Article

Not peer-reviewed version

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

Sana Abdelaziz Bkheet , Johnson I. Agbinya , Gamal Saad Mohamed Khamis *

Posted Date: 14 October 2024

doi: 10.20944/preprints202410.1085.v1

Keywords: The Internet of Things (IoT); Smart objects; Recurrent Neural Network (RNN); Long Short-Term Memory (LSTM)

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] Faculty of Science, Department of Computer Science, Northern Border University (NBU), Arar 73213, Saudi Arabia

[2] Melbourne Institute of Technology, Australia, School of Information Technology and Engineering, Melbourne, Australia

* Correspondence: Gamal.Khamees@nbu.edu.sa or jamalziena@gmail.com

Abstract: This study presents the development of a Recurrent Neural Network (RNN) model for classifying smart home device data. Utilizing a dataset sourced from Kaggle, the study explored the processes of data gathering, loading, normalization, and model construction. The RNN, equipped with Long Short-Term Memory (LSTM), was trained and evaluated, demonstrating significant improvements in training and validation accuracy over 10 epochs, culminating in a test accuracy of 83.25%, and only 35.4% loss. The evaluation of the model on the test set provides a test accuracy result, accompanied by an in-depth analysis of ROC curves and Area Under the Curve (AUC) scores for multi-class classification, along with a confusion matrix. The AUC score of 0.9896 indicates outstanding performance in accurately classifying IoT device categories. These findings indicate that the RNN with LSTM exhibits superior learning efficiency and generalization capabilities, making it more suitable for IoT device classification tasks. This article emphasizes the concept of IoT and reviews recent studies on the application of deep learning models across various IoT domains, including smart homes, industrial systems, and healthcare. Future research could focus on improving real-time processing capabilities, and scalability, and incorporating diverse IoT data types to enhance and broaden the model's practical applications.

Keywords: The Internet of Things (IoT); Smart objects; Recurrent Neural Network (RNN); Long Short-Term Memory (LSTM)

1. Background

The Internet of Things (IoT) connects billions of Physical devices or individuals to the Internet and has become a highly popular 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 both wireless and wired sensor networks. The main characteristics of IoT include:

- Perception: Sensors, Radio Frequency Identification (RFID), and barcodes are used to collect data about any object, anywhere. This also encompasses place and object identification and recognition of things.
- Transmission: With the availability of networking and communication technologies, data can be accessed at any time. IoT ensures reliable transmission between machine-to-machine and mobile-to-machine connections.
- Processing: Cloud computing facilitates the intelligent processing of IoT data. Service providers process millions or billions of data points using cloud computing. [3]

The 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].By utilizing smart devices and the internet, IoT provides innovative solutions to various challenges faced by

 doi:10.20944/preprints202410.1085.v1

businesses, governments, and industries worldwide. IoT is increasingly becoming an integral part of our lives, evident in its widespread presence. Essentially, IoT integrates a vast array of smart systems, frameworks, intelligent devices, and sensors [4][5].

IoT technology has become essential in our lives, it cover a wide range of 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 influencing critical sectors like healthcare, smart water management, surveillance, biomedical applications, industrial processes, data center management, agriculture, body area networks (BANs), and more [1–3][6].

Due to the rapid growth in this field and the increasing number of diverse objects connected to various aspects of it daily, this paper focuses primarily on the identification of 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 apply the same model to other smart environments. These environments play a significant role in IoT applications, as they directly impact human safety, comfort, welfare, and security.

The rest of the paper is organized as follows. Section II, presents the previous work related to the use of deep learning algorithms in the identification of smart objects. Section III, discusses in details the steps and procedures of the development process of the RNN model for classifying of smart home objects. Section IV, reports an analysis of outcomes of th proposed model. Section V, provides the conclusion of the paper and possible recommendation for 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 thefunctionality and intelligence of smart systems. This section will briefly discuss some of the research efforts on IoT device identification is most relevant to this work.

