DATA-DRIVEN COMPUTER VISION SYSTEM FOR MOBILE SECURITY ROBOTS

DATA 298B Project Presentation - Team 4

Project Advisor: Dr. Zeyu Jerry Gao

Table Of Contents

Project Introduction

Background

Market Analysis

Crime Statistics

Motivation

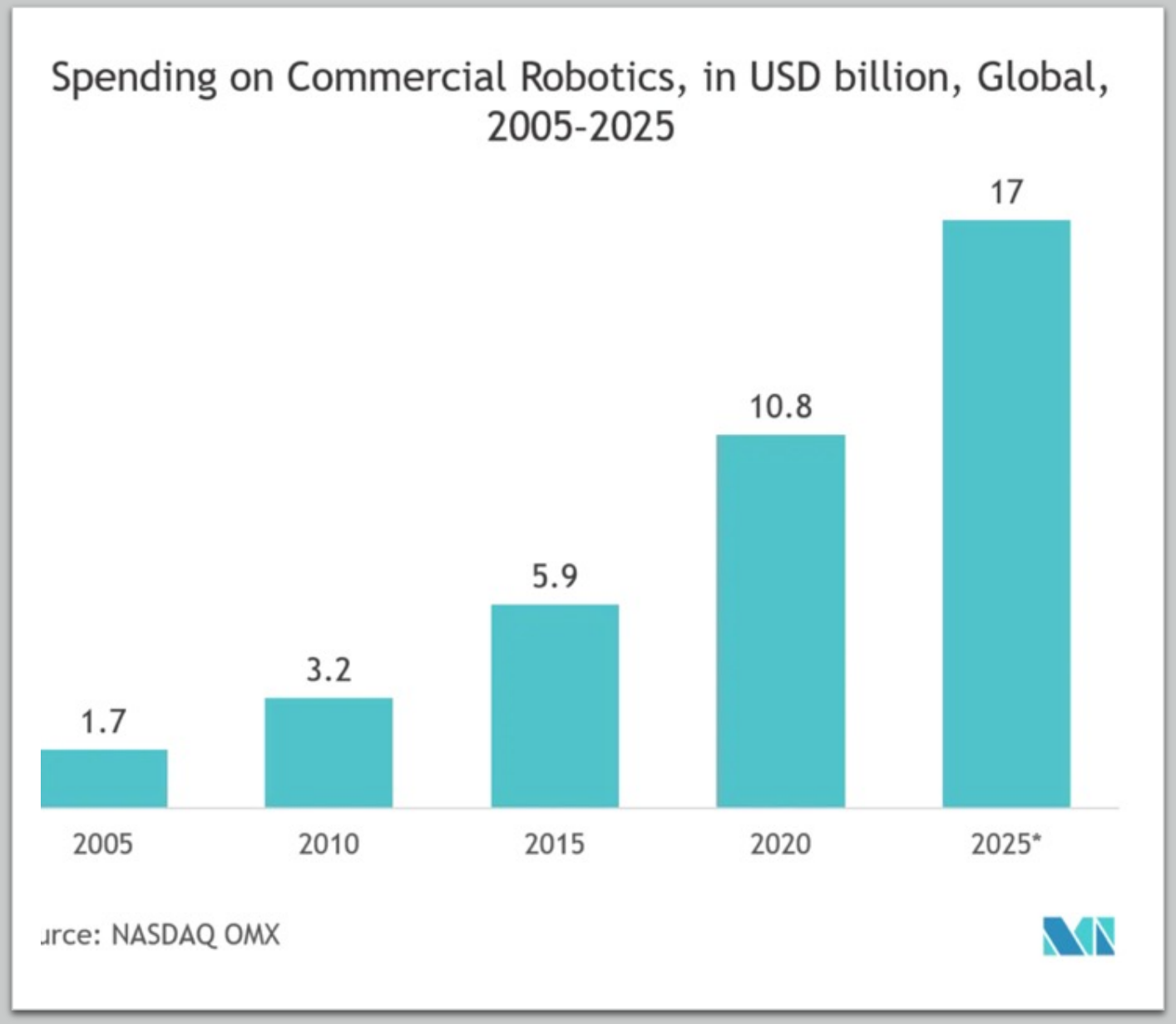
Project Goal

Background

- With the growing population there has been consequential rise in the number of crimes that have occurred.
- CCTV surveillance systems are stationary and can be outsmarted by criminal by changing the moment of their positions and ranges.
- Current commercially available robots are not comprehensive as they do not have many features such as human behaviour detection, weapon detection, or sending out real-time alerts to security person.

Market Analysis

According to report produced by Mordor Intelligence, the market for mobile security robots was estimated to be $8.87 billion in the year 2020 and is expected to reach a whopping value of $19.77 billion by 2026, which is almost twice as much growth in the span of 6 years.



