A Project Report on

Weapon Identification Using Deep Learning

submitted in partial fulfillment for the award of

Bachelor of Technology

in

Computer Science and Engineering

by

Sk.Dinisha(Y20ACS567) **T.Olivyaisya(Y20ACS575)**

Sd.Fiza Ruhi Fathima (Y20ACS573) **Sk.S.Mansoor Ali (Y20ACS564)**

Under the guidance of
Dr.D.N.V.Syam Kumar
Associate Professor

Department of Computer Science and Engineering
Bapatla Engineering College
(Autonomous)
(Affiliated to Acharya Nagarjuna University)
BAPATLA – 522 102, Andhra Pradesh, INDIA
2023-2024

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled **Weapon Identification Using Deep Learning** that is being submitted by Sk.Dinisha (Y20ACS567), T.Olivyaisya(Y20ACS575), Sd.Fiza Ruhi Fathima(Y20ACS573) and Sk.S.Mansoor Ali(Y20ACS564) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide	**Signature of the HOD**
Dr.D.N.V.Shyam Kumar	**Dr.M.Rajesh Babu**
Associate Professor	**Associate Professor**

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Sk.Dinisha (Y20ACS567)
T.Olivyaisya (Y20ACS575)
Sd.Fiza Ruhi Fathima (Y20ACS573)
Sk.S.Mansoor Ali(Y20ACS564)

Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Dr.D.N.V.Syam Kumar, Assoc.Prof.,** Department of CSE, for his/her valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, **Assoc.Prof. & Head of the Dept**. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

Sk.Dinisha (Y20ACS567)
T.Olivyaisya (Y20ACS575)
Sd.Fiza Ruhi Fathima (Y20ACS573)
Sk.S.Mansoor Ali(Y20ACS564)

Abstract

The weapon detection system project employs convolutional neural networks (CNNs) and transfer learning techniques to categorize images into three classes: "No Weapon," "Pistol" and "Rifle," enhancing security measures in surveillance and law enforcement contexts. Leveraging TensorFlow, Keras, OpenCV, and scikit-learn, the project preprocesses the dataset through augmentation and partitioning for effective model training and evaluation. Using pre-trained models like MobileNetV2, fine-tuned with a custom convolutional model, the system achieves robust classification performance.

Deployment via the Flask web framework enables users to upload images for real-time weapon detection through a user-friendly web application interface. Performance evaluation metrics such as accuracy, precision, recall, and F1 score, along with visualization techniques like confusion matrices and ROC curves, provide insights into the models' classification performance. Ultimately, the project aims to bolster security and public safety by addressing contemporary security challenges and contributing to improved security measures.

Table of Contents

List of Figures

List of Tables

1 Introduction

In today's world, concerns regarding safety and security are paramount, with the proliferation of violence and the increasing frequency of armed incidents. In response to these challenges, advanced technological solutions are being sought to enhance surveillance and threat detection capabilities. One crucial area of focus is weapon identification, which involves the detection and classification of firearms in various environments, ranging from public spaces to sensitive facilities.

1.1 Introduction to Weapon Identification

In recent years, the rise in gun violence globally has underscored the urgent need for innovative technologies to assist law enforcement in detecting and identifying firearms. Traditional methods relying on manual inspection are time-consuming and prone to errors. Deep Learning, a branch of artificial intelligence, offers a solution by leveraging advanced algorithms for firearm identification through image analysis. This project report explores the application of Deep Learning, particularly MobileNet, to automate firearm detection with high accuracy and efficiency.

Identification of weapons is essential to maintaining security and safety for the general population. Globally, there is a growing need for automated systems that can identify weapons in a variety of contexts with speed and accuracy due to the growth in violent situations. These kinds of tools can help law enforcement agencies prevent crimes, improve public space security, and support threat assessment processes.

1.2 MobileNet

Deep Learning involves training artificial neural networks to recognize patterns in data. These networks, inspired by the human brain’s structure, excel in tasks such as image recognition, object detection, and classification. Convolutional Neural Networks (CNNs), a key component of Deep Learning, are widely used for image analysis due to their ability to extract features from images.

MobileNet is a specialized CNN architecture designed for mobile and embedded vision applications. Unlike traditional CNNs, MobileNet emphasizes computational efficiency and speed, making it suitable for resource-constrained environments such as mobile devices and real-time video processing systems. Its architecture employs depthwise separable convolutions, which reduce the computational cost without significantly sacrificing accuracy.

1.2.1 Types of MobileNet

MobileNet is a family of convolutional neural network architectures specifically designed for efficient processing on mobile and embedded devices. It was introduced by Google researchers in 2017 as a solution to the computational challenges posed by deploying deep learning models on devices with limited resources. MobileNet achieves efficiency through a combination of depthwise separable convolutions and lightweight network design principles.

Depthwise separable convolutions decompose the standard convolution operation into two separate operations: a depthwise convolution and a pointwise convolution. This reduces both the number of parameters and the amount of computation required, resulting in a more efficient network architecture. There are several versions of MobileNet, each offering different trade-offs between computational efficiency and accuracy.

MobileNetV1: The original MobileNet architecture, introduced in 2017. It laid the foundation for subsequent versions and established the efficiency principles of depthwise separable convolutions.

MobileNetV2: Introduced in 2018, MobileNetV2 builds upon the success of the original MobileNet by introducing inverted residual blocks and linear bottlenecks. It achieves higher accuracy and efficiency compared to MobileNetV1.

MobileNetV3: Released in 2019, MobileNetV3 further improves upon the efficiency and performance of MobileNetV2 by introducing a streamlined architecture with improved training techniques such as squeeze-and-excitation blocks and hard-swish activation functions.

1.2.2 Benefits of MobileNetV2

Each version of MobileNet offers improvements over its predecessors in terms of efficiency, accuracy, and performance, making them suitable for a wide range of mobile and embedded deep learning applications.While both MobileNetV2 and MobileNetV3 offer improvements over their predecessors, each version has its own set of benefits. Here are some potential benefits of MobileNetV2 over MobileNetV3.

Simplicity and Ease of Deployment:

MobileNetV2 has a simpler architecture compared to MobileNetV3. This simplicity can make it easier to understand, implement, and deploy in various applications, especially for developers who are new to deep learning or working with resource constrained devices.

Lower Computational Complexity:

MobileNetV2 may have lower computational complexity compared to MobileNetV3 due to its simpler architecture. This can result in faster inference times and lower resource requirements, making it more suitable for real-time applications or devices with limited computational resources.

Wider Adoption and Community Support:

MobileNetV2 has been available for a longer time and has been widely adopted in various applications and frameworks. As a result, there may be more pre-trained models, tutorials, and community support available for MobileNetV2 compared to MobileNetV3.

Proven Performance:

MobileNetV2 has been extensively tested and benchmarked in various scenarios, demonstrating its effectiveness and performance across a wide range of tasks, including image classification, object detection, and semantic segmentation.However, it’s essential to note that MobileNetV3 also brings several improvements over MobileNetV2, such as enhanced architecture design, improved accuracy, and better optimization techniques. The choice between MobileNetV2 and MobileNetV3 ultimately depends on the specific requirements and constraints of your application, as well as any performance trade-offs you’re willing to make.

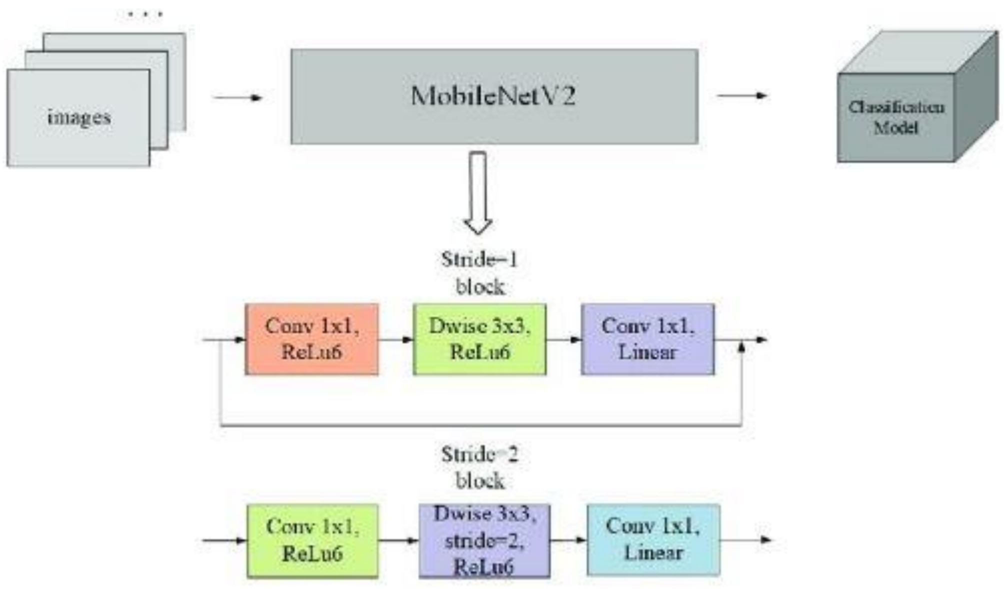


Figure 1.1 MobileNet Structure

1.3 Introduction to Domain Knowledge

For a weapon detection project, it's important to have a basic understanding of object detection, deep learning, dataset preparation, model selection, training and evaluation. This knowledge will help you develop and deploy weapon detection.

1.3.1 Machine Learning:

Machine learning (ML) is a subset of artificial intelligence (AI) that focuses on developing algorithms and statistical models that enable computers to learn and improve from experience without being explicitly programmed. In essence, machine learning allows computers to learn from data and make decisions or predictions based on that learning. The goal of machine learning is to develop models that can generalize well to new, unseen data, enabling them to perform tasks such as classification, regression, clustering, and anomaly detection.

1.3.2 Deep Learning

In technical terms, deep learning is a subfield of machine learning that uses artificial neural networks with multiple layers to model and solve complex problems. These networks are composed of interconnected nodes (neurons) organized into layers. Each layer of neurons processes the input data and passes it to the next layer. The network learns to extract relevant features from the input data as it passes through these layers. This hierarchical representation allows deep learning models to learn intricate patterns and relationships in the data.

Deep learning models are trained using an optimization algorithm called back propagation, which adjusts the weights of the connections between neurons to minimize the difference between the predicted output and the actual output. This process is repeated iteratively until the model achieves a satisfactory level of accuracy. Deep learning has shown remarkable success in various applications, including computer vision, natural language processing, speech recognition, and more. Its ability to automatically learn representations from data has led to significant advancements in AI technology.

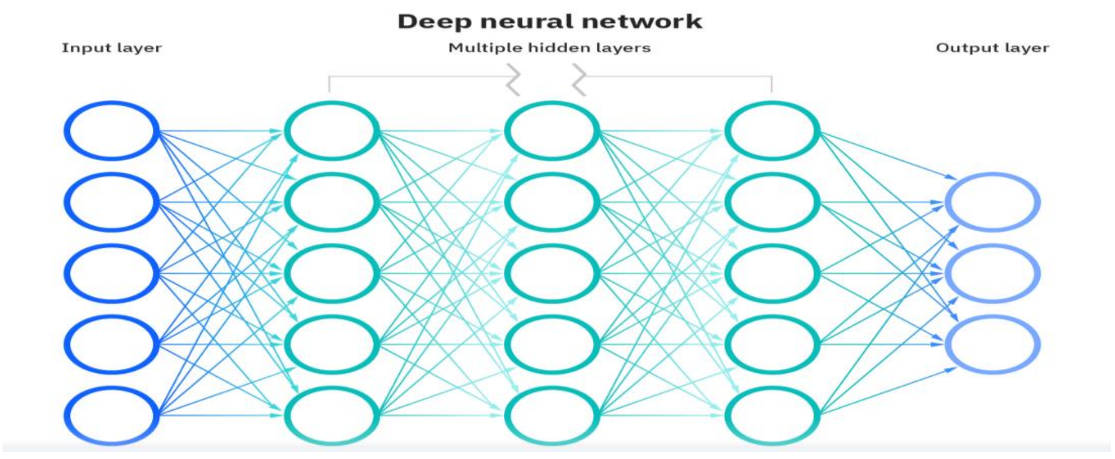


Figure 1.2 Deep neural network Layers

In a neural network, information flows through interconnected nodes called neurons. These neurons are organized into layers: an input layer, one or more hidden layers, and an output layer.

