IoT Device Category Classification System using Advanced Deep Learning Techniques

Charvi Jain
MEng, ECE
University of Waterloo, Canada
c5jain@uwaterloo.ca

Jay Panara
MEng, ECE
University of Waterloo, Canada
jpanara@uwaterloo.ca

Yash Patel
MEng, ECE
University of Waterloo, Canada
yk4patel@uwaterloo.ca

***Abstract*—The Internet of Things (IoT), which allows household appliances, cameras, and other devices to communicate and interact with one another, has brought about a dramatic transformation in today's technological world. Due to the transmission of sensitive data between the devices, the security and protection of this data is a problem that requires extreme caution to resolve. A fundamental worry in today's IoT security is the need for a mechanism or a method to distinguish between types of IoT devices in a network traffic. So, in this research, we aim at developing a novel framework for classifying different types of IoT devices in a network using advanced deep learning techniques. The dataset utilised to train the neural network model was the "IoT device identification" dataset [1]. Additionally, a comparative study between the proposed neural network-based model with other state-of-the-art classical machine learning models is also presented. The proposed pipeline has been shown to surpass state-of-the-art methods in recognising the kind of device on various evaluation metrics such as accuracy, precision, computation time and misclassification rate. Further, comparison of model performance based on features extraction techniques and optimizer used for the neural network model is also presented in this research.**

***Index Terms*—IoT Device, Neural Networks, Data Preprocessing, Deep Learning, Logistic Regression, Decision Tree, Random Forest Classifier, XGBoost classifier**

I. Introduction

In today's cutting-edge technological world, the Internet of Things (IoT) has proven essential in linking objects, from signalized intersections to consumer electronics. These gadgets, also known as Internet of Things (IoT) devices, feature a number of sensors that offer pertinent data. IoT devices may occasionally use actuators to improve their functionality while interacting with one another over a variety of protocols, including Bluetooth, ZigBee, Wi-Fi and Ethernet [2].

A diverse network of IoT devices commonly offers services to society. Gathering and analysing data as well as autonomously responding to environmental changes are both made possible by the connectivity between these devices. The network created by these devices sends data continually and at incredibly rapid rates when they share large volumes of data on occasion. Administrators face operational challenges as a result of the growth of IoT devices in IoT networks. The heterogeneous nature of IoT networks presents extra asset management difficulties given that numerous departments deploy IoT devices in big cities. For instance, the sanitary authority may place various wastewater and sewage sensors, and the local police force may install cameras. Since the malfunctioning devices are invisible, finding them on the network is very challenging [3].

There is no immediate personal contact on the IoT, which sets it apart from the conventional internet. The anonymity, confidentiality, and safety of user data are sacrificed when an IoT device collects information in response to environmental changes, analyses it, and takes autonomous action [4]. According to researchers, there may be security concerns as a result of the increase of unsecured gadgets linked to the Internet. To create a trustworthy security solution for IoT networks, manufacturers and developers have been working hard. Another issue is that, due to limited resources, manufacturers of IoT devices do not automatically update the software on their products unless consumers specifically request it. The use of fully featured security mechanisms is not possible on these devices [5]. Since they often have unpatched bugs and default login credentials, IoT devices are more vulnerable to attacks.

There is a considerable risk that someone may attempt to physically access IoT devices because they frequently operate in an unattended setting. Since IoT devices typically employ wireless connections, trespassers may be able to gain sensitive details over a transmission channel by secretly eavesdropping to the dialogue. Since there aren't enough readily available computer and power resources, many devices don't have appropriate security safeguards. Strong security measures cannot be put in place due to a lack of resources and unexpected environmental effects. Due to the likelihood of susceptible IoT devices, an IoT network requires a strong security solution based on frequently patching the vulnerabilities.

Many businesses are already enabling IoT device connectivity, but this might jeopardise the security of their networks. Recognizing the plugged-in gadgets to their networks is necessary for businesses. They have to establish a method for assessing whether the gadgets linked to their systems are safe that do not represent a risk.

Several concepts of gadget recognition generally and for differentiating between authorised and unauthorised devices have utilised evaluation of real-time network data [6]. Since the data flow patterns of different classes of IoT devices differ significantly from one another, it might be argued that network traffic traces have been shown to be able to distinguish between them. When a trigger happens, an Internet of Things (IoT) device could function. For instance, object-detection sensors only function when a human is moving in front of them.

Even for the same kinds of IoT devices, like Withings Smart Baby Monitor and Drop Camera, that are both webcams from different suppliers , it may be difficult to categorise a device's network information into something like a recognisable pattern and construct an invariant profile. However, there is a certain structure to the information which those two cameras produce. It is crucial to find a trend that can classify the devices in their respective groups, even if they provide the same kind of information, for a stronger categorization of the devices participating in a common environment.

Organization unique identifier (OUI), the prefix of a MAC address, makes it simple to determine a device's maker, however there are several manufacturers of various types of devices, therefore this information is insufficient to deduce the functioning of the device. Since manufacturers of IoT devices may purchase NICs from a third party, it is difficult to identify IoT devices only based on the MAC address's prefix. The possibility of faked MAC addresses for IoT devices is another issue.

Many scientists and academics have recently argued in favour of the automatic recognition and characterization of IoT devices with various restrictions or legislation. The majority of these recommendations centred on discovering and categorising IoT devices that had already been recognised as components of the network under investigation. To far, there is no method for identifying and categorising new devices. For example, it could be challenging to recognise and categorise a new IP base camera that enters the network based on past IP base camera recordings.

IoT networks require a thorough security solution in light of the aforementioned rationale. For this, each connected and inbound gadget must be identified and categorised. The difficulty of categorising IoT devices in the communication environment is addressed in this study. We developed a neural network model to classify IoT devices into the categories (mentioned in section IV-A) using the IOT device identification dataset. An independently created neural network model and specific pre-processing techniques were used to create the pipeline. During the pre-processing phase, a regularization technique known as Min-Max Normalisation and feature extraction technique known as Principle Component Analysis (PCA) were employed. The research also provides comparisons of model performance based on PCA, optimizer, and with traditional classical ML algorithms.

II. Literature Review

Researchers are interested in creating more effective systems using Machine Learning and Deep Learning approaches to give society improved capabilities as a result of the current exponential growth of IoT devices. Analysis of internet traffic from 2005 to 2012 has taken a lot of time and effort from researchers. Online application identification, including peer-to-peer programs, mail, Skype, and web browsing, was the main focus of research during this period. Scholars began concentrating on using ML/DL methodologies in 2015 in order to provide embedded intelligence in IoT devices and address security issues in IoT networks using techniques like attack detection and anomaly detection. This section has covered numerous topics of closely related IoT research, including the classification of different IoT device categories using advanced Machine Learning and Deep Learning methods.

Combining CNN, RNN, and Deep Learning models, researcher Lopez-Martin in [7] developed a framework for a network traffic categorization tool. These characteristics were used to classify several types of traffic flows, including Hyper-Text Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), QUIC, Telnet, Office365 and YouTube. They were obtained from the first 20 traffic flow packets. The characteristics included the source port number, destination port number, payload size, TCP window size, inter arrival time, and traffic direction. It has been demonstrated that the suggested framework, which combines CNN and RNN, outperforms other competing algorithms in terms of detection without the need for feature extraction or selection, which is occasionally necessary when utilizing several machine learning models. It has been demonstrated that the suggested framework, which combines CNN and RNN, outperforms other competing algorithms in terms of detection without the need for feature extraction or feature selection, which is occasionally necessary when utilizing several ML models.

