VIOLENCE DETECTION SYSTEM

A Project Work-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

BY

Shyam Tawli-EN19IT301083
Ved Dubey-EN19IT301100
Yuvraj Kocher-EN19IT301106

Under the Guidance of

Ms. Jyoti Kukade

Department of Information Technology
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331

Jan-June 2023

Report Approval

The project work **“Violence Detection System”** is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner
Name:
Designation
Affiliation

External Examiner
Name:
Designation
Affiliation

Declaration

We hereby declare that the project entitled **"Violence Detection System"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Information Technology' completed under the supervision of **Ms. Jyoti Kukade, Assistant Professor Department of Information Technology,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Signature and name of the student(s) with date

Certificate

I, **Ms. Jyoti Kukade** certify that the project entitled **"Violence Detection System"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Shyam Tawli, Ved Dubey and Yuvraj Kocher** is the record carried out by him/them under my/our guidance and that the work has not formed the basis of award of any other degree elsewhere.

Ms. Jyoti Kukade

Assistant Professor

Information Technology

Medi-Caps University, Indore

Dr. Prashant Panse

Head of the Department

Information Technology

Medi-Caps University, Indore

<u>Acknowledgements</u>

We would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Dilip K Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. We also thank **Prof. (Dr.) D K Panda,** Pro-Vice Chancellor, Medi-Caps University, **Dr. Pramod S. Nair,** Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Prashant Panse** for his continuous encouragement for betterment of the project.

We express my heartfelt gratitude to my Internal Guide, Ms. Jyoti Kukade, Department of Information Technology, MU, without whose continuous help and support, this project would ever have reached to the completion.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

Shyam Tawli
Ved Dubey
Yuvraj Kocher
B.Tech. IV Year
Department of Information Technology
Faculty of Engineering
Medi-Caps University, Indore

Abstract

As the years passed by, computers became more powerful, and automation became the need of generation. Humans tried to automate their work and replace themselves with machines. This effort of transition from manual to automatic gave rise to various research fields, computer vision is one such field. Violence detection is an essential task for ensuring public safety in various domains, such as video surveillance, social media monitoring, and crowd management. Detecting violent events accurately and efficiently requires the use of advanced technologies, such as deep learning, which can analyze complex patterns in large datasets. This documentation presents a comprehensive review of the state-of-the-art methods for violence detection using deep learning, including their architectures, training strategies, and evaluation metrics.

The documentation starts by discussing the main challenges of violence detection, such as the diversity of violent events, the presence of contextual cues, and the class imbalance problem. Then, it presents the main deep learning architectures used for this task, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their combinations. The documentation describes the design choices of these architectures, such as the number of layers, the activation functions. Moreover, it discusses the training strategies used to optimize these architectures, such as transfer learning,

The paper also presents the main evaluation metrics used for violence detection, such as accuracy, precision, recall, and F1 score.

In conclusion, violence detection using deep learning is a challenging and essential task that requires the integration of advanced techniques in computer vision and machine learning. The paper provides a comprehensive review of the state-of-the-art methods and datasets for this task and identifies the open challenges and directions for future research.

Keywords: Computer Vision , Deep Learning, Violence Detection, LSTM, ANN, Action Recognition

Table of Contents

List of Figures

Chapter-1

Introduction

1.1 Introduction

In this project we present Violence Detection based on Deep learning techniques. Violence Detection plays an important role and is currently getting the attention of researchers because the problem of human action recognition at a distance has become tractable by using computer vision techniques. The recent development in the field of human action recognition has led to a renewed interest in the detection of specific action, one of them is violence. Automatic violence detection is basically involved to increase the safety of humans by monitoring their behavior, in addition to prevent social, economic and environmental damages.

Violence detection has become an increasingly important issue in ensuring public safety, particularly in domains such as security, law enforcement, and social media monitoring. Traditionally, detecting violent events has relied on manual annotation and rule-based systems. However, these methods have limited efficacy in handling the complexity and diversity of violent events. Deep learning, on the other hand, has proven to be a promising approach to violence detection. By utilizing large datasets and advanced algorithms, deep learning can analyze complex patterns and learn to detect violent events accurately and efficiently. In this paper, we provide an overview of the state-of-the-art methods and challenges of violence detection using deep learning.

1.2 Objectives

Deep learning models can achieve several objectives in violence detection, including:

1. Accurate detection of violent events: Accurate and efficient detection: Deep learning algorithms can analyze and learn complex patterns and features from datasets, enabling them to accurately and efficiently detect violent events in real-time, outperforming traditional rule-based systems.

2. Real-time detection: Deep learning models can process video and audio streams in real-time, enabling the detection of violent events as they occur.

3. Generalization: Deep learning models can generalize to new domains and contexts, making them adaptable to different scenarios and environments.

4. Contextual awareness: Deep learning models can capture contextual cues, such as body language and environmental factors, which are crucial for detecting violent events.

5. Automated annotation: Deep learning models can automatically annotate violent events, reducing the need for manual annotation and saving time and resources.

Overall, deep learning can enhance the accuracy, speed, and scalability of violence detection, enabling better public safety and security in various domains.

1.3 Significance/Scope

Violence detection can be applied in various domains and applications, where public safety and security are critical concerns. Some of the use-case are listed below:

1. **Security and Surveillance**: Violence detection can be used to monitor and identify violent behavior in public places, such as airports, train stations, and shopping malls, where security cameras are already in place.

2. **Law Enforcement**: Violence detection can help law enforcement agencies detect violent behavior in real-time, enabling them to respond quickly and prevent escalation of violent incidents.

3. **Social Media Monitoring**: Violence detection can be used to monitor social media platforms for violent content and automatically flag post, which can be harmful to individuals and communities.

1.4 Source of Data/ Problem in existing system + Justification

We sourced dataset titled "**Real Life Violence Situations Dataset"** from Kaggle.

Link: https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset

Dataset Contains 1000 **Violence** and 1000 **Non-Violence** videos collected from YouTube videos. Violence videos comprises mostly street fight, sports fight situations in several environments and conditions. Non-violence videos in the dataset contains videos of different human actions like sports, eating, walking etc. The length of videos varies from 2 seconds to 7 seconds.

Problem in Existing Systems that used the same dataset are as following:

1. Most of the system used single frame model, which might be good only for classifying simpler task, like playing guitar, running etc.
2. Accuracy in most systems were not good enough <75%. The Technique we used to train our model achieved 85% accuracy.
3. Most algorithms implemented earlier required huge computation resources for training models. We used pre-trained model to extract features that reduced computation time and required computation resources for training models.
4. Earlier systems required huge computation time and resources, but since we used pre-trained model for feature extraction, we reduced the computation time and resources required by a large extent.

Chapter-2
System Requirement Analysis

2.1 Platform Specification (Development/Deployment)

2.1.1 Software implementation language/ Technology

Software requirements deal with defining software resource requirements and pre requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre requisites are generally not included in the software installation package and need to be install separately before the software is installed. Following software's are required to run the project.

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects. Python is dynamically-typed and garbage collected.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend. Flask is a web application framework written in Python. Flask is based on the **WSGI** toolkit and the **Jinja2** template engine. Unlike the

Django framework, Flask is very Pythonic. It’s easy to get started with Flask, because it doesn’t have a huge learning curve. It’s a microframework,
The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications.
Jinja2 is a popular template engine for Python. A web template system combines a template with a specific data source to render a dynamic web page.

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library[for building user interfaces based on components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies.
React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js. Because React is only concerned with the user interface and rendering components to the DOM, React applications often rely on libraries for routing and other client-side functionality

Web browser is application software for accessing websites. When a user requests a web page from a particular website, the browser retrieves its files from a web server and then displays the page on the user's screen. Browsers are used on a range of devices, including desktops, laptops, tablets, and smartphones.

Chapter-3
System Analysis

3.1 Information flow Representation

3.1.1 Use Cases Diagram

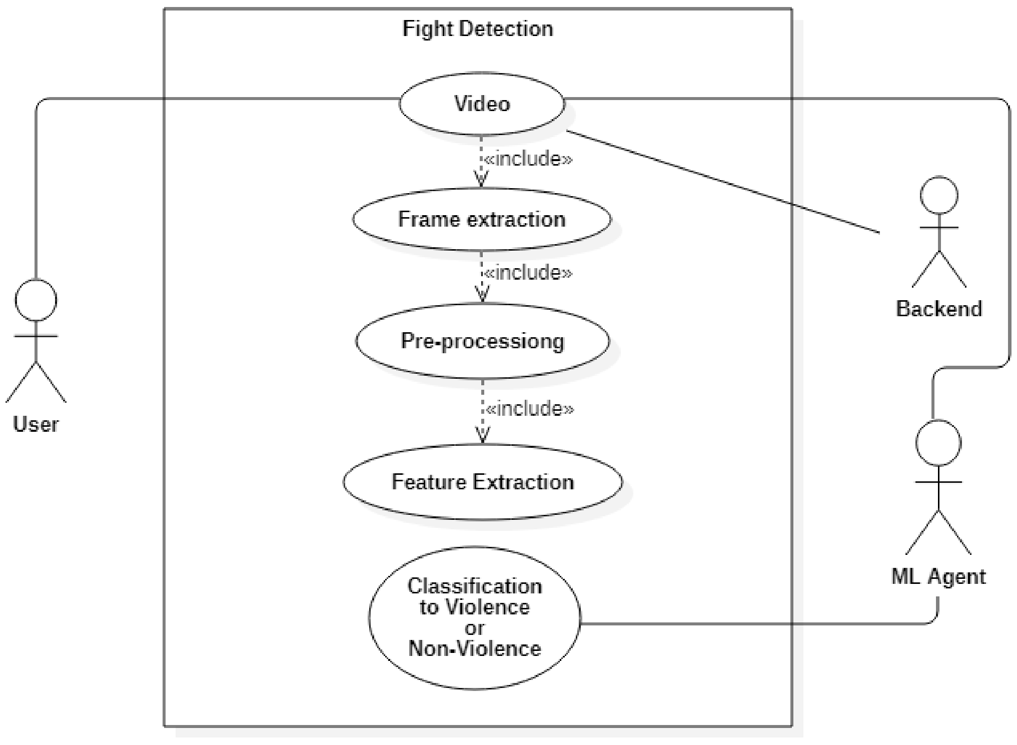


Fig 3.1 Use Case Diagram

1. User gives input the video through form using web browser GUI.
2. Frames are extracted from video.
3. Preprocessed frame input vector then passed to pre-trained model (Inception V3) for feature extraction.
4. Input feature for model is created.
5. Input feature passed to model.
6. Prediction result.
7. Prediction is shown as output on GUI.

