IoT

Article

Advanced Deep Learning Approach for Smart Home Appliance Identification Using Recurrent Neural Networks with LSTM

Sana Abdelaziz Bkheet [1], Johnson I. Agbinya [2] and Gamal Saad Mohamed Khamis [1,*]

[1] Department of Computer Science, College of Science, Northern Border University (NBU), Arar 73213, Saudi Arabia; sana.bkheet@nbu.edu.sa or sanaaziz569@gmail.com
[2] School of Information Technology and Engineering, Melbourne Institute of Technology, Melbourne, VIC 3000, Australia; jagbinya@mit.edu.au
* Correspondence: gamal.khamees@nbu.edu.sa or jamalziena@gmail.com

Abstract: In the Internet of Things (IoT) domain, vast numbers of smart devices are interconnected, generating large volumes of data requiring advanced management mechanisms. One major challenge in smart environments is the ability to accurately distinguish and categorize the various types of objects within these systems. To address this issue, the study introduces a recurrent neural network (RNN) model designed for classifying data from smart home devices. Using a dataset from Kaggle, the research outlines the processes of data collection, loading, normalization, and model development. The RNN, enhanced with long short-term memory (LSTM) layers, was trained and evaluated, showing notable improvements in training and validation accuracy over ten epochs. The model achieved a test accuracy of 83.25%, a loss of 35.4%, a precision of 85%, and a recall of 81%. The evaluation of the model on the test set includes a detailed analysis using ROC curves, area under the curve (AUC) scores for multi-class classification, and a confusion matrix. With an AUC score of 0.9896, the model demonstrated exceptional performance in accurately classifying IoT device categories. These results suggest that the LSTM-equipped RNN offers strong learning efficiency and generalization, making it a highly suitable approach for IoT device classification. Additionally, the article explores the concept of IoT and reviews recent advancements in using deep learning models across various IoT sectors, including smart homes, industrial systems, and healthcare. Future research could aim to improve the model's real-time processing abilities and scalability and incorporate a wider variety of IoT data types to enhance its practical applications and expand its utility across more IoT environments.

Keywords: the Internet of Things (IoT); smart objects; recurrent neural network (RNN); long short-term memory (LSTM)

check for updates

Citation: Bkheet, S.A.; Agbinya, J.I.; Khamis, G.S.M. Advanced Deep Learning Approach for Smart Home Appliance Identification Using Recurrent Neural Networks with LSTM. *IoT* **2024**, *5*, 835–851. https://doi.org/10.3390/iot5040038

Academic Editor: Amiya Nayak

Received: 7 October 2024
Revised: 30 October 2024
Accepted: 18 November 2024
Published: 25 November 2024

1. Background

The Internet of Things (IoT) connects billions of physical devices or individuals to the Internet and has become a trendy and innovative technology over the past decade [1]. These physical devices are known as "things" or "objects" [2]. This intelligent methodology uses various protocols to exchange information through sensing devices. It expands the Internet, enabling the identification, location, and tracking of things. This approach allows for the development of small-scale devices with unique identification and computing capabilities, embedded with sensors and actuators and connected through wireless and wired sensor networks. The main characteristics of IoT include:

- Perception: Involves using sensors, radio frequency identification (RFID), and barcodes for data collection. These technologies allow for the identification and tracking of objects, facilitating the recognition of their properties and locations in various environments. RFID, for example, has been widely adopted for its ability to uniquely identify objects and gather data about them, enabling applications beyond traditional identification purposes. Integrating RFID with wireless sensor networks (WSNs) makes creating more complex IoT systems that enhance data acquisition capabilities possible. Combining these technologies ensures more precise monitoring of environments and assets, which is crucial for advanced IoT applications like smart homes and industrial automation [3,4].

- Transmission: Focuses on the reliable exchange of these collected data over communication networks. IoT systems leverage advanced networking technologies to ensure data are transmitted between devices, enabling machine-to-machine (M2M) and mobile-to-machine interactions. This aspect is crucial in maintaining the seamless connectivity required for real-time monitoring and control in IoT applications. Reliable data transmission ensures that insights derived from sensor data can be accessed and utilized when needed, optimizing the responsiveness of IoT systems. These perspectives are supported by studies highlighting the integration of RFID with WSNs, which helps to overcome connectivity and energy efficiency challenges in IoT settings. Additionally, advances in augmented RFID technologies are expanding the potential of IoT systems, enabling more sophisticated sensing and data transmission capabilities [3–5].
- Processing: Cloud computing facilitates the intelligent processing of IoT data. Service providers use cloud computing to process millions or billions of data points [5].

IoT can be described as the "Internet of Everything" or the "Industrial Internet". It represents the latest technology that connects machines and devices. This field is crucial for the future of technology and has garnered significant attention from users and industries alike [2]. IoT provides innovative solutions to various challenges faced by businesses, governments, and enterprises worldwide by utilizing smart devices and the internet. IoT is increasingly becoming an integral part of our lives, evidenced by its widespread presence. IoT integrates a vast array of smart systems, frameworks, intelligent devices, and sensors [6,7].

IoT technology has become essential in our lives; it covers various applications, from everyday consumer electronics to specialized industrial systems, such as fitness-tracking wristwatches, transport logistics, smart cars, manufacturing, and smart grids. Depending on their implementation, IoT devices can be used for real-time alerts, data archiving, trend analysis, and forecasting by utilizing related technologies like cloud services. Additionally, IoT has proven beneficial for both small- and large-scale networks, leading to a vast array of enabling hardware and software of varying complexities. This technology has influenced critical sectors like healthcare, smart water management, surveillance, biomedical applications, industrial processes, data center management, agriculture, body area networks (BANs), and more [1,2,5,8].

Owing to the rapid growth in this field and the increasing number of diverse objects connected to various aspects daily, this paper focuses primarily on identifying smart home objects, which is necessary to distinguish each object from others. The approach utilizes a recurrent neural network (RNN) with long short-term memory (LSTM). Applying RNNs to smart home objects enhances the ability to transfer and use the same model to other smart environments. These environments play a significant role in IoT applications, impacting human safety, comfort, welfare, and security.

The rest of the paper is organized as follows: Section 2 presents previous work related to the use of deep learning algorithms in the identification of smart objects. Section 3 discusses in detail the steps and procedures of the development process of the RNN model for classifying smart home objects. Section 4 reports an analysis of the proposed model's outcomes. Section 5 provides the paper's conclusion and recommendations for possible future work.

2. Related Works and Studies

In recent years, numerous studies have focused on applying deep learning techniques to identify smart objects within IoT environments. These studies highlight the effectiveness and versatility of deep learning in enhancing the functionality and intelligence of smart systems. This section will briefly discuss some of the research efforts on IoT device identification that are most relevant to this work.

The paper [9] explores generating embeddings for IoT devices in a smart home using Word2Vec, termed IoT2Vec. The study aims to identify IoT devices based on their usage patterns and to find suitable replacements for malfunctioning devices. The proposed method involves creating word embeddings for IoT devices based on their activity footprints, which can be utilized for applications such as identifying similar devices, determining replacements, and building a location classifier based on IoT devices.

The authors review related work on applying machine learning to find similar IoT devices and propose a model to encode usage patterns as word embeddings, which can aid in identifying IoT devices based on their activity patterns. They also present a method to identify the device type of an unknown IoT device from its activity logs based on the similarity of its embeddings with stored embeddings of known devices.

