maxpooling1D to reduce the dimensionality of the output from the convolutional layer.
- Flatten layer: To covert the 2D data in the dataset into 1D
- Input Layer: Its 'Conv1D' layer utilises 32 filters with 3 as the kernel size, joined with the ReLU activation function. To essentially identify potential threats in the IoT

application, this layer was responsible for scanning the time-series data to detect if there are patterns insinuating threats.

```
# Defining and Training the Convolutional Neural Network Model
def train_cnn_model(X_train, y_train, X_test, y_test):
    print("Training CNN Model...")
    cnn_model = Sequential([
        Conv1D(32, 3, activation='relu', input_shape=(X_train.shape[1], 1)),
        MaxPooling1D(2),
        Conv1D(64, 3, activation='relu'),
        MaxPooling1D(2),
        Flatten(),
        Dense(64, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    cnn_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    cnn_model.fit(X_train, y_train, epochs=2, batch_size=32, validation_data=(X_test, y_test),
    return cnn_model
```

Figure 16: CNN Model Implementation

4.4.2 Long Short-Term Memory (LSTM) Model

The other and more flexible model alongside CNN was LSTM because of its efficiency in handling sequential data, both critical for maintaining privacy compliance in IoT applications and real-time threat detection. Just as CNN, this model was trained on the same dataset for two epochs to reduce time and allow more flexibility and the same efficiency as the higher epoch times and learn temporal patterns that are critical with the IoT data for accurate threat detection. Similarly to CNN, the architecture consists of:

- Two LSTM layers: Starting off as one with 'return_sequences=True' ensuring the second LSTM layer takes in sequential data, succeeded by another LSTM layer.
- Dense output layer: Same as CNN when using sigmoid activation for binary classification (threat or not threat).
- Dropout layers: Applied to each LSTM layer and intended to prevent overfitting.

```
# Defining and Training the Long-Short Term Memory Model
def train_lstm_model(X_train, y_train, X_test, y_test):
    print("Training LSTM Model...")
    lstm_model = Sequential([
        LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)),
        Dropout(0.2),
        LSTM(50, return_sequences=False),
        Dropout(0.2),
        Dense(1, activation='sigmoid')
    ])
    lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    lstm_model.fit(X_train, y_train, epochs=2, batch_size=32, validation_data=(X_test, y_test)
    return lstm_model

# Training the models
cnn_model = train_cnn_model(X_train_reshaped, y_train, X_test_reshaped, y_test)
lstm_model = train_lstm_model(X_train_reshaped, y_train, X_test_reshaped, y_test)
```

Figure 17: LSTM Model Implementation

Finally, the last two lines from Figure 17 trains CNN and LSTM models using the reshaped data (X_train_reshaped, y_train) and evaluate them on the test data (X_test_reshaped, y_test).

4.5 Threat Detection Using Deep Learning Models

4.5.1 Detecting Threats & Threshold for Detection

Leveraging both LSTMs Long-term sequence processing and CNNs feature extraction capabilities, combining both strengths from its models to enhance threat detection, the threat detection operation within the framework works by once both models generate the predictions for the test data, their outputs are averaged to produce a final threat score. To classify where a threat is found and detected, a threshold of 0.5 was applied as a benchmark. In the real world, there would be more fixed terms, procedures, and precautions necessitated. The outputs of threat detection can be found in the '4.9 Results of Framework Implementation'.

```
# Detecting threats using CNN and LSTM models
def detect_threats(cnn_model, lstm_model, X_test, threshold=0.5):
    print("Detecting threats...")
    cnn_predictions = cnn_model.predict(X_test)# making the predictions
    lstm_predictions = lstm_model.predict(X_test)
    #cnn_predictions = np.nan_to_num(cnn_predictions, nan=0.5)# handling the new values
    #lstm_predictions = np.nan_to_num(lstm_predictions, nan=0.5)
    final_predictions = (cnn_predictions + lstm_predictions) / 2# combining the predictions
    #scaler = MinMaxScaler()# normalsing predictions to [0, 1]
    #final_predictions_normalized = scaler.fit_transform(final_predictions)
    threats = final_predictions > threshold
    print("Final predictions generated and threat detection completed")
    return threats, final_predictions

# Detecting threats and getting the final predictions
final_predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)
print(final_predictions)
```

Figure 18: Detecting Threats using CNN and LSTM models

4.6 Regulatory Compliance Checking

4.6.1 Compliance Mechanism

The thesis project title implied that there would be a compliance-checking feature against regulatory frameworks and they were GDPR, CCPA, and NIST IoT Cyber Security Guidelines. In this implementation, they each represented a specific threshold from the 'check_compliance()' and 'regulatory_compliance_check()' functions as I wanted to not run the risk of failure in complying with certain frameworks if real information about those regulatory frameworks would be included. From the previous step, if a detected threat exceeds these thresholds, the framework will flag it as non-compliant. From Figure 19, the thresholds are outlined as follows and return a dictionary with Boolean arrays for each standard and overall compliance:

- GDPR: Threats must have a probability below 0.2

- CCPA: Threats must have a probability below 0.3
- NIST: Threats must have a probability below 0.4

4.6.2 Processing Compliance Results

Furthermore, once the compliance results are fully processed and displayed, sowing a percentage of data points that meet the criteria for each regulatory framework aforementioned (0% is fully non-compliant where 100% means fully compliant against the regulations i.e. the specific threshold). This puts it through the process and ensures IoT data handling aligns with legal requirements and standards, aiming to mitigate legal risks.

```python
# Compliance checking against regulatory compliance (GDRP, CCPA, NIST)
def check_compliance(threats, threshold=0.5):
    compliant = threats < threshold
    return compliant

def evaluate_against_gdpr(threats):
    return threats < 0.2

def evaluate_against_ccpa(threats):
    return threats < 0.3

def evaluate_against_nist(threats):
    return threats < 0.4

def regulatory_compliance_check(threats, predictions):
    general_compliance = predictions < 0.7
    gdpr_compliance = predictions < 0.6
    ccpa_compliance = predictions < 0.5
    nist_compliance = predictions < 0.4
    overall_compliance = general_compliance & gdpr_compliance & ccpa_compliance & nist_compliance
    return {
        "General Compliance": general_compliance,
        "GDPR Compliance": gdpr_compliance,
        "CCPA Compliance": ccpa_compliance,
        "NIST Compliance": nist_compliance,
        "Overall Compliance": overall_compliance
    }

def process_compliance_results(compliance_results):
    for standard, result in compliance_results.items():
        compliant_count = np.sum(result)
        total_count = len(result)
        compliance_percentage = (compliant_count / total_count) * 100
        print(f"{standard}: {compliance_percentage:.2f}% compliant")

# Compliance check
threats, predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)
compliance_results = regulatory_compliance_check(threats, predictions)
print(f"Compliance Check Results: {compliance_results}")
```

Figure 19: Compliance Checking Implementation

4.7 Dynamic Policy Adjustment, Enforcing Policies & Alert Mechanism

4.7.1 Loading and Adjusting Policies and Saving Adjusted Policies

In this part of the implementation, the framework adapts to non-compliance by loading and then dynamically adjusting the policies. However, it does not necessarily change the policy but enables the program to tell the user 'Adjust Policy'. This can be found in '4.9 Results of Framework Implementation from the contents of the output. The policies seen in Figure 20 are stored in a word, and loaded via the Python-docx library. Based on the compliance results and regarding the data within the IoT dataset, the policies that fail to meet regulatory standards are flagged for adjustment for example, this IoT device manufacturer will need to adjust that specific policy or policies to adhere to regulatory standards.

Additionally, the adjusted policies are automatically saved back to a new document called 'AdjustedPolicies.docx' as seen in Figure 20. This was achieved by creating an empty dictionary to store the adjusted policies as seen in Figure 21. The function 'adjust_policies()' taking two parameters 'policies', and 'complaince_results' is added to review and modify security policies based on compliance results. Using 'zip()', a loop iterates over the policies and their associated compliance results concurrently. If a policy is not compliant, from Figure 20 on the right, it's marked as 'Action Required', on the other hand, if a policy is complaint, it's noted as 'No Policy Adjustment Required'

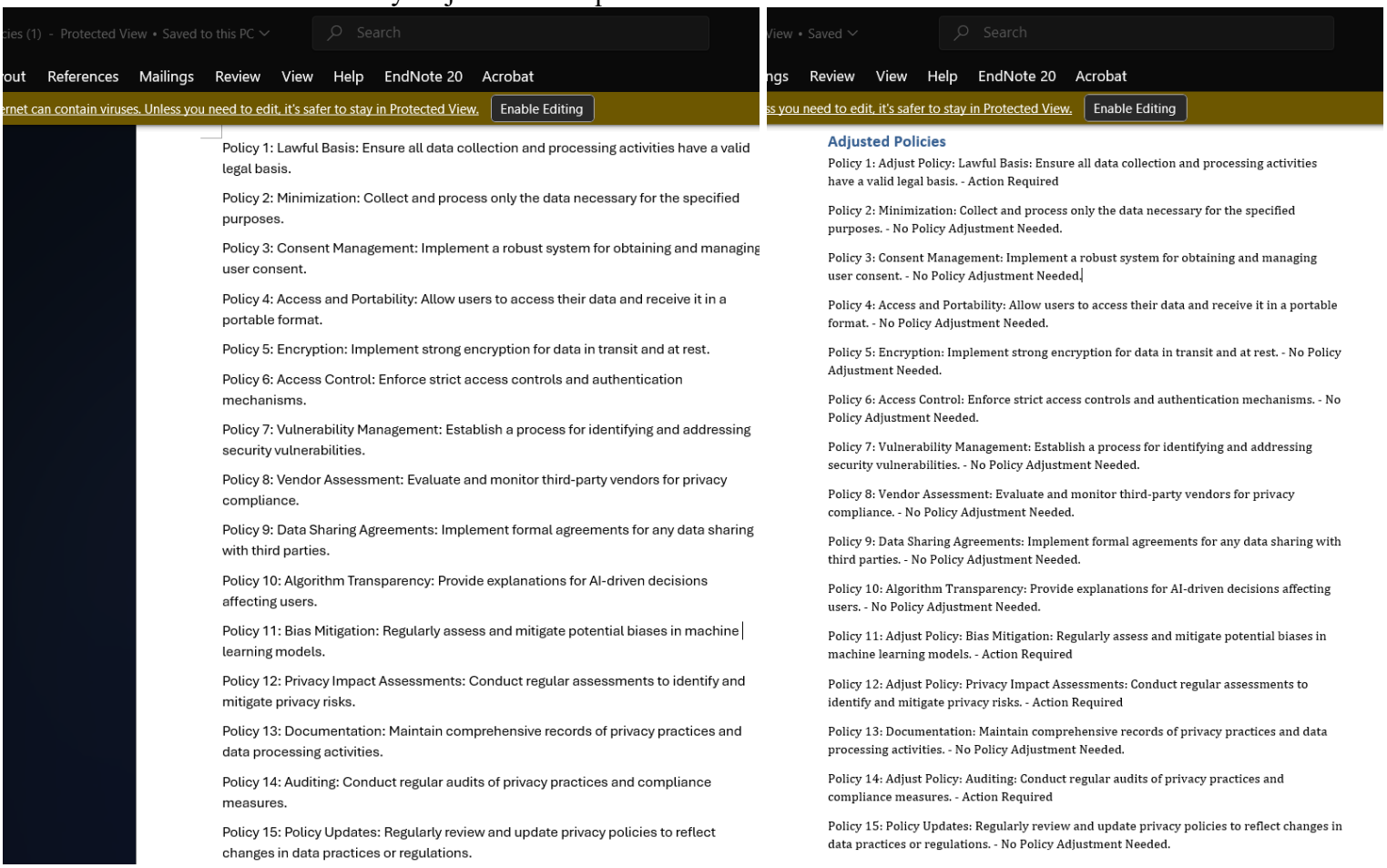


Figure 20: Policies.docx and AdjustedPolicies.docx Word Documents Contents

```python
# Load, Adjust, and Save Policies
def load_policies(doc_path):
    doc = Document(doc_path)
    policies = {}
    for para in doc.paragraphs:
        if para.text and ":" in para.text:
            key_value = para.text.split(":", 1)
            if len(key_value) == 2:
                key, value = key_value
                policies[key.strip()] = value.strip()
            else:
                policies[key_value[0].strip()] = ""
        elif para.text:# Handles paragraphs without a colon
              policies[para.text.strip()] = ""
              #print(policies)
    return policies

def save_policies(policies, doc_path):
    doc = Document()
    doc.add_heading("Adjusted Policies", level=1)
    for key, value in policies.items():
        doc.add_paragraph(f"{key}: {value}")
    doc.save(doc_path)

def adjust_policies(policies, compliance_results):
    adjusted_policies = {}
    for i, (policy, compliant) in enumerate(zip(policies.items(), compliance_results)):
        key, value = policy
        if not compliant:
            adjusted_policies[key] = f"Adjust Policy: {value} - Action Required"
        else:
            adjusted_policies[key] = f"{value} - No Policy Adjustment Needed."
    save_policies(adjusted_policies, "/content/drive/MyDrive/AdjustedPolicies.docx")
    return adjusted_policies
```

Figure 21: Implementation of Loading, Adjusting, and Saving Policies

On a related topic the function ‘compliance_results = […..]’ creates a list of compliance results and for each policy, ‘random.choice([True, False])’ selects either if the policy is True (compliant) or False (non-compliant). A loop is run using an underscore ‘_’ for each policy in ‘initial_policies’ and is used as a placeholder variable because there isn’t a necessity to use the loop variable. The output/result is a list of boolean values either True or false representing the compliance status of every individual policy.

```python
# Load the initial policies
policies_file_path = "/content/drive/MyDrive/Policies.docx"
output_file_path = "/content/drive/MyDrive/AdjustedPolicies.docx"
initial_policies = load_policies(policies_file_path)
print(f"Loaded Policies: {initial_policies}\n")

compliance_results = [random.choice([True, False]) for _ in range(len(initial_policies))]
print(f"Compliance Results: {compliance_results}")

# Adjust policies
adjusted_policies = adjust_policies(initial_policies, compliance_results)
print(f"Adjusted Policies: {adjusted_policies}\n")

#adjust_button.on_click(on_adjust_button_clicked)
```

```
Loaded Policies: {'Policy 1': 'Lawful Basis: Ensure all data collection and processing activities have a valid legal basis.', 'Policy 2': 'Minimization: Collect and process

Compliance Results: [False, True, True, True, True, True, True, True, True, True, False, False, True, False, True]
Adjusted Policies: {'Policy 1': 'Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required', 'Policy 2':
```

Figure 22: Snippet of Privacy Compliance Assessment and Adjustment Process

4.7.2 Policy Enforcement and Alert System

When adjustments of policies have been mentioned and notified to the user, the framework/system enforces them and sends alerts if imperative. As you can see in Figures 20 and 22, if a policy is marked for adjustment, an alert is activated to inform the user of compliance breaches or possible threats if necessary. Inside of the 'enforce_policies()' function from Figure 23, alerts are sent using the 'send_alert()' function to print notifications for the user using the framework.

Ultimately, these features regarding loading and adjusting policies, policy enforcement, and alert system enables the deep learning-based privacy compliance framework to remain up-to-date with changing compliance requirements and may even to regulatory violations in real-time.

```
# Enforce policies
def enforce_policies(policies):
    for policy, action in policies.items():
        if "Adjust Policy" in action:
            print(f"Enforcing Policy: {action}...")
        else:
            print(f"{policy}: No adjustment needed. Data is processed in compliance.")

# Alert mechanism
def send_alert(message):
    print(f"ALERT: {message}")

# Sending alerts for non-compliance and based on policy adjustments
for policy, action in adjusted_policies.items():
    if "Adjust Policy" in action:
        send_alert(f"Threats detected. {action}")
    else:
        print(f"{policy} is compliant and can continue processing data.")

# Enforcing the policies
enforce_policies(adjusted_policies)
```

Figure 23: Policy Enforcement and Alert Mechanism

4.8 Integration and User Interaction

4.8.1 User interface Using 'ipywidgets

Throughout the implementation, using 'ipywidgets' an interactive user interface was included which meant part of the program had to be changed around for example, certain functions to do with loading data and compliance checking were incorporated and contained inside certain buttons. It allows users to load preprocessed data, check compliance, adjust policies, and enforce them through a set of buttons seen in Figure 24. Included buttons can be beneficial and enhance the user experience by making the framework more accessible and dynamic. Inside of the 'load_data()' function from Figures 25 and 26, it loads the IoT dataset from a CSV file, preprocesses the dataset by handling missing values, converts timestamps for clearer and easier viewing of data, and separates the features and target variables.

Additionally, it then goes on to apply further preprocessing steps including encoding categorical data and scaling numeric data.

