ABDUL HANNAN & HASSAN ALI
01-134181-002 & 01-134181-024

Detecting Specific Building Structure in Videos Using Deep Learning

Bachelor of Science in Computer Science

Supervisor: Ma'am Anum Kaleem

Department of Computer Science
Bahria University, Islamabad

January 20, 2022

Certificate

We accept the work contained in the report titled **"Detecting Specific Building Structure in Videos Using Deep Learning"**, written by Abdul Hannan & Hassan Alias a confirmation to the required standard for the partial fulfilment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Ma'am Anum Kaleem

_____

Internal Examiner: Name of the Internal Examiner (Title)

_____

External Examiner: Name of the External Examiner (Title)

_____

Project Coordinator: Dr. Moazam Ali

_____

Head of the Department: Dr. Arif Ur Rahman

_____

November, 2021

Acknowledgments

All of this is possible because of Allah Almighty who is the most beneficent and merciful, whose blessings made it possible for us to come this far. We are extremely thankful to ALLAH and in need of His blessings in future. We would like to thank our supervisor Ma'am Anum Kaleem, Senior lecturer at Department of Computer Science Bahria University, Islamabad. We are grateful for her efforts and dedication towards our project which helped us in the implementation of our idea. With her constructive feedback on every step, we were able to achieve what we were looking for in the best possible way.

ABDUL HANNAN & HASSAN ALI
Islamabad, Pakistan

November, 2021

"The goal is to transfer data into information and information into insight."

Carly Fiorina

Abstract

Data is growing at a colossal rate with the advancement of technology. Video data has massively increased as well, which comprises of images, videos, and audios. The digital videos are sequence of images containing a lot of information. The process of searching is tedious and time consuming, not to mention the amount of time it generally takes to search for something specific. Searching in images and videos is especially challenging. Therefore, there is a need for a system that can effectively detect and recognize certain building object(s) in videos. The idea behind our application is to make content based searching easier and to provide a system that can accurately detect specific building structures from a wide range of videos.

We have chosen six famous buildings of Pakistan for detection. These six famous structure are, Faisal Mosque, Minar e Pakistan, Pakistan Monument, Centaurus, Parliament Building and Bahria University Islamabad Campus. The system detects these specific structures with a precision of 93.75 percent and a recall of 89.05 percent. The system using Convolutional Neural Network CNN to accurately detect different structures, a bounding box is drawn around the detected structures and confidence score is displayed.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
SSD	Single Shot Multi box Detector
RPN	Region Proposal Network
FCNN	Fuzzy Clustering Neural Network
SVM	Support Vector Machine
DCNN	Deep Convolutional Neural Network
YOLO	You Only Look Once
GPU	Graphics Processing Units
XML	Extensible Markup Language
GUI	Graphical User Interface
TF	Tensor Flow
ERD	Entity Relationship Diagram
VOC	Visual Object Classes
CSS	Cascading Style Sheet
HTML	Hypertext Markup Language
API	Application Programming Interface

Chapter 1

Introduction

1.1 Introduction

Data is growing at a colossal rate with the advancement of technology. Video data has massively increased as well, which comprises of images, videos, and audios. The digital videos are sequence of images containing a lot of information. We want to search out specific data from this sequence of images. Our project is about detection and recognition of specific building structures from videos. We will be developing this project to ease the process of finding objects, which is an arduous task when done manually. We want to develop the desktop application to ease users to search specific building structures in videos without manually going through all of the videos. In 2019, 500 hours of videos were uploaded on YouTube every minute. This equates to approximately 30,000 hours of newly uploaded content per hour. The amount of content on YouTube has increased dramatically as consumer's appetite for online video has grown. In fact, the number of video content hours uploaded every 60 seconds grew by around 40 percent between 2014 and 2019[1]. There should be an application that can analyze the video frame by frame and detect specific objects. A human searching for some object can be tiresome and tedious, we can automate this process. Using our application, a user can easily detect specific objects. We used Convolutional Neural Network algorithm to detect specific building structures in videos. In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

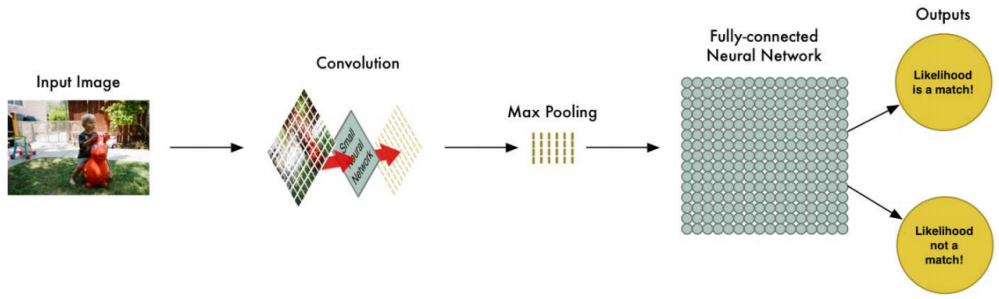

Figure 1.1: Convolutional Neural Network Diagram

1.2 Objective(s)

The detection and recognition of objects from videos is quite difficult, tiresome, time consuming as well as cost consuming if done manually. Also, manual working requires big team for searching purpose and for 24/7 monitoring of all video data. Following are the main objectives of our project.

- To automate the retrieval of specific real-world building structures from videos using image processing and deep learning techniques to get rid of manual searching of videos.
- To check out each frame of a video in search of the specific building structure.
- To make content search from videos cost and time saving.
- To give an accurate description of the specified (detected) object.

The foremost aim of this project is to automate the retrieval of specific building structures from videos using deep learning and image processing techniques. This task is tiresome and tedious, but it can become easy by this desktop application since this application will check out each and every frame of a video. Each frame is detected, recognized, and is given a bounding box, label, and confidence score. The aim of this project is to make content search from videos cost and time saving.

1.3 Problem Description

Object recognition from images has been in limelight now a days and so does the video-based object detection, neural networks, artificial intelligence, and machine learning. To take out a required content from videos is really a hectic job as it requires multiple people to just sit and go through all videos in order to get the required videos. Therefore, a

problem arises that how to retrieve object from videos using some technique to save time, money, and human effort. Using the conventional approach this issue can be resolved but the system can work only to a certain level and on several videos the system will fail to give accurate results. Thus, deep learning is the most convenient technique to cater this problem. The project we have developed is a desktop application based on image processing, deep learning, and artificial intelligence to recognize objects from videos. This is totally an automated system of recognizing video content. It will reduce the manual effort of finding out required videos by watching every single video as this takes many days. Apart from time saving, this application is also cost saving as for manual working, we have to hire multiple but using this app we only require one member to deal with this application. Detection and recognition of objects in an image is very difficult but compulsory task, because now there is an immense data in the form of videos and images. There are so many applications that extract and recognize the objects, but only generic object like building, person etc. but less work has been done on specific objects like building names or person names. This application titled 'Detecting specific building structures in videos using deep learning' recognizes the specific buildings along with labeling their names in videos i.e. it is not confined to bound box the building but it also labels the name of that building and also shows a brief description of the building and the confidence score too. Confidence score is the measured value which shows how much an object is trusted and how much similar it is to trained data. This is a challenging task due to different sizes of images, different resolution of videos and several buildings of nearly similar design and color. This application has three tasks to do i.e. detect the object and then recognize which object it is actually and lastly show a brief description of the object. Therefore, it is more sophisticated system.

1.4 Methodology

Our proposed solution includes creating a Desktop Application that takes video as input and detect any object that our algorithm is trained on. The proposed method of this project is primarily based on techniques of Image processing, deep learning, and artificial intelligence. We will use convolutional neural networks and fully connected convolutional neural networks for object recognition. This proposed system recognizes the objects from videos (i.e., frames) with the help of artificial intelligence and deep learning techniques. From initial understanding of the project, following steps will be taken to extract objects from video.

- System will be provided with images to train on, and to retrieve features from.
- A data-set will be collected, and all the images will have to be labeled.

- System will use Convolutional Neural Network and Fully Connected Neural Networks to Detect Object.
- Testing will be done, and results will be matched.
- If the object is detected, a brief description of the object will be displayed.

