Identify Gate Operations using Action Recognition

DISSERTATION

Submitted in partial fulfillment of the requirements of the

M.Tech Data Science and Engineering Degree programme

By

VISHALI S

ID No.

Under the supervision of:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani (Rajasthan) INDIA

August 2022

Acknowledgements

It is a great pleasure for me to present this project to all of you. I would like to acknowledge each one who had a role to play in making my humble efforts an out-to-out success. I would like to thank my M. Tech supervisor, who provided all facilities, guidance & encouragement during my study. I extend my sincere thanks to for providing the opportunity to pursue this project. I dedicate my wholehearted thanks to for their support, motivation, and technical insights during the project work, without which completing this work could have been a herculean endeavor. I express my sincere thanks to my team at for their endless encouragement and support throughout the project. I would like to dedicate my wholehearted gratitude to my parents and friends who kept my spirits high without whose cooperation the completion of the project would not be possible.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled "Identify Gate Operations using Action Recognition" and submitted by Miss. Vishali S ID No. . in partial fulfillment of the requirements of DSECLZG628T Dissertation, embodies the work done by her under my supervision.

(Signature of the Supervisor)

Place: Hyderabad

Date: 27th August 2022

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

SECOND SEMESTER 2021-22

DSECLZG628T DISSERTATION

Dissertation Title	: Identify Gate Operations using Action Recognition
Name of Supervisor	:
Name of Student	: Vishali S
ID No. of Student	:

Abstract

Key Words: Action Recognition, Computer Vision, Deep Learning

Air Traffic Controller (ATC) plays very crucial role in aviation to schedule/organize the aircrafts safely and efficiently without any collisions through the worldwide airspace system. The scheduling and organization of turnaround activities are critical to optimize the air traffic in airport premises. The turnaround activities need to be well-developed and meticulously orchestrated for many airlines. A precise prediction of aircraft readiness for departure can help Air Traffic Control to plan an optimal pre-departure sequence at which aircraft are dispatched from the parking stands. Airport ramp areas are the most crowded and cluttered spaces. Operations associated with turnaround of the aircraft from the gate represents a significant source of delay and therefore needs to be improved. This turnaround process consists of several activities such as boarding and disembarking passengers, cabin-cleaning and catering, aircraft maintenance, baggage handling and cargo, security checks, etc. and involves several ground support equipment and vehicles. The estimation of delay from individual activities and overall turnaround delay is critical for proper organization of scheduling and pipelining of the process. The prior knowledge or precise estimation of these delays can mitigate the unwanted wait-times as well as improves the efficiency and occupancy of the airport resources. State of the art Computer Vision (CV) techniques can precisely recognize these actions and automatically calculate the turnaround time. Action recognition is one of the active research areas in the CV with emerging real-world application such as, security surveillance, entertainment, and video retrieval, etc. This recognizes various actions from series of observations on the actions of subjects and helps in prediction of future state by inferring the current action being performed. In this project, we propose a model based on visual-analytical approach that can automatically recognize any two airport gate monitoring activities around airport apron area to improve operational efficiency, safety, and security of the servicing operation. Actions in a stream of images, videos, are being recognized in this problem. The spatiotemporal features of the videos are extracted using the image processing techniques. This preprocessing of video datasets is important for extracting crucial features to differentiate different activities with similar attributes/objects. After preprocessing, the models are trained with extracted features using the deep learning algorithms to preciously recognize the actions. The models will be created, trained and the preprocessing of dataset will be performed in Python environment. The performance of the models will be evaluated in terms of classification accuracy and confidence levels. Minimizing the time at the gate is important at busy airports and helps airlines keep aircraft flying.

Abbreviations

ATC	**A**ir **T**raffic **C**ontrol
CV	**C**omputer **V**ision
TAT	**T**urn**a**round **T**ime
AR	**A**ction **R**ecognition
CNN	**C**onvolutional **N**eural **N**etwork
SIFT	**S**cale **I**nvariant **F**eature **T**ransform
HoG	**H**istogram of **O**riented **G**radients
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**Long S**hort-**T**erm **M**emory
LRCN	**L**ong **R**ecurrent **C**onvolution **N**etwork

List of Tables

List of Figures

Contents

Chapter 1

Introduction

In the aviation industry, usually the plane is not making any money while it is on the ground. The flight owners must pay the rent if their flights are on ground. Airport ground operations are one of the main reasons for late departures. Always airlines wish to keep their planes grounded for as short as possible. The flights flown on different airlines between the same airports take different amount of time. There are several nonstop flight options with varying duration or block times. This time includes the total amount of time a flight takes from pushing back from the departure gate, to arriving at the destination gate. Block time is split into three parts - time to taxi-out to the runway, the actual flight duration, and the time to taxi to the arrival gate. In aviation we often use terms like off-block time and on-block time. Off-block time is defined as the taxi-time before take-off that is when the aircraft leaves the parking stand towards the departure runway. On-block time is defined as the taxi-time after landing which means the time taken by the aircraft to reach the final parking position. The displayed schedule while booking flight tickets doesn't show the breakdown of the block time. The airline block times vary for the same routes. Not having a quick turnaround time can cost business. In many cases, the amount of time it takes to complete represents cost and can present risk. Fast turnaround time means less money spent on manpower. Behind the scenes, airlines spend enormous amounts of time and resources in planning the schedule.

1.1 Problem Statement

Every industry has specific processes that are measured for efficiency and effectiveness. Airport ground operations are one of the main causes of late departures. A common mantra in the aviation industry is "The plane is not making any money while it is on the ground". This statement represents the underlying hurdles faced by the airline industry. There is a need to strive for efficiency, as airlines wish to keep their planes grounded for as short as possible. Turnaround times are essential to this industry and becomes more critical in short-haul operations, where they account for a higher percentage of the flight time. Operations associated with turnaround of the aircraft from the gate represent a significant source of delay and therefore needs to be improved. The turnaround time (TAT) of an aircraft is the time that passes from landing until take off for a new flight. Hence, reliable work plans for ground support resources can help mitigate these delays while limiting the under-/over-utilization of equipment. This will improve the efficiency, safety, and security of the airport operations.

1.2 Purpose

Air Traffic Controller (ATC) plays a crucial role in aviation to schedule/organize the aircrafts safely and efficiently without any collisions through the worldwide airspace system. The scheduling and organization of turnaround activities are critical to optimize the air traffic in airport premises. Operations associated with turnaround of the aircraft from the gate represent significant source of delay and therefore needs to be improved. If the airline schedules the flight in an assertive way, then the flight will more often fail to meet its schedule. On the other hand, if the airline schedules in a conservative manner to avoid delayed flights, the scheduled duration will be longer, which may make the flight less attractive to travelers and expensive for the airlines. The turnaround activities need to be well-developed and meticulously orchestrated for many airlines. A precise prediction of aircraft readiness for departure can help Air Traffic Control to plan an optimal pre-departure sequence at which aircraft are dispatched from the parking stands. The reason behind the variation between airlines and flights between same origin and destination is where airlines priorities can come into play.

The objective of this project is to build a system which will take video as an input and recognize the ground activities such as bridge connecting, cabin-cleaning, and catering, aircraft maintenance, re-fueling, loading, and unloading the baggage, security checks, etc. Due to the time constraint, we have limited to recognize three activities which are bridge connecting, baggage handling, and miscellaneous. To achieve this, we have built a model based on visual-analytical approach that can automatically recognize the activities. This requires a large dataset to train our model.

1.3 Significance of block time in Airline Schedules

An airline might schedule a flight more conservatively to meet on-time performance and reduces the impact of disruptions on other flights in the network. However, this longer scheduled block time results at the expense of paying more for flight crews and lower aircraft utilization rate. This means there will be only fewer flights per aircraft per unit of time. On the other hand, when an airline schedules shorter block times, this may be able to lower their costs by operating additional flights, carrying more passengers with a single aircraft, and paying less in-flight crew costs. However, this way of dealing might risk poorer on-time performance and creates more risk that delays on a flight will impact downline flights that were scheduled to use the same aircraft.

Utilizing state of the art action recognition models to recognize the ground activities and this in turn helps to improve the airline operations. The travelers while booking their flight tickets, they tend to look at trip duration which has got good performance numbers closely over time. Real-time flight status data can be good reference point in this area. The pattern you will see is that shorter block times may seem to be attractive, but often do not perform well. Conversely, longer block times will tend to provide more time and greater buffer when connecting to other flights. On-time performance is often an important factor in choosing a flight. Knowing the tradeoffs between shorter schedule times and on-time performance can help one to make better decisions on choosing flights. Poorer on-time performance may result in passengers choosing competitors' flights that are more reliably on-time, even if the scheduled block time is longer. Careful schedule planning based on good historical data is obviously critical to successful airline operations.

1.4 Business Process Flow:

State of the art Computer Vision (CV) techniques can precisely recognize these ground actions. Action recognition is one of the active research areas in the CV with emerging real-world application such as, security surveillance, entertainment, autonomous driving vehicle, and video retrieval, etc. This recognizes various actions from series of observations on the actions of subjects and helps in prediction of future state by inferring the current action being performed.

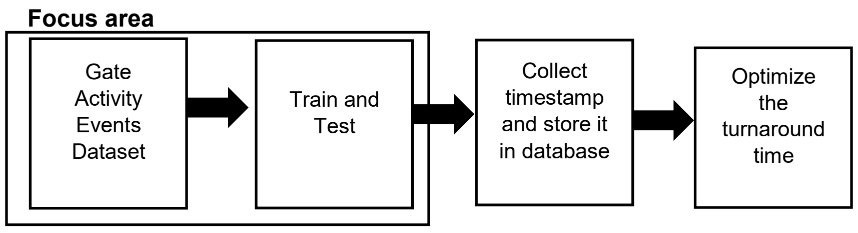

Focus area
Gate Activity Events Dataset
Train and Test
Collect timestamp and store it in database
Optimize the turnaround time

Fig 1.1: Flowchart of the overall Gate Activity System

In this project, we will be implementing the highlighted portion of the flowchart. This model output can be later used to calculate the time taken by each action is observed using the start and stop instances of the predicted action and these time values are used for analysis and optimization of scheduling operations in future. The display feature of historical data can be used to estimate their completion time and plan the activities more precisely and thereby, enhance passenger experience. With this model, occupancy rates and processing times can be tracked and optimized. Safety and security incidents can also be recorded for investigation and hindsight. By this way, we can improve the degree of complexity that can be handled by existing real-time surveillance systems.

Actions and behavior in a stream of images, videos, are being recognized in this problem. The spatiotemporal features of the videos are extracted using the image processing techniques. This preprocessing of video datasets is important for extracting crucial features to differentiate different activities with similar attributes/objects. After preprocessing, the models are trained with extracted features using the deep learning algorithms to preciously recognize the actions. The action recognition algorithms will be applied to the real-time surveillance data to recognize ground activities. The models will be created, trained and the preprocessing of dataset will be performed in Python environment. Refer to figure 1.2 for solution architecture. The performance of the models will be evaluated in terms of classification accuracy and confidence levels. Figure 1.3 represents the workflow to be followed.

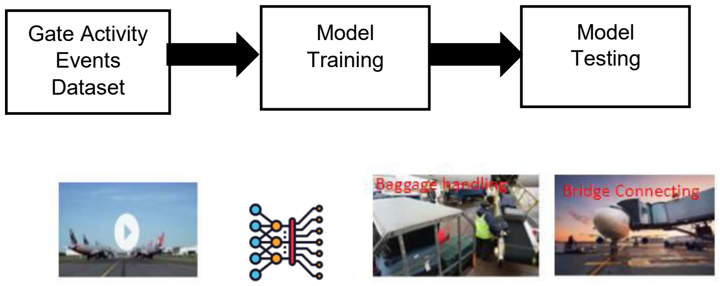


Fig 1.2: Proposed Architecture

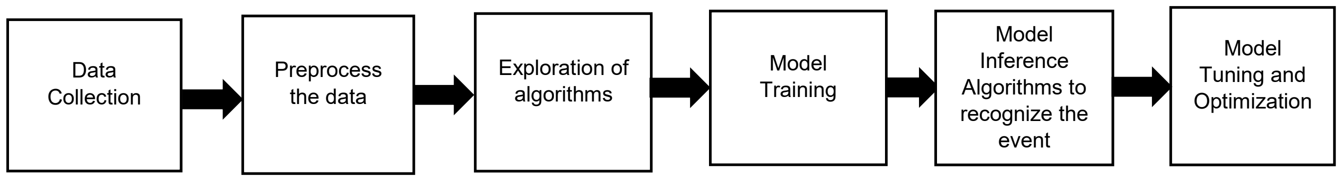

Data Collection
Preprocess the data
Exploration of algorithms
Model Training
Model Inference Algorithms to recognize the event
Model Tuning and Optimization

Fig 1.3: Representation of workflow

1.5 Structure of thesis

This section briefly summarizes the rest of the chapters, outlining the dissertation.

Chapter-2: Literature Survey

This chapter will describe the previous works done in this area. We'll first go through the explore the existing algorithms used for action recognition and their results.

Chapter-3: The Data

This chapter will describe the methods used for extracting features from the video and datasets that are used in this work.

Chapter-4: Experiments

This chapter will describe the model architectures developed as part of this work.

Chapter-5: Future Work

This chapter talks about the future scope to extend this work.