Input Layer: The input layer is where the network receives input data. Each neuron in the input layer represents a feature or attribute of the input data. For example, in an image recognition task, each neuron might represent a pixel's intensity value.

Hidden Layer: Hidden layers are layers between the input and output layers where the network performs computations. Each neuron in a hidden layer takes input from the neurons in the previous layer, applies weights and biases, and passes the result through an activation function to introduce non-linearity. The number of hidden layers and neurons in each layer is a key factor in determining the network's capacity to learn complex patterns in the data.

Output Layer: The output layer is where the network produces its output. The number of neurons in the output layer depends on the nature of the task. For example, in a binary classification task, there might be one neuron in the output layer that represents the probability of the input belonging to one class, while in a multi-class classification task, there would be multiple neurons, each representing the probability of the input belonging to a different class.

Convolutional Neural Network:

A Convolutional Neural Network (CNN) is a type of deep neural network that is well-suited for analyzing visual data such as images. CNNs are composed of multiple layers, including convolutional layers, pooling layers, and fully connected layers

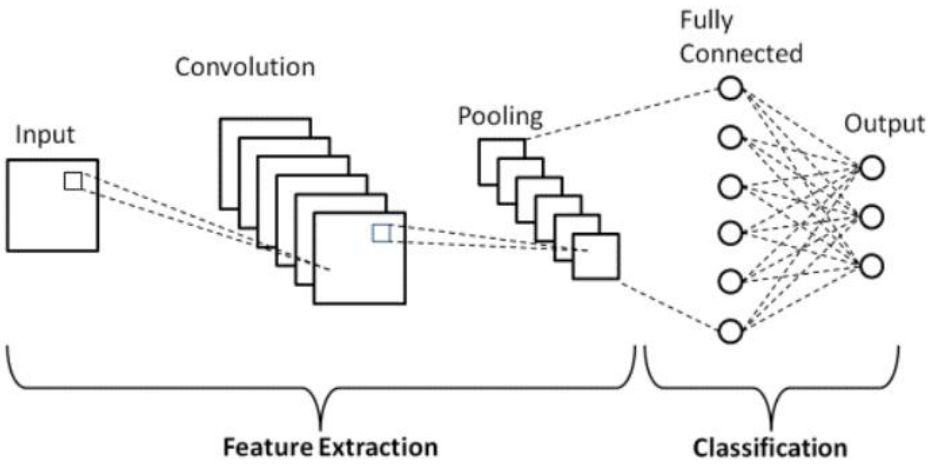


Figure 1.3 CNN Architecture

Convulational Layer: These layers apply convolution operations to the input, using learnable filters (kernels) to extract features from the input data. Each filter is applied across the entire input to produce a feature map, which highlights specific features in the input data. Convolutional layers are responsible for capturing spatial hierarchies of features in the input.

Pooling Layer: Pooling layers down sample the feature maps produced by the convolutional layers, reducing the spatial dimensions of the data while retaining important features. This helps in reducing the computational complexity of the network and controlling overfitting.

a. **Max Pooling**: Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

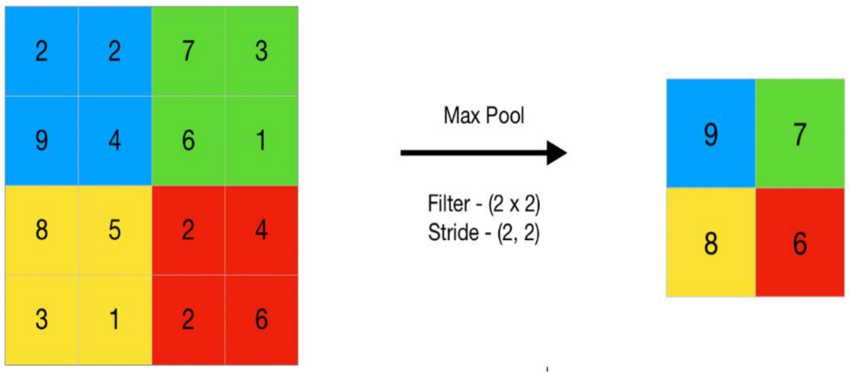


Figure 1.4 Max Pooling

b. **Average Pooling**: Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

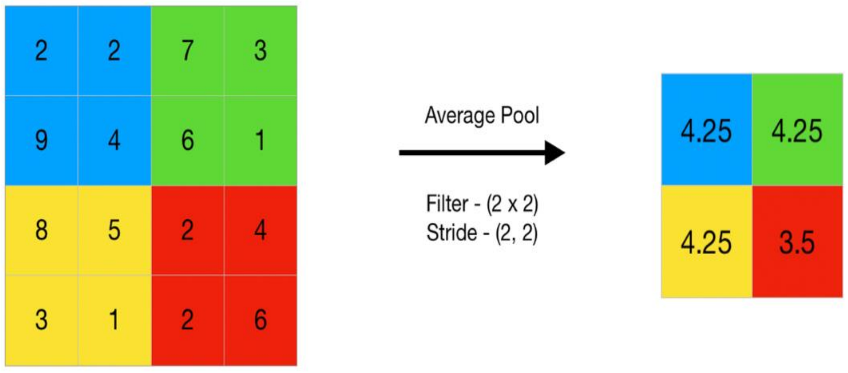


Figure 1.5 Average Pooling

Fully Connected Layer: These layers connect every neuron in one layer to every neuron in the next layer, similar to a traditional neural network. They are typically used towards the end of the network to classify the features extracted by the earlier layers.

A fully connected layer, also known as a dense layer, is a type of artificial neural network layer where each neuron or node is connected to every neuron in the previous layer. In a fully connected layer, the output from each neuron in the previous layer is used as input to each neuron in the current layer, and each connection has its own weight. Fully connected layers are typically used in feed forward neural networks, where the data flows in one direction, from the input layer through one or more hidden layers to the output layer.

These layers are often used for learning patterns in the data and are a key component of deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).In a fully connected layer, each neuron performs a weighted sum of its inputs, adds a bias term, and then applies an activation function to produce the output. This output is then used as input to the neurons in the next layer, and the process is repeated until the final output is produced.

Fully connected layers are powerful because they allow the model to learn complex patterns in the data by adjusting the weights of the connections between neurons during training. However, they also require a large number of parameters, which can make them computationally expensive and prone to overfitting, especially in deep networks

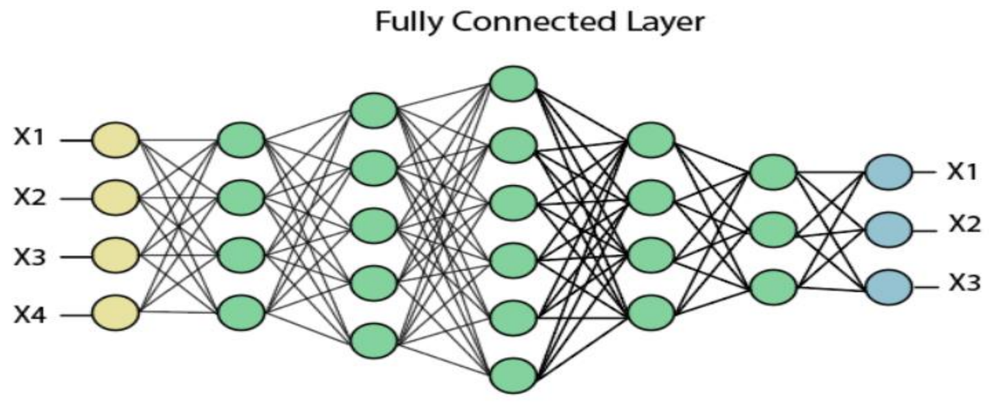


Figure 1.6 Fully Connected Layer

NumPy: NumPy is a fundamental package for scientific computing with Python. It provides support for multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. Here are some key features of NumPy.

Multi-Dimensioal arrays: NumPy provides the ndarray data structure, which represents arrays of a fixed size and allows for efficient storage and manipulation of large datasets.

Mathematical Functions:

NumPy provides a wide range of mathematical functions that can be applied element-wise to arrays, such as trigonometric functions, exponential and logarithmic functions, and more. NumPy's broadcasting feature allows for arithmetic operations between arrays of different shapes and sizes, making it easier to write vectorized code.

Numpy also provides powerful indexing and slicing capabilities for accessing and manipulating elements of arrays. It includes functions for performing various linear algebra operations, such as matrix multiplication, matrix inversion, eigenvalue decomposition, and more..It provides functions for generating random numbers from different probability distributions.

Simply, NumPy is a foundational library for many other libraries in the Python scientific computing ecosystem, such as Pandas, SciPy, and scikit-learn. Its efficient array operations and mathematical functions make it essential for numerical computing tasks in Python.

Pandas:

Pandas is a versatile Python library for data manipulation and analysis, built on top of NumPy. It offers two main data structures: Series, a one-dimensional array with labeled axes, and DataFrame, a two-dimensional labeled data structure resembling a table. Pandas excels in data cleaning tasks, offering functions to handle missing data, remove duplicates, and transform data easily. It also provides powerful tools for data transformation, including filtering, sorting, grouping, and aggregating data.

Pandas is widely used in data analysis due to its ability to perform descriptive statistics, merge and join datasets, and handle time series data efficiently. It's often used alongside other libraries like Matplotlib for plotting and Scikit-learn for machine learning, making it an essential tool for data scientists and analysts working with structured data in Python.

Matplotlib:

Matplotlib is a versatile and powerful library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting tools, including line plots, scatter plots, bar plots, histograms, and more, allowing users to visualize their data in various formats. One of the key strengths of Matplotlib is its flexibility and customization options, which enable users to fine-tune every aspect of their plots, such as colors, fonts, labels, and axes.

Matplotlib is also highly compatible with other libraries in the Python ecosystem, such as NumPy and Pandas, making it a popular choice for data visualization tasks. Whether you're a beginner looking to create simple plots or an advanced user needing to create complex visualizations, Matplotlib offers the tools and flexibility to meet your needs.

Keras:

Keras provides a wide range of pre-built layers, activation functions, optimizers, and loss functions, making it easy to customize models for specific tasks. Additionally, Keras supports both convolutional neural networks (CNNs) and recurrent neural networks (RNNs), as well as combinations of the two, allowing users to tackle a wide range of problems, including image recognition, natural language processing, and more.

Another key feature of Keras is its support for multiple backend engines, including TensorFlow, Theano, and Microsoft Cognitive Toolkit (CNTK). This allows users to choose the backend that best suits their needs, while still being able to use the same Keras API. TensorFlow has become the default backend for Keras since it was integrated into TensorFlow as its official high-level API.

In summary, Keras is a versatile and powerful deep learning library that has become a popular choice among researchers and practitioners due to its ease of use, flexibility, and scalability. Its high-level API and support for multiple backends make it an ideal tool for building and deploying deep learning models for a wide range of applications.

Preprocessing:

In Keras, the preprocessing module provides functions and classes to help prepare data for use in deep learning models. These tools are designed to make it easier to work with different types of data, such as images, text, and sequences. One key tool is the Image Data Generator, which can generate batches of image data for training deep learning models. It can also apply data augmentation techniques, like rotating or flipping images, to create more varied training data and improve model performance.

Another useful tool is the Tokenizer, which can be used to break down text into individual words or tokens. This is helpful when working with natural language processing tasks, where text data needs to be converted into a format that a neural network can understand. The Pad Sequences function is also handy when dealing with sequences of data, such as sentences or time series. It can add padding to sequences so that they all have the same length, which is often required when training neural networks.

