National College of Ireland

Title

MSc Research Project
Programme Name

Raakesh babu Venkateswara
Student ID: X21105227

School of Computing
National College of Ireland

Supervisor: Mr. Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Raakesh babu Venkateswara
Student ID:	21105227
Programme:	M.Sc. Cybersecurity **Year:** 2021-2022
Module:	Research Project
Supervisor:	Mr. Niall Heffernan
Submission Due Date:	15/08/2022
Project Title:	Malware detection using Conventional Neural Network and Regression on smartwatches
Word Count:	…… **Page Count**……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Malware detection using Conventional Neural Network and Regression on smartwatches

Raakesh babu Venkateswara
X21105227

Abstract

Wireless wearable gadgets have become more widely known. Despite the fact that all these gadgets have minimal computing capability, the data they gather is sensitive. Wearable Internet of Things (WIoT) gadgets may function as both a stand-alone platform as well as an extension of smartphones and other mobile devices. Because the WIoT has computing, storage, and capacity restrictions, such gadgets can indeed be interconnected with an intrusion detection system. Such WIoT may pose a serious danger in the near future simply because they are open and easily available to both the customer and the intruder, which makes them more vulnerable. An intrusion into smart wearable gadgets might have disastrous consequences for a victim or a massive network. Wearable devices are largely neglected as a crucial element of a network. Nonetheless, this might have a negative impact on privacy and data security. This research will give an intrusion detection method for wearable smart IoT (WIoT) gadgets by evaluating the WSN-DS dataset by using a Convolutional Neural Network (CNN) and regression model that may be utilized to identify normal and unusual network activity with great accuracy and minimal complication. In addition to the Wireless Body Area Network (WBAN), the Wireless Sensor Network (WSN) represents the most intimately linked network to smart wearable IoT devices (WBAN). A compact model is built and compared with a single CNN model using a database developed on Wireless Sensor Network (WSN) to identify unusual cases of attack with limited processing capabilities that can be applied in tiny IoT devices or portable devices.CNN uses a regression function instead of a classifying method because it works better with output that changes over time. By finding outliers in the network, this model made the single CNN better and got a high accuracy of 97.96%.

1 Introduction

When it comes to the realm of information and communication, the Internet of Things (IoT) is a major player. Smart cities, intelligent buildings, transport, health care, eco-design, etc., are just some of the many areas where IoT is finding widespread use. Wearable technology is becoming increasingly popular, and it has become an essential component of the IoT ecosystem. Risk and safety considerations cannot be disregarded as wearable technology is fast evolving and approaching consumers. Neither the use nor concerns about the safety of wearable gadgets are on the rise. When compared to data acquired from a smartphone, tablet, or other gadget, the data from a smart wearable gadget might have more impact. Many sensors are included in these high-tech wearables, allowing for continual observation of the wearer's behaviour and the collection of massive amounts of information. Among the more well-known sensors are the Global Positioning System (GPS), speech recognition technology, heart-rate recording, and gyroscope. These devices collect Personally Identifiable Information (PII), which might be a significant risk issue for a user, network, or enterprise. The complexity of the processes required for collecting, analysing, storing, and sending data from and into smart wearable gadgets is a serious security risk (Sami Ali & Emaduldin Abdulmunem, 2020). This means that smart wearables have opened a new avenue for assault.

Due to the individual nature of the users, the information gathered by smart wearable gadgets is also really distinct. Smart wearable gadgets have a big challenge because of the constraints of their computing, storage, and battery capabilities. As a result, conventional methods of keeping data safe can't be put into action. Because of the various designs of these gadgets, analysis is a challenging endeavor. These privacy and security concerns around smart wearable gadgets may be addressed by seeing them as an expansion of smart-phones, which is a component of the Internet of Things environment. These gadgets are known to wear on the person and regularly gather information about the user and their environment. This makes these gadgets easy targets for intruders.

A network's regular and abnormal traffic flows may be distinguished via the use of an intrusion detection system, which observes network activity in real time. To distinguish harmful network activity from regular network activity, intrusion detection systems often use binary classification or information on all aspects. While IDS may not stop an attack from happening, it may alert administrators to any intrusions or invasions that have already taken place on their networks. Because of its inclusion in the IoT category, intrusion detection technologies may be designed specifically for smart wearable gadgets. One of the main purposes of intrusion detection is to make it easier for classifiers to spot bad behaviours (Sami Ali & Emaduldin Abdulmunem, 2020). Based on these findings, a lightweight model approach is recommended.

This paper is organized as follows: Section 2 discusses the different approaches used in intrusion detection systems. Section 3 contains a full explanation of the utilized methodology, Section 4 contains the design specification for the implementation, Section 5 covers the model of implementation and assessment, and Section 6 summarizes the article with future work possibilities and constraints.

1.1 A Review of the Diverse Intrusion Detection Systems

The purpose of an intrusion detection system (IDS) is to give a warning if unusual or suspicious behaviours is discovered in a network. IDS advancements are dependent on considerations including positioning, monitoring, and evaluation strategies. Intrusion detection system types and techniques are shown in **Error! Reference source not found.**.

