DEVELOPMENT OF AN INTRUDER DETECTION SYSTEM USING DEEP NEURAL NETWORKS

By

FRANCIS AKINDAPO OYEDIRAN

125/18/1/0121

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF BACHELOR OF ENGINEERING(HONOURS.), B. ENG(HONS.) MECHATRONICS ENGINEERING

DEPARTMENT OF MECHANICAL AND MECHATRONICS ENGINEERING, FACULTY OF ENGINEERING AND TECHNOLOGY, FIRST TECHNICAL UNIVERSITY, IBADAN.

NIGERIA

CERTIFICATION

This is to certify that Francis Akindapo OYEDIRAN, with student registration number 125/18/1/0121 did the project work contained in this thesis under my supervision.

……………………………………..   ……………………………..

Prof. A.M Aibinu   **Date**

………………………………   ………………………………..

Dr. O. A. Adeaga   **Date**

Reader & Ag. Head of Department

B. Eng. (Hons.), M.Sc., Dipl. in Robotics & Automation. Ph. D., CDBL, R.Engr.

MNSE, MIAENG, MASME, MASTFE, MNIMech.E, MNIM,

Research Fellow, Association of Commonwealth Universities

DEDICATION

This work is dedicated to the glory of Almighty God for his protection and care throughout the study and also to my parents, Mr. and Mrs. Oyediran for their care, and financial and moral contributions towards the success of the programme.

ACKNOWLEDGEMENT

Almighty God, the creator of the universe, deserves all glory and adoration since without him, nothing is conceivable. He has given me the ability to live, be healthy, know, be merciful, be gracious, and have the fortitude to continue living.

My supervisor, Prof. A.M. Aibinu, and Mr. Yinka's steadfast advice, knowledge, and patience were crucial in helping to shape my work, for which I am grateful. In the same line, I would also like to express my deep gratitude to my department head, Dr. O.A. Adeaga, for always treating us like his children, even when we didn't deserve it.

I also want to express my gratitude to all of the faculty members who taught me during this course. I want to express my gratitude to Dr. Adebisi, Engr. Ayoade, and Engr. Ilori in particular for their advice, kind words, and patience with my incessant questions, which I know must have been bothersome. I want to thank my colleague Olasunkanmi Oladejo as well for his assistance with the project. I pray that the Almighty would help you reach higher levels in life.

I must specifically thank Dolapo Ajayi and Niniola Oladeji for their unwavering friendship, love, and support throughout my pursuit of this degree.

Again, I will always be grateful to my loving, caring parents and siblings, whose unending support in terms of both money and moral character has served as my compass along my voyage. My successes have been fueled by your sacrifices and steadfast faith in my capability.

I must not forget to express my gratitude to my students, friends, and the rest of my peers who helped make this amazing journey possible in some way. Even though your names might not be listed here, we nevertheless remember you all. I HEART YOU ALL.

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

ROC: Receiver Operating Characteristic

CV: Computer Vision

CCTV: Closed-Circuit Television

CNN: Convolutional Neural Network

DNN: Deep Neural Networks

GPU: Graphics Processing Units

MNIST: Modified National Institute of Standards and Technology

RNN: Recurrent Neural Network

SVM: Support Vector Machine

IoT: Internet of Things

YOLO: You Only Look Once

MAP: Mean Average Precision

GSM: Global System of Mobile Communication

LFW: Labelled Faces in the Wild

PIR: Passive Infrared

RAM: Random Access Memory

CPU: Central Processing Unit

MSE: Mean Square Error

MAE: Mean Absolute Error

RMSE: Root Mean Square Error

TPR: True Positive Rate

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

ABSTRACT

The development of an Intruder Detection System has arisen as a key endeavor in an era marked by technical developments and a growing need for heightened security. This bachelor degree project uses deep learning and computer vision to build a robust system capable of detecting possible intruders in a real time system, providing increased security and safety in a variety of contexts.

Data collection, preprocessing, model construction, training, and performance evaluation are all part of the project path. The system's backbone is a Siamese neural network design known for its picture similarity assessment capabilities. A broad dataset, rigorously enhanced and curated, ensures the model's adaptability to a wide range of scenarios.

Performance indicators such as recall, precision, and ROC curves provide information about the system's effectiveness. The research concludes with the development of a dependable, real-time intruder detection system that demonstrates the potential of artificial intelligence in strengthening security measures.

Using computer vision, the intruder detection system was able to distinguish between authorized personnel and intruders, or unauthorized personnel, in real-time. It was also able to predict and verify from the supplied dataset, producing the evaluation-related outputs provided in the report.

Future efforts could enhance the system by going beyond simple detection and categorization to include an alerting function. Once more, more advanced networks may be trained to operate with complex datasets and enhance training accuracy.

Hence, this project combines cutting-edge technology, careful data science and a dedication to safety. It is a testament to the convergence of innovation and security, paving the way for future advancements and larger applications in the field of intrusion detection.

CHAPTER ONE

INTRODUCTION

1.1 Introduction

A security system that uses cutting-edge machine learning methods to identify prospective intruders or unlawful access is known as an Intruder Detection System utilizing Deep Neural Network (DNN). Inputs like sensor data, network activity, and video footage are analyzed to find patterns of behavior that might point to an intrusion.

Webcams are frequently utilized as a security standard due to the integration of facial recognition technology, which allows authorized users to unlock things such as computers, laptops, and mobile phones. Cameras play an important part in recognition technology because they can track human faces and keep track of the number of individuals in an image or a specific location, such as an entrance or public space.(Dash et al., 2021)

Intruder detection systems benefit greatly from deep neural networks because they can learn intricate patterns and features from vast amounts of data. This enables them to recognize minute behavioral differences that may point to an incursion, even if they are not immediately apparent to humans.

Systems for detecting intruders A strong and efficient method for strengthening security and preventing unauthorized access is to use DNNs.

1.2 Background Study

In the present era, surveillance systems are crucial to maintaining security. Robberies and theft by intruders result in significant losses in a variety of settings, including houses, banks, and other places. The capacity to recognize and stop illegal access, especially invasions, is one of the essential elements of security. To address this problem, intruder detection systems have been created. Deep neural networks (DNNs) can, however, be used to overcome some of the limitations of conventional intrusion detection systems.

A form of machine learning technique called deep neural networks is capable of learning intricate representations and patterns from input. They are made up of many layers of interconnected nodes that organize information according to hierarchy. Deep neural networks have been utilized for object detection, segmentation, and tracking, among other computer vision applications. When employing deep neural networks for intruder detection, a model must be trained on a dataset comprising both intruder- and non-intruder-containing pictures or videos. Based on their visual characteristics, the model learns to categorize fresh photos or videos as either invaders or non-intruders.

Deep neural networks are good in detecting intruders, according to numerous research. Face recognition has been used in a variety of applications, including automatic classroom attendance management systems (Samet & Tanriverdi, 2017)surveillance of access restricted zones including living spaces for intruder detection (Mahdi et al., 2017), recognition of celebrities in public spaces (Ouanan et al., 2018), recognition of house inmates by networked home automation systems (Zuo & De With, 2005), and many others.(Pranav & Manikandan, 2020) Despite these accomplishments, intrusion detection utilizing deep neural networks still has certain restrictions and difficulties. Lack of extensive and varied datasets for model testing and training is one of the key issues. The majority of the datasets now in use are either small or biased toward particular situations or environments. The models' resistance to changes in the weather, lighting, and occlusion presents another difficulty. Intruders might show up in various poses, locations, and scales, which could have an impact on how well the models perform. In addition, the use of surveillance systems for intruder detection raises privacy and ethical issues.

There are a number of directions that deep neural network intrusion detection research has to go in order to address these restrictions and difficulties. The creation of more resilient and adaptable models that can accommodate changes in the environment and the data is one such field. Transfer learning, data augmentation, and adversarial training can all be used to accomplish this. The development of bigger, more varied datasets that can accurately reflect the complexity and variety of invader scenarios is another field. Crowdsourcing, simulation, and cooperation between several fields can be used to accomplish this. The use of surveillance systems for intruder detection raises ethical and privacy considerations that need to be addressed. This can be achieved through the development of transparent and accountable systems, the involvement of stakeholders in the design and evaluation of the systems, and the implementation of legal and ethical frameworks.

1.3 Problem Statement

Traditional intruder detection systems are subject to a number of drawbacks, such as false alarms, insufficient coverage, and a challenge in sleight-of-hand or advanced intrusion detection. The advancement of GPU acceleration techniques, as well as the impact of deep neural networks, boost not only the accuracy, but also the popularity of face recognition systems. Although face recognition technologies help with person identification, their applications introduce a new challenge: face spoofing and presentation attacks.(Lin & Su, 2019)

The creation of more sophisticated systems, such the Intruder Detection System Using Deep Neural Networks, is necessary due to these restrictions. In order to create such a system, it is necessary to solve a number of issues, such as data quality, privacy, and bias in data collecting and preparation.Further study is needed to improve the DNN models' performance as well as determine how well they are able to identify and stop intrusions. In order to solve the drawbacks of conventional intrusion detection systems and address the difficulties involved in their creation, the research issue for this project is to design, build, and test an intrusion detection system using deep neural networks.

1.4 Aim and Objectives

Defining the success standards of this project is judged by the accuracy of the Intruder Detection System using Deep Neural Networks. From this statement the aim and objectives are constructed based on the success criteria.

Aim

The aim of this project is to develop an intruder detection system using Convolutional Neural Networks.

Objectives

The objectives of this project are as follows:

1. To collect datasets for the classification of Authorized and Intruder datasets.
2. To develop a deep learning algorithm to detect presence of intruders in the system.
3. To evaluate and visualize the developed intruder detection system.

1.5 Justification for the research

Modern security systems depend heavily on intrusion detection, yet conventional intrusion detection systems have a number of drawbacks. The main difficulties that typical intrusion detection systems encounter include false alarms, insufficient coverage, and the inability to detect sophisticated intrusions. These restrictions may be removed by the creation of an intrusion detection system utilizing deep neural networks, which would offer a more sophisticated and reliable intrusion detection system.

An important advantage of the Intruder Detection System employing Deep Neural Networks is increased security. By properly identifying and blocking intrusions, the system will improve security by lowering the possibility of unwanted access to crucial systems. The suggested system also has the advantage of fewer false alarms. With fewer false alarms, there will be less strain on resources and less need for human intervention thanks to the system. Again, the proposed system has a sizable advantage in terms of increased effectiveness. By offering better coverage and detecting sophisticated intrusions, the system will be more effective than conventional intrusion detection systems. One more advantage of the suggested system is the potential cost reduction. As a result of the system's improved effectiveness and less false alerts, hiring more security employees may not be necessary, which might save money.

Last but not least, the proposed project will further the study of deep neural network-based intrusion detection systems. The larger community will gain from the research's fresh perspectives and suggestions for enhancing the efficacy of such systems.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The existing manual security system relies heavily on human intervention, which is prone to error, and security is limited to the front entrance, which necessitates subject cooperation. (Vasuki & Priya, 2018)

Access control, security, and surveillance applications all face serious problems with intruder detection. Traditional intruder detection techniques have their limitations and are prone to false positives and false negatives. Deep Neural Networks (DNNs) have recently made significant strides, and these developments have shown promise in improving the reliability and accuracy of intrusion detection systems.

Intruder detection using DNNs is now a topic of active research, summarized in this literature review. The review details different techniques of intruder detection, their problems, and the significance of creating a successful system.

The review concludes with a summary of the major conclusions and their implications for the design of a successful DNN-based intrusion detection system. The areas that require additional study to create more sophisticated and dependable systems that can be used in practical applications are also highlighted.

2.2 Intruder Detection Techniques

Techniques used to spot and notify security staff or systems of the presence of an intruder in a restricted area are referred to as intruder detection techniques. Some of these techniques include:

2.2.1 Conventional Intruder Detection Techniques

Fig. 2.1: Conventional Intruder detection system using CCTV

Physical barriers, like fences and walls, as well as surveillance systems, such as Closed-Circuit Television (CCTV) cameras, and motion sensors, are some of the traditional intruder detection methods that have been in use for a long time. To limit access to specific places and deter intrusion, physical barriers are frequently used. These impediments include walls, fences, and gates. Physical barriers can be useful, but they are not impregnable and can be overcome with enough effort or tenacity.

The fundamental components of conventional surveillance systems (first generation) are video recording, analysis of the events that were captured, or human operators watching a live video feed (CCTV, closed-circuit television). (Gómez et al., 2015)

Conventional intruder detection methods, however, have drawbacks and are frequently prone to false positives or false negatives. False positives happen when a non-intrusive occurrence, like a limb falling or an animal, sets off an alarm. False negatives occur when an intruder is not caught, which can occur if they manage to go around or disable the detecting mechanism.

