Australian Journal of Basic and Applied Sciences

ISSN :1991-8178
EISSN: 2309-8414

2023 April; 17(4): pages 1-18
https://doi.org/10.22587/ajbas.2023.17.4.1

ORIGINAL ARTICLE AENSI Publications

IOT Objects Identification Model Using Deep-Neural Network (DNN)

Sana Abdelaziz Bkheet[1, 2], Johnson I. Agbinya[3]

[1]Sudan University of Science and Technology (SUST) – Khartoum – Sudan, Faculty of Computer Science and Information Technology, Khartoum, Sudan
[2] Northern Border University (NBU), Faculty of Science, Department of Computer Science - Arar - Kingdom of Saudi Arabia
[3]Melbourne Institute of Technology, Australia, School of Information Technology and Engineering, Melbourne, Australia

Correspondence Author: Sana Abdelaziz Bkheet, Sudan University of Science and Technology (SUST), Faculty of Computer Science and Information Technology, Khartoum, Sudan
Northern Border University (NBU), Department of Computer Science, Faculty of Science, Arar, Kingdom of Saudi Arabia
E-mail: sana.bkheet@nbu.edu.sa
https://orcid.org/0000-0001-9050-3397

Received date: 25 January 2023, Accepted date: 30 April 2023

Citation: S. A.Bkheet, Johnson I. Agbinya., 2023. IOT Objects Identification Model Using Deep-Neural Network (DNN). Australian Journal of Basic and Applied Sciences, 17(4): 1-18. DOI: 10.22587/ajbas.2023.17.4.1.

ABSTRACT: This research paper proposes a novel method for identifying smart devices based on their various attributes with the integrated use of Deep Neural Network (DNN). A dataset consisting of over 90 instances of smart home devices gathered completely from online sources is used. It has been structured in Arff and applied to a Weka platform and then classified with Multilayer Perceptron Networks. The instances were classified into specific groups, which made them easier to identify and distinguish from the rest. The method is also applied on a larger dataset which includes over 100 instances, to ensure its accuracy and validation to be used with various smart devices in different IoT environments.

Keywords: Smart home, Deep learning (Dl), Deep Neural Network (DNN), Multilayer Perceptron Network (MLP)

INTRODUCTION

Over the last decade, development in the Internet, where only computing devices could access the Internet, has been improved to include different kinds of devices or objects, creating a new concept called Internet of Things (IOT). The name Internet of Things is a combination of two main words “Internet” that usually defined as the global network that provides a variety of information and communication facilities, consisting of interconnected networks using standardized communication protocols, and “Things” which include various types of device or objects such as automobiles, factories, facilities, animals machines, appliance, farms, etc. These devices must be equipped with sensors, actuators, microprocessors, communication interfaces, and a power source to connect to the Internet. Identification is essential for IOT system success; it is considered proof of identity information regarding each object. It is primarily used to identify objects and allow them to create a connection or relationship with other objects. It enables us to identify and manage many heterogeneous things through the Internet.

Identifying objects is challenging since it concerns applying each object with a unique identity. Furthermore, addressing the IOT objects is necessary because it refers to the object address within a communication network. There are addressing methods for IOT objects like IPv6 and 6LoWPAN addresses, as well as many prior identification methods such as Bluetooth, Radio Frequency Identification (RFID), Barcode/2D code, Near Field Communication (NFC), Electronic product codes (EPC), IP address, etc.

Although all these methods continue to be used, the need to keep up with the rapid development of the Internet of Things and the growth in the number of devices joining the Internet of Things daily has led to the necessity of finding alternative methods for identifying devices that have recently joined an IoT environment, which heavily relies on the use of the software.

As a result of these and other reasons, this work combined Deep Learning and Artificial Neural Networks by implementing Deep Neural Network (DNN) to create a robust and accurate IOT device identification method that will help in easily specifying IOT devices in smart homes and can be extended to be used in different environments.

Artificial Neural Network (ANN) is a part of artificial intelligence (AI) that simulates the behavior and structure of the biological neural network of the human brain, which is a combination of vast numbers of interconnected neurons. Deep Learning (DL) is a modern, powerful, sophisticated version of artificial neural networks. It is essentially concerned with constructing significant and more sophisticated neural networks capable of performing diverse tasks, especially when large amounts of data are involved, such as in IoT applications. Among the main Deep Learning algorithms are Deep Neural Networks (DNNs), Deep Boltzmann Machines (DBMs), Deep Belief Networks (DBNs), Convolutional Neural Networks (CNNs), etc.

Deep Neural Networks (DNNs) have been selected for this work because they can give accurate results in large data sets, such as the data in various IoT environments. So it is applied here on smart home devices. This research used DNN to smart home devices. Among the Internet of Things environments, an intelligent home is one filled with various appliances, which helps to provide sufficient data for this work. The same method can be applied to any future Internet of Things environment.

Smart homes are setups where appliances and devices can be controlled remotely from anywhere with an Internet connection, using a mobile device or other networked devices. With a smart home, all devices are connected through the Internet, allowing the homeowner to remotely control functions such as home security, climate, lighting, and the home theater. Connected devices such as smart doorbells, security systems, and appliances all become part of the Internet of Things (IoT), which consists of a network of physical objects that can gather and share information.

People today are influenced by smart home networks to live a quality life. A smart home provides comfort, a better security system, and more efficiency.

1.1 Related Work

The Smart home has attracted great attention from researchers from various perspectives, including home automation, security, power consumption control, and future development. This part reviews some of the recent related work that has been introduced to provide services that apply to smart homes, such as (Chandramohan et al., 2017) design flexible home control and monitoring system with an integrated micro-web server with internet protocol (IP) connectivity for access and control of equipment and devices remotely using Android-based smartphone application. The main advantage of the proposed system is that it does not need a dedicated server PC as on similar systems. The system introduces a new communication protocol for monitoring and controlling the home environment with more than just switching functionality. The system is equipped with Smart home interfaces to ensure interoperability between Wi-fi devices from various electrical equipment manufacturers, meters and intelligent energy enables products. The authors provide a potent system but implement it only to control lamps and fans; they should extend it to include other devices (Chandramohan et al., 2017). In the same context, (Sagar V & SM Kusuma. 2015) Researchers developed low-cost and expandable home automation systems that allow homeowners to remotely control various appliances, including lights and fans, with the advanced feature of storing device data in the cloud. The plan was implemented using Intel Galileo with cloud networking and wireless communication. Moreover, the system can automatically change the basis of the sensors' data (Sagar V & SM Kusuma, 2015).

Compared with the previous system presented by (Chandramohan J et al., 2017), this one provides control over all home appliances but costs a bit more. Furthermore, (Sagar V & SM Kusuma, 2015) had gotten a significant advantage by enabling remote control of appliances from anywhere around the world through the Internet connection.

According to the article (Beckel et al., 2011), modern homes and buildings have an increasing number of connected devices, which presents numerous opportunities to homeowners such as building managers, device manufacturers and solution providers. Standardized communication protocols like ZigBee and Bluetooth provide physical connectivity, thus providing a basis for smart home applications. All these reasons require a protocol standard to cope with a heterogeneous device landscape, different data formats, managing resource constraints of devices and providing means to react quickly when devices and applications leave or join the system. The paper proposes to address these application-specific requirements in multiple layers of abstraction, and it also provides a classification and detailed analysis of conditions that have to be addressed to enable application development in smart homes. Furthermore, it analyzes WS4D-PipesBox, a multi-layer framework, to illustrate how applications could be developed using multiple layers of abstraction (Beckel et al., 2011). There is much good analysis in this article, but the article would have more strength if they mentioned another approach and made comparisons to demonstrate the best of them. The report presented by (Pradeep et al., 2016) describes the standard connectivity protocols used for smart homes in detail. Furthermore, a list of connectivity challenges was mentioned and categorized based on the described connectivity protocol (Pradeep et al., 2016). The authors presented valuable details about connectivity protocols and challenges but did not make any analysis of the most significant challenge.

Some articles had concerns about security, safety, and privacy issues of smart homes regarding homeowners and properties, such as (Bugeja et al., 2016), which introduced an overview of security and privacy challenges regarding smart home environments, besides identifying the main related constraints, and suggest some solutions and evaluated them (Bugeja et al., 2016). Authors (Lin H & Bergmann N, 2016) reviewed existing solutions for improving IOT security to identify future requirements for creating a trusted Smart Home environment (Lin H & Bergmann N. 2016).

Regarding object identification, the research presented by (Cvitić et al., 2020) aims to analyze the possibilities of applying the various features of smart devices for classifying it. In this research, the authors use the logistic regression method enhanced by supervised machine learning (logitboost) to develop a classification model on a dataset containing 41 IoT devices. They test their Multiclass classification model over 13 network traffic features generated by IoT devices. The results of this research show that it is possible to classify devices into four classes defined by the model and gets a high performance and accuracy of (99.79%) based on the traffic flow features of the devices (Cvitić et al., 2020). The method provided here is powerful and new and gives a high accuracy rate, but the authors don't test the same method with larger data set to ensure it is efficient.

Reviewing the above papers shows that, monitoring and security of smart homes and appliances elicited the most concerns, and, so far, there is no concern or focus on identifying objects in a smart home. In this regard, as well as the ease of obtaining data related to smart home devices due to their vast and readily available, smart home appliances are chosen for use in this study for the implementation of the Deep Neural Network (DNN) model of identifying smart objects. As a first step towards understanding the details of the model, an overview of Deep Learning (DL), Deep Neural Network (DNN) is presented and then concentrates on Multi-layer Perceptron Network (MLP).