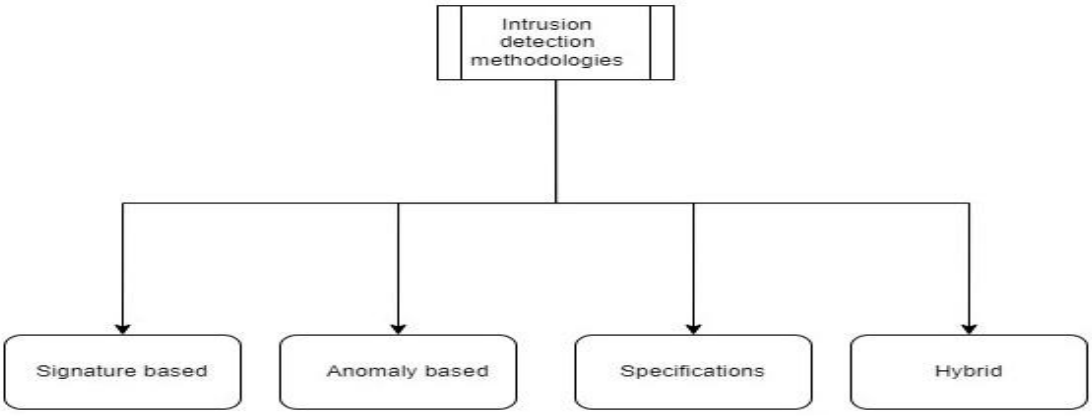

Figure 1

Anomaly based: Due to the fact that this strategy relies on analysing network events to spot intrusions, it goes by another name: the event-based technique. During the investigated time period, the network's typical activity is observed and characterized. A network anomaly or intrusion is detected when the network traffic flow deviates from the norm (Jyothsna et al., 2011). Compared to signature-based methods, anomaly-based intrusion detection is more effective (Nobakht et al., n.d.). An inconspicuous approach, little overhead, and scalability are all important factors in making efficient use of anomaly-based intrusion detection.

Specification based: It's comparable to "anomaly-based detection" techniques. The system's security mechanisms and regulations will establish the expected frequent behaviour. Security policy violations are actions that go against the norm. This is an anomaly-based and misuse-based intrusion detection system (Sami Ali & Emaduldin Abdulmunem, 2020).

Hybrid approach: This strategy, which is a synthesis of the aforementioned techniques, may identify intrusions more precisely and reliably and protect against a broader range of assaults. Given that the aforementioned solutions have their drawbacks, SVELTE and INTI IDS are the two most common hybrid models used for intrusion detection. Instead of relying on a signature model as SVELTE (Cervantes et al., 2015) does, INTI IDS makes use of an anomaly concept.

The content sections of your report should of course be structured into subsections. Note that here there are 2 subsections subsection 2.1 and subsection 2.2.

2 Related Work

The emergence of cutting-edge technology has led to the development of the Internet of Things, which links together all kinds of previously separate everyday devices. From a mere fashion item to more functional and specialized use, smart wearable gadgets have evolved into a vital aspect of the IoT. In order to communicate and exchange information that may inform intelligent action, these gadgets link either directly or via a master-slave relationship to the internet. Unfortunately, the privacy of such wearable technologies is not given the attention it deserves, and thus, there has been a lack of dedicated study in this area. This research synthesis illustrates the cyber threats facing modern wearable Internet of Things gadgets. According to a recent analysis of cyber threats, wearable technology gadgets are poised to become a primary target for ransomware attacks. Data that is unique to a user has a greater risk of being misused. As an extension of the smartphone, wearable technology gadgets can do almost everything a smartphone does. These include behavior detection and biomedical applications, as well as card swipers and home automation management. Human Activity Recognition (HAR) is an integral part of the overarching framework used to track people's daily routines in order to help with anything from clinical diagnosis to recovery to providing elderly care to amusement to espionage (Kuang et al., 2014). A plethora of intrusion detection methods have been created and then mostly rendered obsolete during the past several years. In this part of the report, we'll analyse and contrast these methods so that we may arrive at a more optimal answer.

2.1 Literature review

Intrusion detection and the identification of different kinds of assaults are two areas where machine learning techniques are being employed extensively. Using the NSL-KDD data, researchers concluded that conventional machine learning techniques rely heavily on shallow learning (Raza et al., 2013). Automatic learning machine learning algorithms are becoming

less effective for large-scale intrusion detection as database sizes grow(Amouri et al., 2020). There has been a dramatic and ongoing improvement in deep learning's algorithms and methodology. Using a combination of convolutional layers and regression for ultrasonic waves, the authors of this work suggest a method for detecting intrusions in wearable computers connected to the Internet of Things. Several machine learning-based approaches were presented and shown to be effective for intrusion detection. Some of these are kNN (K-nearest neighbors) (D et al., 2018), Support Vector Machine (SVM) (Almomani et al., 2016), Random Forest (RF) (Naseer & Saleem, 2018), self-organizing maps (SOM) (Jyothsna et al., 2011), and naive Bayes networking (Webster et al., 2016).

The use of deep learning for the identification of intrusions is a relatively recent field of study. To simplify matters, we might say that an RNN (recurrent neural network) is a kind of deep learning artificial neural network. using a dataset with features selected and four infiltration types. As a result, the author of (Dai et al., 2022) suggests a triple-layer RNN architecture. This approach can't handle high-dimensional modeling since layers are only loosely related to one another. In (Raza et al., 2013), an RNN-IDS strategy is suggested for multiclass and binary classification using the NSL-KDD database. The results are compared to those of a traditional machine learning technique.

If compared to a vast amount of data, deep learning improves accuracy and may generate a more representative picture. In (Riecker et al., 2015), the author suggests using a CNN (convolutional neural network) in tandem with an RNN (recurrent neural network) to reap the full benefits of a DNN (Deep Neural Network) using the information on actual internet traffic.

The results of training a threat intelligence dataset using a DBN (Deep Belief Network) and an RBM (Restricted Boltzman Machine) in (Alom et al., 2015) were unimpressive. The cable network's enhanced ability to analyses complex data is thanks, in part, to this algorithm's inspiration from the cerebral cortex. Nonetheless, this was a flawed theoretical framework. The 7-point plan was developed by Bon temps et al.