The paper [7], explores the concept of 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 various applications such as identifying similar devices, determining replacements, and building a location classifier based on IoT devices.

The authors review related work in the area of 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 similar 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 activity generated by smart IoT devices to gain higher-level understanding of user's tasks. In conclusion, the paper presents a method to generate 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

 doi:10.20944/preprints202410.1085.v1

location classifiers based on IoT devices. The authors also outline several further questions for analysis and plan to explore more use cases and higher-level understandings of user tasks in future research [7].

The authors in [8], present a novel IoT device identification method called CBBI, which uses a hybrid neural network model Conv-BiLSTM to automatically learn spatial and temporal features from network traffic. The study aims to address the security risks associated with the increasing number of IoT devices connected to networks and the need for accurate identification of these devices. The proposed approach overcomes the limitations of traditional methods that rely on manually extracted features and prior knowledge, which increases the difficulty and reduces 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 an increase in potential attacks. It emphasizes the importance of accurate identification of 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 the elimination of the need for prior knowledge in feature engineering, extraction of spatial and temporal features, and the use of 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 accurately identify IoT devices. [8]

Authors in [106], 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 takes into account the imbalanced nature of network traffic generated by various devices in both the learning and evaluation phases. The performance of iotID was evaluated on network traffic collected in a typical smart home environment with both IoT and non-IoT devices, achieving a balanced accuracy score of above 99%. Future work will explore evaluating iotID with additional IoT devices and studying deployment scenarios where one may be preferred over another [106].

To validate the efficacy of iotID, authors conducted performance studies utilizing network traffic data collected from a typical smart home environment, comprising a mix of IoT and non-IoT devices. 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. [106]

The research article [10] discusses the framework for the identification and classification of IoT devices for security analysis in a heterogeneous network. The study focuses on the challenges posed by the 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

 doi:10.20944/preprints202410.1085.v1

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 this, achieving 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 the field of IoT security and network management[10].

Some papers have highlighted the use of Deep Neural in classification tasks such as text classification, such as in [11] article, that 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 focuses on minimizing 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 both 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 effectiveness of the approach by comparing the tiny RNN models with compact and compressed RNN models in clean and adversarial test settings.

The paper introduction mentions the importance of mobile artificial intelligence (AI) in various domains and discusses the obstacles in deploying deep neural networks on mobile devices due to high computational and memory requirements. To address these challenges, the paper focuses on designing tiny RNN models for text classification tasks, 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 introduces 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 in comparison 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 key 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 the model compression performance, selecting the smallest 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 [11].

 doi:10.20944/preprints202410.1085.v1

The research [12] 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 is applicable 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 due to their less secure nature. To address this, organizations often implement security policies allowing only white-listed IoT devices. To monitor adherence to these policies, organizations must identify connected IoT devices, particularly unknown ones. A study applied 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 [12].

The Internet of Things (IoT) allows physical objects to communicate, but it also 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 [13] 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 [13].

3. The Data and Method

This part presents the development process of the RNN model for classifying IoT device data. The key steps involved in constructing and assessing the proposed model are depicted in Figure 1. The diagram provides a visual representation of the main procedural stages, offering a highlight of the model development and evaluation process.

Figure 1. Procedural stages of the proposed model.

3.1. Data Preparing and Loading

This paper has uses available datasets on Kaggle.com. This dataset consists of smart home network comprising various IoT devices network traffic analysis data generated by other researchers.

The data used comes in form of two separate CSV files one of them is the test file and the other is train file as shown in Table I and II. To explore the content of each file, both files are loaded into Google Colab. Google Colab is a document that allows you to write, run, and share Python code within your browser. It is a version of the popular Jupyter Notebook within the Google suite of tools. Jupyter Notebooks (and therefore Google Colab) allow user to create a document containing executable code along with text, images, HTML, LaTeX, etc. which is then stored in your Google Drive and shareable to peers and colleagues for editing, commenting, and viewing.