1.2 Deep Learning (DL)

Deep Learning (DL) is a type of machine learning that can analyze unstructured data more effectively than traditional machine learning. It is much more powerful and flexible because it can process many features. Compared with traditional learning methods, deep learning improves classifier performance on a large scale with more data (Mathew A. et al. 2021). A deep learning model predicts new data based on patterns extracted from the input data. Such models are designed to mimic the functional design of the human brain (Shetty D, et al. 2020). The term Deep Learning describes a method of learning that is considered a modern update to artificial neural networks. This involves constructing large and complex neural networks with many hidden layers suitable for handling large data sets (Ndhage S & Raina C, 2016) (Zantalis et al., 2019). Different architectures can be used to implement deep learning, including Deep Neural Networks (DNN), Unsupervised Pre-trained Networks (UPN), Deep Belief Networks (DBN), Convolution Neural Networks (CNN), Recurrent Neural Networks (RNN), and Recursive Neural Networks (RNTN) (Mathew et al., 2021; Shetty et al., 2020).

Currently, deep learning is being used in a huge number of applications, including intelligent video analytics, analyzing hyperspectral imagery, image recognition, Natural Language Processing, automatic e-mail responses, automatic machine translation Fraud detection, and healthcare etc. The main advantages of deep learning are simplicity, versatility, short development time, and high performance (Mathew et al., 2021; Shetty et al., 2020).

1.2.1 Deep neural network (DNN)

Deep neural network is inspired by the multi-layer, hierarchically connected neural networks in the human brain, and proved that the architecture of deep neural networks performs better than a simple artificial neural network. As a result of their flexibility and scalability; deep neural networks are suitable for Supervised, Unsupervised, Reinforcement, and Hybrid learning (Mathew et al., 2021; Shetty et al., 2020).

1.2.2 Multi-layer Perceptron Network (MLP)

The Multi-layer Perceptron (MLP) is the most common and widely used type of neural network, also called feed forward. The simple model of such a network was the single-layer Perceptron (Shetty et al., 2020)(POPESCU et al., 2009). MLP is a complex architecture network composed of many units arranged in layers. There are at least three layers in each MLP, including an input layer, one or more hidden layers, and an output layer. In a fully connected network, every node in one layer is connected to every node in the next layer. The structure of MLP is illustrated in Figure 1. Through the layers of a network, signals are only transferred in one direction from the input to the output. Each layer's output becomes the input for the next layer, and so on. There are thresholds associated with each output node, hidden unit node, and weight. Furthermore, the activation function of the hidden layers is nonlinear, while for the output is linear (Shetty et al., 2020; POPESCU et al., 2009;Nazzal et al., 2008). Compared with other classifiers, training a Multi-layer Perceptron is usually quite slow (POPESCU M, et al., 2009).

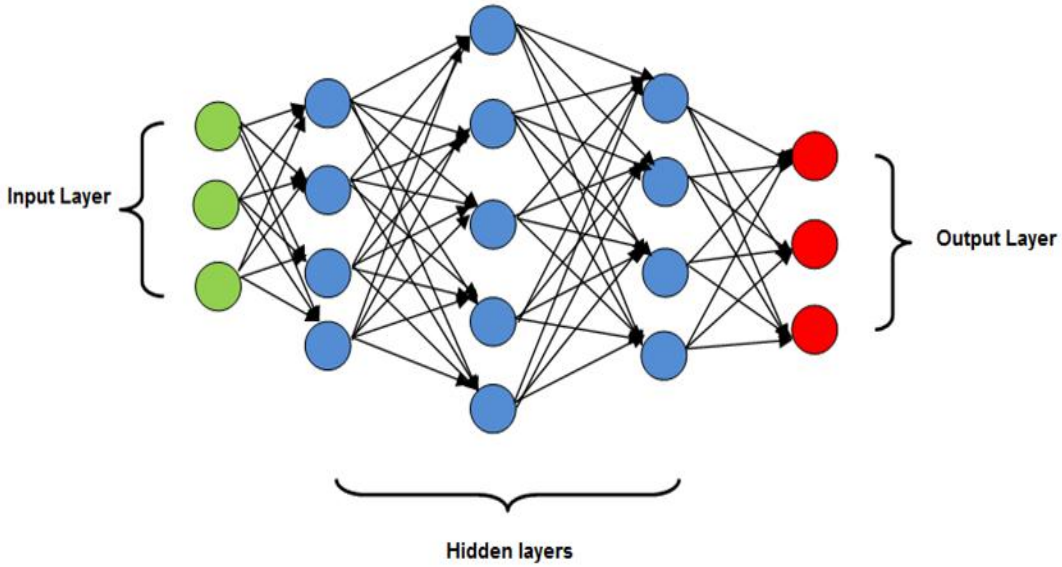


Figure 1: Structure of Multi-layer Perceptron Network (MLP)

Understanding the function of MLP requires first understanding the basic structure of Perceptron which is single layer Neural network shown in the Figure 2. This structure was basically made to simulate the learning ability of biological neuron. The Single layer Neural Network consists of: inputs with its associated weights, activation function and one output.

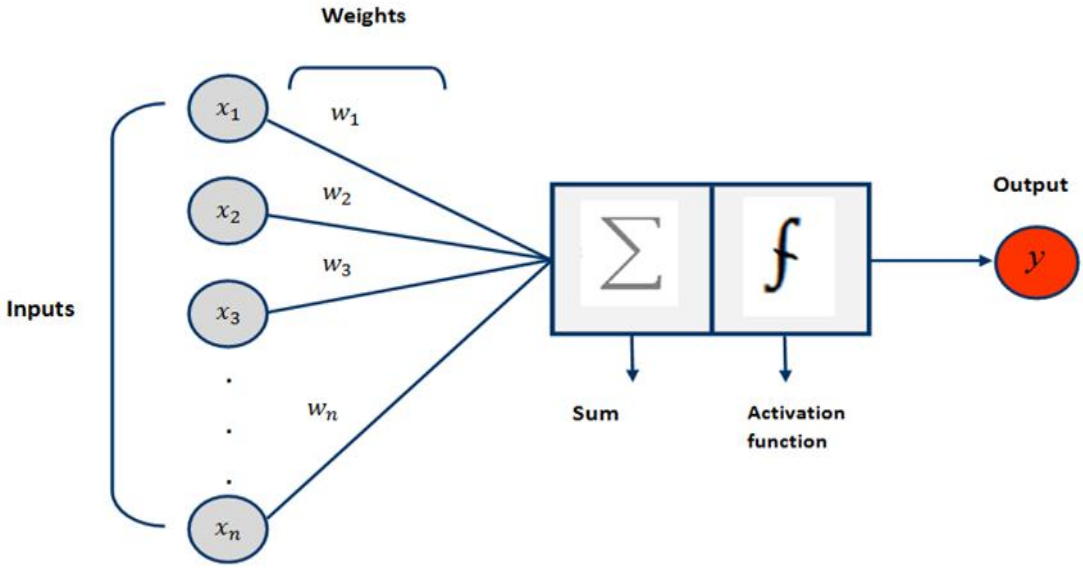


Figure 2: Single layer Neural network (Perceptron)

Input is a real numerical value; weight is a parameter that indicates how strong the connection between units is. The role of activation function is to keep the value of output in the range of (0,1) or (-1,1).

$$y = f\sum_{i=1}^{n} x_i w_i \qquad (1)$$

The expression (1) shows how inputs, weights, and activation function are combined to get the output. The weighted sum is calculated and multiplied by the activation function. The weighted sum is the total sum that multiplies input with its associated weight, denoted by: $(x_1w_1 + x_2w_2 + \ldots\ldots\ldots\ldots\ldots + x_nw_n)$.

This formula has been extended to include bias (b), that is responsible for shifting the curve of activation function up and down.

The following formula represents artificial neuron with weights $(w_1,.....,w_n) \in R$, bias $b \in R$ and activation function $p: R^n \to R$ is defined as the function $f: R^n \to R$ (Kutyniok G. 2022):

$$y = p\left(\sum_{i=1}^{n} x_i w_i + b\right) = p\left(\langle x, w\rangle + b\right) \quad , \qquad (2)$$

Where $X = (x_1,.....,x_n)$, and $W = (w_1,.....,w_n)$.

After running the model if the output does not fit the predefined value, the model gets activated to keep the output in range of (+1 to -1).

The Multi-layer Perceptron (MLP) has the same structure of single layer Perceptron, but it has an extra number of hidden layers and it works through two main steps:

- First step: is usually called the forward step, with the activation function that starts from the first input layer and continues until it reaches the output layer.
- Second step: is known as backward step. In this step the weight and bias values are changed to cope up the model requirements. Moreover, the error between the actual output value and the predefined value is backward on the output layer and ends on the input layer.

Since the MLP has multiple processing layers, the linear activation function of single layer Perceptron does nt fit the MLP, instead various types of activation function can replace linear functions such as: step function, sigmoid function, softplus, etc….

The most popular activation function is the sigmoid function. The sigmoid function is = $\frac{1}{1+e^{-x}}$. The expression (2) will look like the one shown in (3) after adding the sigmoid function:

$$y = \frac{1}{1+e^{-x}}\left(\sum_{i=1}^{n} x_i w_i + b\right) \qquad (3)$$

This paper aims to use the various attributes of smart home devices in developing a Deep Neural Network (DNN) model for identifying smart objects. The rest of the paper is organized as follows: Section 2 contains details and structure of the method proposed for identifying smart home devices. Section 3 contains the analysis of the results. Section 4 concludes the paper.