In [8], based on statistical parameters such port numbers, activity phases, signaling trends, and encryption packages, researchers have described the traffic of IoT Devices. The flow level metrics, such as the flow volume, flow length, flow rate, sleep time, DNS period, and NTP timeframe derived from network traffic, were presented as a multiphase classification model to classify the IoT devices. In this experiment, the authors classified 28 IoT device types and 01 Non-IoT device class using 50,378 labelled examples, with an accuracy rate of over 97%. The complicated multistage design of this study and the requirement that attributes be chosen by a subject-matter expert were limitations. Dependence on elements that providers may easily change, such domains and ports, could be unsafe and risky.

Later In 2014, Falk and Fries et al. [9] provided a number of authentication schemes as a means of device identification and whitelisting (list of authorized devices). These techniques were used to whitelist industrial automation control systems (IACS). The researchers discovered a known communication link between the devices employed in this domain and the IACS environment. As a consequence, the overall system complexity may be addressed. The authors emphasized the dynamic nature of large-scale business systems and the regular introduction of new device categories. As a result, these tactics may be ineffective in this situation.

Meidan et al. [10] used a classical machine learning approach known as Random Forest to extract characteristics from IoT devices network traffic data by utilizing the feature extraction methods described in [5] in order to identify an unauthorized device from several other devices based on a single TCP transaction in intelligent settings. Researchers collected network data from 27 unique IoT devices of 9 distinct

categories in order to train and evaluate a multiclass classifier for each device type. After then, the traffic data was manually labelled. The algorithm successfully identified the ninth illegal device type and correctly categorized the eight remaining device types as belonging to one of the approved (white list) device types. This multiclass classifier had roughly 300 packet and flow level properties. TTL, median and averaged packets, percentage of total bytes sent and received, number of total of packets with reloading tag settings (RST), and Alexa server rank are the most important aspects of these. The limits of this experiment were caused by the researchers' categorization of devices into various device types, despite the fact that there were multiple device types with just one item in each category. This type of generalization is impossible. The devices were linked, but not for intricate mixed real-time traffic, which was the experiment's second fault.

Pêgo and Nunes et al. [11] created a system to recognize the attributes of a new device and classify it. This tool produces the new device's interface as well as the required integration drivers automatically. The major goal of this paper was to identify the devices involved in a network using data supplied by IoT devices. Researchers discovered the efficiency of multiple ML techniques for device identification in the IoT smart environment. This advanced IoT device automation by resolving existing platform integration challenges for platforms that aggregate multiple IoT devices into an intelligent environment. The authors obtained communication information by listening to traffic in a smart environment. In an application developed for the iPhone operating system (iOS), the 'Levenshtein distance' algorithm, TF-IDF tables, synonyms match, and finally multi property matching were used to convert this network information (XML format) into a data set with data about all IoT devices installed in a smart building. This enabled the system to identify the device that communicated successfully in the IoT ecosystem.

The difficulties with IoT device security were discussed by Ferrando and Stacey et al. [12]. The authors proposed a security detection approach that might be applied with early-stage classified threats and data streams. This technology is a step toward the uniqueness of securing IoT devices since it can categorize sensor traffic and discover a wide variety of network anomalies. Because most anomalies in network traffic involve aspects related to data sharing, researchers evaluated the strategy as one for anomaly identification utilizing data provided by a network device. Observing the allocation of features in network activity, according to the theory, was just as acceptable as analyzing the dispersion of diagnostic power in terms of the identification and classification of primary types of anomalies.

Shen et al. [13] explained how several supervised algorithms for machine learning might be used to evaluate data acquired by monitoring the smart environment communication and correctly detect unauthorised IoT connected devices in order to protect an organization's private data. Researchers gathered and manually classified network traffic data from 27 IoT devices of nine distinct categories, using this dataset to train and assess a multiclass classifier. They verified that it correctly classifies the ninth kind as unknown and that the other eight belonged to approved devices.

Suárez and Salcedo et al. [14] used the information gathered from twelve different devices, including cameras, lights, sensors, and refrigerators, to apply several categorization approaches, including K-means and ID3. They used 4 different device classes and utilized ML algorithms with the help of similar properties of these devices, as well as 12 features gathered from IoT device network data such as battery capacity, amount of memory, internet bandwidth required, gateway, Bluetooth enabled, and so on. K-mean classified the devices into four categories: mobile orchestrators, fixed orchestrators, fixed followers, and fake followers. K-means were examined on three, four, and five clusters, respectively.

A technique for classifying the different IoT device types that are linked automatically in an IoT smart environment was presented by Miettinen et al. [15]. The authors used fingerprint classification to enforce network rules and constraints, preventing damage from happening as an outcome of unauthorized network access. The proposed system employs network traffic filtering rules to protect devices interacting in the network grid from threats originating from other very harmful devices on the network. Researchers tried the anticipated technique for isolating traffic for IoT devices currently present in the network. This technique is impractical since so many IoT devices are developed each year. They have not used the technology to handle diverse traffic generated by non-IoT devices.

Cviti et al. [16]'s technique functioned as a conceptual network model for anomaly identification as well as a unique way for detecting distributed denial of service (D-DoS) traffic generated by IoT devices. This model was constructed utilizing device classes, and each class is fully dependent on the traffic generated by each device individually.

Several scholars have spent the last several years researching on IoT device identification based on MAC address and port information. When IoT devices use HTTP/HTTPS ports as channels of communication, identifying an Internet of Things device purely on port characteristics is extremely difficult. Nmap is a free and open-source application that can identify over 2600 different operating system versions. As a result, a dependable framework for detecting and categorizing IoT devices and related subcategories using traffic patterns created by diverse devices in heterogeneous networks is required.

We employed relatively efficient neural network models in this research for device classification and identification with very high efficiency and minimal run time.

III. Methodology

The goal of this project is to develop a distinctive framework that, via network traffic analysis, categorises different IoT device types. The framework of the pipeline comprises a self-developed neural network model that was trained on pre-processed data. Techniques for preprocessing data include checking for missing or null values, Feature Engineering, and