The experimental validation involves analyzing a dataset from the CASAS Kyoto dataset, creating word embeddings for various devices, and using these embeddings to identify devices. The analysis includes examining trends in IoT device activations for different session gaps and determining the contextual similarity of devices based on their activity patterns.

The paper concludes that IoT devices in similar areas in a household exhibit identical usage patterns, making it feasible to recognize IoT devices based on their embeddings. Furthermore, the authors plan to investigate multiple datasets, generalize the approach, and focus on the activity generated by smart IoT devices to gain a higher-level understanding of users' tasks. In conclusion, the paper presents a method to create word embeddings for IoT devices based on their usage patterns, demonstrating the feasibility of recognizing devices based on their activity. The proposed approach has potential applications in identifying similar devices, determining replacements, and building location classifiers based on IoT devices. The authors also outline further questions for analysis and plan to explore more use cases and higher-level understandings of user tasks in future research [9].

The authors in [10] present a novel IoT device identification method called CBBI, which uses a hybrid neural network model, Conv-BiLSTM, to learn spatial and temporal features from network traffic automatically. The study addresses the security risks associated with the increasing number of IoT devices connected to networks and the need to identify these devices accurately. The proposed approach overcomes the limitations of traditional methods that rely on manually extracted features and prior knowledge, increasing the difficulty and reducing the real-time performance of device identification.

The document discusses the rapid growth of IoT technology and the vulnerabilities associated with IoT devices, leading to increased potential attacks. It emphasizes the importance of accurately identifying IoT devices for implementing network access control and security measures. The challenges with existing methods, including the tedious and time-consuming feature extraction process, the complexity of feature engineering, and the limitations in recognizing subtle features, are highlighted.

The proposed CBBI approach consists of three modules: data preprocessing, data augmentation, and Conv-BiLSTM. The data preprocessing module converts raw network traffic into an input suitable for deep learning models, while the data augmentation module addresses data imbalance in deep learning. The Conv-BiLSTM module utilizes a hybrid deep learning model to learn spatial and temporal features simultaneously, improving the accuracy and generalization ability of the model. The study evaluates the CBBI approach using public and laboratory datasets, achieving accurate identification of IoT devices. The main contributions of the proposed approach include eliminating the need for prior knowledge in feature engineering, extracting spatial and temporal features, and using data augmentation to solve data imbalance, resulting in improved model accuracy.

The study also provides an overview of related work on IoT device classification, discussing various methods based on classification models and active detection. Additionally, it presents an in-depth explanation of the proposed framework, including the data preprocessing algorithm, the FGAN module, and the Conv-BiLSTM model.

In summary, the document introduces the CBBI approach for identifying IoT devices based on spatial and temporal features from network traffic. It addresses the limitations of existing methods and provides a comprehensive evaluation of the proposed approach, highlighting its potential to identify IoT devices accurately [10].

The authors in [11] present an effective machine learning-based IoT device identification scheme, iotID. The scheme extracts 70 TCP flow features from three aspects: remote network servers and port numbers, packet-level traffic characteristics like packet inter-arrival times, and flow-level traffic characteristics like flow duration. The study considers the imbalanced nature of network traffic generated by various devices in both the learning and evaluation phases. The performance of iotID is evaluated on network traffic collected in a typical smart home environment with both IoT and non-IoT devices, achieving a balanced accuracy score above 99%. Future work will explore evaluating iotID with additional IoT devices and studying deployment scenarios where one may be preferred over another [11].

To validate the efficacy of iotID, the authors conduct performance studies utilizing network traffic data collected from a typical smart home environment, comprising a mix of IoT and non-IoT devices. The results demonstrate that iotID achieves an outstanding balanced accuracy score exceeding 99%. This underscores its robust capability to accurately identify IoT devices within complex network environments, showcasing its potential for practical implementation in real-world scenarios [11].

The research article [12] discusses the framework for identifying and classifying IoT devices for security analysis in a heterogeneous network. The study focuses on the challenges posed by Internet of Things (IoT) technology and the need to secure and protect the data exchanged by IoT devices. It emphasizes the importance of distinguishing between IoT devices and non-IoT devices and classifying legitimate IoT devices into their specific categories to ensure better quality of service management in the network. The proposed framework utilizes a hierarchical deep neural network (HDNN) to achieve higher accuracy in distinguishing and classifying IoT devices. The paper outlines the structure and functionality of IoT networks, highlighting the diverse nature of IoT devices and the challenges posed by their presence in the network. It discusses the vulnerabilities and security risks associated with IoT devices, emphasizing the need for robust security solutions and classification mechanisms to identify unauthorized devices and ensure data security. It also delves into the details of the methodology employed, including data construction and modeling, hyperparameter setting, and performance evaluation of the proposed framework using a hierarchical deep neural network. It provides a detailed analysis of the accuracy and loss curves, as well as confusion matrices and classification reports, to demonstrate the effectiveness of the proposed framework in accurately identifying and classifying IoT devices in a heterogeneous network.

In conclusion, the document presents a comprehensive overview of the research article, emphasizing the significance of the proposed framework for identifying and classifying IoT devices in a heterogeneous network. It highlights the superior performance of the hierarchical deep neural network in achieving high accuracy in distinguishing and classifying IoT devices, making it a valuable contribution to IoT security and network management [12].

Some papers have highlighted the use of deep neural networks in classification tasks such as text classification, such as the article [13], which discusses the development of tiny recurrent neural network (RNN) models for on-device text classification tasks, aiming to address the challenges of deploying deep neural networks (DNNs) on mobile devices due to high computational and memory requirements. The paper proposes a new training scheme that minimizes information loss during model compression by maximizing the mutual information between the feature representations learned from large and tiny models. Additionally, a certifiably robust defense method, named GradMASK, is introduced to defend against character-level perturbations and word substitution-based attacks. The proposed method involves masking a certain proportion of words in an input text, guided by the gradient values and uses the average logits produced by the large model from the masked adversarial examples for soft label knowledge distillation in the training scheme. The paper presents extensive experiments demonstrating the approach's effectiveness by comparing the tiny RNN models with compact and compressed RNN models in clean and adversarial test settings.

The paper's introduction mentions the importance of mobile artificial intelligence (AI) in various domains. It discusses the obstacles in deploying deep neural networks on mobile devices due to high computational and memory requirements. The paper focuses on designing tiny RNN models for text classification tasks to address these challenges, particularly in natural language processing (NLP) applications. The proposed tiny models are designed to reduce the parameters of the embedding layer, and a new training scheme is introduced to minimize information loss during model compression by maximizing the mutual information between the features learned from large and tiny models. Additionally, the paper presents a certifiably robust defense method, named GradMASK, which masks a certain proportion of words in an input text to defend against adversarial attacks. The proposed method is evaluated through extensive experiments demonstrating its effectiveness compared to other compact and compressed RNN models.

The experimental results demonstrate the superiority of the proposed tiny RNN models over other compact and compressed RNN models in terms of clean sample accuracy and adversarial robustness. Furthermore, an ablation study is conducted to examine the effectiveness of the critical components in the proposed model, revealing that all components contribute to improving the tiny model's classification performance and adversarial robustness. Additionally, the paper investigates the effect of embedding dimension and latent feature size on model compression performance, selecting the most miniature model with an embedding dimension of 5 and a latent feature size of 5 as the final tiny model to be deployed on mobile devices. The document also includes comparisons with other state-of-the-art methods and discusses the significance of the proposed approach in the context of on-device NLP applications [13].