Estimated Market Value for Mobile Security Robots by 2026

$19.77 Billion

Crime Rate Statistics

Aggravated Assaults

+4%

Residential Burglaries

+6%

Non-residential Burglaries

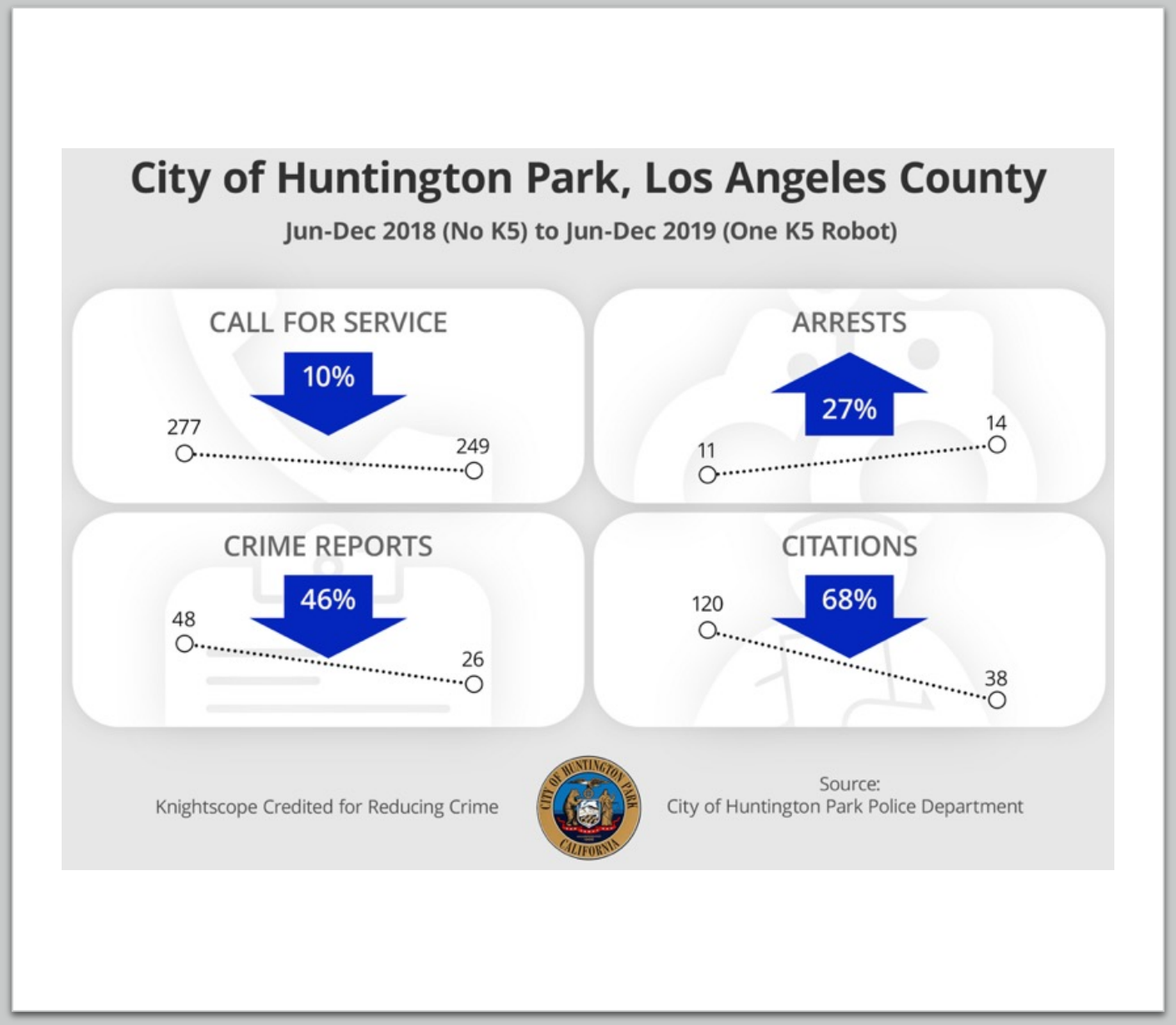
+8%

Vehicle Thefts

+15%

Motivation

- To optimize the response time to crimes
- To efficiently reduce the crime rate and supplement the police and security force with additional help.
- Security robots are useful, productive and can be deployed under adverse conditions where normal human-beings cannot operate.



Project Goal

To build a comprehensive and intelligent surveillance mobile robot, which uses various DL models to navigate, autonomously detect and recognize criminal behaviour, recognize and read license plates, and additionally alert security personnel of anomalous human behaviour (burglary, fights, vehicle thefts).

Burglary Detection Model

Data Engineering

Machine Learning Model

Machine Learning Results

Demo Video

Dataset

Class	*No. of Videos*	*Video Duration*	*No. of Frames*	*Source*
Burglary	100	~ 2 mins	~ 1600	UCF Crime
Normal	1000	~ 5 secs	~ 16000	Kaggle Violence