3.1.3 Activity Diagram

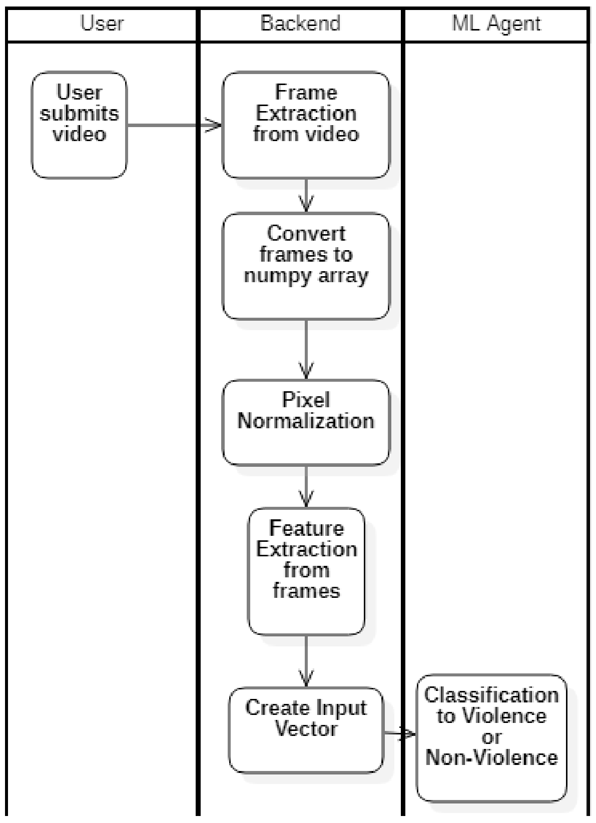

User
Backend
ML Agent
User submits video
Frame Extraction from video
Convert frames to numpy array
Pixel Normalization
Feature Extraction from frames
Create Input Vector
Classification to Violence or Non-Violence

Fig 3.2 Swimlane Activity Diagram

1. User will input the video using web application.
2. Frames will be extracted and cropped from video.
3. The frame will be converted into numpy array.
4. Normalization of frame pixels.
5. Feature extraction from frame using pre-trained model.
6. Creating sequence of input vector
7. Input vector is passed to model
8. Model gives classification prediction output to user

3.1.4 Dataflow Diagram (Level 2)

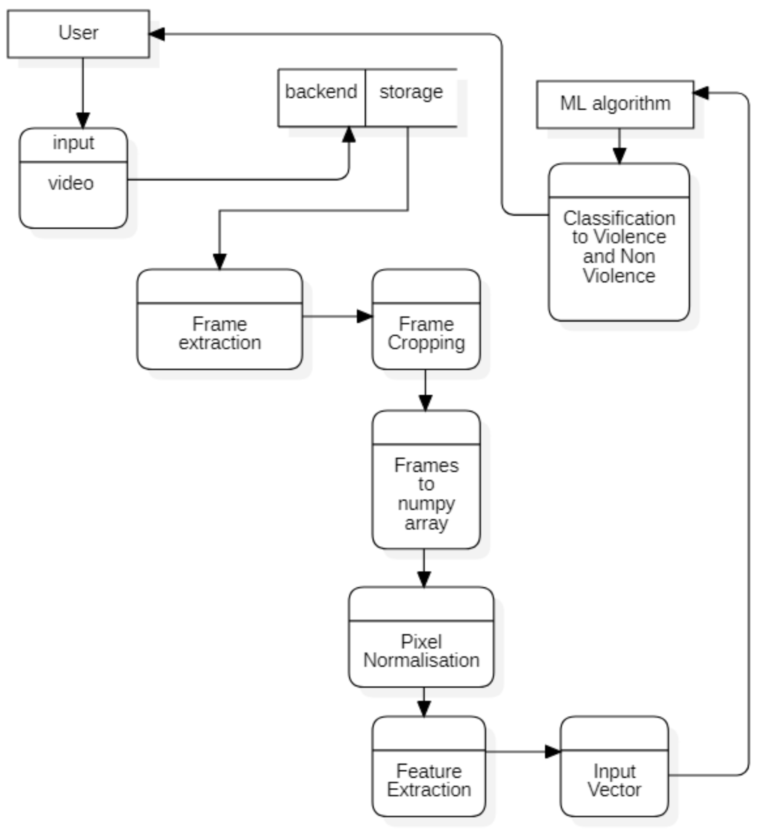


Fig 3.3 Dataflow Diagram

1. User will input the video using web application.
2. Frames will be extracted and cropped from video.
3. The frame will be converted into numpy array.
4. Normalization of frame pixels.
5. Feature extraction from frame using pre-trained model.
6. Creating sequence of input vector
7. Input vector is passed to model
8. Model gives classification prediction output to user

3.1.5 Sequence Diagram

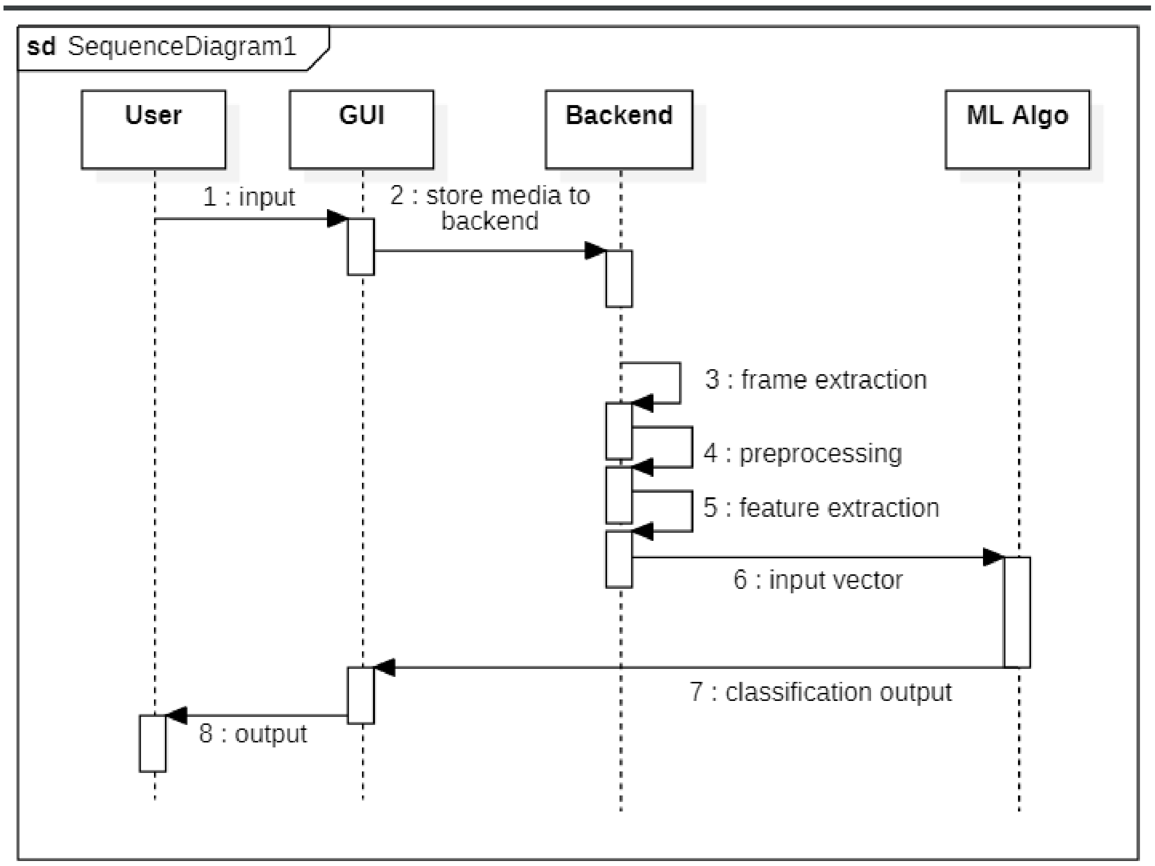


Fig 3.4 Sequence Diagram

1. User will input the video using web application.
2. Video will be saved at backend storage.
3. Frames will be extracted and cropped from video.
4. The frame will be converted into numpy array and normalization of frame pixels.
5. Feature extraction from frame using pre-trained model.
6. Creating sequence of input vector
7. Input vector is passed to model
8. Model gives classification prediction output to user