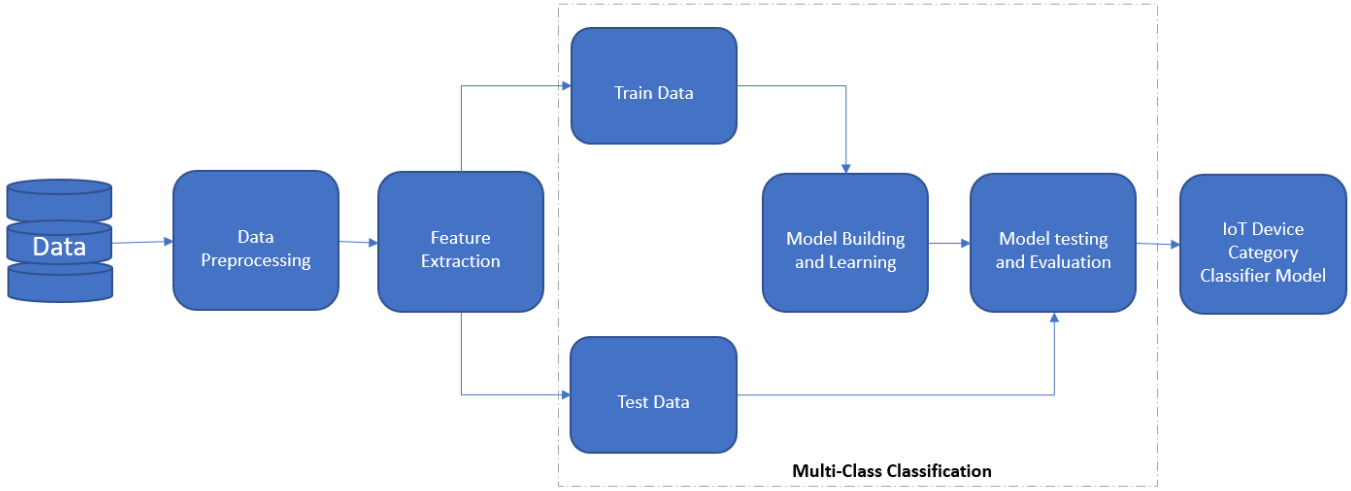

Fig. 1. Methodology

Min-Max Normalization. We have employed feature extraction using the Principal Component Analysis (PCA) approach for the aim of feature engineering and contrasted the performance of the models with and without feature extraction. We experimented with different numbers of hidden layers and neurons in neural network models before settling on the optimum one. Additionally, we used two distinct optimizers, SGD and Adam, and evaluated the performance of the model against each of them. Finally, we settled on the model and data preprocessing methods that performed the best for our framework.

Fig 1 presents the overall structure for our suggested technique. Additionally, we evaluated the performance of the self-developed neural network model in comparison to that of classical machine learning approaches such as the Random Forest classifier, Decision Tree Classifier, and XBG classifier, on several evaluation parameters including accuracy, computational time, and the number of misclassifications. Some of the methodologies and strategies employed in our framework are detailed below.

A. Machine Learning Classical methods:

- **Logistic Regression:** Logistic regression is a classification algorithm, used for prediction of an occurence of a particular event based on the probability. A logistic curve with logit function is used on which the data is fitted and model is trained. The model learns to predict a categorical output based on the features which may be discrete or continuous or a combination of both. The algorithm works by taking the linear combination of input values with weights or coefficients in order to make a prediction of an output value.

$$ln(y) = b_0 + b_1 * X$$

X = Input Variable, y = Output target, b_0 = Bias, b_1 = Weight.

The right side calculation is linear whereas the left side ratio is known as log-odds ratio of the default class ie. ratio of probability of the event over the probability of not of the event.

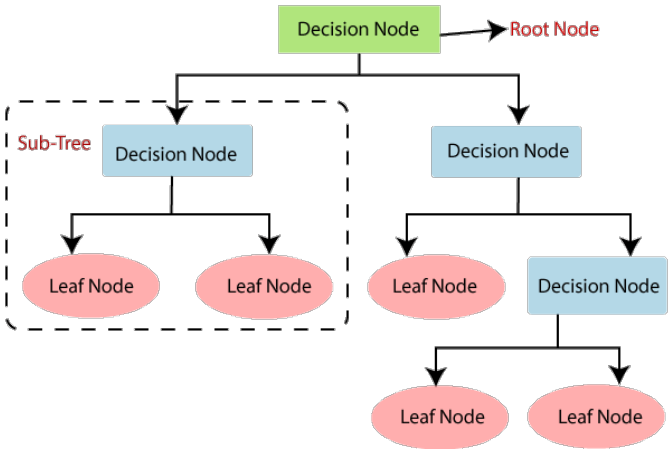

Fig. 2. Decision Tree [17]

- **Decision Tree Classifier:** A comprehensible classification approach that may be accurate in a variety of application situations, including energy-based applications, is offered by decision tree classifiers. By constructing a decision tree, the decision tree classifier develops the classification model. Fig. 2 depicts the decision tree. A test on an attribute is specified by each node in the tree, and each branch descending from that node represents one of the possible values for that property. Each leaf is a representation of the class labels connected to the instance. According to the results of the tests along the path, instances in the training set are categorised by moving them from the tree's root to a leaf. Beginning with the root node, each node divides the instance space into

two or more sub-spaces in accordance with an attribute test condition. The next step is to construct a new node by going down the tree branch that corresponds to the attribute's value. Following that, the same procedure is performed for the subtree rooted at the new node, and so on, until all records in the training set have been assigned a classification. By selecting an attribute test condition at each stage that optimally separates the data, the decision tree construction process typically operates top-down. To decide how to partition the records most effectively, a variety of metrics might be employed. There is a lot of exploitation of the Gini index impurity-based criterion for building the tree. It calculates the probability that a randomly selected instance from the set would be erroneously identified if it were randomly assigned a label based on how those labels are distributed in the subset.

- **Random Forest classifier:** Under the wide category of ensemble-based learning techniques, random forest classifiers are included. They have shown to be incredibly effective across a wide range of sectors and are easy to deploy and quick to use. The main idea behind the random forest technique entails building plenty of "simple" decision trees during the training phase, and then voting with the majority of them during the classification phase. This voting method has the advantage of correcting decision trees' bad tendency to overfit training data, among other things. Random forests use bagging as a general strategy to treat each tree in the ensemble individually during the training phase. A random sample with replacement is repeatedly chosen from the training set via bagging, and then trees are fitted to these samples. Without any pruning, each tree is grown. Using the so-called out-of-bag error, it is simple to automatically learn the ensemble's total number of trees [18].
- **XGB classifier:**XGBoost is a distributed gradient boosting library that has been developed to be very effective, adaptable, and portable. It uses the Gradient Boosting framework to construct machine learning algorithms. Many data science issues are quickly and accurately solved using a parallel tree boosting method offered by XGBoost. The same code operates on the most popular distributed environments (Hadoop, SGE, and MPI) and can tackle issues involving a much larger number of samples.

B. Machine Learning Neural Network Model

The fundamental building block of deep learning, neural networks are renowned for simulating the behaviour of the human brain while tackling challenging data-driven issues. The neurons that make up the neural network architecture replicate the behaviour of the brain. A Neural Network's main function is to convert input into an actionable output. To generate the required output, the input data is processed by many layers of neurons layered together. A neural network typically has an input layer, an output layer, and one or more hidden layers. A layer is made up of many neurons piled in a row, while a multi-layer neural network is made up of several layers that are stacked on top of one another as shown in the Fig. 3 [19].