The research article [14] presents a deep learning approach for identifying known and unauthorized IoT devices in network traffic, with over 99% accuracy. The method is simple, requires no feature engineering, and applies to any IoT device, regardless of the communication protocol. Future research plans to explore applications to different network protocols without a TCP/IP network stack. The increasing use of IoT devices in organizations has increased the risk of attacks owing to their less secure nature. To address this, organizations often implement security policies allowing only white-listed IoT devices. Organizations must identify connected IoT devices to monitor adherence to these policies, mainly unknown ones. The study applies deep learning to network traffic to automatically identify connected devices, achieving over 99% accuracy in identifying 10 different devices and traffic of smartphones and computers [14].

The Internet of Things (IoT) allows physical objects to communicate but poses battery, power, connectivity, and security issues. To address these, an automated system is needed to identify and report abnormalities, distinguish between approved and legitimate devices, and isolate malicious and non-malicious traffic sources. The research in [15] proposes a framework-based convolutional neural network (CNN) to address battery/power, communication, and security challenges in Internet of Things (IoT) devices. The CNN can identify allowed and authentic devices, segregate hostile and malicious IoT devices, and improve QoS management. The system accurately categorizes IoT devices and differentiates between IoT and non-IoT devices, ensuring compliance and security [15].

3. Methodology

3.1. Introduction to Deep Neural Networks (DNNs)

This section provides a comprehensive background on deep neural networks (DNNs), starting from foundational concepts and progressing toward more advanced depths. It aims to build a thorough understanding by covering the fundamental principles before delving into the details of the proposed model.

Artificial neural networks (ANNs) are a subset of artificial intelligence (AI) that mimic the structure and behavior of the human brain. AI-driven data analysis techniques, including machine learning (ML) and deep learning (DL) algorithms, can process extensive data volumes, such as those generated in IoT environments, yielding valuable insights. ANNs rely on three fundamental components: input and activation functions, weights associated with each input connection, and summation function. Deep learning (DL) algorithms, a recent evolution from machine learning and soft computing techniques, leverage ANNs to outperform traditional ML methods by analyzing intricate data without extensive human intervention. DL has heightened power and flexibility owing to its ability to efficiently process many features and exhibit enhanced classifier performance when confronted with large datasets [16–20].

Deep learning has witnessed an unprecedented surge in interest and development in recent years, revolutionizing the landscape of artificial intelligence (AI) applications. Inspired by the intricate neural networks of the human brain, deep learning methods have demonstrated remarkable capabilities in tackling complex problems across diverse domains. The following points provide a brief overview of various deep learning methods and their applications.

1. Convolutional neural networks (CNNs): CNNs have proven instrumental in image recognition, computer vision, and pattern analysis. Recent advancements in CNNs, such as attention mechanisms and transfer learning, have significantly improved their performance in tasks ranging from medical image analysis to autonomous driving [21].
2. Recurrent neural networks (RNNs): Recurrent neural networks (RNNs) are specifically designed to recognize handwritten text, speech recognition, time series analysis, and other sequential data types. They are renowned for their power and versatility and are among the most valuable types of neural networks. Moreover, RNNs can even be applied to images that can be decomposed into patches and treated as sequential data, showcasing their adaptability across various domains and data formats. The introduction of long short-term memory (LSTM) and gated recurrent unit (GRU) architectures has enhanced the ability of RNNs to capture long-term dependencies, addressing challenges in various applications. RNNs have various architectures:
   - ➢ Stacked RNN: The stacked RNN architecture comprises multiple RNN layers stacked together, with each layer possessing its memory unit. Unlike the simple RNN, which consists of a single hidden layer, the stacked RNN architecture allows for deeper networks, enhancing the model's ability to capture context and temporal dependencies.
   - ➢ Bidirectional RNN: Bidirectional RNNs combine two RNNs, one processing the input sequence normally and the other in reverse time order. This configuration enables the neural network to simultaneously leverage past and future information, aiding in more comprehensive sequence understanding. The outputs of the two RNNs are later merged to form the final output.
   - ➢ LSTM (long short-term memory) Networks: LSTM networks address issues encountered in training conventional RNNs, such as vanishing or exploding gradients. They incorporate memory cells and gate mechanisms (input, output, and forget gates) to retain and utilize information over long sequences effectively. LSTMs can be stacked to create deeper architectures, improving performance in tasks requiring long-term dependencies.
   - ➢ GRU (gated recurring unit) networks: Similar to LSTMs, GRU networks utilize gated mechanisms to control the flow of information. However, they have fewer parameters than LSTMs owing to the absence of an output gate. While LSTMs generally excel on larger datasets, GRUs offer a more lightweight alternative with competitive performance [22,23].

3. Generative adversarial networks (GANs): GANs are a deep learning framework that consists of two models: a generative model (G) and a discriminative model (D). The objective of G is to capture the distribution of target data, while D aids in training G by comparing the generated data to actual data. GANs were first introduced by Goodfellow et al. in 2014 as a pair of simple neural networks. The often used analogy of GANs is counterfeit moneymaking, with G as the counterfeiter and D as the bank trying to identify fake bills. GANs have succeeded in computer vision, feature representation, and natural language processing tasks. GANs have gained prominence in generating realistic and high-quality synthetic data. This has applications in image generation, style transfer, and data augmentation. Recent research on GANs has focused on improving training stability, diversity, and controllability, expanding their utility in creative fields and data synthesis [24,25].
4. Unsupervised pre-trained networks (UPN): Unsupervised pre-trained networks (UPN) are deep neural networks trained using unsupervised learning techniques before fine-tuning with supervised learning. These networks initialize their parameters using unsupervised learning algorithms like autoencoders or RBMs. They then use supervised learning methods like backpropagation to adapt their representations for tasks like classification or regression. UPNs capture meaningful patterns and structures in data without relying on labeled examples, improving performance on downstream tasks and avoiding issues like overfitting [26].
5. Deep belief networks (DBNs): DBNs are a type of deep learning with a multi-layered generative graphical model consisting of both restricted Boltzmann machines (RBMs) and autoencoders or other unsupervised learning methods, along with a final layer for supervised learning, such as a softmax classifier. DBNs are typically composed of two main components: the generative pre-training and fine-tuning phases. During the generative pre-training phase, each layer of the DBN is trained greedily in an unsupervised manner, where each layer learns to represent the data hierarchically. This pre-training process initializes the network weights and allows it to learn useful features from the input data. Once the pre-training phase is complete, the DBN is fine-tuned using supervised learning techniques like backpropagation. During fine-tuning, the entire network is trained jointly to minimize a loss function, which enables it to learn to classify or regress on the input data [22].
6. Transformers: Transformers have emerged as a groundbreaking architecture in natural language processing, enabling breakthroughs in machine translation, text summarization, and language understanding. The attention mechanism in transformers has paved the way for models like bidirectional encoder representations from transformers (BERT) and generative pre-trained transformers (GPTs), setting new benchmarks in language-related tasks [27].
7. Autoencoders: Autoencoders play crucial roles in unsupervised learning and data compression. Recent advancements in variational autoencoders (VAEs) and denoising autoencoders have enhanced their ability to learn meaningful representations, contributing to applications in anomaly detection and feature learning [25].