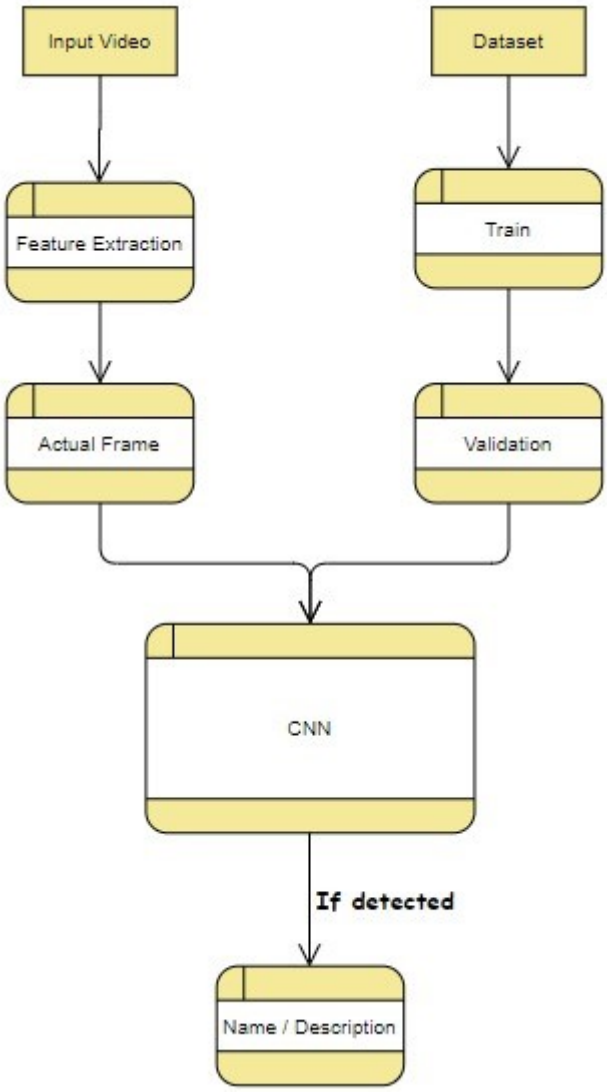

Figure 1.2: Data Flow Diagram

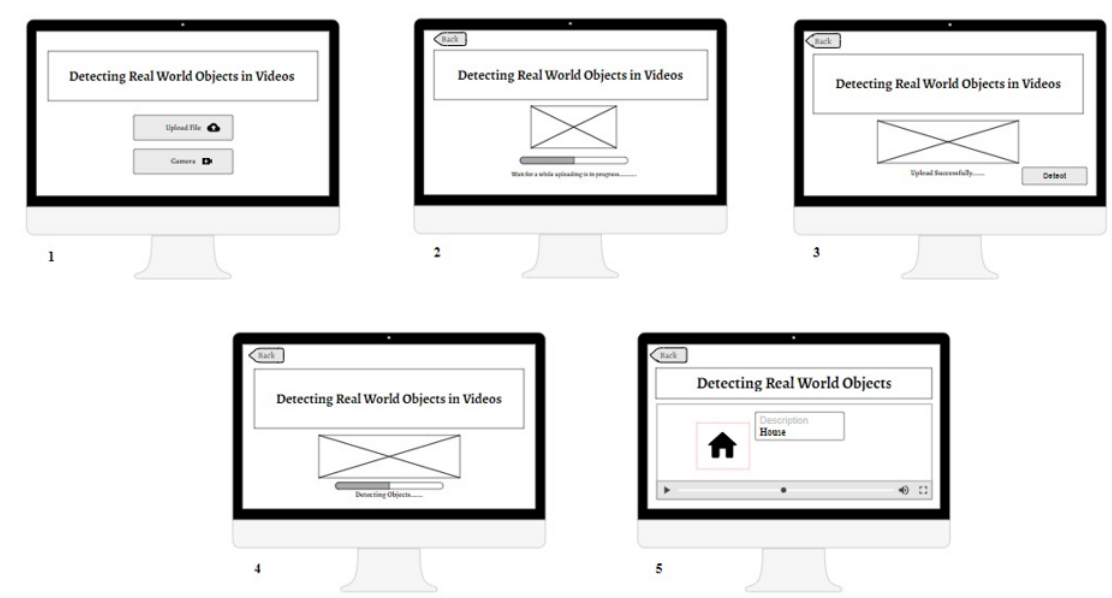

Figure 1.3: Low Fidelity Prototype

1.5 Project Scope

Deep learning has a wide scope all over the world in every industry. In a broader scope various kind of objects can be recognized from videos like person, cars, behavior etc. Our proposed project will create a Desktop Application that will be able to detect and recognize specific buildings from a video and give a brief description of that building. The system will receive a video as an input and the proposed desktop application will scan the whole video frame by frame and if the building is there, it will detect it and show a brief description of that specific building.

1.6 Solution Application Areas

This project could be used in variety of different fields depending on the data it is trained to detect. Our proposed solution could be used in media industry or could be used by the government to put censorship or surveillance on videos. This project can be used by any regulatory body for keeping track of activities and to control the blocked content from being broadcast-ed on Television. Following is some of the areas our desktop application can be used in:

- Can be used by Media houses for keeping track of activities.
- Can be used by government for tracking blocked content.

- Content-based image retrieval.
- Can be used by tourists.
- Can help navigate the city without tour guides.
- Efficient content-based location search.
- Can pinpoint exactly the place, where the video was recorded?

Chapter 2

Literature Review

2.1 Literature review

The respective section, aims to give a detail study of the work completed by a number of scientists and researchers on object detection and recognition. A wide range of methodologies are utilized by researchers to implement an object recognition system, which likewise incorporates the challenge to achieve an efficient system in minimum budget. Subsequently a wide range of algorithmic techniques have been suggested by various researchers to device efficient and ideal systems. One of those techniques is deep learning.

2.2 Deep Learning

Deep learning is a type of machine learning and artificial intelligence that mimics the activities of human brain; In terms of processing data and formulating patterns which can help in decision making. Deep learning is a sub field of machine learning comprised of different algorithms of AI. It provides capability to the system to learn unstructured data and process the information like the human brain. Deep learning gives computer exceptional computational platforms for example the ability to find objects, translate languages, identify speech, and make predictions etc.
Mostly deep learning methods involves neural networks. It comprises of two important phases such as training and constructing. The training phase includes preparation like labeling enormous amount of information and recognizing their similar features. The framework analyzes those attributes and memorize them to come up with some prediction. In construction phase, the model makes deductions based on preliminary information and identify/label/classify unknown data/information.

2.2.1 Advantages

Following are some advantages of Deep learning:

- It helps to analyze huge amount of unstructured data.
- The deep learning approach has the capability of execution of feature engineering by itself.
- In this approach, algorithm finds correlation between features and combine them to make efficient learning.
- Deep learning model allows to perform many tasks repetitively and deliver high quality results because it learns from the previous predictions.
- Deep learning models helps us to recognize defects which would be hard to recognize otherwise.

2.3 Related Work

Object recognition is a vital technology for artificially intelligent applications like automatic travel bus control, visions for robot, and surveillance. The Object Recognition is considered as a tough one in Image Processing. Recognizing an object is a vital part of computer vision since it is closely associated to the accomplishment of many computer vision applications. A complete study on Object Detection Method from Manga Images using CNN was proposed by Hideaki Yanagisawa, T. Yamashita, and Hiroshi Watanabe[2]. They examined the effectiveness of different object detection frameworks in detecting object in manga pictures. Algorithms like CNN, R-CNN, fast and faster R-CNN (an improved version of CNN) and SSD are used for object detection. It is concluded that selective search (R-CNN) is effective for objects with clear boundaries and RPN (another version of R-CNN) is effective for objects whose boundaries are ambiguous while SSD encounters difficulty in detecting manga objects in whole image. Following is the accuracy concluded experimentally in the paper[2].

Table 2.1: Comparison between accuracy of different algorithm

	Fast R-CNN	**Faster R-CNN**	**SSD**
Panel layout	0.959	0.9530	0.897
Speech Balloon	0.969	0.961	0.907
Character Face	0.810	0.816	0.765
Text	0.740	0.898	0.866
mAP (mean AP)	0.870	0.910	0.859

K Visakha and Sidharth S prakash in 2018, proposed detection and tracking of human beings in a video using Haar classifier. The Haar classifier is trained using hundreds of sample views of various human beings, called positive samples, as well as arbitrary images without human presence, called negative samples, all scaled to the same size. After the training, it can be applied to a region of interest in each of the extracted frames. The output for this process will be a "1" if the region is likely to contain a human being; and "0" otherwise. This paper focuses on detection of only human beings in a point of interest using the Haar classifier. Tracking of human beings is done using sampling and re sampling algorithms. The next position where that human can be found is predicted with accuracy. Positions are represented visually using unique colored rectangles enclosing the human being. Compared to other features, using Haar-like feature brings about an increase in the calculation speed and this strategy can be done for any object, by training the Haar classifier accordingly. Another research artifact which was published by M Mazumdar in 2017 deals with Object Recognition in Videos by Sequential Frame Extraction using Convolutional Neural Networks and Fully Connected Neural Networks. The paper provided a method that performs object detection in videos using CNN and FCNN. His method performs supervised learning and classification of objects by extracting the frames from videos, running the classifier and thereby get probabilities for different classes and hence classify the genre and also detect any object in the video. Moreover, results from experiments on different videos shows an increase in accuracy. Increasing the size of data set and changing the hardware configuration to see if the accuracy will increase or not is also experimented in that paper[3].

Table 2.2: Classification with 10 data sets

Video Class	**Forest**	**Desert**	**Living room**	**Train**	**Mountain**	**Kitchen**	**Highway**	**River**	**Tower**
Accuracy	0.95	0.94	0.77	0.89	0.80	0.89	0.79	0.73	0.71
Average Accuracy	77.35%								

Table 2.3: Classification with 20 data sets

Video Class	**Forest**	**Desert**	**Living room**	**Train**	**Mountain**	**Kitchen**	**Highway**	**River**	**Tower**
Accuracy	0.95	0.94	0.77	0.89	0.80	0.89	0.79	0.73	0.71
Average Accuracy	77.35%								

Videos which were shot in forest showed higher accuracy than videos shot in high altitude locations. Moreover, less effect of changing or increasing the data-set is recorded on the overall accuracy.
Another research was conducted on Object Recognition by Hanavi and F.Hidayat in 2020.

If we investigate this research paper, we can find out that how artificial intelligence, machine learning, image processing and computer vision can improve the Conventional Surveillance System. The increasing challenges to better our security systems, this solution can identify suspicious objects in public places etc. For this purpose, researcher recognize that objects depend on the abstraction ability of the data received. For data abstraction problem there is a branch of machine learning which is known as "Deep Learning". There is a class available in deep learning that is capable of autonomous learning for object recognition, object extraction, object classification. This deep learning architecture can also be applied to high-resolution images with a nonparametric distribution model (A model in which data are not taken from established models that are derived from a small number of parameters.) is the Convolutional Neural Network (CNN). This research paper consists of three parts. In first part, researcher just identify that how he collected the data for this system and classify that which model is best for identifying suspicious objects. Second part answers the question that why detection in videos object is important. In third part, this system contains what type of data sets, methods and frameworks can be useful to detect suspicious objects in videos. CNN model produces better results than SVM model. 5822 images were trained on both the models. The result proves that SVM model can identify objects with an accuracy of 56% on the other hand CNN model can identify objects with an accuracy of 74%. Data set used for object detection is Caviar/Left bag pickup, Caviar/Left bag (theft), Caviar fight, Pets2006/S1C3 and Pets2006/S2C3. Framework they used for this system is Kyungroul Framework, Surveillance framework, Smart Surveillance Framework, Rise Framework and Edcar Framework. This can detect suspicious objects according to features for example suspicious face, suspicious activity etc. This research paper concludes that how we improve the security systems through intelligent video analytics for suspicious object detection. This research paper classifies that which model, data-sets, frameworks is suitable for this system[4].