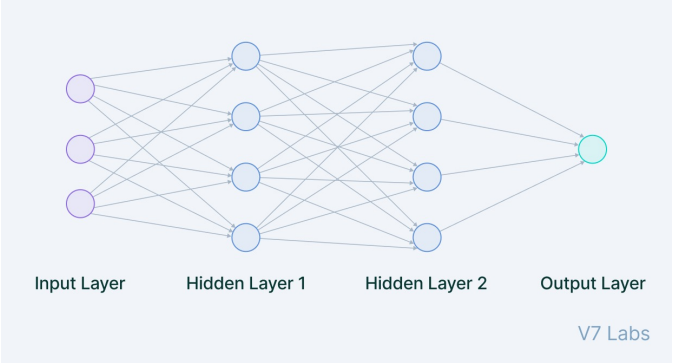

Fig. 3. Multi-layer Neural Network [20]

C. Machine Learning Optimizers

- **SGD Optimizer:** For each iteration of Stochastic Gradient Descent (SDG), a small number of samples are chosen at random rather than the whole data set. The total number of samples from a dataset used to calculate the gradient for each iteration is referred to as the "batch" in the Gradient Descent algorithm. The batch is assumed to be the whole dataset in conventional Gradient Descent optimization techniques like Batch Gradient Descent. Although considering the entire dataset is quite helpful for reaching the minima in a less noisy or random manner, the issue occurs when our datasets are very large.Stochastic Gradient Descent is used to tackle this issue. Each iteration in SGD only utilises a single sample. For the iteration, the sample is chosen and randomly shuffled. Since just one sample is randomly selected from the dataset for each iteration, the route traced by the method to the minima is often noisier than that of a standard Gradient Descent algorithm. However, this isn't really important since as long as we hit the minima and do so with a drastically reduced training time, it doesn't matter which path the algorithm takes [21].
- **Adam Optimizer:** The Adam Optimization Algorithm is a stochastic function optimization method based on first-order gradients. For any model with a lot of parameters and huge datasets, it is a suitable strategy that can be easily implemented. It is highly computationally efficient and uses fewer hardware resources, including less memory. Furthermore, it is suitable for non-stationary targets and situations with noisy and spare gradients. Any effective model must be tuned, while the adam optimization process often only needs a little tweaking. With the use of the Adaptive Moment Estimation (Adam) technique, adaptive learning rates for each parameter are kept distinct, and all changes to weights that don't change throughout training are maintained at a single learning rate. In order to balance the learning rate for

each weight in the model network, Adam calculates the first and second moments of the gradient, which is a mix of RMSprop and Stochastic Gradient Descent [22].

$$m_n = E[X^n]$$

m = Moment, X = Random Variable, n = Moment random variable is assigned.

IV. Experimental Results and Discussions

In this section of result and analysis, we have compared performance of our model based on whether feature extraction is used or not and the type of optimizer used. Further, we used six evaluation parameters to compare performance of our model to three classical models. All the simulations were carried out on Kaggle Notebooks using Python 3.

A. Dataset

Dataset used in this experiment is "IOT device identification" Dataset available at "Machine learning cookbook for cyber security" [1]. It is available as two separate files, namely, test data and train data. For our experiment, we have combined the two files to form a single data file and then split the entire data into test and train data with split ratio as 0.2. The dataset has 297 feature classes and 1 label class. The final label class has 7 IoT device categories which are: security camera, TV, smoke detector, thermostat, water sensor, watch, baby monitor, motion sensor, lights, and socket. The purpose of our deep learning-based framework is to correctly classify the device category when fed with new network data to the model. All the categories have same number of records except for water sensor which has only 50% data than other classes as shown in Fig 4. We have enhanced the dataset as compared to original dataset by checking for NULL/missing values and removing if any, label encoding, feature extraction using PCA and normalisation.

B. Evaluation Metrics

- **Accuracy:** It is metric that determines how the model performed on the data for which it's predictions were to be tested. The accuracy is the ratio of correctly classified examples according to the labels provided divided by the total examples or the total predictions made by the model.

$$\text{Accuracy} = \frac{\text{Number of correctly classified examples}}{\text{Total number of examples}}$$

- **Precision:** For quantifying the correct number of positive predictions, precision metric is used. This helps us in calculation of the accuracy of minority classes. Precision being the ratio of correctly predicted outputs over total number of outputs that belong to that class or category. For binary classification task it is defined as,

$$Precision = \frac{TP}{TP + FP}$$

TP = True Positive FP = False Positive

Similarly, it above formula of precision can be extended for multi-class classification as,

$$Precision = \frac{TP_1 + TP_2}{(TP_1 + TP_2) + (FP_1 + FP_2)}$$

TP_1 = True Positive for class 1, TP_2 = True Positive for class 2, FP_1 = False Positive for class 1, FP_2 = False Positive for class 2.

- **Recall:** The recall is a metric that can be used in quantification of the correct number of positive predictions divided by the total true positives and false negatives. It takes values between 0.0 and 1.0.

$$Recall = \frac{TP}{TP + FN}$$

In case of multi-class classification,

$$Recall = \frac{TP_1 + TP_2}{(TP_1 + TP_2) + (FN_1 + FN_2)}$$

TP_1 = True Positive for class 1, TP_2 = True Positive for class 2, FN_1 = False Negative for class 1, FN_2 = False Negative for class 2.

- **F-Score:** In order to get high precision and high recall which is generally desired, we use F - Score calculations given by,

$$F - Score = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

- **Computational Time:** It is the time taken for the machine learning model to be trained.
- **Number of Misclassifications:** Number of Misclassifications is a measure that determines Misclassification rate which is a machine learning measure that indicates the proportion of data that a classification model erroneously forecasted. It is determined by:

$$\text{Misclassification Rate} = \frac{\text{Number of Wrong Predictions}}{\text{Number of All Predictions}}$$

Or

$$\text{Misclassification Rate} = \frac{\text{False Positive + False Negative}}{\text{Total Prediction}}$$

Henceforth, number of misclassifications are the number of inaccurate predictions made by the classifier. It usually is used along with other evaluation parameters, such as accuracy, to analyse model performance since sometimes a model of high accuracy might still have high number of misclassifications in a particular class. This would suggest that the model is not performing well for records belonging to that particular class.

C. Data Preprocessing

- **Principal Component Analysis (PCA):** A multivariate approach called principal component analysis (PCA) examines a data table in which observations are defined by a number of quantitative dependent variables that are highly associated with one another. The key information from the statistical data will be extracted, represented as

