IOT BASED SMART SECURITY SOLUTION FOR UNIQUE AUTHENTICATION

A PROJECT REPORT

Submitted by

MILI CHATURVEDI [Reg. No.: RA1911004010072]
SAMARPAN DAS [Reg. No.: RA1911004010074]
MATURA GANESH [Reg. No.: RA1911004010113]

Under the guidance of

Mrs. A. BHAVANI
(Assistant Professor, Department of Electronics & Communication Engineering)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
ELECTRONICS & COMMUNICATION ENGINEERING
of

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

S.R.M. NAGAR, Kattankulathur, Chengalpattu District

MAY 2023

SRM UNIVERSITY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**IOT BASED SMART SECURITY SOLUTION FOR UNIQUE AUTHENTICATION**" is the bonafide work of "**SAMARPAN DAS [Reg No: RA1911004010074]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. A. Bhavani
GUIDE
Assistant Professor
Dept. of Electronics &
Communication Engineering

SIGNATURE

Dr. Shanti Prince
HEAD OF THE DEPARTMENT
Dept. of Electronics &
Communication Engineering

Signature of the Internal Examiner

Signature of the External Examiner

DECLARATION

I hereby declare that the Major Project entitled "**IOT BASED SMART SECURITY SOLUTION FOR UNIQUE AUTHENTICATION**" to be submitted for the Degree of Bachelor of Technology is our original work as a team and the dissertation has not formed the basis of any degree, diploma, associate-ship or fellowship of similar other titles. It has not been submitted to any other University or institution for the award of any degree or diploma.

Place:

Date:

SAMARPAN DAS
[RA1911004010074]

ABSTRACT

To ensure the confidentiality of our home stay, office, banks etc., it uses several security measures. Thus, making security the primary facet of our approach. This can help ensure that it keeps unauthorized access out of our way. The use of biometrics-based authentication like fingerprint recognition and face recognition for unique authentication is of paramount importance as it provides a high level of reliability. It is very difficult to imitate fingerprints and faces of an individual. This paper proposes an IOT based unique authentication of fingerprint and face for opening locks in smart homes. To strengthen the traditional door lock system at home or authentication system a fingerprint scanner and a camera is installed. The database will keep a record of anyone trying to access the lock system and the buzzer will go off when the match is not found. The system is controlled by a Raspberry pi 3 processor and an Arduino module. Access to the door will only be provided when a match for both face and fingerprint is found. The SDG followed by us for the project shall be 'Industry, Innovation and Infrastructure'.

Keywords - Fingerprint sensor, Raspberry Pi-3 B+, Raspberry pi camera, SVM Algorithm, CNN, Inception Neural Network version-1, Inception Neural Network version-2, VNC viewer, Wnetwatcher.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere gratitude to all those who have supported me throughout our project. First and foremost, I would like to thank Dr. Shanti Prince, Head of the Department, for providing me with the opportunity to work on this project.

I would like to extend my sincere thanks to Dr. T. Rajalakshmi, my project coordinator, for her valuable guidance and constant support throughout the project. Her expert knowledge and timely feedback have been invaluable in the completion of my work.

I would also like to thank Mrs. A. Bhavani, my guide, for her continuous support and guidance throughout the project. Her inputs and suggestions have been instrumental in the successful completion of this project.

I would like to express my sincere appreciation to all the faculty members of the Department for their valuable inputs, encouragement, and guidance throughout the project.

Finally, I would like to express our gratitude to my loved ones and friends for their constant support and inspiration during this endeavor.

Thank you all for your support and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolution Neural Network
SVM	Support Vector Machine Learning Algorithm
ls	list- a Linux command
mkdir	make directory- a Linux command

CHAPTER 1

INTRODUCTION

In existing security systems there are smart home lock systems such as keypad lock systems where a code is punched in and access is provided if the code is correct, or the voice recognition lock system, there still is scope to potentially enhance these security solutions. Having two factor authentication ensures a better approach to providing a smarter solution for user authentication. One of main inspirations for this project was to experiment with a new standard was an existing 2 factor authentication system approach for verification. This model included finger ridges and voice recognition to authenticate the user.

In this paper a model that includes fingerprint and face recognition for authentication will be developed. Including two biometric factors makes the system even more secure as it is difficult to impersonate both the face and fingerprint of an individual with accuracy. Thus, it will reduce the access risk to unauthorized individuals by fraud means. This project has three phases. The first phase includes fingerprint sensor code and testing of it with a database created consisting of a certain number of datasets. The second phase includes creating face recognition models and comparing them to get the most efficient model so that it can implement them and test with the dataset first to integrate it with the hardware so that working with real time data can be tested. Here it uses a few deep learning techniques and a few machine learning models and it compares the accuracy of all of the four models and finally uses the model that has the highest accuracy score.

The third and last phase includes integration of all the components together to test the final working of the model. In this phase it integrates the final deep learning model with the hardware and integrates a camera module with the same. This entire setup also has a fingerprint module attached which would be needed to capture the fingerprint of the person. The camera model will be used to capture the face imprint of the user and will be sent for processing into the raspberry pi. In the raspberry pi needed computation will occur and only if the fingerprint that has been captured recently matches with the existing fingerprint and the new face picture imprint matches with an existing face picture imprint and this imprint is of the same person as that of whom the fingerprint impression was captured. The solenoid lock will open. Once the solenoid lock opens up, the users will gain access to the lockers, or the secure facility and it is a successful operation. On the other hand, if either of the fingerprint or the face imprint does not match with the existing impressions, the lock will not open, and the user will not get access to the secure facility. This operation will guarantee an enhanced security to any secured facility and will make fraudulent activities much less successful.

1.1 Background

Even with the existence of many different types of smart home lock systems, like voice recognition locks and keypad locks that require a code to be entered before granting entry, there is still room to improve these security measures. Two-factor authentication guarantees a better strategy for delivering a smarter user authentication solution.

Existing systems mainly have pin code based or password-based authentication systems. These systems are pretty accurate and safe but then the user needs to always remember their own passwords and pin codes which makes it a hectic job. The other kind of system that exists right now are fingerprint-based security systems.

Figure 1.1 Fingerprint lock

The problem with these fingerprint-based systems which have only one factor authentication is that in case someone wants to impersonate the real person and gets an imprint of the original user's fingerprint, they might very well misuse the system and bypass the security layer – and get access to the facilities.

1.2 Proposed System

In our two-factor authenticated system, there is no need to remember any details because there is no use of passwords or pins or patterns. Because of not using passwords or pins or patterns or any other details that the user must remember, our system becomes a much more user - friendly option for commercial use. The user can simply appear before the system and use their own fingerprint and face implants to unlock the lock by successfully passing the authentication layers and gaining access to the facility. Thus, the system that it proposes is much more user-friendly and convenient to use than most existing systems.

It also does not compromise on the security layers and have used cutting edge two factor authentication techniques to make sure that cases of unauthorized access to secure facilities can be minimized as much as possible.

1.3 Need for the system

Our system uses a two-factor authentication technique of using fingerprint and Face ID to unlock the lockers and gain access to secure facilities. Only if the two layers of security are passed, the user will get access to the facility and if either of the two security layers fail the solenoid lock will not be fired up and the user will not be given access to the facility. This system thus keeps in mind, both the convenience of the user and the security of the facility.

1.4 Scope of the project

The project aims to propose an improved version of existing security systems by increasing the number of layers in authentication and by using enhanced authentication systems. The addition of face recognition along with existing systems of fingerprint. Implement and identify efficient machine learning and deep learning models against the target dataset and find which performs best by projecting the accuracy scores.

The software end of the systems is integrated with the hardware consisting of microcontrollers and microprocessors. The system is maintained by a Raspberry pi 3 processor and an Arduino module. Access to the door will only be provided when a match for both face and fingerprint is found. This entire hardware system is portable and can easily be installed in walls or doors.

CHAPTER 2

LITERATURE SURVEY

2.1 Finger touch based mobile user authentication using physical vibrations

The paper is based on the surface and physiological properties, which give a low-cost security solution. The technology detects touch using vibrating impulses that operate on surfaces such as a wooden board. This technique uses physical vibrations to provide mobile user authentication utilizing finger inputs on a surface. It expands biometric authentication above and beyond touch screens to solid surface for Internet of Things devices. This technology employs a portable motor and receiver that may be readily connected and detachable from any surface. In the frequency domain, several algorithms such as lock pattern and basic motions are employed to monitor properties such as contact area and touching force.

2.2 Hades IoT

HADES- IOT provides tamper-proof protection for a wide variety of Linux-based devices. Its key advantage is that it has a minimal overhead, making it suited for Linux - based systems. It detects malware in IoT with 100% accuracy while requiring just 5.5% of the available RAM and using a small amount of CPU resources.

2.3 Biometric authentication with finger vein images using QDA

There are many studies made on Physical attributes such as fingerprint, iris, finger ridges, finger knuckles etc. Among those finger vein recognition is a new method in the biometric system. The finger vein can be used for the authentication purpose of an authorized user. This proposed model uses the quadrature discriminant analysis to recognize the finger vein of the individual. The finger vein images go through many steps which includes preprocessing which makes the image more stable for further processing. Then QDA is applied which is followed by the minimum distance classifier.

2.4 Finger Knuckle print

Finger knuckle print is a new biometric identifier. A specific data capturing device is being constructed to capture the print of the finger knuckle. An efficient algorithm is developed to process the captured data. A convex direction map of the finger knuckle print is extracted , based on which the feature extraction process takes place. An efficient coding is also constructed to extract and represent the finger knuckle print features. A database was also established to observe the performance of the proposed system.

