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

Identify Gate Operations using Action Recognition

Problem Statement:

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.

Benefit:

While stationary, the airline is not earning any revenue with its plane, while facing many costs at the same time. 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.

Objective of the Project:

Build a system which will take video as an input and recognize the ground activities which are bridge connecting, baggage handling and miscellaneous. To achieve this, we must build a model based on visual-analytical approach that can automatically recognize the activities.

Resource:

Operating System: High end Machine with 8VRAM or above
Languages/Scripting: Python
Libraries: PyTorch, Tensorflow, Matplotlib, OpenCV, Albumentations

Solution Architecture:

The spatiotemporal features of the videos are extracted using the image processing techniques. The preprocessed data will be given to the models. In Training phase, deep learning algorithms are used to extract the features and preciously recognize the actions.

Risk:

Data collection was at potential risk. We observed the open-source datasets for action recognition like ActivityNet, UCF101 and HMDB51. It has an average of 100 to 150 videos for each action. Downloaded videos from YouTube & performed trimming operation to focus only on the actions we required. Collected around 130 videos for each category.

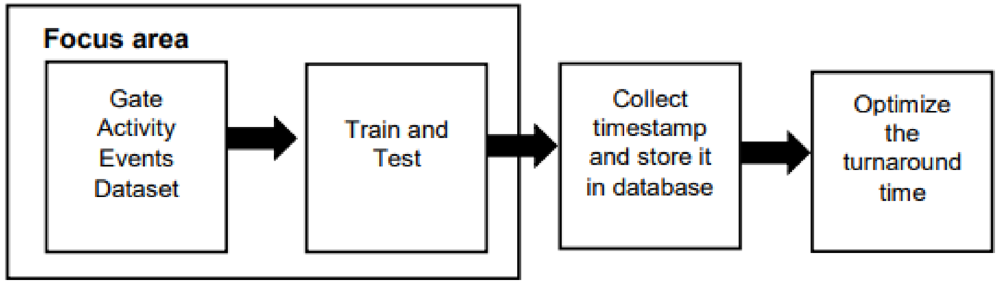


Fig 1: Overall Gate Monitoring System

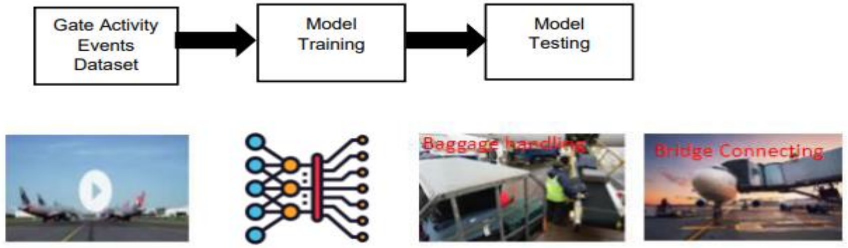


Fig 2: Proposed Solution Architecture

Project Plan & Deliverables:

SL NO	Task	Planned duration (In weeks)	Name of Deliverables	Status
1	Understanding the project requirements and specifications	2 weeks		Done
2	Data collection	2 weeks	Dataset	Done
3	Data Preprocessing	3 weeks	Processed Dataset	Done
4	Exploration of algorithms and initial training	5 weeks	Literature study & Model	Done
5	Adding miscellaneous data as one more category	1 week	Dataset	Done
6	Inference the model	1 weeks	Model	Done
7	Documentation for final report	2 weeks	Code & Report	Done

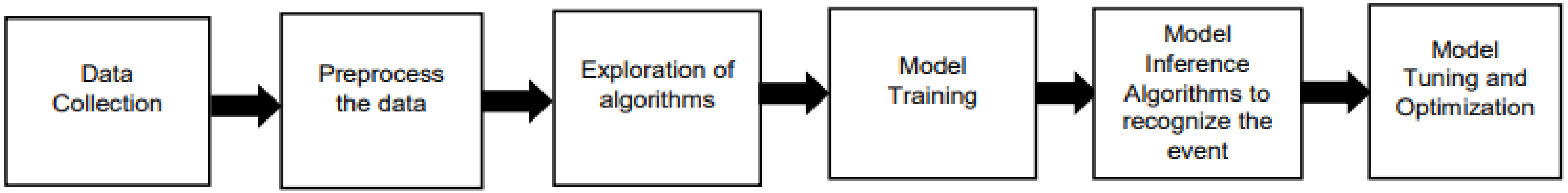
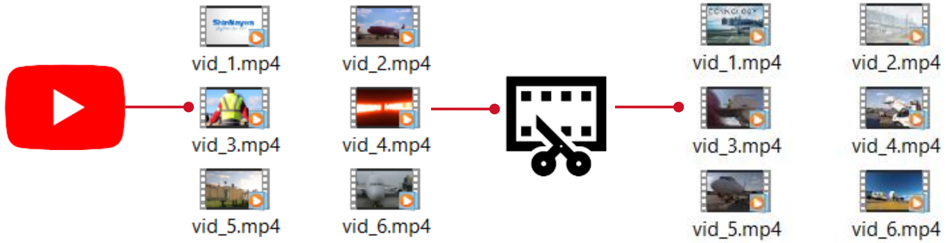