The Basic Mathematics Used in Deep Learning

Deep learning (DL), with its many domains such as computer vision, natural language processing, speech recognition, healthcare and biomedicine, environmental science, gaming, etc. requires a solid foundation in mathematical concepts. Linear algebra, denoted by $R^{m \times n}$, lies at the core of continuous mathematics, encompassing scalars, vectors, matrices, tensors, norms, eigen decompositions, singular value decompositions, and more [22].

Optimization theory, crucial for model training, aims to minimize training errors by adjusting model weights. It involves understanding derivatives. Loss functions $\left[\frac{d}{d\theta}\right]$ concerning parameters are essential for gradient descent optimization. Knowledge of gradients (∇), Hessians, convergence criteria, and related concepts are pivotal.

Probability and statistics $(P(x), \mu, \sigma)$ play an indispensable role in handling uncertain data in machine intelligence. Probability theory aids in addressing uncertainty-related problems, while statistics facilitates normalization, distribution analysis, computation of means, and standard deviations, contributing to faster convergence [22].

Multivariable calculus, applicable in deep learning, is relevant for error minimization and data extrapolation. While single-variable calculus deals with the functions of one variable $y = (f(x))$, multivariable calculus extends to functions of multiple variables $y = (f(x, y))$. This extension enables the application of calculus principles to scenarios in three-dimensional and higher-dimensional spaces, adapting methodologies from single-variable calculus to analogous situations in multiple dimensions. Presently, deep learning finds application across diverse domains, including intelligent video analytics, hyper-spectral imagery analysis, image recognition, natural language processing, automatic email responses, machine translation, fraud detection, and healthcare. The key advantages of deep learning lie in its simplicity, versatility, rapid development cycle, and high-performance capabilities [18,22].

In the Internet of Things (IoT) context, DL algorithms play a pivotal role, offering significant advantages in applications such as intelligent transportation systems, smart manufacturing, traceable logistics, and enhanced social networks. Despite their transformative potential, DL algorithms face challenges. Training DL models requires substantial time due to learning from extensive data, and their execution often demands specialized computing hardware like graphics processing units (GPUs) or tensor processing units (TPUs) for optimal performance. Furthermore, DL models tend to have substantial sizes, leading to increased storage and computational expenses. These stringent requirements impede the widespread adoption of DL in IoT scenarios, where devices frequently possess limited computing and storage capacities [16,17].

There are two primary reasons why an LSTM-based RNN is a well-suited choice for IoT device classification:

Effective Handling of Sequential Data: IoT device data often exhibit temporal patterns that are essential for accurate classification. These patterns could represent daily or weekly usage cycles, specific activation sequences, or variations in sensor readings over time. LSTM networks, by design, are adept at processing such sequential information. They can capture the order and relationships within the data, leading to more robust device identification.

Learning Long-Term Dependencies: Historical usage data can be precious for differentiating between different types of IoT devices. LSTM cells, with their ability to remember and utilize past information, can effectively learn these long-term dependencies. For instance, an LSTM model might discover that a sudden spike in power consumption followed by a sustained period of low activity indicates a specific smart appliance [26,27].

The following section represents the construction of a deep neural network (DNN) model for classifying smart home objects using a recurrent neural network (RNN) with long short-term memory (LSTM) architecture.

3.2. The Data and Method

This part presents the development process of the RNN model for classifying IoT device data. Figure 1 depicts the critical steps in constructing and assessing the proposed model. The diagram visually represents the main procedural stages, highlighting the model development and evaluation process.

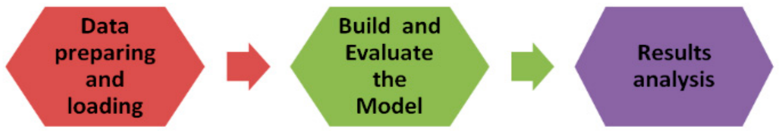


Figure 1. Procedural stages of the proposed model.

3.2.1. Data Preparation and Loading

This paper utilizes a dataset available on Kaggle, which can be accessed through the following URL: [https://www.kaggle.com/datasets/fanbyprinciple/iot-device-identification (30 September 2024)]. According to Kaggle, this dataset is derived from Chapter 5 of the *Machine Learning Cookbook for Cyber Security*. It includes network traffic analysis data from a smart home environment, encompassing various IoT devices, and was initially generated by other researchers.

The data come in two separate CSV files: the test and training files. To explore the content of each file, both files are loaded into Google Colab. The dataset contains 900 rows $\times$ 298 columns; rows represent the total number of instants, while columns represent the attributes.

Table 1 shows an array of strings containing the names of different smart home devices. The array is one-dimensional and has 10 elements. Each component is a string representing a device's name. It represents the label or the class feature.

Table 1. The array string of "device_category" in the dataset.

Title of Device in Array Format	Total Number
array(['baby_monitor', 'lights', 'motion_sensor', 'security_camera', 'smoke_detector', 'socket', 'thermostat', 'TV', 'watch', 'water_sensor'], dtype = object)	10

3.2.2. Split the Dataset

At least three methods are employed for computing a classifier's accuracy. One approach involves splitting the training set. The cross-validation technique partitions the training set into mutually exclusive and equally sized subsets. For each subset, the classifier is trained on the combination of all other subsets, and the average error rate across these subsets provides an estimate of the classifier's error rate. Leave-one-out validation is a specific instance of cross-validation where each test subset consists of a single instance. While computationally more intensive, this validation method is valuable when the most precise estimate of a classifier's error rate is necessary [23]. In this study, we employ percentage split as the evaluation method, where the dataset is split into 20% for testing and 80% for training.

3.2.3. Build and Evaluate the Model

This section will explore the details of using TensorFlow and Keras to construct a recurrent neural network (RNN) featuring long short-term memory (LSTM) architecture.

The main step in constructing the model is reconfiguring the input features for the training and test sets into a three-dimensional format. The prescribed format for LSTM networks is [samples, time steps, features], which ensures effective handling of the sequential structure inherent in the data.

The model defines the architecture of the RNN model, which consists of two LSTM layers with dropout for regularization and a dense layer with softmax activation for the output layer in a multi-class classification scenario. Here is an explanation of each step of the model:

➢ Model Initialization: A sequential model is initialized, creating a structure in which each layer is directly connected to the next, ensuring a straightforward data flow through the model.

➢ Adding the First LSTM Layer: The command 'model. add (LSTM (200, return sequences = True, input shape = (X train. shape [1], X train. shape [2])))' adds the first long short-term memory (LSTM) layer to the model. This layer consists of 200 units (neurons) and uses the 'return sequences=True' parameter, which is necessary when stacking multiple LSTM layers to ensure that the output maintains a sequence format.

The 'input shape' parameter specifies the dimensions of the input data, where 'X_train. shape [1]' indicates the number of time steps, and 'X train. shape [2]' indicates the number of features.

- ➢ Dropout Layer: A dropout layer is introduced with a dropout rate of 0.2. Dropout is a regularization method that randomly sets a portion of the input units to zero during training. This helps to mitigate overfitting by preventing the model from becoming too dependent on specific neurons.
- ➢ Adding the Second LSTM Layer: A second LSTM layer with 200 units is added to the model. Unlike the previous LSTM layer, this one does not require 'return sequences = True' because it is the final LSTM layer in the sequence and does not need to maintain the sequence output for further layers. This includes another dropout layer with a dropout rate of 0.2 for regularization.
- ➢ Adding a dense (fully connected) layer with several units equal to the number of classes in the output. The activation function is set to 'softmax', which is typical for multi-class classification problems.
- ➢ Configuring the model for training. It specifies the optimizer as 'Adam', the loss function as 'categorical_crossentropy' (suitable for multi-class classification), and includes accuracy as the evaluation metric.