Chapter 4

Design

4.1 Architectural Design

4.1.1 Architectural Context Diagram

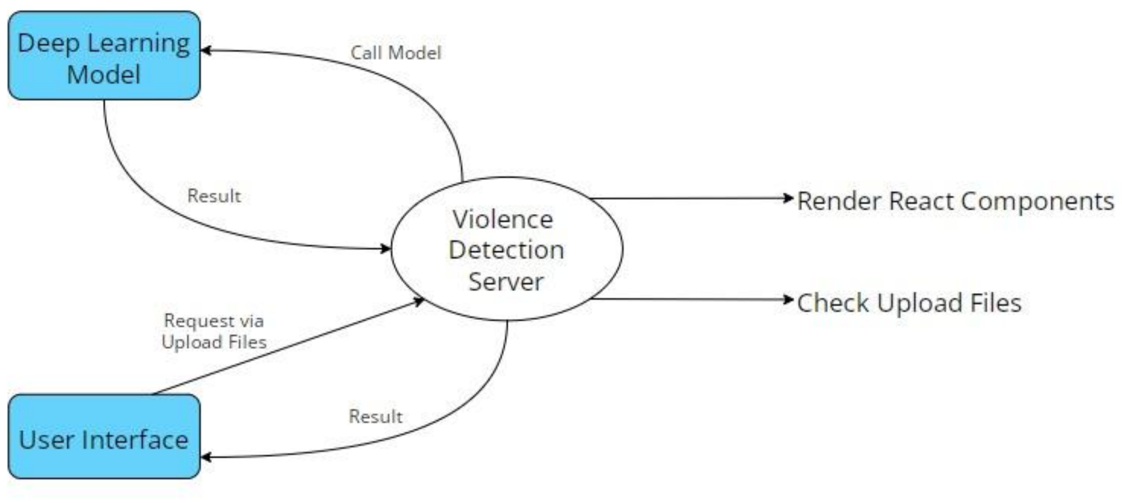


Fig 4.1 Architectural Context Diagram

4.1.2 Description of Architectural Diagram

- User or Applicant hits the post request and upload video file, then Flask backend call the inference deep learning model from Server and get the predicted output from it.
- Then backend renders the resultant React Components to the user with desire output.
- Model launches the inference of video classification form getting the prediction via it.
- User or Applicant sees the prediction on its interface with confidence.

4.2 Model Architecture

4.2.1 Multi-Layer Perceptron (ANN)

The Perceptron consists of an input layer and an output layer which are fully connected. MLPs have the same input and output layers but may have multiple hidden layers in between the aforementioned layers, as seen below.

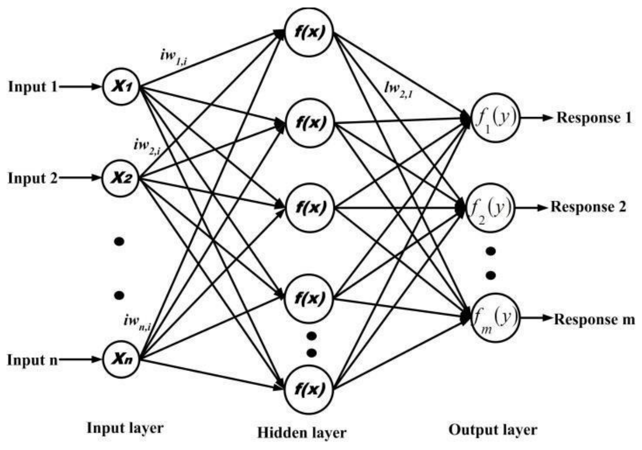


Fig4.2 MLP Neural Network

The algorithm for the MLP is as follows:

1. Just as with the perceptron, the inputs are pushed forward through the MLP by taking the dot product of the input with the weights that exist between the input layer and the hidden layer (WH). This dot product yields a value at the hidden layer. We do not push this value forward as we would with a perceptron though.
2. MLPs utilize activation functions at each of their calculated layers. There are many activation functions to discuss: rectified linear units (ReLU), sigmoid function, tanh. Push the calculated output at the current layer through any of these activation functions.
3. Once the calculated output at the hidden layer has been pushed through the activation function, push it to the next layer in the MLP by taking the dot product with the corresponding weights.
4. Repeat steps two and three until the output layer is reached.
5. At the output layer, the calculations will either be used for a backpropagation algorithm that corresponds to the activation function that was selected for the MLP (in the case of training) or a decision will be made based on the output (in the case of testing).

MLPs form the basis for all neural networks and have greatly improved the power of computers when applied to classification and regression problems. Computers are no longer limited by XOR cases and can learn rich and complex models thanks to the multilayer perceptron.

4.2.2 Long Short Term Memory RNN

Recurrent Neural Networks take the general principle of feed-forward neural networks and enable them to handle sequential data by giving the model an internal memory. The "Recurrent" portion of the RNN name comes from the fact that the input and outputs loop. Once the output of the network is produced, the output is copied and returned to the network as input. When making a decision, not only the current input and output are analyzed, but the previous input is also considered.

Long Short-Term Memory networks can be considered extensions of RNNs, once more applying the concept of preserving the context of inputs. However, LSTMs have been modified in several important ways that allow them to interpret past data with superior methods. The alterations made to LSTMs deal with the vanishing gradient problem and enable LSTMs to consider much longer input sequences.

LSTM models are made up of three different components, or gates. There's an input gate, an output gate, and a forget gate. Much like RNNs, LSTMs take inputs from the previous timestep into account when modifying the model's memory and input weights. The input gate makes decisions about which values are important and should be let through the model. A sigmoid function is used in the input gate, which makes determinations about which values to pass on through the recurrent network. Zero drops the value, while 1 preserves it. A TanH function is used here as well, which decides how important to the model the input values are, ranging from -1 to 1.

After the current inputs and memory state are accounted for, the output gate decides which values to push to the next time step. In the output gate, the values are analyzed and assigned an importance ranging from -1 to 1. This regulates the data before it is carried on to the next time-step calculation. Finally, the job of the forget gate is to drop information that the model deems unnecessary to make a decision about the nature of the input values. The forget gate uses a sigmoid function on the values, outputting numbers between 0 (forget this) and 1 (keep this).

An LSTM neural network is made out of both special LSTM layers that can interpret sequential word data and the densely connected like those described above. Once the data moves through the LSTM layers, it proceeds into the densely connected layers.

The Fig4.3 below shows RNN layer architecture and LSTM cell.

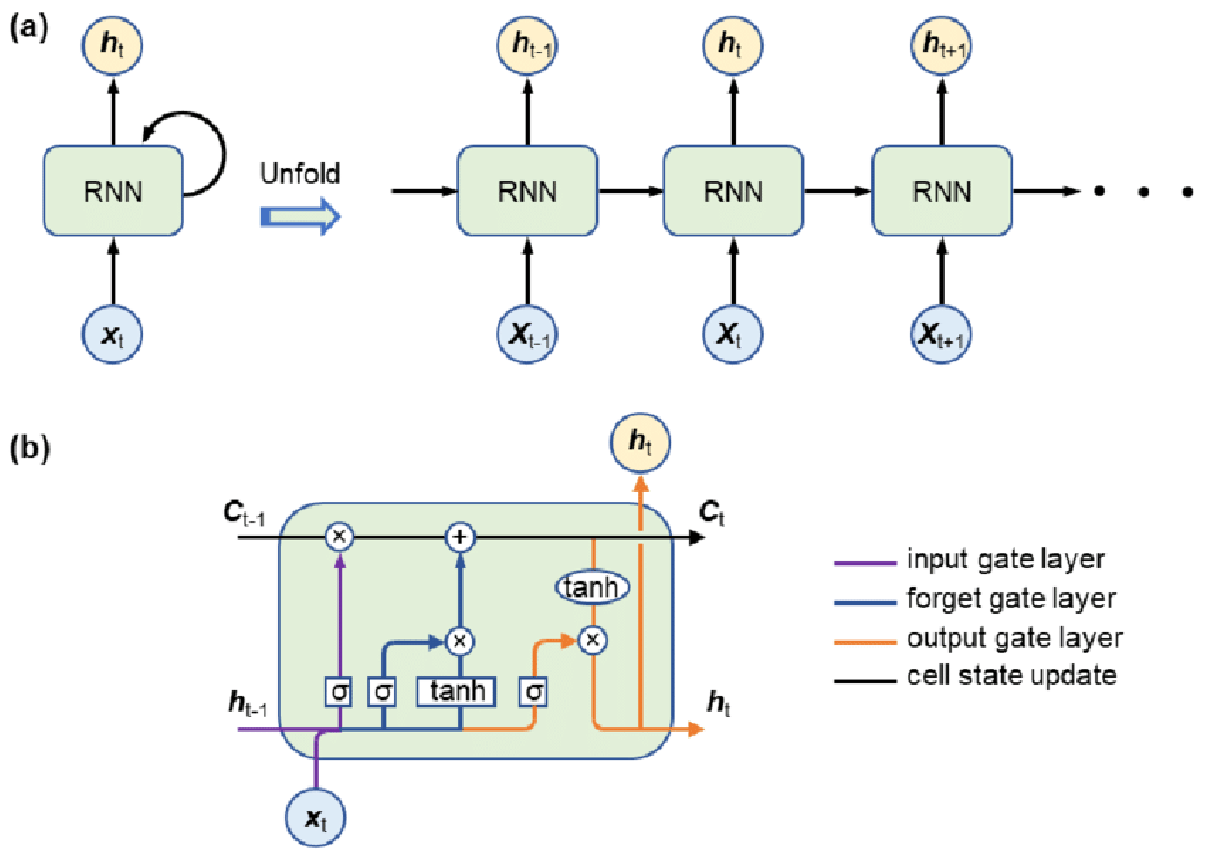


Fig4.3 RNN working and LSTM Cell Architecture

4.2.3 Inception v3 Pretrained Model

Inception v3 is an image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al.

Inception-v3 is a **pre-trained convolutional neural network** that is 48 layers deep, which is a version of the network already trained on more than a million images from the ImageNet database. This pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using Softmax.

Before the model can be used to recognize images, it must be trained using a large set of labeled images. ImageNet is a common dataset to use. ImageNet has over ten million URLs of labeled images. One million of the images also have bounding boxes specifying a more precise location for the labeled objects.