No. of Classes

2

Total Number of Videos Used for Training

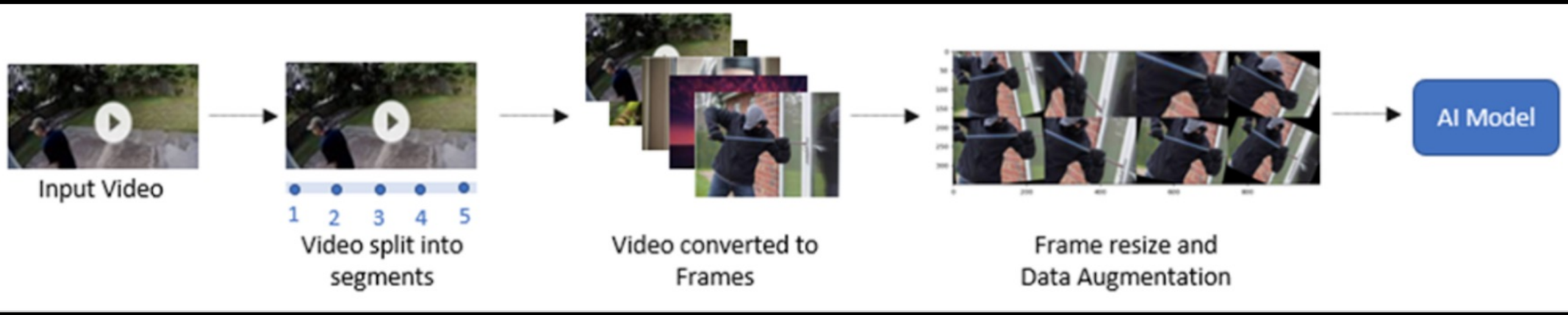
1100

Total Number of Frames

17600

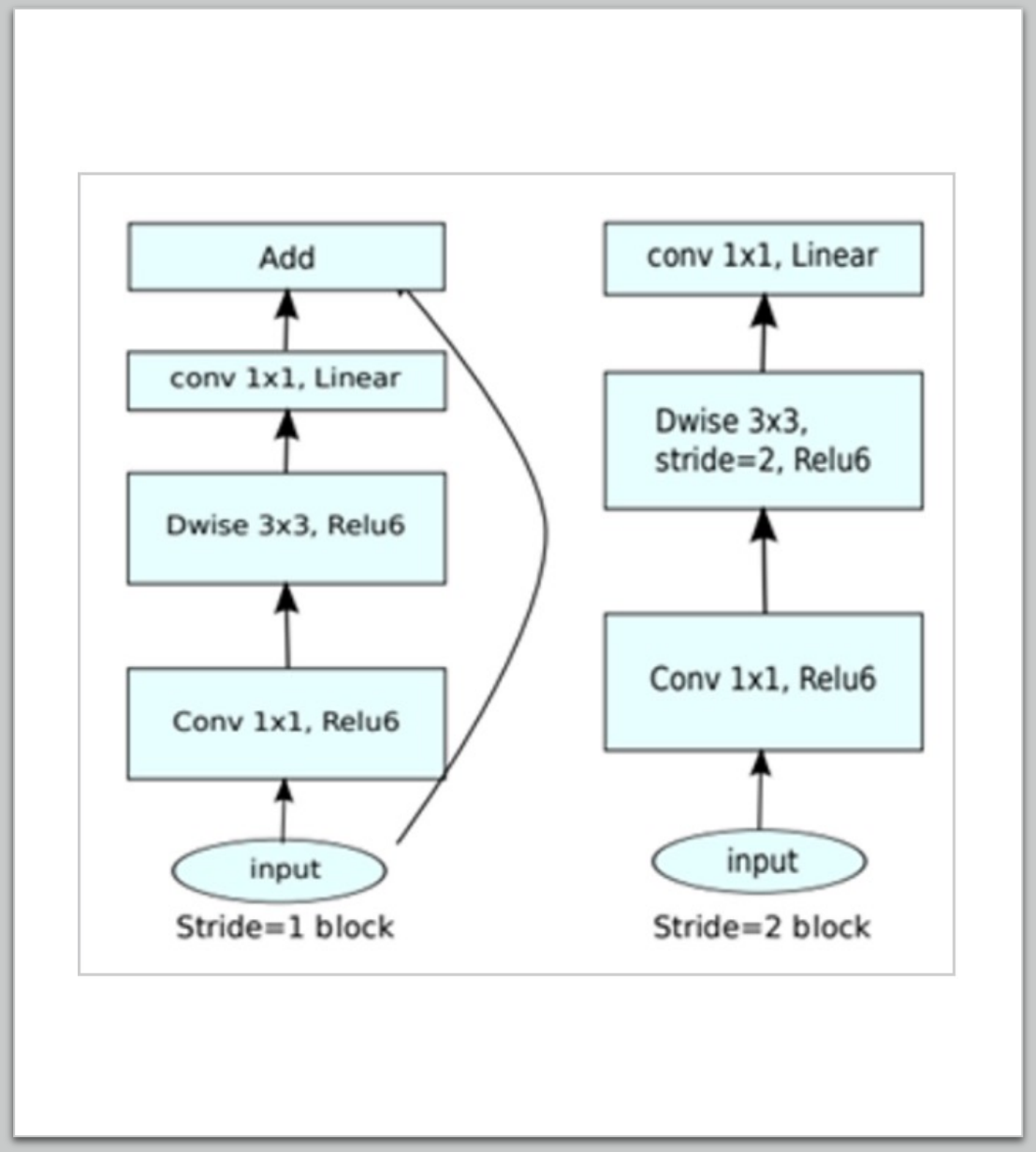
Data Pre-processing

- The MobileNetv2 model used here has been pre-trained using the ImageNet dataset.
- The features from both the burglary and normal videos are extracted after converting the videos into frames.
- The input videos are first broken into segments according to the length of the video, breaking down the video into frames, resizing the frames to uniform weight and height and choosing a fixed number of frames from each segment.
- Data augmentation is applied to the frames so the model can be trained with a greater variety of the dataset.


Input Video
1 2 3 4 5
Video split into segments
Video converted to Frames
Frame resize and Data Augmentation
AI Model

ML Model - MobileNetV2 with LSTM

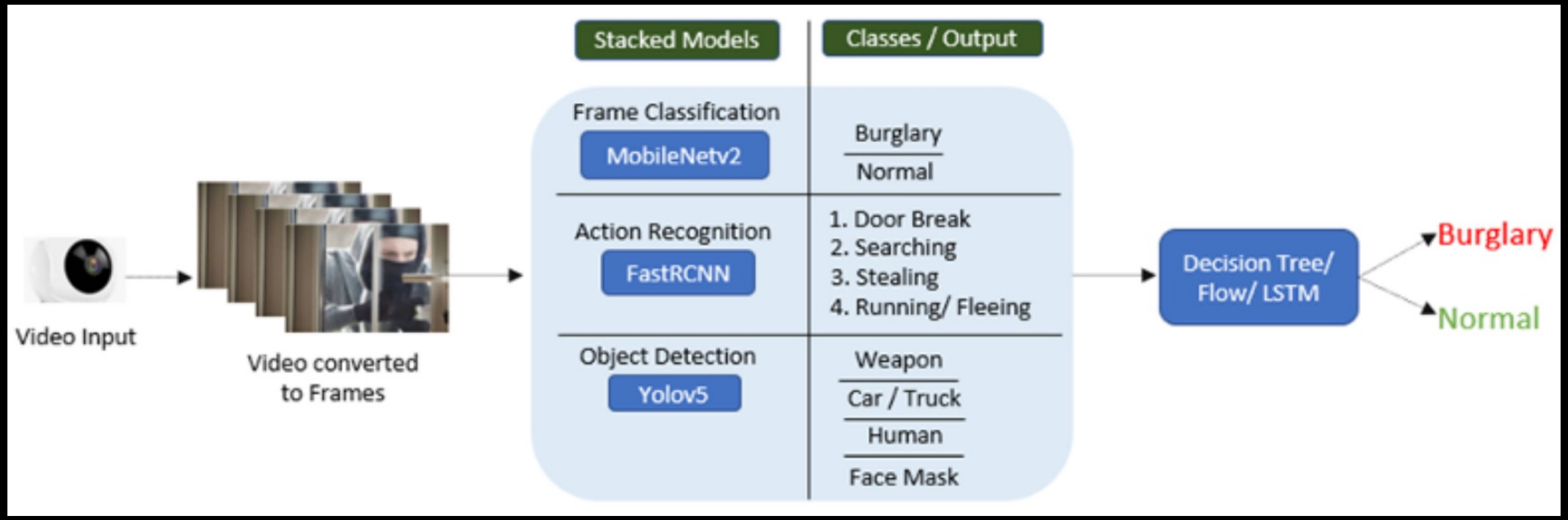
- As seen from the MobileNetv2 architecture diagram, two additional blocks have been included, each with 1 and 2 strides respectively with the previous MobileNet models.
- The last convolution layer is a linear layer instead of a RELU6 non-linear layer and this has been seen to yield greater accuracy.
- Bi-directional LSTM can store both past and future data due to data flow in both directions.
- This is useful in our case where burglary actions can happen in any sequence in the videos.



Stacked Model Implementation

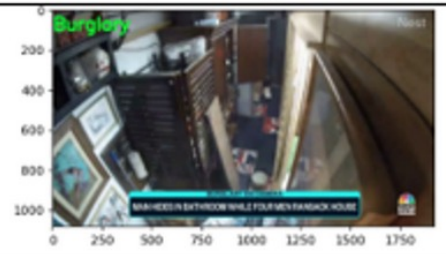
- After each frame has been labelled, the LSTM model that is added after the MobileNetv2 model helps with the classification of sequence of frames.
- The output prediction will be based on the probability of the class obtained for the frames of each video segment and the frames belonging to that segment will be classified accordingly.
- For object detection, YOLOv5 has been chosen as it is one of the most popular object detection models with high accuracy and 80 classes.
- Person, Car, Truck, Tools, Face Mask will be the class of objects detected by Yolov5 model.
- The combined output of the three models will be fed to a Bi-directional LSTM model because our requirement is to detect burglary based on a sequence of events and cannot work on separate images.
- Our LSTM model includes dropouts of 0.25 between the layers to avoid over-fitting.