The provided model architecture comprises two long short-term memory (LSTM) layers with dropout regularization to prevent overfitting. The final layer is a dense layer designed for multi-class classification, with the softmax activation function as shown in Figure 2. The model is compiled using the Adam optimizer and the categorical cross-entropy loss function, making it suitable for training on multi-class classification tasks. For more explanation, Figure 3 shows the RNN structure with 200 units in LSTM layers.

$$y_1 = soft\max(w_x x_1 + w_h h_0) \tag{1a}$$

$$y_2 = soft\max(w_x x_2 + w_h h_1) \tag{1b}$$

$$\cdots$$

$$y_n = soft\max(w_x x_n + w_h h_{n-1}) \tag{1n}$$

The study employs an RNN with LSTM layers owing to their effectiveness in handling sequential data, such as time series and network traffic. LSTM layers are particularly suited for capturing long-term dependencies within the data, making them ideal for tasks where the information order is crucial. To mitigate the risk of overfitting, dropout layers are included after each LSTM layer, randomly deactivating specific neurons during training. This approach encourages the model to learn more generalized and robust features. The model concludes with a dense output layer that uses a softmax activation function, a standard choice for multi-class classification problems. The softmax function converts the output into a probability distribution, ensuring that the predicted probabilities for all classes sum to one, thereby representing the likelihood of each class.

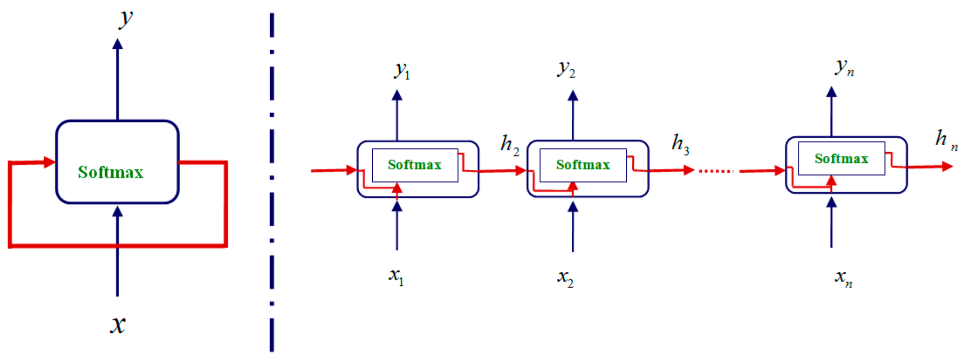


Figure 2. Illustration of applying the softmax activation function.

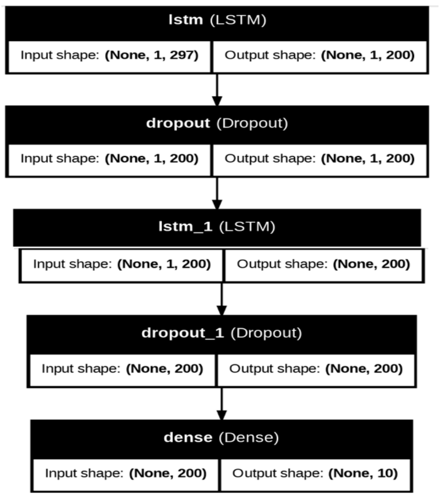

lstm (LSTM)
Input shape: (None, 1, 297)
Output shape: (None, 1, 200)
dropout (Dropout)
Input shape: (None, 1, 200)
Output shape: (None, 1, 200)
lstm_1 (LSTM)
Input shape: (None, 1, 200)
Output shape: (None, 200)
dropout_1 (Dropout)
Input shape: (None, 200)
Output shape: (None, 200)
dense (Dense)
Input shape: (None, 200)
Output shape: (None, 10)

Figure 3. Illustration of the RNN structure with 200 units in LSTM layers.

3.2.4. Train the Model

The previously defined LSTM model was trained using the fit method. Here are the details of each parameter:

➢ X_train and y_train: These represent the training data, where X_train consists of the input features, and y_train contains the corresponding target labels.

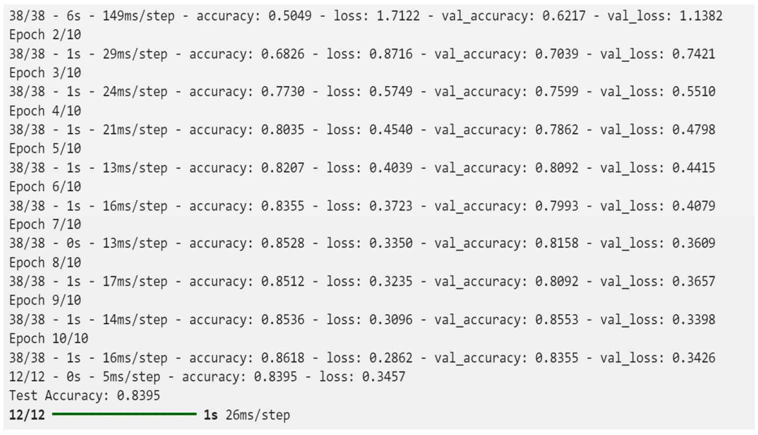
➢ Epochs = 10: The number of epochs indicates how many times the entire training dataset is passed through the neural network in both forward and backward directions. In this case, the model will be trained over 10 epochs.

➢ Batch size = 32: The batch size specifies the number of samples used in each iteration for updating the model weights. A batch size of 32 means that 32 samples from the training dataset will be utilized in each iteration.

➢ Validation split = 0.2: This parameter allocates 20% of the training data as a validation set. The model's performance on this subset is monitored during training, providing insights into its generalization capabilities.

➢ Verbose = 2: This parameter controls the verbosity of the training process. A value of 2 means that training progress will be displayed after each epoch, providing detailed feedback on the training process.

➢ The output returns two values: test loss and test accuracy. Test loss measures how well the model's predictions match the actual values in the test dataset. Lower test loss indicates better performance. Test accuracy measures the percentage of correct predictions made by the model on the test dataset. Higher test accuracy indicates better performance. The training output is illustrated in Figure 4.



```
38/38 - 6s - 149ms/step - accuracy: 0.5049 - loss: 1.7122 - val_accuracy: 0.6217 - val_loss: 1.1382
Epoch 2/10
38/38 - 1s - 29ms/step - accuracy: 0.6826 - loss: 0.8716 - val_accuracy: 0.7039 - val_loss: 0.7421
Epoch 3/10
38/38 - 1s - 24ms/step - accuracy: 0.7730 - loss: 0.5749 - val_accuracy: 0.7599 - val_loss: 0.5510
Epoch 4/10
38/38 - 1s - 21ms/step - accuracy: 0.8035 - loss: 0.4540 - val_accuracy: 0.7862 - val_loss: 0.4798
Epoch 5/10
38/38 - 1s - 13ms/step - accuracy: 0.8207 - loss: 0.4039 - val_accuracy: 0.8092 - val_loss: 0.4415
Epoch 6/10
38/38 - 1s - 16ms/step - accuracy: 0.8355 - loss: 0.3723 - val_accuracy: 0.7993 - val_loss: 0.4079
Epoch 7/10
38/38 - 0s - 13ms/step - accuracy: 0.8528 - loss: 0.3350 - val_accuracy: 0.8158 - val_loss: 0.3609
Epoch 8/10
38/38 - 1s - 17ms/step - accuracy: 0.8512 - loss: 0.3235 - val_accuracy: 0.8092 - val_loss: 0.3657
Epoch 9/10
38/38 - 1s - 14ms/step - accuracy: 0.8536 - loss: 0.3096 - val_accuracy: 0.8553 - val_loss: 0.3398
Epoch 10/10
38/38 - 1s - 16ms/step - accuracy: 0.8618 - loss: 0.2862 - val_accuracy: 0.8355 - val_loss: 0.3426
12/12 - 0s - 5ms/step - accuracy: 0.8395 - loss: 0.3457
Test Accuracy: 0.8395
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step
```

Figure 4. The training process output of the model.