Furthermore, after preprocessing is completed, the program checks for any NaN or infinity values inside of the data, performs a train-test split of 70-30 to the dataset, and finally supplies feedback on the success of each step through print statements as seen in 4.9 Results of Framework Implementation

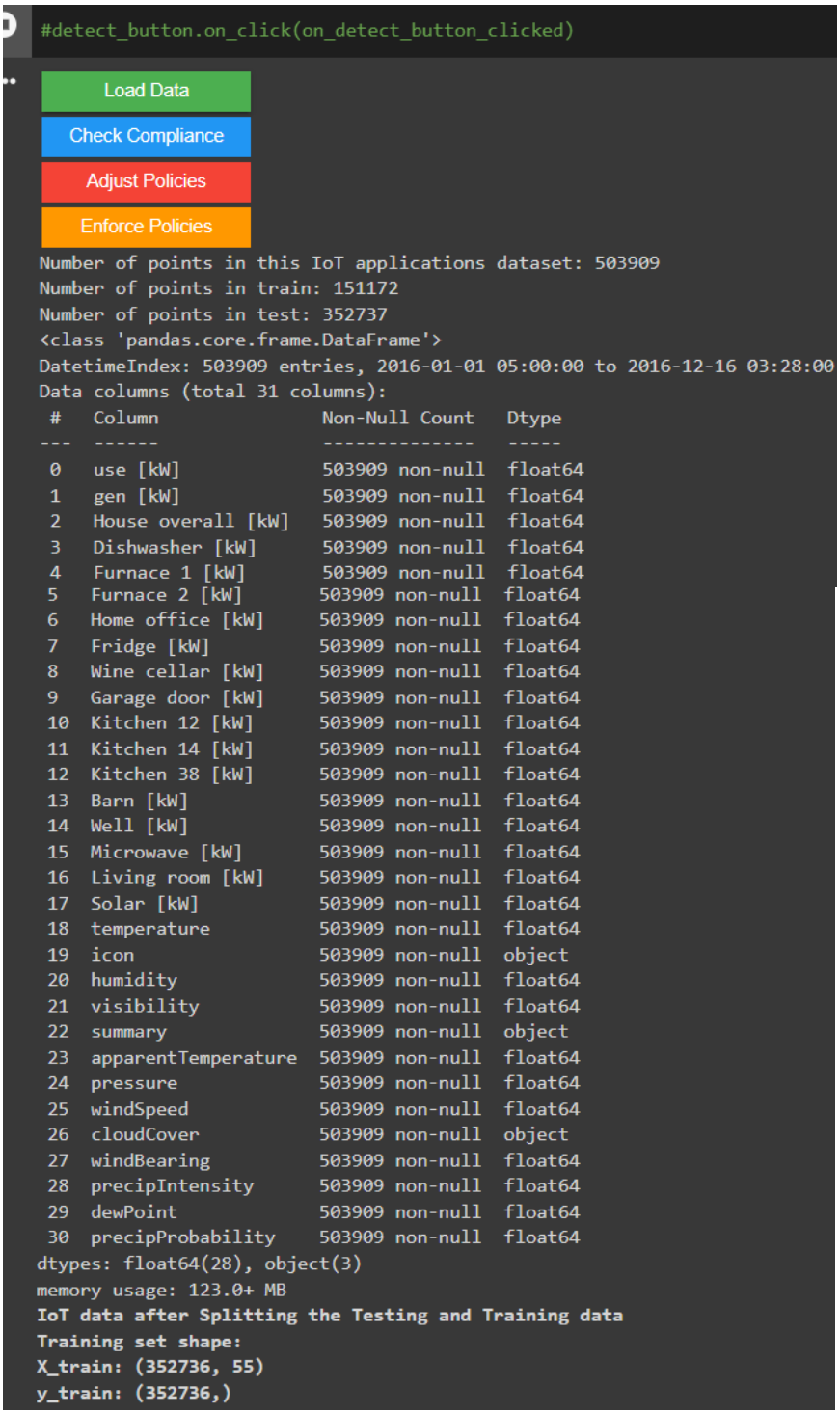


Figure 24: UI of Interactive Buttons

```python
# Widgets for interaction
load_button = widgets.Button(description="Load Data", button_style='success')
#train_button = widgets.Button(description="Train Models", button_style='info')
#detect_button = widgets.Button(description="Detect Threats", button_style='warning')
check_compliance_button = widgets.Button(description="Check Compliance", button_style='primary')
adjust_button = widgets.Button(description="Adjust Policies", button_style='danger')
enforce_button = widgets.Button(description="Enforce Policies", button_style='warning')

# Output widget to display results
output = widgets.Output()

# Display the interface
display(widgets.VBox([load_button, check_compliance_button, adjust_button, enforce_button, output]))


# loading data functions
def load_data(b):
    with output:
        output.clear_output()
        print("Loading data.....")

        # Loading the dataset
        global data, X_preprocessed, Y, X_train, X_test, y_train, y_test
        # Loading data from IoT Dataset
        data = pd.read_csv("/content/drive/MyDrive/HomeC.csv")

        # Handling missing values
        data = data.dropna()
        data = data[:-1]
```

Figure 25: Part 1 of Python Code for Interactive Buttons

```python
        # Separating features and target
        X = data.drop(columns=['use [kW]'])  # Features
        Y = data['use [kW]']  # Target

        # Displays the first few rows of X and Y
        print("Features (X):")
        print(X.head())

        print("Target (Y):")
        print(Y.head())

        # Converting 'cloudCover' to numeric
        X['cloudCover'] = pd.to_numeric(X['cloudCover'], errors='coerce')

        # Identifying numeric and categorical columns
        numeric_features = X.select_dtypes(include=['float64', 'int64']).columns
        categorical_features = ['icon', 'summary']

        # Preprocessor: Scaling numeric data and encoding categorical data
        preprocessor = ColumnTransformer(
            transformers=[
                ('num', StandardScaler(), numeric_features),
                ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features)
            ])

        # Applying preprocessing to the dataset
        X_preprocessed = preprocessor.fit_transform(X)

        # Additional Scaling
        scaler = MinMaxScaler()
        X_preprocessed = scaler.fit_transform(X_preprocessed)
        print("Data is Loaded and preprocessed successfully")

        # Checking for NaN or infinity in the preprocessed data
        if np.isnan(X_preprocessed).sum() > 0 or np.isinf(X_preprocessed).sum() > 0:
            print("Warning: NaN or infinity values found in the preprocessed data.")
        else:
            print("No NaN or infinity values found in the preprocessed data.")

        X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, Y, test_size=0.3, random_state=42)
        print(f"Train-test split done successfully")


# Attaching the function to the load button
load_button.on_click(load_data)
```

Figure 26: Part 2 of Python Code for Interactive Buttons

```
# Attaching the function to the load button
load_button.on_click(load_data)

size = int(len(data)*0.3)
train = data[:size]
test = data[size:]
print('Number of points in this IoT applications dataset:', len(data))
print('Number of points in train:', len(train))
print('Number of points in test:', len(test))
data.info()
data.head()

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, Y, test_size=0.3, random_state=42)

# Printing statement of testing and training data is split
print("\033[1m" + "IoT data after Splitting the Testing and Training data")

# Outputing the shapes of the training and testing sets
print("Training set shape:")
print(f"X_train: {X_train.shape}")
print(f"y_train: {y_train.shape}")

print("\nTesting set shape:")
print(f"X_test: {X_test.shape}")
print(f"y_test: {y_test.shape}")

print("\033[0m")

# Reshaping data for CNN and LSTM
X_train_reshaped = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_reshaped = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
```

Figure 27: Data Loading, Preprocessing and Train-Test Split for Dataset

```
def on_check_compliance_button_clicked(b):
    with output:
        output.clear_output()
        global compliance_results
        # Compliance check
        threats, predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)
        compliance_results = regulatory_compliance_check(threats, predictions)
        process_compliance_results(compliance_results)
        #print(f"Compliance Check Results: {compliance_results}")

check_compliance_button.on_click(on_check_compliance_button_clicked)
```

Figure 28: Part 3 of Python Code for Interactive Buttons, Check Compliance Button

The implementation in terms of user interaction through buttons was successful for training CNN and LSTM models however, this caused my computer to crash due to the significant processing power needed for handling the large IoT dataset which contains 503909 data points seen in Figure 24 and also its complexity with the models. As a result, the model training and detecting threats for its IoT application were left to be handled automatically once the program ran in Google Colab. This allowed the framework to run smoothly, and efficiently, enabling the real-time interaction and functionality that were initially intended.

```
#def train_models(b):
    #with output:
        #output.clear_output()
        #print("Training models.....")

        # Spliting the data
        #global X_train, X_test, y_train, y_test, X_train_reshaped, X_test_reshaped, cnn_model, lstm_model
        # Train-test split
        #X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, Y, test_size=0.3, random_state=42)

        # Reshape data for CNN and LSTM
        #X_train_reshaped = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
        #X_test_reshaped = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

        # Training the models
        #cnn_model = train_cnn_model(X_train_reshaped, y_train, X_test_reshaped, y_test)
        #print("CNN model trained.")
        #lstm_model = train_lstm_model(X_train_reshaped, y_train, X_test_reshaped, y_test)
        #print("LSTM model trained.")

#train_button.on_click(train_models)

#def on_detect_button_clicked(b):
    #with output:
        #output.clear_output()
        #global final_predictions
        # Detecting threats and getting the final predictions
        #final_predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)
        #print("Detecting threats..... Final predictions generated")

#detect_button.on_click(on_detect_button_clicked)
```

Figure 29: Commented Python Code for Train Model and Detect Threats Buttons

4.9. Results of Framework Implementation

When talking about the results of my implementation, the main method contributed to and tied together the results of my proposed product, essentially creating a deep learning-based privacy compliance framework for an IoT application. From what has been written throughout the implementation stage of this report, including the key stages of data preprocessing, model training, threat detection, and compliance checking. Once data is loaded from my Google Drive, preprocessed and split its IoT data into training and testing sets by clicking the 'Load Button' UI from Figure 24, the training data contained 352,736 samples while the test data contained 151,173. Reshaping the IoT data, both LSTM and CNN models were trained through 2 epochs. However, there were convergence issues as the result showed NaN values for both training and validation loss. Despite this, the framework gave the desired and necessary results indicated in Figures 31 to 33.

After training, the threat detection merged the predictions from both models. Subsequently, when the test data was evaluated for compliance providing its evaluation, none of the data points adhered to the requirements set by DFPR, CCPA, or NIST where a threshold of 0.5, for example, was implemented for security reasons. This led to 0% compliance across all regulatory standards.

Proceeding with this, the framework loaded the predefined document containing 15 privacy policies, aimed to ensure adherence to privacy regulations. The indication was that upon the evaluation, the compliance resits revealed required revisions and adjustments in policies, with 10 out of 15 policies flagged and needing corrective action. Policies that were complaint regarding categories of security processes were Encryption, and Vulnerability Management, whereas others that were then flagged for the necessary adjustments included Consent Management, Minimisation, and Lawful Basis.

Thereupon adjusting these policies, the framework enforced the obligatory changes and triggered/generated alerts for instances of non-compliance. The several policies met the requirements of standards (threshold) and were processed without further modifications or adjustments but policies such as Algorithm Transparency, Access Control, and Vendor Assessment prompted immediate alerts and notified the user of the potential threats/ risks and the need for corrective actions stating 'Adjust Policy' and 'Action Required'.

To capitalise and conclude, the results from the framework implementation spotlighted the importance and necessity for optimisation of the CNN and LSTM models to elevate the facet of threat detection, together with the obligation for dynamic policy adjustment to validate and ensure regulatory compliance in IoT applications.

```python
# Main function to tie everything together
def main():
    # Preprocessing the data, training models, etc. #final_predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)
    X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, Y, test_size=0.3, random_state=42)
    # Printing statement of testing and training data is split
    print("\033[1m" + "IoT data points after Splitting the Testing and Training data")
    print("Training set shape:")
    print(f"X_train: {X_train.shape}")
    print(f"y_train: {y_train.shape}")
    print("\nTesting set shape:")
    print(f"X_test: {X_test.shape}")
    print(f"y_test: {y_test.shape}")
    X_train_reshaped = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
    X_test_reshaped = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
    cnn_model = train_cnn_model(X_train_reshaped, y_train, X_test_reshaped, y_test)
    lstm_model = train_lstm_model(X_train_reshaped, y_train, X_test_reshaped, y_test)
    threats, predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)
    compliance_results = regulatory_compliance_check(threats, predictions)
    #compliance_results = regulatory_compliance_check(final_predictions)
    process_compliance_results(compliance_results)
    policies_file_path = "/content/drive/MyDrive/Policies.docx"
    initial_policies = load_policies(policies_file_path)
    print(f"Loaded Policies: {initial_policies}\n")
    compliance_results = [random.choice([True, False]) for _ in range(len(initial_policies))]
    print(f"Compliance Results: {compliance_results}")
    adjusted_policies = adjust_policies(initial_policies, compliance_results)
    print(f"Adjusted Policies: {adjusted_policies}\n")
    print_policies(initial_policies, "Initial Policies")
    print_policies(adjusted_policies, "Adjusted Policies")
    enforce_policies(adjusted_policies)
    # Sending alerts for non-compliance and based on policy adjustments
    for policy, action in adjusted_policies.items():
        if "Adjust Policy" in action:
            send_alert(f"Threats detected. {action}")
        else:
            print(f"{policy} is compliant and can continue processing data.")

if __name__ == "__main__":
    main()
```

Figure 30: Main Method

```
if __name__ == "__main__":
    main()
```

```
IoT data points after Splitting the Testing and Training data
Training set shape:
X_train: (352736, 55)
y_train: (352736,)

Testing set shape:
X_test: (151173, 55)
y_test: (151173,)
Training CNN Model...
Epoch 1/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 69s 6ms/step - accuracy: 8.0401e-06 - loss: nan - val_accuracy: 0.0000e+00 - val_loss: nan
Epoch 2/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 67s 6ms/step - accuracy: 1.9325e-06 - loss: nan - val_accuracy: 0.0000e+00 - val_loss: nan
Training LSTM Model...
Epoch 1/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 543s 49ms/step - accuracy: 1.6054e-06 - loss: nan - val_accuracy: 0.0000e+00 - val_loss: nan
Epoch 2/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 533s 48ms/step - accuracy: 4.7516e-06 - loss: nan - val_accuracy: 0.0000e+00 - val_loss: nan
Detecting threats...
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 10s 2ms/step
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 71s 15ms/step
Final predictions generated and threat detection completed
General Compliance: 0.00% compliant
GDPR Compliance: 0.00% compliant
CCPA Compliance: 0.00% compliant
NIST Compliance: 0.00% compliant
Overall Compliance: 0.00% compliant
Loaded Policies: {'Policy 1': 'Lawful Basis: Ensure all data collection and processing activities have a valid legal basis.', 'Policy 2': 'Minimization: Collect and process or

Compliance Results: [False, False, False, False, True, False, True, False, False, False, False, True, False, True, True]
Adjusted Policies: {'Policy 1': 'Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required', 'Policy 2': 'A
```

Figure 31: Part 1 of Main Method Results

```
if __name__ == "__main__":
    main()
Compliance Results: [False, False, False, False, True, False, True, False, False, False, False, True, False, True, True]
Adjusted Policies: {'Policy 1': 'Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required

Initial Policies:
  Policy 1: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis.
  Policy 2: Minimization: Collect and process only the data necessary for the specified purposes.
  Policy 3: Consent Management: Implement a robust system for obtaining and managing user consent.
  Policy 4: Access and Portability: Allow users to access their data and receive it in a portable format.
  Policy 5: Encryption: Implement strong encryption for data in transit and at rest.
  Policy 6: Access Control: Enforce strict access controls and authentication mechanisms.
  Policy 7: Vulnerability Management: Establish a process for identifying and addressing security vulnerabilities.
  Policy 8: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance.
  Policy 9: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties.
  Policy 10: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users.
  Policy 11: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models.
  Policy 12: Privacy Impact Assessments: Conduct regular assessments to identify and mitigate privacy risks.
  Policy 13: Documentation: Maintain comprehensive records of privacy practices and data processing activities.
  Policy 14: Auditing: Conduct regular audits of privacy practices and compliance measures.
  Policy 15: Policy Updates: Regularly review and update privacy policies to reflect changes in data practices or regulations.

Adjusted Policies:
  Policy 1: Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required
  Policy 2: Adjust Policy: Minimization: Collect and process only the data necessary for the specified purposes. - Action Required
  Policy 3: Adjust Policy: Consent Management: Implement a robust system for obtaining and managing user consent. - Action Required
  Policy 4: Adjust Policy: Access and Portability: Allow users to access their data and receive it in a portable format. - Action Required
  Policy 5: Encryption: Implement strong encryption for data in transit and at rest. - No Policy Adjustment Needed.
  Policy 6: Adjust Policy: Access Control: Enforce strict access controls and authentication mechanisms. - Action Required
  Policy 7: Vulnerability Management: Establish a process for identifying and addressing security vulnerabilities. - No Policy Adjustment Needed.
  Policy 8: Adjust Policy: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance. - Action Required
  Policy 9: Adjust Policy: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties. - Action Required
  Policy 10: Adjust Policy: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users. - Action Required
  Policy 11: Adjust Policy: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models. - Action Required
  Policy 12: Privacy Impact Assessments: Conduct regular assessments to identify and mitigate privacy risks. - No Policy Adjustment Needed.
  Policy 13: Adjust Policy: Documentation: Maintain comprehensive records of privacy practices and data processing activities. - Action Required
  Policy 14: Auditing: Conduct regular audits of privacy practices and compliance measures. - No Policy Adjustment Needed.
  Policy 15: Policy Updates: Regularly review and update privacy policies to reflect changes in data practices or regulations. - No Policy Adjustment Needed.
```