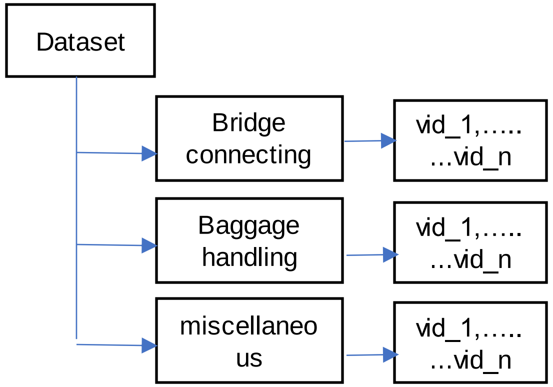
Fig 3: Representation of workflow

Data Acquistion:

Data Collection:



Dataset Folder Structure:

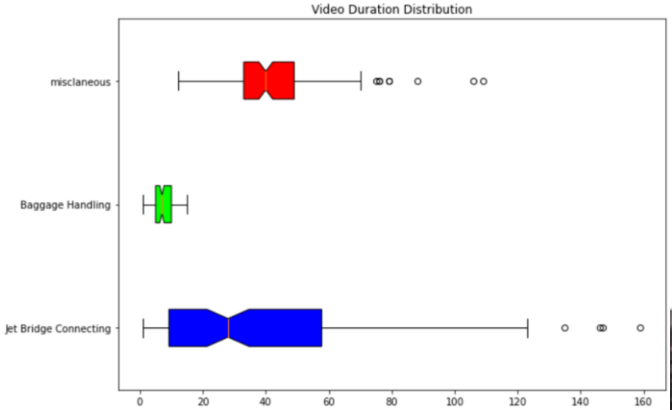


Data Visualization:

Data Exploration:

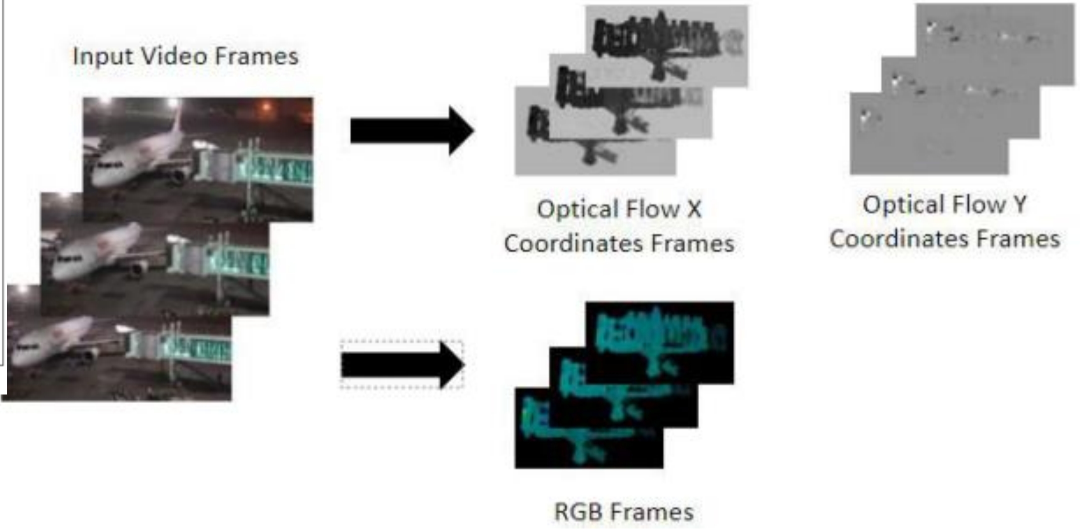
Action Categories	3
Average Videos per Action Category	130
Average Number of Frames per Video	812.85
Average Frames Width per Video	1154.58
Average Frames Height per Video	713.00
Average Frames per Seconds per Video	27.37

Statistic of overall dataset



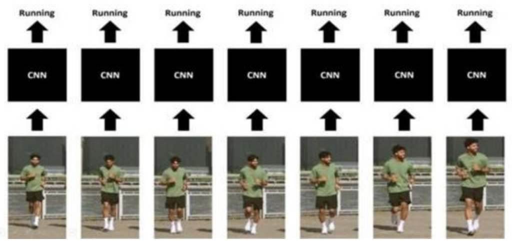
Statistic of overall dataset

Data Preprocessing:



Exploration of algorithms and model training:

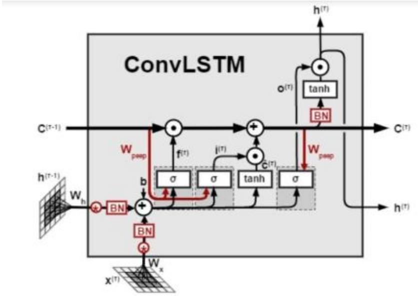
Single Frame CNN



Hyperparameters	Value
Epoch	50
Batch size	4
Loss	categorical_crossentropy
Optimizer	Adam

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.