2.5 Door access based on Unique Authentication

The finger ridges are different for every individual and also the voice of an individual does not match any other individual's voice with accuracy. This paper proposes a smart solution for door unlocking by combining the finger ridges and voice of an authorized user. Here to improve the security, along with a fingerprint sensor a microphone is used to capture the voice of the user. A database is constructed to maintain the user's information. In this model the total system is operated by a Raspberry pi 3 B+ module and PIWIIO software is being used for the task of voice recognition. Door lock will be opened only if the two mention factors of an individual are matched, else a buzzer is activated and a SMS is sent to the authorized user.

2.6 Smart security system using Face recognition

Face recognition plays a big role in the field of biometric security systems. In this paper, a system is built for restricted areas, where people have to use their facial features in the training database for authentication. The face is detected through human motion. To determine the authority of an individual their face recognition is performed. Results proves the effectiveness of using face recognition and helps in restricting the unauthorized user.

2.7 Improving ATM security via face recognition

In this study, face recognition is used in ATMs to authenticate users. There are two steps: the first is verification, in which the individual's face is compared to all other provided persons in the database, and the second is identification, in which the system compares the user's face to all other given individuals in the database. This approach develops the framework analysis with the PCA algorithm for each type of biometric equipment. Face recognition begins with a photograph that attempts to locate the individual in the image. This is accomplished using a variety of techniques such as motions, blurred human outlines, and skin tones.

2.8 Face and speech recognition based Smart home

This paper presents a smart home based on two emerging technologies which is face recognition for security purposes and voice recognition for automation purposes. Security systems ensure the working of the smart homes by the stored database of the authorized users. Voice control turns out to be a great benefit for the handicapped,elders and paralyzed people.

CHAPTER 3

SYSTEM ANALYSIS

To understand the system of our proposed model it may now investigate the algorithms in detail. The algorithms are inspired by principles of Machine Learning and Deep Learning techniques. It has used libraries like Scikit-learn or sklearn and tensorflow to implement these algorithms. The libraries give us mathematical equations that can be imported. These models are highly optimized and can be used directly in code platforms to be used for modeling.

In the Machine Learning space it uses Decision Tree and Support Vector machines and for deep learning techniques it uses Inception Neural Networks. Both the two versions of Inception Neural Networks, i.e., Inception Neural Network -1 and Inception Neural Network - 2 have been implemented. Techniques of transfer learning have been used to make these models adaptive to our use case. By transfer learning it has added the required extra layers of convolutional neural networks or CNNs as they are popularly known by. CNNs specialize in modeling image datasets thus it has been chosen in the mentioned case of work.

Among the various types of convoluted neural networks they are available and efficiently used in the popular research papers of this subject, the inception neural networks are the ones that delivered the highest accuracy scores. The machine learning algorithms are also implemented mainly to check if they perform better than the deep neural networks in our case of work as the dataset was limited. It was then seen that the neural networks performed much better than the machine learning models.

The algorithms that it utilized here to come to a better understanding of which model provides a better accuracy for the required work are:

Let us discuss these algorithms in detail.

1. Deep learning models-

   (i) Inception Neural Network Version 1, Version 2 :

   The Inception neural network was a deep neural network model that researchers in

Google came up with it in 2014. The main task of the Inception networks tries to improve the accuracy and beat conventional ResNet type models in image classification tasks. The researchers also made sure to come with a model that minimizes computational costs such that the training of the model could take less time but also make sure to beat the benchmark models. The Inception network uses an efficient approach of designing convolutional neural networks (CNNs), that are a type of neural network popularly used for image classification and segmentation tasks.

Changing the notion of using a fixed filter size for each of the convolutional layers, the Inception network has filters of different sizes 1x1, 3x3, and 5x5 in that are put parallel to each other. The outputs of these filters are summed and fed in the next layer of the subsequent neural net. This approach lets the network obtain information at different scales and reduces the number of parameters that needs to be learned for efficient modeling of the task in hand.

The Inception network also implements a mathematical technique called "network in network" (NiN), which basically involves replacing some of the authentic convolutional layers with smaller versions of fully connected networks. This allows the network to obtain more complicated features and reduces the amounts of parameters required for each convolutional layer. This essentially reduces the time required for training the models even though the network is sufficiently large.

2. Machine learning models-

   (i) Support Vector Machine :

   It is one of the most appreciated Supervised learning algorithm models and may be applied to both classification and regression. However, in Machine Learning, this model is mostly utilized for classification issues.

   The goal of this technique is to create a decision boundary known as a hyperplane that can divide n-dimensional space into multiple classes so that fresh data points may be placed in the proper category in the future.

   SVM selects extreme vector points, which are referred to as support vectors, thus the term 'Support Vector Machine'.

SVM is classified into two categories:

- Linear SVM – Mainly used for data that can be linearly split, which may be defined as data that can be categorized into two classes using a single straight line.
- In this scenario, the classifier will be referred to as the Linear SVM classifier.
- Non-Linear SVM- A dataset that is not linearly separable cannot be categorized using a straight geometric line. The machine learning based classifier utilized in this case is known as a Non-linear SVM based classifier.

The hyperplane is the optimal judgment boundary for classifying the dataset. The features in the dataset impact the size of the hyperplane, thus when there are only two characteristics, the hyperplane would mainly be a straight line. Furthermore, if three characteristics are present, the hyperplane would basically have just two dimensions. It constructs a hyperplane with the biggest feasible margin, or distance between data points. Support vectors are the closest data points or vectors to the hyperplane and have the largest effect on the hyperplane's position. The vectors are known as support vectors because they help the hyperplane.

To explain how the SVM algorithm works, assume a dataset containing black and white tags and characteristics x1, x2. When plotting the coordinates, it is possible to generate numerous straight lines. SVM assists in determining the optimum decision boundary, which is the hyperplane. SVM selects the nearest points from both classes and refers to them as vectors. Margin is the distance between the hyperplane and these locations. The primary goal of SVM is to maximize the margin.

(ii) Decision Tree Classifier:
Decision Tree is a Supervised learning strategy that may be used for both classification and regression problems, however, it is more commonly employed for classification. It is a tree-structured classifier, with core nodes containing dataset properties, branches containing decision rules, and each leaf node containing the outcome.

The Decision Node and the Leaf Node are the two nodes of a Decision tree. Decision nodes are used to make decisions and have many branches, whereas Leaf nodes are used to reflect the conclusion of those decisions and have no more branches. The evaluations or tests are based on the characteristics of the presented data set. It is a graphical illustration of all likely solutions to a problem statement or decision given specific use cases.

A decision tree is so named because, similar to a tree, it starts with the root node and then extends to branches out to form a tree-like structure. A decision tree then asks a question and then splits the tree into subtrees based on the answer (Yes/No), the technique for predicting the class of a given dataset in a decision tree starts at the tree's root node. The method compares the attribute value to that of the other sub-nodes before proceeding to the next node. It goes through the entire procedure again and again until it reaches the tree's leaf node. These algorithms are implemented using Python in Jupyter Notebook and run again our selected dataset to test the models with respect to our use case. The results of the epochs are recorded and extracted from the integrated development environment or IDE, which in our case was a Jupyter Notebook based on the Anaconda Environment.

The result for the Inception Neural Network Version - 1 is given below –

Epoch Accuracy data for Inception Neural Network V1

Table 3.1: Epoch Accuracy and Loss Data for Inception Neural Network V1

Epoch	Loss	Accuracy	Validation loss	Validation accuracy
1/50	0.539	0.848	0.3475	0.913
10/50	0.662	0.898	0.332	0.919
20/50	0.797	0.978	0.3118	0.927
30/50	0.422	0.987	0.408	0.926
40/50	0.329	0.990	0.454	0.921
50/50	0.019	0.993	0.387	0.928

The accuracy received is 92.8%.

Below are the 2 main graphs plotted to visually represent the table above for better understanding.

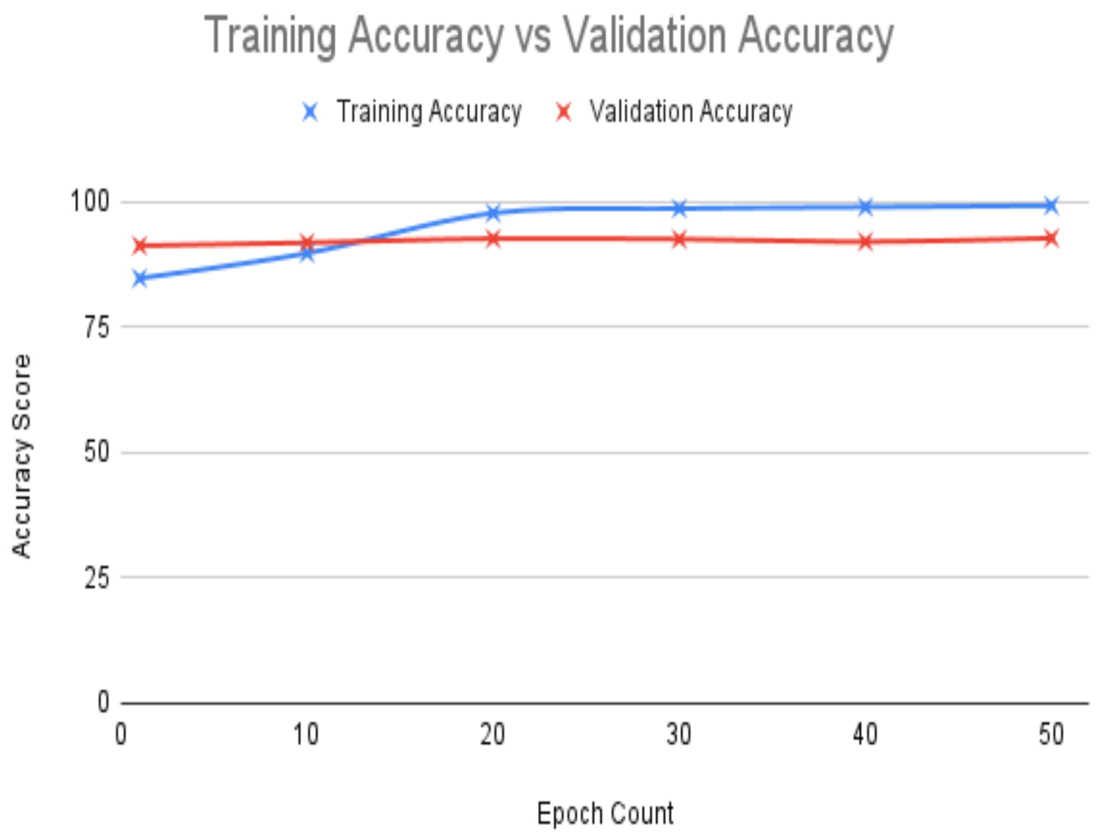

Figure 3.1. Training accuracy vs Validation Accuracy of Inception Neural Network V1

The graph above represents the Training accuracy plotted against the validation accuracy. It can very well understand from the plot that the Training accuracy is better than the validation accuracy, which is a normal phenomena for machine learning and deep learning algorithms.