The dataset contains 900 rows × 298 columns; rows represent the total number of instants, while columns represent attributes.

Table II. The array string of "device_category" in train:.

Title of device in array format	Total number
array(['baby_monitor', 'lights', 'motion_sensor', 'security_camera',	10

 doi:10.20944/preprints202410.1085.v1


```
'smoke_detector', 'socket', 'thermostat', 'TV', 'watch',
        'water_sensor'], dtype=object)
```

Table I and II, shows an array of strings, containing names of different smart home devices. The array is one-dimensional, and has a total of 9 elements in the test file and total of 10 elements in the train file. Each element is a string representing the name of a device. So according to arrays shown in the above tables the element device 'water_sensor' is not included in iot_device_testnew.csv.

The only way to fix this problem is to combine the two sets in one new set and resplit them for the sake of the classification task; this way guarantees the existence of all ten elements in both sets after split.

3.2. Split the Data Set

There are at least three methods employed for computing the accuracy of a classifier. One approach involves splitting the training set. In another method called cross-validation, the training set is partitioned 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 [14][15].

In this model the new dataset that created after combine the two main sets have been splits into training and test sets. In this case, 20% of the data will be used for testing and the remind data for training

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

The model defines the architecture of the RNN model 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's an explanation of each steps of the model:

- Initializes a sequential model, which is a linear stack of layers. This implies that each layer in the model has exactly one input tensor and one output tensor.
- Use 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. It consists of 200 units (or neurons) with the option return_sequences=True, which is suitable when stacking multiple LSTM layers. The input_shape parameter specifies the shape of the input data, where X_train.shape[1] represents the number of time steps, and X_train.shape[2] represents the number of features.
- Introduces a Dropout layer with a dropout rate of 0.2. Dropout is a regularization technique that randomly sets a fraction of input units to zero during training, which helps prevent overfitting.
- Adds a second LSTM layer with 200 units. This layer doesn't need return_sequences=True since it is the last LSTM layer in the stack.
- Includes another Dropout layer with a dropout rate of 0.2 for regularization.
- Adds a Dense (fully connected) layer with a number of units equal to the number of classes in the output. The activation function is set to 'softmax,' which is common for multi-class classification problems.
- Configures 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.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 October 2024 doi:10.20944/preprints202410.1085.v1

In summary, 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 structure of the RNN structure with 200 units in LSTM layers.

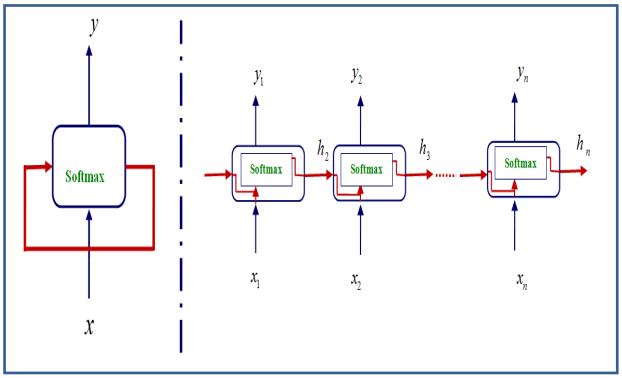

Figure 2. Illustration of applying Softmax activation function.