3.2.5. Evaluate the Model

After training a model, assessing its performance on unseen data is essential to determine how well it generalizes. We present the ROC curve, the area under the curve (AUC), accuracy, recall, F score, and the confusion matrix to evaluate the model's effectiveness.

The ROC Curve and the Area Under the Curve (AUC)

This study examines the receiver operating characteristic (ROC) curve, a two-dimensional classification performance metric. The focus is on the scalar measure, the area under the ROC curve (AUC), which evaluates a specific performance aspect [28].

The paper concludes that the AUC effectively distinguishes the ten IOT device categories; it obtains an output of micro-average ROC score = 0.9896. The ROC AUC score of 0.9896 indicates that the classifier performs excellently distinguishing between different classes. This means that the classifier can correctly classify many positive and negative examples across all classes.

A micro-average ROC AUC score of 0.9896 signifies a highly effective and discriminative model across various classes. However, a more in-depth analysis is recommended for a nuanced interpretation and insights into specific class performance.

The following figure shows the plot of the ROC curves for a multi-class IoT device classification problem.

The plot illustrated in Figure 5 provides a comprehensive visualization of the model's performance in a multi-IoT devices classification setting using ROC curves. It allows for assessing how well the model distinguishes between classes, with the micro-average offering an overall measure. The diversity of colors aids in distinguishing between individual class curves.

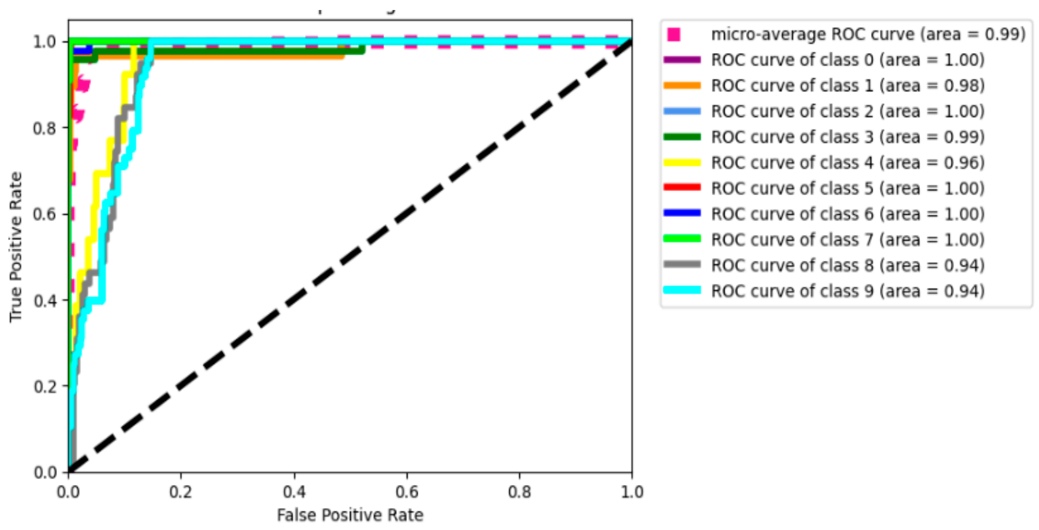


Figure 5. Plot of the micro-average ROC curves.

To further substantiate its performance, we calculated additional metrics: accuracy (0.8395), precision (0.8503), recall (0.8117), and F1-score (0.7938). These results are shown in Table 2 and Figure 6. They affirm the classifier's robustness in achieving high accuracy, precision, recall, and a strong F1-score.

Table 2. IoT device classifier evaluation metrics.

Accuracy	Precision	Recall	F1-Score
0.8395	0.8503	0.8117	0.7938

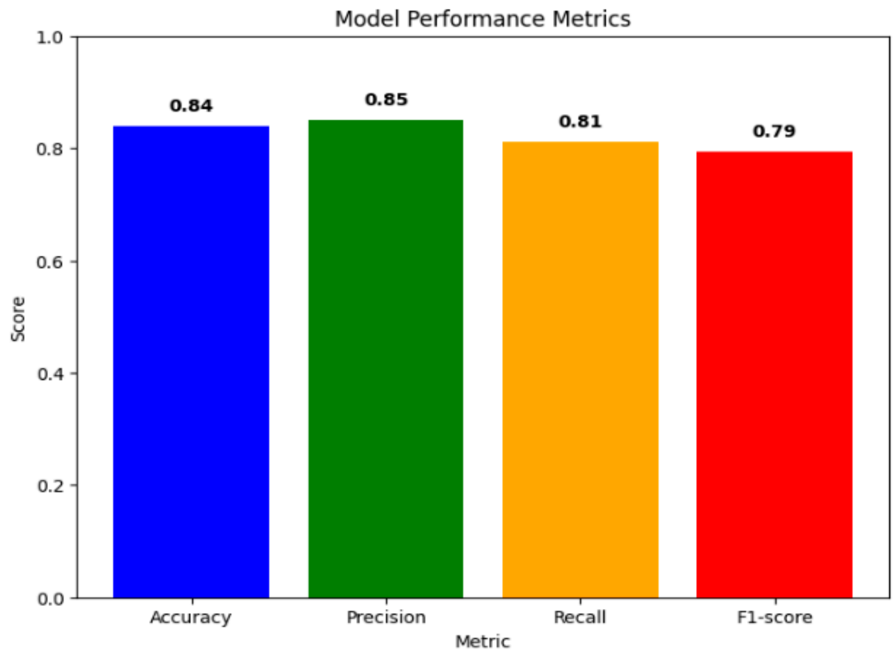


Figure 6. Model performance metrics.