A research paper which is issued in 2020 by X. Wu, D. Sahoo, and S. C. H. Hoi included details regarding advancement in the object detection. The Researcher of this study evaluated that; a lot of work has already been done on Object detection, but the researchers have used very basic techniques to identify object, which is not enough for object detection. But in this research, the researchers noticed that their object detection can be made better with the usage of semantic segmentation or instance segmentation. This research paper aims to solve the problem of distinguishing between multiple objects of the same category. For this purpose, they introduced a new technique, which is comprises of two algorithms (object detection and semantic segmentation) and this new technique is called "Instance segmentation". Overview of method is: (a) Image Classification only needs to assign categorical class labels; (b) Object detection predicts categorical labels but also identifies the location of the instances using bounding box; (c) Semantic Segmentation is used to predict categorical labels for each pixel without differentiating instances; (d)

Instance Segmentation differentiates different objects by pixellevel segmentation masks. For the working of this method, programmers used deep convolutional neural networks (DCNN) for image classification. DCNN is a biologically inspired structure for computing hierarchical features. This spatial-invariant model for image classification was proposed by Fukushima. Currently, object detection using deep learning can be divided into two families: (1) Region-based CNN (R-CNN) and (2) one-stage detectors, such as YOLO (You Look Only Once). However, R-CNN faces some problems. Researcher proposed SPP-net (Spatial Pyramid Pooling) to faster R-CNN as well as learn discriminative features. A lot of research has been done on this method in the past years such as (R-CFN, YOLO9000, DCNN, MASK R-CNN, Cascade RCNN, SSD, CORNERNET, and RETINANET). The major goal achieved by this research paper is we got know various methodologies for object detection. We also got a comparison which one is better. From this research paper, we can conclude that a lot of advancement has been done to improve deep learning techniques for object detection. According to this research object detection are divided into three categories: object detector components, machine learning strategies, and real-world applications[5].

2.4 Summary/Comparison table of literature

Table 2.4 is a summarized view of whole discussion on previous related experiments done by several researchers.

Table 2.4: Comparison between Literature Review

S. No	Description	Algorithm	Year	Accuracy
1	Intelligent Video Analytic for Suspicious Object Detection: A Systematic Review	Two algorithms are used in this research paper: • CNN • SVM In this research paper, they compare the efficiency of these algorithm.	2020	CNN model can identify objects with an accuracy of 74%. SVM model can identify objects with an accuracy of 56%.
2	Neurocomputing	"Instance Segmentation" technique is used. For the working of this method, programmers used deep convolutional neural networks (DCNN) for image classification. • Region-based CNN (R-CNN) • YOLO (You Look Only Once)	2020	75%-80%.
3	Recyclable Material Detection in Video Streams using Neural Networks.	For the fast detection and localization of waste, R-CNN algorithm-based architecture with Inception V2 has been introduced.	2018	98%
4	Improved Inception residual Convolutional Neural Network for Object Detection.	Recurrent Convolutional Neural Network.	2017	72.78%.
5	Object Recognition in V ideos by Sequential Frame Extraction using Convolutional Neural Networks and Fully Connected Neural Networks.	FCNN,CNN	2017	77.35%

Chapter 3

Requirement Specifications

3.1 Requirement Specification

This chapter illustrates the project requirements to make an agreeable connection between the user and developer. This section goes about as primary documentation for the design phase. This section illustrates all the functional and non-functional requirements for the proposed application.

3.2 Proposed System and Scope

This application titles 'Detecting Specific Building Structure in Videos Using Deep Learning' is aiming on the recognition of objects from videos. So far, we have worked hard to get reasonable accuracy on recognition of buildings. Query to this application is a video which may or may not contain the object which needs to be detected. The system goes through all the video and if it detects the label in the video it draws bounding box and displays the video with confidence score and a short description of the detected label. For the object detection we have fine-tuned CNN model and for object classification SSD Mobnet v2 has been used. Furthermore, feature extraction was done by SSD (Single Shot Multibox Detector).

3.2.1 Operating Environment

The technologies required for this project are listed below:

3.2.1.1 Software

- Microsoft Windows 10

- Microsoft Visual Studio 2017 (.net Framework) or newer version.
- Python 3.9 or newer version.
- LabelImg (tool for image labelling).
- Visual Studio Code.
- Anaconda Navigator.

3.2.1.2 Libraries

- Tensor Flow
- Open CV
- Flask

3.2.1.3 Languages

- Python
- Html
- CSS
- Bootstrap

3.2.2 Constraints, Assumptions and Dependencies

This section encloses all constraints, assumptions, and dependencies of this web application 'Detecting Specific Building Structure in Videos Using Deep Learning'.

3.2.2.1 Constraints

- Audio with the video cannot be manipulated. Input is video and output is videos frame(s).
- System works more efficiently on colored images.
- During the processing of first task, another task cannot be allocated to the system to avoid system downtime.
- Training and retrieval from videos require GPU to be installed in the system.
- Only one video or image can be processed at a time.

3.2.2.2 Assumptions

- Training images are colored.
- GPU is necessary for better speed.

3.2.2.3 Dependencies

- The implementation for this system requires highly skilled professionals.
- System is completely dependent on GPU for speedy training and video processing.
- A good quality hard drive is required to store huge data set efficiently.
- Time required in generating output i.e., retrieval of objects from videos is directly dependent on quality of input video.
- Accuracy of algorithm's object recognition is dependent on data set therefore data set must be correctly labeled with proper names, and proper XMLs must be created.
- System quality can be better assured with the help of user feedback.

3.3 Functional Requirements

This section outlines the foremost functional requirements, which must be fulfilled by this application titled 'Detecting Specific Building Structure in Videos Using Deep Learning' for fruitful deliberation. This unit comprises portrayal of core functionality, technical matters, profits, and risks that may occur during the implementation phase.

3.3.1 Description

The core functionality of this application is to detect and recognize building object in videos. We have 6 classes of buildings. Application is given an input video which may or may not contain the label. This image is processed through neural network model, and it then returns frames containing those buildings. Model will process all frames (videos) and will detect buildings. After detection, buildings will be recognized. In the end, recognized building will be given a label, bounding box, confidence score and a brief description. All the resultant frames will be saved in a folder. All videos having searched building will be displayed to the user along with their frames having those detected buildings.

3.3.2 Technical Issues

The system may give incorrect or less accurate results if video have low resolution or less distinction between the background and object. To cater to this problem, it is preferred to use high resolution images and huge data set for our mode. Similarly, we should collect extensive training data for our model with the goal that it may give best results. The more the training data, more accurate the results will be.

3.3.3 Benefits

This application can be deployed at media houses and regulatory bodies to monitor the content being broadcast ed like sports, news, entertainment etc. Instead of monitoring individual channel manually 24/7, the application will reduce human effort.

3.4 Non-Functional Requirements

The non-functional requirements of this application are explained below:

3.4.1 Performance Requirements

System must be fast. Data processing for detection and recognition must be fast. System must take less time to recognize object from videos. System must extract only relevant information from videos.

3.4.2 Safety Requirements

Images and videos must be clean from malicious virus which can crash system. At a time only one query video will be processed in order to avoid system downtime.

3.4.3 Security Requirements

System must not be hack able, and user may not have privilege to access or modify back end code. It must be a fully secure application in order to avoid its misuse.

3.4.4 Space Requirements

Memory must be efficient and must store large amount of data. Images must be in separate clusters of data sets.

3.4.5 Dependability Requirements

Object detection from videos depends on image quality therefore the input frame must be clear and of good resolution and quality. Images must not be blur or vague. Frame extraction time depends on video quality, length, and resolution.

3.4.6 Operational requirements

Graphical User Interface must be designed in simple and decent way. Front end must be easy to use and adheres to the concept of human computer interaction. GUI must be simply but intriguing for user and should be easily understandable for any novice user.

3.4.7 Development Requirements

For implementing this idea, we require GPU for faster processing. It requires fast processing machine for example, a fast hard drive for quick results. Camera is required for taking images or snaps of objects.

3.4.8 Environment Requirements

User must give a basic knowledge of using computers. Deployment machine must be fast and efficient to run this application.

3.4.9 Usability Requirements

System needs to be error less to give accurate outputs. It must be an easy-to-use system, even a new person should be able to get use to this system in under an hour. System should generate minimum error per hour.

3.4.10 Regulatory Requirements

Patent: This is our own idea which require patent for avoiding claim. Trademark: Our project name is decided by us, and we will design the logo ourselves therefore there should be proper trademark.

3.5 Use Cases

Use case is a procedure used in system investigation to classify, simplify, and establish system requirements. The use case comprises all probable sequences of communications between users and systems in a specific scenario. There are two types of use cases, pictorial

and use case table. Utilize case have an on-screen actor which communicate with system. The use case diagram of our application is given below.

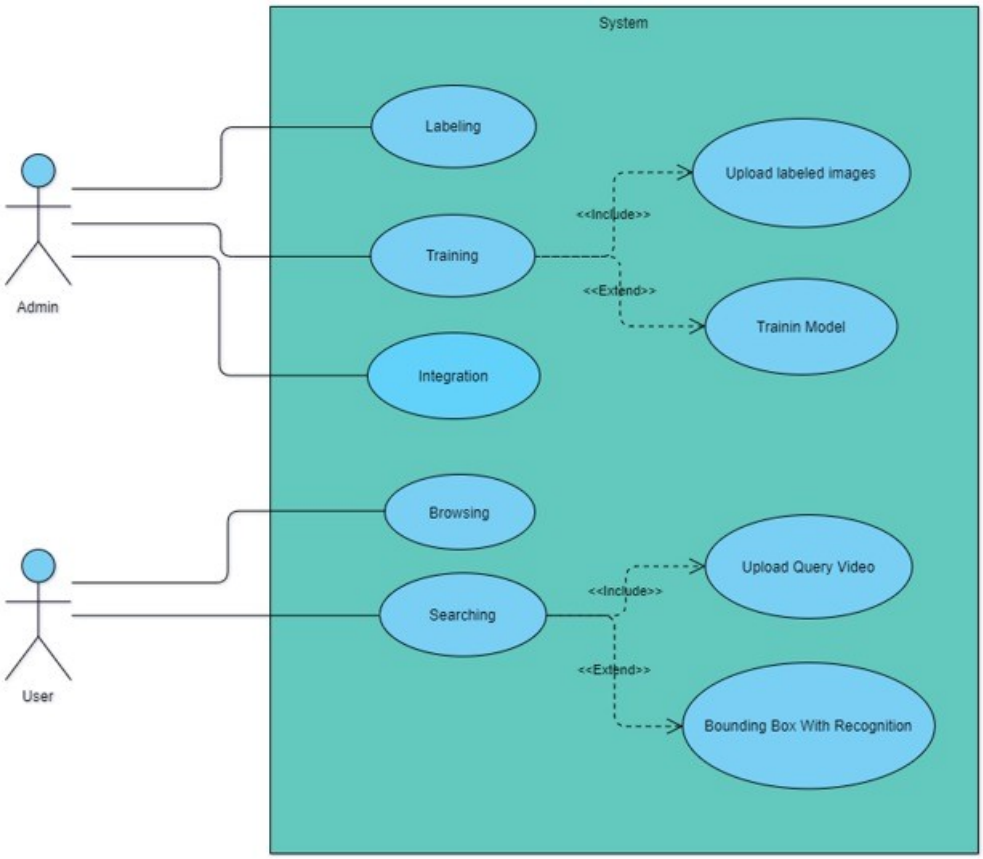

Figure 3.1: Use Case Diagram

3.5.1 Labeling Use Case

The following use case deals with the labelling functionality of our application and can be visualized in figure 3.2 and is explained in table 3.1.