Stacked Model Architecture - Diagram



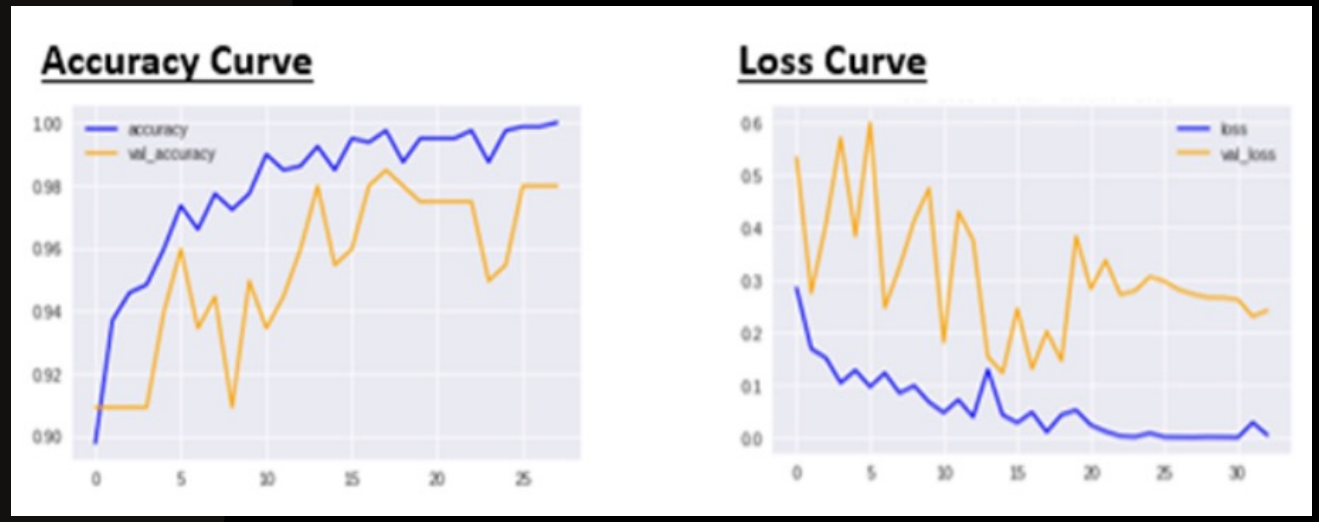
Why MobileNetV2 with LSTM?

- MobileNetv2 is a lightweight model with lesser parameters which is one of the reasons this model has been selected and implemented.
- With decrease in computational requirement, the model requires less operations and memory and consequently runs faster and gives results faster.

Model	ResNet50+LSTM	MobileNetv2+LSTM
Accuracy	91%	99%
F1-Score	0.89	0.95
Parameters	Total params: 25,753,250 Trainable params: 17,997,346 Non-trainable params: 7,755,904	Total params: 3,637,090 Trainable params: 3,060,642 Non-trainable params: 576,448
Training time	920 ms / epoch	376 ms / epoch
Prediction results on unseen test video		

Machine Learning Results

- The accuracy keeps increasing as the model keeps training for more epochs.
- The validation accuracy then peaks at 99% when the learning stops as the training accuracy touches 100%.
- Correspondingly, from the loss curve, we can see that the loss keeps decreasing with increasing number of epochs.




```
model_evaluation_history = MoBiLSTM_model.evaluate(features_test, labels_test)
```

```
4/4 [==============================] - 2s 213ms/step - loss: 0.0756 - accuracy: 0.9909
```

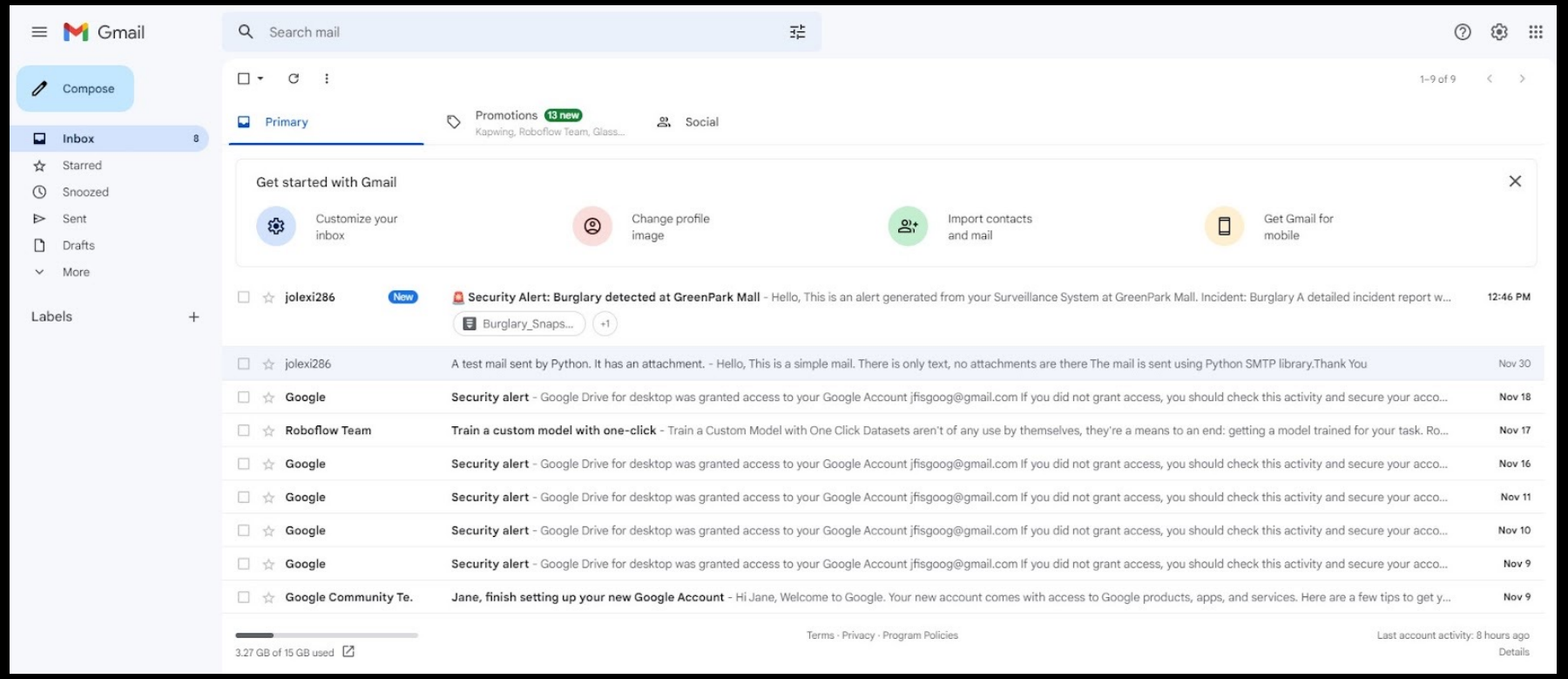
Model Output – Single Frame

The current visualization of the burglary detection model consists of a set of video frames with labels based on its classification. There is also an output video generated for each test video input along with the class labels.