IDENTIFYING SMART HOME DEVICES USING DEEP NEURAL NETWORK

The proposed objects identification method consists of five procedure stages, as shown in Figure 3, and described briefly as follow:

- First stage: Data Description and Collection (DDC).
- Second stage: Data cleaning and Entry (DE).
- Third stage: Building the Identification Model (BIM).
- Fourth stage: The Experiment and Results (XR).
- Fifth stage: Discussion and Result Evaluation (DRE)

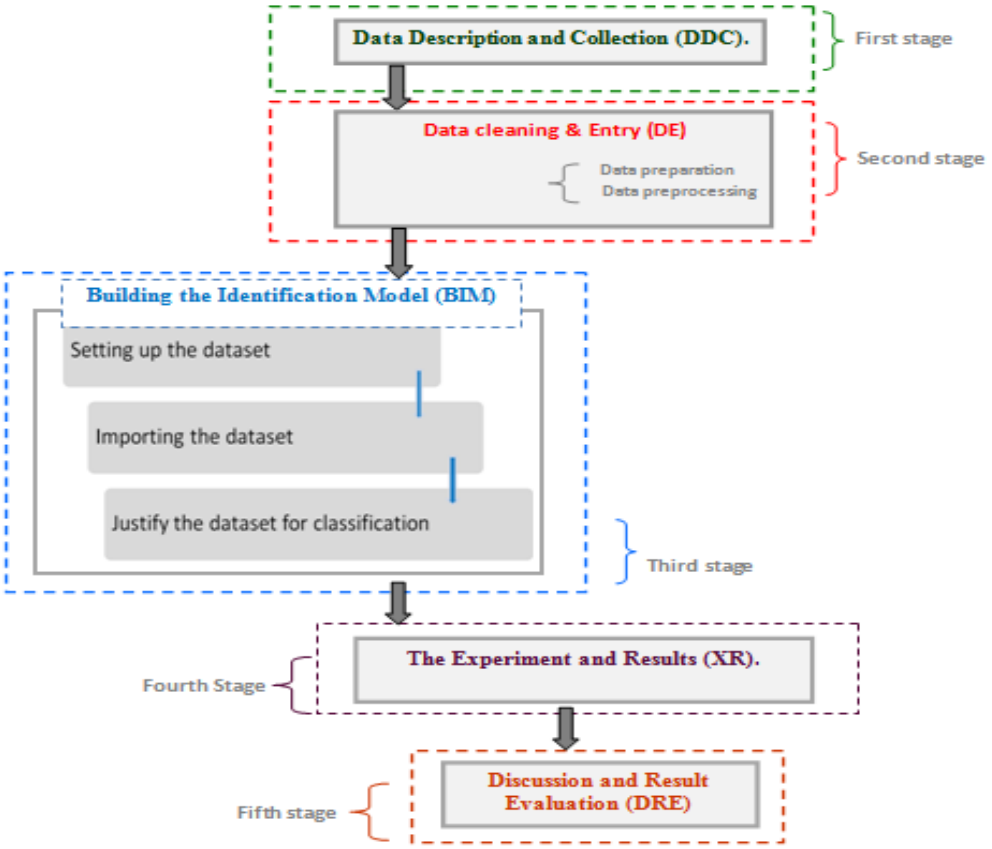


Figure 3: The model structure

2.1 Data Description and Collection (DDC)

This stage starts with listing the essential smart home devices and creates an acronym for each, as shown in Table 1. Next, it adds some features that relate to all of the devices and provides a clear description for each one to help in understanding the meaning of each one as illustrated in Table 2. Furthermore, each device's information is gathered from various online sources such as manufacturer websites. An example of device data are presented in Figure 4. This stage is the model's first stage and is considered a base for the rest.

Table 1: List of essential smart home devices and their acronyms

Number	Device Name	The acronym
1.	Camera	SCam
2.	Smart refrigerator.	KRef
3.	Smart air fryer.	KAfr
4.	Smart microwave.	KMwave
5.	Smart coffee maker.	KCmr
6.	Smart oven.	KOvn
7.	Smart pressure cooker.	KPrc
8.	Smart sous vide.	KSvd
9.	Smart electric kettle.	KKtl
10.	Smart dishwasher.	KDwsh
11.	Smart door bell.	ODbl
12.	Smart lock.	OLoc
13.	Smart garden light.	OGlt
14.	Garage door automation.	OGdr
15.	Smart Air conditioner.	IAcr
16.	Smart Air purifier.	IApr
17.	Smart lights.	ILgt
18.	Smart washing machine	IWmh
19.	Smart switch.	ISch
20.	Smart vacuum.	IVcm
21.	Smart TV.	ETV
22.	Smart speakers.	ESpk

Table 2: Data description

The feature	The description
Device Name	The name and type of the device (Ex. Smart Tv, Smartphone, etc..)
Model number/MPN	The product number, usually a combination of numbers and letters.
Manufacturer	The company that makes the device (ex. LG, Samsung, etc..)
Size/ capacity	Is the dimension or the load capacity of a particular device .
OS type	The Operating system of the device (ex. Android, Ios, etc.)
Operating voltage	The amount of power required to operate the device.
Connectivity	The Internet connectivity method (Ex. WIFI, Bluetooth, etc..)

evice type	Model number/MPN	Manufacturer	Size/ capacity	OS type	operating voltage	connectivi
Smart plug	DSP1100	Wipro	10 × 5 cm	androuid	212 v	Wifi
Camera	LS7976R	Laser	16 Mp	androuid	213 v	Wifi
Camera	NI1732	National Insrument	8 Mp	PowerPC	214 v	Wifi
Camera	NI1712	National Insrument	18 Mp	PowerPC	215 v	Wifi
Camera	FI9831EP	Foscam	8 Mp	androuid	216 v	Wifi
Camera	FI9821EP	Foscam	19 Mp	androuid	217 v	Wifi
Camera	FI9818W	Foscam	20 Mp	androuid	218 v	Wifi
Smart Switch	SHM6993	Toyama	24 × 10 × 5 cm	androuid	219 v	Blutooth
Smart refrigerator.	LFXS26973	LG	26.20 feet	Ios	220 v	Wifi
Smart refrigerator.	LFXS26596S	LG	26 feet	Ios	220 v	Wifi
Smart refrigerator.	RS27T5561SR	Sumsung	26 feet	androuid	220 v	Wifi
Smart refrigerator.	IVFWCC281LBW	Ivation	5.6 feet	androuid	220 v	wifi
Smart refrigerator.	HRF266N6CSE	Hisense	26.6 feet	androuid	220 v	Wifi
Smart air fryer.	CS100-AO	Cosori	2 liter	Ios	220 v	wifi
Smart air fryer.	CS158-AF	Cosori	5 litter	Ios	220 v	wifi
Smart microwave.	MS4295DIS	LG	42 Liter	androuid	220 v	wifi
Smart microwave.	CEB515M2NS5	Café	1.5 Cu. Ft.	androuid	120 v	wifi
Smart microwave.	JES1097SMSS	GE	0.9 Cu. Ft	androuid	220 v	wifi
nart microwave.	COS-3016ORM1SS	Cosmo	1.6 cu.ft	androuid	220 v	wifi
t microwave.	NN-SN97JS	Panasonic	2.2 cu.ft	androuid	220 v	

A

icrowave.	NN-SN97JS	Panasonic	2.2 cu.ft	androuid	220 v	
art Electric Cooker	LREL6323D	LG		androuid	220 v	Wifi
mart coffe and espresso machine.	EP3246/73	PHILIPS	1.8 liter	Ios	120 v	Wifi
Smart coffee maker.	ECAM 350.55.B	De'Longhi	1.8 liter	Ios	220 v	Wifi
Smart coffee maker.	TE651209GB	Siemens	2.0 liter	Ios	220 v	Wifi
Smart coffee maker.	SMC01	Smarter	1.5 liter	Ios	220 v	Wifi
Smart wall oven.	LWC3063BD	LG	1.7 cu.ft.	Ios	220 v	Wifi
Smart oven.	MC368GAAW5A	Sumsung		androuid	220 v	Wifi
Smart oven.	Gen22	Tovala		androuid	220 v	Wifi
Smart oven.	PTS9000SNSS	GE		Ios	220 v	Wifi
Smart pressure cooker.	M92S-E	REDMOND	5 litter	ios 9.0	220 v	Blutooth
Smart pressure cooker.	112012401	Instant Pot	6 Quarts	Ios	220 v	Wifi
Smart pressure cooker.	PP52878CO	CHEF iQ	6 Quarts	Ios	220 v	Wifi
Smart sous vide.	SV985675S	Inkbird	10 liter	androuid	120 v	wifi
Smart sous vide.	A3.2120VUS	Anova	15 liter	androuid	110 v	Wifi
Smart sous vide.	CS10001	Breville	10 liter	androuid	120 v	Wifi
Smart sous vide.	GSV150B	Gourmia	20 liter	androuid	120 v	Wifi
Smart electric kettle.	G200S-E	REDMOND	2 liter	androuid	220 v	Blutooth
Smart electric kettle.	RK-M216S-E	Strix	1.7 liter	androuid	220 v	wifi
Smart electric kettle.	AK000001	Appkettle	1.7 liter	androuid	220 v	wifi
Smart electric kettle.	SK8579322	Appkettle	1.6 liter	androuid	220 v	Wifi
mart electric kettle.	SMKET01-WGUK	Smarter	1.8 liter	androuid	220 v	Wifi
lwasher	SN23EC14CG	Siemens	7 liter	androuid	220 v	Wifi

B

Figure 4: A & B are examples of smart devices data

2.2 Data cleaning and Entry (DE)

At this point, all the necessary data about the smart home devices had been collected. To create a model, data is prepared in a format appropriate for the software tool. At this stage, Weka was selected, and the data was prepared in Weka format. As part of this section, the data is converted into a suitable format and cleaned up to meet the requirements of the model.