Figure 32: Part 2 of Main Method Results

```
  Policy 14: Auditing: Conduct regular audits of privacy practices and compliance measures. - No Policy Adjustment Needed.
  Policy 15: Policy Updates: Regularly review and update privacy policies to reflect changes in data practices or regulations. - No Policy Adjustment Needed.

Enforcing Policy: Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required...
Enforcing Policy: Adjust Policy: Minimization: Collect and process only the data necessary for the specified purposes. - Action Required...
Enforcing Policy: Adjust Policy: Consent Management: Implement a robust system for obtaining and managing user consent. - Action Required...
Enforcing Policy: Adjust Policy: Access and Portability: Allow users to access their data and receive it in a portable format. - Action Required...
Policy 5: No adjustment needed. Data is processed in compliance.
Enforcing Policy: Adjust Policy: Access Control: Enforce strict access controls and authentication mechanisms. - Action Required...
Policy 7: No adjustment needed. Data is processed in compliance.
Enforcing Policy: Adjust Policy: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance. - Action Required...
Enforcing Policy: Adjust Policy: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties. - Action Required...
Enforcing Policy: Adjust Policy: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users. - Action Required...
Enforcing Policy: Adjust Policy: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models. - Action Required...
Policy 12: No adjustment needed. Data is processed in compliance.
Enforcing Policy: Adjust Policy: Documentation: Maintain comprehensive records of privacy practices and data processing activities. - Action Required...
Policy 14: No adjustment needed. Data is processed in compliance.
Policy 15: No adjustment needed. Data is processed in compliance.
ALERT: Threats detected. Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required
ALERT: Threats detected. Adjust Policy: Minimization: Collect and process only the data necessary for the specified purposes. - Action Required
ALERT: Threats detected. Adjust Policy: Consent Management: Implement a robust system for obtaining and managing user consent. - Action Required
ALERT: Threats detected. Adjust Policy: Access and Portability: Allow users to access their data and receive it in a portable format. - Action Required
Policy 5 is compliant and can continue processing data.
ALERT: Threats detected. Adjust Policy: Access Control: Enforce strict access controls and authentication mechanisms. - Action Required
Policy 7 is compliant and can continue processing data.
ALERT: Threats detected. Adjust Policy: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance. - Action Required
ALERT: Threats detected. Adjust Policy: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties. - Action Required
ALERT: Threats detected. Adjust Policy: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users. - Action Required
ALERT: Threats detected. Adjust Policy: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models. - Action Required
Policy 12 is compliant and can continue processing data.
ALERT: Threats detected. Adjust Policy: Documentation: Maintain comprehensive records of privacy practices and data processing activities. - Action Required
Policy 14 is compliant and can continue processing data.
Policy 15 is compliant and can continue processing data.
```

Figure 33: Part 3 of Main Method Results

```
Detecting threats...
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 77s 16ms/step
Final predictions generated and threat detection completed
Compliance Check Results: {'General Compliance': array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]]), 'GDPR Compliance': array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]]), 'CCPA Compliance': array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]]), 'NIST Compliance': array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]]), 'Overall Compliance': array([[False],
       [False],
       [False],
       ...,
       [False],
       [False],
       [False]])}
```

Figure 34: Results of Framework before Policy Stage

```
                     dewPoint  precipProbability
time
2016-01-01 05:00:00      24.4                0.0
2016-01-01 05:01:00      24.4                0.0
2016-01-01 05:02:00      24.4                0.0
2016-01-01 05:03:00      24.4                0.0
2016-01-01 05:04:00      24.4                0.0
Target (Y):
time
2016-01-01 05:00:00    0.932833
2016-01-01 05:01:00    0.934333
2016-01-01 05:02:00    0.931817
2016-01-01 05:03:00    1.022050
2016-01-01 05:04:00    1.139400
Name: use [kW], dtype: float64
Data is Loaded and preprocessed successfully
Warning: NaN or infinity values found in the preprocessed data.
Train-test split done successfully
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-e3d18c31c2cd> in <cell line: 107>()
    105 load_button.on_click(load_data)
    106
--> 107 size = int(len(data)*0.3)
    108 train = data[:size]
    109 test = data[size:]

NameError: name 'data' is not defined
```

Figure 35: Loaded, Preprocessed, and Split Data

Chapter 5 Testing/Evaluation

During the implementation and now the testing stage, the CNN and LSTM models were evaluated based on very standard metrics e.g. accuracy and loss. Successfully identifying threats within the IoT application dataset, both models performed well and to great standards. The compliance-checking mechanism gave further insights into how well the data obeyed and adhered to GDPR, CCPA, and NIST standards which were the chosen regulatory frameworks to test on. A compliance percentage was achieved, evidencing the framework's efficacy in the duality of threat detection and regulatory adherence and its ability to notify if alerts are detected to send alerts as seen in the results section of the implementation from Figure 33.

5.1 Pipeline Implementation

When executing the Google Colab Python cell for running the required Python libraries, ipywidgets buttons for its UI, loading including preprocessing, normalising and splitting testing and training data, along with training the models, detecting predicted threats and checking compliance, what initially happened when testing the framework was an error message popped up and would state ‘X_preproceesed’ in not defined. The reason is that ‘X_preprocessed’ is defined and used inside the ‘load_data()’ function as seen in Figure 24 and Figure 25. The variable had not been initialised or assigned a value before it was used in the ‘train_test_split()’ function. Once the ‘Load Data’ button is pressed triggering everything inside the function, based on what’s seen in Figure 27 where print statements are used corresponding to its output seen in Figure 31, Figure 36 from its displayed information, the outcome shows as ‘Data is loaded and preprocessed successfully’ and ‘Train-test split done successfully’. During the testing stage of this report from Figure 35, similarly ‘data’ was not defined outside the ‘load_data()’ function and for the framework to run, the ‘Load Data’ button will need to be triggered.

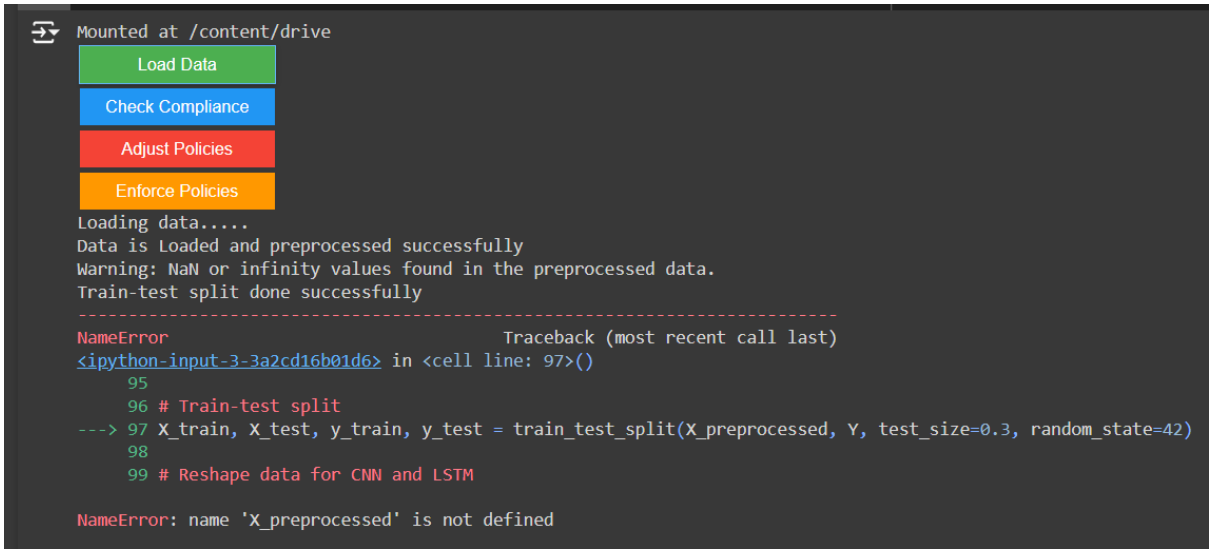


Figure 36: Error of Defining Variables

5.2 Epoch Times for CNN and LSTM Models

In Figure 37, I have successfully trained two CNN and LSTM models outlined from the design stage. The training of both models is progressing through multiple epochs. The CNN and LSTM models go through 11 epochs which is more than enough time for both models to train, detect threats, and check the IoT application's compliance, totalling just under 2 hours. What I found is that training the models with fewer epochs such as '2', just like in Figures 16 and 17, produces the same results while being more efficient in terms of time, early feedback, and computational costs. However, the argument can be told as not a good idea due to underfitting and poor generalisation. Low accuracy on both training and validation sets and not having enough time to learn the underlying patterns can seriously affect the training of deep learning models, causing the program to display the results incorrectly because the model hasn't fully learned yet.