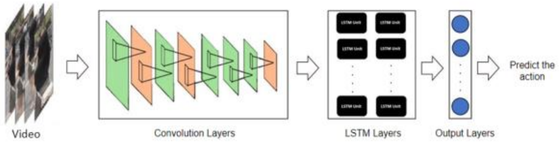
ConvLSTM



Hyperparameters	Value
Epoch	50
Batch size	4
Loss	categorical_crossentropy
Optimizer	Adam

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.

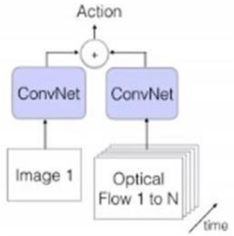
LRCN Model



Hyperparameters	Value
Epoch	50
Batch size	4
Loss	categorical_crossentropy
Optimizer	Adam

This network 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. This way the network learns spatiotemporal features directly in an end-to-end training, resulting in a robust model.

Two Stream Network

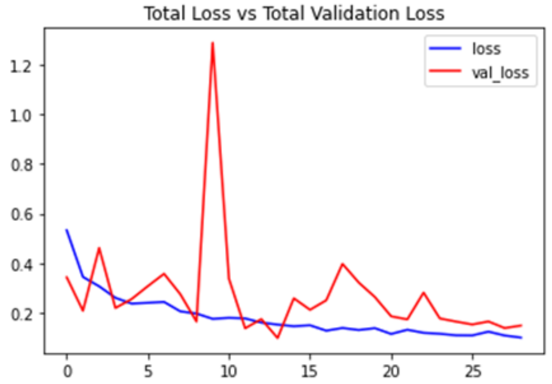


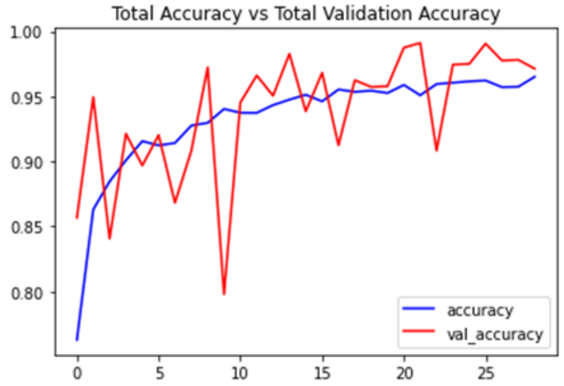
Hyperparameters	RGB Value	Optical Value
Epoch	250	350
Batch size	25	25
Learning rate	0.001	0.001
Loss	Cross entropy	Cross entropy
Optimizer	SGD	SGD

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.

Model Training Results:

Single Frame CNN

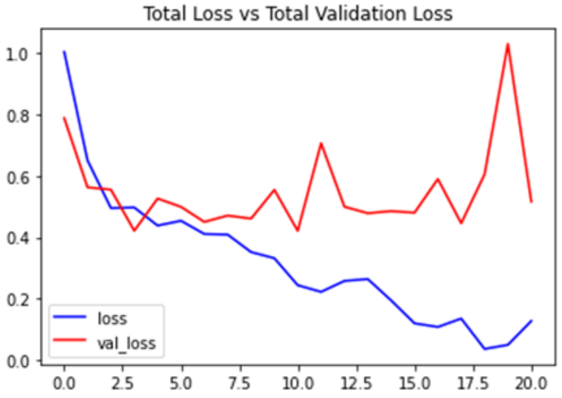


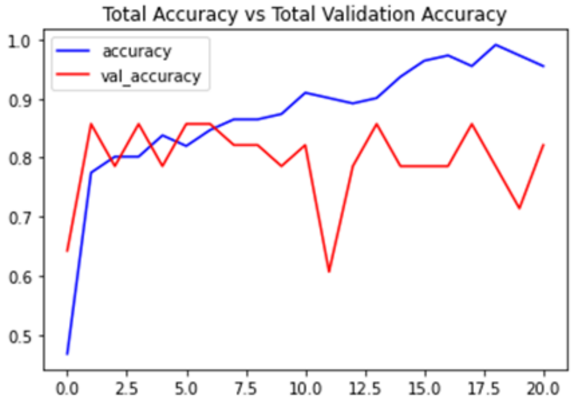


Training Results

Loss	0.09544
Accuracy	0.9787

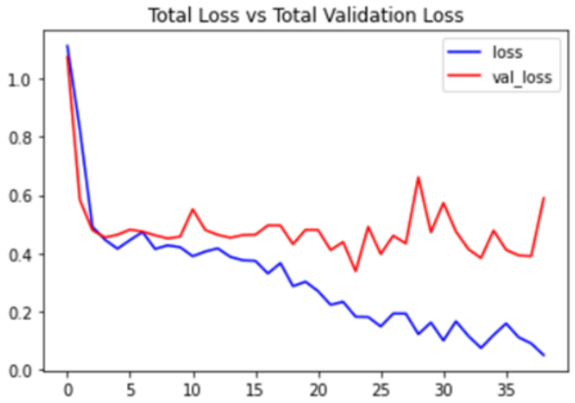
ConvLSTM

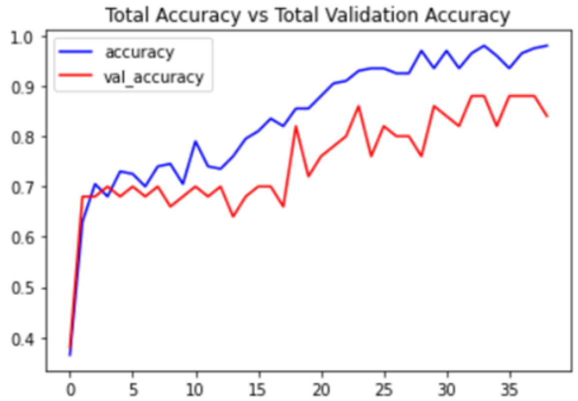




Loss	0.4735
Accuracy	0.80851

LRCN Model





Loss	0.4916
Accuracy	0.84523

Two Stream Network

RGB frames:

```
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)
```

Loss	0.6643
Accuracy	0.63402

Optical frames:

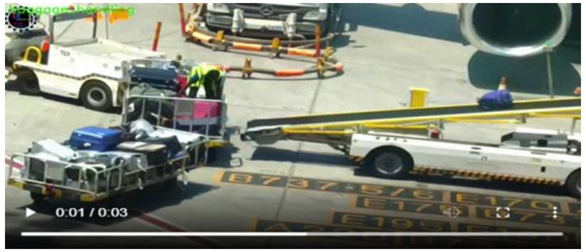
```
Epoch: [349][146/195]   Time 0.317 (0.311)    Loss 0.6056 (0.6558)    Prec@1 100.000 (65.068)
Epoch: [349][148/195]   Time 0.271 (0.310)    Loss 0.5663 (0.6546)    Prec@1 50.000 (64.865)
Epoch: [349][150/195]   Time 0.311 (0.310)    Loss 0.5866 (0.6536)    Prec@1 50.000 (64.667)
Epoch: [349][152/195]   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)
```

Loss	0.6546
Accuracy	0.63402

Model Inference:

Single Frame CNN


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

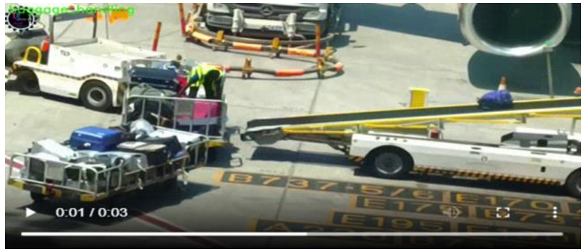



```
CLASS NAME: baggage_handling   AVERAGED PROBABILITY: 9.2e+01
CLASS NAME: bridge_connecting   AVERAGED PROBABILITY: 6.2
CLASS NAME: misclaneous   AVERAGED PROBABILITY: 1.5
```


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

ConvLSTM


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

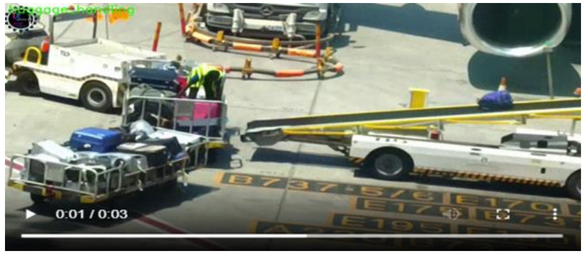



```
Action Predicted: baggage_handling
Confidence: 0.8300662636756897
```


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

LRCN Model


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




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


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

Two Stream Network

RGB frames:

```
(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$
```

Optical frames:

```
(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$
```

Questions?