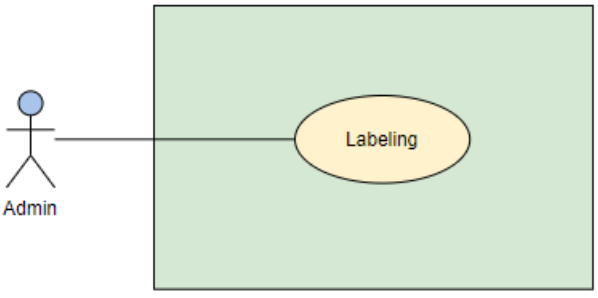

Figure 3.2: Labeling Use Case Diagram

Table 3.1: Labelling Use Case Description

System	**Detecting Specific Building Structure in Videos Using Deep Learning**
Use Case	Labelling
Actors	Admin
Triggers	Admin labels all training images.
Pre-conditions	Data-set images should possibly cater to all situations.
Basic Flow	1. Admin draws bounding box on the object. 2. Labels the bounded area with a name. 3. Save XML of every image.
Post Condition	All images get saved along with their XMLs.

3.5.2 Training Use Case

The following use case deals with the Training of our application and can be visualized in figure 3.3 and is explained in table 3.2.

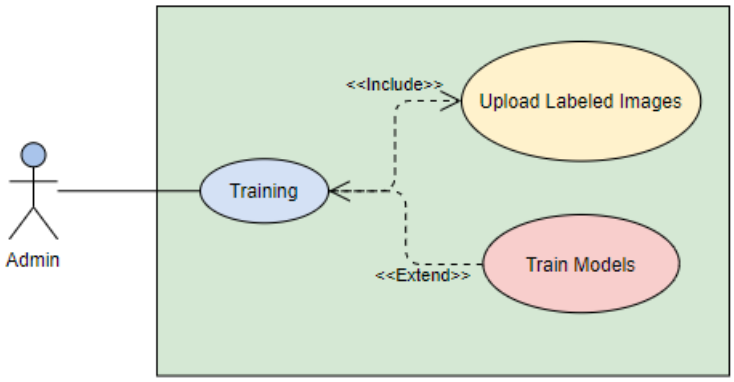

Figure 3.3: Training Use Case Diagram

Table 3.2: Training Use Case Description

System	**Detecting Specific Building Structure in Videos Using Deep Learning**
Use Case	Training
Actors	Admin
Triggers	Admin attempts to train the system.
Pre-conditions	Admin must have labeled Images.
Basic Flow	1. User uploads labeled images. 2. System train models on the user uploaded data. 3. System generates message successful.
Post Condition	The system gets trained on the given data.

3.5.3 Integrating Use Case

The following use case deals with the integration of our application and can be visualized in figure 3.4 and is explained in table 3.3.

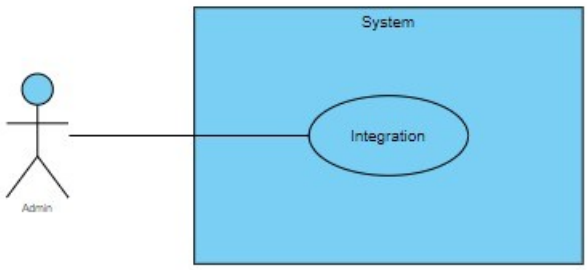

Figure 3.4: Integrating Use Case Diagram

Table 3.3: Integrating Use Case Description

System	**Detecting Specific Building Structure in Videos Using Deep Learning**
Use Case	Integrating
Actors	Admin
Triggers	Admin tries to run python code on web app.
Pre-conditions	Model must be trained.
Basic Flow	1. Admin creates a bridge using flask. 2. Admin runs the code and gets the local hosting link. 3. Admin runs the link on native browser.
Post Condition	A web app runs on the native browser.

3.5.4 Browsing Use Case

The following use case deals with the Searching functionality of our application and can be visualized in figure 3.5 and is explained in table 3.4.

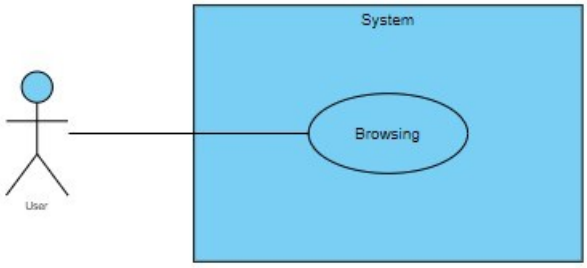

Figure 3.5: Browsing Use Case Diagram

Table 3.4: Browsing Use Case Description

System	**Detecting Specific Building Structure in Videos Using Deep Learning**
Use Case	Browsing
Actors	User
Triggers	User presses the browse button on web app.
Pre-conditions	Application should be integrated.
Basic Flow	1. 1. User opens the local hosting link on native browser. 2. User presses the browse button.
Post Condition	A directory is displayed using which user can upload a video from anywhere.

3.5.5 Searching Use Case

The following use case deals with the Searching functionality of our application and can be visualized in figure 3.6 and is explained in table 3.5.

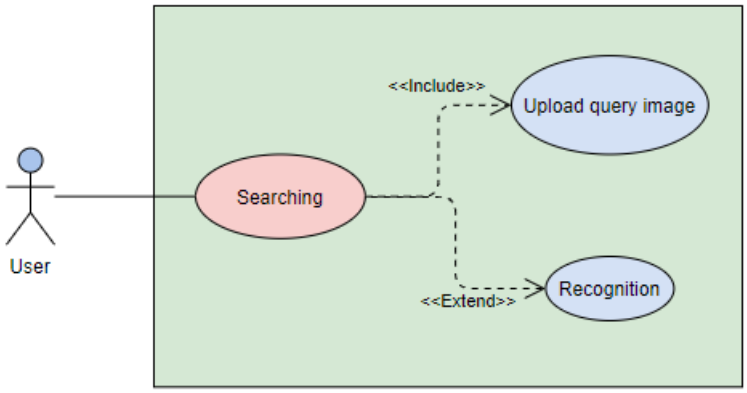

Figure 3.6: Searching Use Case Diagram

Table 3.5: Searching Use Case Description

System	**Detecting Specific Building Structure in Videos Using Deep Learning**
Use Case	Searching
Actors	User
Triggers	User inputs query video.
Pre-conditions	Model must be trained and front end must be integrated with the back-end.
Basic Flow	1. User uploads query video. 2. Model recognizes the object. 3. System generates a description of the searched object. 4. System writes the video on disk.
Post Condition	System displays the entire video.

Chapter 4

Design

4.1 System Architecture

Visual object recognition from videos has two main phases, back-end and front-end. Back-end is subdivided into two pieces. Training phase and testing phases. In training phase, model is given path of the folder containing XMLs of all images of all classes. TFRecord are generated and using the generated TFRecord the system starts training. A total of 2500 images are collected and labeled which are then divided into two parts in 70:30 ratio having 1750 training images and 750 test images. After training phase, testing phase comes, in testing phase test images are given to the system. Once the system verifies that the training is efficient enough to retrieve objects from videos, the model is then tested on videos.
User can interact with the application through user interface. GUI is quite flexible and easy to use. User can search objects from videos as well as user can train the system on new classes.

4.2 High Level Design

In high level design, there are major functionalities of this application i.e., actual flow of its working. It does not explain small details. The main modules of this system are input, processing, and output. In the input module, user enters query video. In processing step, query image is recognized, and if the label is recognized then bounding boxes are drawn and short description of the label object is generated. In output module, the user is shown the video containing the bounding box and the description of the object. Following is the figurative illustration of high-level design.

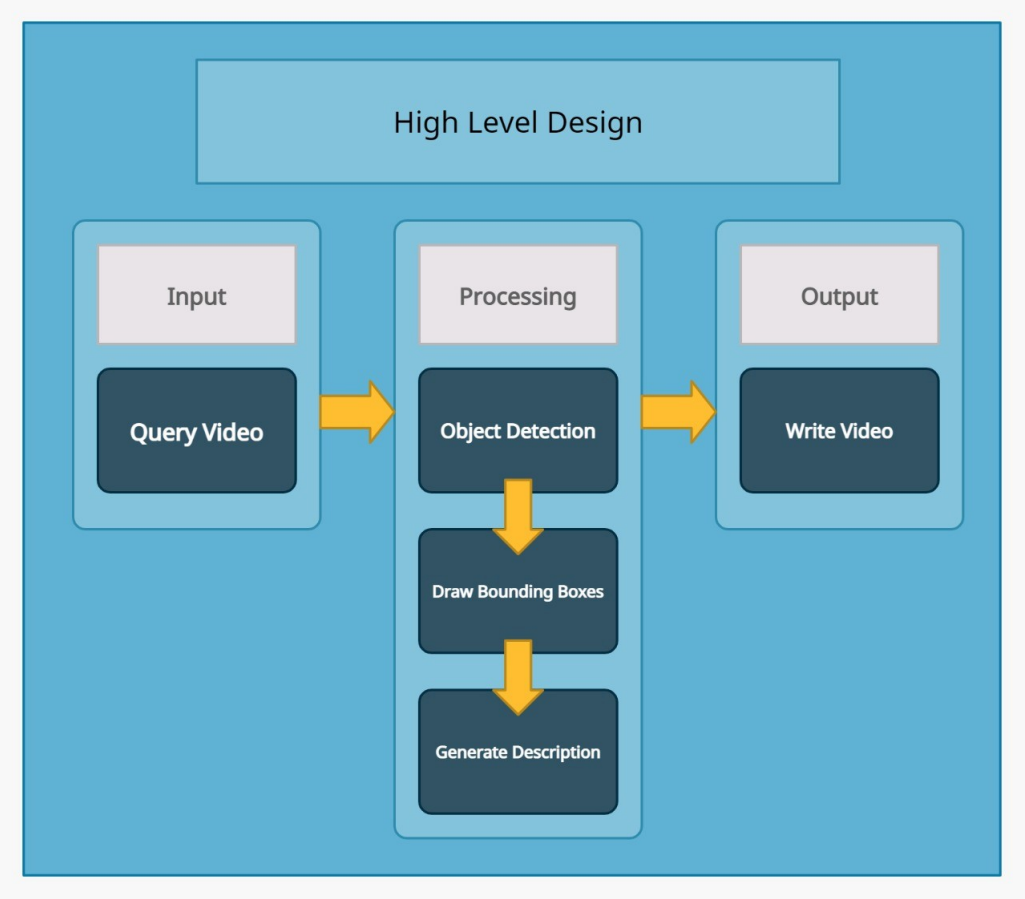

Figure 4.1: High Level Design

4.3 Low Level Design

Low level design of a product defines all the minute details of entire application, software or whatever it is. It follows a step-by-step fine-tuning process. This procedure can be utilized for designing data structures, essential software design and source code. Following are few diagrams to discuss the architecture in detail.