For Inception v3 model, the ImageNet dataset is composed of 1,331,167 images which are split into training and evaluation datasets containing 1,281,167 and 50,000 images, respectively

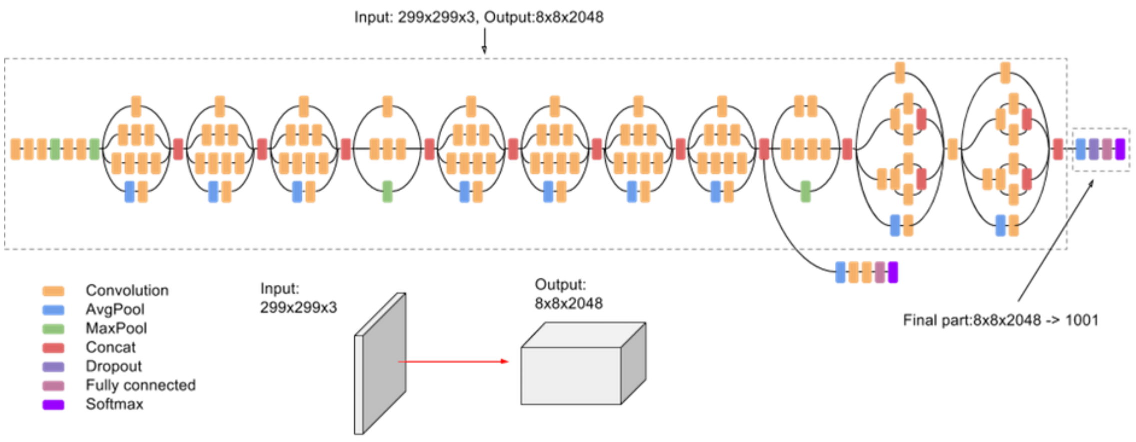


Fig 4.4 High-level diagram of Inception V3 model

4.2.4 Methodology

The following *Fig4.5 and Fig4.6* visualize the design architecture of the Multi-Layer Perceptron and LSTM ***base*** model.

The pre-trained model works as an feature extractor and takes input vector of shape (None , MAX_SEQUENCE_LEN, IMG_SIZE, IMG_SIZE, color_channel)

Where,

- MAX_SEQUENCE_LEN is max number of frames to be captured from the video i.e. 15
- IMG_SIZE is dimension of frame i.e. 224 in our model
- color_channel is number of color channel. 3 represents RGB/BGR/GRB channel and 1 represents grayscale image.

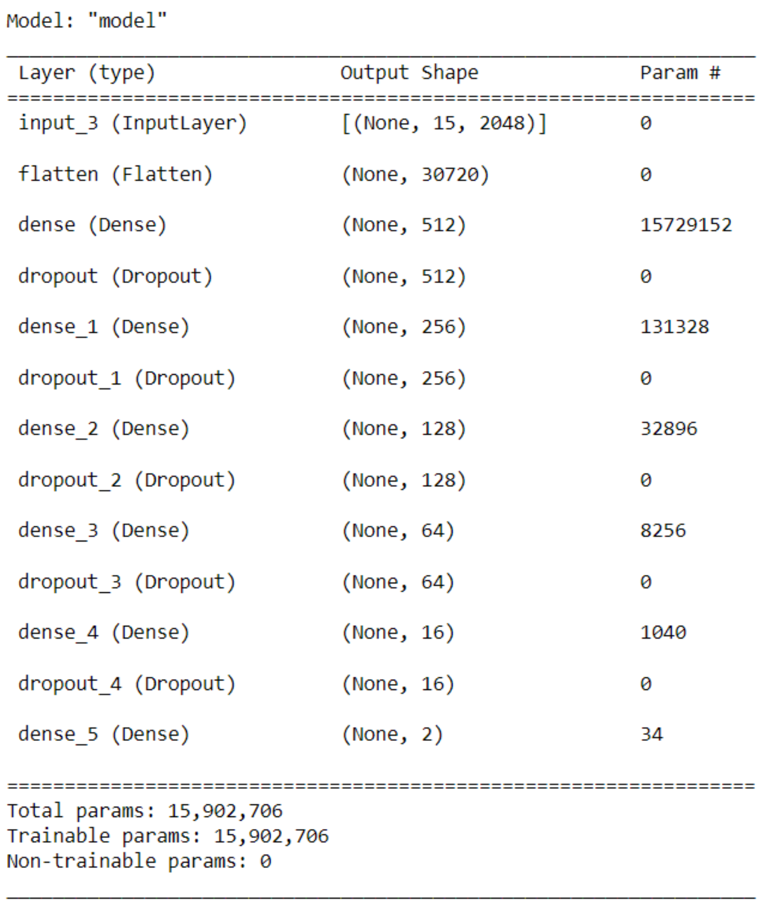
After each frame is captured and feature are extracted the main input vector is formed for base model which has a shape of (Num_of_videos, MAX_SEQUENCE_LEN, NUM_FEATURE)

Where,

- Num_of_videos is length of dataset.
- MAX_SEQUENCE_LEN is max number of frames to be captured from the video i.e. 15
- Since we used Inception V3 model for feature extraction, NUM_FEATURE is 2048

Since we used pre-trained model for feature extraction, we saved a lot of computation resources and time because instead of creating input vector of shape (1796,15,224,224,3) we passed input vector having shape (1796,15,2048) which directly reduced trainable parameters and hence the RAM usage was drastically decreased.

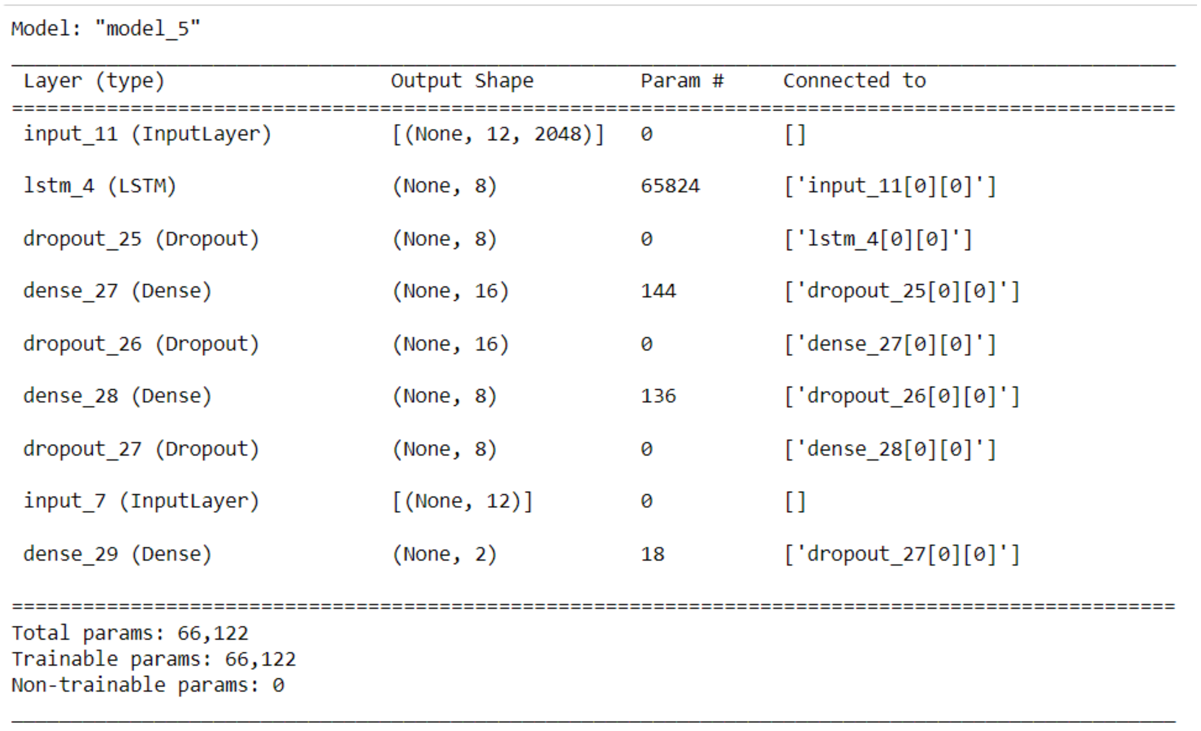
4.2.4.1 Multi-Layer Perceptron Model (ANN)



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 15, 2048)]        0

 flatten (Flatten)           (None, 30720)             0

 dense (Dense)               (None, 512)               15729152

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_2 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dropout_3 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 16)                1040

 dropout_4 (Dropout)         (None, 16)                0

 dense_5 (Dense)             (None, 2)                 34

=================================================================
Total params: 15,902,706
Trainable params: 15,902,706
Non-trainable params: 0
_________________________________________________________________
```

Fig 4.5 Multi-Layer Perceptron model architecture

4.2.4.2 Long Short Term Memory (LSTM) Model



```
Model: "model_5"
____________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
====================================================================================================
 input_11 (InputLayer)          [(None, 12, 2048)]   0           []

 lstm_4 (LSTM)                  (None, 8)            65824       ['input_11[0][0]']

 dropout_25 (Dropout)           (None, 8)            0           ['lstm_4[0][0]']

 dense_27 (Dense)               (None, 16)           144         ['dropout_25[0][0]']

 dropout_26 (Dropout)           (None, 16)           0           ['dense_27[0][0]']

 dense_28 (Dense)               (None, 8)            136         ['dropout_26[0][0]']

 dropout_27 (Dropout)           (None, 8)            0           ['dense_28[0][0]']

 input_7 (InputLayer)           [(None, 12)]         0           []

 dense_29 (Dense)               (None, 2)            18          ['dropout_27[0][0]']