Installation and detection procedures for conventional home security systems frequently depend on the opening and closing of doors and windows. A device that can alert homeowners if someone has tried to break into their home is needed because of the rising number of theft and home invasion incidents, particularly while the victims are not at home. Therefore, the idea of a smart home system was proposed, to overcome the limitations of the systems already available in the market. The user can choose the number of sensors, types of sensors, and the area of coverage of the systems along with the number. (Nwalozie et al., 2015)

The inefficiency of traditional farming practices makes the use of guards to keep an eye on crops and animals at bay impractical. It is necessary to protect crops from animal damage while also rerouting the animals without injuring them in order to ensure the safety of both people and animals (Bhumika et al., 2022). According to Jiang et al. (2017), conventional video surveillance systems based on manual operation cannot meet the security needs of modern society. These limitations have led to a growing interest in developing more advanced intruder detection techniques using machine learning and deep neural networks (DNNs).

2.2.2 Intruder Detection Using Computer Vision

The field of computer vision focuses on the extraction, evaluation, and comprehension of data from images and videos. A method for detecting and identifying intruders or potential dangers in a specific region involves employing cameras and image processing algorithms. This method is known as "intruder detection using computer vision."

OpenCV (open-source computer vision) is the major software for detecting faces using various algorithms like Haar cascade, linear SVM, deep neural network etc. (Adhav et al., 2019)

(Landa et al., 2017), In recent years, the development of computer vision and artificial intelligence has opened up new channels to provide advanced support for home security. Intelligent video analysis technology, which comes from the field of computer vision, applies the automatic analysis of video content by using intelligent algorithms.

Nevertheless, according to (Ahanger et al., 2020), Even though computer vision is a difficult field, you can imagine how difficult it is to provide a system with an understanding of recognition, identification, imagination, and prescient. Intruder detection using computer vision has several advantages over conventional security systems. This is because with a minimal number of cameras, it can work in real-time and cover huge areas. To offer a complete security solution, it can also be coupled with other security systems, such as access control systems.

2.2.3 Intruder Detection based on Internet of Things

The Internet of Things (IoT) is a network of physical objects, including machines, cars, home appliances, and other things, that can connect to one another and share data thanks to electronics, software, sensors, and connectivity. IoT-based intruder detection entails watching the environment and looking for any prospective invaders using a variety of sensors and devices.

IoT technology, in the form of wireless cameras, motion detectors, and intelligent locking systems, substantially adds to the supply of security. The creation of smart appliances, devices, and tools to fulfill a meaningful vision of a smart home has been facilitated by the spread of IoT devices. (Ahanger et al., 2020)

The system suggested by (Sayem & Chowdhury, 2019) learns to recognize people who have been given permission to enter the designated area that is under protection. The system is trained using these saved faces. If the system identifies a face in the dataset while it is in use, the camera displays the matching name with a confidence level that may allow access, but it can also capture a picture of the individual and send it as a warning email notification. The suggested method performs well on both known and unknown datasets and can implement face recognition even from low-quality photos.

2.2.4 Intruder Detection Using Deep Neural Networks

According to (Hussain & Salim Abdallah Al Balushi, 2020), Deep learning is a subtype of machine learning in which algorithms are combined in a manner similar to machine learning, but there are countless levels of these algorithms, each of which offers a distinct interpretation of the data they incorporate. This network of algorithms is known as the "network of artificial neurons" because it was designed to emulate the behavior of the neural networks found in the brains of humans.

DNNs are powerful machine learning algorithms that can automatically learn complex representations of data, making them well-suited for intrusion detection tasks. DNN-based intrusion detection systems typically involve training a neural network model on a large dataset of labeled examples of normal and abnormal behavior and then using this model to detect anomalies in new data.

The methodology presented by (Upadhyay et al., 2020) is based on a simplified architecture that creates lightweight Deep-Neural Networks using depth-wise separable convolutions. Numerous applications and use cases, such as item identification, fine-grained classification, face traits, landmark recognition, and extensive geo-localization, can show the effectiveness of the system. While concentrating on latency optimization, it also produces tiny networks.

Furthermore, for the purpose of detecting people in infrared photos, deep learning methods based on Convolution Neural Networks (CNN) have recently been developed. Due to the improved learning capacity brought about by the increased model complexity, these deep learning-based techniques can improve object detection performance in infrared images. Park et al. (2019).

Again, according to (Hussain & Salim Abdallah Al Balushi, 2020), in recent years, the Convolution neural network has made a substantial contribution to the field of computer vision. In addition, it has made considerable strides in the recognition of objects. The ability for the technology to gain relevance was made possible by the rising cost of computers and the volume of data needed to construct a neural network.

In an effort to overcome the issue posed by the Conventional Intruder Detection Technique, (Bhumika et al., 2022) employed the deep learning concept of convolutional neural networks, a subsection of computer vision, to recognize animals as they enter farmlands. This strategy's major goal is to continuously monitor the entire farm using a camera that records the surroundings throughout the day. The CNN algorithm and XGBoost, a machine learning technique used for structured and tabular data, alert farmers when there is animal infiltration. Gradient-boosted decision trees are implemented using XGBoost, a fast and efficient method.

According to (Xenya et al., 2019), utilizing a highly sophisticated algorithm like CNN provides you an advantage over more conventional algorithms like RNN and SVM. The CNN takes inputs of fixed sizes and generates fixed-size outputs, while RNN can handle arbitrary input/output lengths. Again, Pin et al. (2021), when utilizing a big sample of the MNIST (Modified National Institute of Standards and Technology) dataset, it was found that the accuracy of SVM and CNN and 0.88 and 0.98 respectively; when using a small sample of the COREL1000 dataset, the accuracy of SVM and CNN are 0.86 and 0.83 respectively.

The ability of DNN-based intrusion detection systems to understand complex patterns and correlations in data can result in improved accuracy and fewer false positives. This is one of its key advantages. These systems, however, have several drawbacks, including the requirement for a substantial amount of labeled training data and the possibility of overfitting to the training data.

2.3 Strengths and Weaknesses of the different techniques for intruder detection

Table 2.1: Strengths and Weaknesses of the different techniques for intruder detection. (Hasan et al., 2021)

S/N	Technique	Strength(s)	Weakness(es)
1.	Conventional Intruder Detection	I. Low cost, simple to install and maintain, widely available.	I. Limited detection capability. II. Prone to false alarms. III. May require additional equipment such as security cameras. IV. May not be effective in all environments or situations, requires human intervention for response.
2.	Computer Vision for intruder detection	I. Real-time detection II. Can cover large areas with fewer cameras. III. Can be integrated with other security systems. IV. Can detect multiple intruders simultaneously. V. Can operate in various light conditions, can recognize specific objects, patterns, and behaviors	I. Requires high-quality images II. May require significant computing power and storage III. May not work well in low light or changing environments, may be affected by weather conditions or visual obstructions,

			IV. May have privacy concerns, may have limitations with object recognition in complex environments and cluttered scenes
3.	Deep Learning for intruder detection	I. High accuracy, can detect complex patterns and behaviors. II. Can be integrated with other security systems, can learn from data and improve over time. III. Can work with different types of sensors and modalities, can handle large-scale data.	I. Requires large amounts of labeled data for training. II. May be vulnerable to adversarial attacks and biases. III. Can be computationally intensive and require significant computing resources. IV. May have privacy concerns, may require regular maintenance and updates to the model.
4.	IoT for intruder detection	I. Real-time detection, can operate remotely, can be integrated with other IoT systems. II. Provides a comprehensive solution. III. Can handle different types of sensors and data sources.	I. Requires a significant initial investment. II. May be vulnerable to cyber attacks, may have privacy concerns. III. May require regular maintenance and

		IV. Can be used for various security applications.	updates to the system. IV. May have limitations in certain environments or situations. V. May require specialized knowledge to set up and operate.

2.4 Relationship between the Techniques for Intruder Detection

Table 2.2: Relationship between the Techniques for Intruder Detection. (Guo & Zhang, 2019)

Technique	Accuracy	Speed	Cost	Response time	Privacy	Data	Complexity	Integration
Conventional Intruder Detection	Low	Fast	Low	Slow	Low	Low	Low	Low
Computer Vision for intruder detection	Medium-High	Fast	Medium-High	Fast	Medium	High	High	Medium-High
Deep Learning for intruder detection	High	Slow	High	Fast	Medium-High	High(Big data)	High	Medium
IoT for intruder detection	Medium	Fast-slow	High	Fast	Medium	High	High	High

A. **Accuracy**: Compared to the other techniques, traditional intrusion detection has a low accuracy.
B. **Speed**: While deep learning systems are often slower, computer vision and Internet of Things systems offer quick detection speed.
C. **Cost**: Traditional intrusion detection systems are inexpensive, whereas computer vision, deep learning, and Internet of Things (IoT) solutions are more expensive.

D. **Response-time**: Traditional methods take longer to respond since they can need human interaction, but the other techniques have quick response times.

E. **Privacy concerns**: Deep learning systems have medium-high privacy concerns, while computer vision and Internet of Things systems and traditional systems have low privacy risks.

F. **Data**: Traditional systems demand less data than alternative methods, yet traditional systems have large data needs.

G. **Complexity**: Traditional systems are less complex than other techniques, which are more complex.

H. **Integration**: While deep learning systems have low to medium integration capabilities, computer vision and IoT systems have high integration capabilities.

NOTE: The comparison presented above is based on broad findings, and it may differ depending on how each technique is used and under what circumstances.

2.5 Benefits of the stages of Developing the Intruder detection system

Benefits of Data Preprocessing(Helin et al., 2022)

1. **Improved Convergence**: Preprocessed data allows the model to converge faster during training, leading to quicker and more efficient learning.
2. **Enhanced Generalization**: Clean and normalized inputs help the model generalize well to new and unseen data.
3. **Stable Training**: Consistent data preprocessing stabilizes the training process and reduces the likelihood of convergence issues.

Benefits of Siamese Architecture (Ye et al., 2020)

1. **Effective Learning**: The shared weights enable the network to learn discriminative features for identifying individuals, regardless of variations in lighting, pose, or appearance.
2. **Pairwise Comparisons**: By learning from pairs of images, the model learns to focus on the relevant differences and similarities between individuals.
3. **Robust Verification**: The Siamese architecture excels in verification tasks by directly comparing images and providing a similarity measure.

2.6 Comparison of the Siamese Network with Traditional Convolutional Neural Networks

Table 2.3: Comparison of Siamese network with other Traditional CNN.
(Melekhov et al., 2016)

Aspect	Siamese Network	Traditional CNN
Purpose	One-shot learning, similarity measurement	General image classification
Architecture	Two identical subnetworks (shared weights)	Single network with shared layers
Training	Contrastive loss, triplet loss	Categorical cross-entropy loss
Input Data	Pairs of similar and dissimilar samples	Single samples from different classes
Output	Embeddings (vectors in a similarity space)	Class probabilities
Training Examples	Few-shot (few pairs per class)	Many-shot (many samples per class)
Typical Use Cases	Face recognition, signature verification	Object recognition, image classification
Data Augmentation	Often less dependent on augmentation	Relies on data augmentation techniques
Inference	Compute similarity for matching purposes	Classify into predefined categories
Transfer Learning	Less common due to task specificity	Common for various image-related tasks

Aspect	**Siamese Network**	**Traditional CNN**
Example Libraries	TensorFlow, PyTorch, Keras	TensorFlow, PyTorch, Keras

2.7 Other Related Works

(Menaga et al., 2021) suggested a system that uses a Raspberry Pi as its primary computing unit, together with a camera module and numerous sensors, to identify intruders in an intelligent manner. The machine learning algorithm is trained to detect the presence of people in the field of view while the video feed is being processed in real time using image processing techniques. The device also comes with a mobile app that notifies users of intruders in real time through text messages on their smartphones. The use of a Raspberry Pi and a Pi camera for continuous video monitoring and picture acquisition to find intruders is a key component of this methodology's strength. However, it depends on image processing methods to distinguish between animals and people, which may not be 100% reliable and may result in false alarms or undetected intrusions.

In order to solve security concerns, (Shahid et al., 2017) developed a technique for automatically identifying human invaders without the need for human interaction. The suggested approach uses a Kalman filter to follow the motion and background removal to detect human invader motion. The assignment issue between the individual object and the tracked position predicted by the Kalman algorithm is subsequently addressed using the Hungarian method. This algorithm is capable of distinguishing between objects that are human and those that are not. In order to reduce noise and improve outcomes, morphological operators are used. The suggested approach works with thermal imaging for both natural light and low light situations.