This is because the training data is used to measure the training accuracy and hence the model sees this data and trains itself accordingly. The validation accuracy on the other hand is based on the data it has not seen. A healthy difference is observed and also, it can be seen that the chance of bias has been avoided by not abruptly training the model excessively.

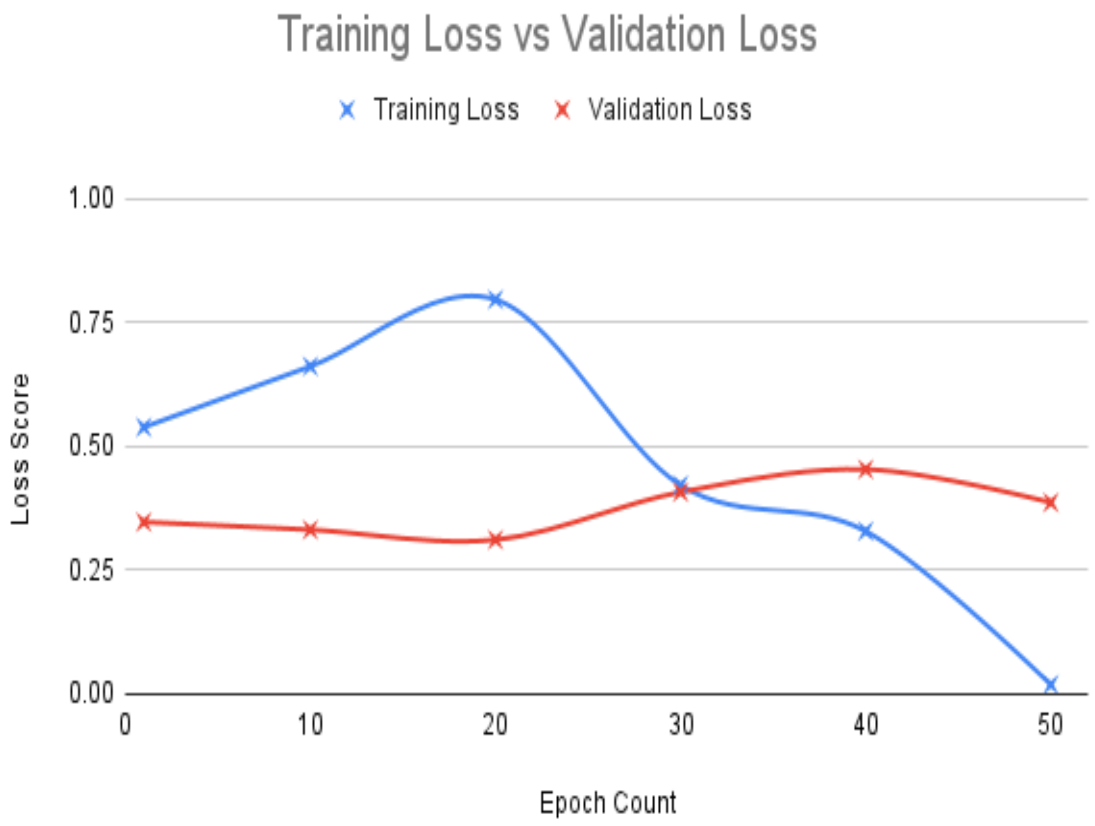

Figure 3.2. Training loss vs Validation loss of Inception Neural Network V1

The graph above represents the Training loss plotted against the Validation loss. Loss is the parameter that shows the deviation between the predicted value and the actual value, i.e., the distance between value predicted by the machine learning or deep learning model vs the ground truth. Here, it can be observed that the training loss is far lesser than the validation loss which is a usual thing in machine learning and deep learning models because training losses are generated by the data that the model has already seen before and thus the predicted values in this case are more accurate. Validation losses on the other hand are More because here the distance between the predicted value and the actual value are more because this is the portion of the data that the model has not seen before and has not been trained on.

Epoch has been kept till 50 because by the 50th Epoch, the validation losses of the training set has already changed. Therefore, there is no point of training the model for more epochs. Training the model for more epochs is mainly resulting in an increase of losses for the validation set.

The result for the Inception Neural Network Version - 2 is given below -

Epoch Accuracy data for Inception Neural Network V – 2

Table 3.2: Epoch Accuracy And Loss Data For Inception Neural Network V2

Epoch	Loss	Accuracy	Validation loss	Validation accuracy
1/50	0.8508	0.7522	0.3478	0.8957
10/50	0.0661	0.9794	0.2007	0.9459
20/50	0.0325	0.9902	0.0.2518	0.9459
30/50	0.0271	0.9915	0.2128	0.9454
40/50	0.0240	0.9937	0.2035	0.9666
50/50	0.0238	0.9946	0.2259	0.9577

The accuracy received is 95.77%. This is the highest accuracy score that was received after all the regularization parameters that were tried upon this model to make sure that the model learn the patterns at the best possible way. Layers of dropout regularization were also added to make sure all the layers optimally learn the parameters.

Below are the 2 main graphs plotted to visually represent the table above for better understanding.

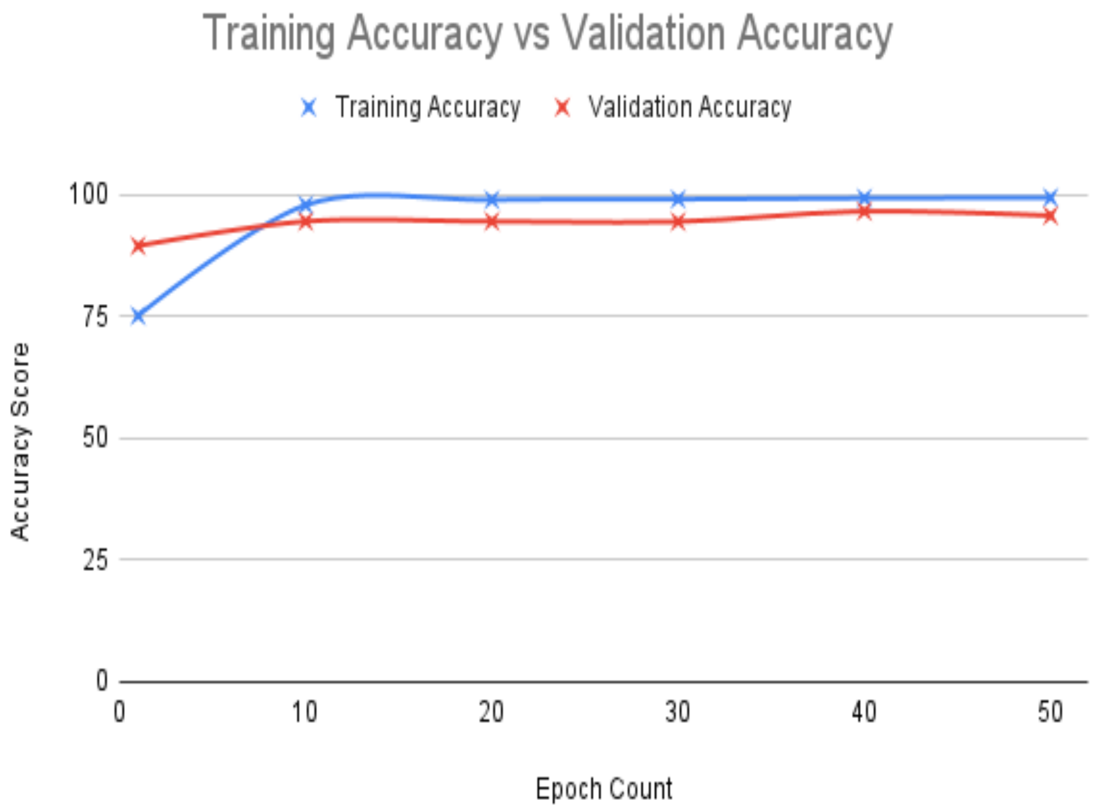

Figure 3.3. Training accuracy vs Validation accuracy of Inception Neural Network V2

The graph above represents the Training accuracy plotted against the validation accuracy. It can very well understand from the plot that the Training accuracy is better than the validation accuracy, which is a normal phenomenon for machine learning and deep learning algorithms. This is because the training data is used to measure the training accuracy and hence the model sees this data and trains itself accordingly.

It can be easily deduced that here for the Inception Neural Network Version 2, the scores of validation accuracy and the training accuracy are better than the ones it received for the Inception Neural Network Version 1. The Inception Neural Network version 2 is a much more deep neural network than its predecessor, the inception neural network version 1, and naturally so the accuracy score for this is better than.