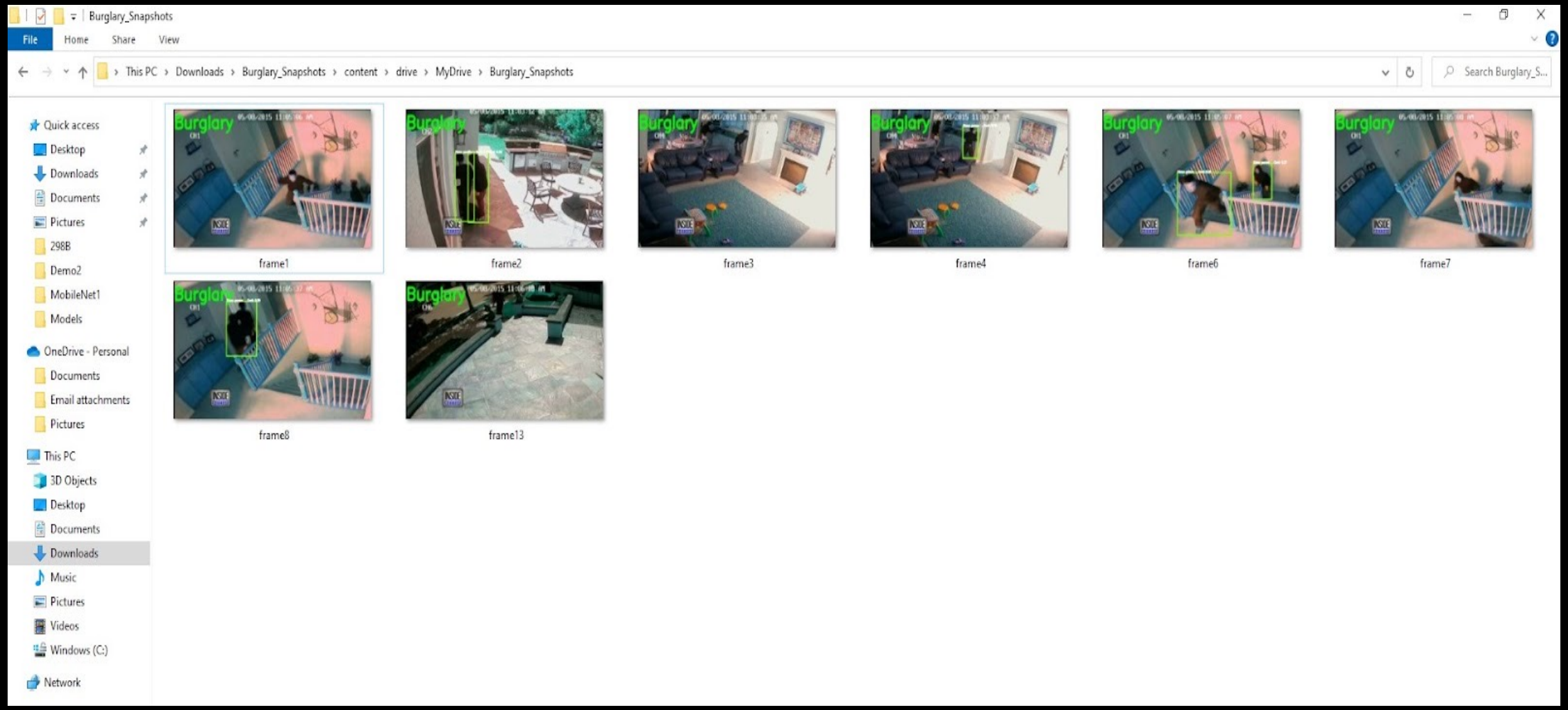
Model Output – Multiple Frames

Email Alerts to Security Personnel

The model is programmed to send emails to the security personnel if burglary was detected along with the frames in which burglary was detected.



Email Attachments



Demo Video


Burglary
Nest
BURGLARY ON CAMERA
MAN HIDES IN BATHROOM WHILE FOUR MEN RANSACK HOUSE
NOW

Fight Detection Model

Data Engineering

Machine Learning Model

Machine Learning Results

Demo Video

Data Collection

- The first dataset that has been used is the UCSD pedestrian dataset for the normal images.
- In this there are some videos of pedestrians walking while some of the videos which are abnormal are fighting, littering people.
- Another video dataset that we are using is from the UCF dataset.
- UCF is a very huge crime dataset which has different classes. From this dataset we are only using the videos of fighting class.
- These are the videos which are one to two minutes long and it also consists of fighting and other image sequences like normal walking people.

Note: For our use we are only using the trimmed videos from fighting videos which have only fighting sequences. So, these are all the actual data preparation that we are using for this model.

No. of Classes

2

Total Number of Videos Used for Training

37

Duration of Each Training Video

~ 60 secs

Total Number of Frames

333,000

Data Pre-processing

- The MobileNetv2 model used here has been pre-trained using the ImageNet dataset.
- The features from both the burglary and normal videos are extracted after converting the videos into frames.
- The input videos are first broken into segments according to the length of the video, breaking down the video into frames, resizing the frames to uniform weight and height and choosing a fixed number of frames from each segment.
- Data augmentation is applied to the frames so the model can be trained with a greater variety of the dataset.