Additionally, (Kalaiselvi et al., 2022) suggested a method for drone detection that entails three steps: 360-degree CCTV is used to record real-time video, a trained model is used to identify the drone in the video, and the model's performance is assessed. For object detection, the YOLO-v4 algorithm was used because of its real-time capabilities, speed, and accuracy. Photos of both birds and drones were gathered from public sources to train the neural network, and a dataset of drone photos was created to test the identification system's effectiveness. Images of birds were picked because of how closely they resemble drones. The performance of the object detector was evaluated using Mean Average Precision (MAP). The dataset that was gathered and produced was used to assess the YOLOv4 neural network's detection capacity, location accuracy, and MAP. Input, backbone, neck, and dense prediction, or head, are the four components that make up the one-stage detector architecture used by YOLOv4. The input is the data set that needs to be detected, and the backbone uses the picture dataset to extract features and build a scalable and reliable object detector. A thorough picture of the surroundings may be obtained through the 360-degree camera, guaranteeing that drones are not overlooked by the detecting system. However, during training, using bird photos in place of drone images. While birds and drones may resemble each other in certain ways, they are not identical. This could have an impact on the system's ability to recognize drones accurately, particularly if the drones utilized in real-world scenarios differ greatly from the birds used in training.

In their work, (Dixith & Jamedarnajath, 2021) developed a strategy to tackle the issue of utilizing CNN technology, which has grown in popularity for image classification which is attributable to innovative designs like Inception and ResNet-50. The system creates an Intruder Detection System using a PIR sensor and Raspberry Pi to get around the problems that many surveillances systems encounter. The camera is triggered and the image is transmitted to a registered email address when an intruder is seen in front of the PIR sensor. This methodology's heavy reliance on the PIR sensor to find the intruder is one of its biggest flaws. Due to the identification of non-human heat sources, such as dogs or even hot air currents, this sort of sensor is susceptible to false alerts.

The work by (Gurnani et al., 2019), suggested that ultrasonic sensors can be used as another technique for spotting intruders. The ultrasonic sensor sends a signal to the camera, which then turns on and takes a picture of the intruder. The intruder is then identified from the collected image using the facial recognition algorithm. Successful implementation of the Ultrasonic sensor paradigm as an intruder detector. An effective technique to present visual proof of the incursion is to utilize a camera to record the intruder's image. The system's total accuracy rate is only 89.25%, which presents a possible flaw that could lead to erroneous warnings or missed intrusions.

A security system based on PIR, ultrasonic, and GSM technology was proposed by (Siva Swetha & Saranya, 2020) and is appropriate for use in houses and other domestic situations. When movement is detected within their range, the PIR sensor and ultrasonic-based system alert the Arduino. The buzzer is subsequently activated and a message is sent via an internet connection by the Arduino to a registered mobile number. The system sets off an alarm and calls the designated number if the sensor detects any motion within its permitted range. Additionally, the range of the PIR and Ultrasonic sensors is constrained, so they can only detect objects that fall inside that range. The sensors must therefore be properly placed to cover the entire region of interest.

Additionally, (Xenya et al., 2019) presented an approach that uses cloud platforms including Twilio, Microsoft's Azure, Raspberry Pi, and convolutional neural network (CNN) to provide an intruder detection system. The classification of input data as either an invader or a user is carried out by the cloud-based CNN algorithm. With the use of Raspberry Pi's camera, learning, and classification operations may be carried out quickly and effectively while utilizing more powerful cloud computing resources. The Raspberry Pi serves as the middleware for this process. When users or intruders are spotted, the cloud system is also set up to notify certain users via multimedia message services (MMS). The study shows that, despite the high computing demands of the CNN on a processor, input data may be obtained using low-cost modules and middleware with minimal processing power while leaving the actual execution of the learning algorithm to a cloud system. The fact that this methodology needs a reliable internet connection and enough bandwidth to guarantee the effective execution of the learning algorithm on the cloud system is a possible limitation. Any lapse in internet connectivity or insufficient bandwidth could cause delays in the detection of intruders or false alarms, jeopardizing the system's effectiveness.

2.6 Summary

This chapter highlights the significant potential of DNNs in intruder detection systems, surpassing the limitations of conventional methods. The literature review provides a foundation for the subsequent chapters, guiding the development and implementation of advanced intruder detection systems using DNNs. It also identifies areas for further research, such as addressing the challenges related to data requirements and model optimization, to enhance the reliability and efficiency of DNN-based intruder detection systems.

CHAPTER 3

METHODOLOGY

3.1 Data Processing Pipeline

This outlines the step-by-step approach followed to collect, preprocess, and create the datasets for the Intruder Detection System using a Siamese network.

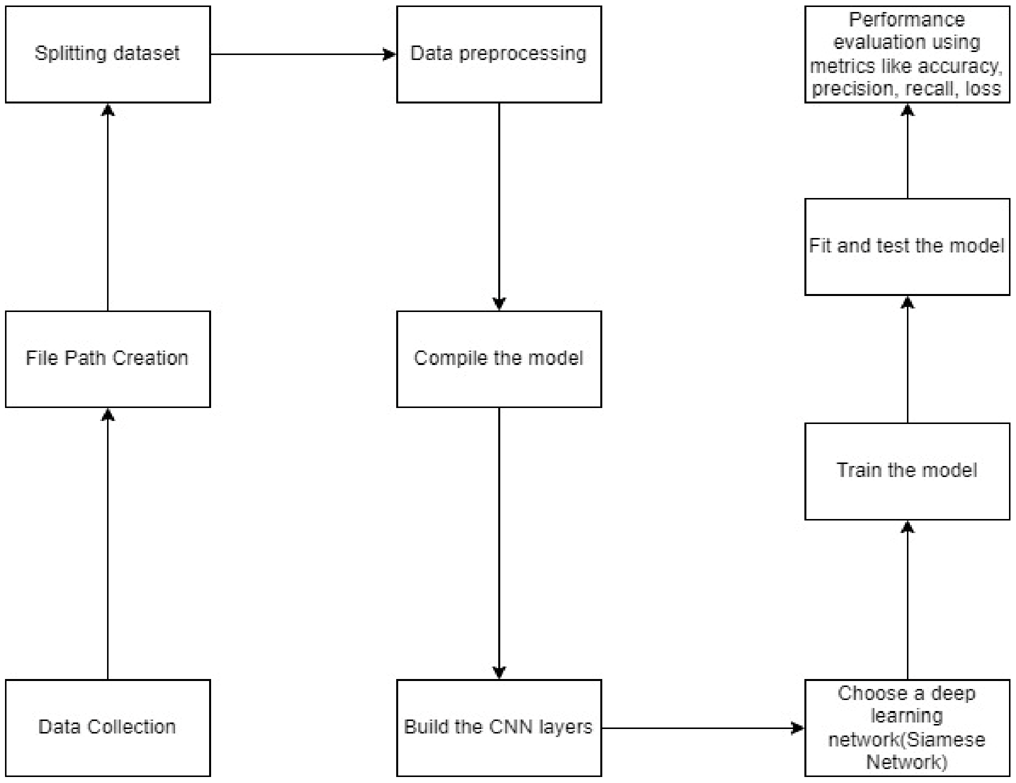


Fig. 3.1: Model Diagram

The stepwise procedures to achieve the Intruder detection system were outlined in fig. 3.1

3.1.1 Data Collection

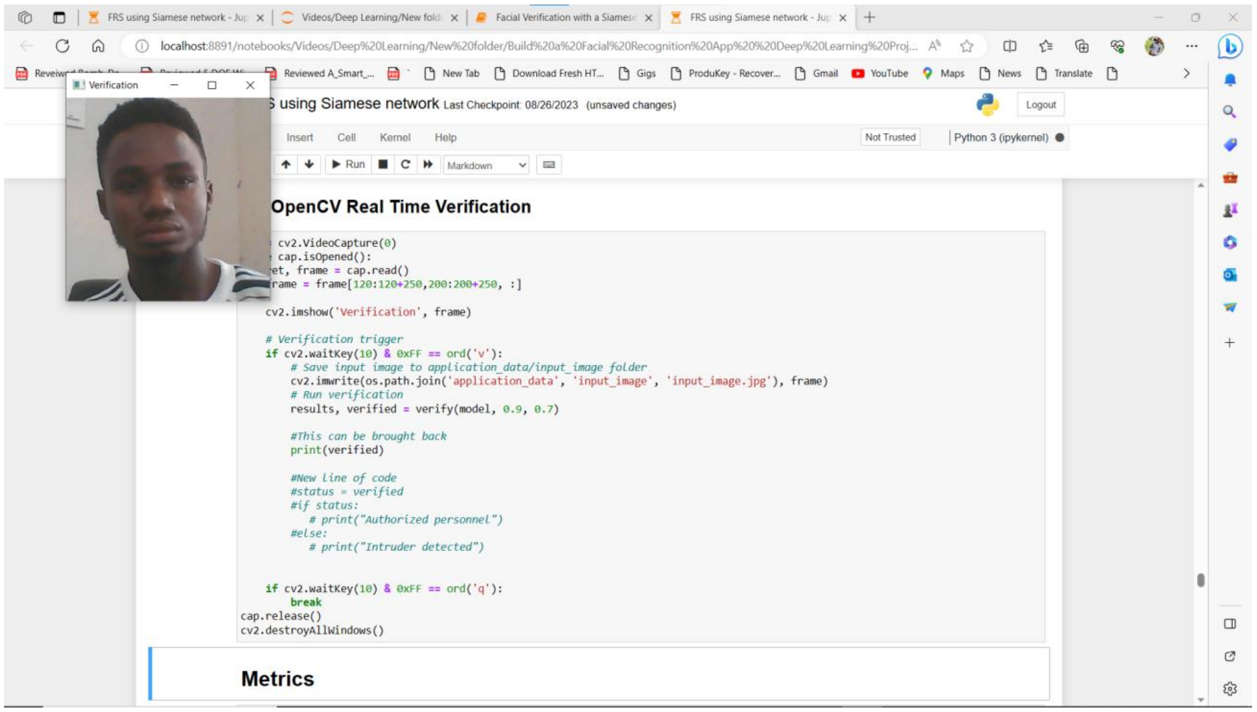
In the initial phase of the project, a dataset of images was collected to train and evaluate the Intruder Detection System. This dataset consists of three categories: anchor images, positive images, and negative images. The data collection

process was crucial in building a effective and reliable Siamese network-based system for the Intruder detection system.
The dataset for random people was obtained from Labelled Faces in the Wild(LFW) dataset, which contains a total of 13,233 images. Then, the dataset of the intended group of people as authorized personnel was acquired using the Webcam of a computer with the following specification:

System Model: HP Zbook 15u G3
BIOS: N75 Ver. 01.57
Processor: Intel® Core™ i7-6500U CPU @2.50Ghz (4 CPUs) ~2.6GHz
Memory: 16384MB RAM
Operating System: Windows 10 Pro 64-bit (10.0, Build 19045)
DirectX Version: DirectX 12

3.1.1.1 Data Collection Process

1. **Webcam Image Capture**: A webcam was utilized to record real-time photos of a group of people who would form the basis of the dataset. The camera was programmed to collect photographs of various people or things, representing both authorized personnel and prospective intruders.


S using Siamese network Last Checkpoint: 08/26/2023 (unsaved changes)

OpenCV Real Time Verification

```
 cv2.VideoCapture(0)
 cap.isOpened():
et, frame = cap.read()
rame = frame[120:120+250,200:200+250, :]

  cv2.imshow('Verification', frame)

  # Verification trigger
  if cv2.waitKey(10) & 0xFF == ord('v'):
      # Save input image to application_data/input_image folder
      cv2.imwrite(os.path.join('application_data', 'input_image', 'input_image.jpg'), frame)
      # Run verification
      results, verified = verify(model, 0.9, 0.7)

      #This can be brought back
      print(verified)

      #New line of code
      #status = verified
      #if status:
         # print("Authorized personnel")
      #else:
         # print("Intruder detected")


  if cv2.waitKey(10) & 0xFF == ord('q'):
      break
cap.release()
cv2.destroyAllWindows()
```

Metrics

Fig. 3.2: Real-time face verification using OpenCV

1. **Anchor Images**: Anchor images serve as a baseline or point of reference for the system. These photographs were taken and recorded as the first dataset. Images of authorized persons, employees, or other entities of the

designated class of people that the running algorithm should recognize as non-intruders are included

2. **Positive Images**: Positive images are photos of people or objects that the system should recognize as being comparable to the anchor images. These images aid in the training of the Siamese network to recognize features of permitted personnel.
3. **Negative Images**: Negative pictures are individuals or items that the system should treat differently than the anchor images. These images assist the network in learning to distinguish between authorized users and possible intruders.