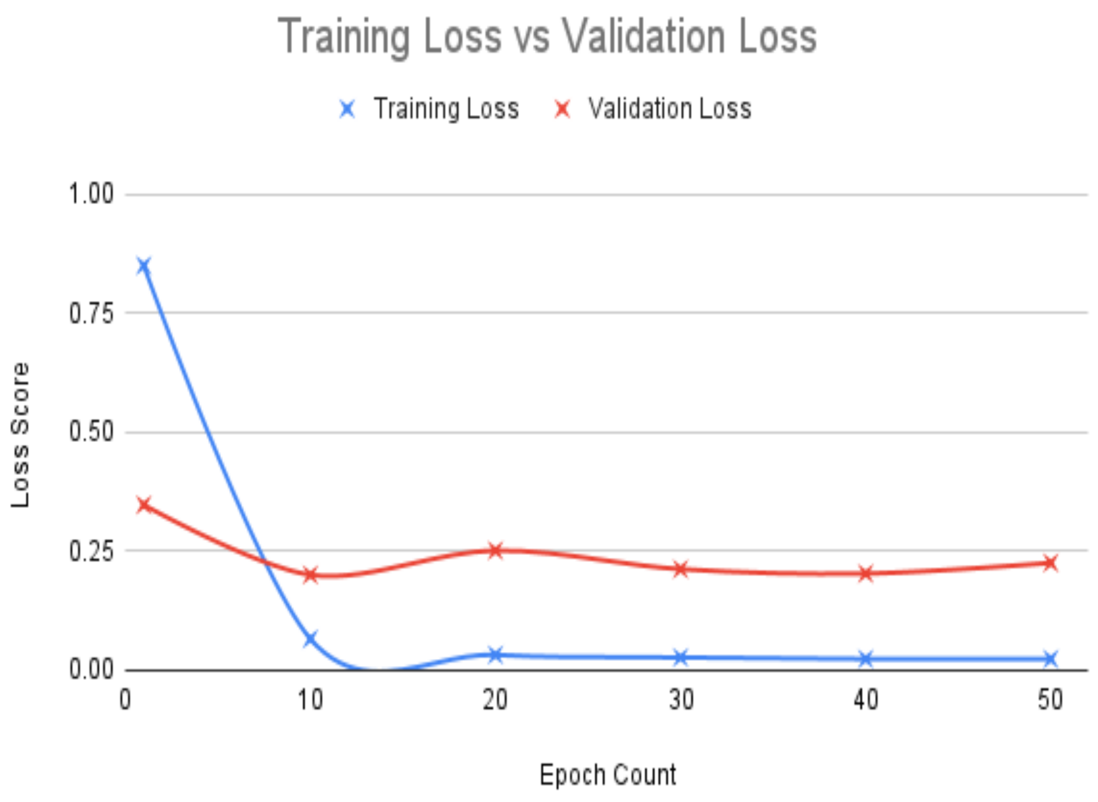

Figure 3.4. Training loss vs Validation loss of Inception Neural Network V2

The graph above represents the Training loss plotted against the Validation loss. Loss is the parameter that shows the deviation between the predicted value and the actual value, i.e., the distance between value predicted by the machine learning or deep learning model vs the ground truth.

In this model it clearly sees validation losses are lesser than the one it observed in the previous model and training loss was more than the one it had seen in the previous model. Which is a clear indication that this model is working better than the one that was previously implemented. The validation loss is lesser, this model works better on the unseen dataset. The training loss is more, which can be taken for lesser bias. Thus, this model works better as per the data that can be seen as its output.

The result for the Support Vector Machine is given below -

Epoch Accuracy data for Support Vector Machine

Table 3.3: Epoch Accuracy and Loss Data for Support Vector Machine

Epoch	Loss	Accuracy	Validation loss	Validation accuracy
1/50	0.6101	0.7993	1.9106	0.7125
10/50	0.0457	0.9834	0.8488	0.8303
20/50	0.0106	0.9961	0.6437	0.8499
30/50	0.0085	0.9971	0.9031	0.8210
40/50	0.0049	0.9977	0.8564	0.8395
50/50	0.0031	0.9977	0.9171	0.8314

The accuracy received is 83.14%. This is less than the accuracy achieved with the Deep learning models mentioned above.

Below are the 2 main graphs plotted to visually represent the table above for better understanding.

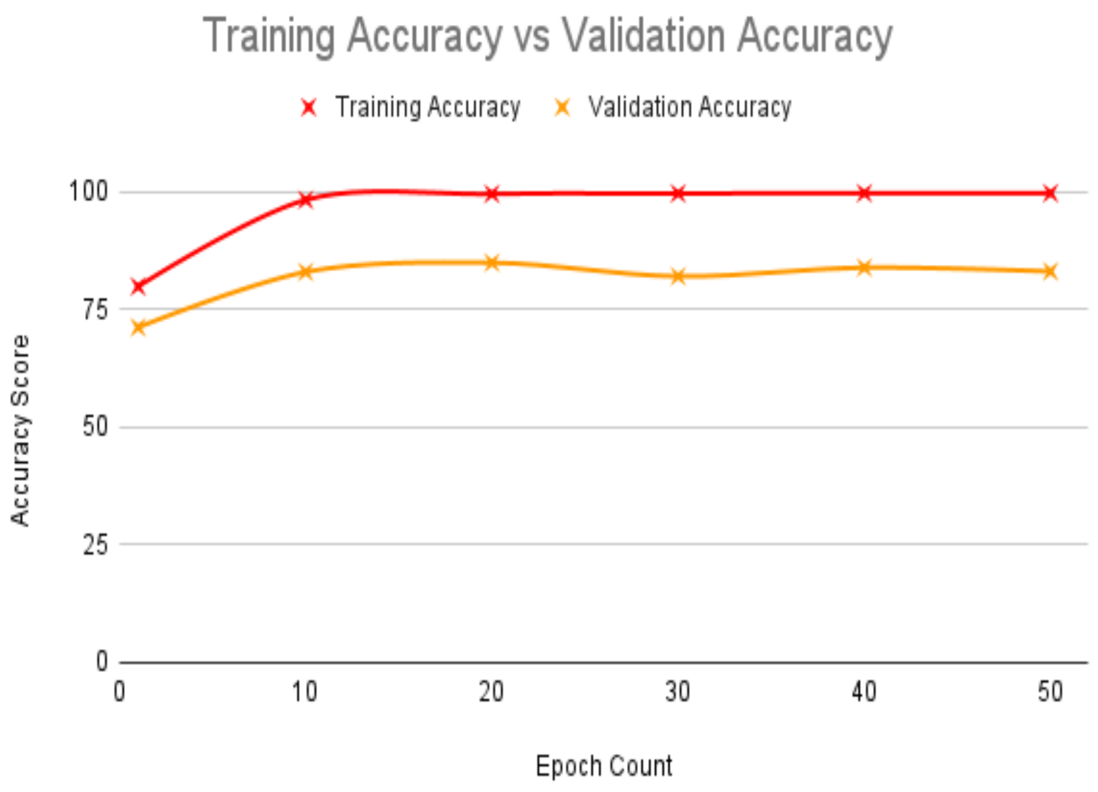

Figure 3.5. Training accuracy vs Validation accuracy of SVM

The graph above represents the Training accuracy plotted against the validation accuracy. It can very it'll understand from the plot that the Training accuracy is better than the validation accuracy, which is a normal phenomenon for machine learning and deep learning algorithms. This is because the training data is used to measure the training accuracy and hence the model sees this data and trains itself accordingly.

Here the training data produces great results, while the validation data does not. This disparity causes this algorithm to not be as good as either of the two previous models. A clear case of bias can be observed here. Which means that the data is almost learning the data and producing results, which is obviously not a great thing. This phenomena are highly destructive for the model. This happens because image classification is a complex task and a machine learning model like Support Vector Machine is not mathematically capable of handling this. Deep Learning models like Convolutional Neural Nets, like the ones it used previously specialized in these tasks.

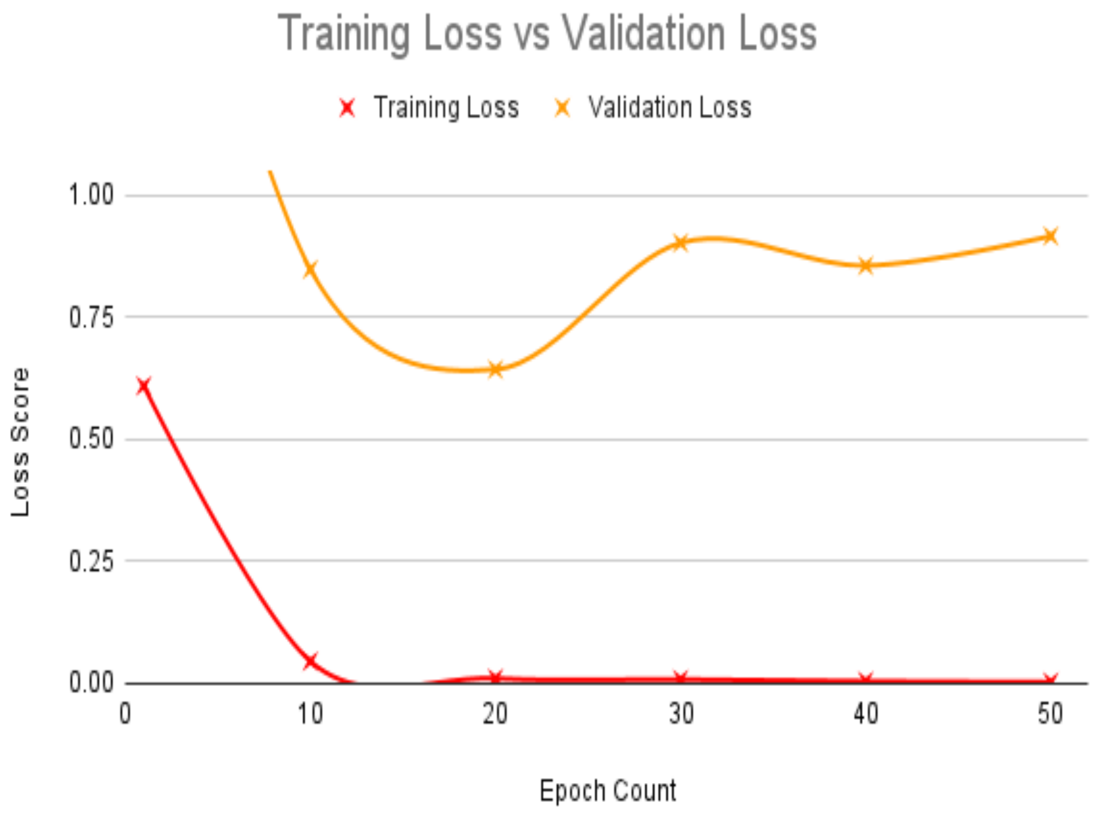

Figure 3.6. Training loss vs Validation loss of SVM

The graph above represents the Training loss plotted against the Validation loss. Loss is the parameter that shows the difference between the predicted value and the actual value, i.e., the difference between value predicted by the machine learning or deep learning model vs the ground truth.