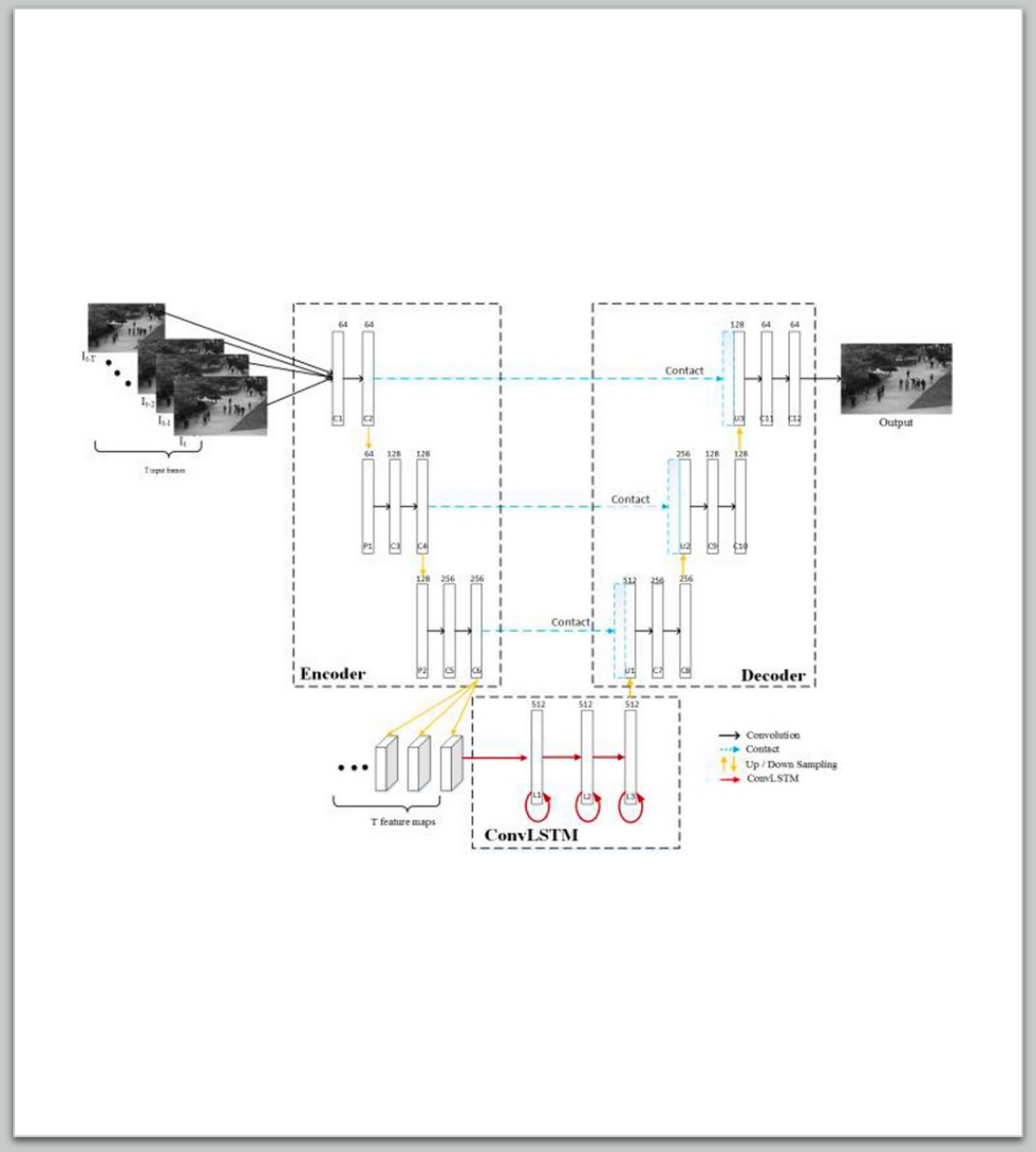
Feature Extraction

- We are starting by extracting the frames from videos using cv2 and PIL libraries.
- All these images are of different sizes, so we are resizing them to 277 x 277 images and then storing them as JPG images. These are basically the extracted gestures.
- Once all the features are extracted from the videos, we are converting them into a NumPy array of images to pass through the neural network of spatio-temporal autoencoders to learn the inner features.

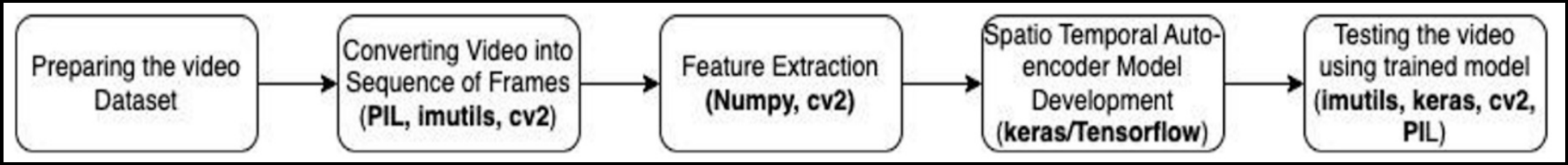
```
vid = f
vidcap = cv2.VideoCapture(vid)
def getFrame(sec):
    vidcap.set(cv2.CAP_PROP_POS_MSEC,sec*1000)
    hasFrames,image = vidcap.read()
    if hasFrames:
        path = "/content/train/frames/" + filename + "frame"+str(count)+".jpg'

        cv2.imwrite(path, image) # save frame as JPG file

    return hasFrames
sec = 0
frameRate = 0.2 #//it will capture image in each 0.5 second
count=1
success = getFrame(sec)
while success:
    count = count + 1
    sec = sec + frameRate
    sec = round(sec, 2)
    success = getFrame(sec)
```

ML Model – Spatio-Temporal Autoencoder

- This DL architecture has two parts, the first part is training the generator network model which predicts the next frame of image sequence.
- That has been done with the GAN module and this module has both generator and discriminative networks that helps in increasing the accuracy of predicted frames.
- Then the generator is replaced with the proposed Spatio-temporal U-Net and this approach uses the model temporal data between frames.
- The comprehensive approach of both spatial and temporal information is making this network more suitable for video processing.



Model Implementation Flow Diagram



Why Spatio-Temporal Autoencoder?

- Spatio-Temporal Autoencoders make use of deep neural networks specifically for video inputs.
- It extracts the features from a video both temporally and spatially by performing 3-D convolutions.
- This proves advantageous specially to capture minute changes within a video.

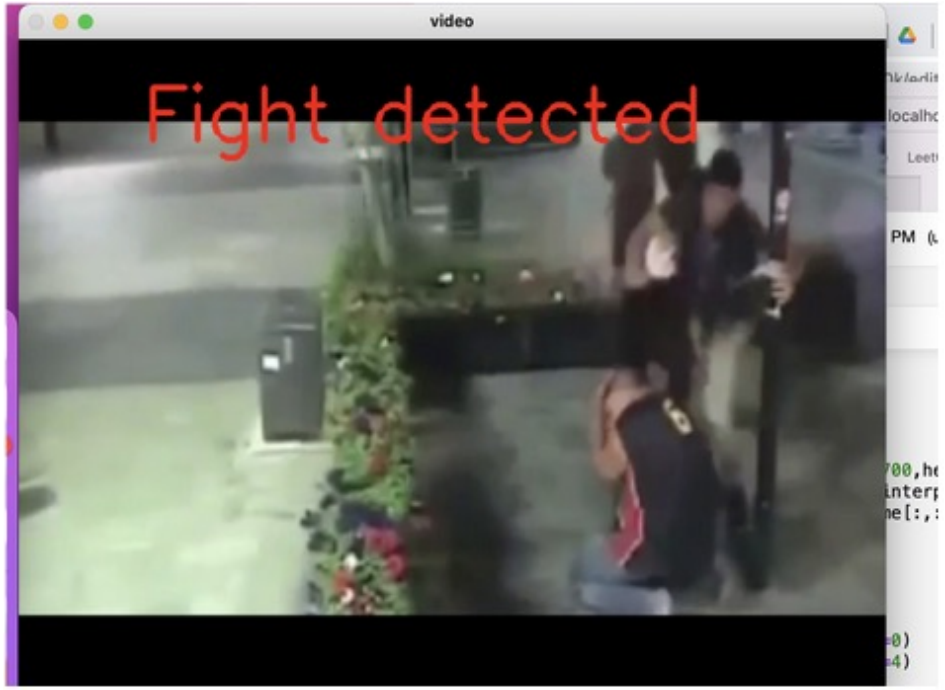
Machine Learning Results

Accuracy: When we are detecting the fight sequence in a video, we are getting a frame with a loss greater than 0.00068. The accuracy of the model is 78% currently and we are working on improving the same.

```
307/307 [==============================] - 73s 237ms/step - loss: 0.0229 - accuracy: 0.7811
Epoch 49/50
307/307 [==============================] - ETA: 0s - loss: 0.0228 - accuracy: 0.7811
```