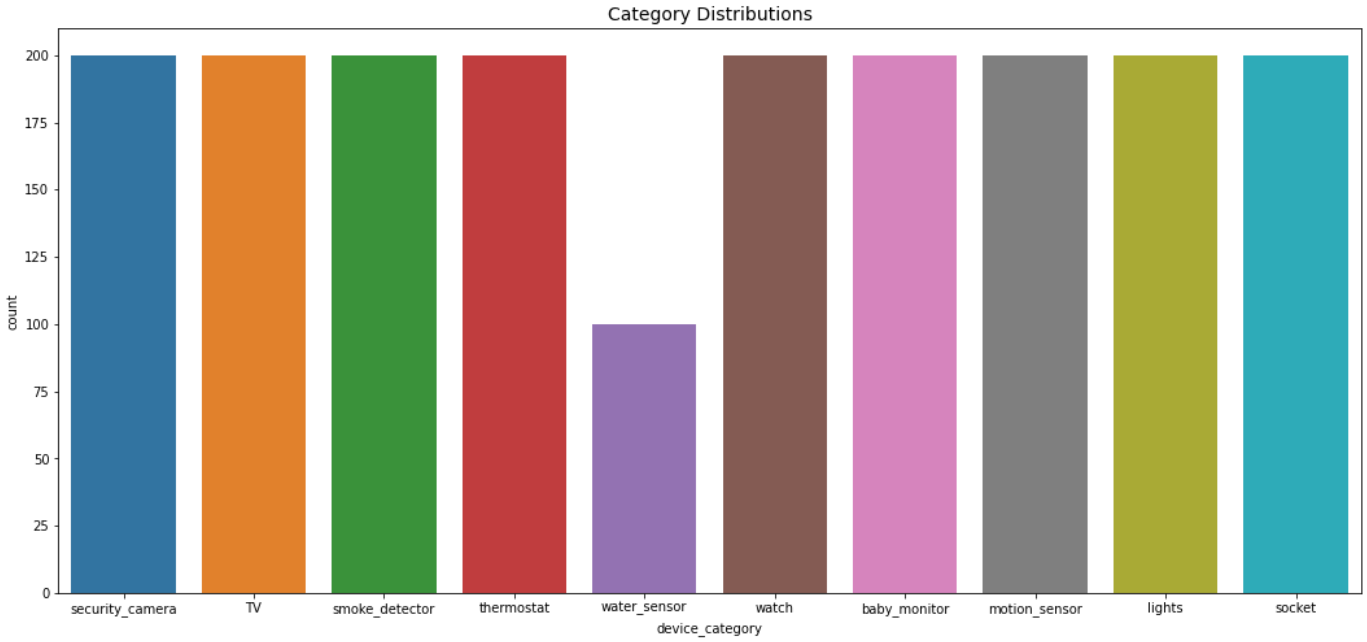

Fig. 4. Data Distribution among all IoT device categories

a collection of new orthogonal variables called principle components, and used to illustrate the pattern of similarities between the observations and the variables as spots on spot maps. Both the singular value decomposition (SVD) of rectangular matrices and the eigen-decomposition of positive semi-definite matrices are necessary for PCA mathematically. Eigenvalues and Eigenvectors are what determines it. Square matrices have eigenvalues and eigenvectors as related numbers and vectors. They work together to create the eigen-decomposition of a matrix, which examines its structure, including any covariance, correlation, or cross-product matrices [23].

- **Min-Max Normalisation:** When preparing data for machine learning, the scaling technique known as normalisation is used to convert the values of the dataset's numeric columns to a standard scale. For every dataset in a model, it is not required. It is only necessary when the ranges of the features in machine learning models differ. Despite the large number of feature normalisation methods available in machine learning, only a few of them are really utilised most frequently [24].One of it is Min-Max Normalization. Scaling is another name for this method. The dataset's attribute values are shifted and rescaled with the aid of the Min-Max scaling algorithm, resulting in a range of values between 0 and 1. The formula for the min-max normalization is as follows:

$$X_n = \frac{X - X_{minimum}}{X_{maximum} - X_{minimum}}$$

where X_n is value of normalization, $X_{maximum}$ is maximum value of a feature and $X_{minimum}$ is minimum value of a feature