Chapter 2

Literature Survey

Action Recognition has been worked upon and studied well in recent years. This can be defined as the task of classifying or predicting the activity or action being performed. This basically deals with a time series classification problem where you need data from a series of timesteps to correctly classify the action being performed in the video [1]. By this way, it differs from normal classification task. Internally, it uses computer vision techniques to extract the features from the video.

In starting years, the methods used to find out interest points of an action were mostly done manually by looking at various feature points for various actions. Researchers have extracted fixed, hand-crafted features from video such as SIFT descriptors, HoG features, template matching, filter banks, etc [3]. These were then used with some machine learning model to classify the images. But in recent years, efforts have been made to detect those feature points automatically by a machine. Though it is difficult to extract perfect interest points from a computer, yet progress has been made towards it to reduce the manual work being done [2]. Although it worked relatively well, such features struggle to generalize across datasets and difficult to adapt to different domains. Additionally, hand-crafted features tend to make simplifying assumptions about the underlying data distribution that degrade performance in real-world applications [5]. Thus, researchers in this domain began to look for alternative methodologies that can be learned directly from data. The ability of CNNs to extract useful features from images created a quick way to learn features. This became a primary area of focus [4].

One of the early methods proposed to provide solution for AR was to process frames from a video separately and do averaging to get results. Although this method is good when one wants to detect an action from one image, but for videos this method is not a good idea since we are ignoring the time-based changes in the actions [6]. This method was based on CNN which basically classifies various images based on the features of the image. This model will only try to learn the environmental context with the help of multiple frames. Hence, the model won't be fully confident about each video frame's prediction. The results will change rapidly and fluctuate. This is because the model is not looking at the entire video sequence but just classifying each frame independently [7].

To improve this model, researchers tried to solve this by moving average model where instead of classifying and displaying results for a single frame, they have taken the average results over n frames [10]. This would effectively get rid of that flickering problem. The value of n needs to be decided in an optimal manner. The drawback of the moving average model was the model learnt only the environmental context instead of the actual action sequence to predict the action. It considers a video just a group of multiple images [8]. This method terribly failed and resulted in over fitting problem. This approach won't work well when the actions are similar. For example, consider the action of standing up from a chair and sitting down on a chair. In both actions, the frames are almost same. The main differentiator is the order of the frame sequence. So, we require temporal information to correctly predict these actions [13].

There was another methodology which was introduced to address the problem of action recognition. This uses spatio-temporal features of video. Space-time approaches recognize activities based on space-time features or on trajectory matching [11]. Basically, there is 3D space-time where the activity occurs which consists of sequence of 2D spaces in time. An activity is associated with a set of space-time feature trajectories. The action being performed is dependent on the sequence of activities performed at various time intervals [12]. This method is based on RNN. We basically try to train a model based on features at various time stamps. This ensures that we consider the sequence of action being performed. Essentially, we mean that action is comprised of a sequence of poses/gestures, and we try to learn that sequence also instead of just learning the poses separately [15]. The final model that we have built is based on CNN+LSTM. In particular, the video was first segmented into smaller groups of frames which are passed into CNN to extract features. However, instead of performing classification on each of these features independently, the features associated with each video segment are passed, in temporal order, as input to an LSTM, which then performs the final classification [14]. Such a methodology was shown to consider the context of the full video and, as a result, achieve significantly improved performance on HAR.

We need to have a clear idea of what type of action we want to recognize. There are three major families of techniques in the literature [17]:

- **Symbolic techniques:** This comes from AI community like temporal reasoning under constraints. This technique doesn't deal well with uncertainty.
- **Probabilistic techniques:** This includes Hidden Markov Models (HMM) or Bayesian Networks. Here the entity to be recognized may be seen as a stochastically predictable sequence of state. This technique requires a large quantity of video samples to build reliable models.
- **Data mining techniques:** This includes association rules mining which is used to discover unknown actions. Here the events are determined by modelling an activity using rules of attributes that describe that event. Each activity can be considered as a set of primitive rules.

Making the right choice of technique depends on the problem statement and on the quality or size of the dataset.

Recently there has been a huge growth in computer vision research about AR. Concluding, we can say that there is huge research being done for AR in the field of CV and in future we may develop a system where we can have much higher accuracy in our predictions.

Chapter 3

The Data

Action Recognition has been worked upon and studied well in recent years. This can be defined as the task of classifying or predicting the activity or action being performed. This basically deals with a time series classification problem where you need data from a series of timesteps to correctly classify the action being performed in the video. By this way, it differs from normal classification task. Internally, it uses computer vision techniques to extract the features from the video. The scope of the project is limited to recognizing only two ground actions due to the time constraints. To train the model, we require a high-end machine as a hardware resource. We have decided to collect “Bridge Connecting”, “Baggage Handling” and “miscellaneous” as three categories to be recognized by the model. This decision was made after discussing with the leads from my company as these are widely available in YouTube videos. We have planned to explore various algorithms used in Action Recognition and choose the best which suits our problem statement.

3.1 Data Collection

A video is simply a collection of temporally ordered images, called frames. When viewed in the correct temporal order, these frames reveal the movement of a scene through time, forming a video. Like images, each of these frames will typically be represented in RGB-format and have the same spatial resolution.

Procuring the data sets has been a challenging task as there were limited dataset available on the internet. The dataset was collected from YouTube having only one action category. After downloading, we have performed trimming operation to focus only on the actions we required. We have collected around 130 videos belonging to the Jet Bridge Connecting and Baggage Handling class. This gives the diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination condition, etc.

The Dataset contains:

Action Categories	3
Average Videos per Action Category	130
Average Number of Frames per Video	812.85
Average Frames Width per Video	1154.58
Average Frames Height per Video	713.00
Average Frames per Seconds per Video	27.37

Table 3.1: Statistics analysis of dataset

Most of the available dataset are realistic. Video datasets are often orders of magnitude larger than image dataset. This increases the storage and computational requirements for training. The next sections will describe about the data preprocessing and exploratory data analysis.

3.2 Data Preprocessing

Trimming operation was performed on all videos post the data collection. Typically, the videos within the dataset are focused upon a single entity that is performing the action in question, and the video does not extend far before or after the action is performed. The datasets contain several variable-length videos that are each associated with a semantic label, corresponding to the action being performed in that video. A categorical label must be assigned to a video clip based upon the action being performed in the clip. We have followed a proper naming convention for all videos. For single Frame CNN, ConvLSTM, and LRCN model, the trimmed videos are provided as input directly to the model. For two stream model, the RGB frames and optical frames are extracted from each video and then provided as input to the model.

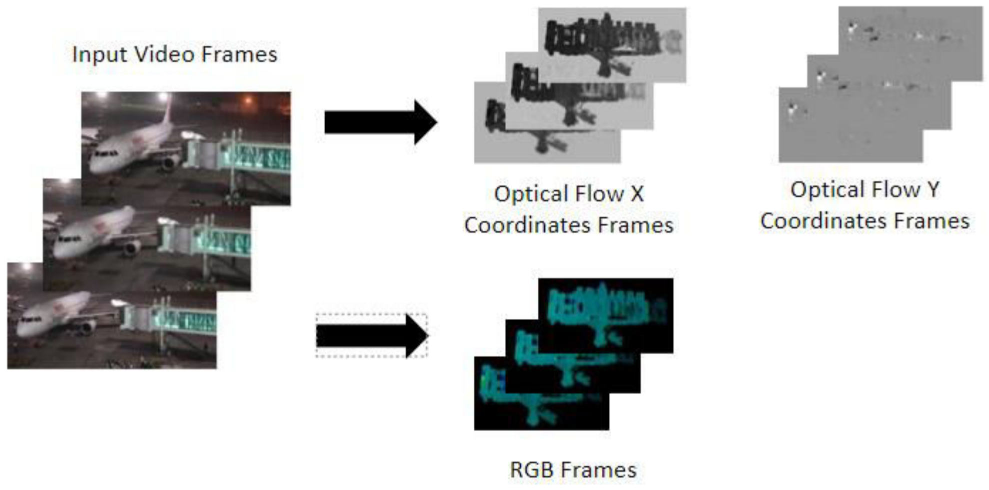

Input Video Frames
Optical Flow X Coordinates Frames
Optical Flow Y Coordinates Frames
RGB Frames

Fig: 3.1 Extraction of RGB and Optical flow frames

3.3 Exploratory Data Analysis

We have visualized the data along with labels to get an idea about what classes we are dealing with. For visualization, we picked random videos from each category and visualize the first frame of the selected videos with their associated labels written. This way, we will be able to visualize a subset of the dataset which we are going to deal with.

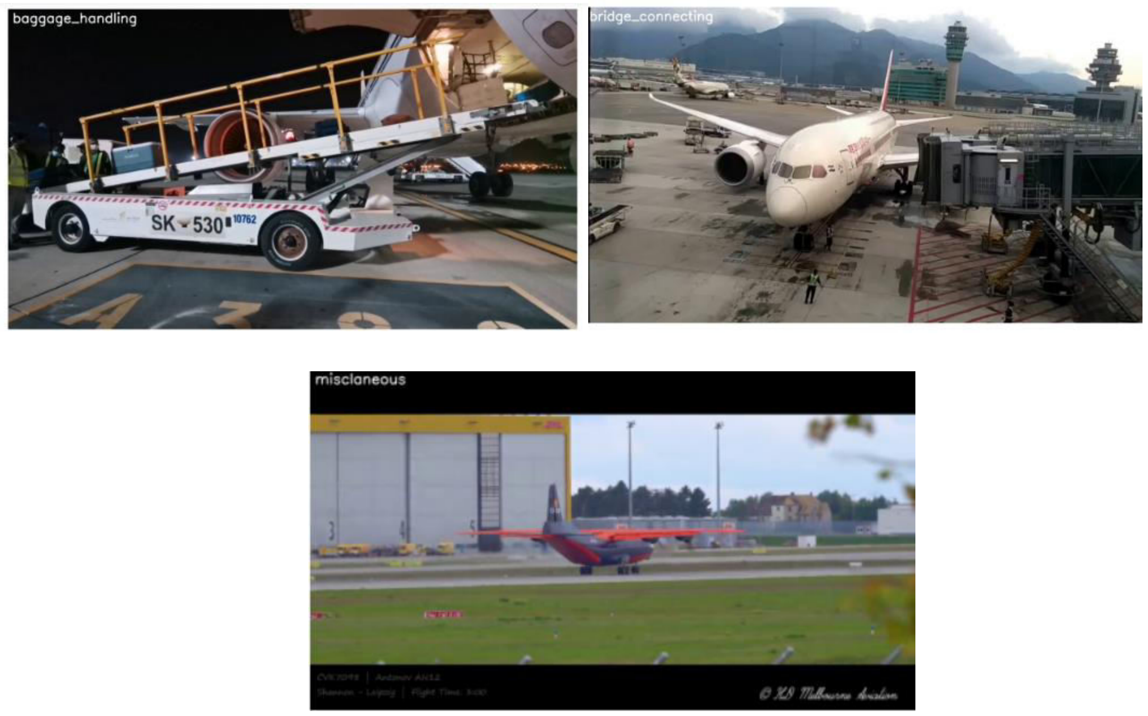


Fig 3.2: Visualization of random video from each category

We have iterated through all the videos collected in both categories and extracted information like frame per second, height, width, duration, and frame count. This data has been saved to a csv file for data exploration. Below figure shows the description of the DataFrame.

	fps	height	width	duration	frame_count
count	390.000000	390.00000	390.000000	390.000000	390.000000
mean	27.373513	713.00000	1154.589744	29.735897	812.858974
std	3.062411	146.78146	278.504177	26.986800	753.435353
min	6.000000	320.00000	360.000000	1.000000	28.000000
25%	25.000000	720.00000	1280.000000	8.000000	240.000000
50%	29.725000	720.00000	1280.000000	23.000000	616.500000
75%	30.000000	720.00000	1280.000000	44.000000	1145.500000
max	30.010000	1080.00000	1280.000000	159.000000	4772.000000

Table 3.2: Statistic description of the dataset

The describe method of Panda library returns description of the data in the DataFrame. This contains information for each column – count, mean, standard deviation, the minimum value, 25% percentile, 50% percentile, 75% percentile and the maximum value.