In summary, a total of 8,132 images as Anchors; and 6,211 images as Positives were acquired using the system with the defined specifications.

3.1.1.2 Dataset Organization

To keep the collection organized and structured, distinct folder directories were constructed for each image category: anchor, positive, and negative.

Data Organization Workflow

The following workflow summarizes the data organization process:

1. Captured images are categorized into anchor, positive, or negative based on their nature.
2. Each image is saved in its respective directory for future use in model training and evaluation.

3.1.2 Data Preprocessing

After collecting a diverse dataset of images, these images were preprocessed to prepare them for model training, by scaling and tuning these images so that have the same specifications.

Data preprocessing is a crucial stage that ensures the Siamese network receives clean and normalized inputs, enhancing the system's performance and accuracy.

3.1.2.1 Data Preprocessing Process

1. **Image Loading**: The collected images were loaded using the TensorFlow library. This allowed access and manipulation of the image's pixel data.
2. **Resizing**: To ensure uniformity in input dimensions, all images were resized to a common size, 250 pixels by 250 pixels. This step is important for consistent processing and compatibility with the model architecture.
3. **Scaling and Normalization**: Scaling the pixel values of the images to a range of [0, 1] by dividing them by 255. This normalization ensures that the model can handle input data within a manageable range.

3.1.2.2 Dataset Preprocessing Workflow

The following workflow summarizes the data preprocessing process:

1. Images are loaded using TensorFlow's image-loading functions.
2. Images are resized to a common size (in this case 250x250 pixels).
3. Pixel values are scaled and normalized for consistent processing.

3.1.3 Dataset Creation

Creating a well-structured and balanced dataset is essential for training an effective Intruder Detection System using a Siamese network. In this stage, data was explored and organized into anchor, positive, and negative pairs to facilitate model learning.

3.1.3.1 Dataset Creation Process

1. **Positive Pairs**: Positive pairs consist of anchor images and corresponding positive images. These pairs help the model learn to recognize authorized personnel based on similar characteristics.
2. **Negative Pairs**: Negative pairs comprise anchor images and non-corresponding negative images. These pairs teach the model to distinguish between authorized individuals and potential intruders.
3. **Dataset Combination**: By combining positive and negative pairs, a comprehensive training dataset was created that captures both similarities and differences.

3.1.4 Model Architecture

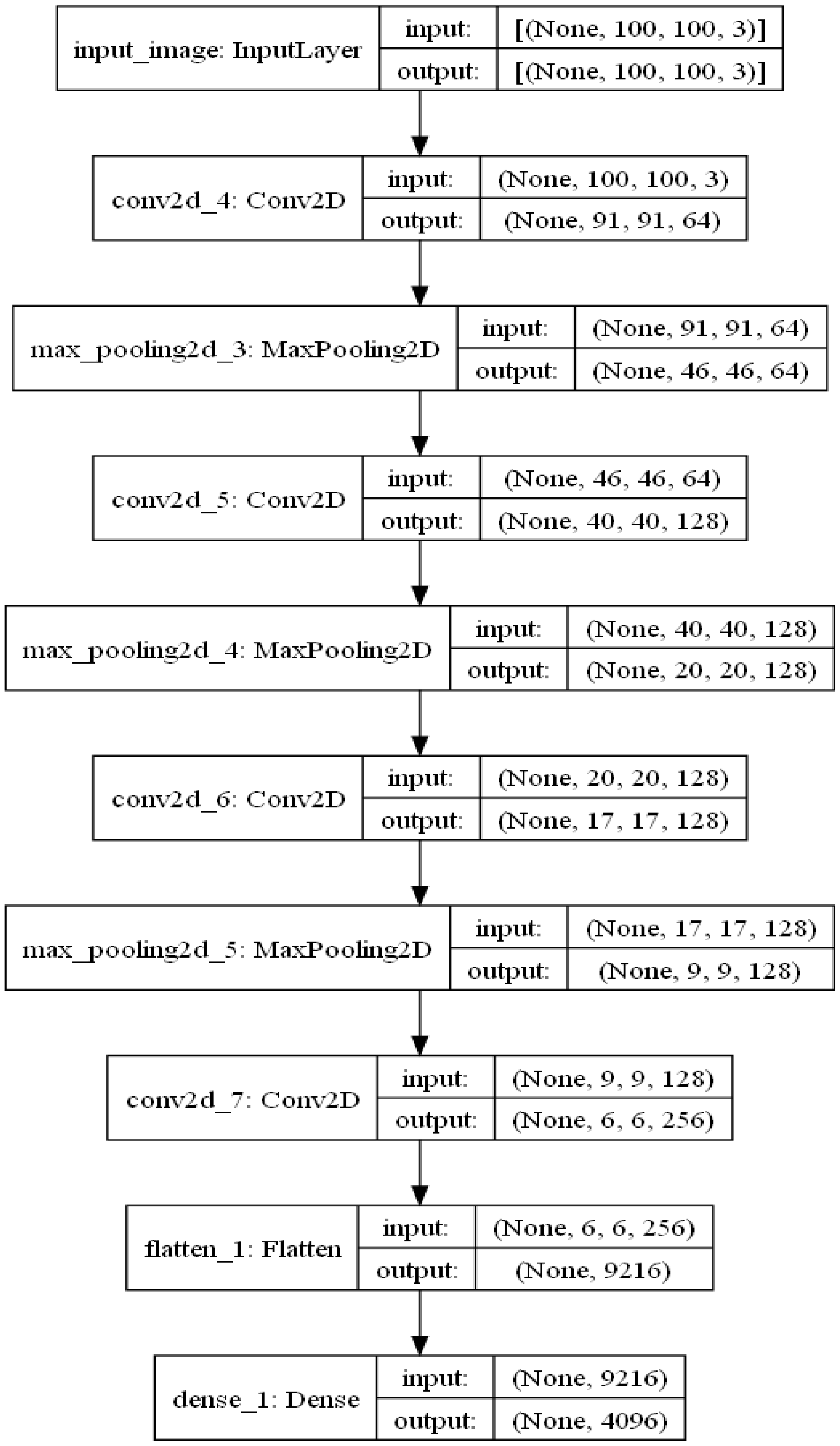
The success of the Intruder Detection System hinges on a well-designed model architecture. Fig. 3.1 describes the architecture of our Siamese network, which forms the core of our system's ability to identify and verify individuals.

3.1.4.1 Siamese Network Architecture

The Siamese network is a neural architecture designed to compare and contrast input pairs. It consists of two identical subnetworks that share weights, enabling the network to learn representations from pairs of images.

3.1.4.2 Embedding Subnetwork

The embedding subnetwork is responsible for transforming input images into meaningful feature vectors. It comprises multiple convolutional layers followed by max-pooling and fully connected layers.



input_image: InputLayer
input: [(None, 100, 100, 3)]
output: [(None, 100, 100, 3)]
conv2d_4: Conv2D
input: (None, 100, 100, 3)
output: (None, 91, 91, 64)
max_pooling2d_3: MaxPooling2D
input: (None, 91, 91, 64)
output: (None, 46, 46, 64)
conv2d_5: Conv2D
input: (None, 46, 46, 64)
output: (None, 40, 40, 128)
max_pooling2d_4: MaxPooling2D
input: (None, 40, 40, 128)
output: (None, 20, 20, 128)
conv2d_6: Conv2D
input: (None, 20, 20, 128)
output: (None, 17, 17, 128)
max_pooling2d_5: MaxPooling2D
input: (None, 17, 17, 128)
output: (None, 9, 9, 128)
conv2d_7: Conv2D
input: (None, 9, 9, 128)
output: (None, 6, 6, 256)
flatten_1: Flatten
input: (None, 6, 6, 256)
output: (None, 9216)
dense_1: Dense
input: (None, 9216)
output: (None, 4096)

Fig. 3.3: Siamese Network Architecture for Intruder Detection

3.1.4.3 Distance Calculation

The network calculates the L1 distance between the embeddings of anchor and validation images. The distance metric captures the dissimilarity between the images.

3.1.4.4 Classification Layer

The distance output from the L1 calculation is fed into a classification layer, where it's used to determine whether the images belong to the same class (positive pair) or different classes (negative pair).

3.1.4.5 Model Summary

Here's a summary of the key components of the developed Siamese network:

- Input: Pairs of images (anchor and validation)
- Architecture: Twin embedding subnetworks with shared weights
- Loss Function: Binary Cross-Entropy
- Optimizer: Adam Optimizer

Layer (type)	Output Shape	Param #	Connected to
input_img (InputLayer)	[(None, 100, 100, 3)	0	
validation_img (InputLayer)	[(None, 100, 100, 3)	0	
embedding (Functional)	(None, 4096)	38960448	input_img[0][0] validation_img[0][0]
distance (L1Dist)	(None, 4096)	0	embedding[2][0] embedding[3][0]
dense_3 (Dense)	(None, 1)	4097	distance[0][0]

Total params: 38,964,545
Trainable params: 38,964,545

Non-trainable params: 0

Table 3.1: Model Summary of Siamese Network for Intruder Detection System

The Siamese network architecture underpins the intelligence of the Intruder Detection System. By leveraging the power of shared weights, embedding subnetworks, and distance calculations, the system can accurately identify authorized individuals and distinguish them from potential intruders.

3.1.5 Training Preparation

It's important to prepare the data and model to ensure a smooth and effective training process.

3.1.5.1 Data Partitioning

To facilitate training and evaluation, the dataset was divided into two main partitions: the training partition and the testing partition.

3.1.5.2 Training Partition

The training partition contains 70% of the dataset, which the model uses to learn and optimize its parameters.

3.1.5.3 Testing Partition

The testing partition which is 30% of the dataset, is reserved for evaluating the trained model's performance on unseen data, providing insights into its generalization capabilities.

3.1.5.4 Batch Processing

During training, the data was processed in batches to efficiently update the model's weights.

3.1.5.5 Batch Size

We configured the batch size to 32 images, which determine the number of samples processed in each training iteration.

3.1.5.6 Data Preprocessing and Model Compilation Workflow

The following workflow summarizes the training preparation process:

1. The dataset is divided into training and testing partitions.

2. Training data is processed in batches for efficient training.
3. Images in each batch are preprocessed using resizing and normalization.
4. The model is compiled with a loss function, optimizer, and evaluation metrics.

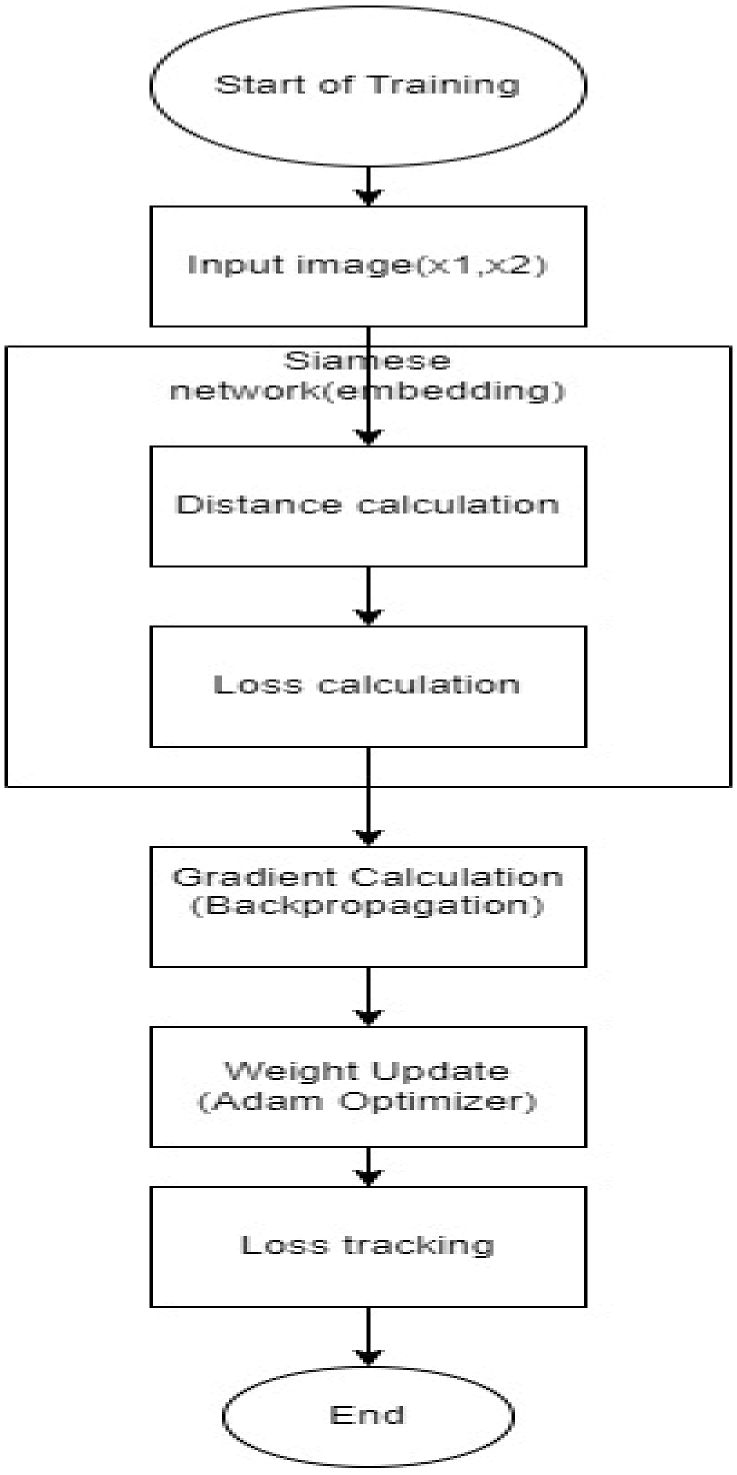

Start of Training
Input image(x1,x2)
Siamese network(embedding)
Distance calculation
Loss calculation
Gradient Calculation (Backpropagation)
Weight Update (Adam Optimizer)
Loss tracking
End

Fig. 3.4: Training Data for Intruder Detection system

3.1.6 Model Training

The heart of the Intruder Detection System lies in training the Siamese network to accurately distinguish authorized personnel from potential intruders. In this section, the key stages and components of the model training process were explored.