Detection of intrusions using a version of the (LSTM) Long Short-Term Memory algorithm produced inaccurate results. Wang et al. (Sami Ali & Emaduldin Abdulmunem, 2020)developed characteristics spatially using network traffic using a convolutional neural network, which they applied to infection detection and traffic classification. In (Bianchi et al., 2019), a new approach to infiltration detection in industrial IoT devices using CNN-Regression synthesis is described. When applied to the NSL-KDD datasets, this approach yielded very precise results.

This study makes use of the WSN-DS datasets because of their relevance to the study of tiny IoT devices. UNSW-ND15 is the most recent intrusion detection dataset, and it has already surpassed the previous benchmarks, KDD99 and NSL-KDD(Mukherjee & Sharma, 2012). However, none of the data sets mentioned above can be used for intrusion detection in wearable devices or other similarly small IoT devices.

2.2 Literature Gaps

The market currently lacks a specialized intrusion detection solution for wearable smart IoT devices. That's why we only analyzed several current general intrusion detection algorithms. As it stands, current models rely heavily on inspecting the payloads of TCP/UDP packets to identify a related or comparable characteristic of the malware. Computer viruses evolve on a

daily basis, and the most recent strains encode both their signatures and their information flows to hide from IDS. As we have shown, signature-based intrusion detection does not yield useful and reliable results.

In this research, we present anomaly-based threat intelligence, a method for identifying attacks by analyzing network data for unusual behavior. Convolutional Neural Networks have been shown to be more effective in detecting network intrusion and producing accurate results than other learning techniques such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) (Bianchi et al., 2019). Smart wearable IoT devices may identify infiltration via the merging of several convolution layers and regression layers, CNN-R and WSN-DS. Due to the permanent nature of the output, CNN employs a regression function rather than a categorization mechanism. In order to train the model to its optimal configuration and assess how well it performs in various settings, we examine the distributions of the regression method's output values.

2.3 Effectiveness of the CNN-R

Convolutional Neural Network is a deep learning algorithm, that can feed on 1-dimension, 2-dimension, and 3-dimensional data.A wide variety of data types, including textual, sound, videos, and pictures, may be processed with great precision using this method. This process takes a dataset and transforms it into an image format that the convolution layers can use. A CNN's layers upon layers, including the pooling and convolution layers, are entirely linked, and regularization is implemented via dropout layers. This lays the groundwork for sending data directly to CNN without any intermediaries. As a result, images can be directly fed into networks, simplifying the segmentation method and data reconstruction. Because of innovations such as sparse connections, pooling, and shared weights, the complexity of manual operations is significantly reduced while productivity is greatly increased. Because 1D-CNN can understand unlabeled data, it has several uses in detecting network intrusions. This has the potential to outperform conventional machine learning algorithms in terms of accuracy. In addition to being able to analyze and comprehend complex data, 1D-CNN-Regression also has the added benefit of being a lightweight model that can be utilized in more compact and less resource-intensive networks. Information from wearable technology devices' network activity analyzed by a 1D-CNN using as few resources as possible makes it possible for low-cost deployment. You may use a regression layer at the network's conclusion to make predictions about distances and angles, both of which are continuous data types.

3 Research Methodology

There are many steps involved in doing research. First, data has to be collected, processed, and converted into pictures. Then, after identifying the outlier values, you may feed them into a regression layer. After completing a dataset's pre-processing stage, the implementation consists of training, testing, and evaluation.

3.1 Dataset description

In this research, we are using the WSN-DS (Li et al., 2014) datasets for usage with wireless sensor networks. The Wireless Sensor Network, from which this information is derived, is widely utilized for a wide variety of real-time apps, including those dealing with building security, health care, education, military applications, and wildfire surveillance, to name a few.

A variety of sensor nodes are dispersed over a large area to collect information that is then sent over a wireless network to a central, highly powered node called the Sink Node or the Base Station. WSN also uses innovative technology, which gathers information from various sensors and sends it to a central hub, in this case, a smartphone. Intruder detection in wearable gadgets is where this dataset shines. This dataset has 19 features, including attack types, ranks, timestamps, and more. To produce a 5x5 matrix, six more attributes are added to this dataset during processing. Gray holes, blackholes, schedules, and floods are different kinds of attacks. There are 3,744,662 entries in this collection. This dataset appears to have 10% more training data and 90% more test samples. Both typical and unusual patterns in the data were identified. The different penetration attempts are part of the phenomenon. Data about the connectivity of smart devices may be gleaned from the density of nodes clustered. This quality is a distinct advantage for maximizing resource efficiency. This data is tailored to lightweight IoT gadgets and other gadgets that use sensors, as opposed to other general datasets like CICIDS18, UBSW-NB15, KDD99, NSL-KDD, UBSW-NB15, DARPA98, and NSL-KDD, CICIDS18, each of which shares similar data (Li et al., 2014).

3.2 Data clustering

CNN-R, a deep learning method, needs visual data as its input. While WSN-DS is an array-based dataset consisting of strings and numeric data, or just data in general, this data is transformed into grayscale image photos for use in the 1D to 3D conversion process. After that, 1D-CNNR is trained and tested using these monochrome pictures. The attributes in the dataset need to be digitized to be used in the grayscale image conversion. To encode the label of the features contained in the dataset, a column for each character feature is created and filled with 0 and 1 values based on their numerical value. The dataset will be ready for training once the label encoding is finished. It is common practice to conduct normalization on a dataset to slow down the convergence speed and dampen the training impact of a model when it encounters characteristics with significant value variations. When the database is adjusted, the values will all be in the range of 0 to 1. Simple regression will be used to normalize just the numeric characteristics.