Similarly, we have created separate csv file for each category. Below figure shows the description of the DataFrame for each category.

	fps	height	width	duration	frame_count
count	130.000000	130.000000	130.000000	130.000000	130.000000
mean	28.242538	740.538462	1065.153846	38.561538	1118.284615
Std	3.727888	198.935117	320.593719	34.527375	1016.560146
Min	6.000000	320.000000	480.000000	1.000000	28.000000
25%	25.000000	720.000000	615.500000	9.250000	271.000000
50%	29.970000	720.000000	1280.000000	28.000000	844.000000
75%	30.000000	720.000000	1280.000000	57.750000	1711.750000
Max	30.010000	1080.000000	1280.000000	159.000000	4772.000000

(a)

	fps	height	width	duration	frame_count
count	130.000000	130.000000	130.000000	130.000000	130.000000
mean	28.839769	678.461538	1118.615385	7.569231	230.930769
std	2.402058	152.896893	325.424836	3.501801	108.700370
min	15.030000	360.000000	360.000000	1.000000	30.000000
25%	29.970000	720.000000	1280.000000	5.000000	151.000000
50%	30.000000	720.000000	1280.000000	7.000000	216.500000
75%	30.000000	720.000000	1280.000000	10.000000	300.750000
max	30.000000	1080.000000	1280.000000	15.000000	451.000000

(b)

	fps	height	width	duration	frame_count
count	130.000000	130.0	130.0	130.000000	130.000000
mean	25.038231	720.0	1280.0	43.076923	1089.361538
std	0.435898	0.0	0.0	15.450654	392.766278
min	25.000000	720.0	1280.0	12.000000	301.000000
25%	25.000000	720.0	1280.0	33.000000	843.250000
50%	25.000000	720.0	1280.0	40.000000	1012.000000
75%	25.000000	720.0	1280.0	49.000000	1241.750000
max	29.970000	720.0	1280.0	109.000000	2736.000000

(c)

Table 3.3: Statistic description of the Jet Bridge Connecting(a), Baggage Handling(b) and miscellaneous(c) dataset

We have also visualized the duration distribution for each category. You can observe that most of the jet bridge connecting dataset has duration distribution around 10 to 60 seconds. For baggage handling dataset has duration distribution around 5 to 10 seconds and for miscellaneous handling dataset had duration distribution around 35 to 55 seconds.

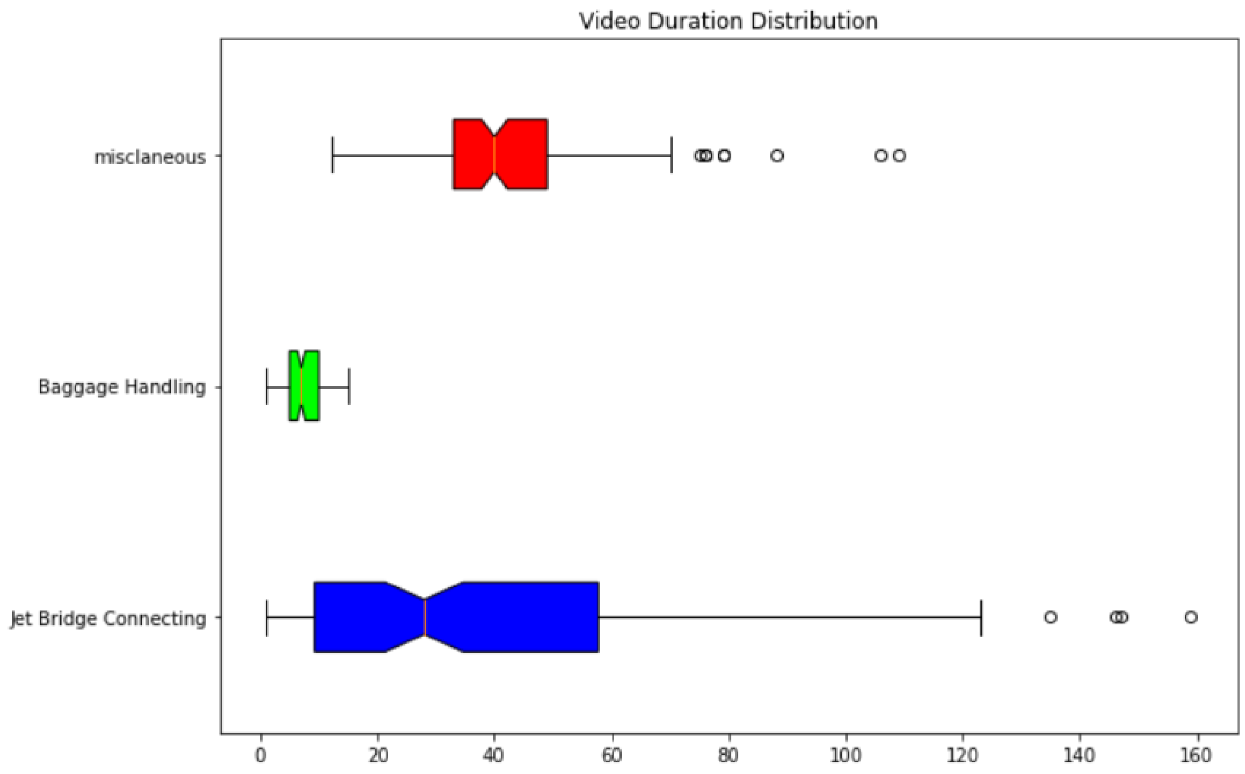


Fig 3.3: Duration Distribution of the dataset

3.4 Augmentation

To increase the dataset, we can perform data augmentation. For this, we have extract frames from each video and applied image augmentation technique with the help of albumentations library. This is a fast and flexible image augmentation library being widely used. We have applied effects like scale rotate, horizontal flip, blur, optical distortion, and saturation to each video. We have investigated the videos count in other well-known dataset for Action Recognition. We observed that those datasets like ActivityNet, UCF101 and HMDB51 has an average of 100 videos for each action. So, we have decided not to include the augmented dataset for training purpose as we had a good amount of dataset for training.

Chapter 4

Experiments

4.1 Exploring Model Architectures

The goal of the underlying model is to predict the semantic action label given the video as input.

Single-Frame CNN Network:

This network uses single architecture that fuses information from all frames at the last stage. It works by running an image classification model on every single frame of the video and then average all the individual probabilities to get the final probabilities vector [18].

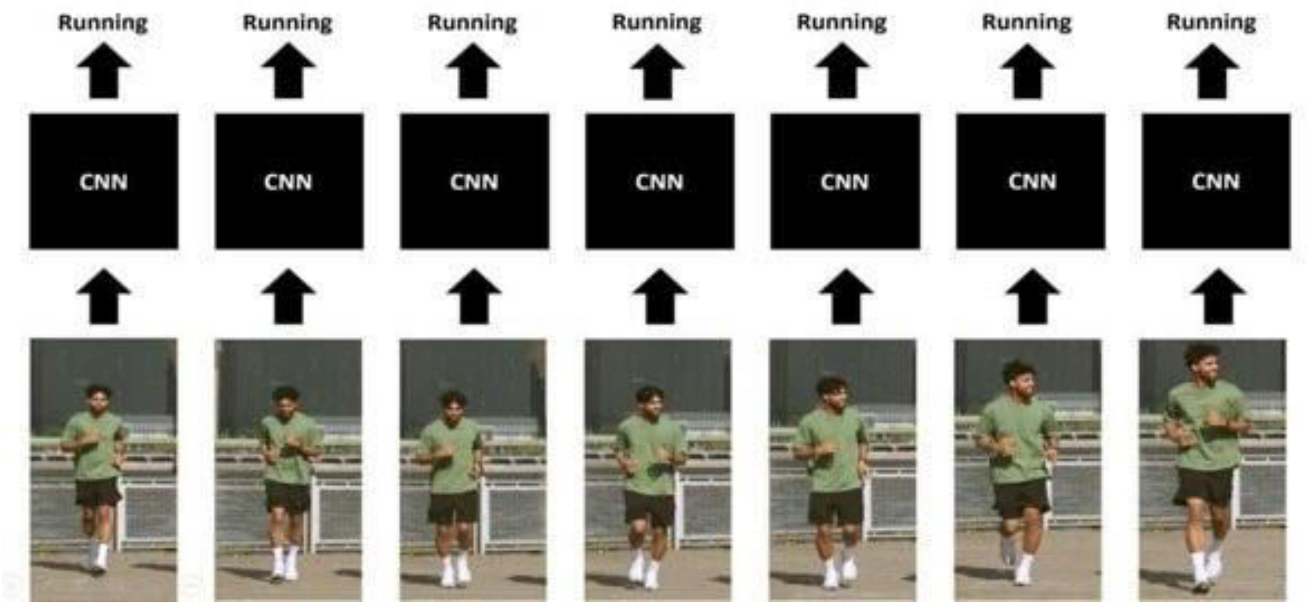
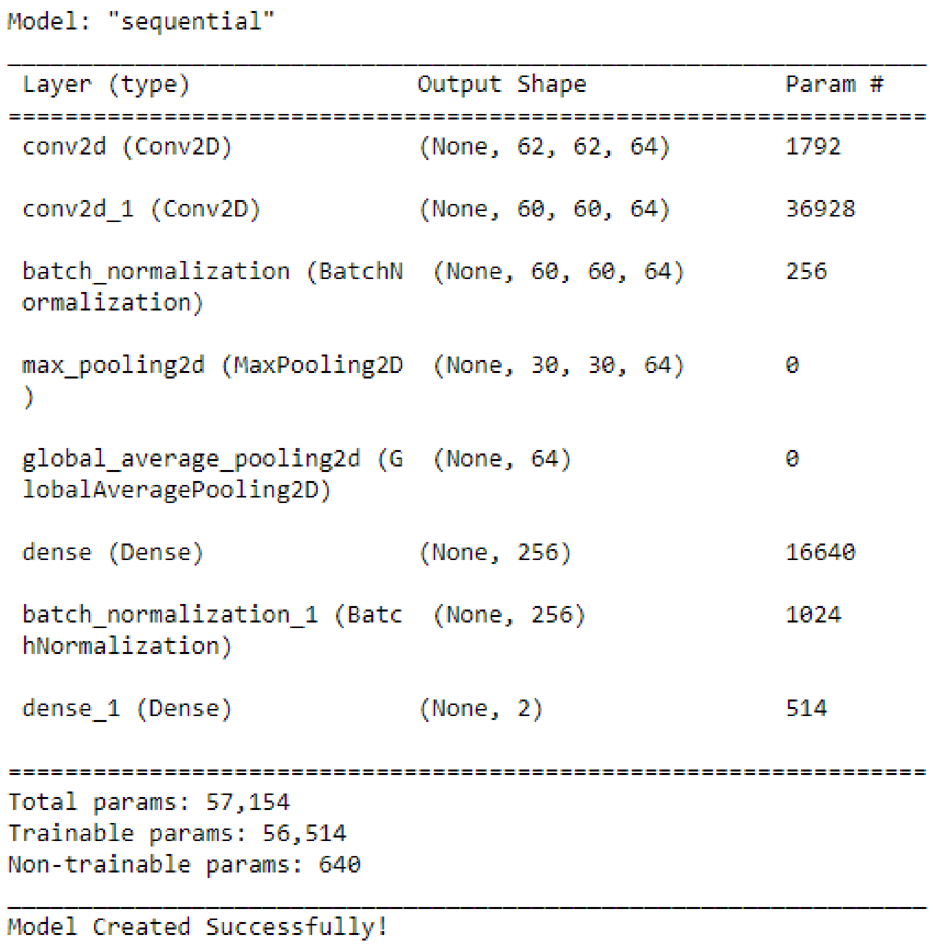


Fig 4.1: Representation of Single Frame CNN Network

We are creating a simple CNN classification model with two CNN layers defined by Conv2D function with 64 filters with kernel size as 3x3 and ReLU as activation layer. Followed by this, Batch Normalization is added to perform normalization technique between the layers of neural network. This is done along mini batches instead of full dataset. It serves to speed up the training and use higher learning rates, making learning easier. It provides some regularization and reduces generalization error. Then, Max pooling layer is added to perform pooling operation that calculates the maximum value for patches of a feature map and uses it to create a down sampled feature map. It is usually used after a convolutional layer. This highlights the most present feature in the patch and reduces the spatial size of the representation by reducing the number of parameters and computation in the network. Internally, it reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer.