The trend it sees is obvious, here also the model performs extremely it'll in the case of the training set while the performance in the validation set is poor. This clearly indicates that the model is almost learning the data and not focusing on pattern recognition and learning the important features. Therefore, this model performed significantly worse than deep learning models that are implemented.

The result for the Decision Tree Classifier is given below -

Epoch Accuracy data for Decision Tree Classifier

Table 3.4 : Epoch Accuracy and Loss Data for Decision Tree Classifier

Epoch	Loss	Accuracy	Validation loss	Validation accuracy
1/50	2.4870	0.3895	0.6569	0.4366
10/50	0.0892	0.9742	0.5141	0.6598
20/50	0.0288	0.9916	0.6755	0.6613
30/50	0.0174	0.9943	0.8187	0.6593
40/50	0.0146	0.9953	0.0363	0.6481
50/50	0.0104	0.9964	0.1411	0.6270

The accuracy received is 62.70%, which is the lowest of the models that are tested.

Below are the 2 main graphs plotted to visually represent the table above for better understanding.

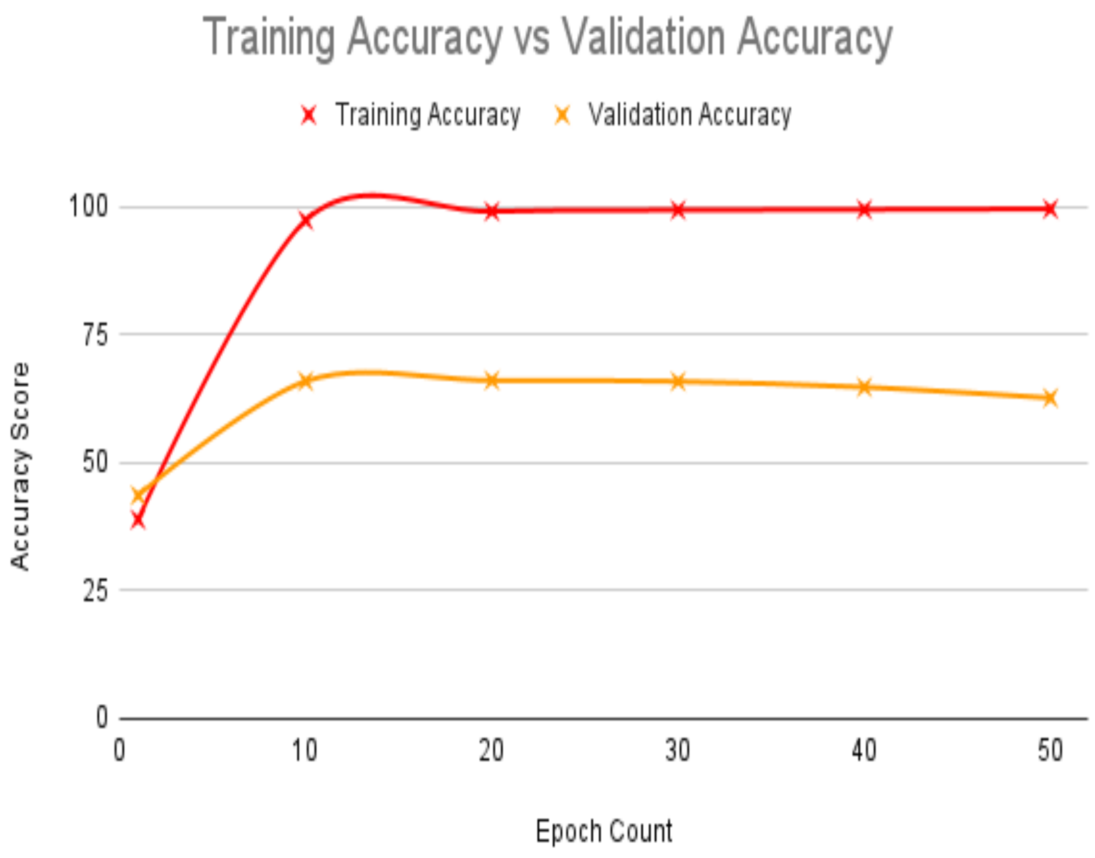

Figure 3.7. Training accuracy vs Validation accuracy of Decision Tree

The graph above represents the Training accuracy plotted against the validation accuracy. It can very it'll understand from the plot that the Training accuracy is better than the validation accuracy, which is a normal phenomenon for machine learning and deep learning algorithms. This is because the training data is used to measure the training accuracy and hence the model sees this data and trains itself accordingly. Here the training data produces great results, while the validation data does not. This disparity causes this algorithm to not be as good as either of the two previous models. A clear case of bias is observed here.

The data can be studied to see that the bias in this model is even more significant than the support vector machine that was previously used. This means that this algorithm is unable to capture the significant features of the model even more than the support vector machine.

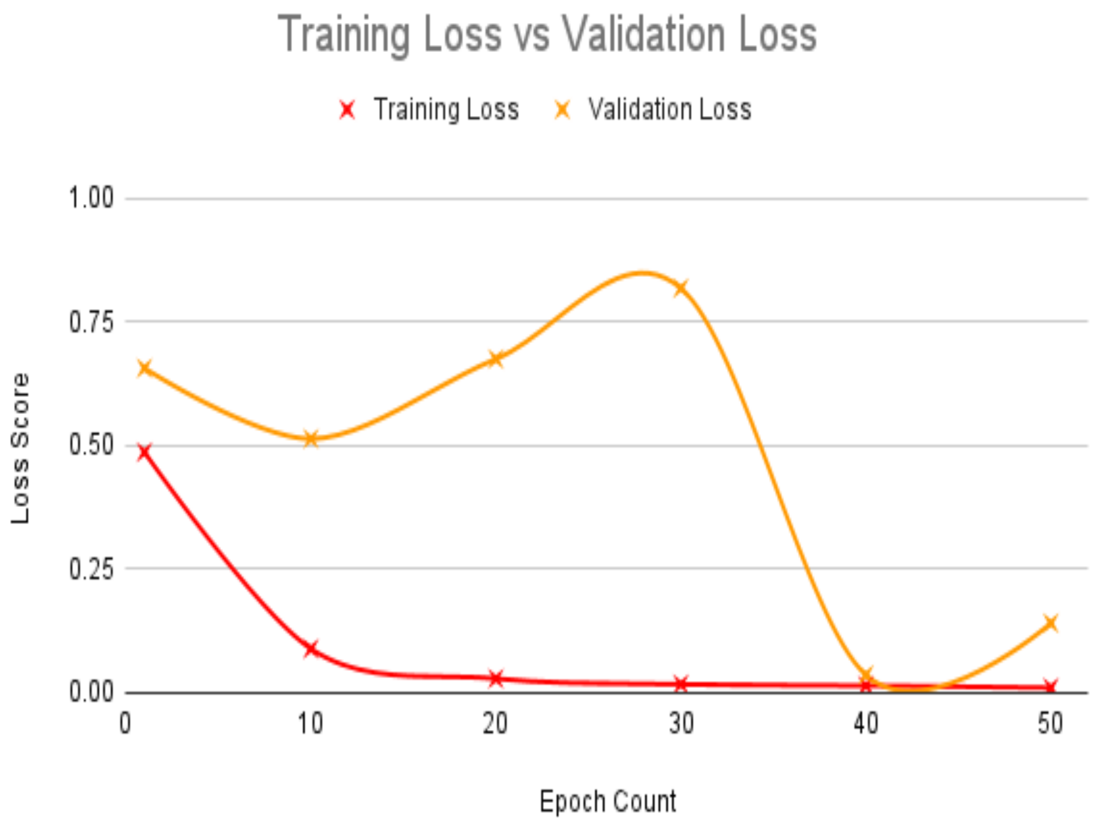

Figure 3.8. Training loss vs Validation loss of Decision Tree

The graph above represents the Training loss plotted against the Validation loss. Loss is the parameter that shows the difference between the predicted value and the actual value, i.e., the difference between value predicted by the machine learning or deep learning model vs the ground truth.

The trend it sees is obvious, here also the model performs extremely it'll in the case of the training set while the performance in the validation set is poor. This clearly indicates that the model is almost learning the data and not focusing on pattern recognition and learning the important features. The loss graphs almost depict the same trend as that of the one that the accuracy graph depicted. The model clearly learns the data and does not pattern recognize the important features. Therefore, this model performed significantly worse than deep learning models that are implemented. Finally, it plots a graph comparing all the four models that are individually implemented. Two of the models are deep learning based and the other than the machine learning based.