Overall, the Preprocessing module in Keras provides a set of tools to help prepare different types of data for deep learning models, making it easier to work with complex datasets and improve model performance.

TensorFlow:

One of the core features of TensorFlow is its computational graph paradigm, where computations are represented as a directed graph. Nodes in the graph represent mathematical operations, while edges represent the flow of data between operations.

This allows TensorFlow to efficiently distribute computations across multiple CPUs or GPUs, making it suitable for training large-scale deep neural networks.

TensorFlow supports a wide range of machine learning tasks, including:

a. **Machine Learning**: TensorFlow offers tools and libraries for traditional machine learning tasks, such as linear regression, logistic regression, decision trees, and clustering.
b. **Custom Models**: TensorFlow allows users to define custom models and algorithms using its flexible programming interface. This enables researchers and developers to experiment with new ideas and implement cutting-edge techniques in machine learning and deep learning.
c. **Deployment**: TensorFlow provides tools for deploying trained models in production environments, including TensorFlow Serving for serving models over a network, TensorFlow Lite for running models on mobile and embedded devices, and TensorFlow.js for running models in web browsers.
d. **Scalability:** TensorFlow is designed to scale from single devices to large clusters of servers, enabling distributed training and inference for training models on large datasets or deploying models with high throughput requirements.

TensorFlow has a large and active community of developers and researchers who contribute to its development and maintenance. It is widely used in both academia and industry for a variety of applications, including computer vision, natural language processing, speech recognition, recommendation systems, and more.

Confusion Matrix:

A Confusion Matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of instances produced by the model on the test data.

a. True Positives (TP): occur when the model accurately predicts a positive data point.
b. True Negatives (TN): occur when the model accurately predicts a negative data point.
c. False Positives (FP): occur when the model predicts a positive data point incorrectly.
d. False Negatives (FN): occur when the model mispredicts a negative data point.

When assessing a classification model's performance, a confusion matrix is essential. It offers a thorough analysis of true positive, true negative, false positive and false negative predictions, facilitating a more profound comprehension of a model's recall, accuracy, precision and overall effectiveness in class distinction. When there is an uneven class distribution in a dataset, this matrix is especially helpful in evaluating a model's performance beyond basic accuracy metrics.

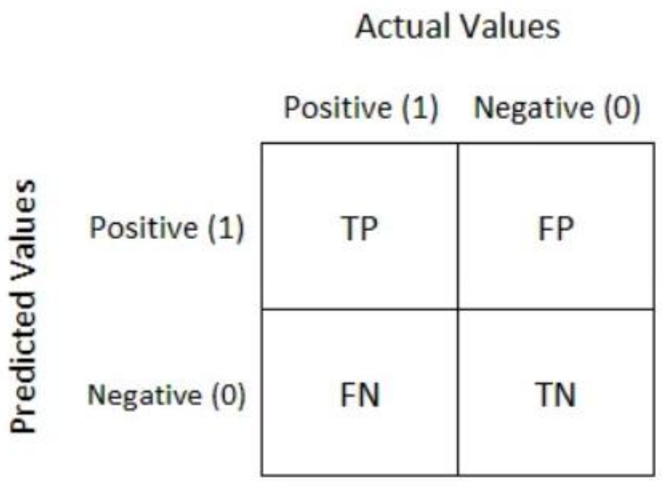


Figure 1.7 Confusion Matrix

Performance Metrics:

a. Accuracy: Accuracy is the ratio of correctly predicted instances to the total instances in the dataset. It is a measure of overall model correctness.

Accuracy=TP+TN/TP+TN+FP+FN

b. Precision: Precision is the ratio of correctly predicted positive instances to the total predicted positive instances.

Precision=TP/TP+FP

c. Recall: Recall is the ratio of correctly predicted positive instances to all actual positive instances.

Recall=TP/TP+FN

d. F1-Score: The F1 score is a single metric that combines both precision and recall into a single number. It provides a way to balance these two metrics, giving you a better overall understanding of how well your model is performing.

$$F1\text{-}Score=2*Precision*Recall/\ (Precision + Recall)$$

Benefits of Performance Metrics:

Certainly, each performance metric accuracy, precision, recall, and F1 score offers unique insights into the performance of a machine learning model. Accuracy gives a general sense of how often the model is correct, which is useful for balanced datasets. Precision focuses on the proportion of correctly predicted positive instances among all positive predictions, helping to minimize false alarms.

Recall, on the other hand, emphasizes the ability to capture all actual positive instances, making it valuable for scenarios where missing positives is costly. The F1 score strikes a balance between precision and recall, providing a single metric to consider both false positives and false negatives equally.

Artificial intelligence encompasses machine learning and deep learning, which allow computers to learn from data and make judgments or predictions without the need for explicit programming. In the framework of this research, the analysis and classification of photographs containing firearms is done using machine learning algorithms, specifically deep neural networks such as MobileNetV2.

1.4 Project Overview

The goal of this research is to apply deep learning methods to construct a weapon identification system. The main goal is to support law enforcement and security services by precisely identifying and categorizing different kinds of weaponry in photographs.

1.5 Objectives

The following are the project’s primary goals:

a. To create a solid machine learning model that can accurately recognize firearms.
b. In order to put the MobileNetV2 algorithm into practice for accurate and efficient weapon identification.
c. To assess the model’s effectiveness using recall, accuracy, and precision metrics.

1.6 Scope

The following topics will be covered by this project:

a. Identification of rifles, and pistols.
b. Image processing in batch and real-time for weapon detection.

1.7 Problem Statement

The problem statement for weapon identification typically revolves around developing a computer vision system capable of detecting and identifying various types of weapons in images or video footage. The objective is to enhance public safety and security by enabling rapid and accurate detection of weapons in real-time scenarios, such as security checkpoints, surveillance systems, or law enforcement operations. The system should be able to distinguish between different types of weapons, such as firearms, knives, or explosives, and potentially classify them based on their threat level or category.

Automated systems that can swiftly and correctly identify firearms in a variety of contexts are becoming more and more necessary as worries about public safety grow. Additionally, the system should be robust to variations in lighting, orientation, and occlusions to ensure reliable performance in diverse environments. The ultimate goal is to provide law enforcement agencies, security personnel, and other stakeholders with a reliable tool to detect and respond to potential threats effectively.

2 Literature Survey

Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review", IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019

In this paper, Convolutional Neural Networks (CNNs) are mentioned as a significant advancement in the field of object detection. Here's how CNNs are used and referenced in the context for the model:

Deep Neural Networks (DNNs): This highlights the emergence of Deep Neural Networks (DNNs), with CNNs being the most representative architecture. It mentions that DNNs, particularly CNNs, have deeper architectures compared to traditional approaches, allowing them to learn more complex features without the need for manual feature design.

Feature Extraction: CNNs are used for feature extraction, providing a semantic and robust representation of objects in images. The passage mentions that CNN features are incorporated into the detection pipeline, allowing for more informative object representations compared to manually engineered features.

State-of-the-Art Results: this explained as that state-of-the-art results in object detection have been achieved with the introduction of CNN features in models like R-CNN. This indicates the effectiveness of CNNs in improving detection accuracy and efficiency.

Overall, CNNs play a central role in advancing object detection techniques, offering deeper architectures, automatic feature learning, and improved detection performance compared to traditional methods. Their usage has led to significant progress in the field, enabling the development of more accurate and efficient object detection systems.

J. Redmon, S. K. Divvala, R. B. Girshick and A. Farhadi, "You only look once: Unified real-time object detection", 2016 IEEE Conference on Computer Vision and Pattern Recognition CVPR 2016, pp. 779-788, 2016

The paper presents YOLO (You Only Look Once), a novel approach to object detection that reframes the task as a regression problem, predicting bounding box coordinates and class probabilities directly from image pixels. Unlike traditional methods, YOLO uses a single convolutional neural network to predict multiple bounding boxes and class probabilities simultaneously for all objects in an image. This unified architecture enables real-time processing speeds, with the base model running at 45 frames per second and a faster version achieving over 150 frames per second.

YOLO reasons globally about the entire image during training and testing, encoding contextual information and achieving high generalizability across different domains. Despite some limitations in precise localization, YOLO's simplicity, speed, and accuracy make it suitable for various applications such as surveillance, security, and robotics. The paper provides detailed insights into the unified detection process, network design, and the availability of open-source code and pretrained models for further exploration and application.

Chauhan, R., Ghanshala, K. K., & Joshi, R. (2018). Convolutional Neural Network (CNN) for Image Detection and Recognition. 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)

The paper discusses the implementation of Convolutional Neural Network (CNN) models for image recognition and object detection tasks. CNNs have shown remarkable success in computer vision due to their ability to extract features and provide multiple levels of abstraction. The deeper CNN architecture is utilized to capture complex features. Various techniques such as data augmentation, dropout regularization, and RMSprop optimizer are employed to enhance model performance and mitigate overfitting.

The CNN models are implemented using Python with TensorFlow library. The models are trained and evaluated using metrics like accuracy, precision, recall, and F1-score. The experimental results demonstrate high accuracy on both datasets, indicating the effectiveness of CNNs in image recognition and object detection tasks.

The paper concludes that CNNs are powerful tools for computer vision tasks and suggests further exploration of advanced architectures, data augmentation techniques, and optimizer algorithms to enhance model performance. Future work also includes the application of transfer learning and ensemble learning approaches for real-world deployment of CNN models. Overall, continuous research and development efforts are emphasized to advance the field of computer vision and unlock new possibilities in artificial intelligence.

B. Venkatesh, E.A. (2023). Study on Deep Learning Techniques for Finding Suspicious Violence Detection in a Video Surveillance. International Journal on Recent and Innovation Trends in Computing and Communication.

This paper explores the use of deep learning techniques for automatic violence detection in video surveillance, showcasing the efficacy of various deep models in classifying suspicious videos for public safety. Deep learning is an intelligent and trustworthy technique for detecting or classifying suspicious data objects.

The methodology involved the use of deep learning techniques such as CNN, ConvLSTM, AlexNet, VGG-16, MobileNet, and GoogLeNet for violence detection in video surveillance. Experimental studies were conducted using benchmarked video datasets to evaluate the models' efficacy, with classification measures used to demonstrate their performance.

Jain, Harsh, Aditya Vikram, Mohana, Ankit Kashyap and Ayush Jain. "Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications." 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC) (2020): 193-198.

The paper focuses on implementing automatic gun or weapon detection using CNN-based algorithms with two types of datasets, achieving good accuracy with a trade-off between speed and accuracy in real-world applications. The methodology involved implementing automatic gun or weapon detection using convolutional neural network (CNN) based SSD and Faster RCNN algorithms with two types of datasets: pre-labelled images and manually labelled images. 8 Algorithms achieve good accuracy, but their application in real situations can be based on the trade-off between speed and accuracy.

Hashmi, Tufail Sajjad Shah, Nazeef Ul Haq, Muhammad Moazam Fraz and Muhammad Ali Shahzad. "Application of Deep Learning for Weapons Detection in Surveillance Videos." 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2) (2021): 1-6.

The paper discusses the challenges of weapon detection, compares YOLOV3 and YOLOV4 models, and evaluates their performance on a large dataset, emphasizing precision metrics. YOLOV4 performs obviously superior to the YOLOV3 in terms of processing time and sensitivity. The methodology involved utilizing CNN-based deep learning approaches, conducting a comparative analysis between YOLOV3 and YOLOV4 models, creating a weapons dataset, manually annotating images, and training the models on a large dataset for testing.

i. Difficulty in detecting weapons of distinctive size and shapes with different background colors.
ii. Comparison limited to YOLOV3 and YOLOV4, no comparison with other models.
iii. Precision metric comparison not detailed.

Bhatti, Muhammad Tahir, Muhammad Gufran Khan, Masood Aslam and Muhammad Junaid Fiaz. "Weapon Detection in Real-Time CCTV Videos Using Deep Learning." IEEE Access 9 (2021): 34366-34382.