Global Average Pooling is added to replace full connected layers in classical CNNs. The idea is to generate one feature map for each corresponding category of the classification task. The 2D Global average pooling block takes a tensor of size (input width) x (input height) x (input channels) and computes the average value of all values across the entire (input width) x (input height) matrix for each of the (input channels). ReLU sets all negative values in the matrix x to zero and all other values are kept constant. ReLU is computed after the convolution and is a nonlinear activation function like tanh or sigmoid. More computationally efficient to compute than Sigmoid. Just needs to pick max (0, x) and not perform expensive exponential operations. ReLU tend to show better convergence performance than sigmoid. Softmax activation function in the final output layer when you are trying to solve the Classification problems where your labels are class values. It predicts a multinomial probability distribution for multi-class classification problems. It is a type of squashing function which limits the output of the function into the range 0 to 1.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 64)        1792      
                                                                 
 conv2d_1 (Conv2D)           (None, 60, 60, 64)        36928     
                                                                 
 batch_normalization (BatchN  (None, 60, 60, 64)       256       
 ormalization)                                                   
                                                                 
 max_pooling2d (MaxPooling2D  (None, 30, 30, 64)       0         
 )                                                               
                                                                 
 global_average_pooling2d (G  (None, 64)               0         
 lobalAveragePooling2D)                                          
                                                                 
 dense (Dense)               (None, 256)               16640     
                                                                 
 batch_normalization_1 (Batc  (None, 256)              1024      
 hNormalization)                                                 
                                                                 
 dense_1 (Dense)             (None, 2)                 514       
                                                                 
=================================================================
Total params: 57,154
Trainable params: 56,514
Non-trainable params: 640
_________________________________________________________________
Model Created Successfully!
```

Fig 4.2: Architecture of Single Frame CNN Network

CNN + LSTM Approach:

CNN are great for image data and LSTM network are great when working with sequence data. When we combine both, we get to solve computer vision problems like video classification. This approach uses CNN to extract spatial features at a given time step in the input sequence and then an LSTM to identify temporal relations between frames.

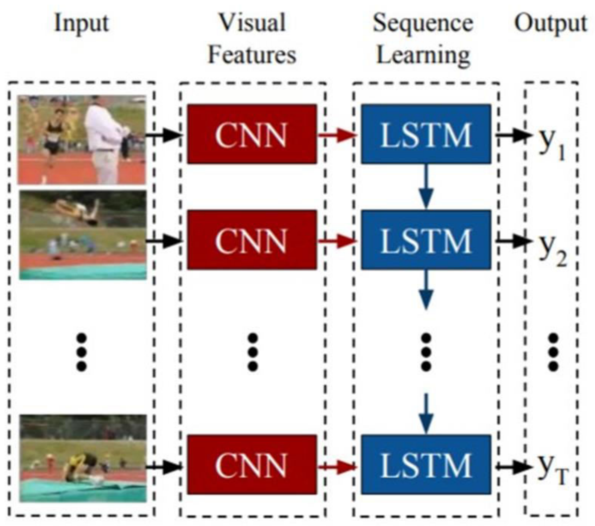


Fig 4.3: Representation of CNN+LSTM approach

We will be experimenting two architecture that will be using CNN along with LSTM. They are

1. ConvLSTM
2. RCN

Both approaches can be implemented using TensorFlow.

ConvLSTM Network:

This is implemented by using a combination of ConvLSTM cells. A ConvLSTM cell is a variant of an LSTM network. It has got convolution embedded in the architecture, which makes it capable of identifying spatial features of the data while keeping into account the temporal relation [19].

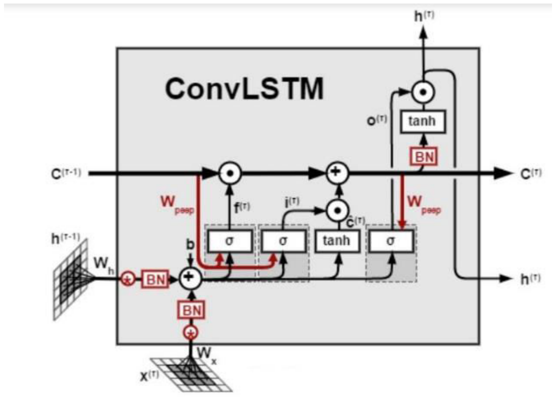
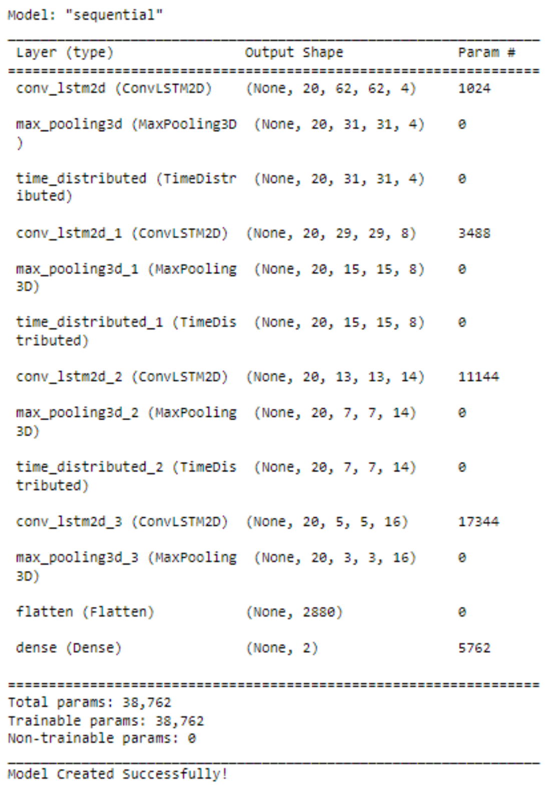


Fig 4.4: Representation of ConvLSTM cell

This approach works effectively for video classification problem as it captures spatial relation in the individual frames and the temporal relations across the different frames. This ConvLSTM's convolution structure can take 3D input (width, height, num_of_channels) whereas a simple LSTM only takes 1D input hence an LSTM is incompatible for modeling spatio-temporal data on its own.

Model Architecture of ConvLSTM Model:

We use Keras convLSTM2D recurrent layers to construct the model. This convLSTM2D layer takes number of filters and kernel size as input for performing convolutional operations. We use MaxPooling3D layers to reduce the dimensions of the frames and avoid unnecessary computations. The dropout layers are added at each layer to prevent the overfitting problem. At the end, the output layer is flattened and fed to the Dense layer with softmax as an activation which outputs the probability of each action category. The architecture is a simple one and has a small number of trainable parameters.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_lstm2d (ConvLSTM2D)    (None, 20, 62, 62, 4)     1024

 max_pooling3d (MaxPooling3D  (None, 20, 31, 31, 4)    0
 )

 time_distributed (TimeDistr  (None, 20, 31, 31, 4)    0
 ibuted)

 conv_lstm2d_1 (ConvLSTM2D)  (None, 20, 29, 29, 8)     3488

 max_pooling3d_1 (MaxPooling  (None, 20, 15, 15, 8)    0
 3D)

 time_distributed_1 (TimeDis  (None, 20, 15, 15, 8)    0
 tributed)

 conv_lstm2d_2 (ConvLSTM2D)  (None, 20, 13, 13, 14)    11144

 max_pooling3d_2 (MaxPooling  (None, 20, 7, 7, 14)     0
 3D)

 time_distributed_2 (TimeDis  (None, 20, 7, 7, 14)     0
 tributed)

 conv_lstm2d_3 (ConvLSTM2D)  (None, 20, 5, 5, 16)      17344

 max_pooling3d_3 (MaxPooling  (None, 20, 3, 3, 16)     0
 3D)

 flatten (Flatten)           (None, 2880)              0

 dense (Dense)               (None, 2)                 5762

=================================================================
Total params: 38,762
Trainable params: 38,762
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!
```

Fig 4.5: Architecture of ConvLSTM Network

LRCN Network:

In this network, we implement Long-term Recurrent Convolutional Network (LRCN), which combines CNN and LSTM layers in a single model. The CNN model can be used to extract spatial features from the frames in the video and LSTM model can then use the spatial features extracted by CNN at each time-steps for temporal sequence modeling [20]. This way the network learns spatiotemporal features directly in an end-to-end training, resulting in a robust model.

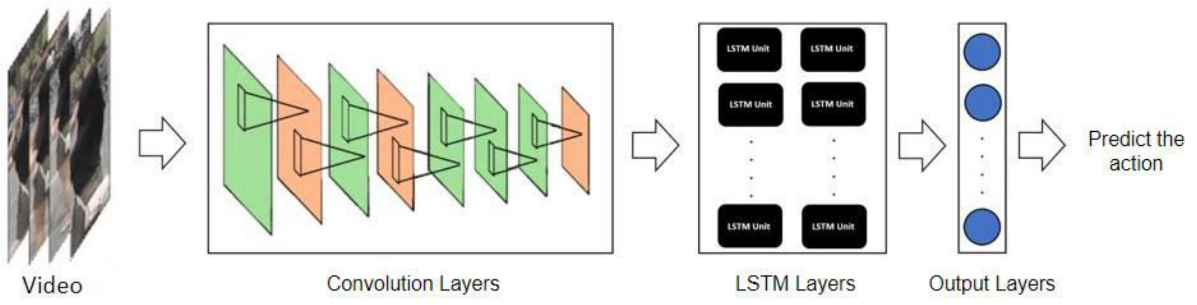
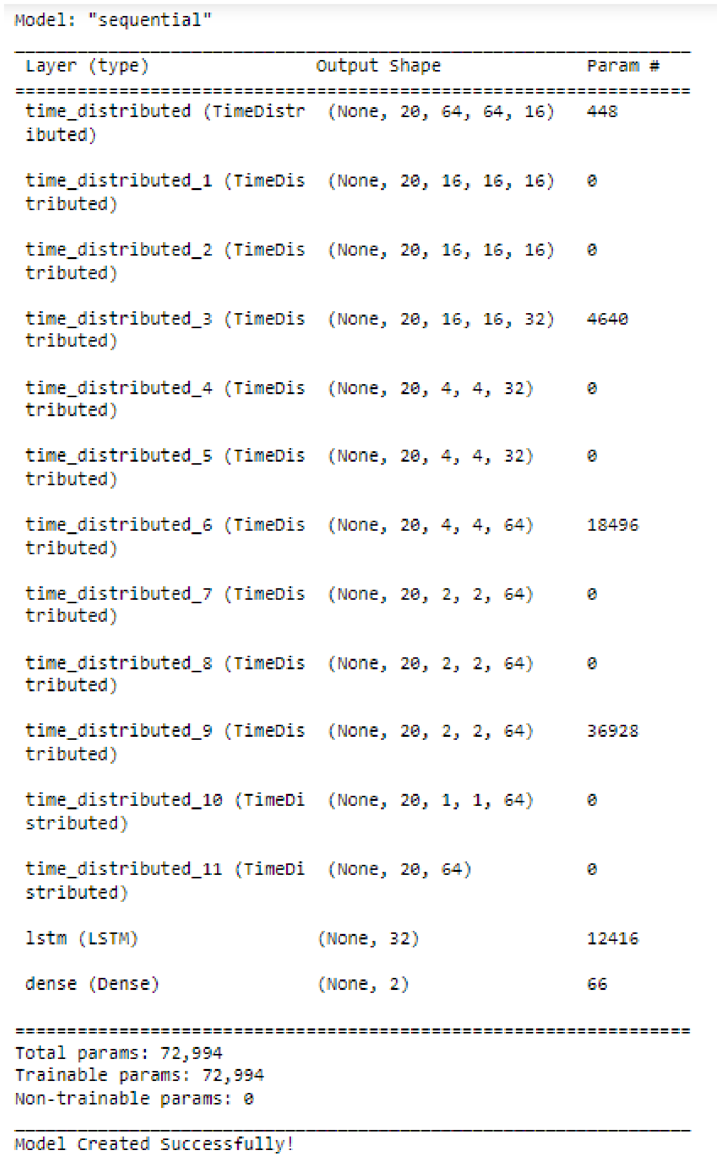


Fig 4.6: Representation of LRCN Network

Model Architecture of LRCN Model:

To implement this architecture, we use time-distributed conv2D layers, which allows applying the same layer to every frame of the video independently. This layer can take input of shape (no_of_frames, width, height, num_of_channels). This is very beneficial as it allows to input the whole video into the model in a single shot. This layer is followed by MaxPooling2D and Dropout layers. The spatial features extracted from the Conv2D layers will be flattened using Flatten layer. This flatten output is then passed onto LSTM layer. The output of LSTM layer is fed into the Dense layer with softmax activation which will be used to predict the action being performed.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDistr  (None, 20, 64, 64, 16)   448
 ibuted)

 time_distributed_1 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_2 (TimeDis  (None, 20, 16, 16, 16)   0
 tributed)

 time_distributed_3 (TimeDis  (None, 20, 16, 16, 32)   4640
 tributed)

 time_distributed_4 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_5 (TimeDis  (None, 20, 4, 4, 32)     0
 tributed)

 time_distributed_6 (TimeDis  (None, 20, 4, 4, 64)     18496
 tributed)

 time_distributed_7 (TimeDis  (None, 20, 2, 2, 64)     0
 tributed)

 time_distributed_8 (TimeDis  (None, 20, 2, 2, 64)     0
 tributed)

 time_distributed_9 (TimeDis  (None, 20, 2, 2, 64)     36928
 tributed)

 time_distributed_10 (TimeDi  (None, 20, 1, 1, 64)     0
 stributed)

 time_distributed_11 (TimeDi  (None, 20, 64)           0
 stributed)

 lstm (LSTM)                 (None, 32)                12416

 dense (Dense)               (None, 2)                 66

=================================================================
Total params: 72,994
Trainable params: 72,994
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully!
```

Fig 4.7: Architecture of LRCN Network

Two Stream Network:

This network is composed of two separate convolution neural networks. One to handle spatial features and one to handle temporal or motion features. The input to the two-stream architecture only contains a single frame for the spatial stream and a fixed-size group of optical flow maps for the temporal stream. The output of these separate network components can be combined to form a spatiotemporal video representation. Long term dependency is modeled by combining temporal network output across time frames.

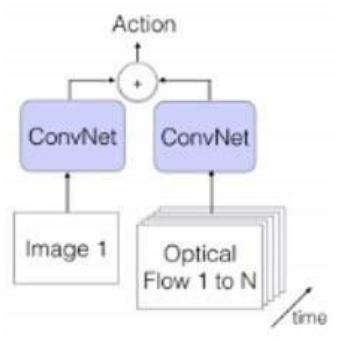
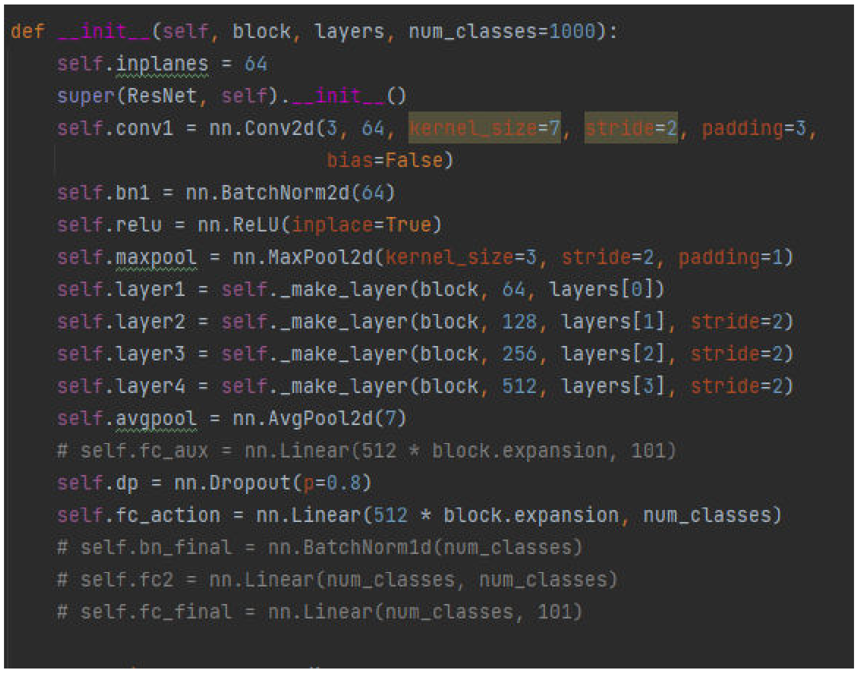


Fig 4.8: Representation of Two Stream Network

Here, we use Resnet CNN model. Below picture shows the model architecture.