$$y_1 = soft\max\left(w_x x_1 + w_h h_0\right) \quad (1\_a)$$

$$y_2 = soft\max\left(w_x x_2 + w_h h_2\right) \quad (1\_b)$$

-
-
-

$$y_n = soft\max\left(w_x x_n + w_h h_{n-1}\right) \quad (1\_n)$$

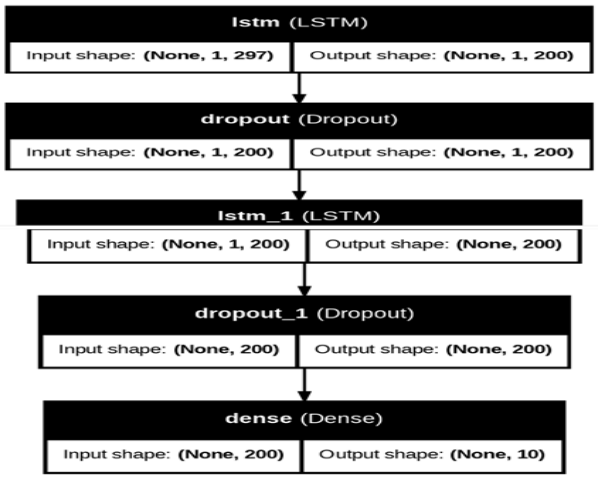

Figure 3. Illustration of the RNN structure with 200 units in LSTM layers.

3.4. Train the Model:

The previously defined LSTM model trained using the fit method. Here's a detail of each parameter:

- X_train and y_train: Training data, where X_train is the input features, and y_train is the corresponding target labels.
- epochs=10: The number of epochs specifies how many times the entire training dataset is passed forward and backward through the neural network. In this case, the model will undergo 10 epochs of training.

 doi:10.20944/preprints202410.1085.v1

- batch_size=32: The batch size determines 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 used in each iteration.
- validation_split=0.2: This parameter designates 20% of the training data as the validation set. The model's performance on this subset will be monitored during training, providing insights into its generalization capabilities.
- verbose=2: This controls the verbosity of the training process during each epoch. A value of 2 means that training progress will be displayed per epoch, showing a progress bar for each epoch.

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 Table III.

Table III. The training process output of the main model.

Epoch 1/10 38/38 - 5s - 135ms/step - accuracy: 0.4885 - loss: 1.7056 - val_accuracy: 0.5888 - val_loss: 1.1190
Epoch 2/10 38/38 - 1s - 14ms/step - accuracy: 0.6776 - loss: 0.8666 - val_accuracy: 0.7171 - val_loss: 0.7538
Epoch 3/10 38/38 - 1s - 22ms/step - accuracy: 0.7673 - loss: 0.5748 - val_accuracy: 0.7566 - val_loss: 0.5595
Epoch 4/10 38/38 - 1s - 37ms/step - accuracy: 0.8141 - loss: 0.4504 - val_accuracy: 0.7961 - val_loss: 0.4872
Epoch 5/10 38/38 - 1s - 37ms/step - accuracy: 0.8215 - loss: 0.3937 - val_accuracy: 0.7895 - val_loss: 0.4293
Epoch 6/10 38/38 - 1s - 29ms/step - accuracy: 0.8380 - loss: 0.3656 - val_accuracy: 0.8191 - val_loss: 0.3961
Epoch 7/10 38/38 - 1s - 16ms/step - accuracy: 0.8372 - loss: 0.3508 - val_accuracy: 0.8289 - val_loss: 0.3778
Epoch 8/10 38/38 - 1s - 15ms/step - accuracy: 0.8512 - loss: 0.3222 - val_accuracy: 0.8322 - val_loss: 0.3635
Epoch 9/10 38/38 - 1s - 17ms/step - accuracy: 0.8553 - loss: 0.3106 - val_accuracy: 0.8355 - val_loss: 0.3478
Epoch 10/10 38/38 - 1s - 15ms/step - accuracy: 0.8586 - loss: 0.2900 – val_accuracy: 0.8355 - val_loss: 0.3412
12/12 - 0s - 5ms/step - accuracy: 0.8316 - loss: 0.3540
Test Accuracy: 0.8316 12/12 [==============================] -1s 31ms/step

 doi:10.20944/preprints202410.1085.v1

3.5. Evaluate the Model

After training a model, it's essential to assess its performance on unseen data to determine how well it generalizes. To evaluate the model's effectiveness, we present the ROC curve, the area under the curve (AUC), and the confusion matrix.