4.3.1 Flow Diagram

This diagram explains the methodology and components of this system in detail. There are two main phases i.e., training and searching.

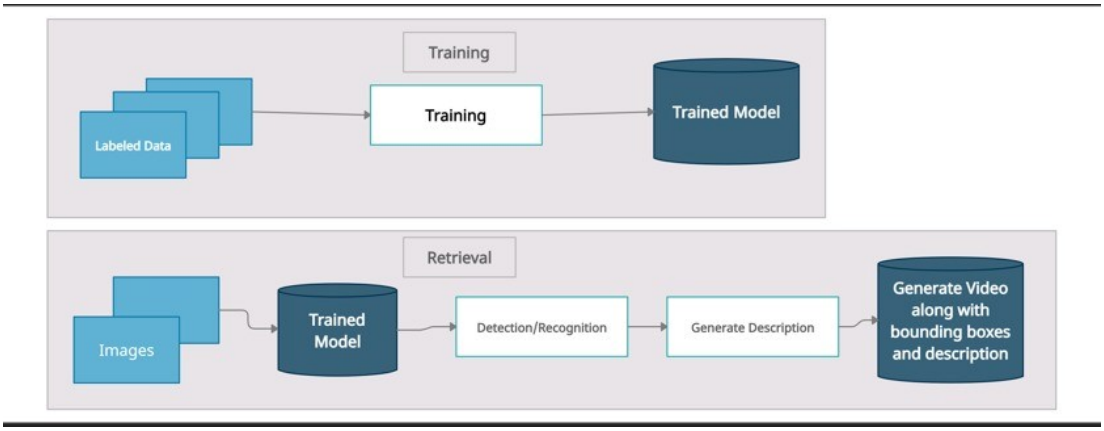

Figure 4.2: Flow Diagram

4.3.2 Entity Relationship Diagram

ERD is a detailed outline of the complete project architecture. In the ERD given below there are all main classes along with their main attributes and how they are interacting with each other.

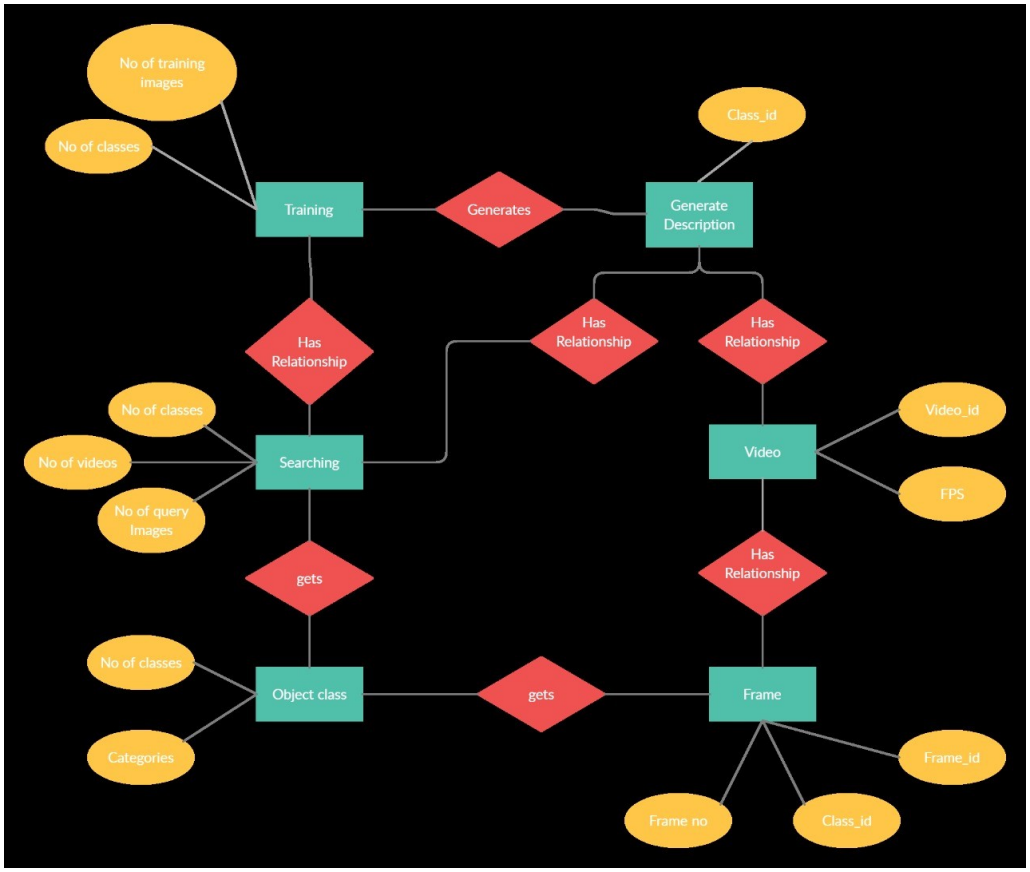

Figure 4.3: Entity Relationship Diagram

4.3.3 Class Diagram

Class diagram is one way to define the low-level design of this application. The figure given below is a class diagram of the project that depicts all minute attributes of each component that depicts how things are relating with each other.

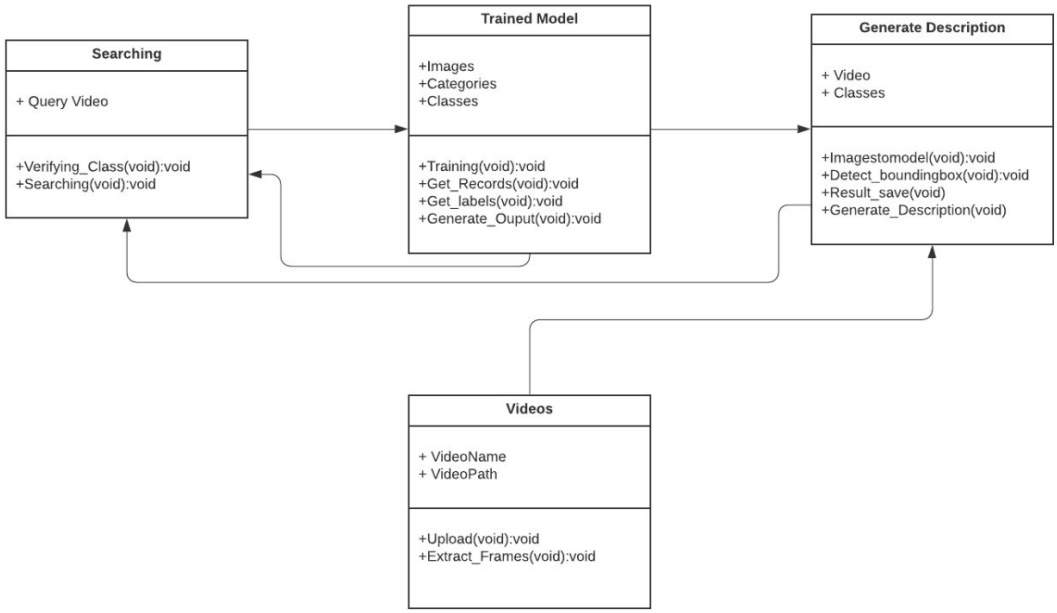

Figure 4.4: Class Diagram

4.3.4 Sequence Diagram

Sequence diagram is low level diagram, which characterizes correct stream of application as for time. It dictates association between various classes of framework and shows dynamic functioning of system.

4.3.4.1 LabelImg

Admin opens directory of images in labeling tool, one by one every image is opened, and a bounding box is drawn on the object and the user assigns each object a label, then saves the images in the form of xml. Labeling sequence diagram is given below.

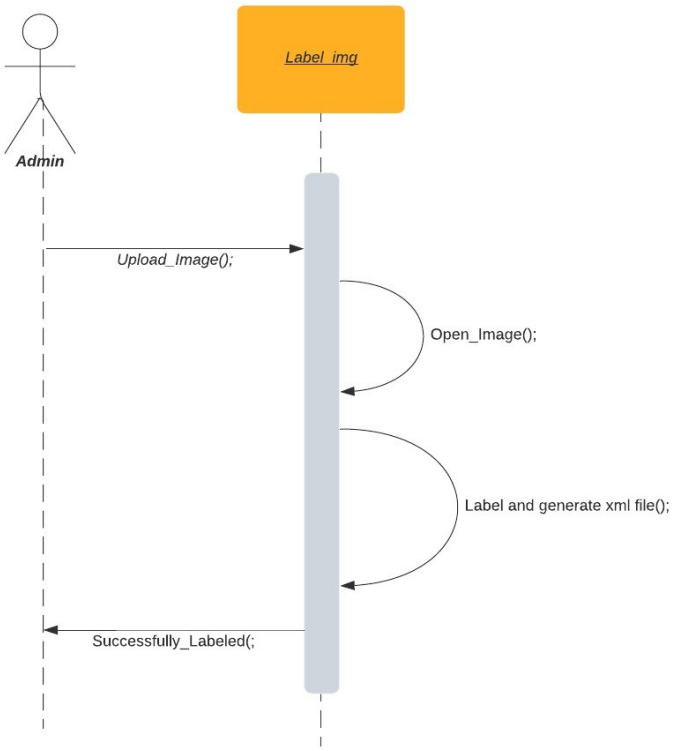

Figure 4.5: Labeling Sequence Diagram

4.3.4.2 Training

Next step is to upload images. Admin uploads images and system trains model on the basis of the uploaded labeled data. After successful training, a message is generated for the user. Training sequence diagram is given below.