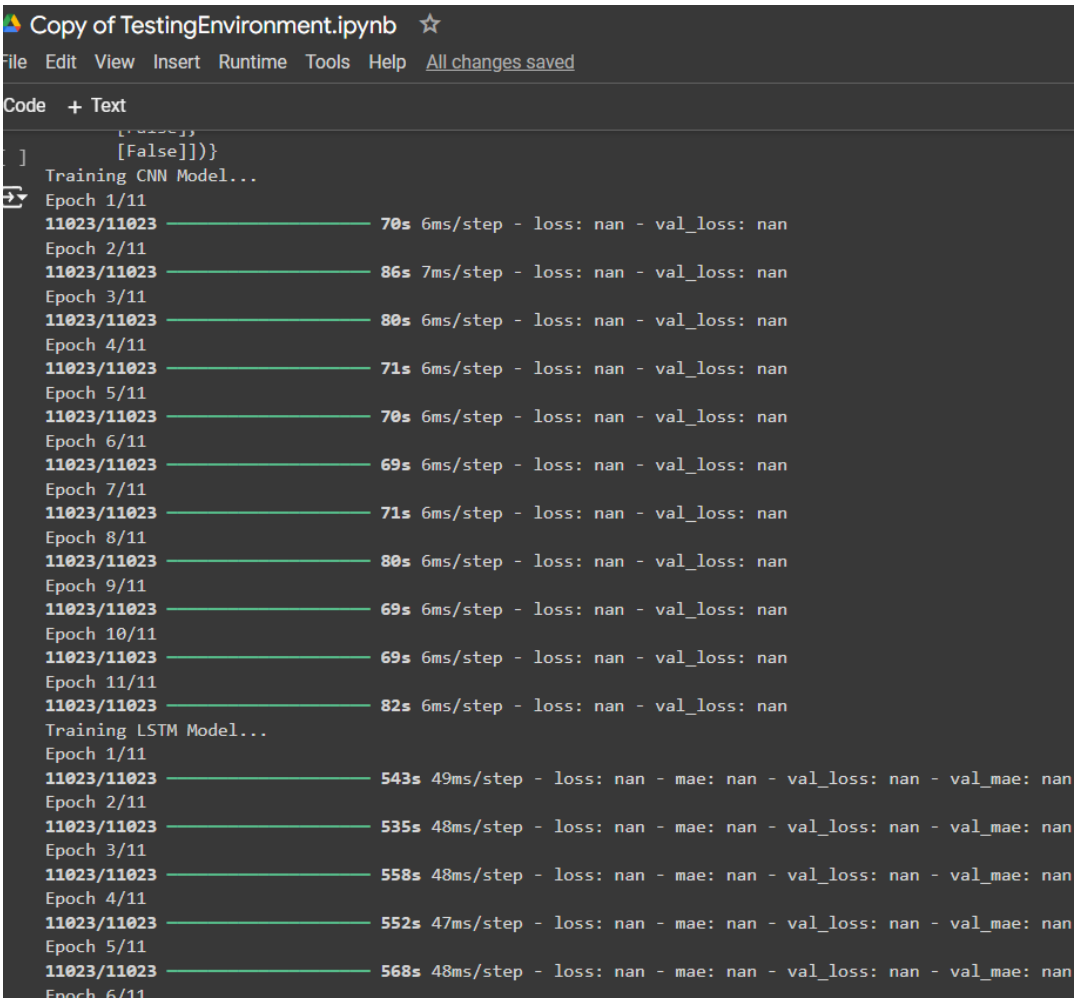


Figure 37: Part 1 of Results of Deep Learning Model's Epoch Situation

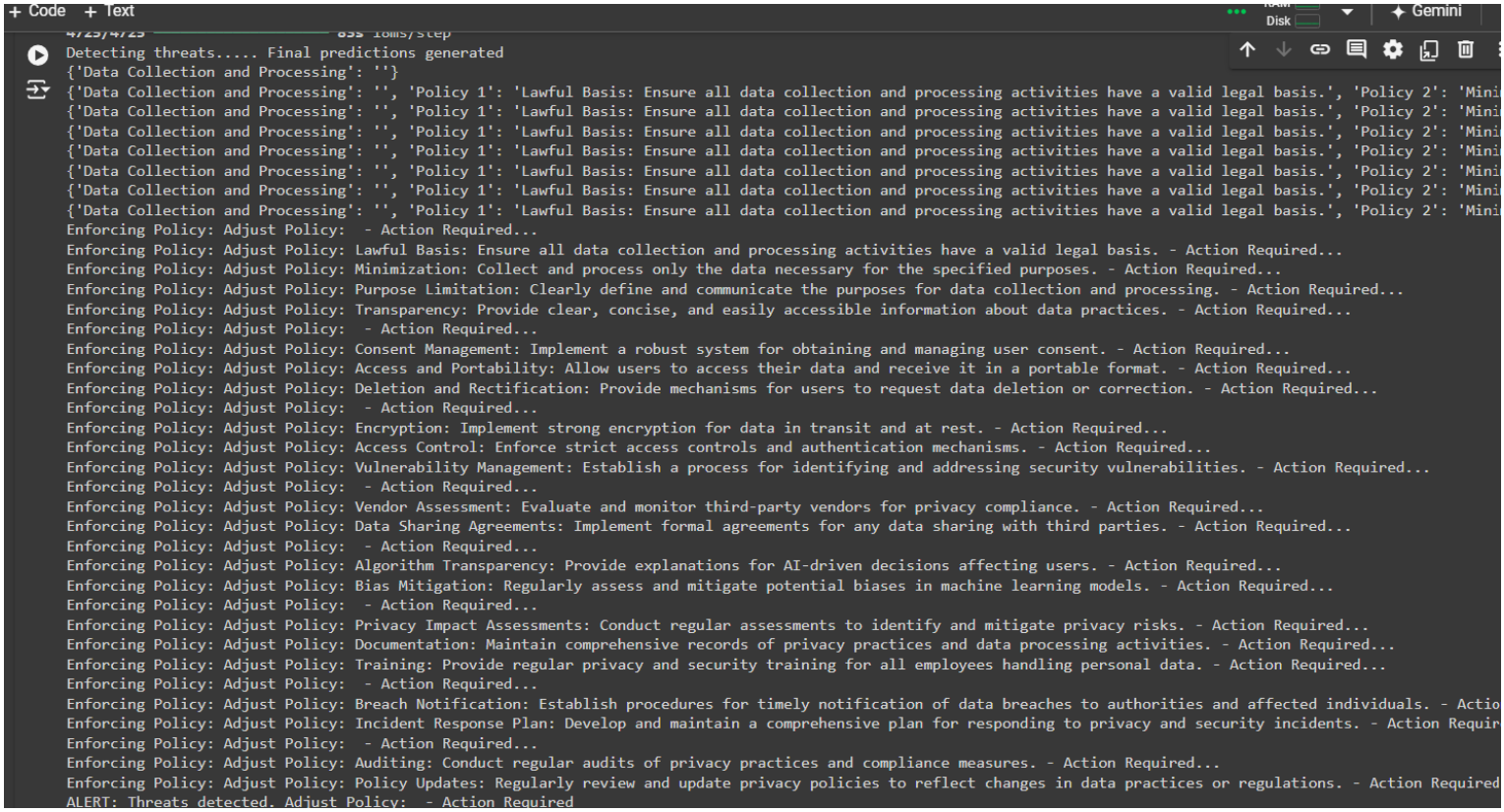


Figure 38: Part 2 of Results of Deep Learning Model's Epoch Situation

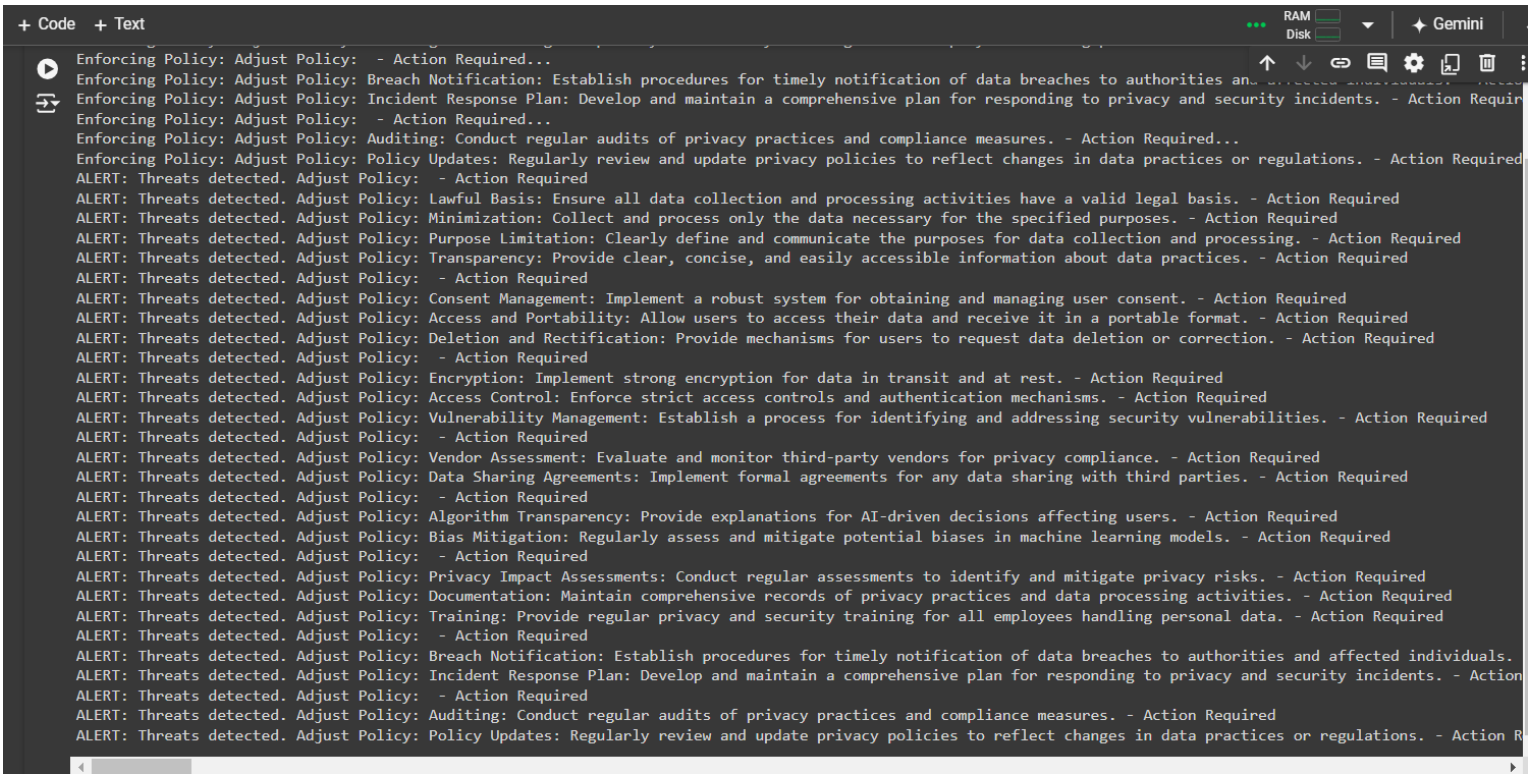


Figure 39: Part 3 of Results of Deep Learning Model's Epoch Situation

5.3 Testing Similar code for handling accuracy for testing and training

While testing to see if CNNs and LSTMs model training results were fixed to produce better accuracy instead of producing the result of 'NaN or infinity values found in preprocessing data' from Figure 36 or form Figure 37, stating 'loss: nan - mae: nan - val_loss: nan - val_mae: nan', the implementation has been adjusted slightly to produce the results found in Figure 40. The accuracy was still slow but still produced improvement and generalisation during training.

```
# Main function to run everything
def main():
    X_train, X_test, y_train, y_test = load_data()

    # Reshape data for CNN and LSTM
    X_train_reshaped, X_test_reshaped = reshape_data(X_train, X_test)

    # Train the models
    cnn_model = train_cnn_model(X_train_reshaped, y_train, X_test_reshaped, y_test)
    lstm_model = train_lstm_model(X_train_reshaped, y_train, X_test_reshaped, y_test)

    # Detect threats
    threats, predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)

    # Compliance check
    compliance_results = regulatory_compliance_check(threats, predictions)
    process_compliance_results(compliance_results)

# Execute main function
if __name__ == "__main__":
    main()
```

```
Loading data.....
Data is loaded and preprocessed successfully.
Train-test split done successfully
Training CNN Model...
Epoch 1/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 86s 8ms/step - loss: 0.1145 - mse: 0.1145 - val_loss: 0.0102 - val_mse: 0.0102
Epoch 2/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 78s 7ms/step - loss: 0.0069 - mse: 0.0069 - val_loss: 0.0041 - val_mse: 0.0041
Training LSTM Model...
Epoch 1/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 593s 54ms/step - loss: 1.1091 - mse: 1.1091 - val_loss: 1.1195 - val_mse: 1.1195
Epoch 2/2
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 628s 54ms/step - loss: 1.0993 - mse: 1.0993 - val_loss: 1.1181 - val_mse: 1.1181
Detecting threats...
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 12s 2ms/step
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 78s 16ms/step
Final predictions generated and threat detection completed
```

Figure 40: Handling Accuracy for Testing and Training Purposes

5.4 Full Rework/Comprehensive Refinement of Model Implementation

5.4.1 Overview of Data Partitioning and Handling NaN values

In the entirety of its thesis report and framework implementation, the dataset utilised for this research comprised of 503,909 data points over 31 columns (training set: 151,172 data points, and testing set: 352,737 data points) where all columns needed to be cleaned, preprocessed, and normalised. All columns displayed complete data integrity with no null entries, guaranteeing robust analysis and contained features associated to energy consumption across

multiple household appliances, together with environmental factors such as wind speed, humidity, and temperature.

```
# Converting 'cloudCover' to numeric and fill NaN with mean to avoid NaN
X['cloudCover'] = pd.to_numeric(X['cloudCover'], errors='coerce')
X['cloudCover'].fillna(X['cloudCover'].mean(), inplace=True)

# Handling any remaining NaN values in the data
X.fillna(0, inplace=True)

# Identifying numeric and categorical columns
numeric_features = X.select_dtypes(include=['float64', 'int64']).columns
categorical_features = ['icon', 'summary']
```

Figure 41: Handling NaN Values in the Data

The line in the middle 'X.fillna(0, inplace=True)' replaces all the Nan values in the DataFrame 'x' with value 0. The parameter lets pandas modify the original DataFrame 'x' directly, avoiding data loss, ensuring numerical consistency, and simplifying further analysis because of its simplicity accomplishing calculations and operating functions to the full DataFrame without continuous accounting for missing data.

5.4.2 CNN and LSTM Training and Evaluation

Just as throughout the implementation, two models were trained for prediction: a CNN and an LSTM network. For this instance, reshaping the data was made into a function for parameterisation, code reusability, modularity, improved readability, etc. Furthermore, the training process involved five epochs, thus both models monitored for loss and mean squared error (MSE) metrics across training and validation datasets.

CNN Model Training Results:

- Final Training Loss: 0.0031, Final Validation Loss: 0.0016

From its outputs, it can be shown that the CNN model demonstrated a significant decrease in both training and validation losses, indicating generalisation during its training computations/process and effective learning too.

LSTM Model Training Results:

- Final Training Loss: 1.1282, Final Validation Loss: 1.0990

Over CNN, the LSTM model exhibited a more moderate decrease in loss values with loss values remaining above 1 all throughout training, meaning the model strained comparatively with the dataset's complexity. On the other hand, the struggle could be due to its architecture and may need further tuning to succeed in optimal performance, or may not be well suited for this dataset.

From Figure 43, the line 'optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)' added during this testing stage sets up the 'Adam optimizer' with a low learning rate, making sure model coverage all through training is stable and efficient. The line 'early_stopping =

tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)' from its callback, supervises the validation loss and ends training if it doesn't progress or improve after 5 epochs, helping to avert overfitting while re-establishing the best weights.

```
Training CNN Model...
Epoch 1/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 83s 7ms/step - loss: 0.1392 - mse: 0.1392 - val_loss: 0.0210 - val_mse: 0.0210
Epoch 2/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 80s 7ms/step - loss: 0.0076 - mse: 0.0076 - val_loss: 0.0032 - val_mse: 0.0032
Epoch 3/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 81s 7ms/step - loss: 0.0045 - mse: 0.0045 - val_loss: 0.0055 - val_mse: 0.0055
Epoch 4/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 84s 7ms/step - loss: 0.0035 - mse: 0.0035 - val_loss: 0.0042 - val_mse: 0.0042
Epoch 5/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 80s 7ms/step - loss: 0.0031 - mse: 0.0031 - val_loss: 0.0016 - val_mse: 0.0016
Training LSTM Model...
Epoch 1/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 595s 54ms/step - loss: 1.1246 - mse: 1.1246 - val_loss: 1.1176 - val_mse: 1.1176
Epoch 2/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 593s 54ms/step - loss: 1.1229 - mse: 1.1229 - val_loss: 1.1187 - val_mse: 1.1187
Epoch 3/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 602s 55ms/step - loss: 1.1080 - mse: 1.1080 - val_loss: 1.1164 - val_mse: 1.1164
Epoch 4/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 624s 55ms/step - loss: 1.1359 - mse: 1.1359 - val_loss: 1.1169 - val_mse: 1.1169
Epoch 5/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 594s 54ms/step - loss: 1.1282 - mse: 1.1282 - val_loss: 1.0990 - val_mse: 1.0990
```

Figure 42: Training Progress of CNN and LSTM with Loss and Validation Over Epochs

Initial Testing Stage.ipynb ☆

File Edit View Insert Runtime Tools Help Last edited on September 30

Comment Share

+ Code + Text Connect Gemini

```
[ ] # Reshaping data for CNN and LSTM
    def reshape_data(X_train, X_test):
        X_train_reshaped = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
        X_test_reshaped = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
        return X_train_reshaped, X_test_reshaped
```

```
[ ] # Defining and Training the Convolutional Neural Network Model
    def train_cnn_model(X_train, y_train, X_test, y_test):
        print("Training CNN Model...")
        cnn_model = Sequential([
            Conv1D(32, 3, activation='relu', input_shape=(X_train.shape[1], 1)),
            MaxPooling1D(2),
            Conv1D(64, 3, activation='relu'),
            MaxPooling1D(2),
            Flatten(),
            Dense(64, activation='relu'),
            Dense(1, activation='linear')  # Changed to linear for regression
        ])
        optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
        cnn_model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mse'])  # Using MSE for regression
        early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
        cnn_model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test), callbacks=[early_stopping], verbose=1)
        return cnn_model

    # Defining and Training the Long-Short Term Memory Model
    def train_lstm_model(X_train, y_train, X_test, y_test):
        print("Training LSTM Model...")
        lstm_model = Sequential([
            LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)),
            Dropout(0.2),
            LSTM(50, return_sequences=False),
            Dropout(0.2),
            Dense(1, activation='linear')  # Changed to linear for regression
        ])
        optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001)
        lstm_model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mse'])  # Using MSE for regression
        early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
        lstm_model.fit(X_train, y_train, epochs=5, batch_size=32, validation_data=(X_test, y_test), callbacks=[early_stopping], verbose=1)
```

Figure 43: Code Framework for Reshaping, Training CNN and LSTM Models with Adam Optimizer and Early Stopping Callback

5.4.3 Compliance Evaluation

The essential and primary objective of this thesis and its research was to assess compliance with various privacy regulations against an IoT application.

Code Output for Compliance Results against Current IoT Application:

```
Detecting threats...
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 80s 17ms/step
Final predictions generated and threat detection completed
General Compliance: 42.60% compliant
GDPR Compliance: 20.84% compliant
CCPA Compliance: 10.35% compliant
NIST Compliance: 2.70% compliant
Overall Compliance: 2.70% compliant
```

Figure 44: Compliance Results Generated against Regulatory Frameworks Threshold

Right after training the modes, the compliance detection period was commenced. The framework evaluated the IoT application's adherence to the privacy regulations explained in the Literature Review such as GDPR, CCPA, NIST. The framework flagged several policies for subsequent measures and next course of action based on non-compliance, yielding the respecting compliance metrics:

- General Compliance: 42.60% compliant, showing overall compliance to privacy requirements.
- GDPR Compliance: 20.84% compliant, revealing low conformity with GDPR requirements.
- CCPA Compliance: 10.35% compliant, demonstrating significant gaps in California-specific data privacy regulations.
- NIST Compliance: 2.70% compliant, emphasising serious shortfalls in the technical standards and best practices commended by NIST
- Overall Compliance: 2.70% compliant, implying there is a need and an urgent requirement for policy and security adjustments to meet regulatory standards for this IoT application.

5.4.4 Policies and Adjustments

The next step along the framework is 'Policies and Adjustments' examining 15 privacy policies. There was no change to this part of the implementation during the testing phase as this worked and produced the same results before when completing this in the implementation stage for this IoT application.

Many policies didn't need adjusting or required reviewing/assessment such as policies related to data minimisation, encryption, consent management, and algorithm transparency. On the other and mentioned in the implementation section of this report, the non-compliant policies mention that action would be required are:

- Vendor Assessment: The evaluation/assessment of third-party vendors for privacy compliance requires improvement.
- Vulnerability Management: The absence of a systematic approach to identifying and addressing security vulnerabilities.
- Access Control: A more stricter authentication mechanisms and access controls need reinforcement.
- Lawful Basis: The data collection of the IoT application much have a valid legal basis.

5.4.5 Threat Detection Alerts

Similarly to the implementation of the framework, the threat detection alert feature works the same way where multiple alerts are generated during the enforcement of the adjusted policies. The key policies aforementioned such as vendor assessment, vulnerability management, access control, and lawful basis triggered compliance issues. Furthermore, now indicating the presence of privacy threats within its IoT application. Conversely, policies such as consent management, algorithm transparency, and data sharing were regarded as compliant and could continue processing with no raising any alerts.

Results from the Testing Stage:

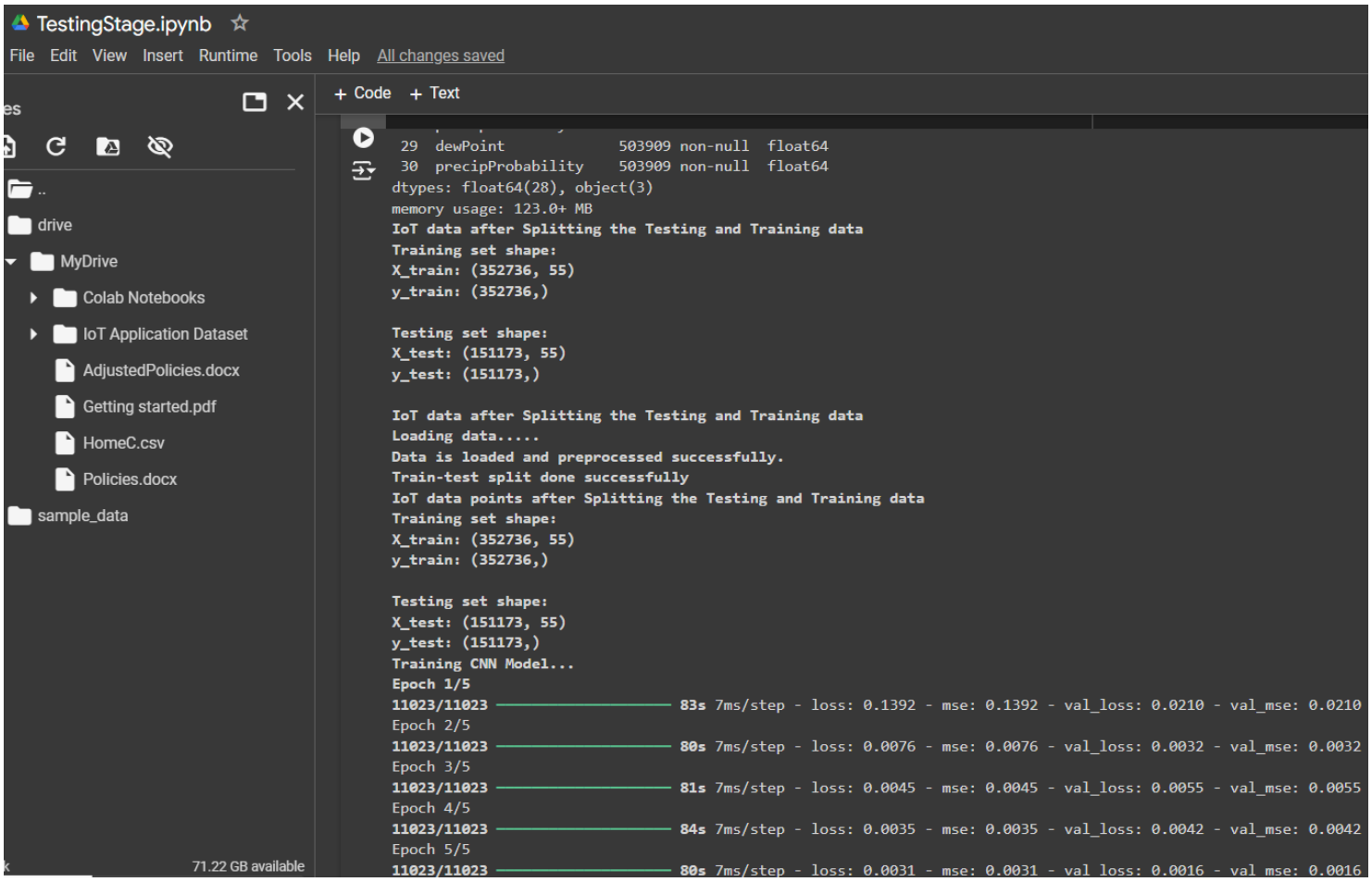


Figure 45: Part 1 of Testing Stage Results

```
y_test: (151173,)
Training CNN Model...
Epoch 1/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 83s 7ms/step - loss: 0.1392 - mse: 0.1392 - val_loss: 0.0210 - val_mse: 0.0210
Epoch 2/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 80s 7ms/step - loss: 0.0076 - mse: 0.0076 - val_loss: 0.0032 - val_mse: 0.0032
Epoch 3/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 81s 7ms/step - loss: 0.0045 - mse: 0.0045 - val_loss: 0.0055 - val_mse: 0.0055
Epoch 4/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 84s 7ms/step - loss: 0.0035 - mse: 0.0035 - val_loss: 0.0042 - val_mse: 0.0042
Epoch 5/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 80s 7ms/step - loss: 0.0031 - mse: 0.0031 - val_loss: 0.0016 - val_mse: 0.0016
Training LSTM Model...
Epoch 1/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 595s 54ms/step - loss: 1.1246 - mse: 1.1246 - val_loss: 1.1176 - val_mse: 1.1176
Epoch 2/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 593s 54ms/step - loss: 1.1229 - mse: 1.1229 - val_loss: 1.1187 - val_mse: 1.1187
Epoch 3/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 602s 55ms/step - loss: 1.1080 - mse: 1.1080 - val_loss: 1.1164 - val_mse: 1.1164
Epoch 4/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 624s 55ms/step - loss: 1.1359 - mse: 1.1359 - val_loss: 1.1169 - val_mse: 1.1169
Epoch 5/5
11023/11023 ━━━━━━━━━━━━━━━━━━━━ 594s 54ms/step - loss: 1.1282 - mse: 1.1282 - val_loss: 1.0990 - val_mse: 1.0990
Detecting threats...
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step
4725/4725 ━━━━━━━━━━━━━━━━━━━━ 80s 17ms/step
Final predictions generated and threat detection completed
General Compliance: 42.60% compliant
GDPR Compliance: 20.84% compliant
CCPA Compliance: 10.35% compliant
NIST Compliance: 2.70% compliant
Overall Compliance: 2.70% compliant
Loaded Policies: {'Policy 1': 'Lawful Basis: Ensure all data collection and processing activities have a valid legal basis.', 'Policy 2': 'Minimization: Collect a

Compliance Results: [False, True, True, True, True, False, False, False, True, True, True, True, True, True, True]
Adjusted Policies: {'Policy 1': 'Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required', '
```