D. ML Model Architecture

For the purpose of this study, we designed and trained multiple different Neural Network models on the available dataset for the task of IoT device category classification. With the help of hyper-parameter tuning, we experimentally found out the optimised neural network model which has the configuration as described in Table I and shown in Fig 5.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_15 (Dense)             (None, 512)               51712
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_16 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_17 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_18 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_19 (Dense)             (None, 10)                650
=================================================================
Total params: 224,842
Trainable params: 224,842
Non-trainable params: 0
```

Fig. 5. Final NN Model architecture

E. Results and Discussion

This section contains the findings from all the experiments performed for the purpose of achieving the most optimised ML framework and comparing it to classical ML methods. The experiment consists of studying the impact of feature extraction and optimizer used on the effectiveness of the neural network model designed and described in previous section.

TABLE I
NN MODEL CONFIGURATION

Hidden Layers	7 hidden layers (second and fifth are dropout layers (dropout rate=0.1) and rest all are dense layers)
Number of Neurons	Each of the dense layers have 512, 256, 128, 64, 10 neurons. Neurons in last layers equals total number of classes in output label
Activation Functions	used relu and softmax (only used in the last dense layer)

With the results of these two sections, the pipeline for our framework consisting of most desirable configurations will be finalised. The experiment further aims to compare our framework model with some classical ML algorithms which were trained under the same configuration. The following sections highlights the model performances:

- **Based on use of feature extraction (using PCA):** A machine learning model may get overfitted if the number of features is very high or exceeds the number of records included in a dataset. To prevent this type of issue, it is vital to use regularisation techniques or dimensionality reduction. Dimensionality reduction refers to the idea of an algorithm that minimise the set of features in a dataset by selecting the most significant ones or by producing new, less features that yet accurately describe the full dataset. We aim at achieving dimensionality reduction by the approach of Principle Component Analysis (PCA) in which new feature are created using the existing features and the new and fewer features created capture maximum variance which is enough for the model to perform well. Since there are 297 features in our dataset, we aim at experimentally determining whether PCA can produce a small set of meaningful features which captures maximum variance and help improve the performance by eliminating highly corelated features or not. Analysing the scree plot, it was observed that 100 features capture enough amount of variance of the original dataset. Hence, the experiment has been performed with 100 PCA extracted features, reducing dimensionality from 297 to 100.

TABLE II
NN MODEL WITH AND WOTHOUT PCA

Evaluation parameters	NN Model with PCA	NN Model without PCA
Training Accuracy	87.75%	86.69%
Testing Accuracy	87%	80.26%
Precision	0.90	0.82
Recall	0.86	0.80
F1 score	0.87	0.80
Computational Time	20.88 sec	17.6 sec
Misclassification rate	48/380 = 0.126	75/380 = 0.197

From Table II, we can observe that training and testing accuracies of the neural network model with PCA extracted features on the dataset are 87.75% and 75% respectively. Whereas training and testing accuracy of neural network model with all the features from the original data comes out to be 85.69% and 80.26% respectively. Time taken to train the model with PCA extracted feature is 20.88 seconds whereas for model with original features is 17.6 seconds. Other evaluation parameters such as precision, recall, F1 score, and misclassification rate can be found in TABLE II for both the models.

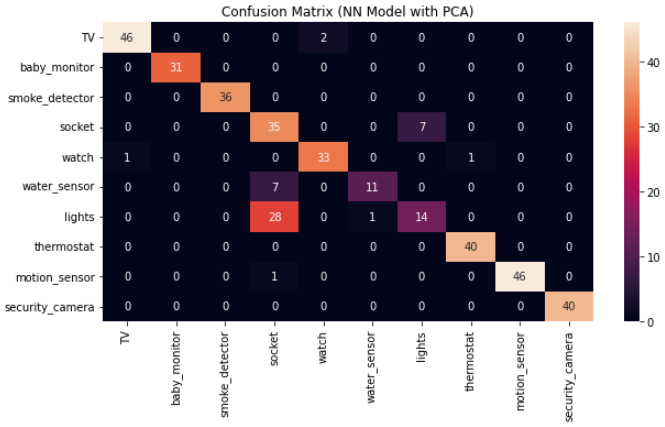

Fig. 6. Heatmap of misclassifications for Model with PCA

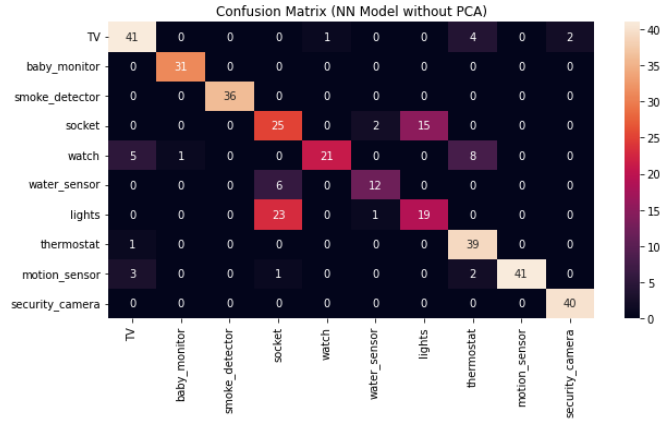

Fig. 7. Heatmap of misclassifications for Model without PCA

Fig.6 and Fig.7 shows a heatmap of the confusion matrix representing misclassifications in each IoT device category for Neural network models with and without PCA transformed features. According to the confusion matrix, the category "light" has the highest number of misclassifications with 23 records misclassified as "socket" in the NN model using the original dataset and 28 misclassified records in the model using features derived by PCA. "Socket" is the category with the second-highest misclassifications, with 17 misclassified records in the original dataset compared to 7 misclassified records in the reduced-dimensionality model.

- **Based on optimizer (SGD and Adam):** When creating a machine learning model, we require the most effective and efficient method possible that minimises errors and yields reliable results. A function or algorithm known as an optimizer alters the neural network's properties, such as its weights and learning rate, in order to effectively meet the objective of accurate outcomes while reducing losses. Therefore, the finest optimizer for our framework is required for this purpose. Therefore, we examined the parameters of two well-known optimizers, Stochastic Gradient Descent (SGD) and Adam, on our neural network model. SGD only works with a limited subset or randomly chosen sample of the dataset, avoiding unnecessary and expensive calculations on the entire dataset. When the learning rate is modest, SGD delivers performance comparable to that of conventional gradient descent. Whereas Adaptive Moment Estimation (Adam) is an algorithm which is really efficient when working with large problem involving a lot of data or parameters.

TABLE III
NN MODEL WITH SDG AND ADAM

Evaluation parameters	NN Model with SDG	NN Model with Adam
Training Accuracy	82.89%	87.99%
Testing Accuracy	85%	86.57%
Precision	0.86	0.88
Recall	0.85	0.86
F1 score	0.85	0.86
Computational Time	20.82 sec	20.89 sec
Misclassification rate	57/380 = 0.150	52/380 = 0.136

TABLE III shows that the neural network model with SDG has training and testing accuracies of 82.89 and 85 percent, respectively. While the neural network model using Adam has a training accuracy of 87.99 percent and a testing accuracy of 86.57 percent, respectively. Model training with SDG takes 20.82 seconds, whereas model training with Adam takes 20.89 seconds. Table II for both models includes additional assessment metrics including precision, recall, F1 score, and misclassification rate.
For neural network models trained with SGD and Adam optimizers, Fig.8 and Fig.9 show a heatmap of the confusion matrix reflecting misclassifications in each of the IoT device categories. When SGD optimizer is utilised, there are a total of 24 and 17 misclassifications in the "lights" and "socket" classes, respectively. When Adam optimizer is employed, the same classes have 13 and 21 misclassifications, respectively.