The formula for A 'is: A-Amin/(Amax-Amin)

Assume that Amax is the maximum value of the characteristic and that Amin is the minimum value. The characteristics with a significant variance will be found, and normalizing will be applied to them. With proper label encoding and normalization, a dataset may be used effectively for training.

3.3 Conversion into matrix

While the dataset has been prepared for implementation, it still exists as an array. After the dataset is transformed into a matrix, it will be used to generate grayscale pictures. Only characteristics that can be converted directly into a matrix are altered. If a feature's value is too low for a straight matrix conversion, it is first expanded using randomly repeated features and later transformed. The need to extract the correlation between characteristics aids the network in its quest for structural information. No information from the dataset is lost in converting it to photos, and this consistency will be sustained throughout the whole procedure to ensure maximum precision. As 255 is the full potential value of a pixel, the final step in the pre-processing phase, just before the picture is input into the 1D-CNN model, is to divide the values in the image by 255.

3.4 Regression layer

Estimating the significance of a continuous outcome variable (y) based on the value of one or more predictor variables is the goal of regression analysis, a kind of machine learning (x). The purpose of a regression model is to establish a mathematical equation that explains the connection between two independent variables, x and y, using a parameter, y.

Regression requires the user to construct a relationship diagram between the variables that best fits the available data. The regression line in malware classification illustrates the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis) (Y-axis). A gradient format is employed when determining the best-fit lines in a regression analysis.

The regression technique seeks the optimal a0 and a1 inputs by locating the line of best fit. To achieve its goal, a process must determine the best-fit line with the least amount of error. The Mean Squared Error (MSE) function is often used in regression analysis. As a result, the optimal values for a0 and a1 may be calculated, and the best-fitting line can be drawn from experimental data. The MSE function may be used to fine-tune the data variables a0 and a1 such that the mse value is exactly zero or one, respectively. This is an approach to minimizing the objective functions via updating a0 and a1.

3.5 Training the model

Different subsets of the dataset all use the same one-dimensional CNN regression framework. CNNR's method of intrusion detection yields a yes-or-no verdict. The building blocks of each model are an input dataset, 3 convolution layers, and a pooling layer. A dropout layer is used for regularization between the fully interconnected and flattened layers. For each layer, the equation is a rectified linear unit (ReLU). The convolutional layer (the second layer) receives the 1D model's convert to 3D model for the input layer's 3D matrix. The 32 2-by-2-cell filters with zero padding retrieve the input data in the second layer. In the third layer, 62 filters with a stride of 1 examine the information. In the same vein, the filter size is raised in the fourth layer. The activation function (ReLU), pooling functional (Max), normalizing function (normalize), and dropout (0.1) are all included in each layer. After the flattening layer in 1D-CNN models, the outputs are concatenated and processed in the regression layer. Consequently, the model's one-of-a-kind attributes.

3.6 Testing the model

The dataset is split into a training and testing set with a ratio of 30:70. In this scenario, 70% of the data is utilized for training, while the remaining 30% is put to the test. The trained model is run over the training sample to find out how accurate the CNN-R model is. Once the model is produced from the training stage, the same procedure is repeated during testing.

3.7 Evaluation

The confusion matrices generated by the "sklearn" package are used to assess the performance of the tested models. Precision, recall, and accuracy may all be determined from the numbers in the confusion matrix. The effectiveness of the suggested model is largely dependent on its level of accuracy. The flow diagram is shown in the Figure 2