Figure 46: Part 2 of Testing Stage Results

```
Compliance Results: [False, True, True, True, True, False, False, False, True, True, True, True, True, True, True]
Adjusted Policies: {'Policy 1': 'Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required', 'Po

Initial Policies:
  Policy 1: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis.
  Policy 2: Minimization: Collect and process only the data necessary for the specified purposes.
  Policy 3: Consent Management: Implement a robust system for obtaining and managing user consent.
  Policy 4: Access and Portability: Allow users to access their data and receive it in a portable format.
  Policy 5: Encryption: Implement strong encryption for data in transit and at rest.
  Policy 6: Access Control: Enforce strict access controls and authentication mechanisms.
  Policy 7: Vulnerability Management: Establish a process for identifying and addressing security vulnerabilities.
  Policy 8: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance.
  Policy 9: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties.
  Policy 10: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users.
  Policy 11: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models.
  Policy 12: Privacy Impact Assessments: Conduct regular assessments to identify and mitigate privacy risks.
  Policy 13: Documentation: Maintain comprehensive records of privacy practices and data processing activities.
  Policy 14: Auditing: Conduct regular audits of privacy practices and compliance measures.
  Policy 15: Policy Updates: Regularly review and update privacy policies to reflect changes in data practices or regulations.

Adjusted Policies:
  Policy 1: Adjust Policy: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - Action Required
  Policy 2: Minimization: Collect and process only the data necessary for the specified purposes. - No Policy Adjustment Needed.
  Policy 3: Consent Management: Implement a robust system for obtaining and managing user consent. - No Policy Adjustment Needed.
  Policy 4: Access and Portability: Allow users to access their data and receive it in a portable format. - No Policy Adjustment Needed.
  Policy 5: Encryption: Implement strong encryption for data in transit and at rest. - No Policy Adjustment Needed.
  Policy 6: Adjust Policy: Access Control: Enforce strict access controls and authentication mechanisms. - Action Required
  Policy 7: Adjust Policy: Vulnerability Management: Establish a process for identifying and addressing security vulnerabilities. - Action Required
  Policy 8: Adjust Policy: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance. - Action Required
  Policy 9: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties. - No Policy Adjustment Needed.
  Policy 10: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users. - No Policy Adjustment Needed.
  Policy 11: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models. - No Policy Adjustment Needed.
  Policy 12: Privacy Impact Assessments: Conduct regular assessments to identify and mitigate privacy risks. - No Policy Adjustment Needed.
  Policy 13: Documentation: Maintain comprehensive records of privacy practices and data processing activities. - No Policy Adjustment Needed.
  Policy 14: Auditing: Conduct regular audits of privacy practices and compliance measures. - No Policy Adjustment Needed.
  Policy 15: Policy Updates: Regularly review and update privacy policies to reflect changes in data practices or regulations. - No Policy Adjustment Needed.
```

Figure 47: Part 3 of Testing Stage Results

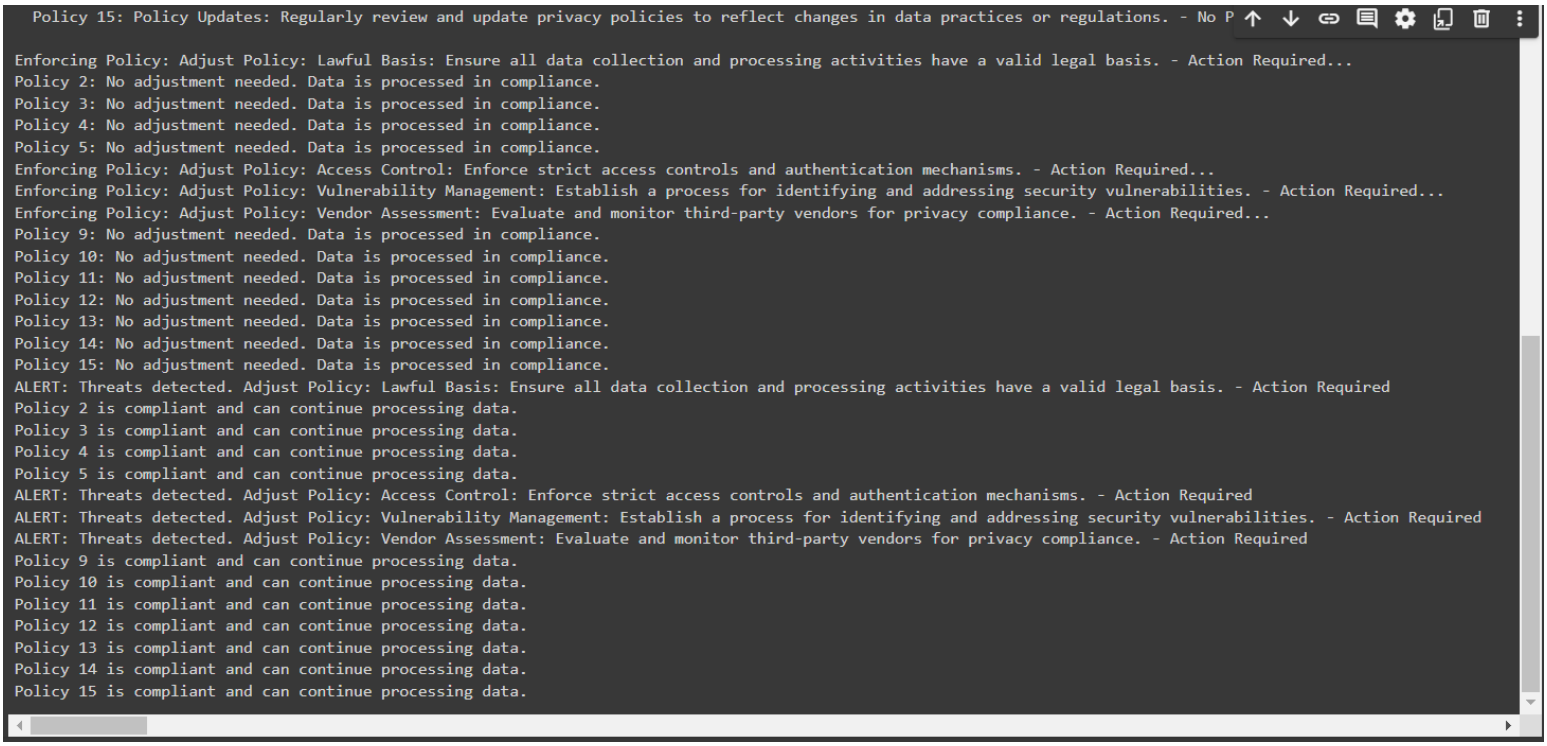


Figure 48: Part 4 of Testing Stage Results

5.5 Testing a Second IoT Dataset / Complete Implementation of Proposed Product

5.5.1 RT-IoT2022 Dataset Exploration

Referencing once more, the final complete implementation of the Proposed Product i.e. Deep Learning-Based Privacy Compliance Framework can be found here: Implementation_Completed_&_Testing_Another_IoT_Dataset.ipynb

Moving towards the final completion of the proposed product, the next IoT dataset (RT-IoT2022) used was from the same website https://www.kaggle.com/datasets/joebeachcapital/real-time-internet-of-things-rt-iot2022?resource=download available at this link with its license being Attribution 4.0 International (CC BY 4.0) allowing for open source and public use. To determine which columns would need changing or modification in the Python program's preprocessing section, the data structure needed analysis. Including both benign and malicious traffic samples, the RT-IoT2022 dataset encompassed network traffic data from various IoT devices. Additionally, featuring numerous attacks types such as Man-in-the-Middle (MitM) and Denial of Service (DoS), enabling the deep-learning framework to identify a broad spectrum of possible security threats and privacy breaches.

The key considerations taken when approaching the next steps of preprocessing so formatting is done correctly and enabling effective training and evaluation of the privacy compliance framework for IoT environments include:

1. Discovering categorical features such as ‘service’ and ‘proto’ for one-hot encoding.

2. Deciding numerical features that require scaling.
3. Validating/Utilising the target variable, probable to be the ‘Attack_type’ column.

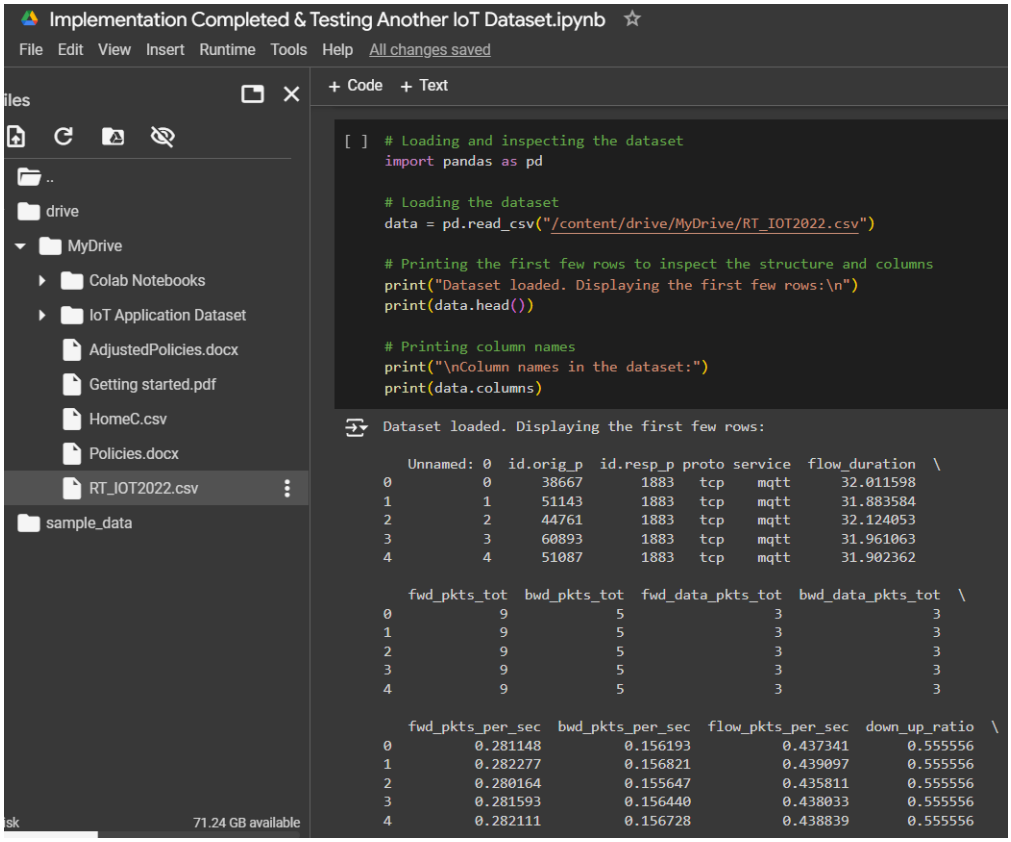


Figure 49: Part 1 Displaying the Columns in a Different IoT Application’s Dataset

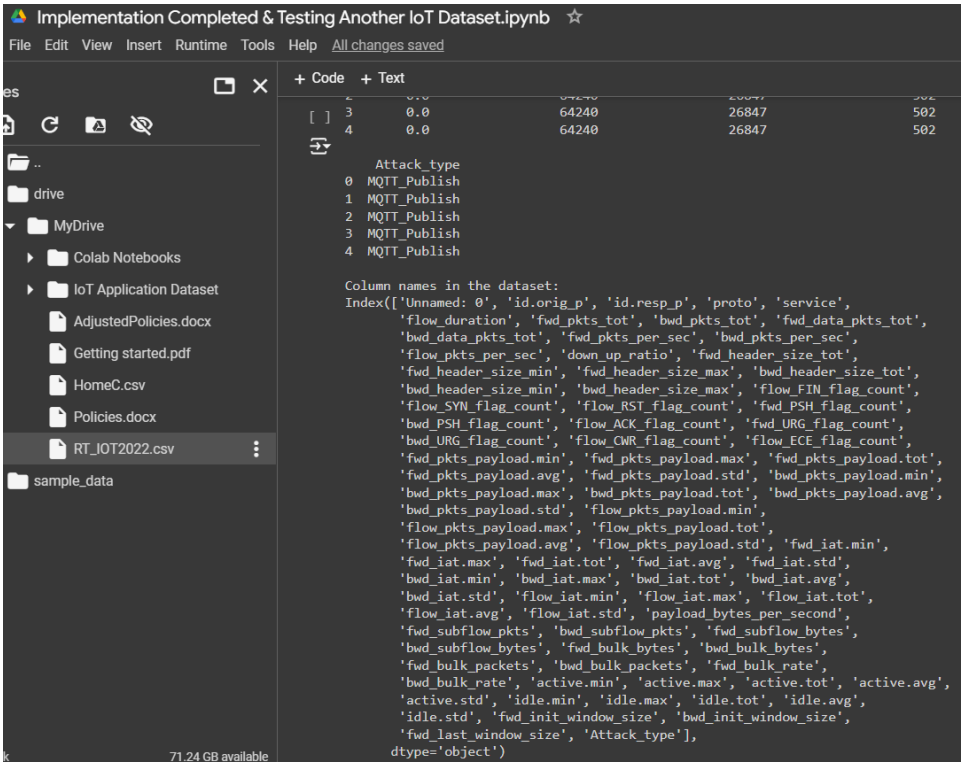


Figure 50: Part 1 Displaying the Columns in a Different IoT Application’s Dataset

I have introduced and added several global variables in my final code for it to store vital data and model components. This will be accessed and modified over the multiple functions used such as the 'load_data()' functions for example. By adding these variables the data and models are more accessible throughout, allowing for effective data flow without the obligation of passing variables between functions.

```
# Global variables that stores data and models
data = None
X_preprocessed = None
Y = None
X_train = X_test = y_train = y_test = None
cnn_model = None
lstm_model = None
X_train_reshaped = X_test_reshaped = None
compliance_results = None
policies = None
adjusted_policies = None

# Loading and preprocessing the data
def load_data(b=None):
```

Figure 51: Global Variables before Accessed and Modifies across Functions

5.5.2 Loading and Preprocessing the Dataset

The implementation of the final proposed product was finalised and completed with minor changes from the original deep learning framework and also its initial testing stage framework implementation including its preprocessing, model training, compliance checking, and its main method.

The findings saw the first few rows confirming the presence of various IoT-related features, incorporating a train_test_split and target variable encoding when preprocessing was successfully completed. The training set entailed 86181 samples whereas the test set had 36,936 samples.

In a similar fashion to the previous iterations of my framework, the 'RT_IoT2022.csv' dataset was first loaded using panda, and a new column, 'time' was inserted to denote the temporal nature of the IoT data. The columns previously used such as 'kw' were dropped, and the categorical features ('service' and 'proto') became encoded capitalising on 'OneHotEncoder', in tandem with numerical features as they were being scaled using 'StandardScaler' and 'MinMaxScaler'.