2.2.1 The transformation of devices data into ARFF format

In this section the smart home devices data gathered from online sources has been transformed into ARFF format as illustrated in Figure 5, through the following steps:

- Open a text file, use the ARFF standard format to specify relation name and attributes, and write the data.
- Write all the features represented in Table 2, by making everyone represent as an attribute of string, numeric, or nominal type.
- Fill each device's data on the data section in sequential order based on the order of attributes in the attributes section, one by one in separate lines.
- Finally save the text file with ARFF extension.

```
@relation SmartDevices
@attribute "Device type"    {SCam,KRef,KAfr,KMwave,KKtl,KCmr,KOvn,KPrc,KSvd,KDwsh,ODbl,OLoc,OGlt,OGdr,IAcr,IApr,ILgt,IWmh,ISch,IVcm,ETV,ESpk}
@attribute "Model Nnmber"   string
@attribute Manufacturer     string
@attribute "cap-value"        numeric
@attribute "cap-unit"       {ft,lt,Kg,ih}
@attribute "OS type"       {Androuid,Ios,PowerPC}
@attribute "operating voltage"   numeric
@attribute connectivity      {Wifi,Blutooth,Zigbee}
@attribute class   {Surveillance,Kitchen,Indoor,Outdoor,Entertainment}

@data
SCam,LS7976R, Laser, ?,?, Androuid,12,Wifi,Surveillance
SCam,NI1732,"National Insrument",?,?,PowerPC,12, Wifi,Surveillance
SCam,NI1712,"National Insrument",?,?,PowerPC,12, Wifi,Surveillance
SCam,FI9831EP,FosSCam,?,?,Androuid,12, Wifi,Surveillance
SCam,FI9821EP,FosSCam,?,?,Androuid,12, Wifi,Surveillance
SCam,FI9818W,FosSCam,?,?,Androuid,12, Wifi,Surveillance
SCam,C3WN,Generic,?,?,Androuid,12,Wifi,Surveillance
KRef,RS27T5561SR,Sumsung,26,ft,Androuid,220,Wifi,Kitchen
KRef,LFXS26973,LG,26, ft,Ios,220,Wifi,Kitchen
KRef,LFXS26596S,LG,26, ft,Ios,220,Wifi,Kitchen
KRef,RS27T5561SR,Sumsung,26,ft,Androuid,220,Wifi,Kitchen
KRef,IVFWCC281LBW,Ivation,5.6, ft,Androuid,220,Wifi,Kitchen
KRef,HRF266N6CSE,Hisense,26, ft,Androuid,220,Wifi,Kitchen
KAfr,CS100-AO,Cosori,2,lt,Ios,220,Wifi,Kitchen
KAfr,CS158-AF,Cosori,5, lt,Ios,220,Wifi,Kitchen
KMwave,MS4295DIS,LG, 42,lt,Androuid,220, Wifi,Kitchen
KMwave,JES1097SMSS,GE,0.9,ft,Androuid,220,Wifi,Kitchen
KMwave,COS-3016ORM1SS,Cosmo,1.6,ft,Androuid,220,Wifi,Kitchen
```

Figure 5: Part of smart home devices dataset in arff format

2.2.2 Introduction to Weka

The Waikato Environment for Knowledge Analysis (WEKA) (Hall et al. 2008), is a platform that contains a collection of machine learning algorithms and data preprocessing tools; it allows users to apply machine learning methods to new datasets quickly and easily. The Weka workspace includes methods for solving all types of data mining problems: regression, classification, clustering, association rule mining, and attribute selection (Frank E, et al. 2010). In Weka, a variety of data sources can be loaded, including files, URLs, and databases. File formats supported include Weka's own ARFF format, CSV, and C4.5's format. It is also possible to generate data using an artificial data source and edit data manually using a dataset editor (Hall M, et al. 2008). The Weka home page is shown in Figure 6.

Figure 6: Screen shot of Weka version 3.9.5 home page

2.2.3 Explanation of smart home devices dataset

The ARFF file structure is represented in Figure 5, includes the following:

- A relation called "SmartDevices".
- Nine attributes that represent the device features are represented in Table 2, and some of them are further broken down into two attributes, as follows:
- The first attribute: Device type, which defines a nominal that contains a list of all device acronyms shown in Table 1, as values.
- The second attribute: Model number, is of type string and contains the device model number.
- The third attribute: Manufacture is of type string and contains the brand of the device.
- The fourth and fifth attributes represent the size/capacity feature that has been broken into two attributes to avoid string value by making it cap_value of numeric value and cap_unit of nominal value.
- The sixth attribute: Os type is a nominal type containing Android, iOS, and PowerPc values.
- The seventh attribute is operating voltage, which contains the power required for operating the device; it is defined as numeric type and contains only the voltage value without adding v.
- The eighth attribute: Connectivity represents the device's connectivity method and has been defined as a nominal type that contains Wifi, Bluetooth, and Zigbee as values.
- The ninth attribute: Class which is not represented in Table 2, and has been added to make the model classify the device according to it. It is defined as a nominal type and contains five values Surveillance, Kitchen, Indoor, Outdoor, and Entertainment. Figure 7 shows a distribution of smart home devices according to these five classes.
- Each device's data is inserted sequentially based on the order of the attributes.

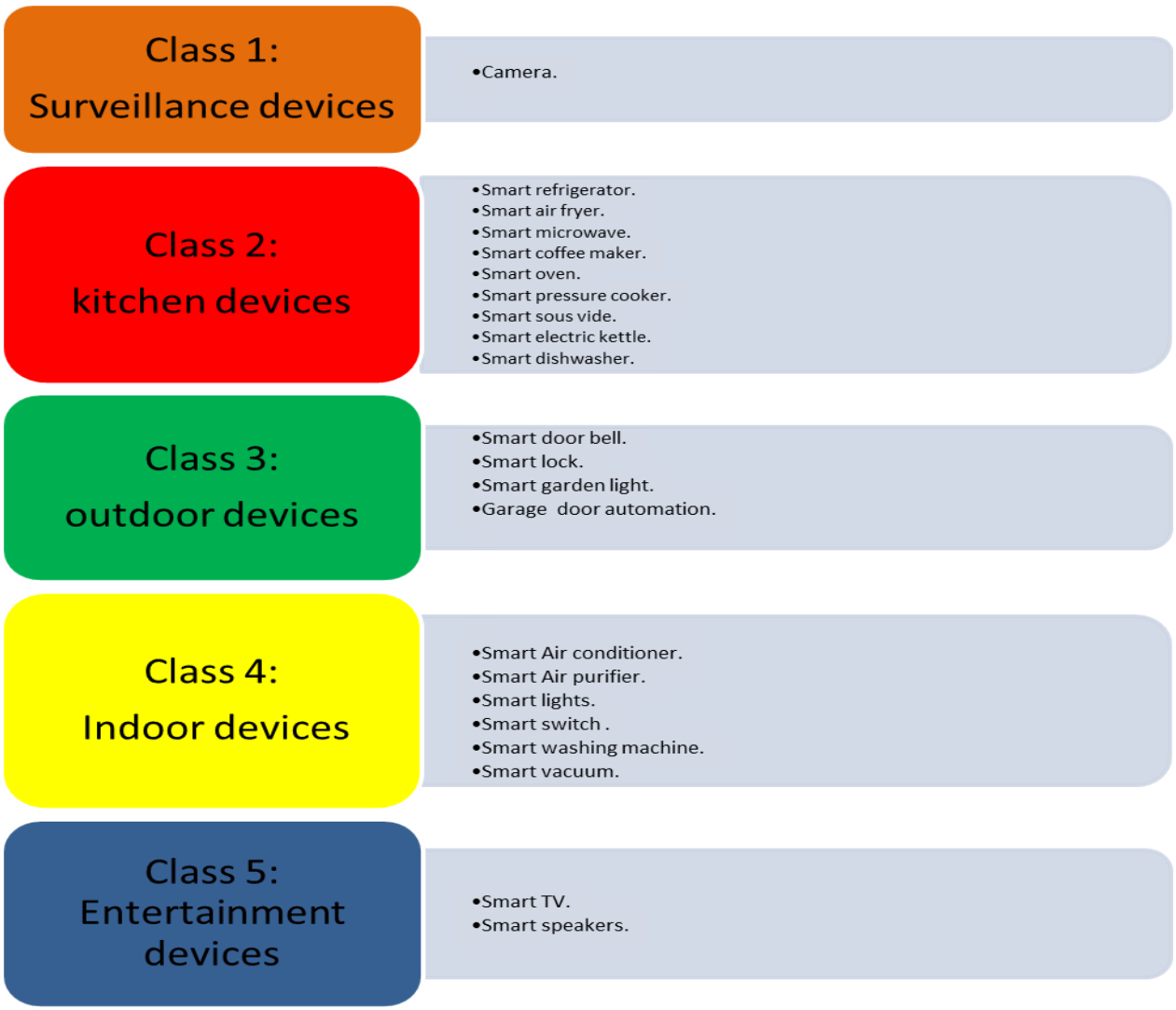


Figure 7: The five classes of smart home devices

2.3 Building the Identification Model (BIM)

This stage aims to leverage the foundation of Deep Neural Network (DNN) concept presented in the first section, to build a model for identifying a set of smart home devices. It is the most important step in constructing the model, and all the steps following it depend on it. This stage will never be completed without the following:

1) Setting up the dataset.
2) Importing the data into Weka.
3) Making sure the dataset is suitable for classification using a Multi-layer Perceptron network.