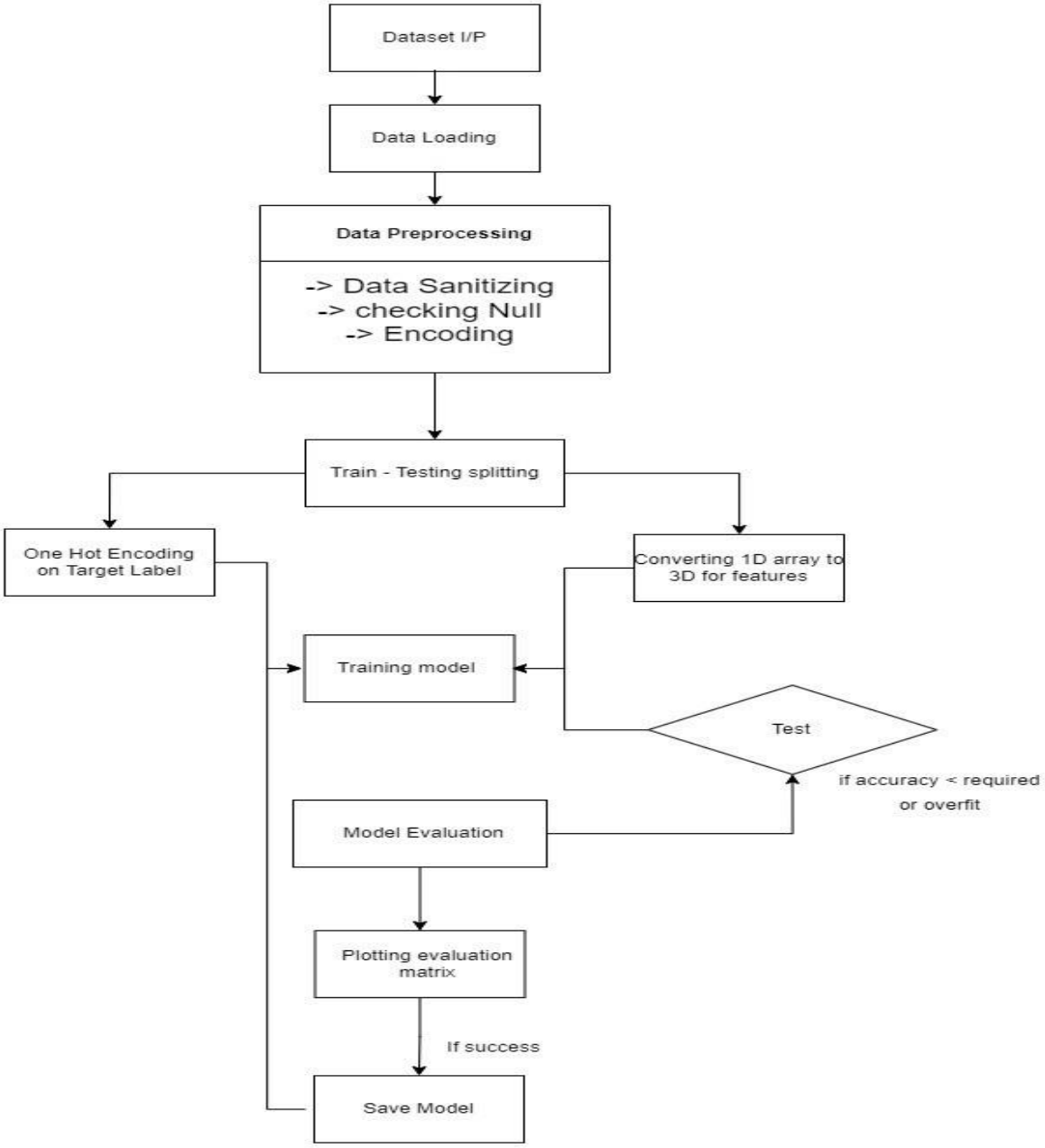

Figure 2

4 Design Specification

There are three main steps constitute the intrusion detection system design flow

Data preparation: The dataset is WSN-DS, and data pre-processing is done by reading the CSV file. The Jupyter notebook, as well as the Numerical Python Framework, are used to process the characteristics of the dataset. This method will purge the dataset of duplicate and void records or null records. The next step involves transforming arrays into matrices, which are later used to produce the primary input for CNN. From the data we acquired the co regulation heat map between columns and labels

Modelling: The creation of the machine learning and deep learning algorithms CNN-R occurs mainly at this implementation stage. The Tensorflow and Keras frameworks are used for this purpose. A graphics driver is used with the Jupyter notebook to improve efficiency and reduce execution time. By using both the graphics processing unit (GPU) and the central processing unit (CPU), all of the system's resources are put to good use. Accuracy, F1-score, recall, and precision are also produced as assessment metrics.

Visualization: In the second step, we'll take the 1D array from the output and convert it to 3D, and then we'll take the metrics we acquired from that into visual forms like graphs and a confusion matrix so that the readers can easily grasp them.

5 Implementation

A deep learning approach, the CNN-R model, can analysed and make sense of complex data. Since CNN-R is an image recognition algorithm, the first step in implementing it is to feed the algorithm data in the form of a gray-scale picture. From the output the final regression layer is use to find the variable.

5.1 Dataset preparation

The WSN-DS dataset has been split into two CSV files, ready for the transformation from array to grayscale pictures. First, the rows with blank values are filtered out of the dataset, and then the row 'label' is extracted. Since the dataset needs to be transformed into a matrix, the next step is label encoding, in which the string values are converted into binary entries by adding a new row for each string in the dataset. From the heat map got the values and labels Then, normalization is performed to lessen the variation in each row's values. In this manner, the preliminary processing of the data is finished.

The dataset is divided into a test set of 30% and a training set of 70% using the "train test split" and "sklearn.model selection" routines. Twenty-five features are reduced to a two-by-two matrix. These matrices' values are kept in separate.jpg files, one for each of the test and training variants. Then splitting data for training and testing is completed from one hot encode we categorised the columns because it not in numerical form so we use the Kreas hot encoder.

5.2 Training the CNN-R

After being preprocessed, the pictures are changed into grayscale images, the matching labels of the images are put into the Y train variable, and the array of grayscale images is loaded into the X train variable. This is done by dividing each value in the matrix by 255, 255 being the maximum number of pixels that may be included in a picture. on data where the hyperparameters utilized were a batch size of 128, and 80 epochs, with validation data that is used to verify the performance of the model on data that has not yet been observed. The program for the CNN is given in the **Figure 3** In the second model, the batch size is 64, and epochs is 80 from the result, getting an accuracy of 97.96 %.

```
model = keras.models.Sequential()
model.add(Conv1D(filters=6, kernel_size=21, strides=1, padding='same', activation='relu',
                 input_shape= (x_train.shape[1],3),kernel_initializer=keras.initializers.he_normal()))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Conv1D(filters=16, kernel_size=5, strides=1, padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84))
model.add(Dense(5, activation='softmax'))
model.summary()
```

Figure 3

The 'Sequential()' method is used to launch a CNN model, and data from the input layer is fed into the three-layer CNN. Sizes of 64 and 128 for filters in layers 1 and 2, with a 2x2 kernel, are used.'Relu' and 'maxpooling' are used to create a 2x2 layer for each successive layer. The summary model of the CNN is given in the **Figure 4**.