The paper focuses on weapon detection in real-time CCTV videos using deep learning algorithms, with YOLOv4 achieving the best performance. The best algorithm for object detection in real-time is Yolov4. The methodology involved creating a dataset from various sources, utilizing two main approaches for weapon detection, and testing multiple deep learning algorithms, with YOLOv4 showing the best performance.

i. Lack of a standard dataset for real-time scenarios.
ii. Reliance on a self-made dataset which may not fully represent real-world scenarios.
iii. The need for a system that can automatically detect illegal activities.

Araveti, S.H., Gosika, B., & Narregudem, M. (2022). Armaments Detection Using Artificial Intelligence and Deep Learning for Security Application. International Journal for Research in Applied Science and Engineering Technology.

The paper focuses on implementing automatic gun or weapon detection using artificial intelligence and deep learning for security applications, emphasizing the importance of security and the use of computer vision for abnormal detection. The system is entailed with automatic detection of the handgun/ knife or the other crime objects without manual intervention.

The methodology involved using two types of datasets (pre-labeled and manually labeled images) and implementing Convolution Neural Network (CNN) based algorithms for automatic gun or weapon detection without manual intervention, with a focus on the trade-off between speed and accuracy. The limitations of the study include the trade-off between speed and accuracy in 9 real-life applications and the necessity for automatic detection without manual intervention.

Dr. S. Nikkath Bushra ,Ms. G. Shobana,K. Uma Maheswari and Nalini Subramanian, "Smart Video Survillance Based Weapon Identification Using Yolov5" , IEEE Paper publish on 18 August 2022.

Based on the document snippets provided, it appears that the focus is on smart video surveillance-based weapon identification using the YOLOv5 algorithm. This algorithm is aimed at detecting various activities such as holding weapons, face detection, and abnormal actions in public places with higher accuracy and efficiency. The YOLOv5 algorithm has shown significant improvements in size, speed, and accuracy compared to YOLOv4, making it a preferred choice for object detection tasks.

Moreover, previous studies have explored different approaches to weapon detection, such as using MobileNet architecture and comparing various models like Faster RCNN, Masked RCNN, YOLO, and SSD_mobilenet_v2_coco. These studies have highlighted the importance of accuracy, speed, and model selection for effective weapon identification in surveillance settings.

Overall, the research in this domain emphasizes the use of advanced algorithms, deep learning techniques, and neural networks to enhance surveillance capabilities and address security concerns in public spaces. By leveraging these technologies, the goal is to improve the detection of weapons and suspicious activities, ultimately enhancing public safety and security measures.

Muralidhar Pullakandam,Keshav Loya,Pranav Salota , Rama Muni Reddy Yanamala and Pavan Kumar Javvaji ,"Weapon Object Detection Using Quantized YOLOv8" ,IEEE Paper Publish on 09 August 2023.

Video surveillance is essential for creating a secure and hassle-free environment in all areas of life. It helps identify theft, detect unusual events in crowded locations, and monitor the suspicious behavior of individuals. However, monitoring surveillance cameras manually is quite challenging, and thus, fully automated surveillance with smart video-capturing capabilities is gaining popularity. This approach uses deep learning methodology to remotely monitor unusual actions with accurate information about the location, time of occurrence, and identification of criminals.

Detecting criminal conduct in public settings is difficult due to the complexity of real-world scenarios. CCTV cameras can record suspicious incidents in public areas, such as carrying weapons, which helps authorities to take preventive measures to protect citizens. The proposed system employs the state-of-the-art YOLOv8 model for real-time weapon detection, which is faster, more accurate, and better than YOLOv5. To ensure fast performance, the weights of YOLOv8 were quantized. In our experiments, we evaluated the performance of the YOLOv8 and YOLOv5 models for weapon detection.

The mean Average Precision (mAP) value achieved using YOLOv8 was 90.1%, which outperformed the mAP value of 89.1% obtained with YOLOv5. Furthermore, by applying weight quantization to the YOLOv8 model, we reduced the inference time by 15% compared to the original YOLOv8 configuration.

3 Proposed System

Here we are using MobileNetV2 model .When Compared to the Yolov4 we can get the good performance by using MobileNetV2 those are:

a. **Efficiency:** MobileNetV2 is designed to be lightweight and efficient, making it suitable for deployment on devices with limited computational resources. This efficiency is crucial for real-time or embedded systems where processing power is constrained. In weapon identification scenarios, where speed and responsiveness are important (e.g., in surveillance systems), the efficiency of MobileNetV2 can be a significant advantage.

b. **Feature Extraction**: MobileNetV2 has been trained on a large dataset (ImageNet) and has learned to extract hierarchical features from images effectively. These features capture important visual patterns and characteristics of objects present in the images. In the context of weapon identification, MobileNetV2 can automatically learn to extract relevant features such as the shape, texture, and configuration of various types of weapons (e.g., handguns, rifles) from images.

c. **Transfer Learning**: One of the key advantages of MobileNetV2, like many other pre-trained models, is its ability to perform transfer learning. This means that the model can be fine-tuned on a smaller dataset specific to the weapon identification task, leveraging the knowledge it gained from ImageNet pretraining. With transfer learning, MobileNetV2 can adapt its learned features to better discriminate between different types of weapons based on the characteristics present in the training dataset.

d. **Versatility**: MobileNetV2 is a versatile architecture that can be adapted to various image classification tasks, including weapon identification. Its flexibility allows researchers and developers to experiment with different configurations, fine-tuning strategies, and training datasets to optimize performance for specific use cases and deployment scenarios.

Finally MobileNetV2 offers a compelling combination of efficiency, effectiveness, and versatility, making it a valuable tool for weapon identification tasks, particularly in scenarios where computational resources are limited, and real-time processing is required.

Loading MobileNetV2: Here Keras is using as the backend, from that we are importing the MobileNetv2.

a. **Adding Custom Head**: A custom classification head on top of the MobileNetV2 backbone. This head consists of layers such as average pooling, dense, and dropout layers, followed by a softmax layer with the number of output classes corresponding to the number of weapon categories you want to identify.

b. **Freezing Layers**: Optionally, freeze the weights of the MobileNetV2 backbone to prevent them from being updated during training. This is done to retain the knowledge learned from ImageNet and prevent overfitting, especially when training with limited data. Compiling the Model: Compile the model using an appropriate loss function (e.g., categorical crossentropy), optimizer (e.g., Adam), and evaluation metric (e.g., accuracy).

Training: Train the model using the training dataset. During training, the model learns to extract features from the input images using the MobileNetV2 backbone and predicts the corresponding weapon categories using the custom classification head. **Evaluation:** Evaluate the trained model using the testing dataset to assess its performance on unseen data. Common evaluation metrics include accuracy, precision, recall, and F1-score. The inner workings of MobileNetV2 involve several key architectural components and operations designed to achieve efficiency while maintaining good performance in various computer vision tasks. Here's an overview of the inner processes in MobileNetV2:

Depthwise Separable Convolutions:

i. MobileNetV2 primarily relies on depthwise separable convolutions, which decompose the standard convolution into two separate operations: depthwise convolution and pointwise convolution.

ii. Depthwise convolution applies a single convolutional filter per input channel, resulting in a set of intermediate feature maps.

iii. Pointwise convolution performs 1x1 convolutions on the intermediate feature maps to combine information across channels.

iv. By separating spatial and channel-wise information, depthwise separable convolutions reduce the computational cost while preserving representational capacity.

Inverted Residuals:

a. MobileNetV2 introduces the concept of inverted residuals, which are structured similarly to traditional residual blocks but with a different order of operations.
b. In an inverted residual block, the input is first expanded using a pointwise convolution (increasing the number of channels), followed by depthwise convolution and another pointwise convolution to reduce the dimensionality back to the original size.
c. The use of inverted residuals allows MobileNetV2 to efficiently capture both low-level and high-level features within the network.

Width Multiplier and Resolution Multiplier:

i. MobileNetV2 introduces parameters known as width multiplier and resolution multiplier, which control the size and computational cost of the model.
ii. The width multiplier scales the number of channels in each layer, allowing for trade-offs between model size and accuracy.
iii. The resolution multiplier scales the input resolution of the model, further adjusting the computational cost and memory requirements.

Efficient Feature Extraction:

i. The combination of depthwise separable convolutions, inverted residuals, linear bottlenecks, and adjustable model size enables MobileNetV2 to efficiently extract features from input images.
ii. These features capture important visual patterns and characteristics present in the images, making MobileNetV2 suitable for various computer vision tasks such as image classification, object detection, and semantic segmentation.

Finally the inner process of MobileNetV2 focuses on leveraging efficient architectural components and optimization techniques to achieve a balance between computational efficiency and performance in deep learning tasks. By carefully designing the network structure and operations, MobileNetV2 provides a lightweight and effective solution for a wide range of computer vision applications.

3.1 MobileNet Model

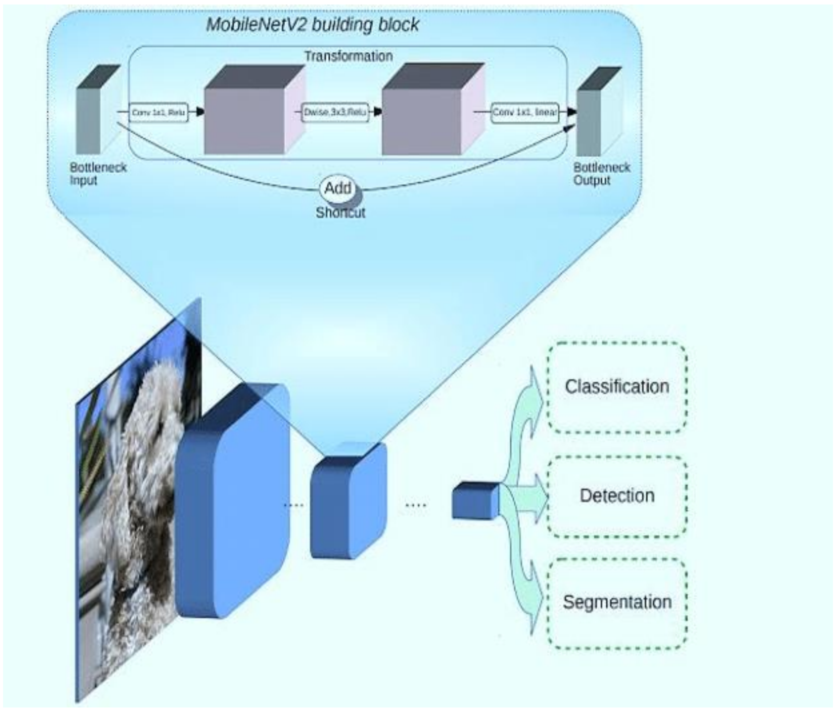


Figure 3.1 MobileNet Architecture

MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

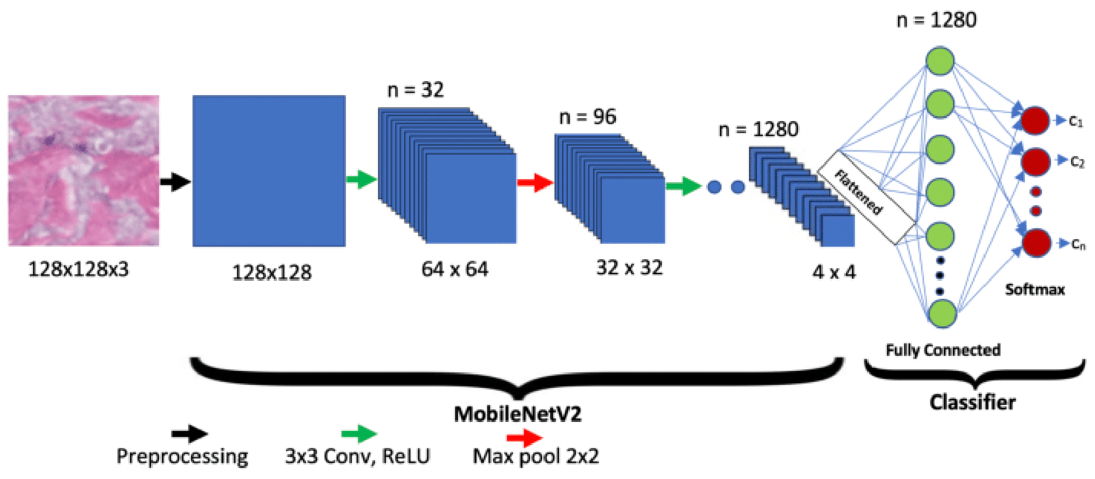


Figure 3.2MobileNet Model

3.2 MobileNetV2 Architecture

The architecture of MobileNetV2 consists of a series of convolutional layers, followed by depthwise separable convolutions, inverted residuals, bottleneck design, linear bottlenecks, and squeeze-and-excitation (SE) blocks. These components work together to reduce the number of parameters and computations required while maintaining the model's ability to capture complex features.