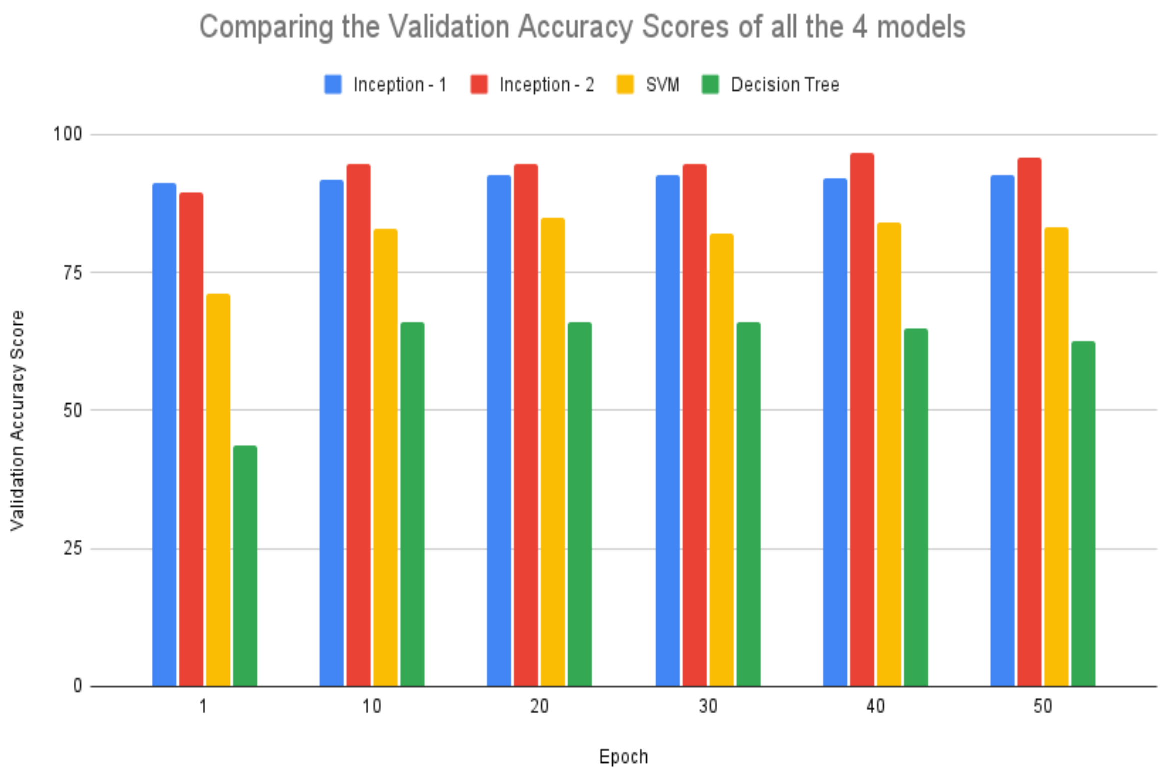

Figure 3.9. Validation accuracy comparison for all the 4 models

The graph compares the validation accuracies of the four models in a similar band of 50 epochs. Here the blue line depicts the Inception Neural Network. Version - One, the red line depicts the Inception Neural Network Version two, the yellow line depicts the Support Vector Machine algorithm and the green line depicts the Decision Tree Classifier. Therefore, the red and blue line depicts the two deep learning models and the yellow line and green line depicts the two machine learning models that are used to build and test the models. It can be clearly seen and observed that the deep learning models perform way better than the machine learning models. This is not surprising because in the case of image recognition and image classification tasks, deep learning models that use large neural networks, especially convolutional neural networks or CNNs tend to perform much better than conventional machine learning models. It can be observed that the Inception neural network version one performs better than the inception neural network version two initially, but after some epochs the inception neural network version two outperforms its predecessor. It can also be observed that either of the two inception networks do not have much scope of increasing their accuracy after the 50th epoch and hence the training was stopped after the 50th epoch.

Also, the observed that the support vector machine algorithm does not increase its accuracy much with an increasing number of epochs, but it performs better than the decision tree algorithm and much worse than those two deep learning models. And the worst performing model is the decision tree algorithm which does not even touch 70% of accuracy even after the 50th epoch. Therefore, it can be comfortably deduced that the deep learning models implemented using large neural networks with a specialized form of convolution layers, performs much better than the machine learning models where the mathematical complexities are way lesser.

CHAPTER 4

SYSTEM DESIGN

4.1 Hardware

1. Fingerprint Sensor:

   This optical fingerprint sensor contains several LEDs that are used by the CMOS or CCD sensors to illuminate both the fingertip areas and the reflected light waves. Currently available sensor versions have the ability of detecting fingers and come in sizes as small as 1mm. It takes a finger impression and compares it to previously scanned patterns.

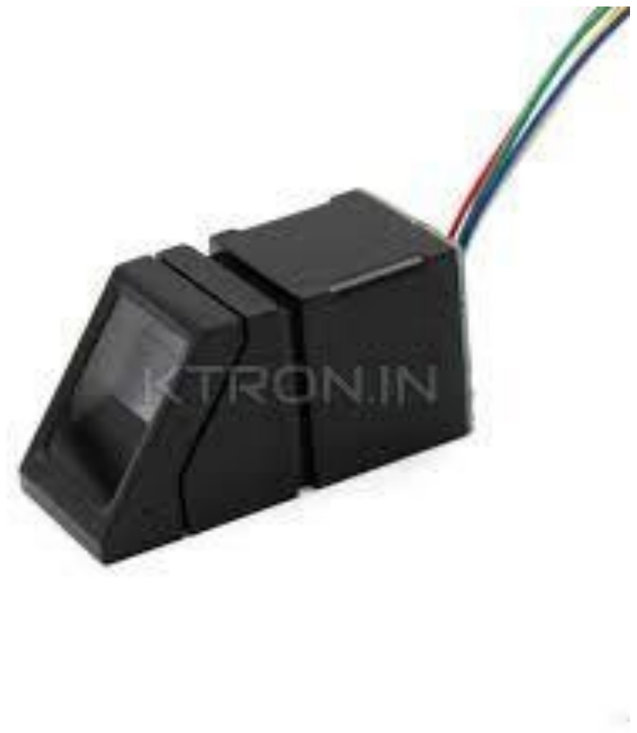

Figure 4.1.1 Fingerprint sensor

2. Raspberry pi 3 B+ :

   On-board is a Broadcom 1.4GHz quad-core 64-bit processor with low energy. 4 USB 2.0 ports are present. The power supply required is via Micro USB or Gpio at 5V/2.5A DC. The interface for the graphics card is already built-in. It contains an expanded 40-pin GPIO header, IEEE 802.11 B/G/N/Ac wireless LAN at 2.4GHz and 2.4GHz, and Bluetooth 4.2. Its use is in word processing, spreadsheets, high-definition video, games, and programming due to the quick processing it offers.

With the aid of this module, the fingerprint and the face database are interfaced to meet our needs. Python is used for programming it. This module is connected to the buzzer, fingerprint reader, face reader, and locking system.

Figure 4.1.2 Raspberry pi 3 B+ model

3. Alarm / Buzzer :

   Any inconsistency in the entries is signaled by the buzzer. The signal's primary function is to transform audio to sound. It frequently uses DC voltage for electricity. The color is black. The range of frequencies is 3,300 Hz. operating voltages of 3 to 24 volts direct current.

Figure 4.1.3 buzzer

4. Central Lock Actuator :

   When an interruption is activated, a low-voltage solenoid in a solenoid lock pulls the latch back into the door. The latch will remain in place until the interruption is activated. The raspberry pi 4 model b module is attached to the solenoid lock. The Raspberry Pi's GPIO Pins can supply external points, which are utilized to activate the lock via the relay. The lock is unlocked when the fingerprint and face of the individual attempting to get access match the pre-stored fingerprint and face.

5. Arduino UNO :

   This is an ATmega328-based, compact, and breadboard-friendly board. Instead of a regular USB cable, it functions using a mini-B version. It can be used to interact with a computer, another Arduino, or another microcontroller, among other things. It is used in our system to communicate with the Raspberry Pi model that is being used.

\

Figure 4.1.4 Arduino UNO board

6. Raspberry Pi Camera :

   In order to use the camera for face recognition, desired picture quality must be achieved. The camera module might be easily connected to the Raspberry Pi board using an existing connector. It has a 5 MP lens and is equipped with CSI (Camera Serial Interface).

Figure 4.1.5 Raspberry pi camera

4.2 Hardware and Software Interfacing

Real-world worries about the safety of material assets and identity theft are becoming more prevalent each day. There are already mechanisms in place, such as pins to punch in to access a secure workplace or biological patterns to authenticate or keys for locker systems. However, these existing models have issues such as forgotten pins, carrying a physical device (key or RFID card) which could be lost. Our two-factor authentication system irradiates the need to carry any physical chip - based device by the user or the need for the user to remember a pin / password, thus reducing chances of human error. The project aims to propose an improved version of existing security systems by increasing the number of layers in authentication and by using enhanced authentication systems. The addition of face recognition along with existing systems of fingerprint. Implement and identify efficient machine learning and deep learning models against the target dataset and find which performs best by projecting the accuracy scores.