All the above three steps must be implemented accurately to avoid any problems in the experiment step.

2.3.1 Preparation for Building the Multi-layer Perceptron Network (MLP) using Weka

Here the smart home devices dataset applied to the arff file in Figure 5 has been loaded into Weka as a preparation step, so it can be classified using a Multi-layer Perceptron network, as shown in the following Figures.

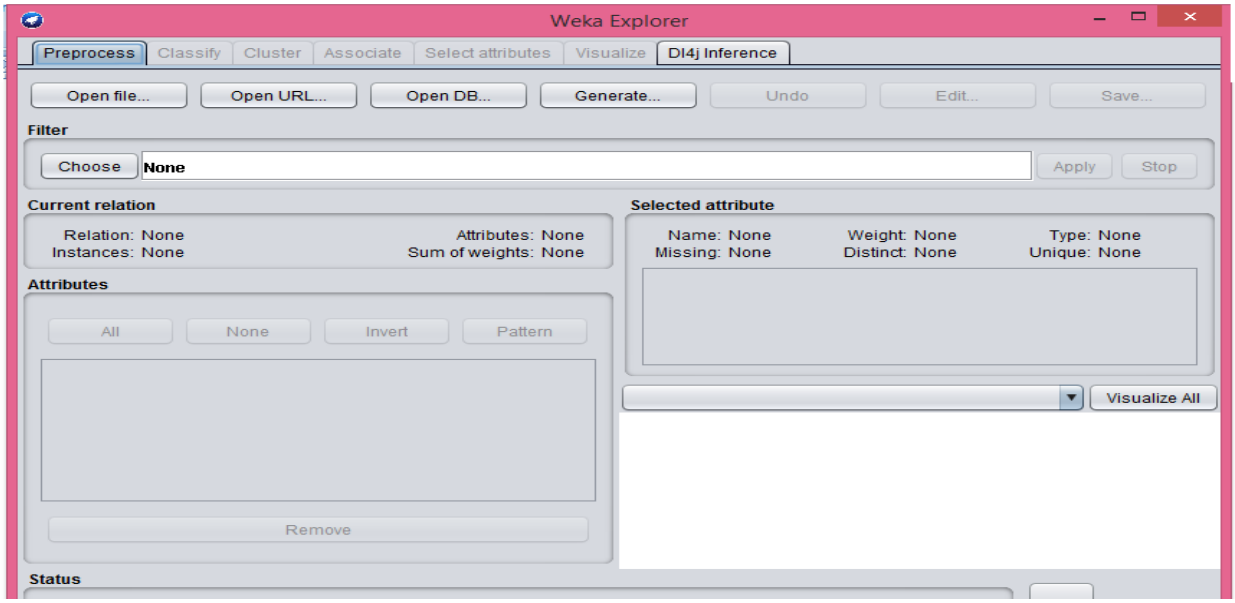


Figure 8: The first step of loading the dataset

The process of loading the arff file into Weka goes through two main steps, the first one is starting the Weka Explorer as shown in Figure 8. In the second step, opening the file from "Open file" button and shows its content in the Explorer window as shown in Figure 9.

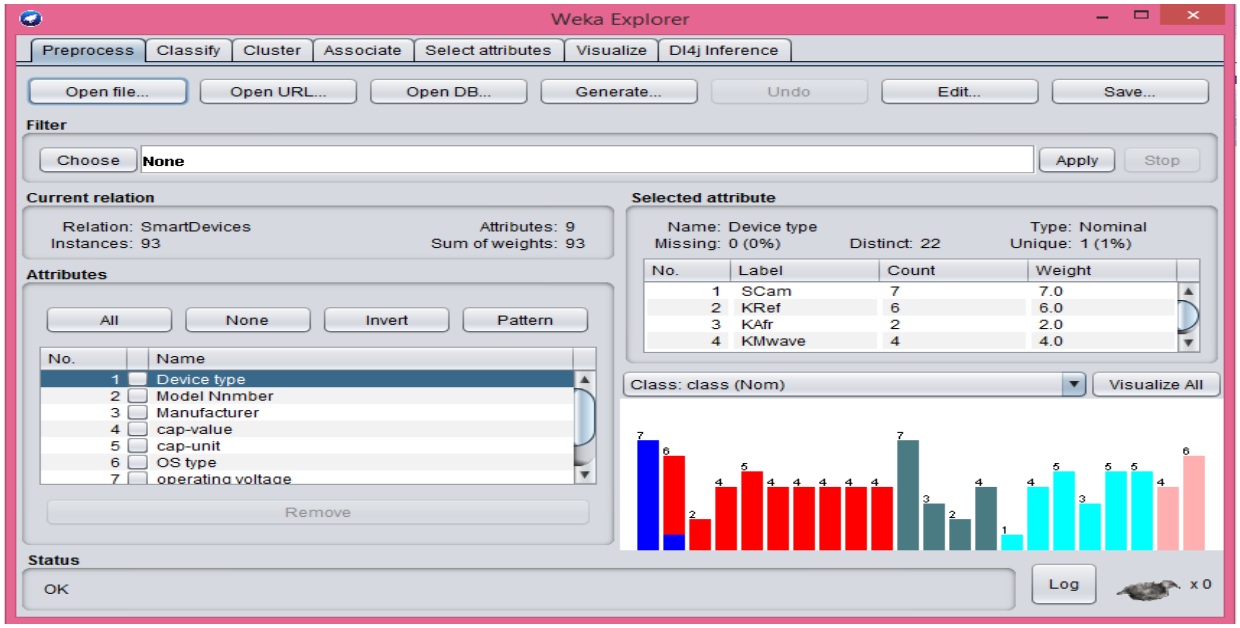
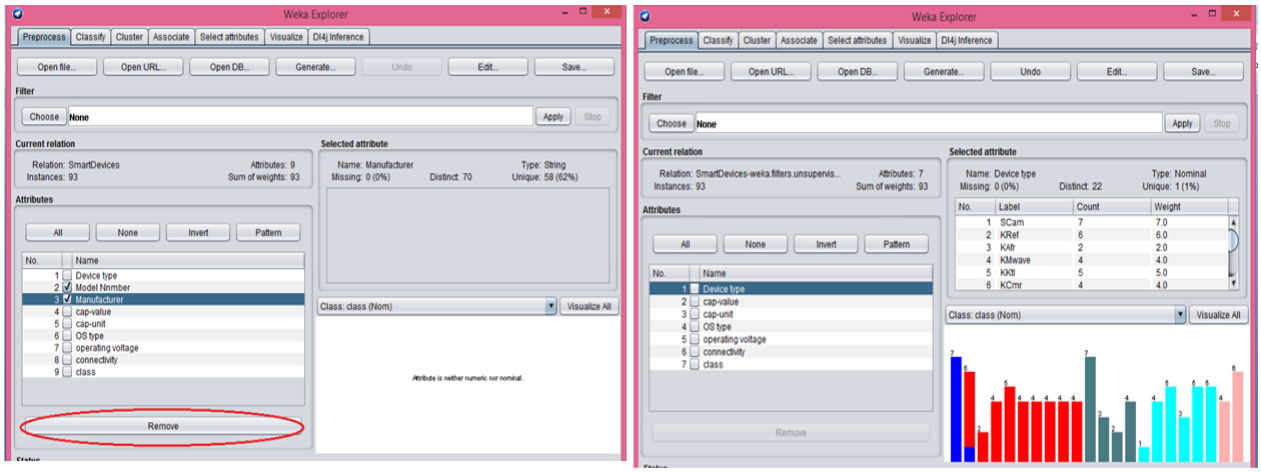


Figure 9: The second step of loading the dataset

As shown in Figure 9, the dataset has 93 instances and nine attributes. Attributes are varied in type; some are string, numeric, and nominal.

Before starting the classification process, it is necessary to address the problem that Multi-layer Perceptron networks cannot handle string attributes. It can only handle attributes of either numeric or nominal type. This problem must be solved by removing

all attributes of type string at this stage. So that the Model number and Manufacturer will be removed directly from the explorer window, by ticking the attributes and pressing the remove button as depicted in Figure 10.



A B

Figure 10: (A) Demonstrates the attributes being removed, (B) demonstrates the seven remaining attributes after Model number & Manufacturer are removed.

2.4 The Experiment and Results (XR)

At this stage, the data is ready for classification using Multi-layer Perceptron, which is trained until the optimal classification result is achieved.

For more concreteness, assume that the dataset consists of m devices d denoted by $D := \{d_1, \ldots\ldots\ldots, d_m\}$, and assume classes S_i denoted $S := \{s_1, \ldots\ldots\ldots\ldots, s_i\}$, where the classes satisfy $S_i \in \{class_1, class_2, class_3, class_4, class_5\}$, each class represents one category of smart home devices.

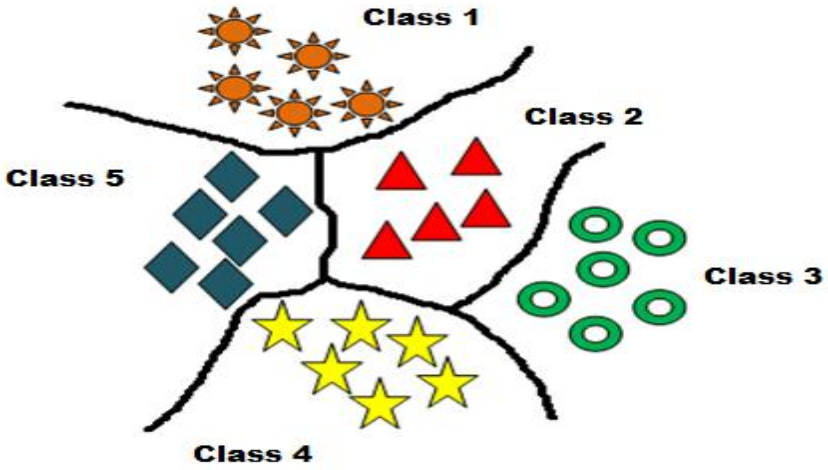


Figure 11: The classification.