5.5.3 Splitting the Data

Exactly like the earlier attempts at the implementation, right after preprocessing, the dataset was split 70-30 where 30% was used as the testing data. Making things differently, the 'train_test_split' function was leveraged to maintain stratified sampling based on the 'Attack_type' target variable, which was additionally encoded using 'LabelEncoder'.

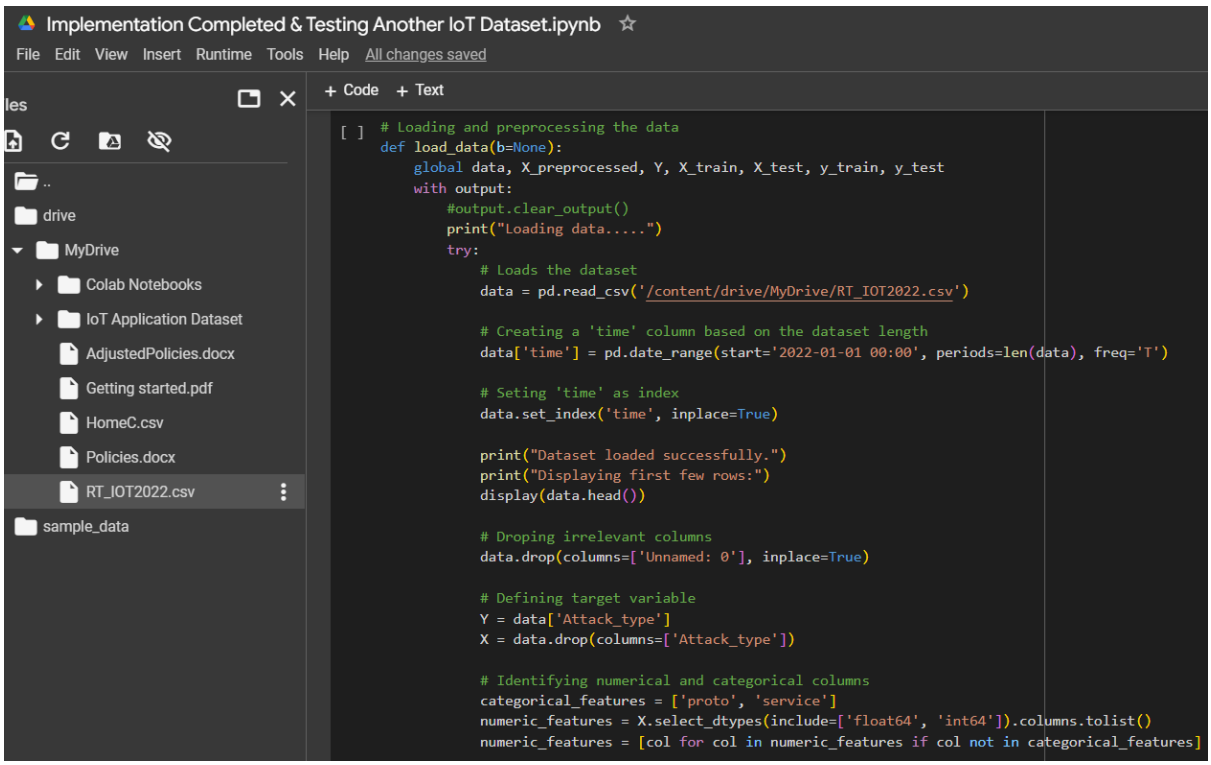


Figure 52: Part 1 Contents of Load Data Function / Preprocessing Data with Train-Test Splitting

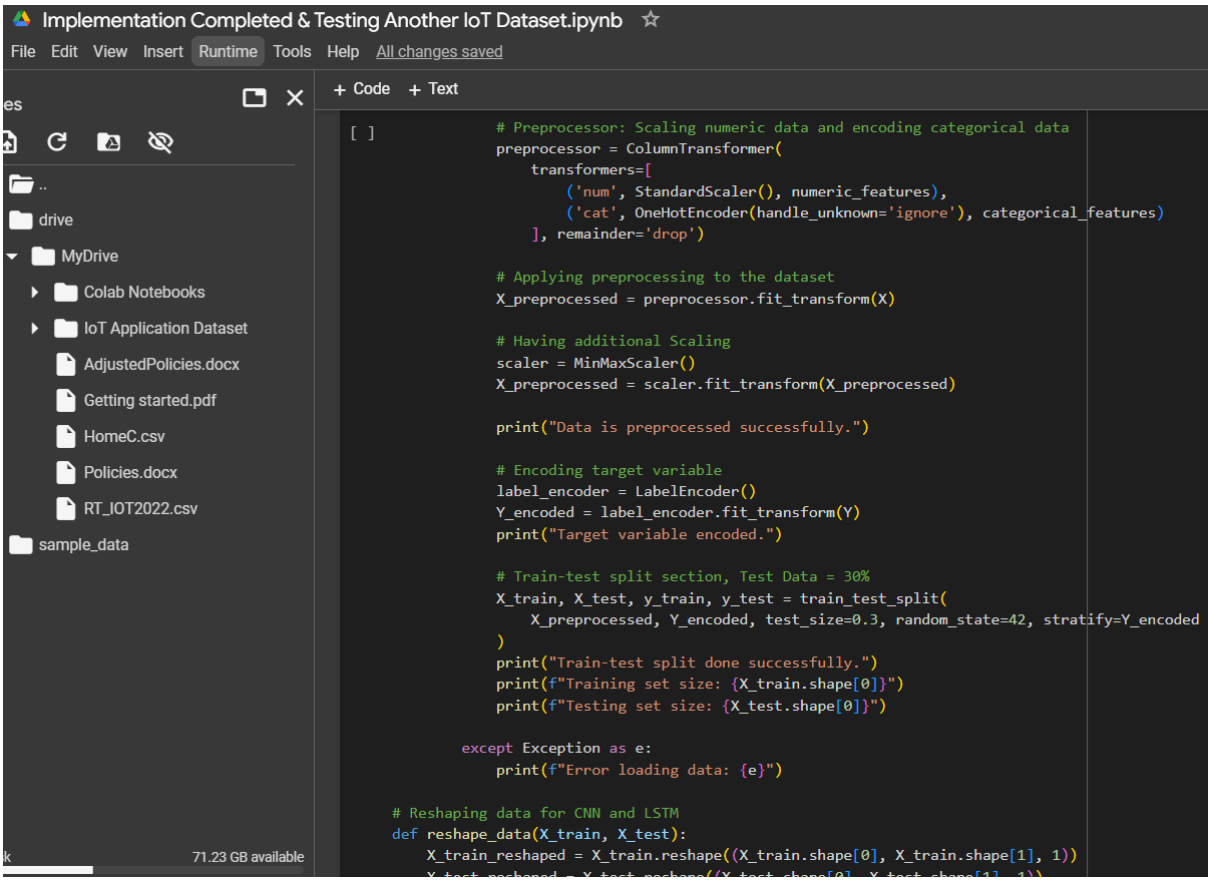
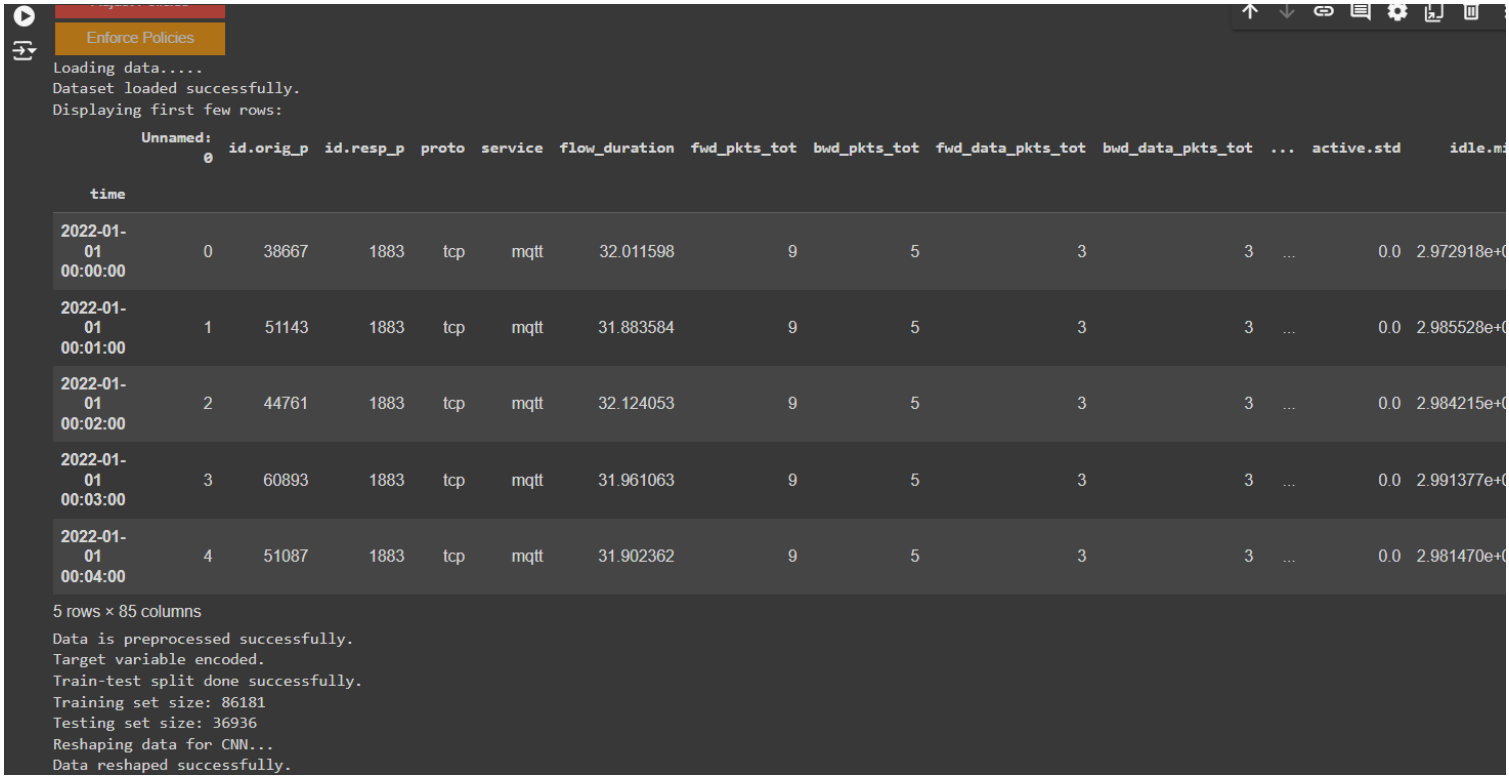


Figure 53: Part 2 Contents of Load Data Function / Preprocessing Data with Train-Test Splitting



Enforce Policies

```
Loading data.....
Dataset loaded successfully.
Displaying first few rows:
```

time	Unnamed: 0	id.orig_p	id.resp_p	proto	service	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_data_pkts_tot	bwd_data_pkts_tot	...	active.std	idle.mi
2022-01-01 00:00:00	0	38667	1883	tcp	mqtt	32.011598	9	5	3	3	...	0.0	2.972918e+0
2022-01-01 00:01:00	1	51143	1883	tcp	mqtt	31.883584	9	5	3	3	...	0.0	2.985528e+0
2022-01-01 00:02:00	2	44761	1883	tcp	mqtt	32.124053	9	5	3	3	...	0.0	2.984215e+0
2022-01-01 00:03:00	3	60893	1883	tcp	mqtt	31.961063	9	5	3	3	...	0.0	2.991377e+0
2022-01-01 00:04:00	4	51087	1883	tcp	mqtt	31.902362	9	5	3	3	...	0.0	2.981470e+0

```
5 rows × 85 columns
Data is preprocessed successfully.
Target variable encoded.
Train-test split done successfully.
Training set size: 86181
Testing set size: 36936
Reshaping data for CNN...
Data reshaped successfully.
```

Figure 54: Part 2 Results of Preprocessing Data with Train-Test Split Initiated

5.5.4 CNN & LSTM Model Modifications

Both models have been implemented starting with If statements and they play an important role in model dependency management and data validation. The if statement in CNN makes sure that 'X_train' and 'y_train' is loaded right before training. On the other hand, the LSTM model checks if the reshaped data i.e. 'X_train_reshaped' 'y_train' are available, all assuming the model has been trained and was trained using a sparse categorical cross-entropy loss function, Adam optimizer with early stopping applied to prevent overfitting.

The key differences between the two frameworks concerning CNN and LSTM training are their network type, layer structure, epochs, and input handling.

CNN was trained over 5 epochs, triumphantly reaching a high accuracy of 99.28% on the training data and a validation accuracy of 99.39%. This has made it a working framework with correct results ready for organisational use. The loss also progressively lowered, pointing to successful training.

Regarding LSTM, it also performed well, though with a marginally lower accuracy than CNN. Its model was trained over 2 epochs because the time required per epoch was 2 to 3 times longer than CNN. Ultimately, the results showed a final accuracy of 95.19% on the training data and a validation accuracy of 96.63%.

```python
# Defining and training the CNN Model
def train_cnn_model():
    global cnn_model, X_train_reshaped, X_test_reshaped
    with output:
        #output.clear_output()
        if X_train is None or y_train is None:
            print("Please load the data first.")
            return
        print("Reshaping data for CNN...")
        try:
            X_train_reshaped, X_test_reshaped = reshape_data(X_train, X_test)
            print("Data reshaped successfully.")

            print("Training CNN Model...")
            cnn_model = Sequential([
                Conv1D(32, 3, activation='relu', input_shape=(X_train_reshaped.shape[1], 1)),
                MaxPooling1D(2),
                Conv1D(64, 3, activation='relu'),
                MaxPooling1D(2),
                Flatten(),
                Dense(64, activation='relu'),
                Dense(len(np.unique(y_train)), activation='softmax')
            ])
            cnn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
            early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
            cnn_model.fit(
                X_train_reshaped, y_train,
                epochs=5,
                batch_size=32,
                validation_data=(X_test_reshaped, y_test),
                callbacks=[early_stopping],
                verbose=2
            )
            print("CNN Model trained successfully.")
        except Exception as e:
            print(f"Error training CNN model: {e}")
```

Figure 55: Contents of Training CNN Model

```python
# Defining and training the LSTM Model
def train_lstm_model():
    global lstm_model
    with output:
        #output.clear_output()
        if X_train_reshaped is None or y_train is None:
            print("Please load the data and train CNN model first.")
            return
        print("Training LSTM Model...")
        try:
            lstm_model = Sequential([
                LSTM(50, return_sequences=True, input_shape=(X_train_reshaped.shape[1], 1)),
                Dropout(0.2),
                LSTM(50, return_sequences=False),
                Dropout(0.2),
                Dense(len(np.unique(y_train)), activation='softmax')
            ])
            lstm_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
            early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
            lstm_model.fit(
                X_train_reshaped, y_train,
                epochs=2,
                batch_size=32,
                validation_data=(X_test_reshaped, y_test),
                callbacks=[early_stopping],
                verbose=2
            )
            print("LSTM Model trained successfully.")
        except Exception as e:
            print(f"Error training LSTM model: {e}")
```

Figure 56: Contents of Training LSTM Model

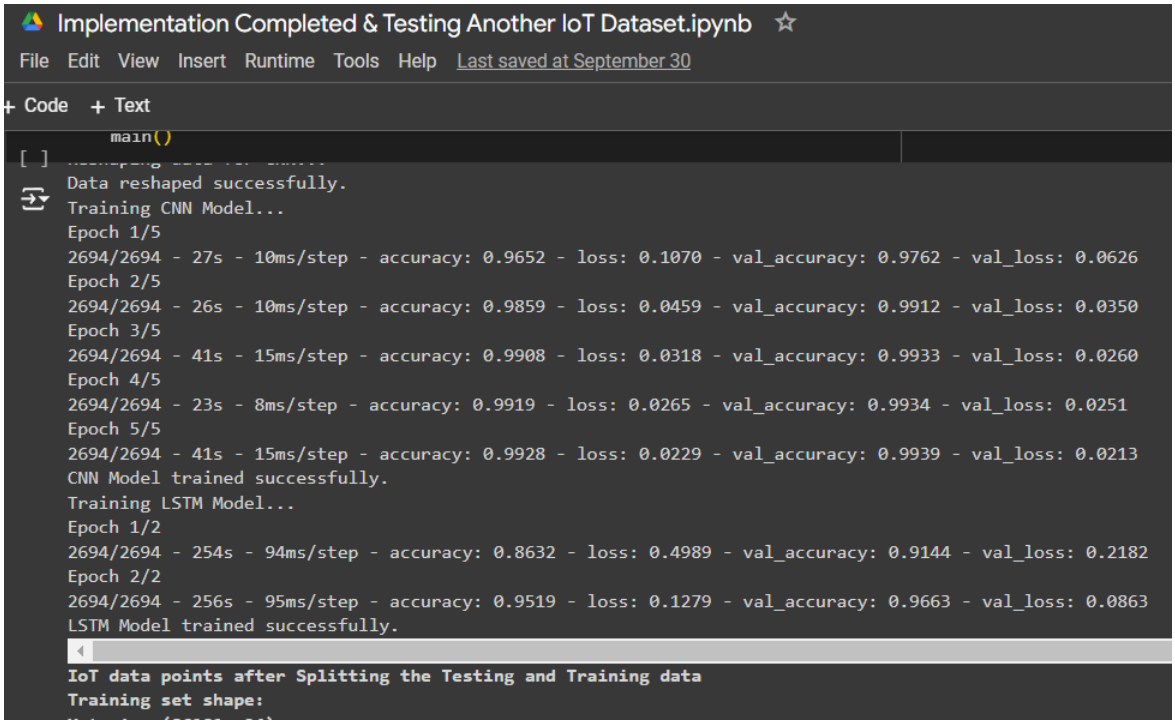


Figure 57: Results of Training CNN and LSTM Model

5.5.5 Threat Detection

There has been no change to the threat detection stage of the deep-learning framework from its first attempt to the final completion of the code. Predictions were generated using both models, and the final threat detection results could be assembled. The hybrid approach I achieved leveraged temporal and spatial idiosyncrasies of the data. The results were averaged, resulting in a definitive decision on plausible threats that was constructed using the threshold values.