3.1.6.1 Training Workflow

Model training process involves iterative steps to update the model's weights based on the training data.

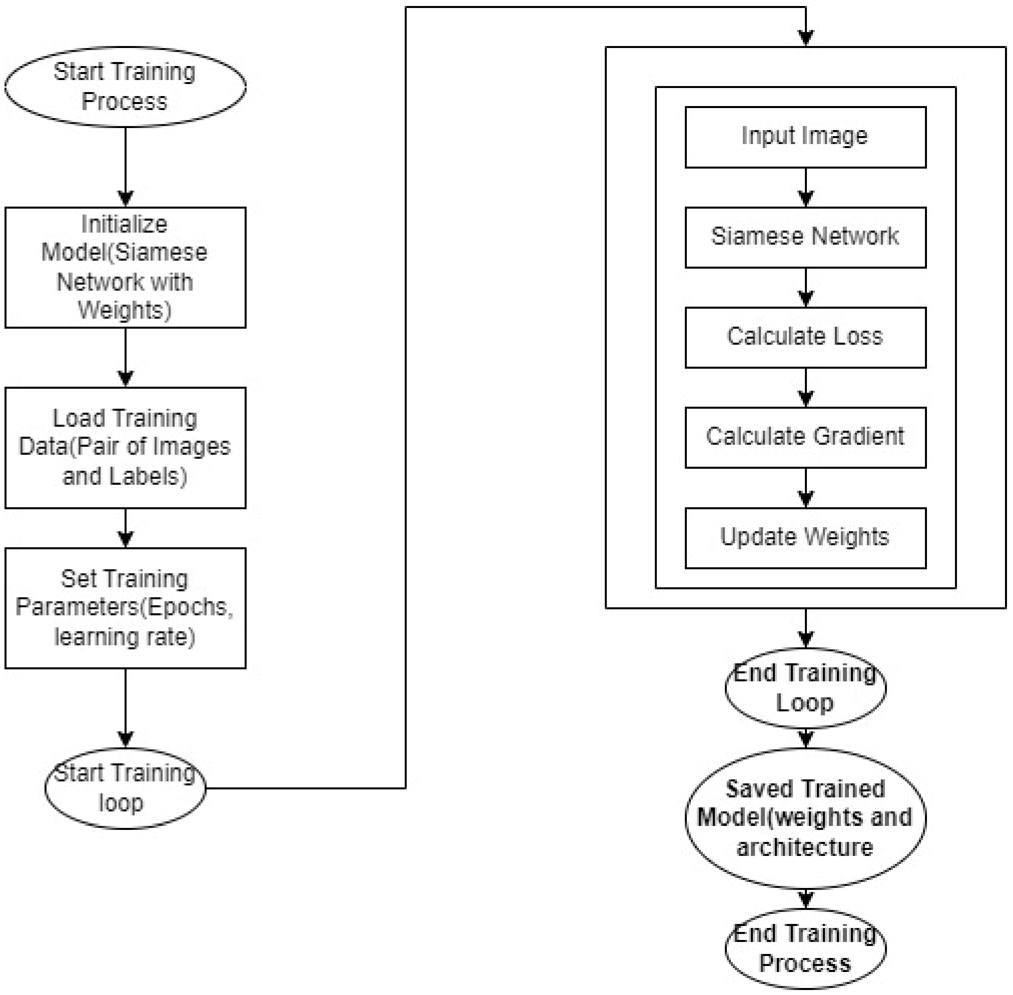

Start Training Process
Initialize Model(Siamese Network with Weights)
Load Training Data(Pair of Images and Labels)
Set Training Parameters(Epochs, learning rate)
Start Training loop
Input Image
Siamese Network
Calculate Loss
Calculate Gradient
Update Weights
End Training Loop
Saved Trained Model(weights and architecture
End Training Process

Fig. 3.5: Training Workflow of the Intruder detection system

3.1.6.2 Loss Function

The use the Binary Cross-Entropy loss function to quantify the difference between predicted and actual labels.

3.1.6.3 Optimizer

The Adam optimizer is employed to adjust model weights and minimize the loss function.

3.1.6.4 Training Step

The training step involves computing gradients, applying them to update weights, and tracking the loss.

3.1.6.5 Training Progress

Monitoring the model's progress by observing the loss reduction over epochs.

3.1.6.6 Hyperparameters

Hyperparameters like learning rate, batch size, and the number of epochs influence the training process.

3.1.6.7 Validation

During training, the model's performance on the validation set was evaluated to prevent overfitting.

3.1.6.8 Training Results

At the end of training, the model's performance on the testing partition to gauge its real-world capabilities was assessed..

The model training process equips the Intruder Detection System with the ability to make accurate predictions based on learned features. Through continuous optimization, validation, and evaluation, the system hones its skills to differentiate authorized personnel from potential intruders.

3.2 Software Stack

Using a carefully chosen software stack that includes a variety of tools, libraries, and frameworks in the creation of the Intruder Detection System. From data processing through model training and evaluation, this software ecosystem was essential throughout the project.

3.2.1 Python Programming Language

The Python programming language, which is recognized for its adaptability and simplicity of use, lies at the heart of the software stack. The implementation of the various parts of this project, including data preprocessing, model architecture, and training, was done using Python..

3.2.2 Data Collection and Preprocessing Tools

We employed various data collection tools to capture anchor, positive, and negative images. Additionally, we used Python's libraries for image loading, resizing, and normalization to preprocess the data before feeding it into the model.

3.2.2.1 TensorFlow and Keras

We harnessed the power of TensorFlow, an open-source machine learning framework, to build and train the Siamese network model. TensorFlow's robust ecosystem enabled seamless integration with Keras, a high-level neural networks API, allowing us to design, compile, and train complex models with ease.

3.2.2.2 OpenCV

OpenCV, a library primarily used for computer vision tasks, played a crucial role in handling image input and manipulation. We utilized OpenCV to capture webcam feeds, preprocess images, and extract essential features for training the model.

3.2.2.3 Matplotlib

Visualizing data and results is an integral part of any project. Matplotlib, a versatile plotting library, allowed the create informative visualizations of training progress, loss curves, and image samples.

3.2.2.4 Pydot and Graphviz

To visualize the architecture of the Siamese network, utilizing Pydot and Graphviz. These tools allowed to generate clear and concise diagrams of the model's structure, aiding in better understanding and communication.

3.2.3 Code Editors

Throughout the project, code editors such as Visual Studio Code and Jupyter Notebook. provided a comfortable environment for writing, editing, and organizing the codebase.

3.2.3.1 Jupyter Notebook

For interactive development and experimentation, we turned to Jupyter Notebook. This web-based environment facilitated code execution in a step-by-step manner, enabling us to iterate on model design, visualize data, and analyze results efficiently.

3.2.4 Version Control with Git

Git served as the version control system, enabling collaborative development and seamless tracking of code changes. By utilizing Git, we ensured code integrity and facilitated teamwork.

3.2.5 Deployment

While not covered in this project's overview, potential deployment tools and platforms like Docker, cloud services, and hosting providers could play a role in making the Intruder Detection System accessible to end-users.

The software stack, carefully selected to cater to the project's needs, enabled us to seamlessly navigate the complexities of creating an Intruder Detection System. Through the integration of Python, TensorFlow, OpenCV, and various other tools, we were able to streamline data processing, model training, and visualization, ultimately leading to the successful implementation of the intelligent system.

3.3 Performance Judgement

The performance metrics provide insight into how well the model is performing and whether it meeting the desired criteria. This was done using the following metrics.

3.3.1 Accuracy

Accuracy is one of the performance evaluation measures in machine learning. It is the percentage of correct predictions made by the model. This is also known as the number of true negatives and true positives divided by the number of true positives (TP), true negatives (TN), false negatives (FN), and false positives (FP). A true positive or true negative data point is one that was successfully classified by the algorithm, whereas a false negative or false positive data point was one that was wrongly identified by the algorithm. The accuracy calculation formula is presented below.

Accuracy = $\frac{TN+TP}{TP+TN+FP+FN}$

3.3.2 Precision

This is one of the machine learning algorithm's performance indicators; it is the accuracy of a positive prediction generated by the algorithm. It is the TP number divided by the overall number of positive predictions, i.e., TP and FP. The precision formula is shown below.

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

3.3.3 Recall

The true positive rate (TPR) is another name for this measure. It is the percentage of data samples that an ML model correctly classifies as belonging to a class of interest, which is the positive class out of all class samples. The recall calculation formula is shown below.

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

3.3.4 F1-Score

The F1 Score is the harmonic mean of precision and recall and provides a balanced assessment of a model's performance. It is particularly useful when there is an imbalance between the classes (intruders vs. non-intruders). An F1

Score closer to 1 indicates a well-balanced trade-off between precision and recall, signifying that your system can identify intruders effectively while minimizing false alarms.

3.3.5 Specificity

The trade-off between the true negative rate (specificity) and the false positive rate at different threshold levels is depicted graphically by the specificity curve. You can see how changes in the threshold affect the system's capacity to limit false alerts while retaining high accuracy in recognizing non-intruders by charting specificity versus various thresholds.

CHAPTER FOUR

RESULTS AND DISCUSSION

The initial objective encompasses the acquisition of a dataset dedicated to the classification of authorized personnel and intruders. Subsequently, the second objective is centered on the formulation of a deep learning algorithm, specialized in the identification of intruders within the aforementioned datasets.

Furthermore, the third objective involves the development of a real-time validation and testing mechanism for both the dataset and individuals employing the aforementioned algorithm. This process includes an assessment of performance metrics, including accuracy, recall, and precision, as well as the generation of a confusion matrix. Cost metrics, comprising mean square error (MSE), mean absolute error (MAE), and root mean square error (RMSE), are also incorporated.

To facilitate an enhanced understanding of the outcomes, graphical representations and tabular data are employed for result visualization.