Note that the classifier must be able to distinguish each shape and classify it to the correct class.

In Figure 11, each class is represented by a distinct shape, such as surveillance by suns, kitchen by triangles, outdoor by circles, indoor by stars, and entertainment by diamonds. The classifier must classify each instance into its appropriate class, as illustrated by shapes in Figure 11. The classification process using Multi-layer Perceptron and analysis of the results of different test options is the main goal of this stage.

According to Figure 11, each instance should be classified into its appropriate class. This model of classification uses Multi-layer Perceptron and analyzing the results of different test options is the primary goal of this stage. A classification model is constructed by going through the following steps:

1) Training.
2) Testing.
3) Validation.

In each of the above three steps, accuracy, errors, and sensitivity must be evaluated. The training and testing will be carried out at this stage. Validation, which consists of assessing the model based on different methods and applying a larger dataset, will be carried out in the next stage.

Basically, Weka offers four test options; to test and train the data. Weka uses these options to determine how to proceed with the dataset. The classifier must run on one of them at a time. The four test options are:

- Use training set: That is, test the classifier on the same data set it was trained on.
- Supplied test set: That tests on user-specific datasets. It is an external file that can be used for training. This is useful when testing an algorithm's knowledge against a particular test set.
- Cross validation: Performs an n-fold cross validation. It works like many percentage splits. If you fold the data into 10 folds (for example), then you repeat the process 10 times (because it is 10-folds) as follows: 9 folds are for training, 1-fold is for testing. Each time leaving a different fold for testing. This method is most common in many studies.
- Percentage split: Splits the data and separates n% of it. Then train on percentage of the data, and test on the reminder.

In this model MLP is applied by using cross validation and percentage split, for training and testing a dataset of 93 instances. The results are summarized in the form of tables and a confusion matrix.

Each of the tables shows the testing results in numbers and percentages, it contains the total number of instances, number of correctly classified instances, number of incorrectly classified instances, percentage of correctly classified instances, percentage of incorrectly classified instances, kappa statistic, mean absolute error, root mean squared error, and relative absolute error. However, the confusion matrix is a special matrix used in machine learning to visualize the classification algorithm's performance, typically with supervised learning. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class, or vice versa. The confusion matrix is also known as an error matrix.

This model shows the number of instance distribution on each class, where a represents Surveillance, b represents Kitchen, c represents Indoor, d represents Outdoor, and e represents Entertainment.

2.4.1 MLP with percentage split test option:

2.4.1.1 66% percentage split:

Here the dataset is split as 66% for training, remainder for testing. The summary of the results is shown in Table 3.

Table 3: The Summary of MLP Results with 66% percentage split

Total Number of Instances	Correctly Classified Instances	Percentage of Correctly Classified Instances	Incorrectly Classified Instances	Percentage of Incorrectly Classified Instances	Kappa statistic	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
32	32	100%	0	0%	1	0.0276	0.0784	9.2038%	20.2371%

The above table shows the results of a 66% percentage split for the dataset that contains 93 instances, which means 61 instances are for training and 32 are for testing. The results show that all the instances are correctly classified. In the confusion matrix below, the 32 instances are distributed among the five classes as follows: three instances are classified as "a", 13 instances as "b", eight instances as "c", another three instances are classified as "d", and five instances as "e".

```
a  b  c  d  e   <-- classified as
3  0  0  0  0   | a = Surveillance
0 13  0  0  0   | b = Kitchen
0  0  8  0  0   | c = Indoor
0  0  0  3  0   | d = Outdoor
0  0  0  0  5   | e = Entertainment
```

2.4.1.2 50% percentage split

Table 4: The results summary of MLP with 50% percentage split

Total Number of Instances	Correctly Classified Instances	Percentage of Correctly Classified Instances	Incorrectly Classified Instances	Percentage of Incorrectly Classified Instances	Kappa statistic	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
46	37	80.4348%	9	19.5652%	0.7383	0.0976	0.2468	32.5834%	63.1436%

Table 4 shows classification results after splitting the dataset equally as 50% for training and for testing. The results show that total numbers of instances are 46, 37 of them are correctly classified and nine are incorrectly classified representing 80.4% and 19.6% respectively.

The following confusion matrix shows the number of instance distribution to each class.

```
a  b  c  d  e   <-- classified as
4  0  0  0  0   | a = Surveillance
0 17  0  0  0   | b = Kitchen
0  0  5  5  0   | c = Indoor
0  0  0  8  0   | d = Outdoor
0  2  2  0  3   | e = Entertainment
```

The confusion matrix above shows that the distribution of the 46 instances among the five classes is: four instances are classified as "a", 17 instances as "b", five instances as "c", eight are classified as "d", and three instances as "e". That completes the 37 instances which are classified correctly. This matrix also shows the distribution of the nine instances that are incorrectly classified as follows: Five instances are unclear whether they belong to class "c" or "d", two instances are unknown whether they belong to class "e" or "c", and two instances are unknown whether they belong to class "e" or "b".

2.4.2 MLP with Cross-validation test option:

Here the dataset is applied to MLP with 10 and 5 folds Cross-validation. The summary of results is shown in Table 5. During cross-validation, the dataset isdivided into n subsets (also known as folds). Initially, the model is trained on all but one (n -1) of the subsets, and then it will be evaluated on the remaining subset.

Table 5: The results summary of MLP with 10 & 5 folds Cross-validation

The number of folds	Total Number of Instances	Correctly Classified Instances	Percentage of Correctly Classified Instances	Incorrectly Classified Instances	Percentage of Incorrectly Classified Instances	Kappa statistic	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
10 Folds	93	90	96.7742%	3	3.2258%	0.9565	0.0452	0.1209	15.1567%	31.3784%
5 Folds		88	94.6237%	5	5.3763%	0.9277	0.029	0.1276	9.7282%	33.1169%

The table above represents the results after dividing the dataset into 10 & 5 folds. The results in both cases are so close; it produces 90 correctly classified instances in the first and 88 instances in the second cases, in percentages equal to 96.7% & 94.6% respectively . The number of incorrectly classified instances is only three on 10 folds with percentage equal to 3.2%, and five incorrectly classified instances equal to 5.3% in the case of using 5 folds.

The confusion matrices of the above results EW shown below:

1- The confusion matrix of applying 10 folds

```
 a  b  c  d  e   <-- classified as
 7  1  0  0  0   | a = Surveillance
 0 35  0  0  1   | b = Kitchen
 0  0 22  0  1   | c = Indoor
 0  0  0 16  0   | d = Outdoor
 0  0  0  0 10   | e = Entertainment
```

2- The confusion matrix of applying 5 folds

```
 a  b  c  d  e   <-- classified as
 7  1  0  0  0   | a = Surveillance
 2 33  1  0  0   | b = Kitchen
 0  1 22  0  1   | c = Indoor
 0  0  0 16  0   | d = Outdoor
 0  0  0  0 10   | e = Entertainment
```

The first row in both matrices shows that seven instances are classified as "a", with confusion in one instance if it belongs to class "a" or "b". In the second row in the ten folds matrix, 35 elements are classified as "b" and 33 in the case of applying five folds. The confusion appears in one instance as if it belongs to class "b" or "e" in the first case, but in the second matrix, confusion is in three instances as if they belong to class "b" or "a" or "c". The third row has the same number of correctly classified instances in both matrices but with slice confusion difference which is one instance in first case and two instances in the second case. The fourth and fifth row are identical in the two cases, with 16 instances correctly classified as class "d" and ten instances classified as "e". The total number of incorrection classification is three instances in the 10 folds case and five instances for the 5 folds.

2.4.3 Run MLP with 66% percentage split, and 10 folds cross validation, with adding hidden layers:

The best results from the above tables were taken to be extended by adding hidden layers. The best results were 66% percentage split and ten folds cross-validation.

The following table presents the results of using two random values for the additional hidden layers. The first part of Table 6, contains the results of adding two layers, the results show that 29 and 90 instances are correctly classified. Respectively for 66% percentage split and ten folds cross-validation. The number incorrectly classified is equal, which are three instances. The percentages of correctly classified instances are 90.6% and 96.7. The percentage of incorrectly classified instances are 9.3% and 3.2%.

The second part of the table contains the results of adding three layers; the results show that 23 and 84 instances are correctly classified, respectively for 66% percentage split and ten folds cross-validation. The number incorrectly classified is equal in both

training tests, which are nine instances. The percentages of correctly classified instances are 71.8% and 90.3. The percentages of incorrectly classified instances are 28.1% and 9.6%.

Table 6: Investigation of Results after adding extra layers to both models:

The model type	Two hidden layers (10,10)					Three hidden layers (5,6,4)				
	Total Number of Instances	Correctly Classified Instances	Percentage of Correctly Classified Instances	Incorrectly Classified Instances	Percentage of Incorrectly Classified Instances	Total Number of Instances	Correctly Classified Instances	Percentage of Correctly Classified Instances	Incorrectly Classified Instances	Percentage of Incorrectly Classified Instances
66% percentage split	32	29	90.625 %	3	9.375%	32	23	71.875%	9	28.125%
10 folds cross validation	93	90	96.7742%	3	3.2258%	93	84	90.3226%	9	9.6774%

When comparing the results in Tables 3 and 5 with the results in Table 6 it was found that applying MLP using 66% percentage split and ten folds cross-validation without adding hidden layers gives more accurate results and better performance.