====================================================================================================
Total params: 66,122
Trainable params: 66,122
Non-trainable params: 0
____________________________________________________________________________________________________
```

Fig 4.6 LSTM model architecture

4.3 Modules Used

4.3.1.1 NumPy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behaviour is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also, it is optimized to work with latest CPU architectures. NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

4.3.1.2 Pandas is an open-source library in Python. It provides ready to use high-performance data structures and data analysis tools. Pandas module runs on top of NumPy and it is popularly used for data science and data analytics. NumPy is a low-level data structure that supports multi-dimensional arrays and a wide range of mathematical array operations. Pandas has a higher-level interface. It also provides streamlined alignment of tabular data and powerful time series functionality. DataFrame is the key data structure in Pandas. It allows us to store and manipulate tabular data as a 2-D data structure. Pandas provides a rich feature-set on the DataFrame. For example, data alignment, data statistics, slicing, grouping, merging, concatenating data, etc. DataFrame is the most important and widely used data structure and is a standard way to store data. DataFrame has data aligned in rows and columns like the SQL table or a spreadsheet database. We can either hard code data into a DataFrame or import a CSV file, tsv file, Excel file, SQL table, etc. We can use the below constructor for creating a DataFrame object.

4.3.1.3 Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy. In this tutorial we will learn to code python and apply Machine Learning with the help of the scikit-learn library, which was created to make doing machine learning in Python easier and more robust. Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.

4.3.1.4 Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib’s APIs (Application Programming Interfaces) to embed plots in GUI applications.

A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot.

4.3.1.5 TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well. TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages.

4.3.1.6 Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend. Flask is a web application framework written in Python. Flask is based on the **WSGI** toolkit and the **Jinja2** template engine.

WSGI The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications.

Jinja2 is a popular template engine for Python. A web template system combines a template with a specific data source to render a dynamic web page. This allows you to pass Python variables into HTML templates.

4.3.1.7 OpenCV (Open Source Computer Vision) is a popular computer vision library that provides a wide range of image and video processing functions, as well as machine learning algorithms. It is an open-source library and is available for use under the BSD license. The library provides a set of tools and algorithms for various tasks, such as image processing, feature detection, object recognition, and machine learning. OpenCV supports several programming languages, including C++, Python, and Java, making it accessible to a wide range of developers. Some of the features of OpenCV include image and video manipulation, object detection and recognition, face detection and recognition, optical flow and motion estimation, and camera calibration. OpenCV also provides support for deep learning frameworks such as