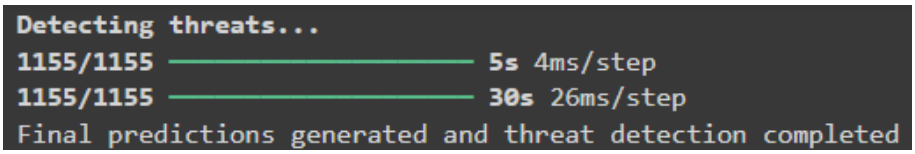


Figure 58: Results of Detecting Threats

5.5.6 Compliance Checking

The next step is compliance checking where the framework evaluates the compliance with key regulatory standards including GDPR, CCPA, and NIST IoT Cybersecurity Guidelines (also stated as NIST for ease of use). Each compliance metric was analysed, built upon prediction confidence levels. Where there were instances of non-compliance being flagged, the predictions exceeded certain thresholds. The assessment/compliance check across the regulatory standards aforementioned resulted in an overall compliance rate of 91.62% signifying effective performance overall of the framework and correct outage of results. All 15 privacy policies were found to be compliant, and there were no adjustments to be made.

```python
# Compliance checking against regulatory compliance (GDPR, CCPA, NIST)
def regulatory_compliance_check(threats, predictions):
    # Compute compliance based on thresholds
    general_compliance = predictions < 0.7
    gdpr_compliance = predictions < 0.6
    ccpa_compliance = predictions < 0.5
    nist_compliance = predictions < 0.4
    overall_compliance = general_compliance & gdpr_compliance & ccpa_compliance & nist_compliance

    # Returns the compliance as percentages against whats outputs from its threshold
    return {
        "General Compliance": general_compliance,
        "GDPR Compliance": gdpr_compliance,
        "CCPA Compliance": ccpa_compliance,
        "NIST Compliance": nist_compliance,
        "Overall Compliance": overall_compliance
    }

def process_compliance_results(compliance_results):
    for standard, results in compliance_results.items():
        # Calculates compliance percentage correctly
        compliant_count = np.sum(results)  # Counts the number of True values
        total_count = results.size  # Total number of elements
        compliance_percentage = (compliant_count / total_count) * 100  # Calculates its percentage now
        print(f"{standard}: {compliance_percentage:.2f}% compliant")

# Compliance checking function
def check_compliance(b=None):
    with output:
        #output.clear_output()
        global compliance_results
        if cnn_model is None or lstm_model is None:
            print("Please train the models first.")
            return
        print("Checking compliance...")

        # Uses the final predictions for compliance checking
        threats, predictions = detect_threats(cnn_model, lstm_model, X_test_reshaped)
        if predictions is None:
            print("Error during threat detection.")
            return

        compliance_results = regulatory_compliance_check(predictions)
        process_compliance_results(compliance_results)
        print(f"Compliance Results: {compliance_results}")

# Binding widget buttons to their functions
load_button.on_click(load_data)
check_compliance_button.on_click(check_compliance)
```

Figure 59: Python code for Checking Compliance of IoT's Application

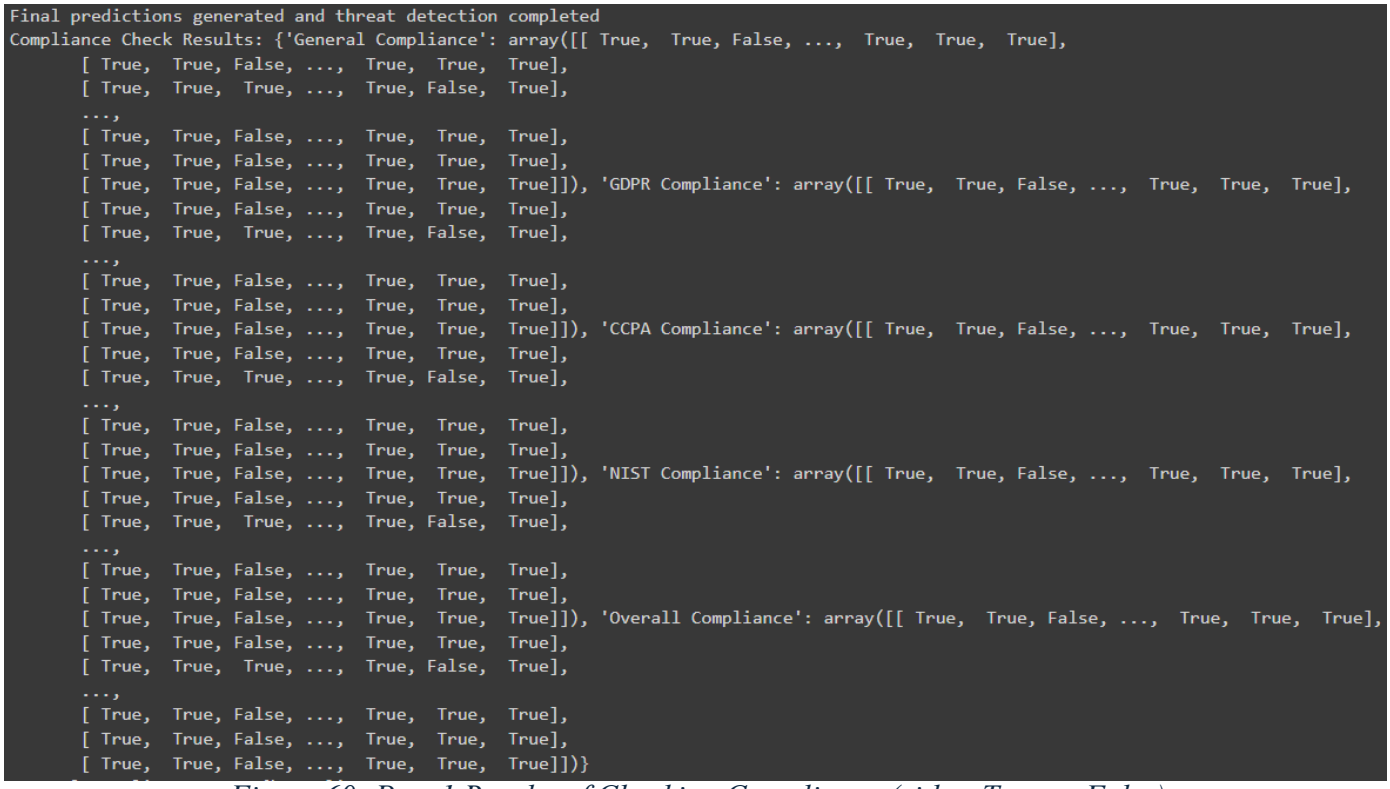


Figure 60: Part 1 Results of Checking Compliance (either True or False)

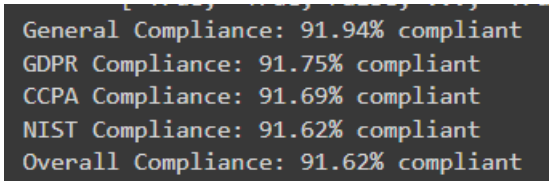


Figure 61: Part 2 Results of Checking Compliance (Percentage)

5.5.7 Policy Adjustments, Enforcement and Alerts

There were no modifications or changes to this section from the previous iterations of the Framework which are Deep_Learning_Framework_Implementation.ipynb and Initial_Testing_Stage.ipynb where the policies were imported from a 'docx' file and adjusted based on the compliance results. Moreover, the framework proposed corrective actions for policies needing its updates and allowed and updated 'docx' file to be produced called 'AdjustedPolicies.docx' noting down the necessary revisions. The adjusted policies were enforced stating 'Action Required' or 'No Policy Adjustment Needed'. Notifications would triggered if any identified threats or maybe any instances of non-compliance have occurred, facilitating a forward-thinking strategy for opportunities to deal with regulatory infractions or data breaches.

The screenshots of this section 'Policy Adjustment, Enforcement, and Alerts' will be similar and/or potentially the same as the same code used in all the implementation iterations e.g. Figures 21 and 23 from Chapter 4 Implementation.

```
Overall Compliance: 91.62% compliant
Loaded Policies: {'Policy 1': 'Lawful Basis: Ensure all data collection and processing activities have a valid legal basis.', 'Policy 2 : Minimization: Collect and

Compliance Results: [True, True, True, True, True, True, True, True, True, True, True, True, True, True, True]
Adjusted Policies: {'Policy 1': 'Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - No Policy Adjustment Needed.', 'Poli

Initial Policies:
  Policy 1: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis.
  Policy 2: Minimization: Collect and process only the data necessary for the specified purposes.
  Policy 3: Consent Management: Implement a robust system for obtaining and managing user consent.
  Policy 4: Access and Portability: Allow users to access their data and receive it in a portable format.
  Policy 5: Encryption: Implement strong encryption for data in transit and at rest.
  Policy 6: Access Control: Enforce strict access controls and authentication mechanisms.
  Policy 7: Vulnerability Management: Establish a process for identifying and addressing security vulnerabilities.
  Policy 8: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance.
  Policy 9: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties.
  Policy 10: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users.
  Policy 11: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models.
  Policy 12: Privacy Impact Assessments: Conduct regular assessments to identify and mitigate privacy risks.
  Policy 13: Documentation: Maintain comprehensive records of privacy practices and data processing activities.
  Policy 14: Auditing: Conduct regular audits of privacy practices and compliance measures.
  Policy 15: Policy Updates: Regularly review and update privacy policies to reflect changes in data practices or regulations.

Adjusted Policies:
  Policy 1: Lawful Basis: Ensure all data collection and processing activities have a valid legal basis. - No Policy Adjustment Needed.
  Policy 2: Minimization: Collect and process only the data necessary for the specified purposes. - No Policy Adjustment Needed.
  Policy 3: Consent Management: Implement a robust system for obtaining and managing user consent. - No Policy Adjustment Needed.
  Policy 4: Access and Portability: Allow users to access their data and receive it in a portable format. - No Policy Adjustment Needed.
  Policy 5: Encryption: Implement strong encryption for data in transit and at rest. - No Policy Adjustment Needed.
  Policy 6: Access Control: Enforce strict access controls and authentication mechanisms. - No Policy Adjustment Needed.
  Policy 7: Vulnerability Management: Establish a process for identifying and addressing security vulnerabilities. - No Policy Adjustment Needed.
  Policy 8: Vendor Assessment: Evaluate and monitor third-party vendors for privacy compliance. - No Policy Adjustment Needed.
  Policy 9: Data Sharing Agreements: Implement formal agreements for any data sharing with third parties. - No Policy Adjustment Needed.
  Policy 10: Algorithm Transparency: Provide explanations for AI-driven decisions affecting users. - No Policy Adjustment Needed.
  Policy 11: Bias Mitigation: Regularly assess and mitigate potential biases in machine learning models. - No Policy Adjustment Needed.
  Policy 12: Privacy Impact Assessments: Conduct regular assessments to identify and mitigate privacy risks. - No Policy Adjustment Needed.
  Policy 13: Documentation: Maintain comprehensive records of privacy practices and data processing activities. - No Policy Adjustment Needed.
```

Figure 62: Part 1 Results of Policy Adjustment, Enforcement, and Alerts

```
  Policy 13: Documentation: Maintain comprehensive records of privacy practices and data processing activities. - No Policy Adjustment Needed.
  Policy 14: Auditing: Conduct regular audits of privacy practices and compliance measures. - No Policy Adjustment Needed.
  Policy 15: Policy Updates: Regularly review and update privacy policies to reflect changes in data practices or regulations. - No Policy Adjustment Needed.

Policy 1: No adjustment needed. Data is processed in compliance.
Policy 2: No adjustment needed. Data is processed in compliance.
Policy 3: No adjustment needed. Data is processed in compliance.
Policy 4: No adjustment needed. Data is processed in compliance.
Policy 5: No adjustment needed. Data is processed in compliance.
Policy 6: No adjustment needed. Data is processed in compliance.
Policy 7: No adjustment needed. Data is processed in compliance.
Policy 8: No adjustment needed. Data is processed in compliance.
Policy 9: No adjustment needed. Data is processed in compliance.
Policy 10: No adjustment needed. Data is processed in compliance.
Policy 11: No adjustment needed. Data is processed in compliance.
Policy 12: No adjustment needed. Data is processed in compliance.
Policy 13: No adjustment needed. Data is processed in compliance.
Policy 14: No adjustment needed. Data is processed in compliance.
Policy 15: No adjustment needed. Data is processed in compliance.
Policy 1 is compliant and can continue processing data.
Policy 2 is compliant and can continue processing data.
Policy 3 is compliant and can continue processing data.
Policy 4 is compliant and can continue processing data.
Policy 5 is compliant and can continue processing data.
Policy 6 is compliant and can continue processing data.
Policy 7 is compliant and can continue processing data.
Policy 8 is compliant and can continue processing data.
Policy 9 is compliant and can continue processing data.
Policy 10 is compliant and can continue processing data.
Policy 11 is compliant and can continue processing data.
Policy 12 is compliant and can continue processing data.
Policy 13 is compliant and can continue processing data.
Policy 14 is compliant and can continue processing data.
Policy 15 is compliant and can continue processing data.
```

Figure 63: Part 2 Results of Policy Adjustment, Enforcement, and Alerts

Chapter 6 Evaluation and Conclusion of Proposed Framework & Thesis

The framework's results outline and demonstrate both limitations and also promise, laying the groundwork for future advancements in privacy compliance for the rapidly expanding IoT ecosystem and everything it contains. The proposed deep learning-based privacy compliance framework for IoT applications developed in this project serves as a valuable tool for assessing IoT applications against stringent privacy regulations. Despite its capabilities and functionality, it has been determined that there are significant compliance gaps. Calculating the right overall compliance will require fixing to display the correct percentage. To a great achievement, this was amended with successful results where the second IoT dataset calculated an accurate compliance percentage of over 90% for GDPR, CCPA, and NIST with its overall compliance being 91.62%.

Additionally, although the CNN model indicates promise and assuredness in real-time compliance detection and monitoring, further advancements to the LSTM model are needed for better optimisation to enhance and better capture time-dependent compliance issues and its ability to detect temporal compliance risks accurately. Further efforts will need to focus on refining the accuracy of the models and strengthening vendor relationships to achieve comprehensive privacy compliance in IoT environments. Moreover, instead of using thresholds in the form of numbers, a more enhanced and thorough proposed deep learning framework would include all the sets of guidelines and policies from GDPR, CCPA, and NIST IoT Cybersecurity Guidelines

Testing both IoT datasets effectively worked and achieved positive results when successfully processed using CNN and LSTM architectures. Although designed to capture temporal patterns, the LSTM model's performance was hindered because it struggled with predictive accuracy. To improve its compliance detection capabilities, predominantly over time-sensitive patterns, LSTM needs extra optimisation. This changed during its final implementation when testing another IoT dataset achieving an accuracy of over 90% on average. The research papers and articles found/used in the literature review and related works section proved useful when designing and implementing my proposed product with its key information on privacy compliance, deep learning, and IoT applications functionalities from germane authors. There were notable gaps in privacy adherence and policy structure which can be improved upon moving forward. The deep learning framework's capability in its enforcement of policies and presenting recommendations for adjustments constitutes a successful conclusion, validating the system's performance and potential not exclusively to detect but also to address compliance issues dynamically.

Moreover, the evaluation has concluded that the system i.e. implementing and testing the deep learning framework against an IoT application and checking its compliance with regulatory frameworks while seeing if the application adhered to policies set out in a document, its conclusion yielded significant insights into its effectiveness and potential for real-world application. The framework can more or less handle processing large volumes of

IoT data efficiently observing real-time threat detection, offering a solution to IoT security and privacy challenges. Additional strengths entail the framework’s design being aligned greatly with current regulatory standards, turning it into a significant asset for organisations seeking to maintain compliance.

Areas for improvement surround its resource optimisations, explainability, privacy-preserving techniques, and broader dataset testing. Deep learning models necessitate substantial computational resources, forthcoming iterations would bring attention to optimising resource usage for operations/arrangement on resource-constrained IoT devices. By using broader datasets, enhancing the framework’s explainability, and integrating privacy-preserving techniques, trust improvement can be reached hypothetically by third parties. Furthermore, the trust ensured would validate the system's effectiveness across a wider range of scenarios that involve IoT applications for relevant organisations.