- **Based on models:** For comparison with classical methods, we build decision tree, random forest, XGBoost and Logistic regression classifier using parameters which are most optimised to the dataset used. This was achieved by hyperparameter tuning using GASearchCV from the sklearn_genetic [25] and GridSearchCV from sklearn.model_selection [26]. The following hyper-parameters were optimised using this technique for each of the following classifiers:

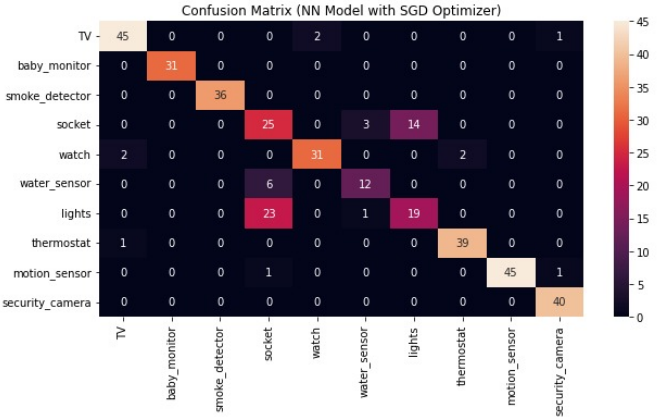

Fig. 8. Heatmap of misclassifications for Model with SGD

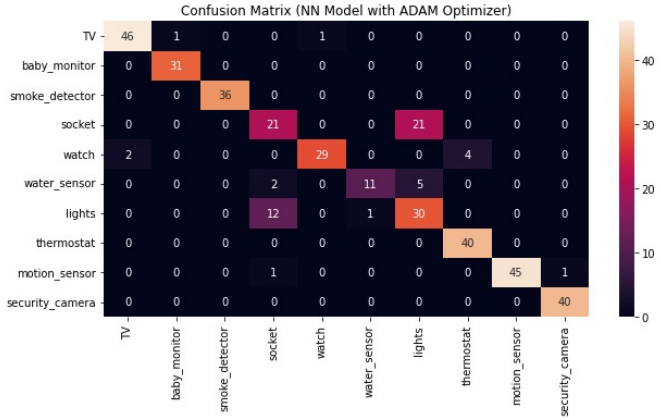

Fig. 9. Heatmap of misclassifications for Model with Adam

Table IV shows the comparisions of four Machine Learning models (Logistic Regressor, Random Forest Classifier, XGBoost Classifier and Decision Tree Classifier) and a Neural Network model models based on the Training Accuracy, Test Accurarcy, Precision, Recall, F1 Score, Computational Time taken by the Models and the Total Misclassification rate. The Misclassification rate is calculated by dividing the total number of misclassified examples with the total number of examples.
We found out experimentally that the Neural Network that we created outperformed all the best hypertuned machine learning models with the accuracy of 87.4% on the test set. Our observation shows that the simple machine learning algorithms like Logistic Regression performed poor compared to different models in prediction of the IoT devices with the accuracy of 79% on the test set. The XGBoost classifier performed best among all the different machine learning models with the test accuracy of 85%. The Decision Tree Classifier and Random Forest Classifier performed almost similar with 83% and 84% accuracies on the test set respectively. The F1 Score for

Classical Methods	Hyper-parameters Tuned
Random Forest	'ccp_alpha': 0.0, 'criterion': 'gini', 'max_depth': 30, 'max_features': 'log2', 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 30
Decision Treee	'criterion': 'entropy', 'max_depth': 20, 'max_leaf_nodes': 100, 'min_samples_leaf': 3, 'min_samples_split': 2
XGBoost	'learning_rate' : 0.1, 'n_estimators' : 1000, 'objective' : 'binary:logistic', 'n_thread' : 4, 'scale_pos_weight':1, 'subsample' : 0.8
Logistic Regression	'penalty': 'l2', 'solver': 'newton-cg', 'tol': 0.01

the Neural Network model is highest 0.87 which suggests that the model is having high precision as well as recall such models are capable of classifying the categories with low bias and low variance. The F1 Scores for Logistic Regressor, Random Forest Classifier, Decision Tree Classifier and XGBoost Classifier gradually increase as per the model complexity ranging in between 0.76 to 0.82. The computation time for the Random Forest Classifier turned out to be high as we performed hyperparameter tuning on it using Randomized Search CV which is 513 seconds. Whereas, the lowest Computational time is taken by Logistic Regressor, 3.68 seconds. The Neural Network models was efficient in terms of computational time, it took almost 20 seconds to learn the mappings from feature space to the target prediction. Lastly, the misclassification rate for the Neural Network was least 0.118 among all the other models. Hence, the Neural Network model is capable of predicting the IoT device categories efficiently.

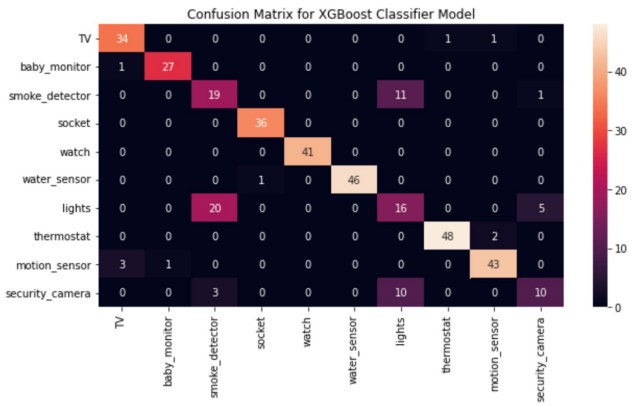

Fig. 10. Confusion Matrix for XGBoost Classifier Model

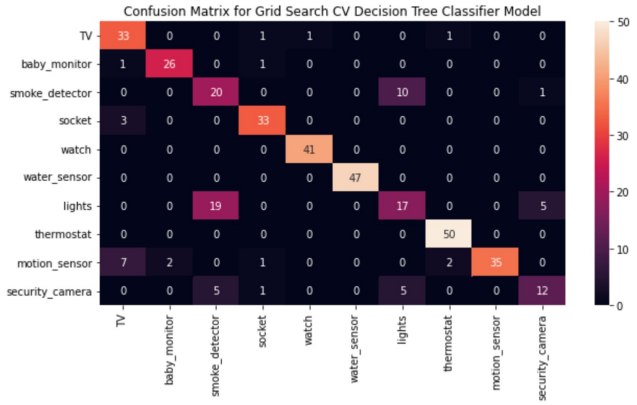

Fig. 11. Confusion Matrix for Decision Tree Classifier Model

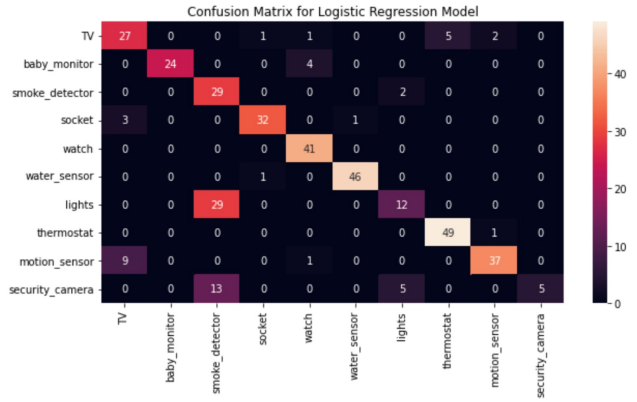

Fig. 12. Confusion Matrix for Logistic Regression Model

The heatmaps are shown in Fig.10, Fig.11, Fig.12, Fig.13, and 14 of the models utilizing XGBoost, Decision Trees, Random Forest, Logistic Regression, and Neural Networks. It can be seen that all the models were confused when predicting "lights" and predicted them as "smoke detectors" instead and vice versa. Also, the models get confused in predicting the "security cameras" and "smoke detector" classifying them as "lights". The Neural Network performed the best among all the models in classifying those devices with very less misclassification between those devices. Hence, the parameters learned by our Neural Network model are efficient in predicting the IoT device category from the stream of data.

F. Comparative Analysis

- **Comparisons Based on Use of Feature Extraction (using PCA):** In order to find the best pre-processed data for our ML Model, we conducted two separate experiments. The first experiment utilized the original data as it is for training, whereas the second experiment employed PCA transformed data. The assessment parameters that were provided were used to compare the two experiments. TABLE II shows that for the IoT Device Category Classification task, PCA transformed data has shown to be more

TABLE IV
NN VS CLASSICAL METHODS

Evaluation parameters	NN Model	Decision Tree Classifier	XGBoost Classifier	Random Forest Classifier	Logistic Regression
Training Accuracy	87.4%	95.07%	96.38%	88.68%	80.72%
Testing Accuracy	87.4%	82.63%	84.21%	83.95%	79.47%
Precision	0.90	0.81	0.83	0.83	0.84
Recall	0.86	0.81	0.82	0.82	0.77
F1 Score	0.87	0.81	0.82	0.82	0.76
Computational Time	20 sec	157.41 sec	5.34 sec	513 sec	3.68 sec
Misclassification Rate	45/380 = 0.118	66/380 = 0.174	60/380 = 0.158	61/380 =0.161	78/380 =0.205

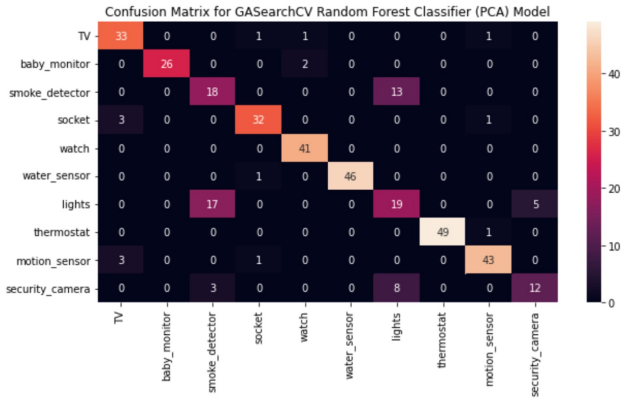

Fig. 13. Confusion Matrix for Random Forest Classifier Model

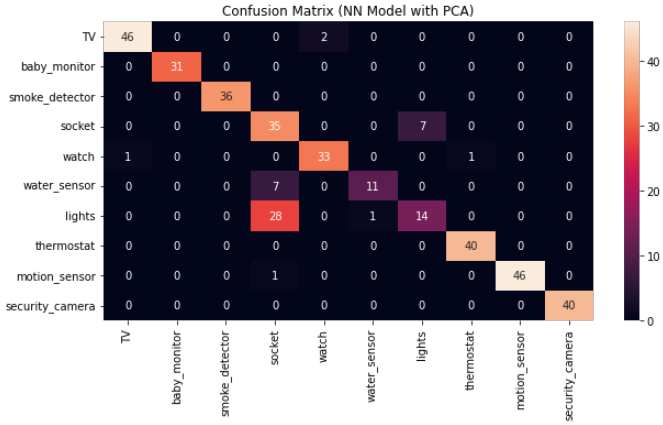

Fig. 14. Confusion Matrix of Proposed Model

useful, with training accuracy of 87.75 percent and testing accuracy of 87 percent. The results achieved using PCA transformed data are better than those obtained using the original data in terms of average Precision, Recall, and F1-Score for all labels. When the original dataset was employed, a slight loss in training accuracy was seen, however a significant decline in testing accuracy from 87 to 80.26 percent was seen. The main benefit of using PCA processed data is that it reduces the multi-collinearity that exists among the dataset's characteristics. Dimensionality reduction reduced the number of features in our high-feature dataset while retaining the majority of the variation included in the original dataset. By reducing this noise, the ML model is trained more effectively, leading to improved accuracy and a lower misclassification rate (0.126%). The PCA processed data aided in lowering the false positives count for Socket from 17 to 7 and for the Watch category from 14 to 2 alone, as seen under the Heatmap shown in Fig.6 and Fig.7.

- **Comparison Based on Optimizers (Adam/SGD):** The neural network model has been evaluated using Adam and SGD, two of the most popular optimizers. It is evident from the study results presented in TABLE III that Adam offered superior training and testing performance compared to SGD. Adam optimizer usage has resulted in a small increase in the average Precision, Recall, and F1 Score of all output labels. With a misclassification rate of 0.136 and testing accuracy of 86.57 percent, Adam also generated fewer misclassifications than SGD, whose misclassification rate was 0.15 percent and testing accuracy of 85 percent. While Adam employs coordinate wise gradient cutting, SGD uses a single global threshold method, making it difficult for it to handle data noise. Adam also employs Momentum and Adaptive learning rates in addition to fewer tuning parameters than SGD to achieve quicker coverage.