3.2.1 Inverted Residuals

Inverted residuals are a key component of MobileNetV2 that helps improve the model's accuracy. They introduce a bottleneck structure that expands the number of channels before applying depthwise separable convolutions. This expansion allows the model to capture more complex features and enhance its representation power.

3.2.2 Bottleneck Design

The bottleneck design in MobileNetV2 further reduces the computational cost by using 1×1 convolutions to reduce the number of channels before applying depthwise separable convolutions. This design choice helps maintain a good balance between model size and accuracy.

3.2.3 Linear Bottlenecks

Linear bottlenecks are introduced in MobileNetV2 to address the issue of information loss during the bottleneck process. By using linear activations instead of non-linear activations, the model preserves more information and improves its ability to capture fine-grained details.

3.2.4 Squeeze-and-Excitation (SE) Blocks

Squeeze-and-excitation (SE) blocks are added to MobileNetV2 to enhance its feature representation capabilities. These blocks adaptively recalibrate the channel-wise feature responses, allowing the model to focus on more informative features and suppress less relevant ones.

3.3 How to Train MobileNetV2?

Training MobileNetV2 involves several key steps to achieve optimal performance. First, you need to prepare your dataset by collecting and preprocessing images, ensuring they are well-labeled and organized. Next, initialize the MobileNetV2 model architecture using a deep learning framework like TensorFlow or PyTorch. To leverage the power of transfer learning, use pre-trained weights from MobileNetV2 on a large dataset such as ImageNet, which can significantly accelerate training and improve the model's performance.

Optionally, you can add custom layers to adapt the model to your specific task, whether it's classification, object detection, or another computer vision application. Define the training parameters, including learning rate, batch size, and epochs, and then train the model using an optimizer like SGD or Adam, monitoring key metrics like loss and accuracy. Once training is complete, validate the model's performance on a separate validation dataset to assess its generalization capability. Finally, evaluate the trained MobileNetV2 model on a test dataset to measure its accuracy and overall performance, ensuring it meets the desired criteria for your application.

3.3.1 Data Preparation

Before training MobileNetV2, it is essential to prepare the data appropriately. This involves preprocessing the images, splitting the dataset into training and validation sets, and applying data augmentation techniques to improve the model's generalization ability.

3.3.2 Training and Testing the Dataset

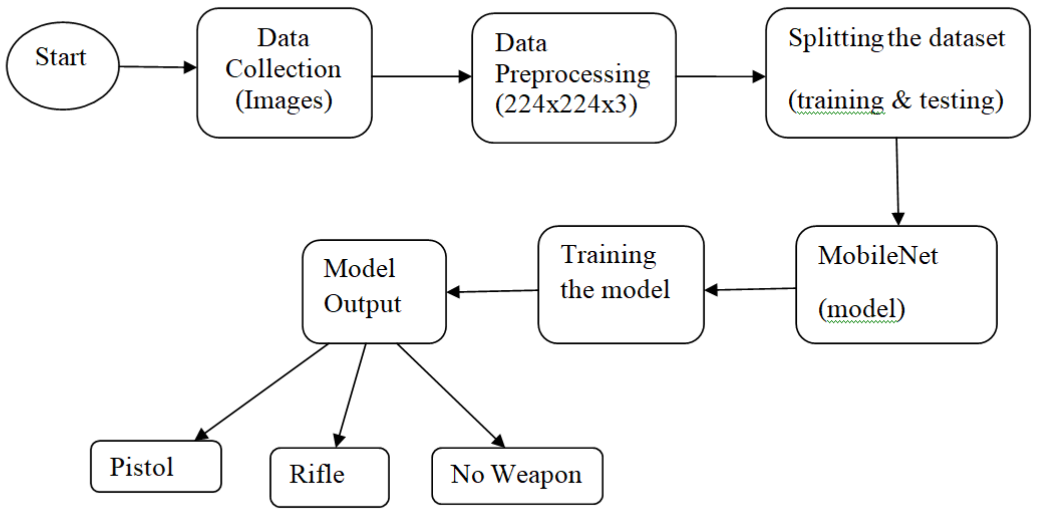


Figure 3.3 Model Design

Data Collection:

a. **Source Identification**: Identify reliable sources to collect images from. This could include public datasets, online repositories, or private collections.

b. **Data Acquisition**: Download or collect images containing weapons (e.g., rifles, pistols) and non-weapons (e.g., everyday objects, landscapes).

Data Cleaning:

a. **Data Inspection**: Inspect the collected data to identify any anomalies, corrupted images, or mislabeled samples.

b. **Data Preprocessing**:

   i. **Image Resizing**: Resize images to a uniform size suitable for training (e.g., 224x224 pixels).

   ii. **Normalization**: Normalize the pixel values to a range between 0 and 1.

   iii. **Augmentation**: We can apply data augmentation techniques to increase the diversity of the dataset (e.g., rotation, flipping, zooming).

c. **Data Splitting**: Split the dataset into training, validation, and test sets to evaluate the model's performance accurately.

d. **Handling Imbalance**: If the dataset is imbalanced (e.g., fewer weapon images compared to non-weapon images), apply techniques like oversampling, undersampling, or synthetic data generation to balance the classes.

e. **Data Verification**: Verify the correctness of annotations and remove any duplicates or irrelevant images.

i. Resizing the input image (256 x 256)
ii. Applying the color filtering (RGB) over the channels (Our model MobileNetV2 supports 2D 3 channel image)
iii. Scaling / Normalizing images using the standard mean of PyTorch build in weights.
iv. Center cropping the image with the pixel value of 224x224x3.
v. Finally, Converting them into tensors (Similar to NumPy array)

3.3.3 Transfer Learning

Transfer learning is a popular technique used with MobileNetV2 to leverage pre-trained models on large-scale datasets. By initializing the model with pre-trained weights, the training process can be accelerated, and the model can benefit from the knowledge learned from the source dataset.

3.3.4 Fine-tuning

Fine-tuning MobileNetV2 involves training the model on a target dataset while keeping the pre-trained weights fixed for some layers. This allows the model to adapt to the specific characteristics of the target dataset while retaining the knowledge learned from the source dataset.

3.3.5 Hyperparameter Tuning

Hyperparameter tuning plays a crucial role in optimizing the performance of MobileNetV2. Parameters such as learning rate, batch size, and regularization techniques need to be carefully selected to achieve the best possible results. Techniques like grid search or random search can be employed to find the optimal combination of hyperparameters.

3.4 Evaluating Performance of MobileNetV2

MobileNetV2 is optimized for efficiency, making it ideal for mobile applications. When evaluating its performance, consider accuracy, speed, model size, memory usage, power consumption, robustness, and the potential for fine-tuning. Accuracy measures correct classifications, while speed assesses inference time. Model size and memory usage impact deployment and storage, and lower power consumption is beneficial for battery life. Robustness tests the model's performance across various conditions. Comparing MobileNetV2 with other models can provide valuable insights into its strengths and weaknesses. Tailor your evaluation to meet the specific requirements of your application.

3.4.1 Metrics for Image Classification Evaluation

When evaluating the performance of MobileNetV2 for image classification, several metrics can be used. These include accuracy, precision, recall, F1 score, and confusion matrix. Each metric provides valuable insights into the model’s performance and can help identify areas for improvement.

3.4.2 Comparing MobileNetV2 Performance with Other Models

To assess the effectiveness of MobileNetV2, it is essential to compare its performance with other models. This can be done by evaluating metrics such as accuracy, model size, and inference time on benchmark datasets. Such comparisons provide a comprehensive understanding of MobileNetV2’s strengths and weaknesses.

3.4.3 Case Studies and Real-world Applications

Various real-world applications, such as object recognition, face detection, and scene understanding, have successfully utilized MobileNetV2. Case studies that highlight the performance and practicality of MobileNetV2 in these applications can offer valuable insights into its potential use cases.

4 Requirement Analysis

Requirement analysis is essential for understanding stakeholders' needs and defining the project's scope. It helps in identifying potential risks, allocating resources, and estimating costs and timelines accurately. This analysis sets quality standards, facilitates effective communication, and aids in managing changes throughout the project. Clear requirements ensure everyone has a shared understanding, reducing misunderstandings and improving project outcomes. Proper documentation from this analysis serves as a valuable reference for the project team and stakeholders. In essence, requirement analysis lays the groundwork for successful project planning and execution.

4.1 Requirements:

Requirement analysis identifies two main types of requirements: functional and non-functional. Functional requirements describe what the system should do, like allowing users to log in or transferring funds in a banking app. Non-functional requirements focus on how the system performs, covering aspects like performance, usability, and security. For example, response time for transactions or system accessibility is non-functional requirements. Both types are essential; functional requirements ensure the system's features meet user needs, while non-functional requirements ensure the system performs reliably and efficiently.

4.1.1 Functional Requirements:

The functional requirements encompass a systematic approach to developing a weapon identification system and a lung cancer detection model. The initial phase involves collecting a comprehensive dataset of annotated weapon images, including pistols, rifles, and "No Weapon" labels, to facilitate accurate weapon identification. Subsequently, the collected data undergoes rigorous preprocessing, cleaning, and augmentation to enhance its quality and prepare it for model training. A specialized deep learning model, potentially a Convolutional Neural Network (CNN), is then developed using frameworks like TensorFlow or PyTorch, tailored specifically for detecting lung cancer based on intricate image features.

Following the model's development, extensive training is conducted using the annotated dataset to enable the model to recognize and learn patterns indicative of lung cancer accurately. Post-training, the model's performance is rigorously validated using key metrics such as accuracy, sensitivity, and specificity to ensure its reliability and efficacy in real-world applications. To facilitate user interaction and accessibility, a user-friendly interface is designed and developed, allowing users to effortlessly upload medical images for predictions.

Furthermore, the model's generalization ability is assessed through comprehensive testing on a separate dataset to evaluate its performance across diverse scenarios and ensure robustness. Continuous optimization efforts are undertaken to fine-tune the model's hyperparameters and refine its architecture, aiming to enhance prediction accuracy and overall performance. Finally, the optimized and validated model is deployed in a production environment, enabling seamless integration and delivering real-time predictions for medical image analysis, thereby fulfilling the project's functional requirements.

4.1.2 Non-functional Requirements:

The system should prioritize Performance, ensuring fast and accurate predictions for both weapon identification and lung cancer detection. Scalability is crucial to accommodate a growing dataset and increasing user demand over time without compromising performance. Reliability is paramount, requiring the model to consistently produce accurate results with minimal errors or false positives/negatives to ensure trustworthiness in medical applications.

Usability is another key aspect, demanding a user-friendly interface that is intuitive and accessible to users with varying levels of technical expertise. Security measures must be implemented to safeguard sensitive medical data and ensure compliance with privacy regulations, such as HIPAA, during data collection, storage, and transmission.

Maintainability is essential for long-term success, necessitating well-documented code, modular design, and efficient update mechanisms to facilitate future enhancements and modifications to the system. Availability is critical, requiring the system to be operational and accessible 24/7 to cater to the needs of healthcare professionals and patients, especially in critical situations.