The Confusion Matrix

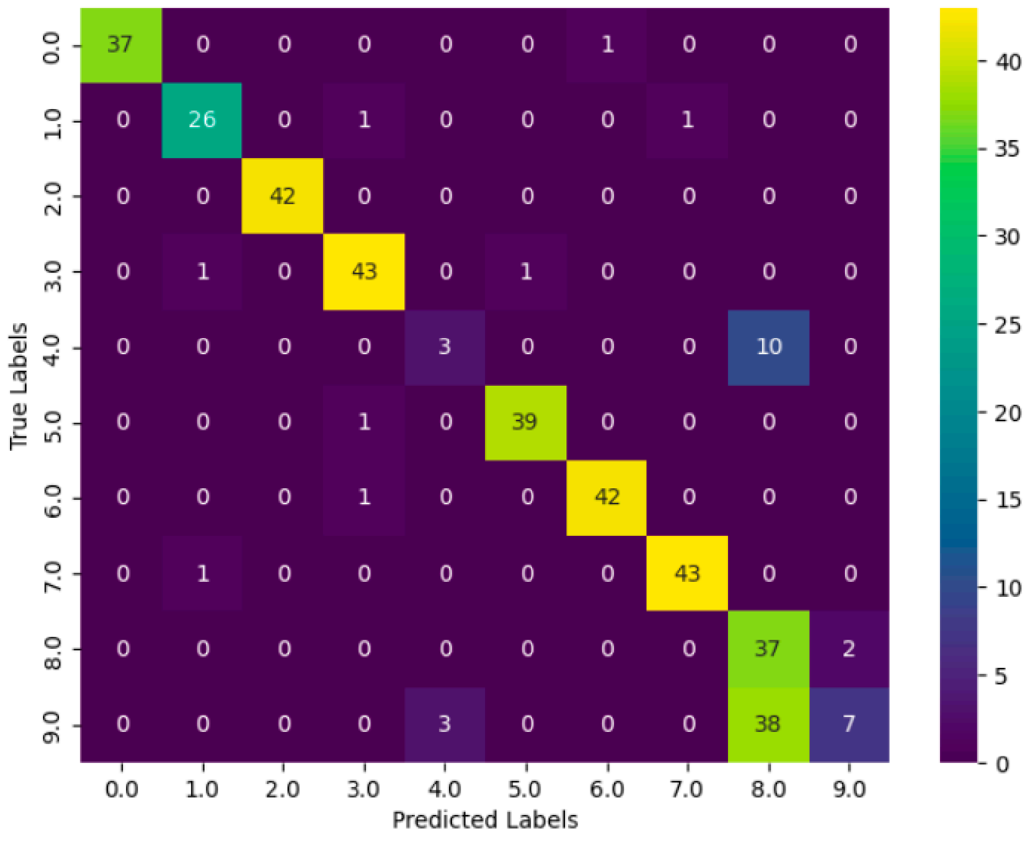
The confusion matrix is a crucial tool for evaluating the performance of classification models, especially in the context of multi-class classification problems. It provides a detailed overview of how well a model distinguishes between different classes by presenting a matrix format that summarizes true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions for each class. For multi-class classification, each matrix row represents the instances in an actual class, while each column represents the instances in a predicted class [29].

Using a confusion matrix allows researchers and practitioners to gain insights into a model's specific strengths and weaknesses. It highlights areas where the model is performing well and making errors, which can be particularly important for classes that are underrepresented or imbalanced in the dataset. Metrics such as precision, recall, F1-score, and overall accuracy can be derived from the confusion matrix, offering a comprehensive view of the model's performance for each class. For example, precision provides the ratio of correctly predicted positive observations to the total predicted positives. At the same time, recall measures the ability of the model to find all relevant instances of a class.

In multi-class scenarios, the confusion matrix helps to detect misclassification between different classes, which is essential for fine-tuning models, such as neural networks or support vector machines, in applications like image recognition, text classification, and medical diagnosis. Through its detailed error analysis, the confusion matrix aids in identifying where a model might require further training or adjustment, ensuring more balanced and accurate predictions across all classes [30].

These references provide insights into the confusion matrix's application for multi-class classification and its role in model evaluation and error analysis. Figure 7 displays the confusion matrix for a multi-class IoT device classification problem, where an RNN with an LSTM model was used.

The confusion matrix in Figure 7 shows that the model is highly accurate, as most of the predictions are located on the diagonal. For example, the model correctly predicted 37 samples as security cameras (class 0). It incorrectly predicted one sample as a baby monitor (class 6), which was actually a security camera. Looking at another example, the model correctly predicted 37 samples as lights (class 8). It incorrectly predicted two samples as sockets (class 9), which were actually sockets.



True \ Predicted	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
0.0	37	0	0	0	0	0	1	0	0	0
1.0	0	26	0	1	0	0	0	1	0	0
2.0	0	0	42	0	0	0	0	0	0	0
3.0	0	1	0	43	0	1	0	0	0	0
4.0	0	0	0	0	3	0	0	0	10	0
5.0	0	0	0	1	0	39	0	0	0	0
6.0	0	0	0	1	0	0	42	0	0	0
7.0	0	1	0	0	0	0	0	43	0	0
8.0	0	0	0	0	0	0	0	0	37	2
9.0	0	0	0	0	3	0	0	0	38	7

Figure 7. The confusion matrix.

All the string values in the "device_category" array, shown in Table 1, are mapped to corresponding numerical representations. This transformation prepares the data for use in the RNN model, as represented below:

- security_camera → 0.0
- TV → 1.0
- smoke_detector → 2.0
- thermostat → 3.0
- water_sensor → 4.0
- watch → 5.0
- baby_monitor → 6.0
- motion_sensor → 7.0
- lights → 8.0
- socket → 9.0

4. Results Analysis

This section critically evaluates the proposed model's methodologies, architectures, and results, offering insights into its strengths, weaknesses, and overall efficacy in real-world applications. The proposed approach presents a comprehensive methodology for developing and assessing a deep learning model specifically for IoT device classification. It adopts a systematic process that is crucial for ensuring the reproducibility and reliability of the research findings.

A significant focus of the analysis is on data handling, addressing potential discrepancies between training and test sets to ensure the robustness of the model. This emphasis on thorough data preprocessing and validation techniques aims to mitigate biases and prevent overfitting, thus enhancing the model's reliability. Furthermore, the model's architecture is centered around an LSTM-based recurrent neural network (RNN). The study details the architectural choices and training parameters, highlighting how LSTM networks are particularly well-suited for sequence data. This indicates that the model is designed to handle the temporal aspects inherent in IoT device data.

The training process achieves high accuracy on the training set, suggesting that the model effectively learns from the data. However, challenges in generalizing to new, unseen data underscore the need for robust evaluation metrics beyond training accuracy. The evaluation of the test set includes metrics such as ROC curves and AUC scores, with an AUC score of 0.9896 demonstrating exceptional performance in distinguishing between different IoT device categories. A confusion matrix is also provided, offering a deeper analysis of the model's performance.

The model demonstrates robust performance, with notable precision, recall, and F1-score values of 0.8503, 0.8117, and 0.7938, respectively. These strong metrics underscore the model's overall proficiency in accurately classifying IoT devices.

The marginally superior precision relative to recall implies that the model may be exercising prudence in minimizing false positives, which could lead to overlooking a limited number of true positives. Importantly, the F1-score, as an equilibrium metric, showcases the model's capacity to strike an acceptable balance between mitigating false positives and capturing true positives.

The research contributes a detailed guide for developing and evaluating deep learning models in the context of IoT device classification. This has important implications for a range of IoT applications, including smart homes, industrial automation, and healthcare monitoring, highlighting the potential impact of the proposed methodology across various domains.

5. Conclusions

This research presents a comprehensive framework for developing and evaluating deep learning models specifically designed for IoT device classification. By addressing potential biases in the training data and employing a well-suited LSTM-based RNN architecture, the proposed model demonstrates strong performance in distinguishing between various IoT device categories.

The model achieved an accuracy of 0.8395, precision of 0.8503, and recall of 0.8117, indicating its effectiveness in accurately classifying IoT devices. These metrics, combined with the high AUC score of 0.9896, demonstrate the model's ability to achieve a balance between avoiding false positives and capturing true positives.

The study's findings highlight the potential of deep learning techniques in advancing IoT device classification. The proposed framework, with its emphasis on data handling, model architecture, and rigorous evaluation, offers a valuable resource for researchers and practitioners working in this field.