3.5.1. The ROC Curve and the Area under Curve AUC

The Receiver Operating Characteristic (ROC) curve, a two-dimensional metric for classification performance, is examined in this study. The focus is on the scalar measure known as the area under the ROC curve (AUC), which evaluates a specific aspect of performance [16].

The paper concludes that AUC effectively distinguishes the ten IOT device categories, it gets an output of micro-average ROC score = 0.9896. The ROC AUC score of 0.9896 indicates that the classifier has excellent overall performance in distinguishing between different classes. This means that the classifier can correctly classify a high proportion of both positive and negative examples across all classes.

In summary, a micro-average ROC AUC score of 0.9896 signifies a highly effective and discriminative model across various classes, but 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 IoT devices classification problem.

The plot illustrated in Figure 4, provides a comprehensive visualization of the model's performance in a multi-IoT devices classification setting using ROC curves. It allows for the assessment of 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.

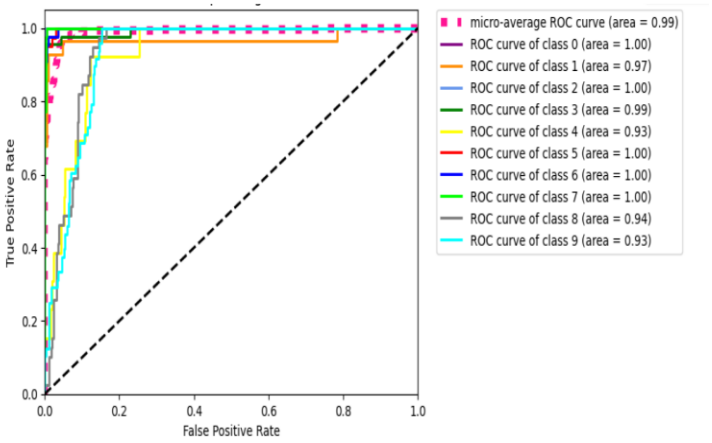

Figure 4. Plot the micro-average ROC curve.

3.5.2. The Confusion Matrix

A confusion matrix is a key tool in classification used to evaluate model performance. It provides a detailed breakdown of predictions by showing the counts of:

- True Positives (TP): Correctly predicted positive cases.
- True Negatives (TN): Correctly predicted negative cases.
- False Positives (FP): Incorrectly predicted positives (Type I error).
- False Negatives (FN): Incorrectly predicted negatives (Type II error). [17][15]

By visualizing these results, it helps identify class confusions and calculate important metrics such as accuracy, precision, recall, and F1-score, offering insights into the model's effectiveness.

The figure below displays the confusion matrix for a multi-IoT device classification problem, where an RNN with an LSTM model was used.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 October 2024 doi:10.20944/preprints202410.1085.v1

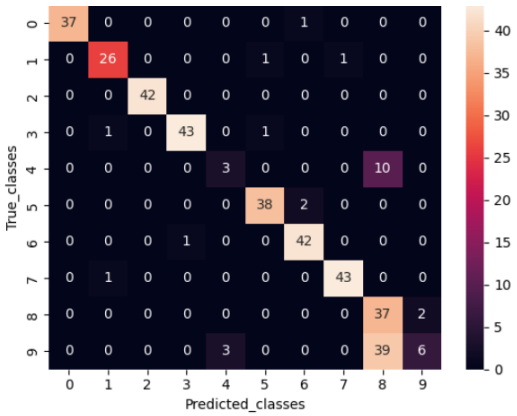

Figure 5. The confusion matrix**.**

The confusion matrix in the above figure shows that the model has high accuracy, 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 1 sample as a baby monitor (class 6) which was actually a security camera. Looking at another example, the model correctly predicted 38 samples as watches (class 5). It incorrectly predicted 2 samples as baby monitors (class 6), which were watches.