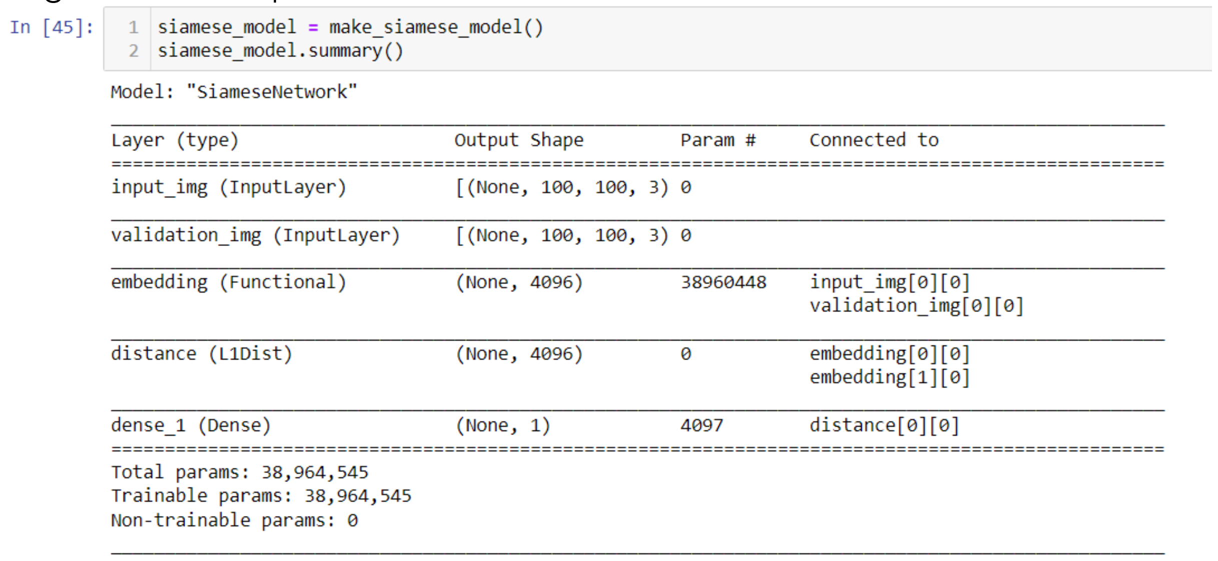
4.1 Source Of Data

The dataset utilized in this study was sourced from two distinct origins. Firstly, the Negative dataset, consisting of 13,278 images, was derived from a collection of random individuals accessible through the Labeled Faces in the Wild Home, which was established in 2019 at the University of Massachusetts. Secondly, the Positive dataset, comprising 4,922 images, and the Anchor dataset, comprising 7,309 images, were meticulously compiled and formed from a group of finalists within the Department of Mechatronics Engineering at the First Technical University, Ibadan.

4.2 Developed Model Summary

Using the finished data, a CNN-based Intruder Detection model was created. Due to the analysis of aim 1 (to collect datasets for the categorization of Authorized and Intruder datasets), Anaconda, Jupyter Notebook, and libraries such as pandas, NumPy, matplotlib, dataset were utilized.

The figure below depicts how the model was created.



```
In [45]:  siamese_model = make_siamese_model()
          siamese_model.summary()
```

```
Model: "SiameseNetwork"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_img (InputLayer)          [(None, 100, 100, 3) 0

__________________________________________________________________________________________________
validation_img (InputLayer)     [(None, 100, 100, 3) 0

__________________________________________________________________________________________________
embedding (Functional)          (None, 4096)         38960448    input_img[0][0]
                                                                 validation_img[0][0]
__________________________________________________________________________________________________
distance (L1Dist)               (None, 4096)         0           embedding[0][0]
                                                                 embedding[1][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 1)            4097        distance[0][0]
==================================================================================================
Total params: 38,964,545
Trainable params: 38,964,545
Non-trainable params: 0
__________________________________________________________________________________________________
```

Fig. 4.1 Model summary of the Siamese Neural Network

4.3 Test And Validate The Model Algorithm

Metrics for Evaluating CNN Performance, the CNN method was used to test and validate the model algorithm, which yielded average results of 148.87m per time, 94.97% accuracy, 0.9763 precision, and 0.9578 recall. These indicators aided in providing a comprehensive assessment of the system's performance.

The following tables shows the validation metrics in order of epochs and average on the datasets.

Table 4.2: Training results on the metrics

Epoch Number	Loss	Accuracy (%)	Precision	Recall	Time(m)
1	0.3844	88.83%	0.9230	0.9251	135
2	0.2728	90.21%	0.9712	0.9002	134.16
3	0.3041	90.75%	0.9372	0.9425	99.33
4	0.2238	91.45%	0.9799	0.9124	113.52
5	0.1475	95.37%	0.9785	0.9615	104.58
6	0.0995	98.59%	0.9921	0.9880	163
7	0.1008	98.55%	0.9937	0.9873	85.07
8	0.1030	98.95%	0.9933	0.9917	85.05
9	0.0841	97.75%	0.9957	0.9770	85.3
10	0.0606	99.29%	0.9980	0.9924	84.5

Table 4.3: Average Metrics Results on the dataset

S/N	Metrics	Average
1	Loss	0.1781
2	Accuracy	94.97%
3	Precision	0.9763
4	Recall	0.9578

The following figures show the graphical representation derived from the training of the Siamese Network for Intruder Detection using Deep Neural network

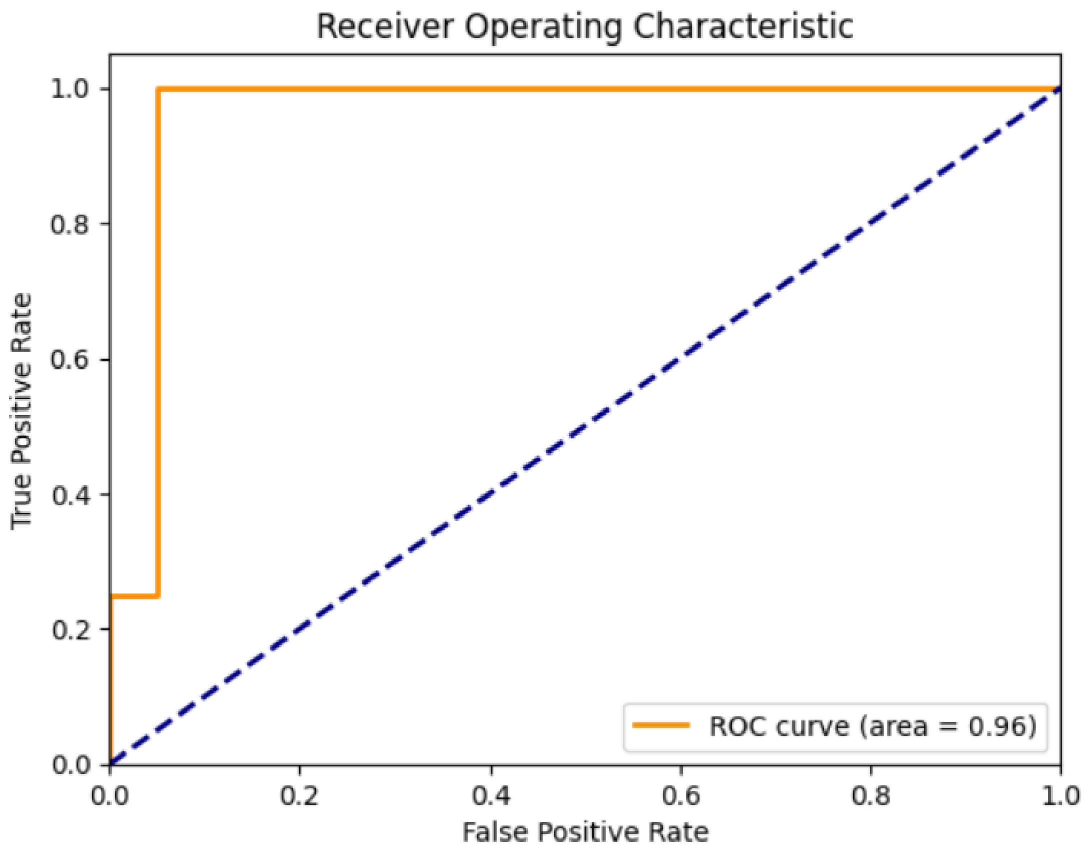


Fig. 4.2: Receiver Operating Characteristic graph for the Intruder detection system.

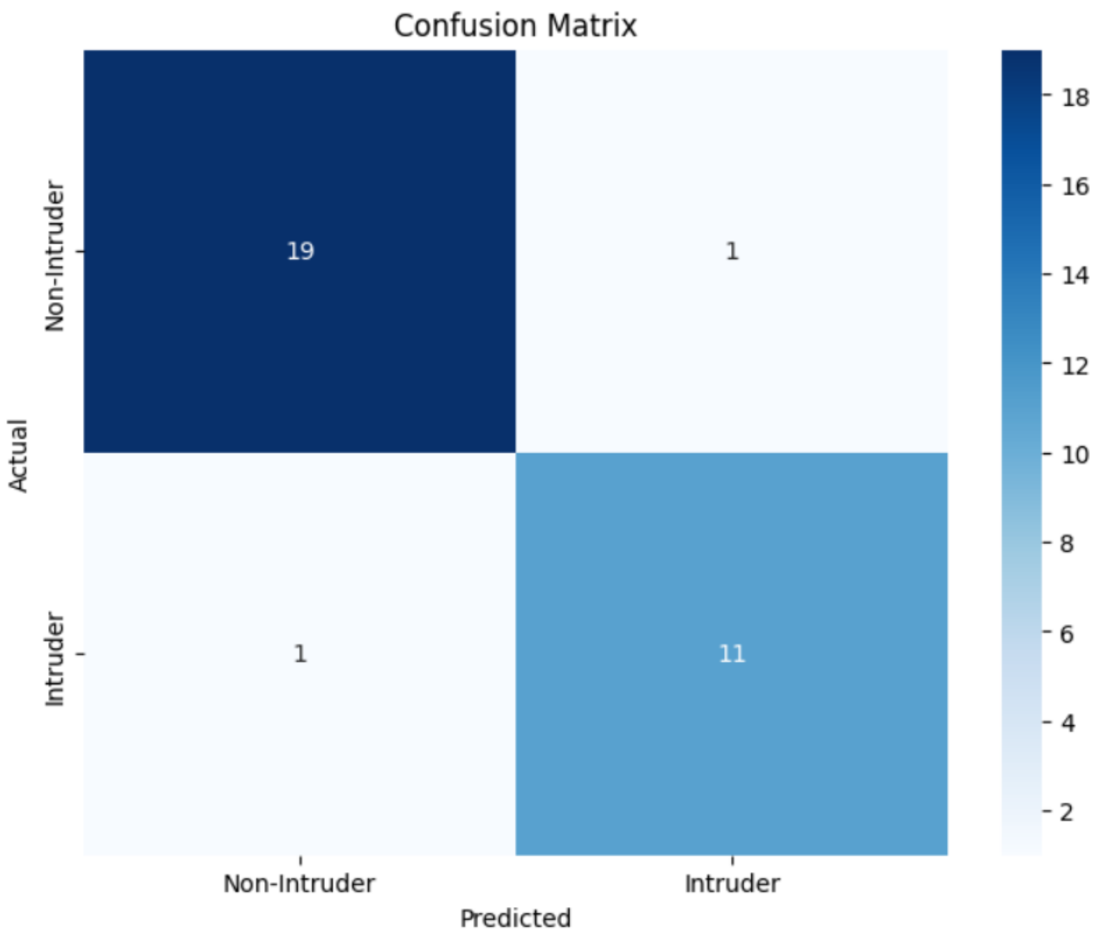


Fig. 4.3: Confusion matrix of the Intruder Detection system.