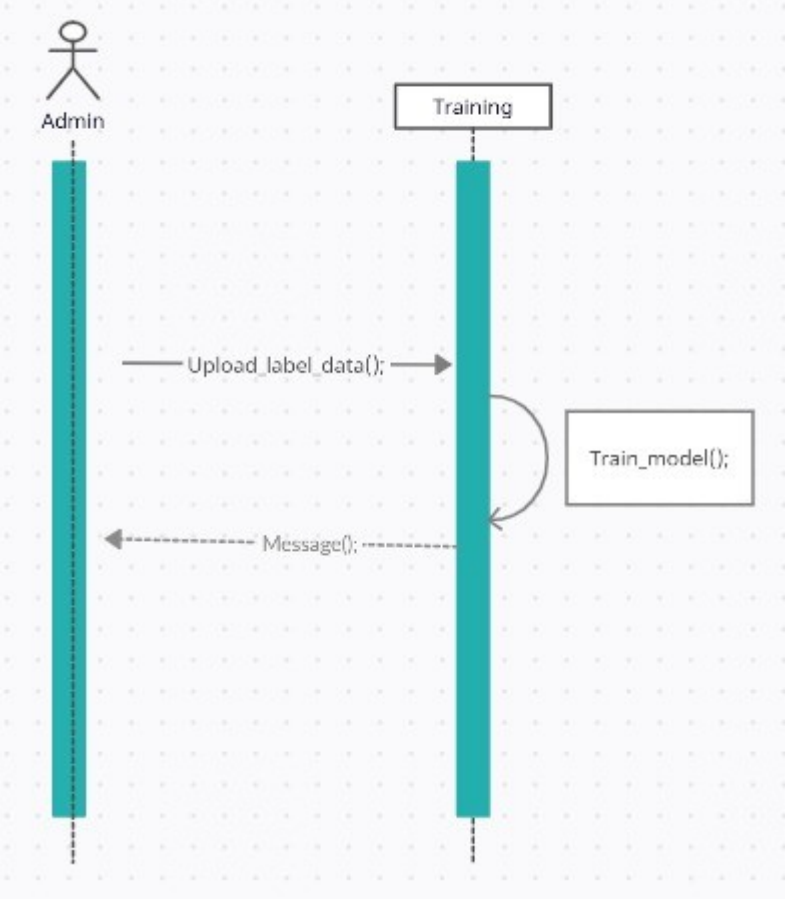

Figure 4.6: Training Sequence Diagram

4.3.4.3 Searching

User inputs a query video. System recognizes that object in the image with the help of trained model and returns the label of that object. System draws bounding boxes on the detected object.

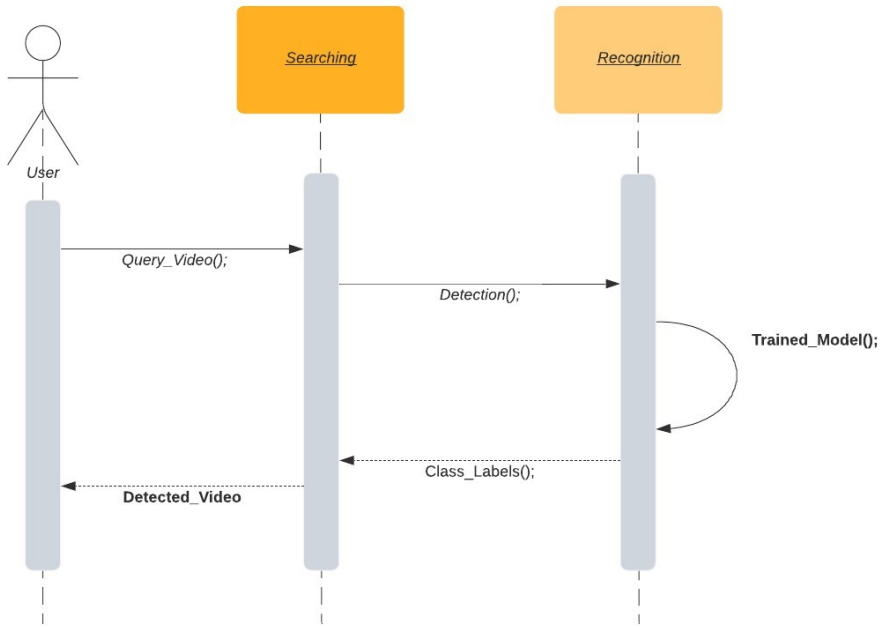

Figure 4.7: Searching Sequence Diagram

4.3.4.4 Generating Description

The label of the detected object is then sent to the system. The system according to the label writes a short description of the detected object on the detected frame. The newly written video is then shown to the user.

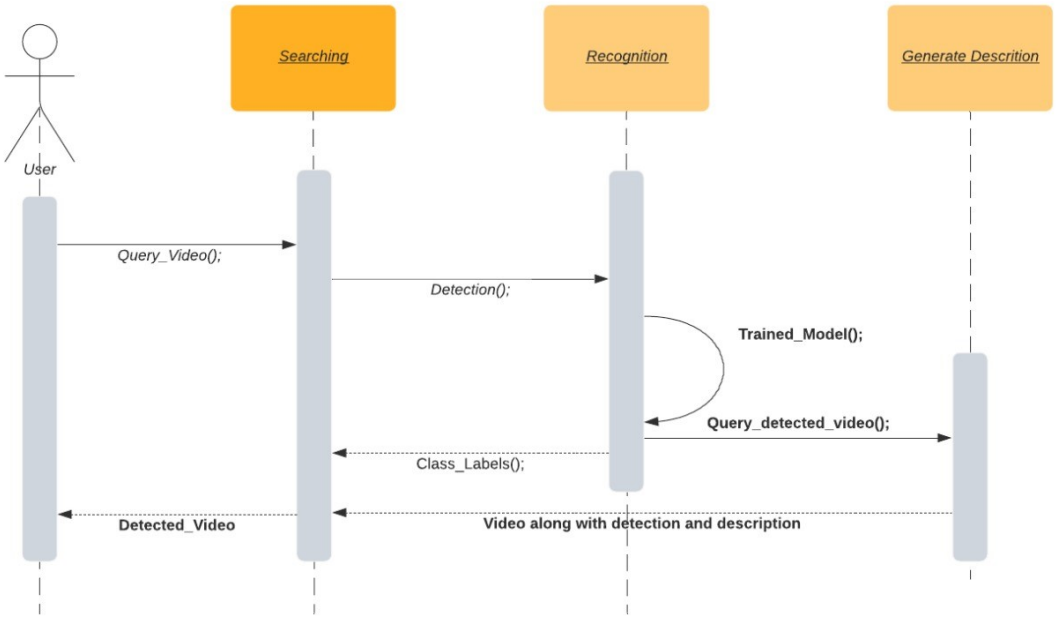

Figure 4.8: Generating Description Sequence Diagram

Chapter 5

Implementation

5.1 System Implementation

This chapter presents the detailed implementation the desktop application named 'Detecting Specific Building Structure in Videos Using Deep Learning' together with the tools and technologies used and the algorithmic approaches used in the system.

5.2 System Overview

This chapter presents the detailed implementation the desktop application named 'Detecting Specific Building Structure in Videos Using Deep Learning' together with the tools and technologies used and the algorithmic approaches used in the system.

5.2.1 Tools and Technology Used

The tools used in our system are as following:

5.2.1.1 Visual Studio Code

Visual Studio Code is an editor for source code. It was developed by Microsoft for Windows, Linux and macOS. It supports syntax highlighting, debugging, and code refactoring. It is also used for customizing editor's theme, keyboard shortcuts, and preferences. Visual Studio Code was used in this project for running commands of training model.

5.2.1.2 Anaconda

Anaconda is an open source distribution for the python language and R programming language which are used for data science and machine learning related applications. It targets to streamline package management and deployment. Anaconda is used to run python language because the back-end of this application is written in Python. Anaconda command prompt is used to run the opening command of LabelImg tool.

5.2.1.3 LabelImg Tool

LabelImg is a graphical image labeling tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. All the data set images (training images) were labeled using this tool.

5.2.1.4 PyQt

PyQt is an open source Python binding of cross-platform GUI toolkit Qt. It is a free software developed by a British company. PyQt implements nearly 440 classes and more than 6,000 methods and functions. PyQ14 is used for the graphical interface of the tool LabelImg.

5.2.2 Development Environment/Language Used

The development environment of our system is as following:

5.2.2.1 Python 3.9

Python is a high level coding language for multipurpose programming. Python language is mostly used language for data sciences and deep learning because the deep learning models are written in python language. Therefore the back-end is written in python.

5.2.2.2 Tensor Flow

Tensor Flow is an open-source library presented by Google. It is primarily focusing data flow programming in a range of tasks. It is a figurative math library which is mostly used in machine learning based applications including neural networks.

5.2.2.3 Open CV

EmguCV is an open-source library developed by Intel. It is cross platform and free to use. Open CV is mainly aimed at real time computer vision and graphics. This library was integrated in our system for video manipulation especially for frame extraction.

5.2.2.4 Flask

Flask is an open-source micro web framework written in python. It does not require any particular tools or libraries. Flask depends on the jinja template. Flask is mainly used to provide a bridge between back-end python and front-end web application languages (HTML, CSS, bootstrap).

5.2.2.5 HTML

The Hypertext Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

5.2.2.6 CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

5.2.2.7 Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

5.3 Methodology

Through a system flow diagram. System's methodology has been explained as shown in the figure 4.2. This diagram is all about the steps how this application actually works. There are two phases of this project i.e. training and searching. In the first phase, labeled data is uploaded to train the model. System itself trains the model on the given labeled data. After the training is complete, searching phase comes. In the searching phase. Trained model is applied on the videos. System extracts frames from videos, and model recognizes

objects in them. System then shows the entire video and if an object is there it creates a bounding box over that object and shows the name of that object.

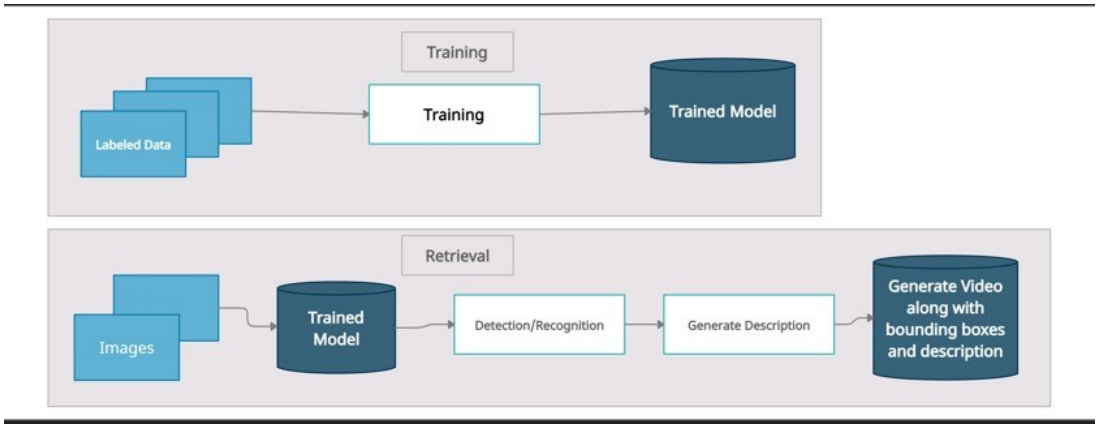

Figure 5.1: Methodology

Following is the detailed methodology of this application named 'Detecting Specific Building Structure in Videos Using Deep Learning'. This application has three steps of development; training, indexing and searching.