2.5 Discussion and Result Evaluation (DRE)

This stage concerns assessing the classification model based on different methods and applying a larger dataset. This stage is necessary to ensure that the model is working properly.

2.5.1 Applying Deep learning 4j (Dl4j) classifier

In this section (Dl4j) is applied on the same dataset to investigate its performance compared with MLP. The results are shown in the Table 7:

Table 7: The results summary of applying Dl4j with both models

The model type	Total Number of Instances	Correctly Classified Instances	Percentage of Correctly Classified Instances	Incorrectly Classified Instances	Percentage of Incorrectly Classified Instances	Kappa statistic	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
66% percentage split	32	22	68.75%	10	31.25%	0.5913	0.1947	0.2914	64.9799%	75.2088%
10 folds cross validation	93	69	74.1935%	24	25.8065%	0.6554	0.1645	0.2577	55.2273%	66.8589%

In Table 7, the best results from the above Tables 3 and 5 had also taken to be applied with Deep learning 4j (Dl4j), the results show that, of the total number of 32 instances in the case of 66% percentage split option, only 22 instances had correctly classified which 68.7%, and ten instances are incorrectly classified equal to 31.2%. In the second case using 10 folds cross validation, produces 69 correctly classified instances, which 74.1%, and 24 instances are incorrectly classified equal to 25.8%. However, applying (Dl4j) with tenfold cross-validation was better than using it with a 66% percentage split, but applying MLP was better in both cases, that when applying Dl4j with both models 66% percentage split and 10 folds cross-validation, gives less accuracy compared with MLP.

Table 8: The confusion matrix of applying Dl4j with the both models

66% percentage split	10 folds cross validation
a b c d e <-- classified as	a b c d e <-- classified as
3 0 0 0 0 | a = Surveillance	7 1 0 0 0 | a = Surveillance
0 9 3 1 0 | b = Kitchen	0 31 0 4 1 | b = Kitchen
0 0 4 4 0 | c = Indoor	0 3 11 9 0 | c = Indoor
0 0 2 1 0 | d = Outdoor	0 0 5 10 1 | d = Outdoor
0 0 0 0 5 | e = Entertainment	0 0 0 0 10 | e = Entertainment

Based on the results in Table 7, the total number of correctly classified instances is 22; in the case of using a 66% percentage split, are distributed among the five classes according to confusion matrix in Table 8 as follows: three as "a" , nine as "b" , four as "c", one as "d", and five as "e". The ten incorrectly classified instances appear as confusion in three instances if it belongs to class "c" or "b", one instance if it belongs to class "d" or "b", four instances if it belongs to class "d" or "c", and two belongs to class "d" or "c".

In the case of applying (Dl4j) with ten folds cross-validation, there were 69 correctly classified instances that had distributed among the five classes represented in the second column of the confusion matrix in the above table as seven as "a", 31 as "b" , 11 as "c", ten as either "d" and "e". Moreover, there is confusion in 24 instances that appear in the above confusion matrix as follows: confusion in one instance if it belongs to class "b" or "a", four instances if it belongs to class "d" or "b", one instance if it belongs to class "e" or "b", three instances if it belongs to class "b" or "c", nine instances if it belongs to class "d" or "c", five instances if it belongs to class "c" or "d", and one instance if it belongs to class "e" or "d".

2.5.2 Applying MLP on dataset with further instances

To provide a more precise validation of using MLP for identifying smart objects, it is essential to ensure its efficiency by applying it to additional instances in the dataset. This study applied MLP to a dataset consisting of 130 instances. The results of using the two models, namely the 66% percentage split and 10-fold cross-validation, are discussed in Table 9.

Table 9: The results summary of applying MLP with both models over 130 instances of the dataset

The model type	Total Number of Instances	Correctly Classified Instances	Percentage of Correctly Classified Instances	Incorrectly Classified Instances	Percentage of Incorrectly Classified Instances	Kappa statistic	Mean absolute error	Root mean squared error	Relative absolute error	Root relative squared error
66% percentage split	44	42	95.4545%	2	4.5455%	0.9413	0.0275	0.1058	9.0087%	26.5176%
10 folds cross validation	130	129	99.2308%	1	0.7692%	0.9897	0.0104	0.0451	3.4758%	11.6585%

In Table 9, MLP with 66% percentage split and ten folds cross-validation have been applied to a dataset with 130 instances. In the first row of the table, we see that the total number of instances tested is 44; 42 were classified correctly with a percentage equal to 95.4%, and two cases were classified incorrectly with a percentage equal to 4.5 %. In the second row, we show the results of ten-fold cross-validation with 130 examples, producing 129 correctly classified instances, or 99.2%, and only one incorrectly classified instance, or 0.7%.

RESULT

The dataset contains over 90 instances of smart home appliances, collected from online sources and prepared in ARFF format to be compatible with the Weka platform, as shown in Figure 4. The dataset was classified using both MLP and Deep learning 4j (Dl4j) classifiers. The final results of both classifiers were compared to determine the best outcome.

Weka had been chosen for the implementation of this model for some reasons:

- Weka is easy to learn and use.
- Weka is a very rich platform; it has various classification tools that can be applied to different studies.
- Weka is so powerful; it can produce accurate results even if the dataset contains missing values.

In this study, MLP was applied first to a dataset containing 93 instances and tested upon different test options to see what results could be produced each time. The results are analyzed below:

1) MLP with percentage split test option and no hidden layers added:

When the percentage split is 66%, which means 66% of instances of the dataset have been used for training and the remainder is for testing, the result shows that the total number of tested cases is 32. All of them are correctly classified, so the result in percentage is 100% correctly classified. These results are shown in Table 3.

The same experiment is repeated to the same dataset with percentage split 50%. The result shows that the total number of tested instances is 46, 37 of them are correctly classified, which is equal to 80.5%, and only 9 cases are incorrectly classified that approximately 19% of instances so, as shown in Table 4.

According to both results, 100% of the instances have been correctly classified or very close to it. Therefore, the number of instances that are correctly classified is very satisfying.

2) MLP with Cross-validation test option and no hidden layers added:

MLP with Cross-validation of 10 and 5 folds is applied on the same dataset of 93 instances. The results of this experiment is very amazing, in that, in both cases, a large number of instances are correctly classified. According to Table 5 the percentage of correctly classified instances reached 96% in the case of 10 folds, and 94% in the case 5 folds. This proves that the numbers of correctly classified instances in both cases are so close, so it clearly indicates that this model is working properly.

According to the above results, when comparing the results obtained from MLP with the Cross-validation test option with the results of MLP with the percentage split test option, the result of using of MLP with the Cross-validation test option with any fold value is close to the result of using percentage split is 66%, but overcome the result gets of using percentage split is 50%.

3) Applying MLP with 66% percentage split, and 10 folds cross-validation with adding hidden layers:
At this point the best methods from the previous points has been chosen to apply additional hidden layers. The results prove that applying MLP using both models without adding hidden layers gives more accurate results and better performance, as shown in Table 6.

4) Applying Deep learning 4j (Dl4j) classifier:
Applying Deep learning 4j (Dl4j) classifier on the dataset of 93 instances with 66% percentage split, and 10 folds cross validation gives low accuracy when compared with the performance of a baseline model (MLP) using a dataset containing 93 instances, in which the percentage of correctly classified instance doesn't exceed 75%. The result details and confusion matrix are illustrated in Tables 7 and 8.

5) Applying MLP on dataset with further instances:
In order to assess the effectiveness of the proposed model, its performance has been compared using a dataset containing 130 instances with 66% percentage split and 10-fold cross-validation to evaluate the models. The results in Table 9 demonstrate that the proposed model significantly outperforms the baseline model on this dataset. However, it is important to note that the dataset's size and representativeness may influence the model's generalizability. Therefore, testing the model on a larger and more diverse dataset would be beneficial to ensure its effectiveness in a real-world setting.

In general, it is important to consider both the size and quality of the dataset when evaluating the performance of a machine learning model. A large dataset may improve performance, but ensuring that the data represents the real-world problem being solved is important to ensure the model's generalizability (Hastie et al., 2009; Foster, 2003; Wang et al., 2016).

Machine learning models improve with larger datasets because they provide more information for the model to learn from (Dietterich, 2002; Vapnik, 1995). However, there is a point at which the model's performance will plateau and the benefits of a larger dataset will diminish (Foster, 2003). There are also practical considerations when working with large datasets, such as the time and resources required for training and running them. It is difficult to determine when the MLP method becomes inefficient concerning the size of the dataset, as this will depend on the specific problem and available resources (Bengio et al., 2012; Goodfellow et al., 2010).

DISCUSSION

The widespread adoption of IoT devices has led to the development of smart homes, which allow homeowners to remotely control their appliances, lighting, heating, security systems and electronics using a smartphone or another networked device, which has led to a fundamental shift in how we live. Intelligent homes are among the most complex and data-rich IoT environments, making them suitable for developing and testing machine learning models.

Deep Learning (DL) methods, which involve using artificial neural networks with many hidden layers to process and learn from large datasets, are well-suited for handling the large amounts of data generated by IoT systems. DL has been successfully applied to various tasks, including intelligent video analytics, hyperspectral imagery analysis, image recognition, natural language processing, automatic e-mail responses, machine translation, fraud detection, and healthcare.