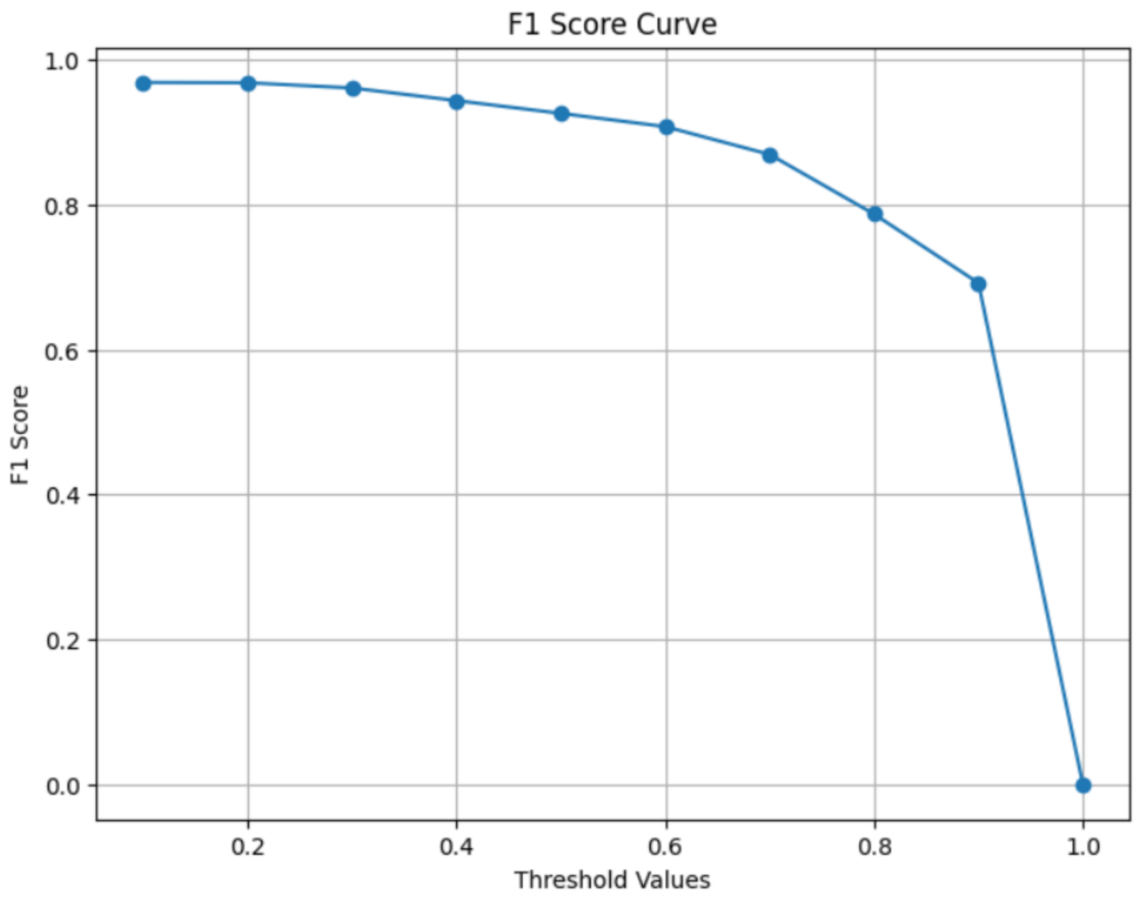


Fig. 4.4: F1-score graph

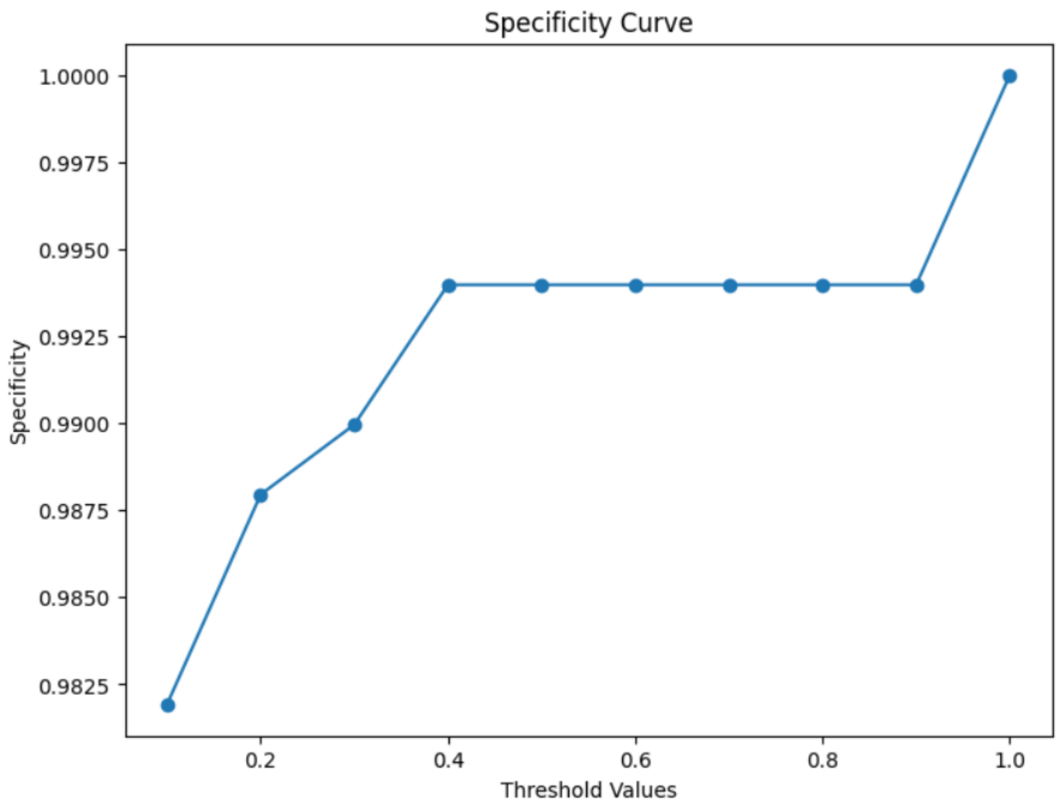
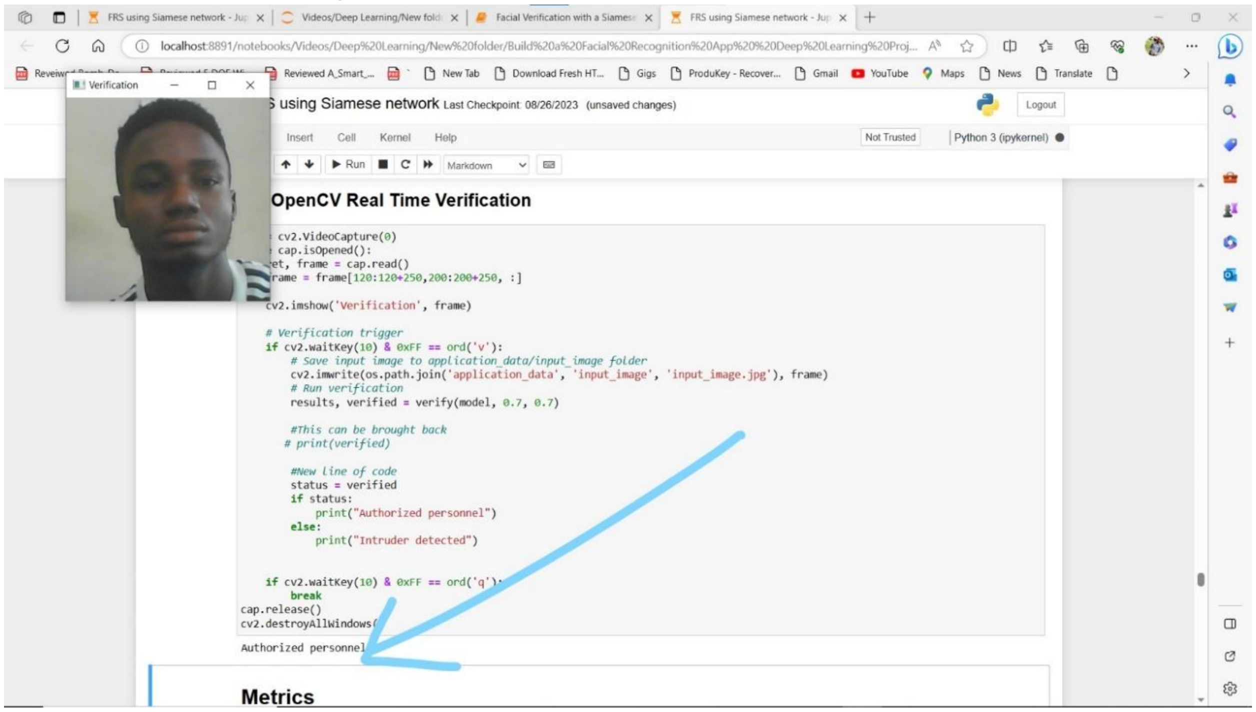


Fig. 4.5: Specificity Curve

4.4 Real-Time Testing Results



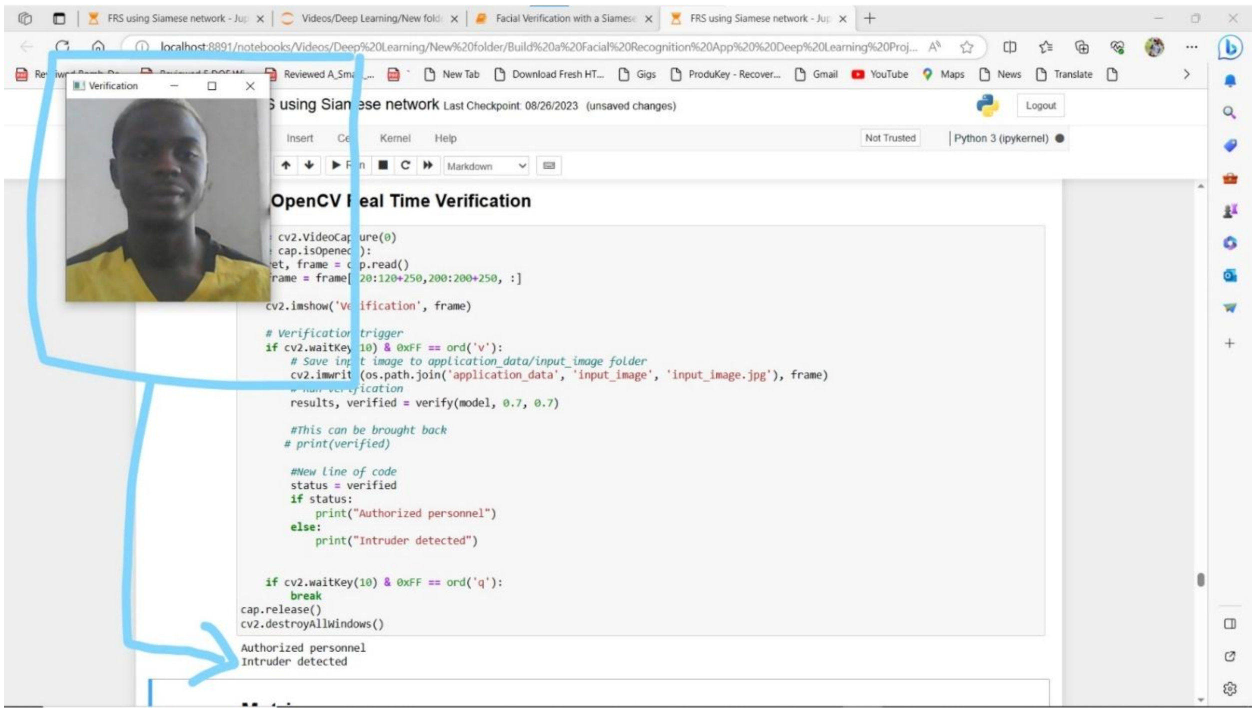
OpenCV Real Time Verification

```
 cv2.VideoCapture(0)
 cap.isOpened():
et, frame = cap.read()
rame = frame[120:120+250,200:200+250, :]

cv2.imshow('Verification', frame)

# Verification trigger
if cv2.waitKey(10) & 0xFF == ord('v'):
    # Save input image to application_data/input_image folder
    cv2.imwrite(os.path.join('application_data', 'input_image', 'input_image.jpg'), frame)
    # Run verification
    results, verified = verify(model, 0.7, 0.7)

    #This can be brought back
   # print(verified)

    #New line of code
    status = verified
    if status:
        print("Authorized personnel")
    else:
        print("Intruder detected")


if cv2.waitKey(10) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows(
```

```
Authorized personnel
```

Metrics

Fig. 4.6: Real-time testing (authorized personnel)



OpenCV eal Time Verification

```
: cv2.VideoCap ure(0)
: cap.isOpenec ):
et, frame = c p.read()
rame = frame[ 20:120+250,200:200+250, :]

  cv2.imshow('Ve ification', frame)

  # Verificatior trigger
  if cv2.waitKey 10) & 0xFF == ord('v'):
      # Save inp t image to application_data/input_image folder
      cv2.imwrit (os.path.join('application_data', 'input_image', 'input_image.jpg'), frame)
              ication
      results, verified = verify(model, 0.7, 0.7)

      #This can be brought back
     # print(verified)

      #New line of code
      status = verified
      if status:
          print("Authorized personnel")
      else:
          print("Intruder detected")


  if cv2.waitKey(10) & 0xFF == ord('q'):
      break
cap.release()
cv2.destroyAllWindows()
```

```
Authorized personnel
Intruder detected
```

Fig. 4.7: Real-time testing (Intruder)

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Recommendations

The project demonstrated a solid framework for detecting and warning on the presence of intruders in a given environment. However, there is always space for improvement and refinement in any technology endeavor. Here are some suggestions for improving the system:

1. Transfer Learning: Examining the possibilities of transfer learning using pre-trained models such as VGG, Inception, or MobileNet. Transfer learning can reduce training time while increasing accuracy.
2. Real-time Alerting: Integrate real-time alerting techniques to expand the system's capabilities. When an incursion is identified, consider sending email notifications, SMS alerts, or even push notifications to mobile devices.
3. User Interface (UI) Enhancements: Enhance the user interface as needed to make it more user-friendly and intuitive. Include tools like a dashboard for viewing past intrusion data and adjusting system parameters.

5.2 Conclusions

Finally, the creation and deployment of the Intruder Detection System represent an important step in improving security and safety in a variety of environments. Using cutting-edge technology such as deep learning and computer vision, being able to successfully construct a system capable of detecting and alerting to prospective intruders in real-time. The system's adaptability to many circumstances and lighting conditions demonstrates its versatility and potential for larger application.