Lastly, Interoperability is vital, enabling seamless integration with existing healthcare systems and medical imaging technologies to ensure compatibility and efficient data exchange, thereby enhancing the overall functionality and utility of the system. These non-functional requirements collectively ensure the system's effectiveness, efficiency, and adaptability in delivering reliable and user-centric solutions for weapon identification and lung cancer detection.

4.2 System Requirements:

System requirements for gun detection typically involve a combination of hardware and software components. This includes high-resolution cameras or sensors capable of capturing detailed images, powerful processors for real-time analysis, and sufficient memory and storage for storing and processing data. Additionally, software requirements include robust algorithms for object recognition and classification, as well as user interfaces for system control and monitoring. Compliance with privacy regulations and standards may also influence system design and implementation.

4.2.1 Hardware Specifications:

i. Processor (CPU):
   a. Minimum Requirement: Intel Core i5 or AMD Ryzen 5
   b. Recommended: Intel Core i7 or AMD Ryzen 7 for better performance

ii. Memory (RAM):
   a. Minimum Requirement: 8 GB RAM or more for smoother performance

iii. Storage:
   a. Minimum Requirement: 50 GB HDD/SSD storage
   b. Recommended: 128 GB SSD for faster data access

iv. Operating System:
   a. Supported: Windows 10, macOS 10.12+, Linux (Ubuntu 18.04 LTS or higher).
   b. Ensure the chosen OS is compatible with Python and required libraries

4.2.2 Software Specifications:

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

a) OPERATING SYSTEM	:	Windows 7 and above.
b) FRONT END	:	Html, CSS.
c) FRAMEWORK	:	Flask.
d) LANGUAGE	:	Python version 3.7.
e) LIBRARIES	:	Pandas, Numpy,Keras,TensorFlow.
f) EDITOR	:	Jupyter Notebook.

5 Model Design

UML diagram can be used as a replacement for flowcharts. They provide both a more standardized way of modelling workflows as well as a wider range of features to improve readability and efficiency. Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system. Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use cases. Together this use case collection specifies all the ways the system. An association provides a pathway for communication.

The communication can be between use cases, actors, classes or interfaces. Associations are the most general of all relationships and consequentially the most semantically weak. If two objects are usually considered independently, the relationship is an association. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency. Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system. Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use cases.

The workflow in this case begins from importing the dataset by the developer and then replacing missing values with mean value of corresponding column, model building, validating that model by generating a confusion matrix and finally predicting the test sample class label. Transitions are used to show the passing of the flow of control.

The various UML diagrams are

i. Class Diagram
ii. Use Case diagram
iii. Data Flow diagram
iv. Activity Diagram
v. State Chart Diagram

5.1.1 Class Diagram

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

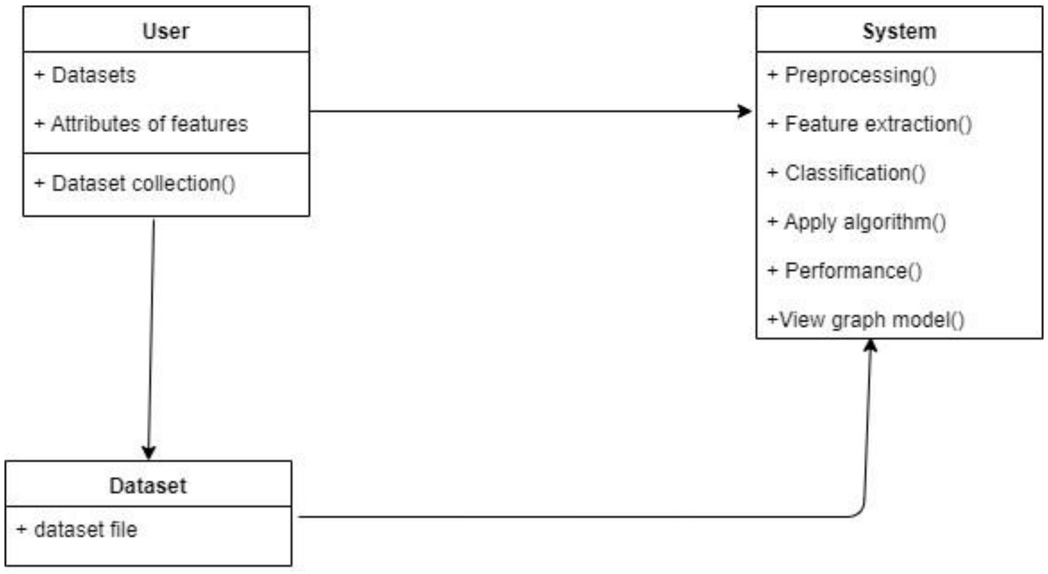


Figure 5.1 Class Diagram

5.1.2 Use Case Diagram

A use case diagram consists of an actor graph, a collection of use cases bounded by a system boundary, associations between users and actors in terms of communication (participation), and generalization between use cases. The behavior of the system is defined by the use case model on the outside (actors) and inside (use cases). The actors are outside the system. Anyone or everything that interacts with the system—inputs into it or receives output from it is represented by an actor. Use-case diagrams are a useful tool for capturing system requirements and understanding system functionality during analysis.

Use-case diagrams can be used in the design stage to define how the system will behave when it is put into use. A use case is a series of activities taken by a system that results in a quantifiable set of values for a certain actor. All possible uses for the technology are listed in the use cases.

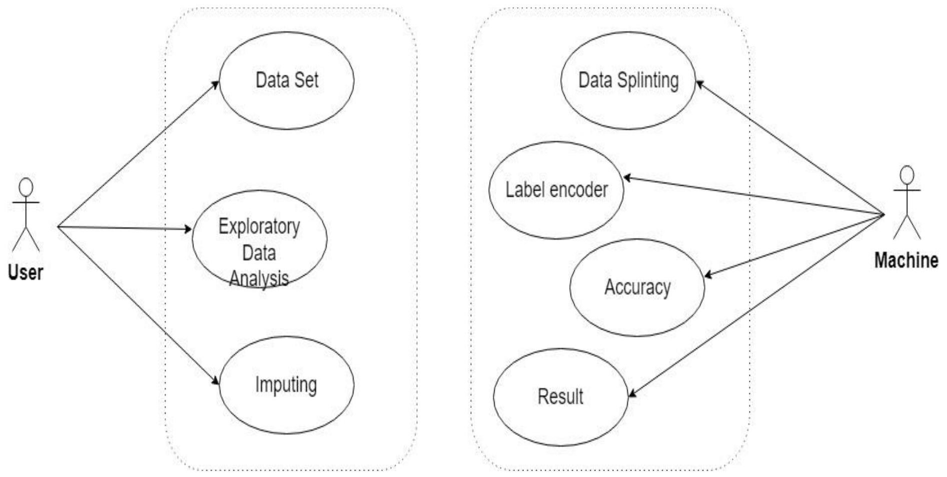


Figure 5.2 Usecase Diagram

5.1.3 Data Flow Diagram

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

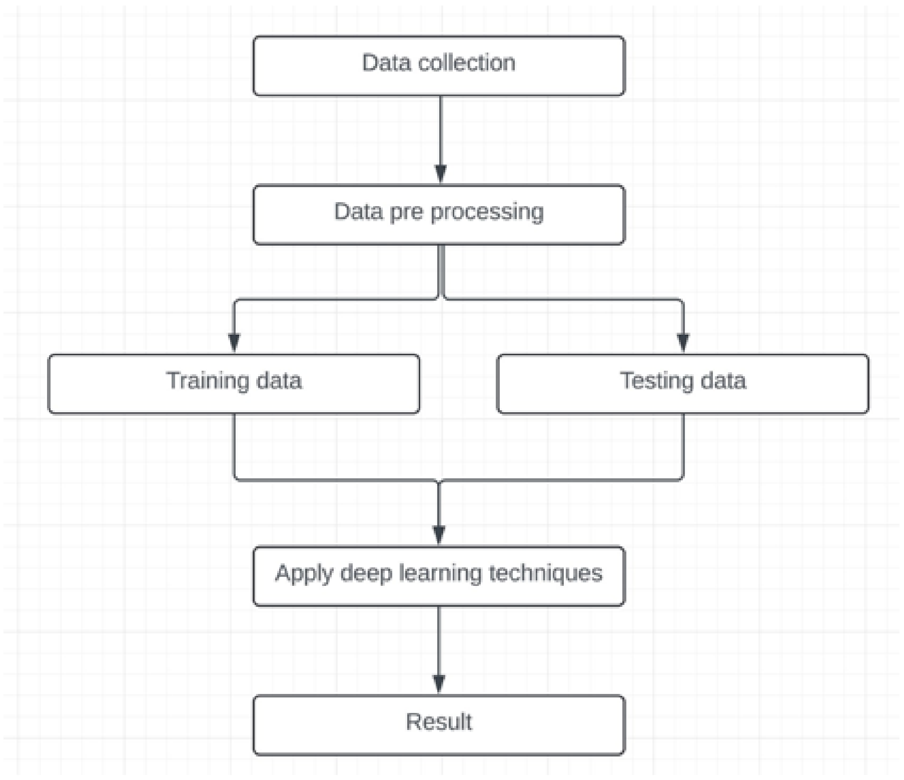


Figure 5.3 Data Flow Diagram

5.1.4 Activity Diagram

Activity diagrams are essential visual tools in software development projects, offering a clear representation of workflows and processes. They aid in understanding the sequence of activities, decision points, and interactions within the system, facilitating effective communication among stakeholders. These diagrams help in identifying bottlenecks and areas for optimization, improving system performance and user experience.

Additionally, they serve as valuable documentation for development, testing, and maintenance phases, ensuring consistency and clarity. Activity diagrams can be integrated with other UML diagrams to provide a comprehensive view of the system's architecture and behavior. Overall, they play a vital role in planning, designing, and implementing software systems by providing structured insights into the system's activities and workflows.

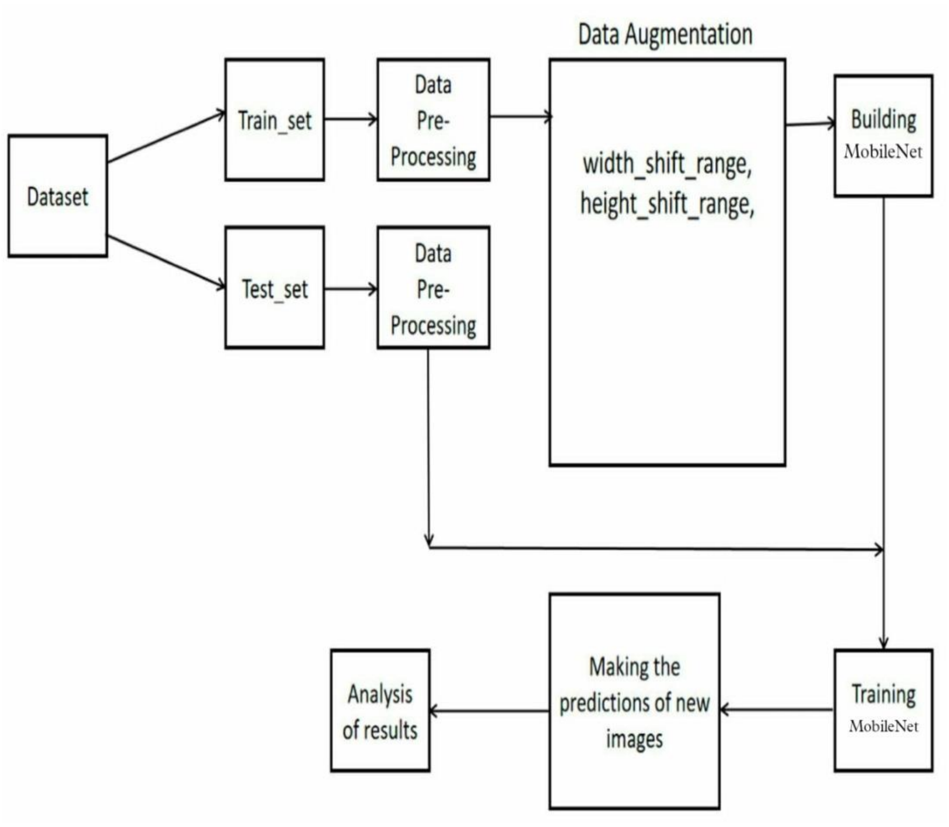


Figure 5.4 Activity Diagram

5.1.5 State Chart Diagram

The below state chart diagram describes the flow of control from one state to another state (event) in the flow of the events from the creation of an object to its termination.