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_12 (Conv1D)          (None, 17, 6)             384

 batch_normalization_12 (Bat  (None, 17, 6)            24
 chNormalization)

 max_pooling1d_12 (MaxPoolin  (None, 9, 6)             0
 g1D)

 conv1d_13 (Conv1D)          (None, 9, 16)             496

 batch_normalization_13 (Bat  (None, 9, 16)            64
 chNormalization)

 max_pooling1d_13 (MaxPoolin  (None, 5, 16)            0
 g1D)

 flatten_6 (Flatten)         (None, 80)                0

 dense_18 (Dense)            (None, 120)               9720

 dense_19 (Dense)            (None, 84)                10164

 dense_20 (Dense)            (None, 5)                 425

=================================================================
Total params: 21,277
Trainable params: 21,233
Non-trainable params: 44
```

Figure 4

After all these layers have been processed, the final matrix is sent to the flatten layer, where the 'flatten()' code transforms it into a matrix with just one column.The flatten layer's output is combined with the previous one and sent to the hidden layers as a single input. The array output is given in the Figure 5

In the last layer of regression, the output is used to assign new values to the input.There are two layers under the surface: a dense layer and a dropout layer. Next, the CNN-R model's output layer uses a softmax activation function with the two-unit values "normal" and

"anomaly" as inputs.Model training starts with a batch size of 128, and after 80 iterations, the "adam" optimization is used to find the optimal loss value for the binary cross-entropy error function, which gives the most accurate prediction.

```
x_train, x_test, y_train, y_test = train_test_split(np.asarray(x), np.asarray(y), test_size = 0.3, random_state = 42)
```