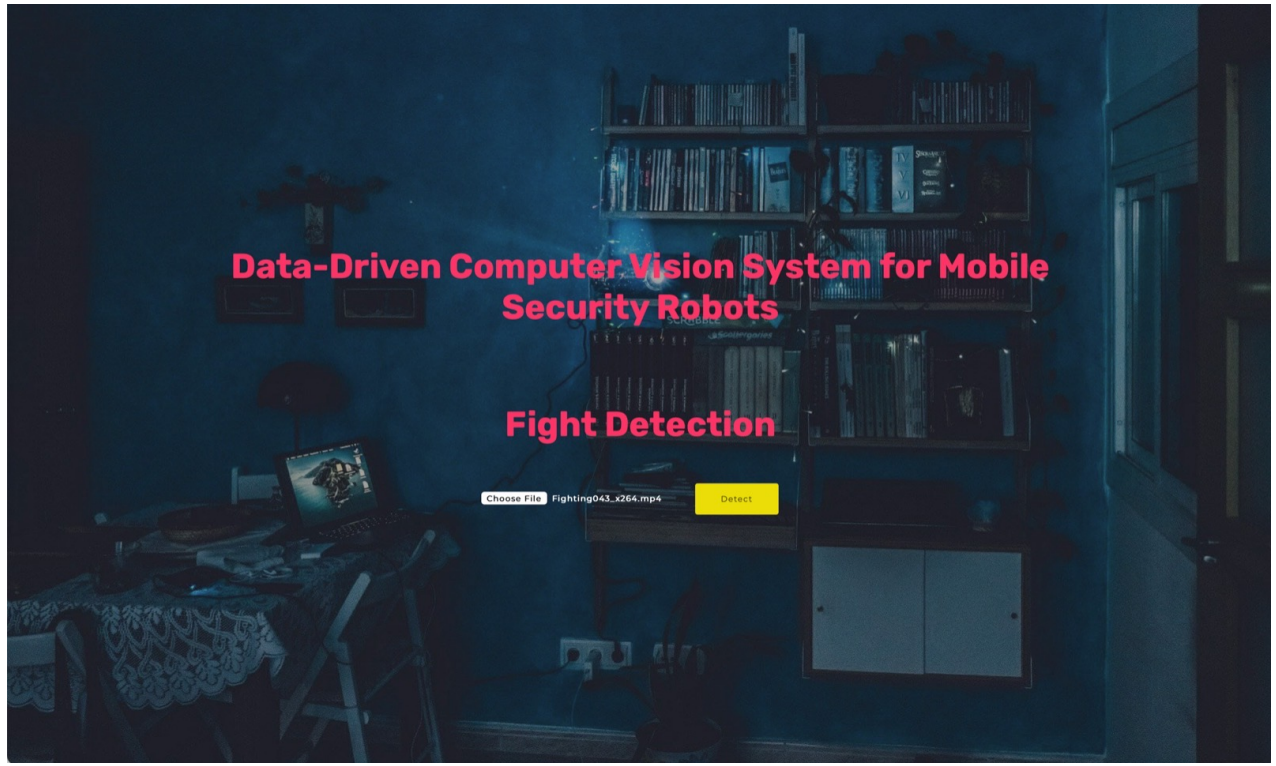
Predicted Output

The following image shows the predicted output from a testing video where the fight is detected within a frame and an overlapping text, 'Fight Detected' is generated.

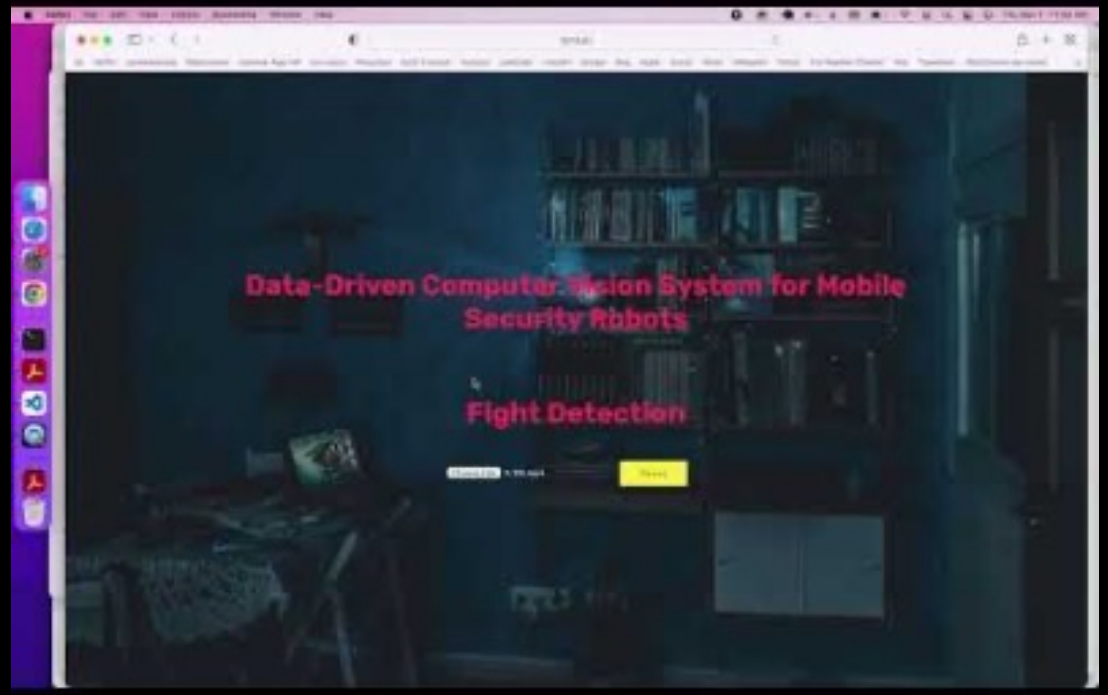


Web-based GUI

Created a webpage where the video that needs to be analyzed can be given as input and it shall detect if or not a fight is taking place in each frame in the video.



Demo Video


Data-Driven Computer Vision System for Mobile Security Robots
Fight Detection

License Plate Recognition and Detection Model

Data Engineering

Machine Learning Model

Machine Learning Results

Demo Video

Data Collection and Pre-processing

- Firstly, the open-source image dataset of YOLOv5 containing images and XML files of license plates. This image dataset contains 400 images and 400 annotation files in the XML format. The dataset consists of images and XML files.
- It is then being copied to YOLOv5.
- Preparing the dataset and putting the image from 'yolo data' folder to 'train' folder for training and 'val' folder for validation.
- Finally, the annotations have been performed where the XML file is getting converted to txt file and images to jpg format.
- The definition function of the bounding box is applied where XML files are converted to txt files.