As the IoT landscape continues to evolve, the development of accurate and reliable classification models will be essential for unlocking the full potential of IoT applications. This research offers a valuable contribution to the field of IoT device classification. By employing a robust LSTM-based RNN, the model demonstrates exceptional performance, implying significant potential for more efficient and reliable IoT applications.

Author Contributions: Conceptualization, S.A.B. and J.I.A.; methodology, S.A.B.; software, S.A.B.; validation, S.A.B., J.I.A. and G.S.M.K.; formal analysis, S.A.B., G.S.M.K. and J.I.A.; investigation, S.A.B.; resources, G.S.M.K.; data curation, S.A.B.; writing—original draft preparation, S.A.B.; writing—review and editing, G.S.M.K.; visualization, Gamal Saad Mohamed and S.A.B.; supervision, J.I.A.; project administration, G.S.M.K.; funding acquisition, G.S.M.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by to the Deanship of Scientific Research at Northern Border University, Arar, grant number "NBU-FFR-2024-2267-02" and The APC was funded Deanship of Scientific Research.

Data Availability Statement: The original data presented in the study are openly available in Kaggle at https://www.kaggle.com/datasets/ayamoheddine/iot-device-identification/data, accessed on 6 October 2024.

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number "NBU-FFR-2024-2267-02".

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rosero, J.; Gonzalez, G.; Williams, J.M.; Liu, H.; Khanna, R.; Pisharody, G. Internet of Things: A Scientometric Review. *Symmetry* **2017**, *9*, 301. [CrossRef]
2. Nimodiya, A.; Ajankar, S. A Review on Internet of Things. *Int. J. Adv. Res. Sci. Commun. Technol.* **2022**, *113*, 135–144. [CrossRef]
3. Landaluce, H.; Arjona, L.; Perallos, A.; Falcone, F.; Angulo, I.; Muralter, F. A Review of IoT Sensing Applications and Challenges Using RFID and Wireless Sensor Networks. *Sensors* **2020**, *20*, 2495. [CrossRef] [PubMed]
4. Tarricone, L.; Grosinger, J. Augmented RFID Technologies for the Internet of Things and Beyond. *Sensors* **2020**, *20*, 987. [CrossRef]
5. Dhiviya, S.; Malathy, S.; Rajesh Kumar, D. Internet of Things (IoT) Elements, Trends and Applications. *J. Comput. Theor. Nanosci.* **2018**, *15*, 1639–1643. [CrossRef]
6. Alshaya, S.A. IoT Device Identification and Cybersecurity: Advancements, Challenges, and an LSTM-MLP Solution. *Eng. Technol. Appl. Sci. Res.* **2023**, *13*, 11992–12000. [CrossRef]
7. Kumar, S.; Tiwari, P.; Zymbler, M. Internet of Things is a revolutionary approach for future technology enhancement: A review. *J. Big Data* **2019**, *6*, 111. [CrossRef]
8. Alsulami, R.; Alqarni, B.; Alshomrani, R.; Mashat, F.; Gazdar, T. IoT Protocol-Enabled IDS based on Machine Learning. *Eng. Technol. Appl. Sci. Res.* **2023**, *13*, 12373–12380. [CrossRef]
9. Singla, K.; Bose, J. IoT2Vec: Identification of Similar IoT Devices via Activity Footprints. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018. [CrossRef]
10. Yin, F.; Yang, L.; Ma, J.; Zhou, Y.; Wang, Y.; Dai, J. Identifying IoT Devices Based on Spatial and Temporal Features from Network Traffic. *Secur. Commun. Netw.* **2021**, *2021*, 27131211. [CrossRef]
11. Mainuddin, M.; Duan, Z.; Dong, Y.; Salman, S.; Taami, T. IoT Device Identification Based on Network Traffic Characteristics. In Proceedings of the IEEE Global Communications Conference, Rio de Janeiro, Brazil, 4–8 December 2022. [CrossRef]
12. Zahid, H.; Saleem, Y.; Hayat, F.; Khan, F.; Alroobaea, R.; Almansour, F.; Ahmad, M.; Ali, I. A Framework for Identification and Classification of IoT Devices for Security Analysis in Heterogeneous Network. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 8806184. [CrossRef]
13. Qiang, Y.; Kumar, S.; Brocanelli, M.; Zhu, D. Tiny RNN Model with Certified Robustness for Text Classification. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022. [CrossRef]
14. Kotak, J.; Elovici, Y. IoT Device Identification Using Deep Learning. *arXiv* **2020**, arXiv:2002.11686. [CrossRef]
15. Thouti, S.; Venu, N.; Rinku, D.R.; Arora, A.; Rajeswaran, N. Investigation to identify the multiple issues in IoT devices using Convolutional Neural Networks. *Meas. Sens. J.* **2022**, *24*, 100509. [CrossRef]
16. Mathew, A.; Amudha, P.; Sivakumari, S. Deep Learning Techniques: An Overview. In Proceedings of the International Conference on Advance Machine Learning Technologies and Application, Singapore, 20–22 March 2021. [CrossRef]
17. Sandhya, N.; Charanjeet, K.R. A review on Machine Learning Techniques. *Int. J. Recent Innov. Trends Comput. Commun.* **2016**, *4*, 451–458.
18. Zantalis, F.; Koulouras, G.; Karabetsos, S.; Kandris, D. A Review of Machine Learning and IoT in Smart Transportation. *Future Internet* **2019**, *11*, 94. [CrossRef]
19. Tan, M.; Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2020**, arXiv:1905.11946v5.
20. Shetty, D.; Harshavardhan, C.A.; Varma, M.; Navi, S.; Ahmed, M. Diving Deep into Deep Learning: History, Evolution, Types and Applications. *Int. J. Innov. Technol. Explor. Eng.* **2020**, *9*, 2835–2846. [CrossRef]
21. Kotsiantis, S.B.; Zaharakis, I.; Pintelas, P. Supervised Machine Learning: A Review of Classification Techniques. *Emerg. Artif. Intell. Appl. Comput. Eng.* **2007**, *160*, 3–24.
22. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *arXiv* **2014**, arXiv:1406.2661. [CrossRef]
23. Karras, T.; Laine, S.; Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv* **2019**, arXiv:1812.04948.
24. Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2007; pp. 153–160.
25. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805. [CrossRef]

26. Lee, H.; Kim, J. Understanding Model Performance through Confusion Matrix in Multi-Class Classification. *IEEE Trans. Artif. Intell.* **2021**, *10*, 987–995. [CrossRef]
27. Marzban, C. The ROC Curve and the Area under It as Performance Measures. *Weather. Forecast.* **2004**, *19*, 1106–1114. [CrossRef]
28. Alenazi, M.; Mishra, S. Cyberatttack Detection and Classification in IIoT systems using XGBoost and Gaussian Naïve Bayes: A Comparative Study. *Eng. Technol. Appl. Sci. Res.* **2024**, *14*, 15074–15082. [CrossRef]
29. Hussein, L.; Mohan, M.; Rajeswari, P.; Gurupandi, D.; Saranya, N.N. Anomaly Detection in IoT Data Streams Based on Long Short-Term Memory. In Proceedings of the 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 23–24 August 2024; pp. 1–4. [CrossRef]
30. Yahya, B.; Roy, S.; Hajra, S. A Comprehensive Survey on Machine Learning for Activity Recognition in Smart Homes. *arXiv* **2023**, arXiv:2308.07724.