```
x_train
```

```
array([[1.0300000e+02, 0.0000000e+00, 1.0206900e+05, ..., 0.0000000e+00,
        2.0000000e+00, 6.1750000e-02],
       [3.5530000e+03, 0.0000000e+00, 1.0011000e+06, ..., 1.7047201e+02,
        0.0000000e+00, 6.9159000e-01],
       [9.5300000e+02, 0.0000000e+00, 1.1900100e+05, ..., 0.0000000e+00,
        4.0000000e+00, 5.7320000e-02],
       ...,
       [5.5300000e+02, 0.0000000e+00, 1.1110000e+05, ..., 0.0000000e+00,
        8.0000000e+00, 1.5219000e-01],
       [4.5300000e+02, 1.0000000e+00, 1.0901400e+05, ..., 1.0341345e+02,
        0.0000000e+00, 1.9326300e+00],
       [1.0030000e+03, 0.0000000e+00, 2.0110000e+05, ..., 1.5031682e+02,
        0.0000000e+00, 1.6315500e+00]])
```

Figure 5

After all these layers have been processed, the final matrix is sent to the flatten layer, where the 'flatten()' code transforms it into a matrix with just one column.The flatten layer's output is combined with the previous one and sent to the hidden layers as a single input.

In the last layer of regression, the output is used to assign new values to the input.There are two layers under the surface: a dense layer and a dropout layer. Next, the CNN-R model's output layer uses a softmax activation function with the two-unit values "normal" and "anomaly" as inputs.Model training starts with a batch size of 128, and after 80 iterations, the "adam" optimization is used to find the optimal loss value for the binary cross-entropy error function, which gives the most accurate prediction.

Training the models using accuracy metrics helps verify the reliability of the testing dataset. The.h5 file extension is used to save and append the trained model because it is a small model that can be used on IoT devices in real time.The one-dimensional CNN-R model is shown in Figure 3.

5.3 Testing with CNN-R

Images for testing are read from the specified directory path and processed into an image list before being compared to the entire data set. Then, the procedure used during training is repeated. The "load model" method is used to access the trained model, and the "model predict" function is used to send the inputs to the model. An accuracy of 97.96% is achieved after running the input through the trained model. The accuracy, precision, recall, and F1-score are all calculated using the results of the testing step, which yields the confusion matrix. This testing is a simulation of how the model would function in a real-world application; in reality, the value from network traffic will be substituted for the datasets.

6 Evaluation

The effectiveness of deep learning is judged based on a small number of metrics, which are determined with the help of the "sklearn.metrics" library package.

6.1 Evaluation metrics

For an intrusion detection system built on the CNN-R model, accuracy is the most crucial indication of functionality. In addition to the binary classification result, other metrics like precision, false positive rate (FPR), recall, and F-score are also employed.
True positive (TP): This demonstrates that the system correctly identified Malware as malicious.
True negative (TN): This means that a program using optimistic visuals is correctly classified as having benign code.
False positive (FP): Because of this, a harmless piece of malware has been mistakenly labeled as malicious.
False negative (FN): This means that malicious code hasn't been found yet and is being treated like normal software.
Accuracy is defined as the ratio of records that were properly categorized to the total number of records in the dataset.
Accuracy = (TP + TN) / (TP + TN + FP + FN)
Precision: the ratio of accurate to incorrect identifications is used as a measure of performance.
Precision = TP / (TP + FP)
A false positive rate (FPR) is the number of wrongly classified records as a percentage of the total number of outliers.
FPR = FP / (FP + TN)
Recall that it's the proportion of legitimately categorized records to the total number of outliers. The True Positive Rate (TPR) is another name for this.
Recall = TP / (TP + FN)
To get an F-score, use the harmonic mean of the recall and accuracy metrics. This means that a program using optimistic visuals is correctly classified as having benign code.
F-score = (2 * Precision * Recall) / (Precision + Recall).

From the implementation we found the confusion matrix chart is given **Figure 6**

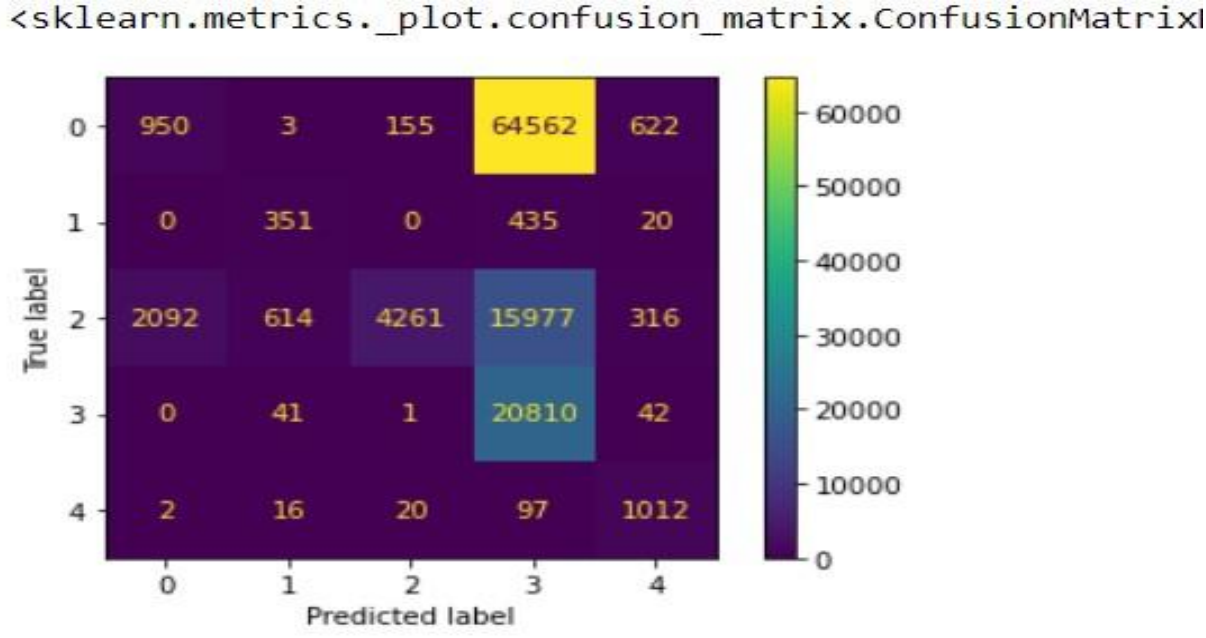

Figure 6

The accuracy and loss that occurred throughout the training phase of the CNN-R model are graphically shown in the Figure 7. Within 80 epochs, the accuracy went from 95% to 99%, and the loss went from 0.14% to 0.02% shown in Figure 7. The CNNR model works well because both accuracy and loss go up as the model gets better.

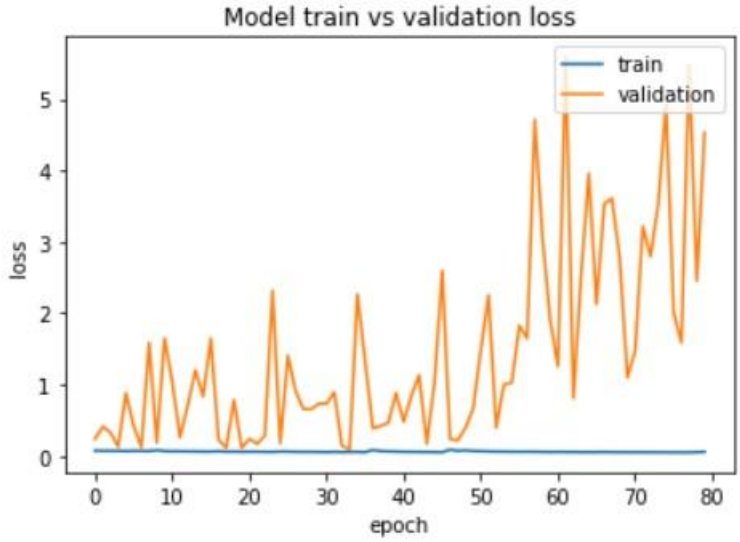

Figure 7

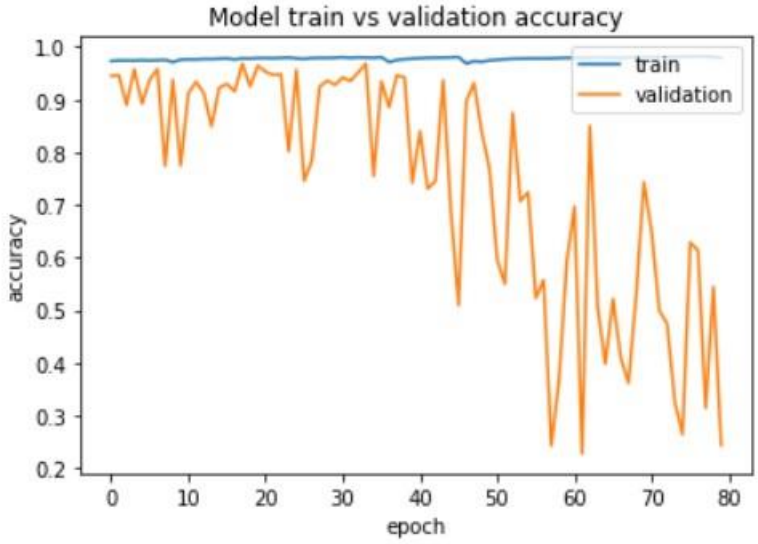

Figure 8

7 Discussion

The major goal of this study is to develop an intrusion detection system tailored to wearable technology and Internet of Things devices. As a result of this study, a lightweight model capable of identifying anomalies in network flow was developed using the WSN-DS dataset implemented in a 1D-CNN-R model. Since smart watches and bracelets are essentially an extension of smartphones, this compact design may be easily integrated into gadgets and other compact IoT devices. Due to the rapid development of technology, wearable gadgets will soon have enough storage and processing capability to house a lightweight form of an intrusion

detection system. Our finished model only took up 401 kilobytes (KB), which is tiny compared to even the most minimalistic smartphone app or intelligent wearable device software. Thus, the possibility of the lightweight model being used in any mobile device or smaller IoT device is increased. When TensorFlow light is used, a lot less computing power is needed to run this model.