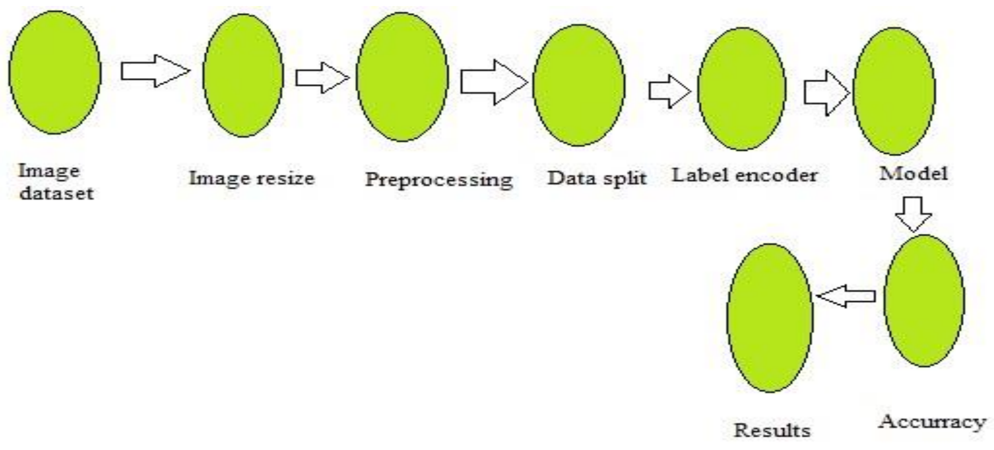


Figure 5.5 State chart Diagram

6 Implementation

Implementation is vital in computer science projects as it transforms ideas and plans into actual solutions. This phase bridges the gap between concept and reality, turning designs into functional software or systems. Successful implementation ensures that the project meets its intended goals and addresses real-world challenges. It's a critical step where coding, testing, and refining take place to create a working product. Proper implementation is key to delivering a reliable, efficient, and user-friendly solution. It validates the project's feasibility and effectiveness, setting the stage for deployment and further refinement.

6.1 Algorithm

An algorithm is a step-by-step procedure or formula for solving a problem. In computer science, algorithms are a fundamental concept used to develop software and solve computational problems efficiently. They provide a clear, unambiguous method for solving a specific type of problem and can be implemented in various programming languages to automate tasks or processes.

Here's an algorithmic overview of how MobileNetV2 can be used for weapon detection with an image size of 224x224:

Input: Receive an input image.

Preprocessing: Preprocess the input image with dimensions (224, 224, 3) (height, width, channels)

MobileNetV2 Base Model: Load the pre-trained MobileNetV2 model without the top classification layer (include_ top=False).Include the 'imagenet' weights to leverage pre-trained features.

Feature Extraction: Pass the preprocessed image through the MobileNetV2 base model to extract features. The output will be a tensor with spatial dimensions reduced from 224x224

Custom Classifier: Add a custom classifier on top of the MobileNetV2 base model for weapon detection. Flatten the output tensor from the base model to prepare it for the custom classifier. The final dense layer should have 3 units (pistol, rifle, no weapon) with a softmax activation function for classification.

Model Compilation: Compile the model using an appropriate optimizer (e.g., Adam) and loss function (e.g., categorical cross-entropy) for multi-class classification. Optionally, define metrics such as accuracy to monitor during training.

Data Preparation: Prepare the dataset with labeled images of pistols, rifles, and images with no weapons. Split the dataset into training (80%) and testing (20%) datasets.

Training: Train the model on the training dataset.

Evaluation: Evaluate the trained model on the testing dataset to assess its performance. Calculate metrics such as accuracy, precision, recall, and F1 score to measure the model's effectiveness.

Prediction: Use the trained model to make predictions on new images to detect the presence of pistols, rifles, or no weapons. This algorithm outlines the process of using MobileNetV2 for weapon detection in images.

To access the source code and project files, please visit our GitHub repository at https://github.com/ShaikDinisha/WeaponIdentification.git. The repository contains all the necessary files, including scripts, configuration files, and documentation, to reproduce and explore the project implementation. Feel free to clone the repository and contribute to the project.

7 Results

The F1 scores, precision, and recall metrics collectively evaluate the performance of each scenario in target identification and capture. "No Weapon" achieves exceptional results with precision and recall both exceeding 99%. "Pistol" exhibits commendable precision and recall scores at 92.49% and 88.07% respectively, resulting in a high F1 score. Conversely, "Rifle" shows slightly lower precision and recall scores of 76.14% and 85.04%, respectively, reflecting its effectiveness in target identification and capture. These metrics provide valuable insights into each scenario's performance.

Table 7.1 Evaluation Metrics

Metrics	Precision	Recall	F1 Score
Pistol	92%	88%	90%
Rifle	76%	85%	80%
No Weapon	100%	99%	99%

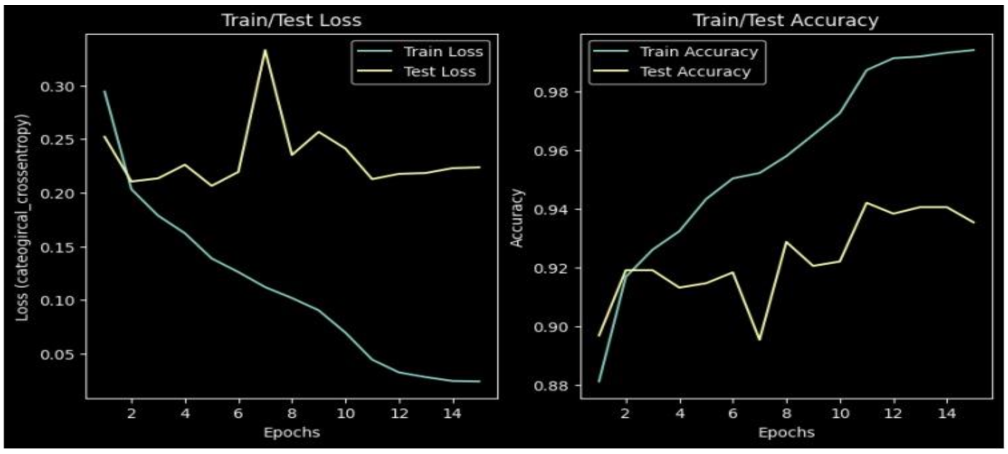


Figure 7.1 Accuracy

Case I: As we can observe in Figure 7.2 when we train the model with 2 epochs then we can see more number of wrong predictions.

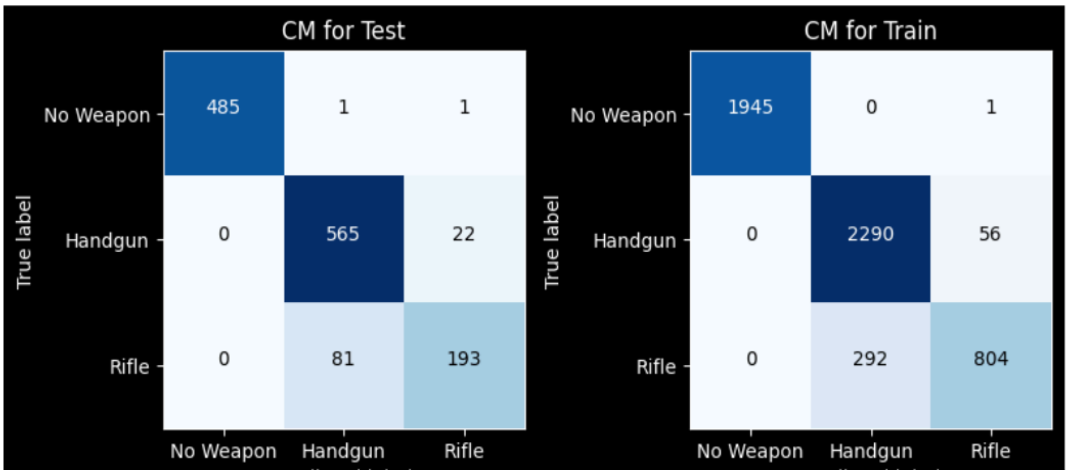


Figure 7.2 Confusion Matrix for 2 epochs

Case II: Similarly, we can also observe in Figure 7.3 here we train the model with 5 epochs there the number of wrong predictions are decrease when compared with Figure 7.2.

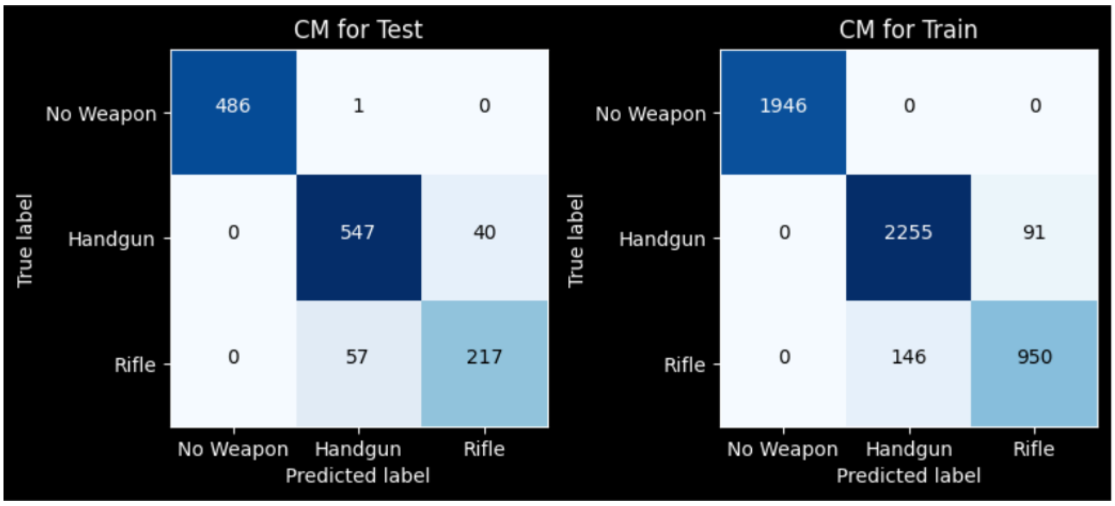


Figure 7.3 Confusion Matrix for 5 epochs

Case III: Finally, we can observe in Figure 7.4 here we train the model with 17 epochs there the number of wrong predictions are decrease when compared with Figure 7.3.