```
def __init__(self, block, layers, num_classes=1000):
    self.inplanes = 64
    super(ResNet, self).__init__()
    self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3,
                           bias=False)
    self.bn1 = nn.BatchNorm2d(64)
    self.relu = nn.ReLU(inplace=True)
    self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
    self.layer1 = self._make_layer(block, 64, layers[0])
    self.layer2 = self._make_layer(block, 128, layers[1], stride=2)
    self.layer3 = self._make_layer(block, 256, layers[2], stride=2)
    self.layer4 = self._make_layer(block, 512, layers[3], stride=2)
    self.avgpool = nn.AvgPool2d(7)
    # self.fc_aux = nn.Linear(512 * block.expansion, 101)
    self.dp = nn.Dropout(p=0.8)
    self.fc_action = nn.Linear(512 * block.expansion, num_classes)
    # self.bn_final = nn.BatchNorm1d(num_classes)
    # self.fc2 = nn.Linear(num_classes, num_classes)
    # self.fc_final = nn.Linear(num_classes, 101)
```

Fig 4.9: Architecture of Two Stream Network

Chapter 5

Conclusions

Single Frame CNN model averages all the individual probabilities to get the final probabilities vector. Even though the accuracy of this model is good, it only try to learn the environmental context with the help of multiple frames. Hence, the model won't be fully confident about each video frame's prediction. The results will change rapidly and fluctuate. This is because the model is not looking at the entire video sequence but just classifying each frame independently. The most effective action recognition needs the combination of spatial and temporal information. Image-based models that operate independently on frames in a video perform well on HAR, revealing that only minimal temporal reasoning is required to provide a reasonable solution. CNN are great for image data and LSTM network are great when working with sequence data. When we combine both, we get to solve computer vision problems like video classification. This approach uses CNN to extract spatial features at a given time step in the input sequence and then an LSTM to identify temporal relations between frames. The accuracy of ConvLSTM and LRCN model is 80.85 and 84.52 respectively. Two Stream network is composed of two separate convolution neural networks. One to handle spatial features and one to handle temporal features. We are getting similar accuracy for both streams. Long term dependency is modeled by combining temporal network output across time frames. From the below table, we can conclude that LRCN model works well compared to other models.

Model Name	**Accuracy**
Single Frame CNN	97.87
ConvLSTM	80.85
LRCN	84.52
Two Stream	63.402 for RGB frames & 63.402 for Optical flow

Table 5.1: Comparison of models

Chapter 6

Future Work

This model output can be later used to calculate the time taken by each action is observed using the start and stop instances of the predicted action and these time values are used for analysis and optimization of scheduling operations in future. The display feature of historical data can be used to estimate their completion time and plan the activities more precisely and thereby, enhance passenger experience. With this model, occupancy rates and processing times can be tracked and optimized. Safety and security incidents can also be recorded for investigation and hindsight. By this way, we can improve the degree of complexity that can be handled by existing real-time surveillance systems.

Bibliography

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential Deep Learning for Human Action Recognition, pages 29–39. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[2] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. arXiv preprint arXiv:1511.06432, 2015.

[3] Caffe2-Team. Caffe2: A new lightweight, modular, and scalable deep learning framework. https://caffe2.ai/.

[4] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In CVPR, 2017.

[5] N. Dalal, B. Triggs, and C. Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. In A. Leonardis, H. Bischof, and A. Pinz, editors, European Conference on Computer Vision (ECCV ’06), volume 3952 of Lecture Notes in Computer Science (LNCS), pages 428– 441, Graz, Austria, May 2006. Springer-Verlag.

[6] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In Proc. ICCV VS-PETS, 2005.

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2625–2634, 2015.

[8] G. Farneback. Two-frame motion estimation based on poly- ¨ nomial expansion. In Image Analysis, 13th Scandinavian Conference, SCIA 2003, Halmstad, Sweden, June 29 - July 2, 2003, Proceedings, pages 363–370, 2003.

[9] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal residual networks for video action recognition. In NIPS, 2016. 2, 7, 8 [10] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In CVPR, 2016.

[11] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. C. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In CVPR, 2017.

[12] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, ´ L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: training imagenet in 1 hour. arXiv preprint arXiv:1706.02677, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[14] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. In CVPR, 2017.

[15] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. IEEE TPAMI, 35(1):221–231, 2013.

[16] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In CVPR, 2014.

[17] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. CoRR, abs/1705.06950, 2017.

[18] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal ¨ descriptor based on 3d-gradients. In BMVC, 2008.

[19] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012. 1, 2

[20] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In ICCV, 2011.

Appendix A

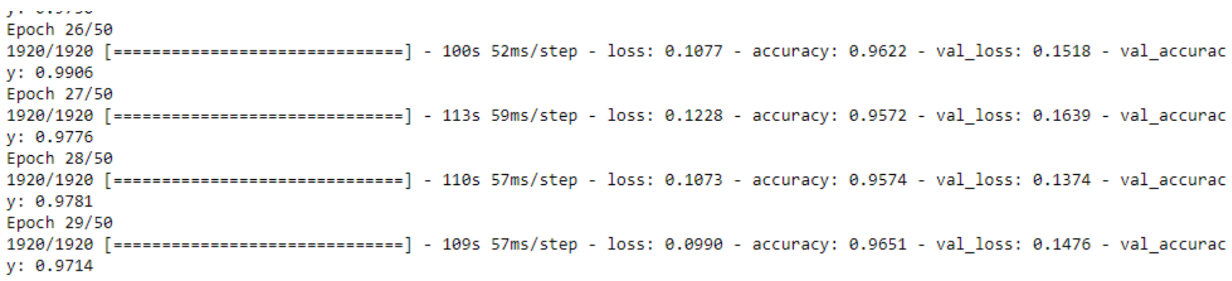
Experimental Results

The goal of the underlying model is to predict the semantic action label given the video as input.

Single Frame CNN Model:

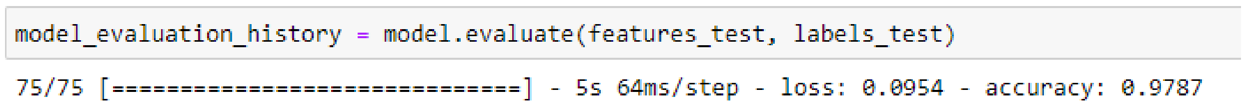
Hyperparameters	Value
Epoch	50
Batch size	4
Loss	categorical_crossentropy
Optimizer	Adam

Table A.1: Hyperparameters for Single Frame CNN model



```
Epoch 26/50
1920/1920 [==============================] - 100s 52ms/step - loss: 0.1077 - accuracy: 0.9622 - val_loss: 0.1518 - val_accurac
y: 0.9906
Epoch 27/50
1920/1920 [==============================] - 113s 59ms/step - loss: 0.1228 - accuracy: 0.9572 - val_loss: 0.1639 - val_accurac
y: 0.9776
Epoch 28/50
1920/1920 [==============================] - 110s 57ms/step - loss: 0.1073 - accuracy: 0.9574 - val_loss: 0.1374 - val_accurac
y: 0.9781
Epoch 29/50
1920/1920 [==============================] - 109s 57ms/step - loss: 0.0990 - accuracy: 0.9651 - val_loss: 0.1476 - val_accurac
y: 0.9714
```

Fig A.1: Training of Single Frame CNN Model



```
model_evaluation_history = model.evaluate(features_test, labels_test)
75/75 [==============================] - 5s 64ms/step - loss: 0.0954 - accuracy: 0.9787
```

Fig A.2: Evaluation of Single Frame CNN Model

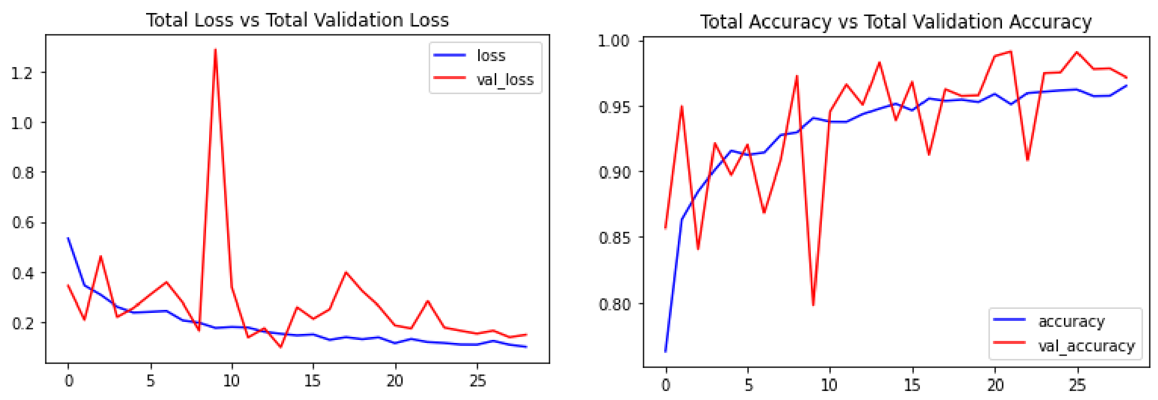


Fig A.3: Loss and Accuracy for Single Frame CNN Model

When we give bridge connecting video as input, we see the below result:

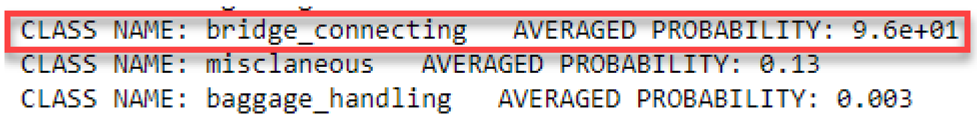

```
CLASS NAME: bridge_connecting   AVERAGED PROBABILITY: 9.6e+01
CLASS NAME: miscleaneous   AVERAGED PROBABILITY: 0.13
CLASS NAME: baggage_handling   AVERAGED PROBABILITY: 0.003
```

Fig A.4: Prediction of Bridge Connecting action using Single Frame CNN Model

When we give baggage handling video as input, we see the below result:

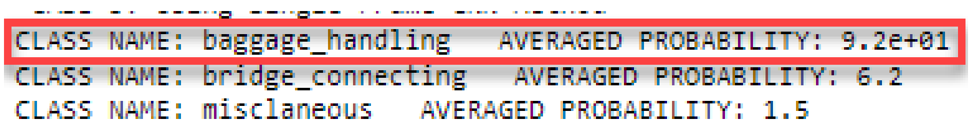

```
CLASS NAME: baggage_handling   AVERAGED PROBABILITY: 9.2e+01
CLASS NAME: bridge_connecting   AVERAGED PROBABILITY: 6.2
CLASS NAME: miscleaneous   AVERAGED PROBABILITY: 1.5
```

Fig A.5: Prediction of Baggage Handling action using Single Frame CNN Model

When we give miscellaneous video as input, we see the below result:

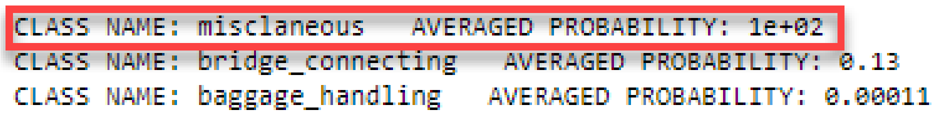
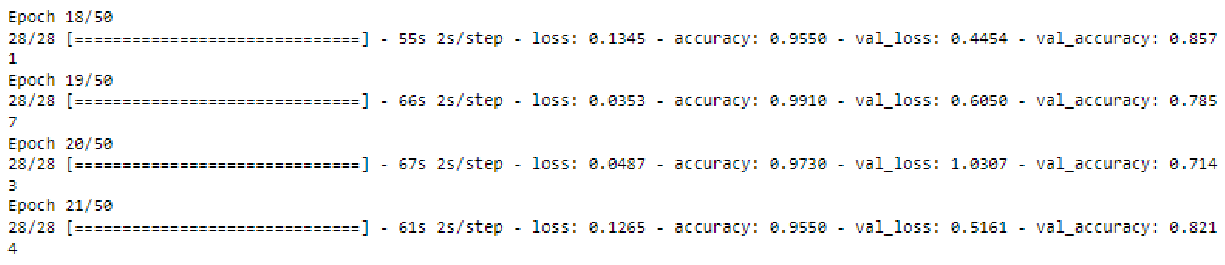