8 Conclusion

The study used a one-dimensional-CNN-R lightweight model to identify intrusions in smart wearable IoT devices with an accuracy of 97.96%. The results from this model were more precise than those from a standard CNN model. The Wireless Sensor Network includes innovative wearable Internet of Things (IoT) devices, and to identify intrusions in this network with a high degree of accuracy, the dataset developed for intrusion detection is employed. Mobile or smaller IoT devices may use the CNN-R light model to detect infiltration in real-time. Although it only used a subset of the libraries' features, the lightweight model functioned well. This method will provide very accurate intrusion detection in wearable devices.

Problems Existing benchmark intrusion detection statistics are not tailored to the latest miniaturized hardware. Wearable IoT devices and network data flow were critical to the WSN-DS datasets. Unfortunately, no similar information exists for Wireless Body Area Networks, which are the precise networks for smart wearable IoT devices. However, due to a lack of computing resources, it was not possible to compare the suggested model against a wide variety of machine learning and deep learning algorithms, which would have provided even stronger evidence for 1D-CNNR-superiority. R's' Since each epoch loop took between 18 and 21 seconds to finish, the iteration allotted was 80. More than 97.96% accuracy may have been achieved by increasing the number of epochs beyond 100.

Due to the extensive pre-processing and image-generation required by 2-dimensional CNN-R, this concept was executed with some delay. If the CNN-R model can shorten the time it takes to process 2-dimensional data before it is used, it will be even more useful for small IoT devices.

References

Almomani, I., Al-Kasasbeh, B., & Al-Akhras, M. (2016). WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks. *Journal of Sensors*, *2016*. https://doi.org/10.1155/2016/4731953

Alom, M. Z., Bontupalli, V., & Taha, T. M. (2015). Intrusion detection using deep belief networks. *Undefined*, *2016-March*, 339–344. https://doi.org/10.1109/NAECON.2015.7443094

Amouri, A., Alaparthy, V. T., & Morgera, S. D. (2020). A Machine Learning Based Intrusion Detection System for Mobile Internet of Things. *Sensors (Basel, Switzerland)*, *20*(2). https://doi.org/10.3390/S20020461

Bianchi, V., Bassoli, M., Lombardo, G., Fornacciari, P., Mordonini, M., & de Munari, I. (2019). IoT Wearable Sensor and Deep Learning: An Integrated Approach for Personalized Human Activity Recognition in a Smart Home Environment. *Undefined*, *6*(5), 8553–8562. https://doi.org/10.1109/JIOT.2019.2920283

Cervantes, C., Poplade, D., Nogueira, M., & Santos, A. (2015). Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. *Proceedings of the 2015 IFIP/IEEE*

International Symposium on Integrated Network Management, IM 2015, 606–611. https://doi.org/10.1109/INM.2015.7140344

Dai, Q. Y., Zhang, B., & Dong, S. Q. (2022). A DDoS-Attack Detection Method Oriented to the Blockchain Network Layer. *Security and Communication Networks*, *2022*. https://doi.org/10.1155/2022/5692820

D, B., Chakrabarti, A., & Midhunchakkaravarthy, D. (2018). Smart Devices Threats, Vulnerabilities and Malware Detection Approaches: A Survey. *European Journal of Engineering Research and Science*, *3*(2), 7. https://doi.org/10.24018/EJERS.2018.3.2.302

Jyothsna, V., v. Rama Prasad, V., & Munivara Prasad, K. (2011). A Review of Anomaly based Intrusion Detection Systems. *International Journal of Computer Applications*, *28*(7), 26–35. https://doi.org/10.5120/3399-4730

Kuang, F., Xu, W., & Zhang, S. (2014). A novel hybrid KPCA and SVM with GA model for intrusion detection. *Applied Soft Computing*, *18*, 178–184. https://doi.org/10.1016/j.asoc.2014.01.028

Li, W., Yi, P., Wu, Y., Pan, L., & Li, J. (2014). A new intrusion detection system based on KNN classification algorithm in wireless sensor network. *Journal of Electrical and Computer Engineering*, *2014*. https://doi.org/10.1155/2014/240217

Mukherjee, S., & Sharma, N. (2012). Intrusion Detection using Naive Bayes Classifier with Feature Reduction. *Procedia Technology*, *4*, 119–128. https://doi.org/10.1016/J.PROTCY.2012.05.017

Naseer, S., & Saleem, Y. (2018). Enhanced network intrusion detection using deep convolutional neural networks. *KSII Transactions on Internet and Information Systems*, *12*(10), 5159–5178. https://doi.org/10.3837/TIIS.2018.10.028

Nobakht, M., Sivaraman, V., & Boreli, R. (n.d.). *A Host-based Intrusion Detection and Mitigation Framework for Smart Home IoT using OpenFlow*.

Raza, S., Wallgren, L., & Voigt, T. (2013). SVELTE. *Ad Hoc Networks*, *11*(8), 2661–2674. https://doi.org/10.1016/J.ADHOC.2013.04.014

Riecker, M., Biedermann, S., el Bansarkhani, R., & Hollick, M. (2015). Lightweight energy consumption-based intrusion detection system for wireless sensor networks. *International Journal of Information Security*, *14*(2), 155–167. https://doi.org/10.1007/S10207-014-0241-1

Sami Ali, A., & Emaduldin Abdulmunem, M. (2020). Image classification with Deep Convolutional Neural Network Using Tensorflow and Transfer of Learning. *Journal of the College of Education for Women*, *31*(2), 156–171. https://doi.org/10.36231/COEDW/VOL31NO2.9

Webster, B., Arrasmith, W., & Acharya, L. (2016). Optimizing Mobile Asset Protection in Areas Where Protective Resources Are Limited. *International Journal of Modeling and Optimization*, *6*(1), 49–55. https://doi.org/10.7763/IJMO.2016.V6.502