TensorFlow, Caffe, and PyTorch, enabling developers to leverage the power of deep learning for computer vision tasks.

4.3.1.8 Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result as fast as possible is key to doing good research.*

Keras is:

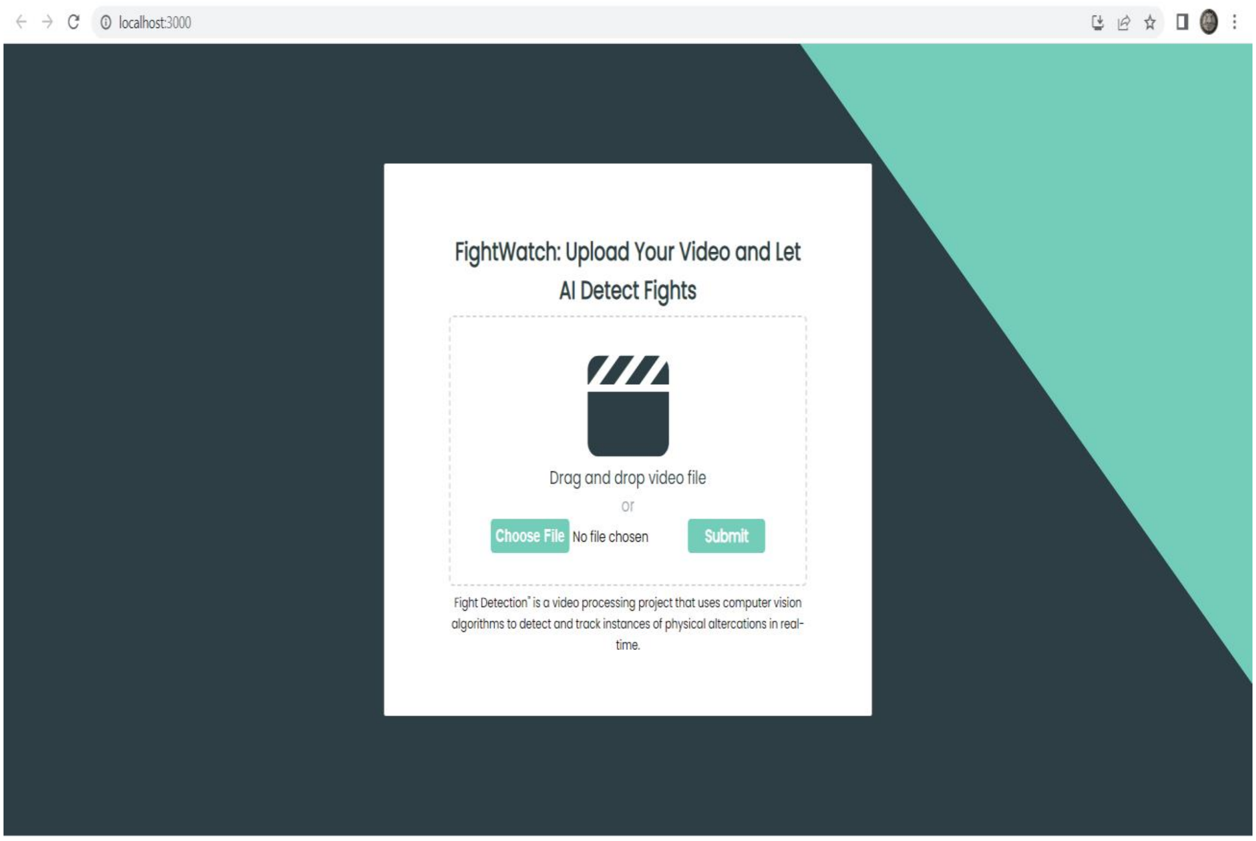
- Simple -- but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, and Waymo.

4.3.1.9 React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library[3] for building user interfaces based on components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js. Because React is only concerned with the user interface and rendering components to the DOM, React applications often rely on libraries for routing and other client-side functionality

4.4 Interface Design

4.4.1 Human-machine interface design specification

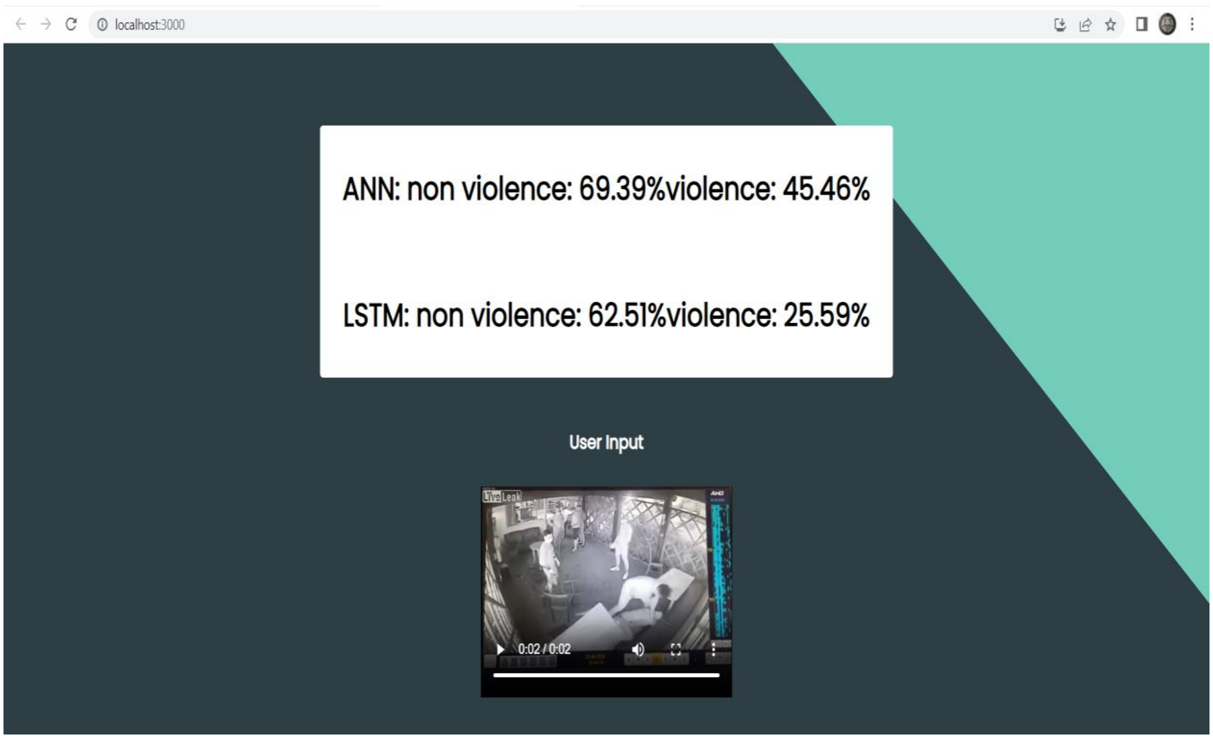
1. HTML pages are rendered after launching React Server
2. User clicks on “Choose File” button and browse video or drag and drops the file.
3. Video undergoes preprocessing and is then given as input to model. Model generates resultant output, that is displayed on Result page along with the video that user submitted.


localhost:3000

FightWatch: Upload Your Video and Let AI Detect Fights

Fight Detection" is a video processing project that uses computer vision algorithms to detect and track instances of physical altercations in real-time.

Fig 4.7 – Home Page UI


localhost:3000

ANN: non violence: 69.39%violence: 45.46%

LSTM: non violence: 62.51%violence: 25.59%

Fig 4.8 – Result UI

4.5 Data Design / Data Results

4.5.1 Data objects and resultant data

4.5.1.1 Confusion Matrix

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**. Some features of Confusion matrix are given below:

- For the 2 prediction classes of classifiers, the matrix is of 2*2 table, for 3 classes, it is 3*3 table, and so on.
- The matrix is divided into two dimensions, that are **predicted values** and **actual values** along with the total number of predictions.
- Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations.
- It looks like the below table:

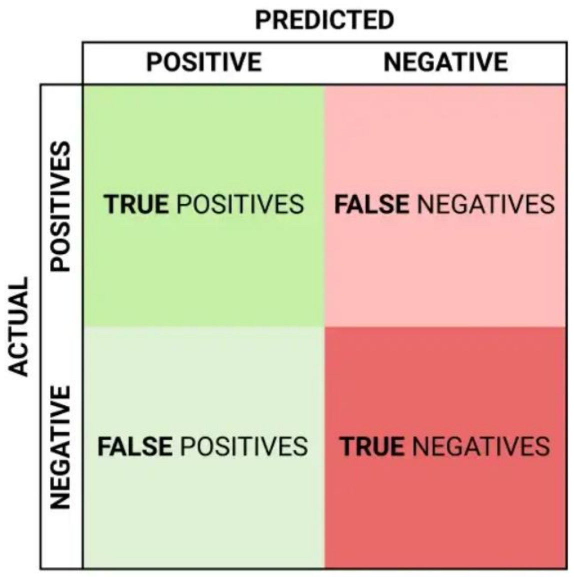


Fig 4.9 Confusion Matrix Format

The above table has the following cases:

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

The following Confusion matrix represents the performance of our sequential model. The dataset consists of 2 classes hence we have 2x2 matrix.

Where 0 = Non-Violence and 1= Violence

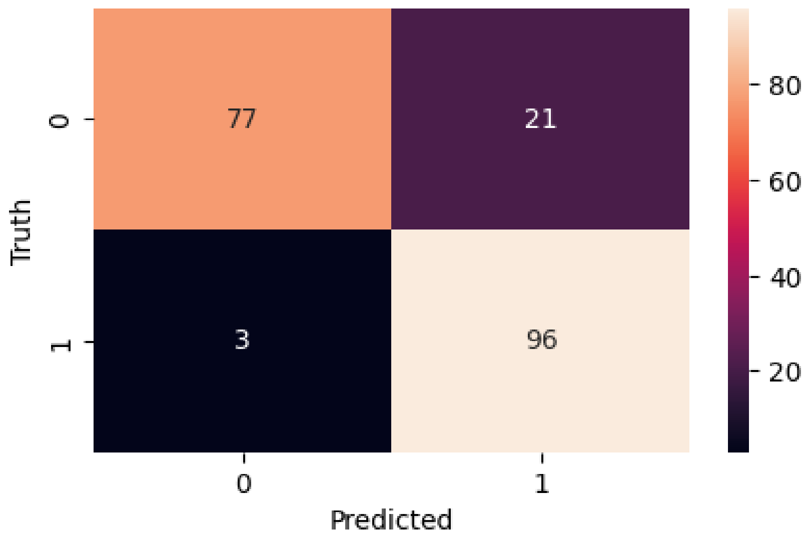


Fig 4.10 Confusion matrix of MLP model

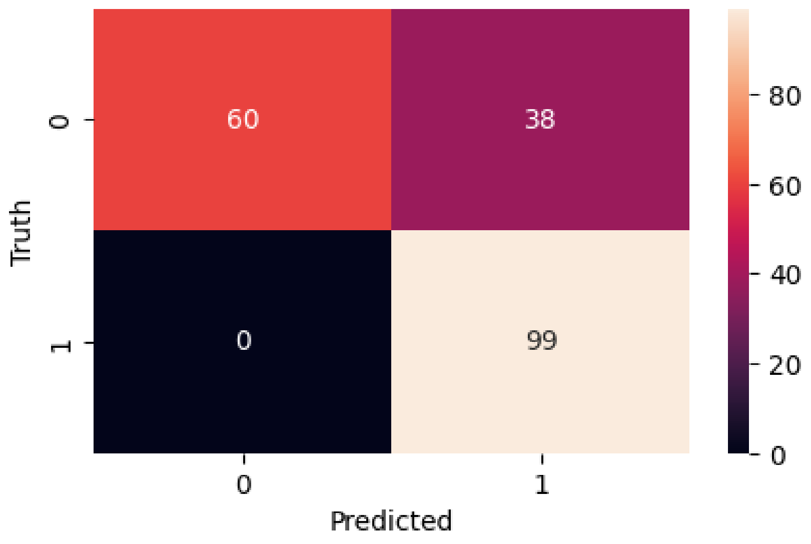


Fig 4.11 Confusion matrix of LSTM model

4.5.1.2 Training vs Validation

The training loss is a metric used to assess how a deep learning model fits the training data. It assesses the error of the model on the training set. Note that, the training set is a portion of a dataset used to initially train the model. Computationally, the training loss is calculated by taking the sum of errors for each example in the training set. It is also important to note that the training loss is measured after each batch. This is usually visualized by plotting a curve of the training loss.

On the contrary, validation loss is a metric used to assess the performance of a deep learning model on the validation set. The validation set is a portion of the dataset set aside to validate the performance of the model. The validation loss is similar to the training loss and is calculated from a sum of the errors for each example in the validation set.