4. Results Analysis

This section aims to assess their methodologies, architectures, and results, this analysis seeks to provide insights into the relative strengths and weaknesses of the model and provide insights into its efficacy and reliability in real-world applications. The proposed method outlines a comprehensive methodology for developing and evaluating a deep learning model for IoT device classification. Here's a breakdown and analysis of the key components:

1. Methodology for Model Development and Evaluation:

The research presents a step-by-step approach for developing and evaluating a deep learning model specifically designed for IoT device classification. This suggests a structured and systematic process, which is crucial for reproducibility and reliability in research.

2. Data Handling Process:

Addressing discrepancies between training and test sets is crucial for ensuring the robustness of the model. This indicates a thorough consideration of data preprocessing and validation techniques to mitigate potential biases or overfitting issues.

3. Model Architecture:

The use of an LSTM-based RNN model is explained in detail, highlighting architectural choices and training parameters. LSTM (Long Short-Term Memory) networks are well-suited for sequence data, indicating that the model is tailored to handle temporal aspects inherent in IoT device data.

4. Training Process and Performance:

The high accuracy achieved on the training set suggests that the model effectively learns from the training data. However, the challenges in generalizing to unseen data underscore the importance of robust evaluation metrics beyond training accuracy.

5. Model Evaluation:

Evaluation on the test set includes metrics such as ROC curves and AUC scores for multi-class classification. The AUC score of 0.9896 signifies outstanding performance in classifying various IoT device categories, indicating the model's effectiveness in distinguishing between different classes.

6. Contributions and Implications:

The research contributes a comprehensive guide for developing and evaluating deep learning models in IoT device classification scenarios. This suggests potential applications in various IoT domains, such as smart homes, industrial automation, and healthcare monitoring [15].

5. Conclusions

This paper presents a comprehensive methodology for the development and evaluation of a deep learning model tailored for IoT device classification. The data handling process addresses potential biases between training and test sets, ensuring a robust foundation for model training. The utilization of a Long Short-Term Memory (LSTM)-based Recurrent Neural Network (RNN) model is thoroughly explained, highlighting key architectural decisions and training parameters. During the training process, the model achieves high accuracy on the training set, indicating its ability to learn from the provided data. However, challenges arise in generalizing to unseen data, underscoring the importance of robust evaluation metrics. Model evaluation on the test set yields a test accuracy result and includes a detailed analysis of ROC curves and Area Under the Curve (AUC) scores for multi-class classification, as well as a confusion matrix. The obtained AUC score of 0.9896 signifies exceptional performance in classifying various IoT device categories.

The final results show that the proposed model emerged as the superior approach, offering stable learning without overfitting and demonstrating superior test accuracy. Its structured methodology and robust evaluation metrics position it as a reliable framework for IoT device classification tasks.

In summary, this article contributes a comprehensive guide for developing and evaluating deep learning models in IoT device classification scenarios. It underscores the importance of addressing data handling challenges, robust evaluation techniques, and further analysis for nuanced insights into model performance.

In conclusion, the Internet of Things (IoT) has transformed device connectivity, providing intelligent solutions across sectors like smart homes, healthcare, and industry. The application of deep learning models, particularly Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) architectures, has demonstrated strong effectiveness in classifying and differentiating IoT devices. This study lays the groundwork for further advancements in developing reliable IoT classification systems and driving innovation in IoT technologies.