Throughout the project's path, challenges were overcome, algorithms were fine-tuned, and the model was constantly refined to achieve an outstanding degree of accuracy and reliability. The successful integration of performance measurements such as recall, precision, and ROC curves enabled a full assessment of the system's capabilities.

There is still plenty of space for development and growth of this system in future investigations. Recommendations like improved architectures and real-time warning methods provide a road map for future improvements. Collaboration with security professionals and end users will be invaluable in guaranteeing the system's continued efficacy.

This project not only produced a working Intruder Detection System, but it also demonstrated the potential of artificial intelligence in enhancing security measures. It exemplifies the possibilities that emerge when cutting-edge technology meets the critical requirement for safety and alertness.

REFERENCES

Adhav, S., Manthira Moorthi, R., & Prethija, G. (2019). IMAGE PROCESSING BASED INTRUDER DETECTION USING RASPBERRY PI. *International Journal of Trendy Research in Engineering and Technology*, *3*. www.trendytechjournals.com

Ahanger, T. A., Tariq, U., Ibrahim, A., Ullah, I., & Bouterra, Y. (2020). Iot-inspired framework of intruder detection for smart home security systems. *Electronics (Switzerland)*, *9*(9), 1–17. https://doi.org/10.3390/electronics9091361

Dash, S. S., Das, S., Ketan, B., & Editors, P. (n.d.). *Advances in Intelligent Systems and Computing 1172 Intelligent Computing and Applications Proceedings of ICICA 2019*. http://www.springer.com/series/11156

Dixith, M., & Jamedarnajath, M. (2021). Intruder Detection and Alerting Mechanism Using CNN and Iot. *International Journal of Advances in Engineering and Management (IJAEM)*, *3*, 3142. https://doi.org/10.35629/5252-030731423146

Gómez, M. J., García, F., Martín, D., De La Escalera, A., & Armingol, J. M. (2015). Intelligent surveillance of indoor environments based on computer vision and 3D point cloud fusion. *Expert Systems with Applications*, *42*(21), 8156–8171. https://doi.org/10.1016/j.eswa.2015.06.026

Guo, G., & Zhang, N. (2019). A survey on deep learning based face recognition. *Computer Vision and Image Understanding*, *189*. https://doi.org/10.1016/j.cviu.2019.102805

Gurnani, D., Gaikwad, V., & Gurnani, P. V. (2019). Intruder Detection and Recognition System. In *International Journal of Computer Applications* (Vol. 178, Issue 20).

Hasan, M. K., Ahsan, M. S., Abdullah-Al-Mamun, Newaz, S. H. S., & Lee, G. M. (2021). Human face detection techniques: A comprehensive review and future research directions. *Electronics (Switzerland)*, *10*(19). https://doi.org/10.3390/electronics10192354

Helin, R., Indahl, U. G., Tomic, O., & Liland, K. H. (2022). On the possible benefits of deep learning for spectral preprocessing. *Journal of Chemometrics*, *36*(2). https://doi.org/10.1002/cem.3374

Hussain, S. A., & Salim Abdallah Al Balushi, A. (2020). A real time face emotion classification and recognition using deep learning model. *Journal of Physics: Conference Series*, *1432*(1). https://doi.org/10.1088/1742-6596/1432/1/012087

Institute of Electrical and Electronics Engineers. (n.d.). *The 2nd International Conference on Computer and Communication Systems : ICCCS 2017 : July 11-14, 2017, Krakow(Cracow), Poland.*

K Bhumika, G Radhika, & CH Ellaji. (2022). Detection of animal intrusion using CNN and image processing. *World Journal of Advanced Research and Reviews*, *16*(3), 767–774. https://doi.org/10.30574/wjarr.2022.16.3.1393

Kalaiselvi, V. K. G., Poorna Pushkala, K., Shanmugasundaram, H., & Sivadarshini, R. (n.d.). DRONE DETECTION USING DEEP LEARNING. *EPRA International Journal of Multidisciplinary Research (IJMR)-Peer Reviewed Journal*. https://doi.org/10.36713/epra2013

Landa, J., Jun, C., & Jun, M. (2017). *Implementation of a remote real-time surveillance security system for intruder detection*. https://doi.org/10.1109/ICMTMA.2017.31

Lin, H. Y. S., & Su, Y. W. (2019). Convolutional neural networks for face anti-spoofing and liveness detection. *2019 6th International Conference on Systems and Informatics, ICSAI 2019*, 1233–1237. https://doi.org/10.1109/ICSAI48974.2019.9010495

Mahdi, F. P., Habib, M. M., Ahad, M. A. R., McKeever, S., Moslehuddin, A. S. M., & Vasant, P. (2017). Face recognition-based real-time system for surveillance. *Intelligent Decision Technologies*, *11*(1), 79–92. https://doi.org/10.3233/IDT-160279

Melekhov, I., Kannala, J., & Rahtu, E. (n.d.). *Siamese Network Features for Image Matching*.

Menaga, S., Priyadharshini, A., Subalakshmi, V., Priyadharshini, J., & Velammal, P. (n.d.). *A Smart Intruder Detection System*. www.ijert.org

Nwalozie, G., Aniedu, A., & S, N. C. (2015). *Enhancing Home Security Using SMS-based Intruder Detection System Jamming Attack analysis and Design of Improved Security and Privacy Framework for Internet of Things View project Development and Implementation of RFID Car Tracking System Using Arduino Uno View project*. https://www.researchgate.net/publication/292616959

Ouanan, H., Ouanan, M., & Aksasse, B. (2018). Pubface: Celebrity face identification based on deep learning. *IOP Conference Series: Materials Science and Engineering*, *353*(1). https://doi.org/10.1088/1757-899X/353/1/012022

Pranav, K. B., & Manikandan, J. (2020). Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks. *Procedia Computer Science*, *171*, 1651–1659. https://doi.org/10.1016/j.procs.2020.04.177

Samet, R., & Tanriverdi, M. (2017). Face recognition-based mobile automatic classroom attendance management system. *Proceedings - 2017 International Conference on Cyberworlds, CW 2017 - in Cooperation with: Eurographics Association International Federation for Information Processing ACM SIGGRAPH*, *2017-January*, 253–256. https://doi.org/10.1109/CW.2017.34

Sayem, I. M., & Chowdhury, M. S. (2019). Integrating face recognition security system with the internet of things. *Proceedings - International Conference on Machine Learning and Data Engineering, ICMLDE 2018*, 19–21. https://doi.org/10.1109/iCMLDE.2018.00013

Siva Swetha, C., & Saranya, M. S. (2020). INTRUDER DETECTION SECURITY SYSTEM. *International Research Journal of Engineering and Technology*. www.irjet.net

Upadhyay, A., Chaudhari, K., Bhere, P., & Thomas, J. (n.d.). Body Posture Detection Using Computer Vision. In *SSRG International Journal of VLSI & Signal Processing (SSRG-IJVSP)* (Vol. 7). www.internationaljournalssrg.org

Vasuki, P., & Priya, S. A. (2018). A Smart Watchdog-Intruder Detection System. In *International Journal of Engineering & Technology* (Vol. 7). www.sciencepubco.com/index.php/IJET

Xenya, M. C., Kwayie, C., & Quist-Aphesti, K. (2019). Intruder Detection with Alert Using Cloud Based Convolutional Neural Network and Raspberry Pi. *Proceedings - 2019 International Conference on*

Computing, Computational Modelling and Applications, ICCMA 2019, 46–50. https://doi.org/10.1109/ICCMA.2019.00015

Ye, X., Leake, D., Huibregtse, W., & Dalkilic, M. (n.d.). *Applying Class-to-Class Siamese Networks to Explain Classifications with Supportive and Contrastive Cases*.

Zuo, F., & De With, P. H. N. (2005). Real-time face recognition for smart home applications. *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, 35–36. https://doi.org/10.1109/icce.2005.1429704

APPENDIX

```
!pip install tensorflow==2.4.1 tensorflow-gpu==2.4.1 opencv-python matplotlib


#Import standard dependencies

import cv2

import os

import random

import numpy as np

from matplotlib import pyplot as plt


#Import tensorflow dependencies

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Layer, Conv2D, Dense, MaxPooling2D, Input, Flatten

import tensorflow as tf


#Avoiding out of memory errors by setting GPU memory consumption growth

gpus = tf.config.experimental.list_physical_devices('GPU')

for gpu in gpus:

   tf.config.experimental.set_memory_growth(gpu, True)

#setup paths

POS_PATH = os.path.join('data', 'positive')

NEG_PATH = os.path.join('data', 'negative')

ANC_PATH = os.path.join('data', 'anchor')


os.makedirs(POS_PATH)

os.makedirs(NEG_PATH)

os.makedirs(ANC_PATH)


## Untar labelled faces in the wild dataset
```

```
# uncompress Tar GZ laballed faces in the  wild dataset(lfw)
!tar -xf lfw.tgz

#move lfw images to the following repository: data//negative
for directory in os.listdir('lfw'):
    for file in os.listdir(os.path.join('lfw',directory)):
        EX_PATH = os.path.join('lfw', directory, file)
        NEW_PATH = os.path.join(NEG_PATH, file)
        os.replace(EX_PATH, NEW_PATH)

# Import uuid library to generate unique image names
import uuid

os.path.join(ANC_PATH, '{}.jpg'.format(uuid.uuid1()))

# Establish a connection to the ourbcam
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    # Resize/cut down  the frame to 250 x 250 px
    frame = frame[150:150+250, 200:200+250, :]

    # Collect anchors
    if cv2.waitKey(1) & 0XFF == ord('a'):
        #create unique file path
        imgname = os.path.join(ANC_PATH, '{}.jpg'.format(uuid.uuid1()))
        #write out anchor image
        cv2.imwrite(imgname, frame)

    # Collect positives
    if cv2.waitKey(1) & 0XFF == ord('p'):
        #create unique file path
        imgname = os.path.join(POS_PATH, '{}.jpg'.format(uuid.uuid1()))
        #write out positive image
        cv2.imwrite(imgname, frame)

    # Collect negatives
    if cv2.waitKey(1) & 0XFF == ord('n'):
        #create unique file path
        imgname = os.path.join(NEG_PATH, '{}.jpg'.format(uuid.uuid1()))
        #write out  negative image
        cv2.imwrite(imgname, frame)

    # show image back to screen
    cv2.imshow('Image Collection', frame)

    #breaking gracefully
    if cv2.waitKey(1) & 0XFF == ord('q'):
        break

# Release the webcam
cap.release()
# Close the image show frame
cv2.destroyAllWindows()
```

```
#ce Get Image Directories
anchor = tf.data.Dataset.list_files(ANC_PATH+'\*.jpg').take(1200)
positive = tf.data.Dataset.list_files(POS_PATH+'\*.jpg').take(1200)
negative = tf.data.Dataset.list_files(NEG_PATH+'\*.jpg').take(3000)

dir_test = anchor.as_numpy_iterator()
dir_test.next()

## Preprocessing - Scale and Resize¶

def preprocess(file_path):
    # read in image from file path
    byte_img = tf.io.read_file(file_path)

    #load in the image
    img = tf.io.decode_jpeg(byte_img)

    # Preprocessing steps - resizing the image to be 100x100x3
    img = tf.image.resize(img, (100,100))

    # Scale image to be between 0 and 1
    img = img / 255.0

    # Return image
    return img

img =  preprocess('data\\anchor\\ab2b9138-471d-11ee-b598-3464a906fd38.jpg')
img.numpy().min()
```

```
plt.imshow(img)

# Create Labelled Dataset

positives = tf.data.Dataset.zip((anchor, positive, tf.data.Dataset.from_tensor_slices(tf.ones(len(anchor))))) #zip allows us to iterate through all three

negatives = tf.data.Dataset.zip((anchor, negative, tf.data.Dataset.from_tensor_slices(tf.zeros(len(anchor)))))

data = positives.concatenate(negatives)

samples = data.as_numpy_iterator()

example = samples.next()

example


anch = preprocess('data\\anchor\\f980af94-48bd-11ee-a51a-e4b318b8a3a7.jpg')

posit = preprocess('data\\positive\\a45ea168-4cc6-11ee-91d3-3464a906fd38.jpg')


## Build Train and Test Partition

def preprocess_twin(input_img, validation_img, label):

   return(preprocess(input_img), preprocess(validation_img), label)


res = preprocess_twin(*example)

plt.imshow(res[1])
```

NOTE:

The remainder of the code can be found on:

https://github.com/CASTDesigns/Intruder-Detection-System.git