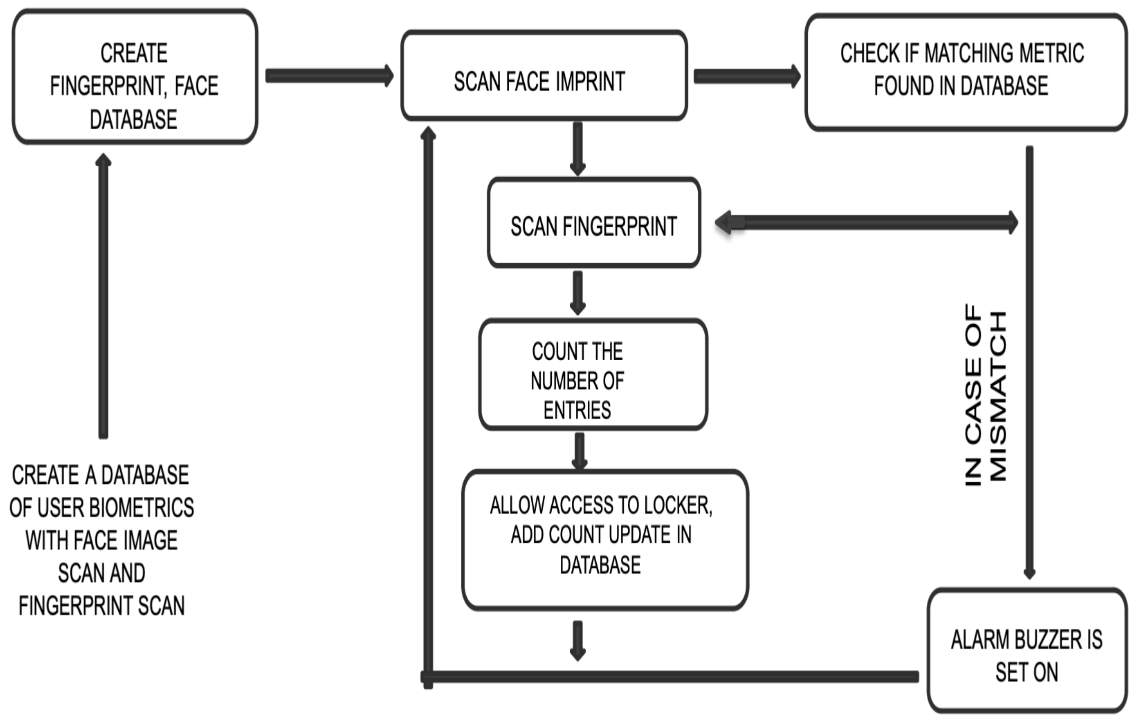
CREATE FINGERPRINT, FACE DATABASE
SCAN FACE IMPRINT
CHECK IF MATCHING METRIC FOUND IN DATABASE
SCAN FINGERPRINT
COUNT THE NUMBER OF ENTRIES
ALLOW ACCESS TO LOCKER, ADD COUNT UPDATE IN DATABASE
IN CASE OF MISMATCH
ALARM BUZZER IS SET ON
CREATE A DATABASE OF USER BIOMETRICS WITH FACE IMAGE SCAN AND FINGERPRINT SCAN

Figure 4.2.1 Proposed workflow

To interface hardware with the software, it has used software. First the code to save the database of fingerprints was completed in C++. The face recognition code that is written in python is integrated with the raspberry pi 3 B+ model. For the purpose of programming in the Raspberry pi it used the vnc viewer software.

A delay for 4 seconds is provided for the fingerprint recognition part. A delay of 5 seconds is provided for face recognition. The Arduino board plays the role of a database storage system. The raspberry pi camera and the fingerprint sensor are connected to the USB ports in the raspberry pi board. The processing happens in the raspberry pi board. Using basic Linux commands, the calls are given in the raspberry pi terminal window to input data and get the final output.

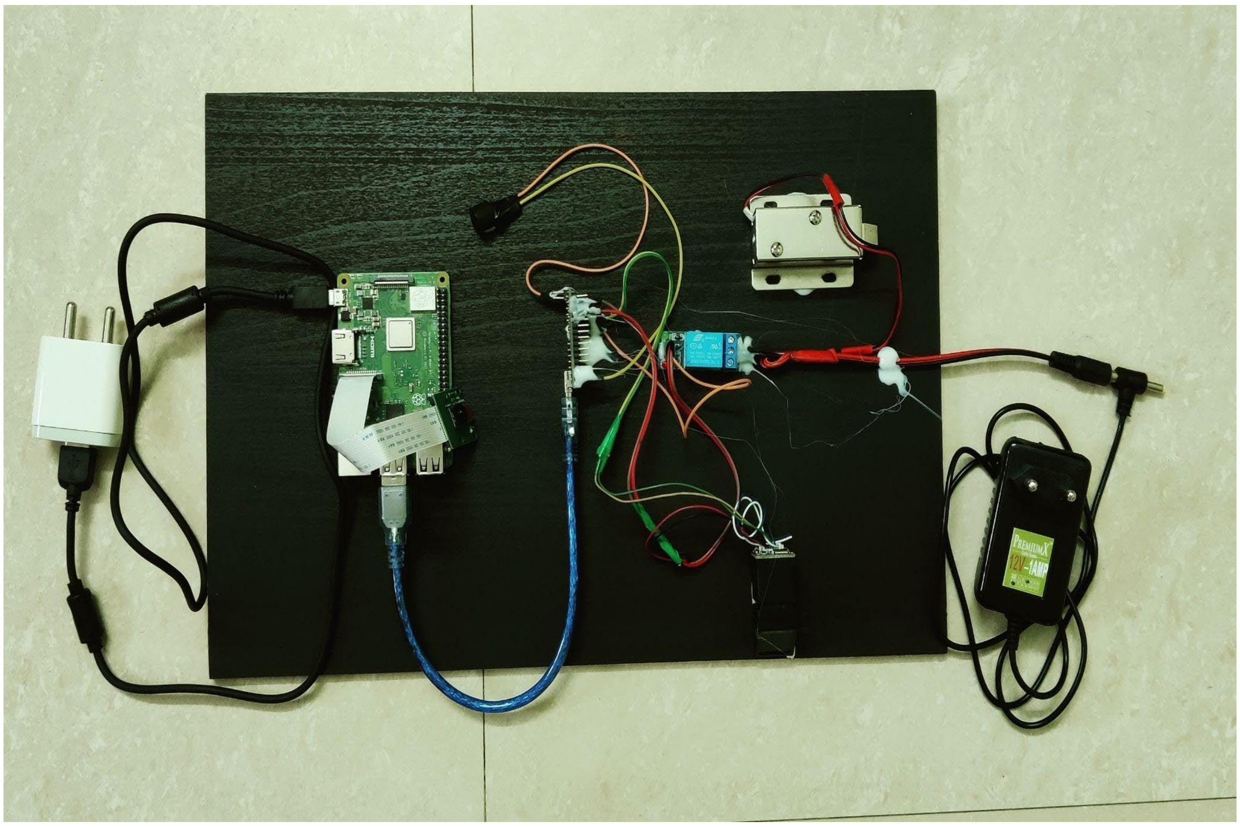

Figure 4.2.2 Hardware connection

The lock is connected to the Arduino board, after the processing in raspberry pi, when the command goes to the Arduino depending upon the match or not the lock and buzzer connected to it will respond.

CHAPTER 5

CODING, TESTING

The fingerprint code is written in C language and then put in alignment with the Arduino IDE. It first stores the input of each person's finger which is done serially to keep the data in series for ease of access. The next step is to interface the code with the Arduino board. The Arduino board acts as the database storage unit. For the face recognition code, it used Python language. Then the code is fed to the Raspberry pi using Wnet watcher and Vnc viewer was used for working into the raspberry pi operating system.

Basics of Linux commands were a prerequisite for understanding how we call the code into the working environment and call for inputs from the user. As mentioned in the system analysis chapter it was concluded to go forward with the Inception Neural network V2. To call the code file into the raspberry pi system we called the python code using the Linux commands. Then with the input of other commands we called for input of the face images of the users. 'mkdir' command was used to make a file (or say a directory) that stored the face image captured by the raspberry pi camera. To list the directories available 'ls' command was used.

The main work of the raspberry pi environment is to help with the face recognition and storage of the face image data in the same file as that of the fingerprint. Only when the input matches the data stored in the systems the lock opens or else the buzzer goes off.

CHAPTER 6

CONCLUSION

After all the connections are successfully plotted, the working of the model will be tested. The expected output is; after the face image of the user is taken as an input, if it matches with the database, then the fingerprint will be sensed and if a match is found, the lock will be unlocked. If a match is not found then the buzzer would go off. The efficiency and the accuracy of the system would however only become clear after the test is completed.

This system can also include a third gateway for authentication by including an OTP verification as the last step. The system then would work as Fingerprint matching, Face recognition, OTP verification. If all three is a match, then the lock would open, if not a message can be sent to the user on the registered mobile number, saying that an attempt was made to access the system.

The system can work either as a fingerprint lock system, face recognition lock system or as two-way authentication system by using a switch. This system uses a laptop as a display for the raspberry pi camera, instead a screen can be connected to the hardware as a development to reduce the usage of laptop. This system can also include a GSM which would help the authorized user to know any fault attempts.

CHAPTER 7

FUTURE ENHANCEMENT

As a future enhancement, the above proposed system can have a GSM attached to it which helps in sending a message to the authorized person in the case of frauds. GSM uses radio communication which makes it sensitive for unauthorized users. This helps in assigning more security to the existing system. This would make the system more hack – proof for being deployed in highly secured facilities. It can also use cryptographic algorithms in the case of using GSM modules to encrypt and decrypt the messages which helps in achieving the cryptographic goals of data security which is an added advantage for the authentication. It can add a display which helps in displaying the face of the user trying to access the system which also states the temperature and time of the location.

It can include an OTP based multi lock system which is very flexible and more secure. When a face is scanned and it matches with the database, fingerprint scanning is done, which when matches with the authorized user, an OTP is sent to the user which is then verified by the user, once the process is complete, the lock gets opened. In the case of any mismatch in the above-mentioned process a message is sent to the authorized user stating false access. This system produces a 3-way authentication which helps in making things more secure. Iris recognition can be included in this system as a future enhancement. Iris recognition being the most accurate will best suit the biometric authentication system. Apart from this, pupil recognition can also be implemented with the use of high-definition cameras.

The iris for everyone is different, as the left and right iris is different for the individual, separate recognition can be done for both eyes. As infrared cameras are being used for the recognition of iris makes it easier at the night or in the dark. Iris recognition is a promising technology in the biometric system. The recognition using physical features are far more reliable than the standard pattern recognition and password systems. As iris recognition is done at a distance from the camera and the iris, it avoids any physical touch which becomes hygiene friendly. Unlike any other biometric recognition like voice recognition or fingerprint recognition which changes through life time, iris will not change for an individual at any time of their life.