References

1. Juan Ruiz-Rosero 1,* ID , Gustavo Ramirez-Gonzalez 1 ID , Jennifer M. Williams 2 ID ,Huaping Liu 2, Rahul Khanna 3 ID and Greeshma Pisharody .(2017).Internet of Things: A Scientometric Review. Symmetry,MDPI. doi:10.3390/sym9120301.
2. Aditi Rajesh Nimodiya1 and Shruti Sunil Ajankar. (2022). A Review on Internet of Things. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT). DOI: 10.48175/IJARSCT-2251.
3. S. Dhiviya, S. Malathy, D. Rajesh Kumar. (2018). Internet of Things (IoT) Elements, Trends and Applications. Journal of Computational and Theoretical Nanoscience. doi:10.1166/jctn.2018.7354.
4. Alshaya, S.A. (2023). IoT Device Identification and Cybersecurity: Advancements, Challenges, and an LSTM-MLP Solution. Engineering, Technology & Applied Science Research. 13, 6 (Dec. 2023), 11992–12000. DOI:https://doi.org/10.48084/etasr.6295.
5. Sachin Kumar1* , Prayag Tiwari2 and Mikhail Zymbler.(2019). Internet of Things is a revolutionary approach for future technology enhancement: a review.Joournal of big Data. https://doi.org/10.1186/s40537-019-0268-2.
6. Alsulami, R., Alqarni , B., Alshomrani, R., Mashat, F. and Gazdar, T. (2023). IoT Protocol-Enabled IDS based on Machine Learning. Engineering, Technology & Applied Science Research. 13, 6 (Dec. 2023), 12373–12380. DOI:https://doi.org/10.48084/etasr.6421.
7. Singla K, Bose J. (2018). IoT2Vec: Identification of Similar IoT Devices via Activity Footprints. International Conference on Advances in Computing, Communications and Informatics (ICACCI). DOI:10.1109/ICACCI.2018.8554398.
8. Yin F, Yang L, Ma J, Zhou Y, Wang Y, Dai J. (2021). Identifying IoT Devices Based on Spatial and Temporal Features from Network Traffic. Hindawi. Article ID 2713211. DOI: https://doi.org/10.1155/2021/2713211.
9. Mainuddin M, Duan Z, Dong Y, Salman S, Taami T. (2022). IoT Device Identification Based on Network Traffic Characteristics. IEEE Global Communications Conference. DOI:10.1109/GLOBECOM48099.2022.10001639
10. Zahid H, Saleem Y, Hayat F, Khan F, Alroobaea R, Almansour F, Ahmad M, Ali I. (2022). A Framework for Identification and Classification of IoT Devices for Security Analysis in Heterogeneous Network. Hindawi, Wireless Communications and Mobile Computing. DOI: https://doi.org/10.1155/2022/8806184.

11. Qiang Y, Kumar S, Marco Brocanelli, Zhu D. (2022). Tiny RNN Model with Certified Robustness for Text Classification. International Joint Conference on Neural Networks (IJCNN).**DOI:**10.1109/IJCNN55064.2022.9892117.
12. Jaidip Kotak and Yuval Elovici. (2020). IoT Device Identification Using Deep Learning. arXiv:2002.11686.DOI:https://doi.org/10.48550/arXiv.2002.11686.
13. Swapna Thouti a, Nookala Venu b, Dhruva R. Rinku a, Amit Arora a, N. Rajeswaran. (2022). Investigation on identify the multiple issues in IoT devices using Convolutional Neural Network. ELSEVIER, Measurement Sensors Journal. DOI:https://doi.org/10.1016/j.measen.2022.10050.
14. B. Kotsiantis.(2007). Supervised Machine Learning: A Review of Classification Techniques. informatica. DOI: 31:249-268.
15. Part of the cited section is written with aid of either ChatGPT or Google Colab.
16. Marzban C. (2004).The ROC Curve and the Area under It as Performance Measures. American Meteorological Society (AMS). DOI: https://doi.org/10.1175/825.1.
17. Alenazi, M. and Mishra, S. (2024). Cyberatttack Detection and Classification in IIoT systems using XGBoost and Gaussian Naïve Bayes: A Comparative Study. Engineering, Technology & Applied Science Research. 14, 4 (Aug. 2024), 15074–15082. DOI:https://doi.org/10.48084/etasr.7664.