The training and validation loss is usually visualized together on a graph. The purpose of this is to diagnose the model's performance and identify which aspects need tuning.

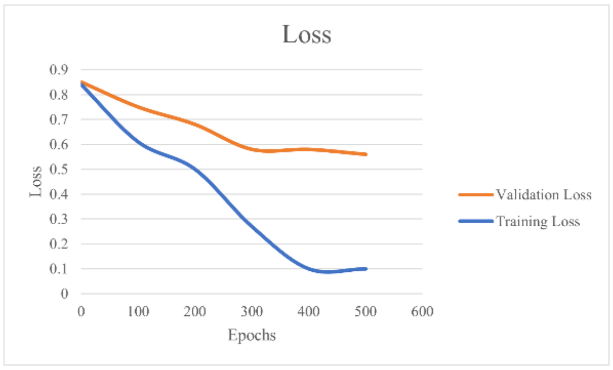


Fig 4.12 Underfitting

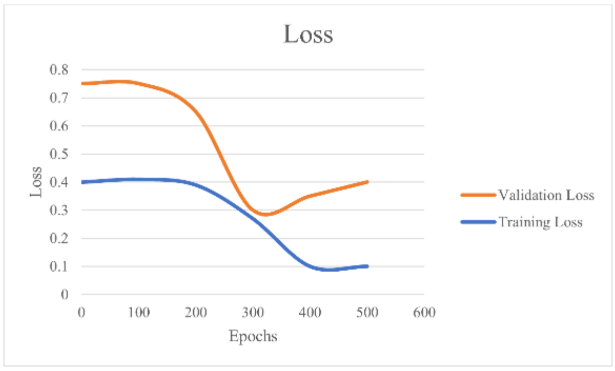


Fig 4.13 Overfitting

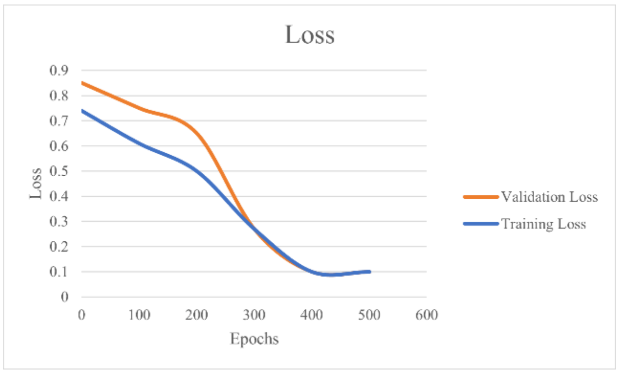


Fig 4.14 Good Fit

\

Accuracy is a method for measuring a classification model's performance. It is typically expressed as a percentage. Accuracy is the count of predictions where the predicted value is equal to the true value. It is binary (true/false) for a particular sample. Accuracy is often graphed and monitored during the training phase though the value is often associated with the overall or final model accuracy. Accuracy is easier to interpret than loss.

The following plot depicts Training vs Validation Accuracy and Training vs Validation Loss of our designed Sequential Models for Violence Detection.

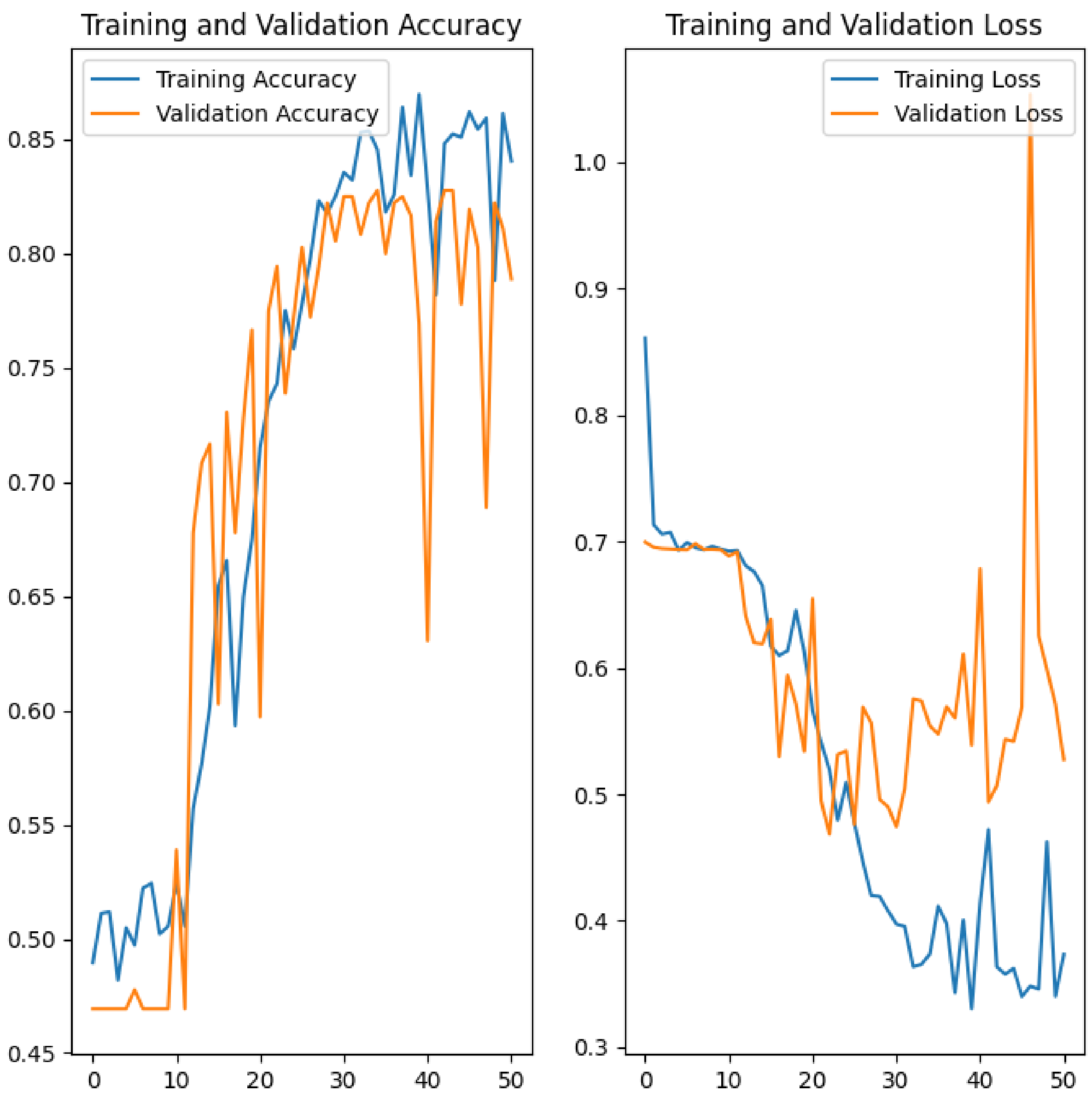


Fig 4.15 Training VS Validation accuracy and loss plots of LSTM

By analyzing the above plots we can conclude that the our Recurrent Neural Network achieved good fit and provides more than 85+% accuracy on both Training and Validation dataset.

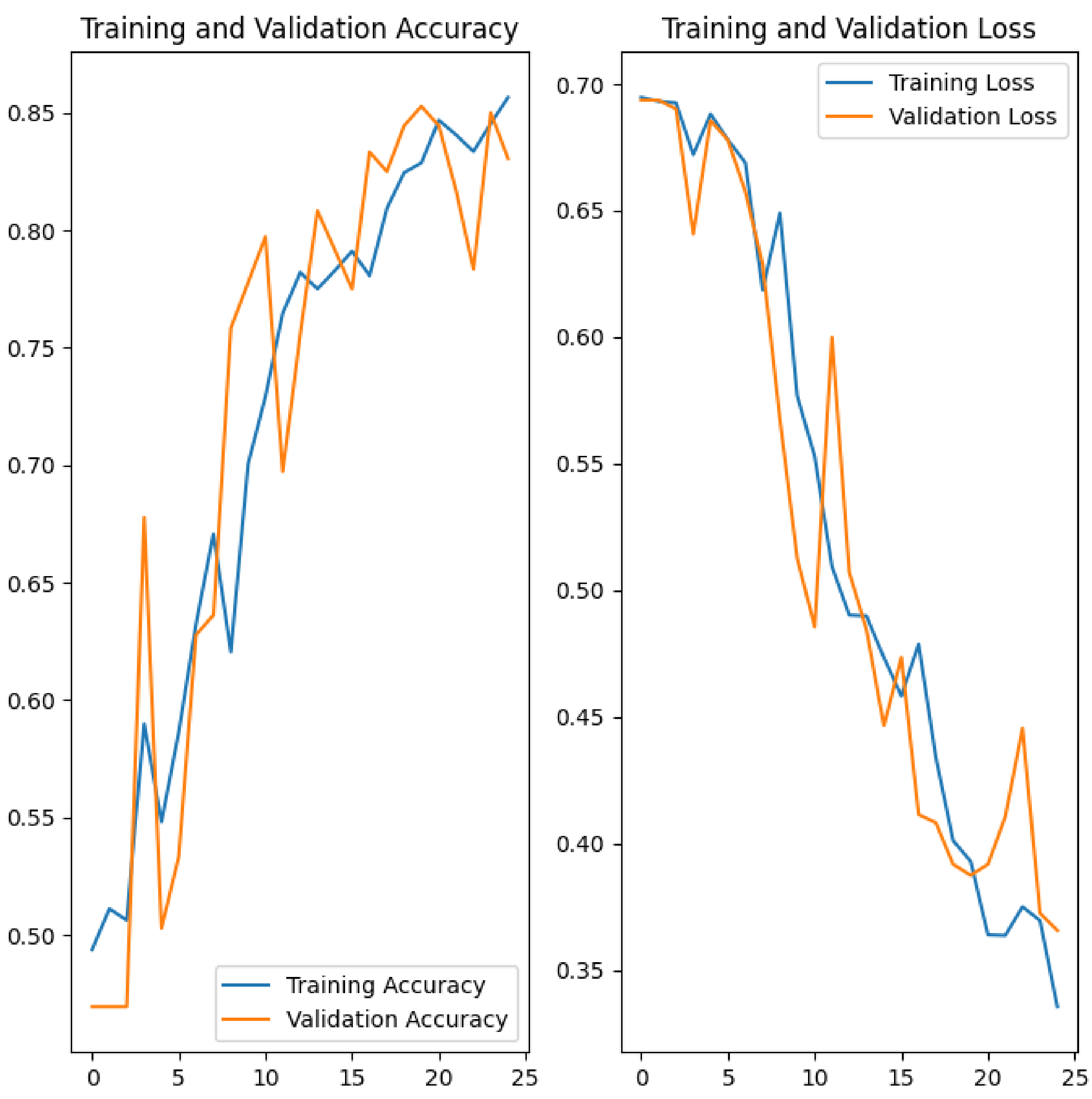


Fig 4.16 Training VS Validation accuracy and loss plots of MLP

By analyzing the above plots we can conclude that the our Multi-Layer Perceptron Neural Network achieved good fit and provides more than 85+% accuracy on both Training and Validation dataset.

Chapter – 5

Testing

5.1 Testing Objective

In machine learning, a programmer usually inputs the data and the desired behavior, and the logic is elaborated by the machine. This is especially true for deep learning. Therefore, the purpose of machine learning testing is, first of all, to ensure that this learned logic will remain consistent, no matter how many times we call the program

5.2 Testing Scope

Evaluating the performance of the model using different metrics is integral to every data science project. Here is what you have to keep an eye on: Accuracy is a metric for how much of the predictions the model makes are true. The higher the accuracy is, the better. However, it is not the only important metric when you estimate the performance.

Accuracy is a metric for how much of the predictions the model makes are true. The higher the accuracy is, the better. However, it is not the only important metric when you estimate the performance.

Machine Learning testing scope is determined by the following resultant parameters. Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal i.e., TP = TP +FP, this also means FP is zero. As FP increases the value of denominator becomes greater than the numerator and precision value decreases (which we don’t want).

Recall should ideally be 1 (high) for a good classifier. Recall becomes 1 only when the numerator and denominator are equal i.e. TP = TP +FN, this also means FN is zero. As FN increases the value of denominator becomes greater than the numerator and recall value decreases (which we don’t want).

So ideally in a good classifier, we want both precision and recall to be one which also means FP and FN are zero. Therefore, we need a metric that takes into account both precision and recall. F1-score is a metric which takes into account both precision and recall and is defined as follows:

$$
\begin{aligned}
precision &= \frac{TP}{TP + FP} \\
recall &= \frac{TP}{TP + FN} \\
F1 &= \frac{2 \times precision \times recall}{precision + recall}
\end{aligned}
$$

Fig 5.1 Testing metrics formula

Following figures shows metrics reports of our model:

```
              precision    recall  f1-score   support

 NonViolence       0.96      0.79      0.87        98
    Violence       0.82      0.97      0.89        99

    accuracy                           0.88       197
   macro avg       0.89      0.88      0.88       197
weighted avg       0.89      0.88      0.88       197
```

Fig5.2 metrics report of MLP model

```
              precision    recall  f1-score   support

 NonViolence       1.00      0.61      0.76        98
    Violence       0.72      1.00      0.84        99

    accuracy                           0.81       197
   macro avg       0.86      0.81      0.80       197
weighted avg       0.86      0.81      0.80       197
```

Fig5.3 metrics report of LSTM model

5.3 Testing Methods

Used First of all, you split the dataset into three non-overlapping sets. You use a training set to train the model. Then, to evaluate the performance of the model, you use two sets of data

Validation set:

Having only a training set and a testing set is not enough if you do many rounds of hyperparameter-tuning (which is always). And that can result in overfitting. To avoid that, you can select a small validation data set to evaluate a model. Only after you get maximum accuracy on the validation set, you make the testing set come into the game.

Test set:

Your model might fit the training dataset perfectly well. But where are the guarantees that it will do equally well in real-life? In order to assure that, you select samples for a testing set from your training set — examples that the machine hasn't seen before. It is important to remain unbiased during selection and draw samples at random. Also, you should not use the same set many times to avoid training on your test data. Your test set should be large enough to provide statistically meaningful results and be representative of the data set as a whole.

But just as test sets, validation sets "wear out" when used repeatedly. The more times you use the same data to make decisions about hyperparameter settings or other model improvements, the less confident you are that the model will generalize well on new, unseen data

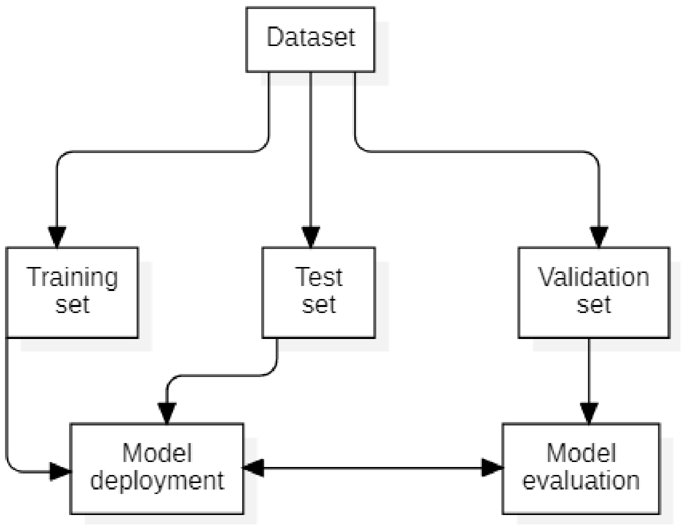


Fig 5.4 Train Test Validation Split

5.4 Sample Test Data & Results

Performing ML tests is necessary if you care about the quality of the model. ML testing has a couple of peculiarities: it demands that you test the quality of data, not just the model, and go through a couple of iterations adjusting the hyperparameters to get the best results.

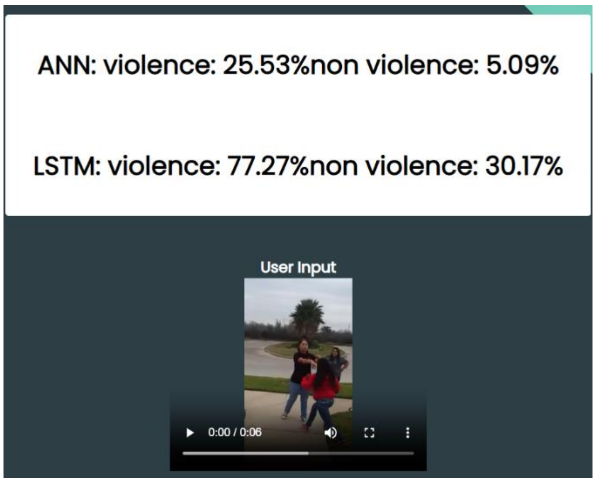
The test dataset comprised of 197 videos selected randomly from original dataset. The MLP model achieved 87% accuracy and LSTM model achieved 81% accuracy on test dataset.

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

Here are the generic steps followed to carry out any type of Black Box Testing.

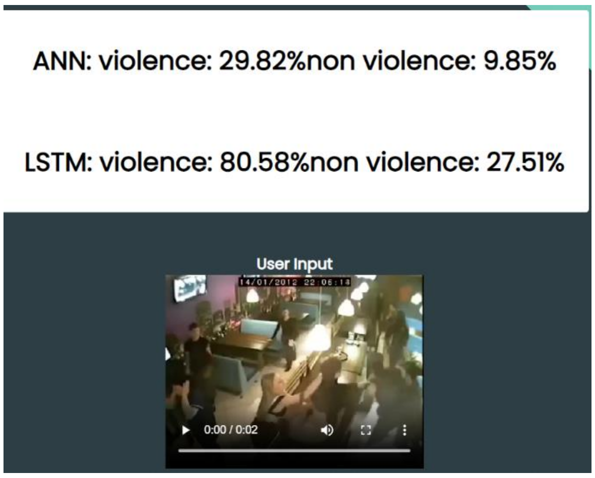
- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

We performed black box testing on unseen data, and we can conclude that model performed 90% accuracy as only 1 out of 10 video was classified wrong by both models. Below are some images visualizing the outputs.


ANN: violence: 25.53%non violence: 5.09%

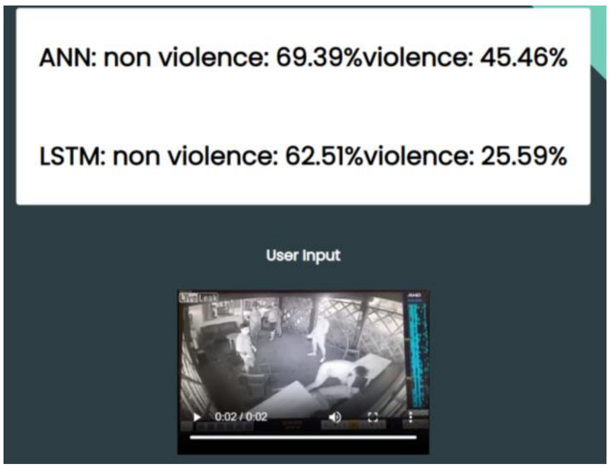
LSTM: violence: 77.27%non violence: 30.17%

Fig 5.5 Manual test result 1


ANN: violence: 29.82%non violence: 9.85%

LSTM: violence: 80.58%non violence: 27.51%

Fig 5.6 Manual test result 2


ANN: non violence: 69.39%violence: 45.46%

LSTM: non violence: 62.51%violence: 25.59%

Fig 5.6 Manual test result 3

Chapter 6

Limitations

Nothing is perfect in this world, so are our model as there are few limitation while using the proposed system.

- The system accuracy will decrease drastically if the video does not have proper light effects i.e., bright spots, dark environment such that object aren't properly visible.

- Since we crop the frame during preprocessing, it is possible that we may crop the pixels where actual fight is taking place. For example in the images below the object are at left corner before cropping(fig 6.1). But after cropping we lost the actual fight scene from frame (fig 6.2) which resulted in misclassification.

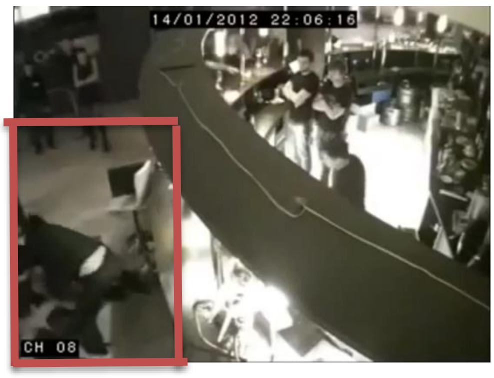


Fig6.1 Before Cropping

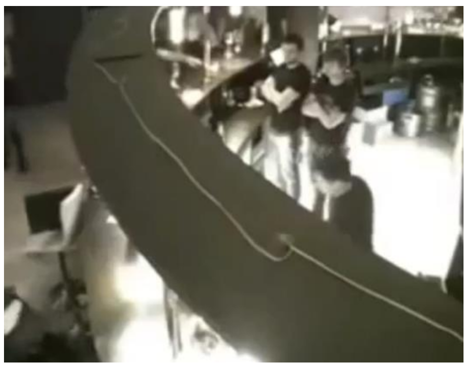


Fig6.2 After Cropping

- Both MLP and LSTM model gave high False Negative which affects the sensitivity of model.

Chapter 7

Future Scope

The future scope of violence detection using machine learning is vast and promising. Here are some potential areas of growth:

1. Improving accuracy: There is still room for improvement in the accuracy of violence detection algorithms. Researchers can focus on developing more advanced deep learning models that can better detect and classify violent events.

2. Multimodal analysis: Currently, most violence detection algorithms rely on visual and audio cues. In the future, researchers can explore the use of other modalities, such as biometric and environmental data, to improve the accuracy and reliability of violence detection.

3. Scalability: Violence detection can be applied in various domains, from public safety to social media monitoring. Future research can focus on developing scalable models that can be adapted to different domains and applications.

4. For Feature extraction, *Pose Estimation Algorithm* or *Object detection algorithms like YoloV8* can be used to detect humans which may help in increasing the accuracy and other limitations which we are facing now.

5. A system with Ensemble technique can be used where final output is by voting of outputs from different algorithms, which can help increasing accuracy.

Chapter 8

Conclusion

Demand of human–computer interaction has elevated this field. The researchers of this field are putting in great efforts for developing computer vision systems which can be practically implemented to aid society. With the increasing development of surveillance cameras in many areas of life to watch human behavior, the need for systems that automatically identify violent occurrences increases proportionally. Violent action detection has become a prominent subject in computer vision attracting new researchers. Many academics have suggested various methods for detecting such actions in videos. The techniques discussed in this project are relevant works of various researchers with a common aim of improvement in recognition rates. It can be concluded that implemented classification techniques correctly classifies videos with good accuracy over the legacy systems.

The model and system were developed in python and libraries like keras, numpy, matplotlib, PIL, openCV2, flask etc. were used.

The model trained in this project was trained, tested and validated on dataset consisting of 1996 videos.

The MLP model achieved 87% accuracy and LSTM model achieved 81% accuracy on test dataset which was comprised 198 videos with equal number of videos from each class.

The MLP model achieved 0.89 F1 score and LSTM model achieved .84 F1 score.

Chapter 9

Bibliography and References

1. *Naik, Anuja Jana, and M. T. Gopalakrishna. "Violence detection in surveillancevideo-a survey." International Journal of Latest Research in Engineering and Technology (IJLRET) 1 (2017): 1-17.*

2. *Ding, Chunhui, et al. "Violence detection in video by using 3D convolutional neural networks." Advances in Visual Computing: 10th International Symposium, ISVC 2014, Las Vegas, NV, USA, December 8-10, 2014, Proceedings, Part II 10. Springer international publishing, 2014.*

3. *Fenil, E., et al. "Real time violence detection framework for football stadium comprising of big data analysis and deep learning through bidirectional LSTM." Computer Networks 151 (2019): 191-200.*

4. *Arandjelovic, Relja, et al. "NetVLAD: CNN architecture for weakly supervised place recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.*

5. *Mohtavipour, Seyed Mehdi, Mahmoud Saeidi, and Abouzar Arabsorkhi. "A multi-stream CNN for deep violence detection in video sequences using handcrafted features." The Visual Computer (2022): 1-16.*

6. *Meng, Quanling, et al. "Action recognition using form and motion modalities." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 16.1s (2020): 1-16.*

7. *Ullah, Fath U. Min, et al. "Violence detection using spatiotemporal features with 3D convolutional neural network." Sensors 19.11 (2019): 2472.*