Iris recognition being a contactless recognition technology helps to avoid any physical contact. Iris recognition can also be used in the field of biometric key cryptography. In addition to that, iris recognition is human friendly as humans do not have to carry any card to access and humans don't have to remember a pattern or password which reduces the human error. The iris recognition can be used as a living passport at the national border controls. Birth certificates can also be developed with the help of iris recognition.

CHAPTER 8

APPENDIX

1.1 Theory for fingerprint scanner:

The code was primarily written in C / C++. The code was then inserted into the Arduino using a USB cable once the Arduino was connected to a laptop where the C++ code was pre - written and tested. The pre – scanned authorized fingerprints were stored in the memory unit of the fingerprint scanner for operations purposes later when the unit was to be used.

1.2 Theory for face recognition:

Only the best performing deep learning algorithm, i.e., the Inception Neural Network Version – 2 was finally used to recognize the face images in the final model. The deep neural network was written in the Jupyter Notebook IDE and tested. Once satisfactory results were obtained, the model was finalized.

The finalized model was then inserted into the Raspberry pi model by connecting the Raspberry pi to a computer using a USB cable. After connecting the raspberry pi to the computer, the module's window can be accessed from the computer. Once we get access to the raspberry pi's window, the terminal or command prompt of the raspberry pi can be used to program the module. Linux commands work perfectly well for this purpose. The module can be made to run codes and various other functionalities can be accessed using this feature. The camera module that has been attached to the raspberry pi in this project is also operated using this feature. Once the code is inserted into the raspberry pi, the camera is fired up and a few pictures of the authorized personnel are taken to better fit the model and hence improve performance. The pictures that are taken here are primarily stored in the memory element of the raspberry pi.

These model compares the images of the persons that the camera sees during the operation with the pictures captured this the previous step. This comparison step also happens in the raspberry pi for which the code was pre – written and sent into the raspberry pi. For the cases were the image of the subject matches the training images. The next step takes places, i.e., the step of fingerprint comparison. The Arduino module is used to compare the fingerprint that the subject punches into the fingerprint sensor during the time of operation versus the fingerprint that was captured during the setup phase. The fingerprint of only the user whose face was captured during the previous face recognition phase was compared during this stage. This one-on-one classification reduces the energy consumption on the Arduino module and also optimizes speed. Once the verification message is sent from the Arduino module to the relay in the form of a digital zero – one signal, the relay either fires up and triggers the solenoid lock or it does not fire up the lock and the lock is not opened.

2.1 Code for fingerprint scanner :

```
from pyfingerprint.pyfingerprint import PyFingerprint
try:
  f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)
  if ( f.verifyPassword() == False ):
    raise ValueError('The given fingerprint sensor password is wrong!')
except Exception as e:
  print('The fingerprint sensor could not be initialized!')
  print('Exception message: ' + str(e))
  exit(1)

try:
  print('Waiting for finger...')
  while ( f.readImage() == False ):
    pass
  f.convertImage(0x01)
  result = f.searchTemplate()
  positionNumber = result[0]
  if ( positionNumber >= 0 ):
    print('Template already exists at position #' + str(positionNumber))
  else:
    print('Finger was not found in the database.')
except Exception as e:
  print('Operation failed!')
```

```
    print('Exception message: ' + str(e))
    exit(1)
```

2.2 Code for face recognition :

```
import numpy as np
import cv2
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import accuracy_score

# Load the dataset
dataset_path = "/Users/samarpan/Documents/Major Project/CASIA1"
images = []
labels = []
for folder in os.listdir(dataset_path):
    folder_path = os.path.join(dataset_path, folder)
    for image_path in os.listdir(folder_path):
        image = cv2.imread(os.path.join(folder_path, image_path), 0)
        image = cv2.resize(image, (64, 64))
        images.append(image.flatten())
        labels.append(folder)

# Split the dataset into features and target variables
features = np.array(images)
target = np.array(labels)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=0)

# One-hot encode the target variables
```

```
encoder = LabelEncoder()
encoder.fit(y_train)
y_train = encoder.transform(y_train)
y_test = encoder.transform(y_test)

# Build the neural network model
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test))

# Predict the iris class using the trained model
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred.round())
print("Accuracy:", accuracy)
```

2.3 Code for an inception block :

```
def inception_block_1a(X):
    """
    Implementation of an inception block
    """

    X_3x3 = Conv2D(96, (1, 1), data_format='channels_first', name ='inception_3a_3x3_conv1')(X)
    X_3x3 = BatchNormalization(axis=1, epsilon=0.00001, name = 'inception_3a_3x3_bn1')(X_3x3)
    X_3x3 = Activation('relu')(X_3x3)
    X_3x3 = ZeroPadding2D(padding=(1, 1), data_format='channels_first')(X_3x3)
```

```
 X_3x3 = Conv2D(128, (3, 3), data_format='channels_first', name='inception_3a_3x3_conv2')(X_3x3)
    X_3x3 = BatchNormalization(axis=1, epsilon=0.00001, name='inception_3a_3x3_bn2')(X_3x3)
    X_3x3 = Activation('relu')(X_3x3)

    X_5x5 = Conv2D(16, (1, 1), data_format='channels_first', name='inception_3a_5x5_conv1')(X)
    X_5x5 = BatchNormalization(axis=1, epsilon=0.00001, name='inception_3a_5x5_bn1')(X_5x5)
    X_5x5 = Activation('relu')(X_5x5)
    X_5x5 = ZeroPadding2D(padding=(2, 2), data_format='channels_first')(X_5x5)
    X_5x5 = Conv2D(32, (5, 5), data_format='channels_first', name='inception_3a_5x5_conv2')(X_5x5)
    X_5x5 = BatchNormalization(axis=1, epsilon=0.00001, name='inception_3a_5x5_bn2')(X_5x5)
    X_5x5 = Activation('relu')(X_5x5)

    X_pool = MaxPooling2D(pool_size=3, strides=2, data_format='channels_first')(X)
    X_pool = Conv2D(32, (1, 1), data_format='channels_first', name='inception_3a_pool_conv')(X_pool)
    X_pool = BatchNormalization(axis=1, epsilon=0.00001, name='inception_3a_pool_bn')(X_pool)
    X_pool = Activation('relu')(X_pool)
    X_pool = ZeroPadding2D(padding=((3, 4), (3, 4)), data_format='channels_first')(X_pool)

    X_1x1 = Conv2D(64, (1, 1), data_format='channels_first', name='inception_3a_1x1_conv')(X)
    X_1x1 = BatchNormalization(axis=1, epsilon=0.00001, name='inception_3a_1x1_bn')(X_1x1)
    X_1x1 = Activation('relu')(X_1x1)

    # CONCAT
    inception = concatenate([X_3x3, X_5x5, X_pool, X_1x1], axis=1)

    return inception
```

REFERENCES

1. X. Yang, S. Yang, J. Liu, C. Wang, Y. Chen and N. Saxena, "Enabling Finger-Touch-Based Mobile User Authentication via Physical Vibrations on IoT Devices," in IEEE Transactions on Mobile Computing, vol. 21, no. 10, pp. 3565-3580, 1 Oct. 2022, doi: 10.1109/TMC.2021.3057083.
2. D. Breitenbacher, I. Homoliak, Y. L. Aung, Y. Elovici and N. O. Tippenhauer, HADES-IoT: A Practical and Effective Host-Based Anomaly Detection System for IoT Devices (Extended Version)," in IEEE Internet of Things Journal, vol. 9, no. 12, pp. 9640-9658, 15 June 15, 2022, doi: 10.1109/JIOT.2021.3135789.
3. C. Choi and J. Choi, "Ontology-Based Security Context Reasoning for Poitr IoT-Cloud Security Service," in IEEE Access, vol. 7, pp. 110510-110517, 2019, doi: 10.1109/ACCESS.2019.2933859.
4. W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf and Y. A. Bangash, "An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security," in IEEE Internet of Things Journal, vol. 7, no. 10, pp. 10250-10276, Oct. 2020, doi: 10.1109/JIOT.2020.2997651.
5. J. Srinivas, A. K. Das, M. Wazid and A. V. Vasilakos, "Designing Secure User Authentication Protocol for Big Data Collection in IoT-Based Intelligent Transportation System," in IEEE Internet of Things Journal, vol. 8, no. 9, pp. 7727-7744, 1 May 1, 2021, doi: 10.1109/JIOT.2020.3040938
6. S. Batool, A. Hassan, M. A. K. Khattak, A. Shahzad and M. U. Farooq, "IoTAuth: IoT Sensor Data Analytics for User Authentication Using Discriminative Feature Analysis," in *IEEE Access*, vol. 10, pp. 59115-59124, 2022, doi: 10.1109/ACCESS.2022.3178635.
7. L. Zhang, L. Zhang and D. Zhang, "Finger-knuckle-print: A new biometric identifier," *2009 16th IEEE International Conference on Image Processing (ICIP)*, Cairo, Egypt, 2009, pp. 1981-1984, doi: 10.1109/ICIP.2009.5413734.
8. M. Rukhiran, S. Wong-In and P. Netinant, "IoT-Based Biometric Recognition Systems in Education for Identity Verification Services: Quality Assessment Approach," in *IEEE Access*, vol. 11, pp. 22767-22787, 2023, doi: 10.1109/ACCESS.2023.3253024.
9. J. J. Robertson, R. M. Guest, S. J. Elliott and K. O'Connor, "A Framework for Biometric and Interaction Performance Assessment of Automated Border Control Processes," in *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 6, pp. 983-993, Dec. 2017, doi: 10.1109/THMS.2016.2611822.

10. K. M. Alashik and R. Yildirim, "Human Identity Verification From Biometric Dorsal Hand Vein Images Using the DL-GAN Method," in *IEEE Access*, vol. 9, pp. 74194-74208, 2021, doi: 10.1109/ACCESS.2021.3076756.