```
CLASS NAME: miscleaneous   AVERAGED PROBABILITY: 1e+02
CLASS NAME: bridge_connecting   AVERAGED PROBABILITY: 0.13
CLASS NAME: baggage_handling   AVERAGED PROBABILITY: 0.00011
```

Fig A.6: Prediction of miscellaneous action using Single Frame CNN Model

ConvLSTM Model:

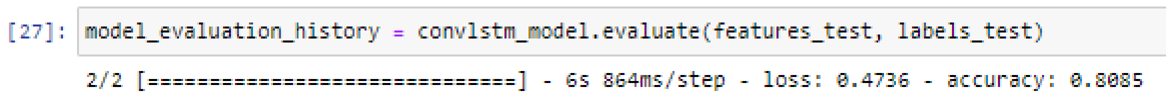
Hyperparameters	Value
Epoch	50
Batch size	4
Loss	categorical_crossentropy
Optimizer	Adam

Table A.2: Hyperparameters for ConvLSTM model



```
Epoch 18/50
28/28 [==============================] - 55s 2s/step - loss: 0.1345 - accuracy: 0.9550 - val_loss: 0.4454 - val_accuracy: 0.857
1
Epoch 19/50
28/28 [==============================] - 66s 2s/step - loss: 0.0353 - accuracy: 0.9910 - val_loss: 0.6050 - val_accuracy: 0.785
7
Epoch 20/50
28/28 [==============================] - 67s 2s/step - loss: 0.0487 - accuracy: 0.9730 - val_loss: 1.0307 - val_accuracy: 0.714
3
Epoch 21/50
28/28 [==============================] - 61s 2s/step - loss: 0.1265 - accuracy: 0.9550 - val_loss: 0.5161 - val_accuracy: 0.821
4
```

Fig A.7: Training of ConvLSTM Model



```
[27]: model_evaluation_history = convlstm_model.evaluate(features_test, labels_test)

      2/2 [==============================] - 6s 864ms/step - loss: 0.4736 - accuracy: 0.8085
```

Fig A.8: Evaluation of ConvLSTM Model

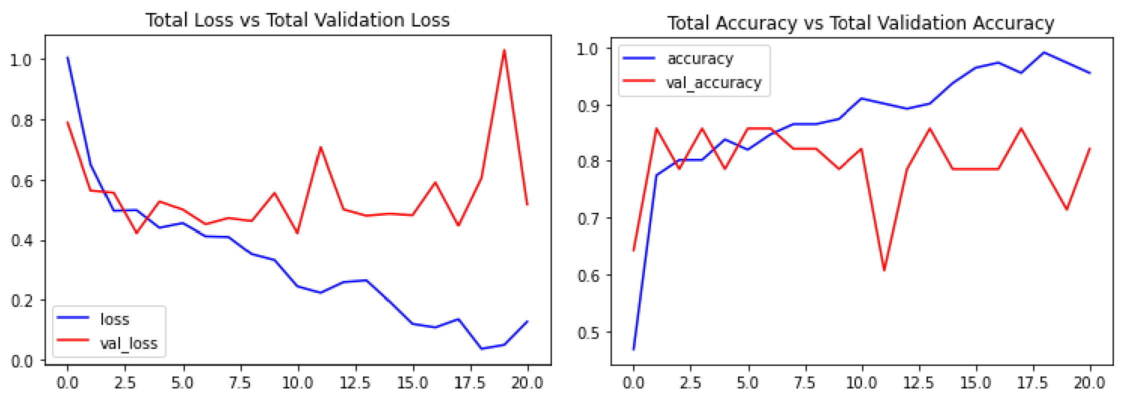
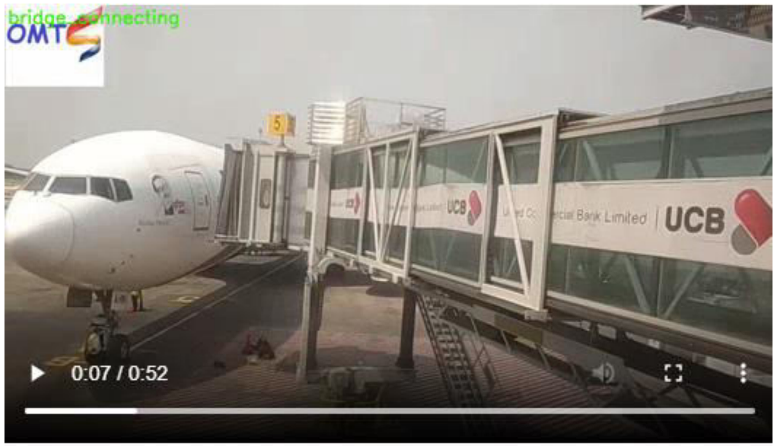


Fig A.9: Loss and Accuracy for ConvLSTM Model

We provided bridge connecting video as input, and we observe the below results:

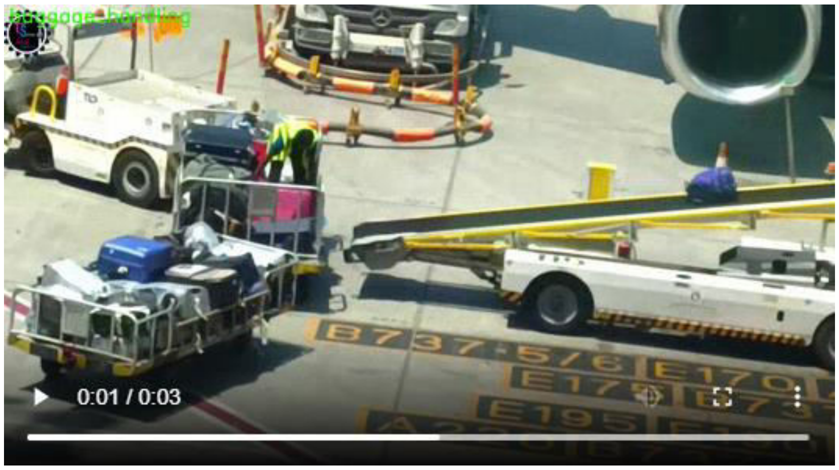


```
Action Predicted: bridge_connecting
Confidence: 0.8162801265716553
```

Fig A.10: Prediction of Bridge Connecting action using ConvLSTM Model

The model detected the action with 0.81 as confidence score.

We provided baggage handling video as input, and we observe the below results:

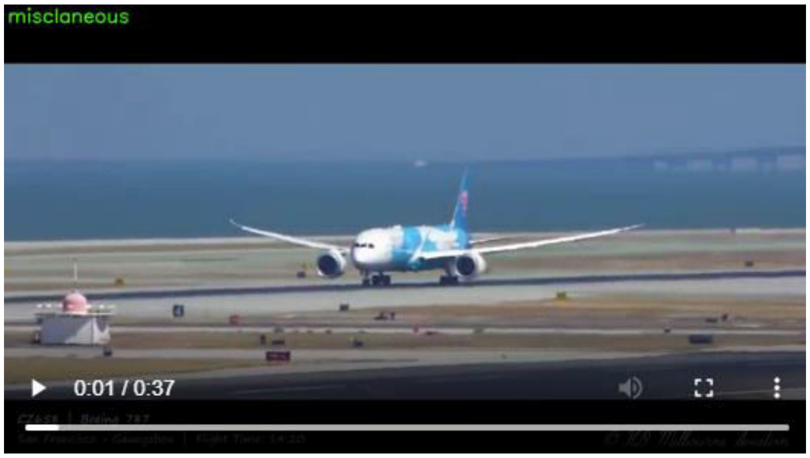


```
Action Predicted: baggage_handling
Confidence: 0.8300662636756897
```

Fig A.11: Prediction of Baggage Handling action using ConvLSTM Model

The model detected the action with 0.83 as confidence score.

We provided miscellaneous video as input, and we observe the below results:



```
Action Predicted: misclaneous
Confidence: 0.9995587468147278
```

Fig A.12: Prediction of miscellaneous action using ConvLSTM Model

The model detected the action with 0.99 as confidence score.

LRCN model:

Hyperparameters	Value
Epoch	70
Batch size	4
Loss	categorical_crossentropy
Optimizer	Adam

Table A.3: Hyperparameters for LRCN model

```
Epoch 37/70
50/50 [==============================] - 1s 22ms/step - loss: 0.1116 - accuracy: 0.9650 - val_loss: 0.3932 - val_accuracy: 0.
8800
Epoch 38/70
50/50 [==============================] - 1s 22ms/step - loss: 0.0907 - accuracy: 0.9750 - val_loss: 0.3903 - val_accuracy: 0.
8800
Epoch 39/70
50/50 [==============================] - 1s 21ms/step - loss: 0.0503 - accuracy: 0.9800 - val_loss: 0.5898 - val_accuracy: 0.
8400
```

Fig A.13: Training of LRCN Model

```
3/3 [==============================] - 0s 24ms/step - loss: 0.4916 - accuracy: 0.8452
```

Fig A.14: Evaluation of LRCN Model

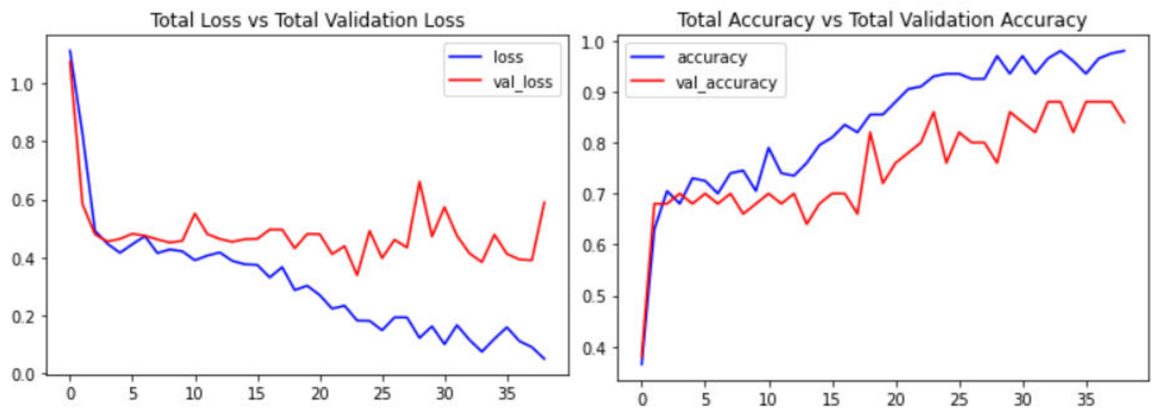
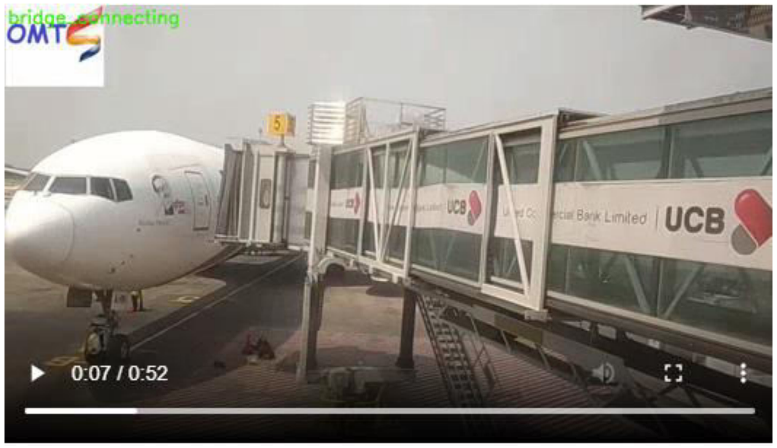


Fig A.15: Loss and Accuracy for LRCN Model

We provided bridge connecting video as input, and we observe the below results:

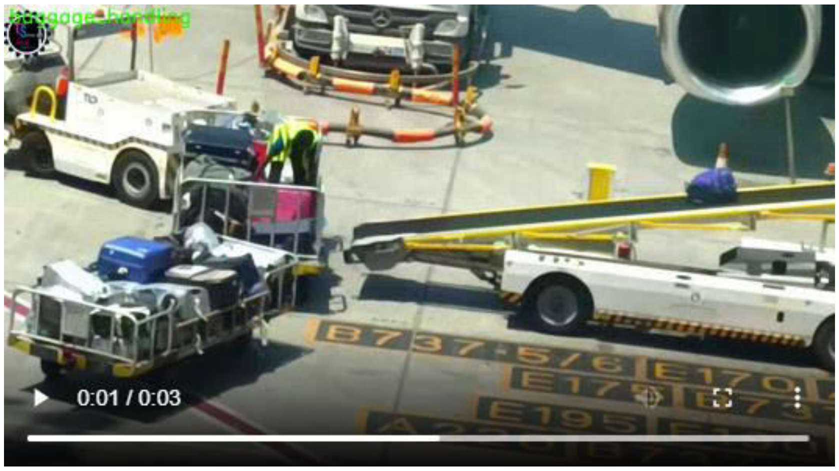