5.3.1 Training

Training phase has several sub steps which are described below.

5.3.1.1 Ground Truth Labeling

There is less data of our classes (buildings of Pakistan) on internet therefore we need enormous labeled data. We did ground truth labeling manually using LabelImg tool. Using this tool all training images were labeled. All buildings were outlined and labeled by a name. On saving the image is saved as xml containing x, y coordinates, width and height of bounding box, label name and image name. This XML will be used for detection purpose.

5.3.1.2 TFRecord Generation

For storing a sequence of binary records TFRecord is used. This is the final file that is given to the model for training. The TFRecord file contains each and every information about the whole data set in a structure of binary strings. Training of model starts when the path of TFRecord is provided to the mode. Model learns the objects on the basis of their distinct features like pattern, colors and pixel based features.

5.3.1.3 Object detection

The SSD framework is used for object detection. Its approach is centered on a feed-forward CNN that yields a static size pool of bounding boxes and tallies for the existence of object in those bounding boxes.
The initial network layer is a standard architecture used as high quality image classification. A convolution feature layer shrink in size gradually and allow estimates of detections at numerous scales. The offset output values of bounding box are calculated comparative to a default box position qualified to each feature map location.

5.3.1.4 Object recognition

Once the object is detected i.e. exactly at what position an object is lying the SSD MobileNet model recognizes the object and classifies it by giving it its class label. SSD MobileNet model matches the object with the features of all classes and generates a probability of all classes and then displays the highest probability.

5.3.1.5 Searching

User queries a video. Videos are sequence of frames. Each frame has an image. All the frames are sent to model where each frame is manipulated by model, objects are detected and then recognized. Bounding box is created on the detected object and their labels are shown. Then the whole video is played displaying the bounding box on the detected object.

5.3.1.6 Integration of Python and Flask (micro web framework)

The back-end of our application is developed in python while the front-end is developed using HTML, CSS and Bootstrap. In order to integrate them we used Flask. Flask basically provides you with the tools and technologies to build a web application. Flask is an API of python which allows us to build web based application. Flask easily connects the Python code with HTML front-end.

5.4 Algorithmic Development

Following is the algorithmic approach used for the development of this application.

5.4.1 CNN

CNN: Convolutional neural network is the existing state-of-the-art exemplary architecture for object classification activities. CNN apply a sequence of filters to the pixels of an

image in order to extract and to learn high level features, which can be used by the model for classification purpose. CNN covers three components i.e. convolution layer, pooling layer and dense.

Usually CNN is made up of a pile of convolutional units that extract features. Each unit has a convolutional layer which is then followed be a pooling layer.

5.4.2 SSD

SSD: Single Shot multi-box Detector is for object detection in real-time. Faster R-CNN practices a section proposal network to generate boundary boxes and it uses those boundary boxes to categorize various objects. Whereas it is a milestone in accuracy, the entire procedure runs at seven fps (frames per second) which is a requirement of real time processing. SSD hurries the procedure by removing the requirement of the region proposal network. To improve the fall in accuracy the SSD put on a little enhancements counting multi features and default boxes. SSD achieves the real-time processing speed and in fact knocks down the accuracy of the Faster R-CNN.

The SSD object detection has two main parts i.e. extraction of feature map and then application of filters in order to detect objects. For extraction of feature maps, SSD uses VGG16 and then detects the objects using a convolution layer. It is used in our project to detect object from videos. This algorithm draws a bounding box on detected building.

5.4.3 SSD MobileNet V2 FPNLite 320x320

It is a classifier used to classify object classes. The training images are scaled to 320x320. Pooling is vital for the accomplishment of current CNNS, the inception module included a supplementary pooling path. The outputs of all filters are first concatenated and then passed on to the next layer as input. This architecture is used to classify objects efficiently. Following is the architecture of this model in figure 2.2. In our project this model is used for classification of building.

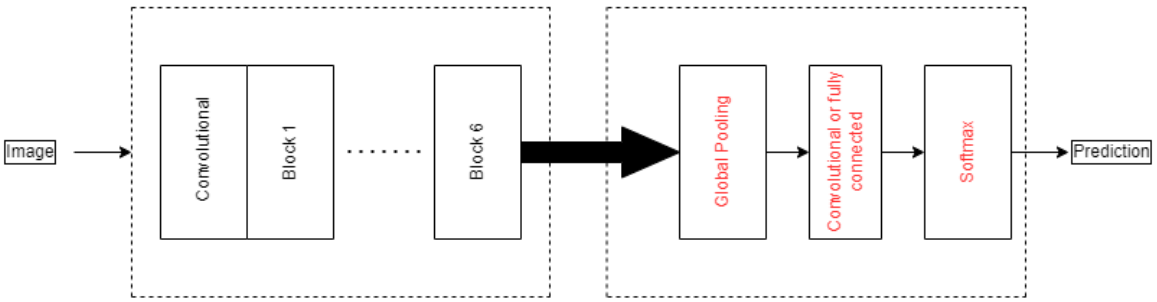

Figure 5.2: Model Architecture

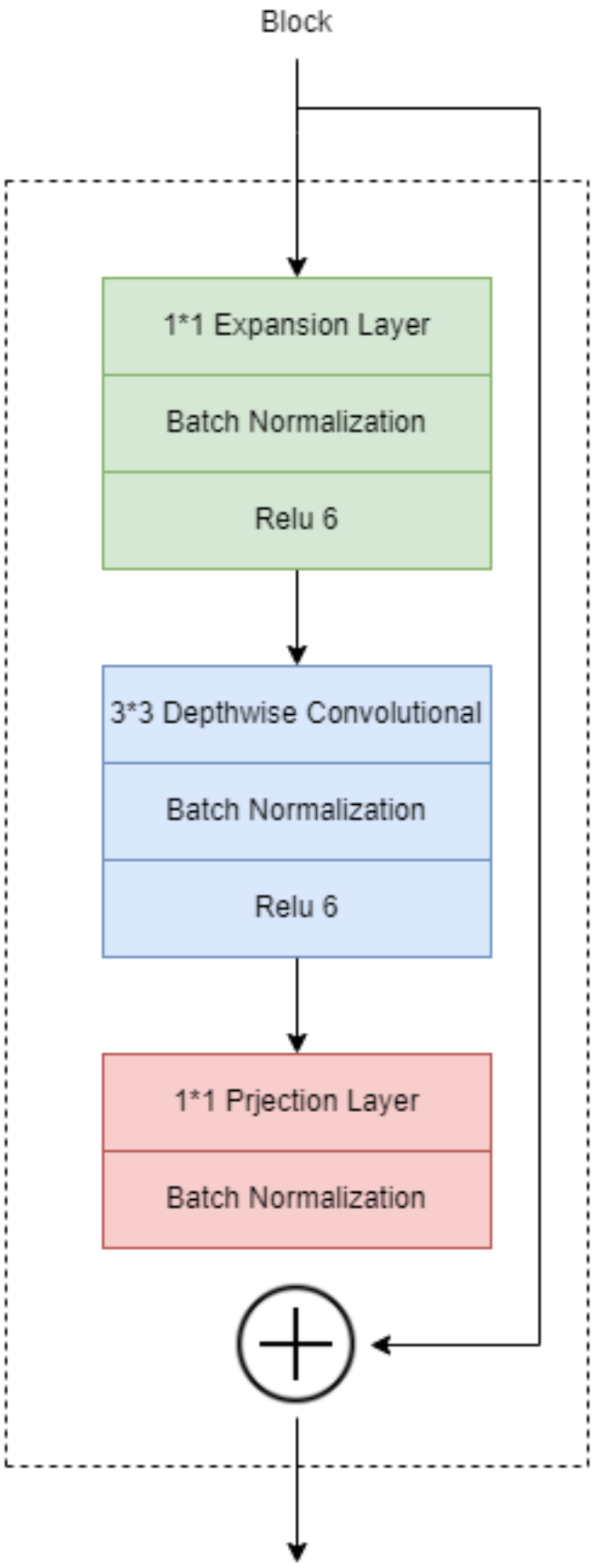

Figure 5.3: Block Architecture

Chapter 6

Testing

6.1 System Testing and Evaluation

This chapter will explain the performance of each and every module of the project 'Detecting Specific Building Structure in Videos Using Deep Learning'. Testing is a very important phase of software development life cycle because this phase enhance the quality of the system. System requirements are validated by this phase. Moreover exceptions are handled in this system. All and minor functionalities of this system are tested to check if there is any error and so that the error may be removed before final deployment.

6.2 Selecting Testing Methodology

For the testing of our application we have followed the Test-case based testing method. Each test case that is mentioned below is thoroughly checked by the developers and supervisor.

6.3 Software Testing Technique

For the testing of our application we have followed the Test-case based testing method. Each test case that is mentioned below is thoroughly checked by the developers and supervisor.

6.3.1 White Box Testing

White box testing is established on internal philosophy of an applications program. For this software and program functioning should be well-known for operating this type of

testing. With the help of this we performed tests of our application based on the coverage of code statements, predicates, branch and statement coverage.

6.3.2 Black Box Testing

Testing technique in which the internal process of the system are ignored and main target is to focus on the outcomes that are generated with respect to any input and systems execution.

6.3.3 Graphical User Interface (GUI) Testing

This 'Detecting Specific Building Structure in Videos Using Deep Learning' has a user friendly front-end for the ease of user. GUI is made user friendly so that every user of any intellect may interact with the system without any problem. All buttons windows, labels etc. are for specific purpose with proper name for easy understanding. The GUI is designed in such a way that user may easily understand what his next step should be.

6.3.4 Unit Testing

This is the testing technique in which the main target is to test the functioning of the separate units of the system. Our application contained different units so we applied unit testing as each unit was tested separately and after the correct observation of results of each they were integrated progressively.