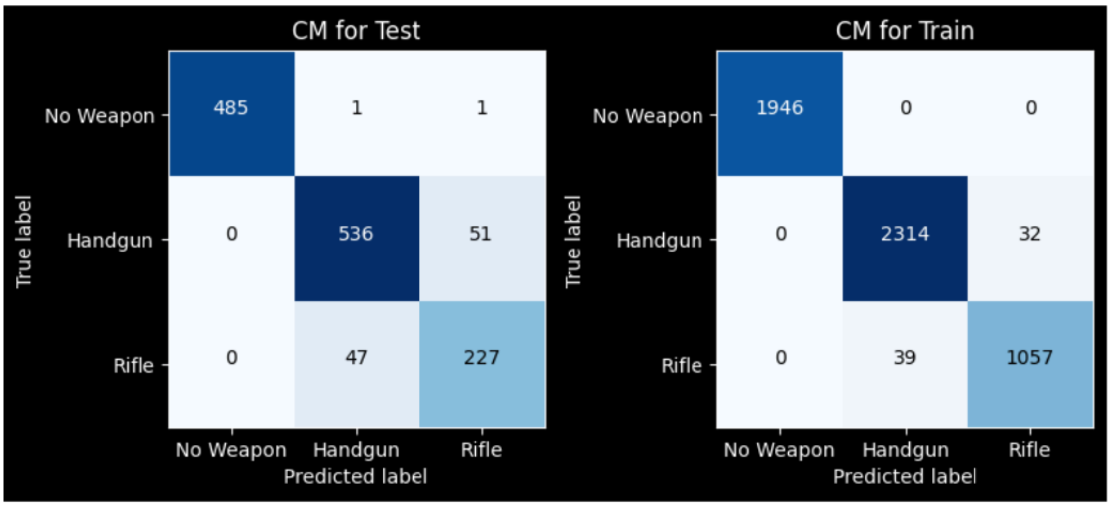


Figure 7.4 Confusion Matrix for 17 epochs

From Figure 7.2, Figure 7.3 and Figure 7.4 when the number of epochs are increasing in the training then the accuracy is also increased .We can observe from Figure 7.4 gives more accurate predictions than Figure 7.2 and Figure 7.3. Based on this observation we conclude that more number of epochs gives more accurate results. In future we are working on this model to increase more accuracy.

The UI for our project, Here we are using the flask to design the UI. The home page is looks like Figure 7.5 when we click on the “Try for Image” button. It will ask for the image as we see in the Figure 7.6. After that click on “Predict” button. The model will predict whether there is weapon or no weapon. If the model find there is a weapon it will give the weapon classification.

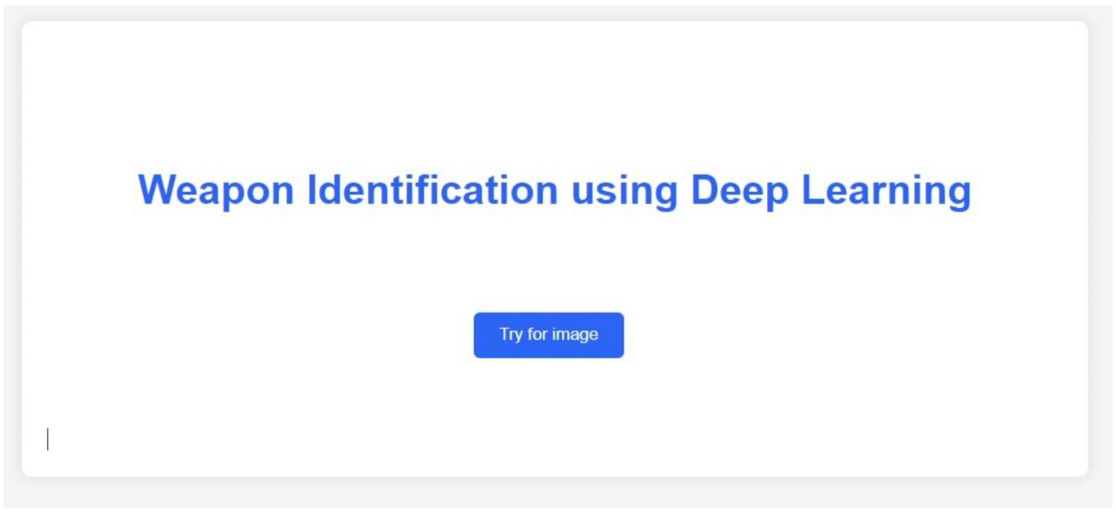


Figure 7.5 Home Page

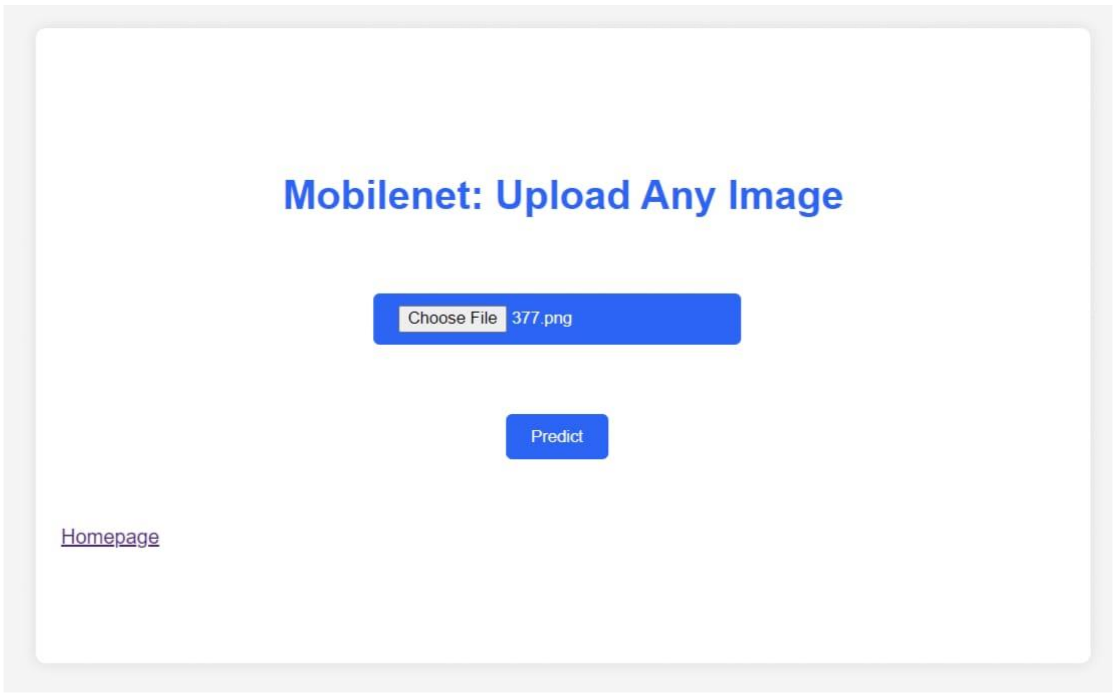
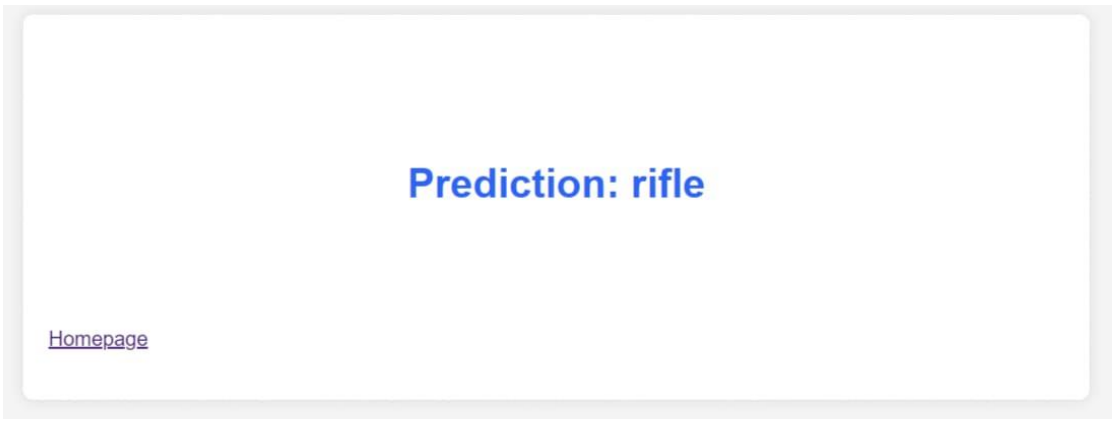


Figure 7.6 Input


Prediction: rifle

Homepage

Figure 7.7 Output

8 Conclusions

MobileNetV2 model for weapon identification by increasing the number of training epochs improves its accuracy, precision, recall, and F1 scores. comparing MobileNetV2 with YOLOv3 and YOLOv4, MobileNetV2 stands out for several advantages. Firstly, its speed and efficiency make it notably faster than YOLOv3 and YOLOv4. With pre-trained models available on datasets like ImageNet, it facilitates transfer learning for improved performance.

When comparing MobileNetV2 with YOLOv3 and YOLOv4, MobileNetV2 stands out for several advantages. Firstly, its speed and efficiency make it notably faster than YOLOv3 and YOLOv4, which is crucial for real-time applications and scenarios where low computational overhead is essential. Additionally, MobileNetV2's lightweight architecture makes it highly suitable for mobile and embedded devices. Another advantage of MobileNetV2 is its flexibility, offering a good balance between speed, efficiency, and performance.MobileNetV2 also comes with pre-trained models on datasets like ImageNet, facilitating transfer learning, which is a valuable feature for improving model performance with limited data.

Additionally, the accuracy, precision, recall, and F1 score of a MobileNetV2 model for gun detection would depend on various factors including the quality of the training data and the complexity of the detection task. These metrics are essential for evaluating the performance of the model and determining its effectiveness in real-world applications.

Finally, MobileNetV2's efficient architecture enables real-time processing, which is crucial for applications requiring quick decision-making. Overall, MobileNetV2's advantages in speed, efficiency, flexibility, and suitability for mobile and embedded devices make it a strong choice compared to YOLOv3 and YOLOv4 in various machine learning applications.

9 Future Scope

For future work, expanding the number of classes recognized by the MobileNetV2 model could be beneficial, allowing it to identify a wider range of objects. Additionally, efforts should be made to improve the accuracy of predictions and reduce false alarms. Moreover, integrating MobileNetV2 with other complementary technologies could further enhance the overall detection capabilities of security systems.

Continued research and development in these areas will be crucial for advancing weapon detection technologies and ensuring a safer environment for everyone. By enhancing the model's recognition abilities and minimizing false alarms, we can strengthen security systems and better protect against potential threats.

10 Bibliography

1. **Z. Zhao, P. Zheng, S. Xu and X. Wu.** *Object Detection With Deep Learning: A Review,.* doi:10.1109/TNNLS.2018.2876865. : IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, Nov. 2019. pp. 3212-3232.

2. **X. Bai, Xinggang Wang, L. J. Latecki, W. Liu and Z. Tu.** *Active skeleton for non-rigid object detection.* Kyoto : 2009 IEEE 12th International Conference on Computer Vision, 2009.

3. **W. Anser, M M Fraz, M Shahzad.** *Two Stream Deep CNN-RNN Attentive Pooling Architecture for Video-based Person Re-identification.* Madrid , Spain : Proceedings of the 23rd Iberoamerican Congress on Pattern Recognition, Nov, 2018.

4. **S. Batool, M. Z. Ali and Fraz, M. Shahzad and M. M.** *End to End Person Re-Identification for Automated Visual Surveillance.* Sophia Antipolis , France : Proceedings of the International Conference on Image Processing, Applications and Systems (IPAS),\, Dec, 2018.

5. **Rein-Lien Hsu, M. Abdel-Mottaleb and A. K. Jain.** *Face detection in color images.* s.l. : IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no.5, May 2002.

6. **R. M. Alaqil, J. A. Alsuhaibani, B. A. Alhumaidi, R. A. Alnasser, R. D. Alotaibi and H. Benhidour.** *Automatic Gun Detection From Images Using Faster R-CNN.* s.l. : 2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH), 2020. pp. 149- 154.

7. **N. Pervaiz, M. M. Fraz, M. Shahzad.** *Person Re-Identification Using Hybrid Representation Reinforced by Metric Learning.* s.l. : IEEE Access,Vol. 7 , No. 1, Dec, 2018.

8. **N. Pervaiz, M. M. Fraz, M. Shahzad.***Hierarchical Refined Local Associations for Robust Person Re-Identification.* Islamabad , Pakistan. : Proceedings of the International Conference on Robotics and Automation in Industry (ICRAI), Oct, 2019.

9. **M. Szarvas, A. Yoshizawa, M. Yamamoto and J. Ogata.** *Pedestrian detection with convolutional neural networks.* Las Vegas, NV, USA : IEEE Proceedings. Intelligent Vehicles Symposium,, 2005.