```
Action Predicted: bridge_connecting
Confidence: 0.8911964893341064
```

Fig A.16: Prediction of Bridge Connecting action using LRCN Model

The model detected the action with 0.89 as confidence score.

We provided baggage handling video as input, and we observe the below results:

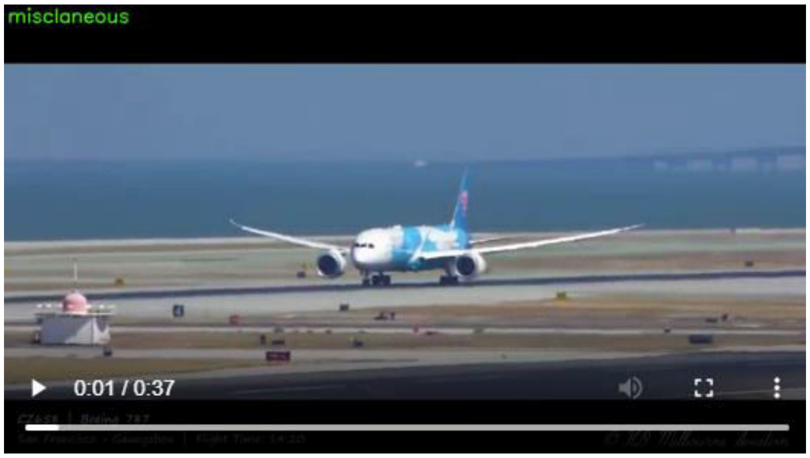


```
Action Predicted: baggage_handling
Confidence: 0.8003833293914795
```

Fig A.17: Prediction of Baggage Handling action using LRCN Model

The model detected the action with 0.80 as confidence score.

We provided miscellaneous video as input, and we observe the below results:



```
Action Predicted: misclaneous
Confidence: 0.9992884397506714
```

Fig A.18: Prediction of miscellaneous action using LRCN Model

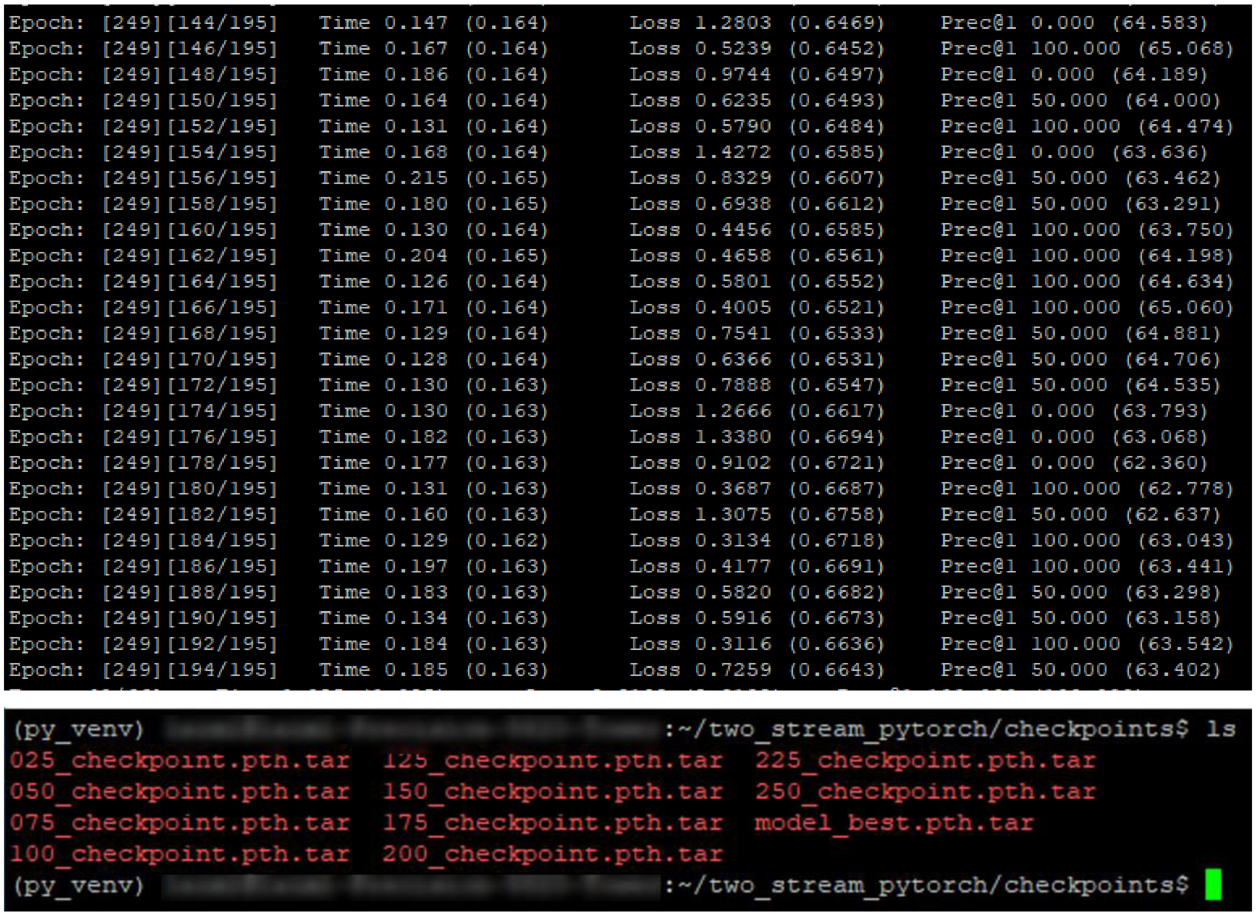
The model detected the action with 0.99 as confidence score.

Two stream Network:

Hyperparameters	RGB Value	Optical Value
Epoch	250	350
Batch size	25	25
Learning rate	0.001	0.001
Loss	Cross entropy	Cross entropy
Optimizer	SGD	SGD

Table A.4 Hyperparameters for Two Stream model

RGB Training:

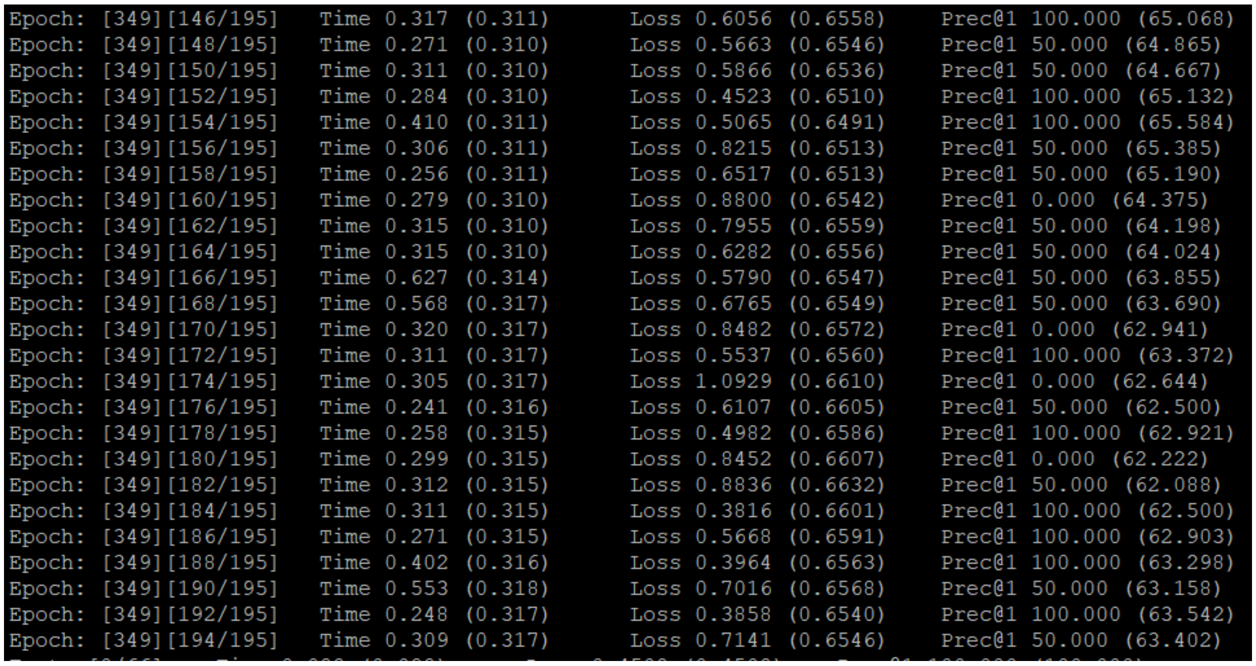


```
Epoch: [249][144/195]   Time 0.147 (0.164)      Loss 1.2803 (0.6469)     Prec@1 0.000 (64.583)
Epoch: [249][146/195]   Time 0.167 (0.164)      Loss 0.5239 (0.6452)     Prec@1 100.000 (65.068)
Epoch: [249][148/195]   Time 0.186 (0.164)      Loss 0.9744 (0.6497)     Prec@1 0.000 (64.189)
Epoch: [249][150/195]   Time 0.164 (0.164)      Loss 0.6235 (0.6493)     Prec@1 50.000 (64.000)
Epoch: [249][152/195]   Time 0.131 (0.164)      Loss 0.5790 (0.6484)     Prec@1 100.000 (64.474)
Epoch: [249][154/195]   Time 0.168 (0.164)      Loss 1.4272 (0.6585)     Prec@1 0.000 (63.636)
Epoch: [249][156/195]   Time 0.215 (0.165)      Loss 0.8329 (0.6607)     Prec@1 50.000 (63.462)
Epoch: [249][158/195]   Time 0.180 (0.165)      Loss 0.6938 (0.6612)     Prec@1 50.000 (63.291)
Epoch: [249][160/195]   Time 0.130 (0.164)      Loss 0.4456 (0.6585)     Prec@1 100.000 (63.750)
Epoch: [249][162/195]   Time 0.204 (0.165)      Loss 0.4658 (0.6561)     Prec@1 100.000 (64.198)
Epoch: [249][164/195]   Time 0.126 (0.164)      Loss 0.5801 (0.6552)     Prec@1 100.000 (64.634)
Epoch: [249][166/195]   Time 0.171 (0.164)      Loss 0.4005 (0.6521)     Prec@1 100.000 (65.060)
Epoch: [249][168/195]   Time 0.129 (0.164)      Loss 0.7541 (0.6533)     Prec@1 50.000 (64.881)
Epoch: [249][170/195]   Time 0.128 (0.164)      Loss 0.6366 (0.6531)     Prec@1 50.000 (64.706)
Epoch: [249][172/195]   Time 0.130 (0.163)      Loss 0.7888 (0.6547)     Prec@1 50.000 (64.535)
Epoch: [249][174/195]   Time 0.130 (0.163)      Loss 1.2666 (0.6617)     Prec@1 0.000 (63.793)
Epoch: [249][176/195]   Time 0.182 (0.163)      Loss 1.3380 (0.6694)     Prec@1 0.000 (63.068)
Epoch: [249][178/195]   Time 0.177 (0.163)      Loss 0.9102 (0.6721)     Prec@1 0.000 (62.360)
Epoch: [249][180/195]   Time 0.131 (0.163)      Loss 0.3687 (0.6687)     Prec@1 100.000 (62.778)
Epoch: [249][182/195]   Time 0.160 (0.163)      Loss 1.3075 (0.6758)     Prec@1 50.000 (62.637)
Epoch: [249][184/195]   Time 0.129 (0.162)      Loss 0.3134 (0.6718)     Prec@1 100.000 (63.043)
Epoch: [249][186/195]   Time 0.197 (0.163)      Loss 0.4177 (0.6691)     Prec@1 100.000 (63.441)
Epoch: [249][188/195]   Time 0.183 (0.163)      Loss 0.5820 (0.6682)     Prec@1 50.000 (63.298)
Epoch: [249][190/195]   Time 0.134 (0.163)      Loss 0.5916 (0.6673)     Prec@1 50.000 (63.158)
Epoch: [249][192/195]   Time 0.184 (0.163)      Loss 0.3116 (0.6636)     Prec@1 100.000 (63.542)
Epoch: [249][194/195]   Time 0.185 (0.163)      Loss 0.7259 (0.6643)     Prec@1 50.000 (63.402)
```

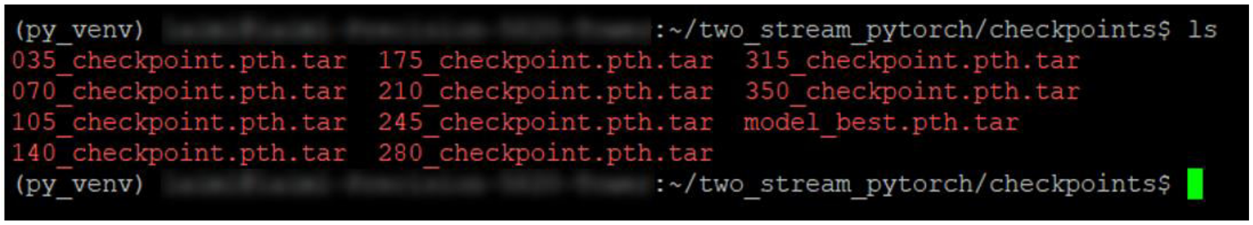
```
(py_venv) [redacted]:~/two_stream_pytorch/checkpoints$ ls
025_checkpoint.pth.tar  125_checkpoint.pth.tar  225_checkpoint.pth.tar
050_checkpoint.pth.tar  150_checkpoint.pth.tar  250_checkpoint.pth.tar
075_checkpoint.pth.tar  175_checkpoint.pth.tar  model_best.pth.tar
100_checkpoint.pth.tar  200_checkpoint.pth.tar
(py_venv) [redacted]:~/two_stream_pytorch/checkpoints$
```

Fig A.19: Training of Two Stream Network for RGB Frames

Optical flow training:



```
Epoch: [349][146/195]   Time 0.317 (0.311)      Loss 0.6056 (0.6558)    Prec@1 100.000 (65.068)
Epoch: [349][148/195]   Time 0.271 (0.310)      Loss 0.5663 (0.6546)    Prec@1 50.000 (64.865)
Epoch: [349][150/195]   Time 0.311 (0.310)      Loss 0.5866 (0.6536)    Prec@1 50.000 (64.667)
Epoch: [349][152/195]   Time 0.284 (0.310)      Loss 0.4523 (0.6510)    Prec@1 100.000 (65.132)
Epoch: [349][154/195]   Time 0.410 (0.311)      Loss 0.5065 (0.6491)    Prec@1 100.000 (65.584)
Epoch: [349][156/195]   Time 0.306 (0.311)      Loss 0.8215 (0.6513)    Prec@1 50.000 (65.385)
Epoch: [349][158/195]   Time 0.256 (0.311)      Loss 0.6517 (0.6513)    Prec@1 50.000 (65.190)
Epoch: [349][160/195]   Time 0.279 (0.310)      Loss 0.8800 (0.6542)    Prec@1 0.000 (64.375)
Epoch: [349][162/195]   Time 0.315 (0.310)      Loss 0.7955 (0.6559)    Prec@1 50.000 (64.198)
Epoch: [349][164/195]   Time 0.315 (0.310)      Loss 0.6282 (0.6556)    Prec@1 50.000 (64.024)
Epoch: [349][166/195]   Time 0.627 (0.314)      Loss 0.5790 (0.6547)    Prec@1 50.000 (63.855)
Epoch: [349][168/195]   Time 0.568 (0.317)      Loss 0.6765 (0.6549)    Prec@1 50.000 (63.690)
Epoch: [349][170/195]   Time 0.320 (0.317)      Loss 0.8482 (0.6572)    Prec@1 0.000 (62.941)
Epoch: [349][172/195]   Time 0.311 (0.317)      Loss 0.5537 (0.6560)    Prec@1 100.000 (63.372)
Epoch: [349][174/195]   Time 0.305 (0.317)      Loss 1.0929 (0.6610)    Prec@1 0.000 (62.644)
Epoch: [349][176/195]   Time 0.241 (0.316)      Loss 0.6107 (0.6605)    Prec@1 50.000 (62.500)
Epoch: [349][178/195]   Time 0.258 (0.315)      Loss 0.4982 (0.6586)    Prec@1 100.000 (62.921)
Epoch: [349][180/195]   Time 0.299 (0.315)      Loss 0.8452 (0.6607)    Prec@1 0.000 (62.222)
Epoch: [349][182/195]   Time 0.312 (0.315)      Loss 0.8836 (0.6632)    Prec@1 50.000 (62.088)
Epoch: [349][184/195]   Time 0.311 (0.315)      Loss 0.3816 (0.6601)    Prec@1 100.000 (62.500)
Epoch: [349][186/195]   Time 0.271 (0.315)      Loss 0.5668 (0.6591)    Prec@1 100.000 (62.903)
Epoch: [349][188/195]   Time 0.402 (0.316)      Loss 0.3964 (0.6563)    Prec@1 100.000 (63.298)
Epoch: [349][190/195]   Time 0.553 (0.318)      Loss 0.7016 (0.6568)    Prec@1 50.000 (63.158)
Epoch: [349][192/195]   Time 0.248 (0.317)      Loss 0.3858 (0.6540)    Prec@1 100.000 (63.542)
Epoch: [349][194/195]   Time 0.309 (0.317)      Loss 0.7141 (0.6546)    Prec@1 50.000 (63.402)
```

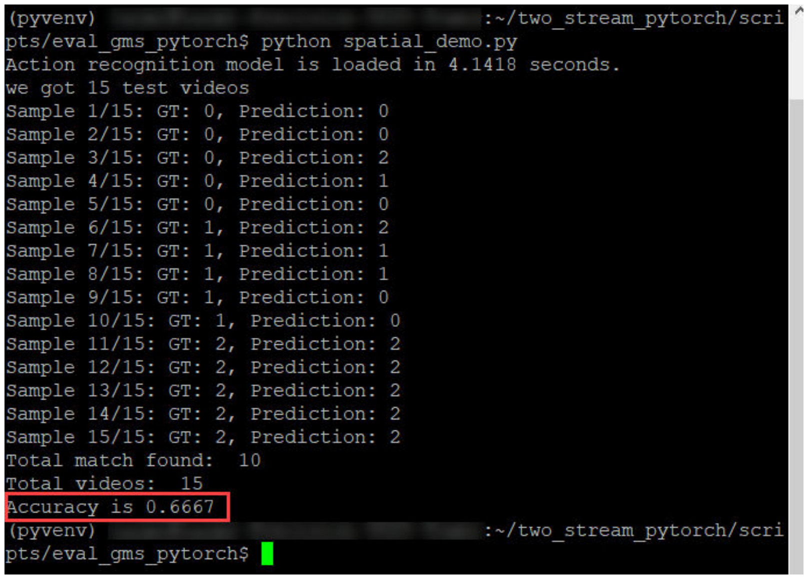


```
(py_venv) [redacted]:~/two_stream_pytorch/checkpoints$ ls
035_checkpoint.pth.tar  175_checkpoint.pth.tar  315_checkpoint.pth.tar
070_checkpoint.pth.tar  210_checkpoint.pth.tar  350_checkpoint.pth.tar
105_checkpoint.pth.tar  245_checkpoint.pth.tar  model_best.pth.tar
140_checkpoint.pth.tar  280_checkpoint.pth.tar
(py_venv) [redacted]:~/two_stream_pytorch/checkpoints$
```

Fig A.20: Training of Two Stream Network for Optical Flow Frames

Evaluation of Two Stream Network:

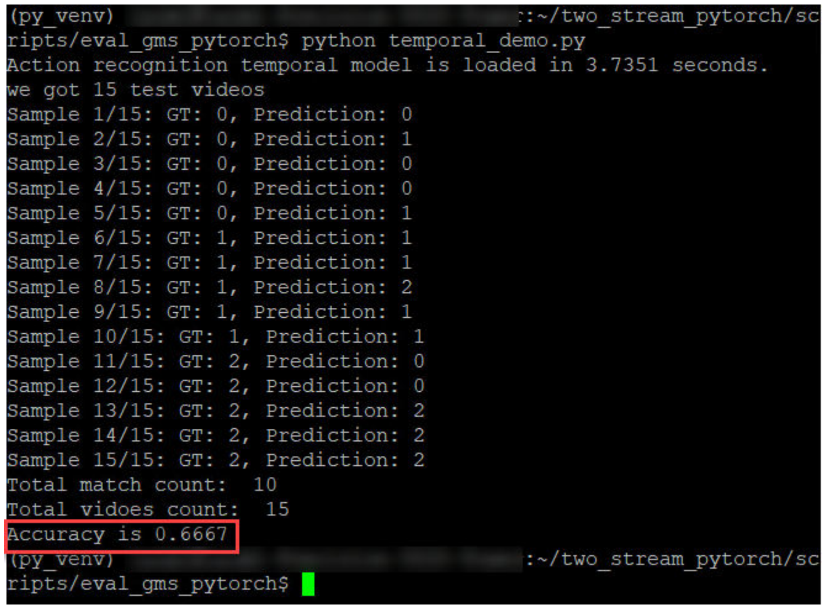
Prediction of spatial stream network:



```
(pyvenv) [redacted] :~/two_stream_pytorch/scri
pts/eval_gms_pytorch$ python spatial_demo.py
Action recognition model is loaded in 4.1418 seconds.
we got 15 test videos
Sample 1/15: GT: 0, Prediction: 0
Sample 2/15: GT: 0, Prediction: 0
Sample 3/15: GT: 0, Prediction: 2
Sample 4/15: GT: 0, Prediction: 1
Sample 5/15: GT: 0, Prediction: 0
Sample 6/15: GT: 1, Prediction: 2
Sample 7/15: GT: 1, Prediction: 1
Sample 8/15: GT: 1, Prediction: 1
Sample 9/15: GT: 1, Prediction: 0
Sample 10/15: GT: 1, Prediction: 0
Sample 11/15: GT: 2, Prediction: 2
Sample 12/15: GT: 2, Prediction: 2
Sample 13/15: GT: 2, Prediction: 2
Sample 14/15: GT: 2, Prediction: 2
Sample 15/15: GT: 2, Prediction: 2
Total match found:  10
Total videos:  15
Accuracy is 0.6667
(pyvenv) [redacted] :~/two_stream_pytorch/scri
pts/eval_gms_pytorch$
```

Fig A.21: Prediction of Two Stream Network for RGB Frames

Prediction of temporal stream network:



```
(py_venv) [redacted] :~/two_stream_pytorch/sc
ripts/eval_gms_pytorch$ python temporal_demo.py
Action recognition temporal model is loaded in 3.7351 seconds.
we got 15 test videos
Sample 1/15: GT: 0, Prediction: 0
Sample 2/15: GT: 0, Prediction: 1
Sample 3/15: GT: 0, Prediction: 0
Sample 4/15: GT: 0, Prediction: 0
Sample 5/15: GT: 0, Prediction: 1
Sample 6/15: GT: 1, Prediction: 1
Sample 7/15: GT: 1, Prediction: 1
Sample 8/15: GT: 1, Prediction: 2
Sample 9/15: GT: 1, Prediction: 1
Sample 10/15: GT: 1, Prediction: 1
Sample 11/15: GT: 2, Prediction: 0
Sample 12/15: GT: 2, Prediction: 0
Sample 13/15: GT: 2, Prediction: 2
Sample 14/15: GT: 2, Prediction: 2
Sample 15/15: GT: 2, Prediction: 2
Total match count:  10
Total vidoes count:  15
Accuracy is 0.6667
(py_venv) [redacted] :~/two_stream_pytorch/sc
ripts/eval_gms_pytorch$
```

Fig A.22: Prediction of Two Stream Network for Optical Flow Frames

Appendix B

Mathematical Concepts

B.1 Confidence Score:

The confidence score is a number between 0 and 1. A score of 1 is likely an exact match, while a score 0 means, that no matching answer was found. The highest score, the greater the confidence in the answer. It is calculated as an evaluation standard. This score shows the probability of the image being detected correctly by the algorithm and is given as a percentage. Higher confidence scores indicate greater probable accuracy, while lower confidence scores indicate less probable accuracy. The scores are taken on the mean average precision at different IoU (Intersection over Union) thresholds.

Intersection over Union (IoU) is measured by the magnitude of overlap between two bounding boxes of two objects. The formula of IoU is given below:

$$\mathrm{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{X \cap Y}{X \cup Y}$$

Fig B.1: IoU Formula

Precision is calculated at the threshold value depending on the True Positives (TP), False Positives (FP), and False Negatives (FN) which are the result of comparing the predicted object to the ground truth object. FN is not applicable for our case. A TP is counted when the ground truth bounding box is matched with the prediction bounding box. A FP is counted when a predicted object had no association with the ground truth object. A FN is counted when a ground truth object had no association with the predicted object. The confidence score is given as the mean over all the precision scores for all thresholds. The average precision is given as:

$$\text{Confidence} = \text{Average precision} = \frac{1}{|\text{Thresholds}|} \sum_T \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

Fig B.2: Confidence score Formula

Duly Completed Checklist

Check list of items for the Final report

a)	Is the Cover page in proper format?	[✓ Y] / N
b)	Is the Title page in proper format?	[✓ Y] / N
c)	Is the Certificate from the Supervisor in proper format? Has it been signed?	[✓ Y] / N
d)	Is Abstract included in the Report? Is it properly written?	[✓ Y] / N
e)	Does the Table of Contents page include chapter page numbers?	[✓ Y] / N
f)	Does the Report contain a summary of the literature survey?	[✓ Y] / N
i.	Are the Pages numbered properly?	[✓ Y] / N
ii.	Are the Figures numbered properly?	[✓ Y] / N
iii.	Are the Tables numbered properly?	[✓ Y] / N
iv.	Are the Captions for the Figures and Tables proper?	[✓ Y] / N
v.	Are the Appendices numbered?	[✓ Y] / N
g)	Does the Report have Conclusion / Recommendations of the work?	[✓ Y] / N
h)	Are References/Bibliography given in the Report?	[✓ Y] / N
i)	Have the References been cited in the Report?	[✓ Y] / N
j)	Is the citation of References / Bibliography in proper format?	[✓ Y] / N

(Signature of Student)
Date: 27/8/2022

(Signature of Supervisor)
Date: 27/8/2022