6.3.5 Performance Testing

This is the testing technique in which the main target is to test the functioning of the separate units of the system. Our application contained different units so we applied unit testing as each unit was tested separately and after the correct observation of results of each they were integrated progressively.

6.3.6 Acceptance Testing

This is the testing technique in which the main target is to test the functioning of the separate units of the system. Our application contained different units so we applied unit testing as each unit was tested separately and after the correct observation of results of each they were integrated progressively.

6.3.7 Regression Testing

This is the testing technique in which the main target is to test the functioning of the separate units of the system. Our application contained different units so we applied unit testing as each unit was tested separately and after the correct observation of results of each they were integrated progressively.

6.4 Compatibility Testing

As mentioned earlier that GPU based Tensor Flow is used for training of this system. Tensor Flow has two versions one is CPU based and the other one is GPU based version of Tensor Flow, GPU version of Tensor Flow is more efficient than the CPU version Tensor Flow. But on deployment often GPU version systems generates error on compatibility with the CPU systems but this GPU version of Tensor Flow is compatible with both GPU and CPU.

6.5 System specification

As mentioned earlier that GPU based Tensor Flow is used for training of this system. Tensor Flow has two versions one is CPU based and the other one is GPU based version of Tensor Flow, GPU version of Tensor Flow is more efficient than the CPU version Tensor Flow. But on deployment often GPU version systems generates error on compatibility with the CPU systems but this GPU version of Tensor Flow is compatible with both GPU and CPU.

- 64-bit Operating system
- RAM 16GB
- GPU 4GB NVIDIA GEFORC GTX 1050 Ti
- Processor 2.80 GHz and 2.81 GHz
- Python 3.9
- Tensor Flow 2.7.0

6.6 Test Cases

6.6.1 SDD Processing (Detection)

The following test case deals with the detection functionality of the application, the input condition and the result is detailed in the table below.

Table 6.1: SSD Processing Test Case

Test Case ID	TC-01
Test Case Name	SSD processing.
Test Case Requirement	GPU based Computer.
Test Execution	1. SSD detects objects in frames that on which location the object is lying. 2. A bounding box is generated around the object.
Expected Result	Object should be correctly detected in a frame.
Actual Result	Object detection is successful and bounding box is created on correct location
Status	Pass.

6.6.2 SSD MobileNet V2 Processing (Recognition)

The following test case deals with the correct labeling functionality of the application, the input condition and the result is detailed in the table below.

Table 6.2: SSD MobNet V2 Test Case

Test Case ID	TC-02
Test Case Name	SSD MobileNet V2 Processing
Test Case Requirement	GPU based Computer.
Test Execution	1. Detected object in the bounding box is recognized by the mobile net classifier. 2. Object is given label.
Expected Result	Object must be given an accurate label along with confidence score.
Actual Result	Object recognition is successful. Label is given to the object along with confidence score.
Status	Pass.

6.6.3 Image Testing

The following test case deals with the correct label detection within an image, the final product does not provide this functionality, the input condition and the result is detailed in the table below.

Table 6.3: Image Test Case

Test Case ID	TC-03
Test Case Name	Detection in an image
Test Case Requirement	GPU based Computer.
Test Execution	1. User uploads image
Expected Result	Object in image must be successfully detected.
Actual Result	Object in image is successfully detected.
Status	Pass.

6.6.4 Video Testing

The following test case deals with the correct label detection within a video, the final product does not provide this functionality, the input condition and the result is detailed in the table below.

Table 6.4: Video Test Case

Test Case ID	TC-04
Test Case Name	Video Testing
Test Case Requirement	GPU based Computer.
Test Execution	1. User uploads a video
Expected Result	Object in the video must be successfully detected.
Actual Result	Object in the video is successfully detected.
Status	Pass.

6.6.5 Integration with C# Testing

The following test case checks whether the python back-end is integrated properly with the C front-end or not the input condition and the result is detailed in the table below.

Table 6.5: Integration with C# Test Case

Test Case ID	TC-05
Test Case Name	Integration With C# Testing
Test Case Requirement	GPU based Computer.
Test Execution	1. Admin runs the python script through C# code.
Expected Result	Python Script runs successfully.
Actual Result	Python Script does not run successfully.
Status	Fail.

6.6.6 Integration with flask Testing

The following test case checks whether the python back-end is integrated properly with the flask front-end or not the input condition and the result is detailed in the table below.

Table 6.6: Integration with Flask Test Case

Test Case ID	TC-06
Test Case Name	Integration with flask Testing
Test Case Requirement	GPU based Computer.
Test Execution	1. Admin runs the python script.
Expected Result	A local hosting link is generated and the python script is integrated successfully.
Actual Result	The link is generated and the python script ran successfully.
Status	Pass.

6.6.7 Upload Video in Application

The following test case checks whether the video is correctly uploaded on the web application or not input condition and the result is detailed in the table below.

Table 6.7: Upload Video Test Case

Test Case ID	TC-07
Test Case Name	Upload Video.
Test Case Requirement	GPU based Computer.
Test Execution	1. User browse the upload button. 2. User selects video.
Expected Result	Video must be uploaded successfully successfully.
Actual Result	Video is uploaded successfully.
Status	Pass.

6.6.8 Video Testing in Application

The following test case checks whether the object in video is correctly detected or not input condition and the result is detailed in the table below.

Table 6.8: Video Test Case in Application

Test Case ID	TC-08
Test Case Name	Video Testing in Application.
Test Case Requirement	CPU based.
Test Execution	Media player displays the video and when the object is in frame it should put it in a bounding box.
Expected Result	Detected object must be displayed in video.
Actual Result	Detected object is displayed in video.
Status	Pass.

6.6.9 No object in Video Testing in Application

In the following test case a video is provided which does not contain any object, the test case checks what will the application do in such condition, input condition and the result is detailed in the table below.

Table 6.9: No object in video testing in Application Test Case

Test Case ID	TC-09
Test Case Name	No Object in video.
Test Case Requirement	CPU based.
Test Execution	1. User browse the upload button. 2. User selects video.
Expected Result	No object should be Detected and a message is displayed that no object was detected.
Actual Result	No object is detected and the message is displayed.
Status	Pass.

6.6.10 Again Uploading Video

In the following test case a video is provided again, the test case checks what will the application do in such condition, input condition and the result is detailed in the table below.

Table 6.10: Again Uploading Video Test Case

Test Case ID	TC-10
Test Case Name	Again Uploading Video.
Test Case Requirement	GPU based.
Test Execution	1. User browse the upload button. 2. User selects video.
Expected Result	New Video should be detected and displayed.
Actual Result	New video is detected and displayed.
Status	Pass.

6.7 Integration testing

It is already discussed multiple times that this system is developed in multiple languages. Front end is developed in HTML, CSS and bootstrap while the back end is developed in Python and therefore we need to integrate both of them to have a proper presentation of the products. For integration of both frameworks we have used flask library. Flask basically bridges the python back-end with HTML/CSS to create a working web application. Now we have to validate the results by testing this integration. Initially, we integrated a

simple python script with web application through Flask Bridge which ran successfully. Afterwards, we ran our actual python script and validated if the detection is taking place in the web application. After thorough stress test and repeated initialization we concluded that the languages integrated successfully.

6.8 Bug report

Integrating the python code with C# yielded undesirable results. A normal python script ran successfully from C# code but when the actual python script (which contained all the back-end code) ran on the C# code, there was an error. After thorough deduction we yielded that the C# code was not able to upload the trained model successfully. After careful analysis we came to the conclusion that we should not create our front-end on C# but rather create a web application and bridge the front-end and back-end through flask.

6.9 Evaluation Metric

After training we also ran evaluation on our test frames. The results having Precision and Recall of the entire trained model is displayed in the table given below.

Table 6.11: Evaluation Metric

Precision	Recall
93.75	89.05

Chapter 7

Conclusions

7.1 Conclusions

Comprehension of object retrieval from videos is an immense issue however it elucidates many other issues like searching of objects from videos or images, text search from videos etc. to manually retrieve objects from videos or to gather required data from a video is quite a time taking activity and it takes a lot of peoples to sit and watch videos. With the advancement in technology, the idea of object detection came and it helped a lot in searching content from videos.

In order to cater this problem, we have proposed this system which retrieves objects from videos. It is much easy to detect and recognize objects from images but in videos we have other factors like noise, audio, low resolution etc. thus it become complex to detect object in videos. Ignoring all these secondary issues we focused on the visual content only. We have used image processing and deep learning technique to ease the finding of specific object from videos.

We have used deep convolutional neural network to detect objects from videos and then to recognize them that which class they detected object belongs. In the pre-processing phase we collected data set, resized the images to a standard size and labeled them. In post-processing phase the training started in data set. We trained object detection Tensor Flow model to detect objects in videos. SSD is used for object detection and Mobile-Net-v2 has been used for classification of object. We have trained this system on 6 classes of buildings (6 renowned buildings of Pakistan). User can input image and can search through text also. In output the user gets all videos containing that specific object. This is a general framework and this can be trained for other buildings or objects. This type of application is a time effective and cost effective solution for regulatory bodies.

7.2 Future Work

Since this system has been trained on 6 classes and these classes are only buildings. This can be further extended by adding more classes. Even different objects can be added in classes e.g. military equipment and other objects like flowers, cars etc. we have ignored sounds in videos, they can be processed to retrieve information from auditory data. Moreover text based information can be retrieved. In this way a complete search engine can be developed.

Bibliography

[1] L. Jiao, R. Zhang, F. Liu, S. Yang, B. Hou, L. Li, and X. Tang, "New generation deep learning for video object detection: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2021. Cited on p. 1.

[2] S. Kaarmukilan, S. Poddar *et al.*, "Fpga based deep learning models for object detection and recognition comparison of object detection comparison of object detection models using fpga," in *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2020, pp. 471–474. Cited on p. 8.

[3] M. Mazumdar, V. Sarasvathi, and A. Kumar, "Object recognition in videos by sequential frame extraction using convolutional neural networks and fully connected neural networks," in *2017 International conference on energy, communication, data analytics and soft computing (ICECDS)*. IEEE, 2017, pp. 1485–1488. Cited on p. 9.

[4] Y. Zhao, J. Zhao, C. Zhao, W. Xiong, Q. Li, and J. Yang, "Robust real-time object detection based on deep learning for very high resolution remote sensing images," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2019, pp. 1314–1317. Cited on p. 10.

[5] Z.-Q. Zhao, P. Zheng, S.-t. 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, 2019. Cited on p. 11.