DNN is used for classification, and classification is considered an identification method in which each classified item must belong to a particular category. From this perspective, Multi-layer Perceptron has been used in this study to identify smart devices within the context of smart homes. The model uses five classes to categorize each device and separates it from the rest by identifying each one as belonging to one of them. The classes considered are the main categories that smart home devices belong. The dataset mainly contains 93 instances with basic 9 attributes to each and applied to MLP with only 7 attributes. The results after investigating MLP with 66% split test option, 10-fold cross-validation, and adding hidden layers to the models. The results are varied fomr one model to another, but MLP with a 66% split gives 100% accuracy, in which the full number of instances has been correctly classified and zero incorrectly classified and considered the optimal one for identifying smart home devices.

The MLP model was compared to a baseline model Deep Learning 4j (Dl4j) using a dataset containing 93 instances. The results show that the MLP model can accurately classify devices with 100% accuracy and outperformed the (Dl4j). Furthermore, to assess the effectiveness of the proposed model, its performance has been compared using a dataset containing 130 instances and proved that MLP gives better results.

These results suggest that the use of the MLP model for identifying smart devices in a smart home environment is accurate and could potentially be applied to other IoT environments as well. The use of DL and the MLP model can enable the effective processing and analysis of large amounts of data, enabling the development of more advanced and intelligent IoT systems.

CONCLUSION

The research presented in this paper proposes a method for identifying smart home devices using Deep neural networks (DNN) with the integration of the Weka platform and Multi-layer Perceptron Networks (MLP), presents significant results in accurately identifying and distinguishing between various smart home devices based on their attributes. Using a dataset consisting of 93 instances of smart home devices collected from online sources allows the model to be trained and tested on diverse devices.

The highlights of the results include testing the dataset upon two different test options, percentage split and Cross-validation test option. The results option in both cases are very accurate. The method extended to include adding hidden layers with the two test options, proving that using both models without adding hidden layers gives more accurate results. Additionally, the dataset was classified using Deep Learning 4j (Dl4j) classifier with 66% percentage split and ten folds cross-validation. The results show that (Dl4j) gives low accuracy compared to the performance of (MLP). Moreover, the proposed method was successfully implemented on a larger dataset of over 100 instances, demonstrating its potential to be used with various smart devices in different IoT environments. Deep Neural Networks (DNN) incorporation of specific groups for classification using Multi-layer Perceptron Networks (MLP) allows for efficient identification of smart devices, which can have significant implications in areas such as home automation and appliance management. Future work should focus on testing the model on even larger and more diverse datasets to ensure its effectiveness in real-world settings. Also, further research can be conducted to explore the potential of incorporating additional features or attributes to improve the model's accuracy.

ACKNOWLEDGEMENTS

I want to extend my gratitude and appreciation to my family for their continuous support, as well as to my supervisor Prof. Johnson I. Agbinya for his valuable guidance, and my colleague Dr. Jamal Khamis for his support and encouragement.

AUTHOR'S CONTRIBUTIONS

1st author contributed to the design, implementation, analysis of the results, and manuscript writing. 2nd author involved in planning and supervising the work.

CONFLICT OF INTEREST

As the authors of this manuscript, we certify that we have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in the manuscript.

COMPETING INTERESTS

None: There is none competing interests.

ETHICS COMMITTEE

None.

FUNDING INFORMATION

The work presented here has been privately funded.

AUTHORS BIOGRAPHY

1st Sana Abdelaziz Bkheet received her BSc in computer science from Sudan University college for girls in 2004, and Msc in computer science from University of Science and Technology in 2011, Khartoum- Sudan. PHD in progress since 2015. She is currently a lecturer at faculty of science- computer Science department in Northern Border University (NBU)- Arar – Kingdom of Saudi Arabia.

Sana Abdelaziz Bkheet current research interests include the Internet of Things (IoT), Wireless Sensor Network, and Big Data.

2nd Johnson I. Agbinya received the B.Sc. degree in electronic/electrical engineering from Obafemi Awolowo University (OAU), Ife Nigeria, M.Sc. (Research) degree in electronic control from the University of Strathclyde Glasgow Scotland, and Ph.D. degree in microwave radar systems from La Trobe University, Melbourne Australia. He is currently the Head with the School of Information Technology and Engineering, Melbourne Institute of Technology, Melbourne, VIC, Australia. Prior to that, he was an Associate Professor with the Department of Electronic engineering, La Trobe University, Melbourne. He is also an Honorary Professor at the University of Witwatersrand, Johannesburg, South Africa, Extraordinary Professor at the University of the Western Cape, Cape Town, South Africa, and the Tshwane University of Technology, Pretoria, South Africa. Prior to joining La Trobe University in November 2011, he was Senior Research Scientist with CSIRO Telecommunications and Industrial Physics (now CSIROICT) from 1993 to 2000, Principal Research Engineering with Vodafone Australia from 2000 to 2003, and Senior Lecturer with UTS Australia from 2003 to 2011. His research interests include remote sensing, Internet of things (machine to machine communications), biomonitoring systems, wireless power transfer, mobile communications, and biometrics systems. He has authored/coauthored nine books in telecommunications, some of which are used as textbooks. He is the Founder of the International Conference on Broadband Communications and Biomedical Applications, Pan African Conference on Science, Computing and Telecommunications, and the African Journal of Information and Communication Technology.

REFERENCES

Beckel C,Serfas H,Zeeb E, Moritz G,Golatowski F.(2011). Requirements for Smart Home Applications and Realization withWS4D-PipesBox. IEEE. DOI: 10.1109/ETFA.2011.6059229.

Bengio, Y., Courville, A., & Vincent, P. (2012). Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(2), 1792-1828.

Bugeja J, Jacobsson A, Davidsson P. (2016). On Privacy and Security Challenges in Smart Connected Homes. IEEE European Intelligence and Security Informatics Conference. 978-1-5090-2857-3/16, DOI 10.1109/EISIC.2016.21.

Chandramohan J, Nagarajan R, Satheeshkumar K, Ajithkumar N, Gopinath P, Ranjithkumar S. (2017). Intelligent Smart Home Automation and Security System Using Arduino and Wi-fi . International Journal Of Engineering And Computer Science (IJECS). ISSN:2319-7242, Volume 6, Issue 3, Page No. 20694-20698, DOI: 10.18535/ijecs/v6i3.53.

Cvitić I , Peraković D , Periša M, Gupta B. (2020). Ensemble machine learning approach for classification of IoT devices in smart home. International Journal of Machine Learning and Cybernetics. https://doi.org/10.1007/s13042-020-01241-0.

Dietterich, T. G. (2002). Machine learning research: Four current directions. AI Magazine, 23(4), 97-136.

Frank E, Hall M, Holmes G, Kirkby R, Pfahringer B, Witten I.(2010), WEKA A Machine Learning Workbench for Data Mining,ACM, DOI: 10.1007/978-0-387-09823-4_66.

Foster, D. P. (2003). The value of large data sets for machine learning research. ACM SIGKDD Explorations Newsletter, 5(2), 1-10.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). Deep learning. MIT Press, DOI:10.1007/s10710-017-9314-z.

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P,Witten I.(2008), The WEKA Data Mining Software: An Update, ACM, DOI: 10.1145/1656274.1656278.

Hastie T, Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer, DOI.org/10.1007/978-0-387-21606-5.

Kutyniok G. (2022). The Mathematics of Artificial Intelligence, https://arxiv.org/abs/2203.08890.

Lin H, Bergmann N. (2016). IoT Privacy and Security Challenges for Smart Home Environments. Information. DOI:10.3390/info7030044.

Mathew A , Amudha P, Sivakumari S. (2021). Deep Learning Techniques: An Overview. Springer, International conference on advance machine learning technologies and application. DOI:10.1007/978-981-15-3383-9_54.

Nazzal J, El-Emary I, Najim S. (2008).Multi-layer Perceptron Neural Network (MLPs) For Analyzing the Properties of Jordan Oil Shale. World Applied Sciences Journal. Pg 546 – 552. ISSN 1818-4952.

Ndhage S & Raina C, (2016). A review on Machine Learning Techniques, International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), ISSN: 2321-8169.

Popescu M, Balas V, Popescu L, Mastorakis N.(2009). Multi-layer Perceptron and Neural Networks. WSEAS Transactions on Circuits and Systems. Volume 8, Issue 7. Pg 579–588 ISSN: 1109-2734.

Pradeep S, Kousalya T, Suresh K, Edwin J. (2016) . IoT AND ITS CONNECTIVITY CHALLENGES IN SMART HOME. International Research Journal of Engineering and Technology (IRJET). Volume: 03 Issue: 12 | Dec -2016. p-ISSN: 2395-007.

Sagar V, SM Kusuma. (2015). Home Automation Using Internet of Things. International Research Journal of Engineering and Technology (IRJET).Volume: 02 Issue: 03 | Jan-2015. Pg 1965 -1970. e-ISSN: 2395-0056.

Shetty D, Harshavardhan C, Varma M, Navi S, Ahmed M.(2020). Diving Deep into Deep Learning: History, Evolution, Types and Applications. International Journal of Innovative Technology and Exploring Engineering (IJITEE). Volume 9, Issue 3.ISSN: 2278-3075. DOI: 10.35940/ijitee.A4865.019320.

Vapnik, V. (1995). The nature of statistical learning theory (2nd ed.). Springer, doi.org/10.1007/978-1-4757-2440-0.

Wang L, Li C, Ye J, Zhang C. (2016). A survey of multi-view machine learning. IEEE Transactions on Knowledge and Data Engineering, 28(1), 3-12.

Zantalis F, Koulouras G, Karabetsos S, Kandris D. (2019). A Review of Machine Learning and IoT in Smart Transportation, future Internet, DOI:10.3390.