- **Comparisons Based on Type of Models (NN vs Classical Methods:** On the same assessment benchmarks, we compared our self-built neural network with the four current state-of-the-art classical classifier methods (Decision Tree, XGBoost, Random Forest, and Logistic Regression) in order to assess the performance of our neural network model. With a testing accuracy of 87.4%, results in TABLE IV clearly show that Neural Network Model has surpassed all other approaches. The maximum accuracy, recall, and F1-score for all categories were likewise achieved by the neural network model. Contrary to traditional approaches, neural network models are able to learn and represent the complicated and non-linear correlations between dataset characteristics, which aids in the development of a far more effective pattern for each category.

V. Conclusion and Future Work

In this research, we have successfully achieved the goal of developing a novel deep learning based framework that classifies the network traffic into one of the IoT device categories. The goal was to achieve a competitive accuracy by the Neural Network model as compared to other state-of-the-art classical approaches and as per the results based on the available performance metrics, our proposed neural network model has outperformed all other approaches with a competitive accuracy of 87.4%.

In addition, scaling the data and reducing dimensionality using PCA proved out to be helpful in preventing the model to become biased and overfit on the available data. This in turn helped train the model better and produced better accuracy. Further we focused on hyperparameter tuning of different classical machine learning models and finding the optimal set of parameters that allow for optimised performance.

The proposed strategy fulfills the requirements of the recently developed IoT environment, where the number of connected devices is rapidly increasing and it is not feasible to know the characteristics for each device (that would necessitate significant resources), but it is adequate to understand which class the device belongs to. This novel strategy may serve as a springboard for several more initiatives and research directions in the area of IoT challenges. One of the upcoming research paths that can make use of the data and insights obtained in this study is the identification of abnormalities in the network connection brought on by IoT devices and identification of particular devices, rather than just the category of the device. In addition, the established model may be used in the actual world as a software solution to enhance the capabilities of the current solutions for device and network monitoring in a setting where several different IoT devices are present.

References

[1] E. Sukerman, "Machine-Learning for Cybersecurity Cookbook," https://github.com/PacktPublishing/Machine-Learning-for-Cybersecurity-Cookbook/tree/master/Chapter05/IoT%20Device%20Type%20Identification%20Using%20Machine%20Learning, 2019, [Online; accessed 30-July-2022].

[2] H. Zahid, D. Y. Saleem, F. Hayat, F. Khan, R. Alroobaea, F. Almansour, M. Ahmad, and A. Ihsan, "A framework for identification and classification of iot devices for security analysis in heterogeneous network," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–16, 04 2022.

[3] H. H. G. A. R. C. W. A. V. Arunan Sivanathan, Daniel Sherratt and V. Sivaraman, "Characterizing and Classifying IoT Traffic in Smart Cities and Campuses," http://www2.eet.unsw.edu.au/ vijay/pubs/conf/17infocom.pdf.

[4] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, pp. 1250–1258, 2017.

[5] A. Alkhalil and R. A. Ramadan, "Iot data provenance implementation challenges," *Procedia Computer Science*, vol. 109, pp. 1134–1139, 2017, 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917311183

[6] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, "Automatic device classification from network traffic streams of internet of things," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018, pp. 1–9.

[7] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[8] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, pp. 1745–1759, 2019.

[9] R. Falk and S. Fries, "Managed certificate whitelisting – a basis for internet of things security in industrial automation applications," 11 2014.

[10] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici, "Detection of unauthorized iot devices using machine learning techniques," 2017. [Online]. Available: https://arxiv.org/abs/1709.04647

[11] P. R. J. Pêgo and L. Nunes, "Automatic discovery and classifications of iot devices," in *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, 2017, pp. 1–10.

[12] R. Ferrando and P. Stacey, "Classification of device behaviour in internet of things infrastructures: Towards distinguishing the abnormal from security threats," in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, ser. IML '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3109761.3109791

[13] J. Shen, Y. Li, B. Li, H. Chen, and J. Li, "Iot eye an efficient system for dynamic iot devices auto-discovery on organization level," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2017, pp. 294–299.

[14] J. N. Suárez and A. Salcedo, "Id3 and k-means based methodology for internet of things device classification," in *2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, 2017, pp. 129–133.

[15] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel demo: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2511–2514.

[16] I. Cvitić, D. Peraković, M. Periša, and M. Botica, "Novel approach for detection of iot generated ddos traffic," *Wirel. Netw.*, vol. 27, no. 3, p. 1573–1586, apr 2021. [Online]. Available: https://doi.org/10.1007/s11276-019-02043-1

[17] T. Penumudy, "Decision Trees for Dummies," https://medium.com/analytics-vidhya/decision-trees-for-dummies-a8e3c00c5e2e, 2021, [Online; accessed 31-July-2022].

[18] P. D. Caie, "Random Forest Classifier," https://www.sciencedirect.com/topics/computer-science/random-forest-classifier, 2021, [Online; accessed 31-July-2022].

[19] upGrad, "Neural Network: Architecture, Components Top Algorithms," https://www.upgrad.com/blog/neural-network-architecture-components-algorithms/, 2020, [Online; accessed 31-July-2022].

[20] P. Baheti, "The Essential Guide to Neural Network Architectures," https://www.v7labs.com/blog/neural-network-architectures-guideh7, 2022, [Online; accessed 31-July-2022].

[21] M. Glossary, "Optimizers," https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.htmlsgd.

[22] M. Alom, "Adam optimization algorithm," 06 2021.

[23] S. Mishra, U. Sarkar, S. Taraphder, S. Datta, D. Swain, R. Saikhom, S. Panda, and M. Laishram, "Principal component analysis," *International Journal of Livestock Research*, p. 1, 01 2017.

[24] Javapoint, "Normalization in Machine Learning," https://www.javatpoint.com/normalization-in-machine-learning.

[25] Sklearn, "GASearchCV," https://sklearn-genetic-opt.readthedocs.io/en/stable/api/gasearchcv.html.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.