Chapter 7 Conclusion and Future Directions

In conclusion, this thesis has truly demonstrated the development and implementation of a deep learning-based privacy compliance framework specifically tailored for IoT applications. Most if not all IoT devices have a sheer volume of data due to their practical use with the additional factor of their exponential growth over the years. Challenges have been introduced in ensuring privacy compliance and detecting potential threats in real time. From my proposed framework, leveraging LSTM and CNNs can prove its effectiveness in both adhering to compliance with privacy regulations i.e. GDPR, CCPA, and NIST IoT Cybersecurity Guidelines, and anomaly detection. Adaptability plays a crucial part within this framework in terms of the dynamic policy adjustment mechanism, making it capable of enforcing policy changes in its consideration of detected threats, warranting continuous regulatory adherence.

Despite the encouraging and favourable outcomes, the deep learning framework underscores key areas for advancement. An area that notably needs refinement is optimising real-time compliance mechanisms and streamlining threat detection accuracy across various IoT environments. A different domain for refinement is the aspect of preprocessing. It is required to change, add or drop columns when a new dataset is used while adding and changing required libraries such as ‘OneHotEncoder’ for example. The next step would be creating a deep learning-based privacy compliance framework that has a one size fits all for various IoT applications catering to the IoT dataset’s needs. Further improvements in strengthening the system's effectiveness and scalability could happen with the integration of more diverse datasets while exploring advanced deep-learning techniques.

This research engaged in offers significant value and can be a key factor in the evolving field of IoT security and privacy by presenting a robust solution addressing both regulatory and technical challenges, guaranteeing the secure handling/protection of data within highly interconnected and complex IoT ecosystems. My proposal stated in ‘Chapter 1 Introduction’ from ‘1.2 Proposal’ while addressing everything stated in ‘2.7 Gaps and Future Directions’ from ‘Chapter 2 Literature Review’, meeting all the requirements set out in these sections. The framework automates privacy monitoring combining threat detection with regulatory enforcement, improves overall compliance with privacy regulations, and reduces manual overhead.

References

- Abbas, Z. *et al.* (2024) 'Exploring deep federated learning for the internet of things: A GDPR-compliant architecture', *IEEE Access*, 12, pp. 10548–10574. doi:10.1109/access.2023.3344029.
- S. Chandra Sekaran *et al.* (2024) 'Nonlinear analysis and topological approaches towards a deep intelligent framework for privacy assurance of autonomous IOT Systems', *Communications on Applied Nonlinear Analysis*, 31(2), pp. 83–95. doi:10.52783/cana.v31.521.
- Garg, S. *et al.* (2023) 'Deep reinforcement learning for next-generation IOT Networks', *Computer Networks*, 228, p. 1. doi:10.1016/j.comnet.2023.109760.
- Abdel-Basset, M., Moustafa, N. and Hawash, H. (2022) 'Deep learning approaches for security threats in Iot Environments', *IEEE Press / Wiley*, pp. 1–384. doi:10.1002/9781119884170.
- Tariq, Usman, *et al.* (2024) Securing the evolving IoT with deep learning: a comprehensive review. Kurdish Studies, 12 (1). pp. 3426-3454. ISSN 2051-4883
- Goknil, A. *et al.* (2023) 'A systematic review of data quality in CPS and IOT for industry 4.0', *ACM Computing Surveys*, 55(14s), pp. 1–38. doi:10.1145/3593043.
- Okafor, N.U., Alghorani, Y. and Delaney, D.T. (2020) 'Improving data quality of low-cost IOT sensors in environmental monitoring networks using data fusion and machine learning approach', *ICT Express*, 6(3), pp. 220–228. doi:10.1016/j.icte.2020.06.004.
- Shurrab, M. *et al.* (2024) 'Overcoming cold start and sensor bias: A deep learning-based framework for IOT-enabled monitoring applications', *Journal of Network and Computer Applications*, 222, p. 103794. doi:10.1016/j.jnca.2023.103794.
- Janardhana, D.R., Manu, A.P. and Pavan Kumar, V. (2023) 'Detecting privacy attacks in IOT network using Deep Learning Models', *2023 IEEE 3rd Mysore Sub Section International Conference (MysuruCon)* [Preprint]. doi:10.1109/mysurucon59703.2023.10397002.
- Zhang, L. *et al.* (2023) 'A Bert-based empirical study of privacy policies' compliance with GDPR', *2023 IEEE Conference on Communications and Network Security (CNS)* [Preprint]. doi:10.1109/cns59707.2023.10288797.
- Rahat, T.A., Long, M. and Tian, Y. (2022) 'Is Your Policy Compliant?: A Deep Learning-based Empirical Study of Privacy Policies' Compliance with GDPR', *ACM Digital Library | WPES'22: Proceedings of the 21st Workshop on Privacy in the Electronic Society* [Preprint]. doi:10.1145/3559613.3563195.
- Ahmad, J., Li, F. and Luo, B. (2022) 'IoTPrivComp: A measurement study of privacy compliance in IOT apps', *Computer Security – ESORICS 2022*, pp. 589–609. doi:10.1007/978-3-031-17146-8_29.
- Subahi, A. and Theodorakopoulos, G. (2018) 'Ensuring compliance of IOT devices with their privacy policy agreement', *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)* [Preprint]. doi:10.1109/ficloud.2018.00022.
- Kumar, P. *et al.* (2021) 'PPSF: A privacy-preserving and secure framework using blockchain-based machine-learning for IOT-driven Smart Cities', *IEEE Transactions on Network Science and Engineering*, 8(3), pp. 2326–2341. doi:10.1109/tnse.2021.3089435.

- Selvakumar, B. *et al.* (2021) 'Deep Learning Framework for anomaly detection in IOT enabled systems', *Deep Learning for Security and Privacy Preservation in IoT*, pp. 99–111. doi:10.1007/978-981-16-6186-0_5.
- Gokdemir, A., & Calhan, A. (2022). Deep learning and machine learning based anomaly detection in internet of things environments. Journal of the Faculty of Engineering and Architecture of Gazi University, 37(4), 1945-1956.
- Otoum, Y., Liu, D., & Nayak, A. (2022). DL-IDS: a deep learning–based intrusion detection framework for securing IoT. Transactions on Emerging Telecommunications Technologies, 33(3), e3803.
- Kathamuthu, N.D. *et al.* (2022) 'Deep Q-learning-based neural network with privacy preservation method for secure data transmission in internet of things (IOT) healthcare application', *MDPI*, 11(1), p. 157. doi:10.3390/electronics11010157.
- Chakraborty, S., Pandey, S.K., Maity, S.K., & Dey, L. (2023). Detection and Classification of Novel Attacks and Anomaly in IoT Network using Rule based Deep Learning Model. *ArXiv, abs/2308.00005*.
- Kumar, C. and Chittora, P. (2023) 'Secure and privacy-preserving framework for IOT-enabled Smart Grid Environment', *Arabian Journal for Science and Engineering*, 49(3), pp. 3063–3078. doi:10.1007/s13369-023-07900-y.
- Aljuhani, A. *et al.* (2024) 'A deep-learning-integrated blockchain framework for securing industrial IOT', *IEEE Internet of Things Journal*, 11(5), pp. 7817–7827. doi:10.1109/jiot.2023.3316669.
- Golec, Muhammed *et al.* (2024) 'Priceless: Privacy enhanced ai-driven scalable framework for IoT applications in serverless edge computing environments', *Wiley Online Library* [Preprint]. doi:10.1002/itl2.510.
- Schiliro, F. *et al.* (2023) 'DeepCog: A trustworthy deep learning-based human cognitive privacy framework in Industrial Policing', *IEEE Transactions on Intelligent Transportation Systems*, 24(7), pp. 7485–7493. doi:10.1109/tits.2022.3166631.
- Kumar, R. and Tripathi, R. (2021) 'DBTP2SF: A deep blockchain-based trustworthy privacy-preserving secured framework in Industrial Internet of Things Systems', *Transactions on Emerging Telecommunications Technologies*, 32(4). doi:10.1002/ett.4222.
- Lin, H. *et al.* (2023) 'Privacy-aware access control in IOT-enabled healthcare: A federated deep learning approach', *IEEE Internet of Things Journal*, 10(4), pp. 2893–2902. doi:10.1109/jiot.2021.3112686.
- Popoola, S.I. *et al.* (2022) 'Federated deep learning for Zero-Day botnet attack detection in IOT-edge devices', *IEEE Internet of Things Journal*, 9(5), pp. 3930–3944. doi:10.1109/jiot.2021.3100755.
- Sivaranjani, R., Rao, P.M. and Saraswathi, P. (2021) 'ECC-based privacy-preserving mechanisms using deep learning for industrial IOT', *Deep Learning for Internet of Things Infrastructure*, pp. 25–59. doi:10.1201/9781003032175-2.
- Farooq, M.S. *et al.* (2022) 'Blockchain-based smart home networks security empowered with fused machine learning', *Sensors (Basel).*, 22(12), p. 4522. doi:10.3390/s22124522.
- Dr. G. Yedukondalu, Dr. Channapragada Rama Seshagiri Rao and Dugyala, R (2022) 'A novel framework for trustworthy privacy preserving machine learning model for industrial IOT systems using blockchain techniques', *International Journal of Scientific Research in Science and Technology*, pp. 611–618. doi:10.32628/ijsrst229498.

- Echenim, K.U. and Joshi, K.P. (2023) 'IOT-Reg: A comprehensive knowledge graph for real-time IOT data privacy compliance', *2023 IEEE International Conference on Big Data (BigData)* [Preprint]. doi:10.1109/bigdata59044.2023.10386545.
- Zhang, Y. *et al.* (2024) 'Pacta: An IOT data privacy regulation compliance scheme using TEE and Blockchain', *IEEE Internet of Things Journal*, 11(5), pp. 8882–8893. doi:10.1109/jiot.2023.3321308.
- Lane, N.D. *et al.* (2015) 'An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices', *Proceedings of the 2015 International Workshop on Internet of Things towards Applications*, pp. 7–12. doi:10.1145/2820975.2820980.
- Das, R. *et al.* (2018) 'A deep learning approach to IOT authentication', *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6. doi:10.1109/icc.2018.8422832.
- HaddadPajouh, H. *et al.* (2018) 'A deep recurrent neural network based approach for internet of things malware threat hunting', *Future Generation Computer Systems*, 85, pp. 88–96. doi:10.1016/j.future.2018.03.007.
- Tama, BA and Rhee, KH. (2017) 'Attack classification analysis of IOT network via Deep Learning Approach', *Research Briefs on Information and Communication Technology Evolution*, 3, pp. 150–158. doi:10.56801/rebicte.v3i.54.
- McDermott, C.D., Majdani, F. and Petrovski, A.V. (2018) 'Botnet detection in the internet of things using Deep learning approaches', *2018 International Joint Conference on Neural Networks (IJCNN)* [Preprint]. doi:10.1109/ijcnn.2018.8489489.
- Mohamed Shakeel, P. *et al.* (2018) 'Retracted article: Maintaining Security and Privacy in health care system using learning based deep-Q-networks', *Journal of Medical Systems*, 42(10). doi:10.1007/s10916-018-1045-z.
- Anselmi, G. *et al.* (2023) 'COPSEC: Compliance-oriented IOT security and Privacy Evaluation Framework', *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking* [Preprint]. doi:10.1145/3570361.3615747.
- Rizvi, S., Campbell, S. and Alden, K. (2020) 'Why compliance is needed for internet of things?', *2020 International Conference on Software Security and Assurance (ICSSA)*, pp. 66–71. doi:10.1109/icssa51305.2020.00019.
- Kibria, M.G. *et al.* (2017) 'A framework to support data interoperability in web objects based IOT environments', *2017 International Conference on Information and Communication Technology Convergence (ICTC)* [Preprint]. doi:10.1109/ictc.2017.8190935.
- *2020 unit 42 IOT threat report* (2024) *Unit 42*. Available at: https://unit42.paloaltonetworks.com/iot-threat-report-2020/ (Accessed: 18 July 2024).
- Raikar, A.S. *et al.* (2023) 'Advances and challenges in IOT-based Smart Drug Delivery Systems: A comprehensive review', *Applied System Innovation*, 6(4), pp. 62–70. doi:10.3390/asi6040062.

Appendix A

Authors	Proposed Systems	Strengths	Weaknesses	Adopted Strengths	Addressed Weakness
Abbas, Z et al. (2024)	Federated learning for GDPR-compliant IoT	Minimises data transfer and protects user privacy by reducing the amount of data that needs to be sent to the central server after each IoT device processes the data it collects locally. Integrity and confidentiality from the data being communicated due to the trustworthiness and secureness of the network the IoT device is connecting to.	Complexity of systems as it is not easy to implement. delayed processing times due to decentralised processing. Latency issues to federated learning-based framework	Federated Learning for decentralised data processing	The project will reduce overhead, model convergence, and latency
Chandra Sekeran, S. et al. (2024)	Nonlinear analysis and topological approaches towards a deep intelligent framework for privacy assurance of autonomous IoT Systems	Provides a robust framework for privacy in autonomous systems	Very complex to implement such a system. High computational complexity due to nonlinear and topological methods.	The robustness of the framework will be considered.	Streamlines computational processes. I will try to simplify the computational complexity for practical implementation
Garg, S, et al. (2023)	Deep reinforcement learning for next-generation IoT	Adaptive approach to dynamic network conditions and threats. Utilises Multi-Agent Reinforcement (MARL), enabling real-time adaptive decision making.	Very long training times for DLR models due to significant computational resources. High-quality, large datasets are necessary to train effective models.	The adaptive optimisation techniques can be used into my framework. Uses Deep Q-Networks for decision-making processes.	Ensuring the availability of high-quality data and optimising its usage to train models effectively.

Abdel-Basset, M., Moustafa, N. and Hawash, H. (2022)	Deep learning approaches for security threats in IoT environments	Only a review of deep learning models/frameworks and not an implementation	Solutions presented may need adaptation to be applicable to specific IoT contexts.	My project will leverage various deep learning models, such as CNNs, RNNs for comprehensive security threat detection	Focus on specific deep learning techniques, detailing implementation strategies to ensure practical applicability.
Tariq, Usman, *et al.* (2024)	Securing the evolving IoT with deep learning: 'Survey Paper'	Reviews multiple deep learning approaches	Most solutions may not be applicable to all IoT contexts	There is an understanding of the emerging IoT security threats	There are tailored solitons specific to IoT applications
Goknil, A. *et al.* (2023)	A systematic review of data quality in CPS and IoT for industry 4.0	Focusses on practical Applications in Industry 4.0	Does not have a proposed system, and focuses on data quality, rather than privacy or security. (Survey Paper)	Systematic approach to data quality	Proposed project can integrate data quality with privacy and security measures
Shurrab, M. *et al.* (2024)	Overcoming cold start and sensor bias: A deep learning-based framework for IOT-enabled monitoring applications	Focuses on practical challenges in IoT data quality and reliability (DL for IoT monitoring)	Limitations in scalability for diverse IoT applications.	Data quality enhancement	It does not directly address private compliance aspects
Lin, H et al. (2023)	Privacy-aware access control in IoT-enables healthcare: a deep learning approach	Enhances privacy by processing data locally.	Only tailored for healthcare applications (IoT-Health) utilising FDL. Not using an attribute-based Secure access control mechanism (SACM)	The framework incorporates access control policies to regulate data access, ensuring compliance with privacy regulations and protecting patient information from unauthorized access.	Limited applicability to other IoT domains

Critically Analysed Papers:

Janardhana, D.R., Manu, A.P. and Pavan Kumar, V. (2023)	Detecting privacy attacks in IoT using deep learning	Utilises advanced deep learning models for accurate detection. Good for privacy detection	Focuses on detection rather than prevention. Reactive approach Limited compliance discussion	The DL techniques from here can be used to detect privacy attacks in IoT networks.	Ensure regulatory compliance. The proposed project can use proactive measures to prevent privacy attacks.
Ahmad, J., Li, F. and Luo, B. (2022)	Measurement study of privacy compliance in IoT apps	Gives insight into the compliance levels of various IoT apps. Introduces a framework for assessing privacy compliance. IoTPrivComp achieves a high overall accuracy of 94% in identifying privacy disclosures and inconsistencies.	Very limited scope. Resource intensive. The integrations of BERT and the extensive ontology processing requires significant computational resources, which might be a limitation.	The detailed consistent analysis between data flows and privacy policies ensures thorough privacy compliance.	Can focus on enhancing compliance rather than just measuring it.
Proposed Product	Leveraging a Deep Learning-Based Privacy Compliance Framework for IoT Applications	Utilises deep learning techniques (CNNs and RNNs) for real-time privacy threat detection. Aims to dynamically adjusts privacy policies based on detected threats. Focuses on compliance with regulations (GDPR, CCPA, NIST IoT Cybersecurity Guidelines)	The complexity of the implementation of this product is due to the scale and heterogeneity of IoT application. Difficulty in addressing the ever-evolving and comprehensive privacy regulations. Challenges in real-time data flow monitoring in IoT environment.	Incorporates and adopts deep learning techniques from various studies for privacy threat detection. Utilises a systematic approach to data quality.	Combining threat detection and policy adjustment, a basic framework is leveraged/implemented to work together for a more generalisable solution applicable to the IoT context, in this case, a specific IoT application. Focusing on both the proactive measures and privacy protection.