No. of Classes

2

Total Number of Images Used for Training

413

Total Number of Annotated Images Used for Training

413

Number of Testing Videos

2

Total Number of Frames from Testing Video

18,000

Feature Extraction

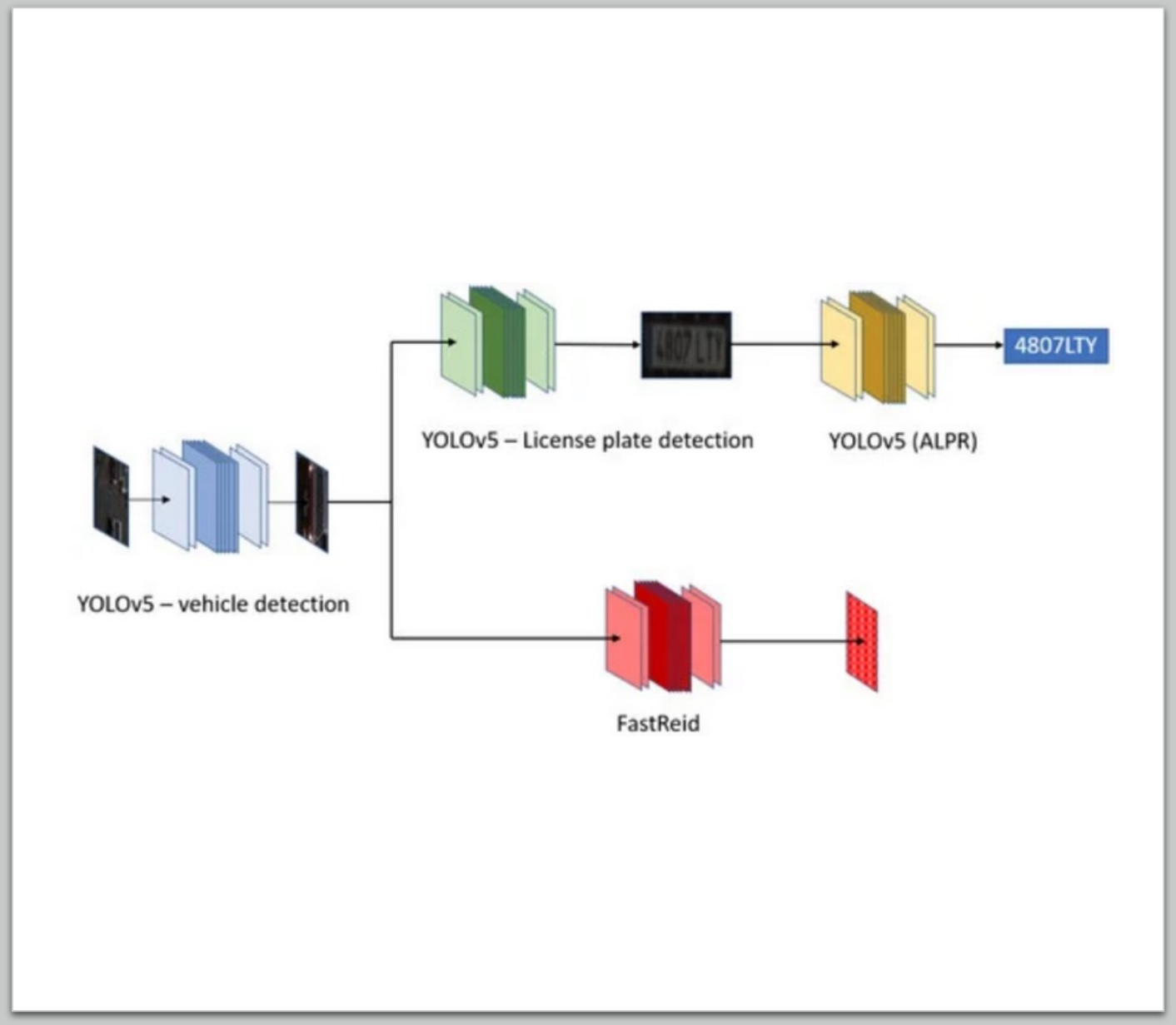
- We cannot use this raw images data for model training. So, prior to that we are going to read the images for license plates only and then crop that part of the image to use for model training.
- After cropping the license plate images in the next step, we are also going to filter out the bad images which have brightness and contrast problems and will only be keeping the good proportion as labels.
- We are generating the labels artificially and in that the input is the image while the label is clear.

ML Model – YOLOv5

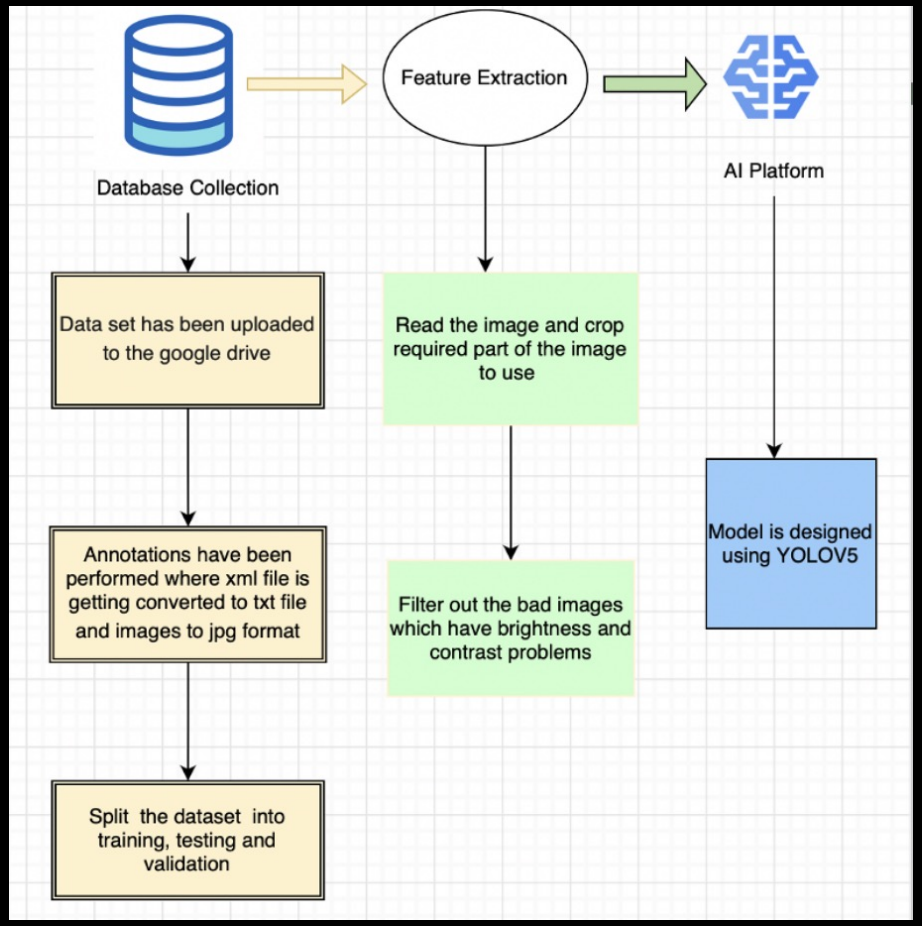
- YOLOv5 makes use of a single neural network to the whole image, which splits the image up into regions and projects bounding boxes and probabilities for each region.
- These bounding boxes are weighted by the projected probabilities and finally, the model can make its detections. The data set has been uploaded to google drive.
- YOLOv5 is an object detector, which provides the bounding boxes for all required vehicles present in the image.
- Secondly it is responsible to perform license plate detection, along with character recognition, in a multi-stage method.
- We have trained the model using 100 epochs. Once training is done, use the 'best.pt' weights for predicting the license plates in the images.
- The prediction results have been achieved with good accuracy results during day-time and night-time, respectively.

ML Model Architecture

- ***First Stage***: An object detector, which provides the bounding boxes for all required vehicles present in the image.
- ***Second Stage***: A parallel branch which is responsible to perform license plate detection, along with character recognition, in a multi-stage method.
- ***Third Stage***: The FastReid is used to run vehicle re-identification.



Model Implementation Flow Diagram

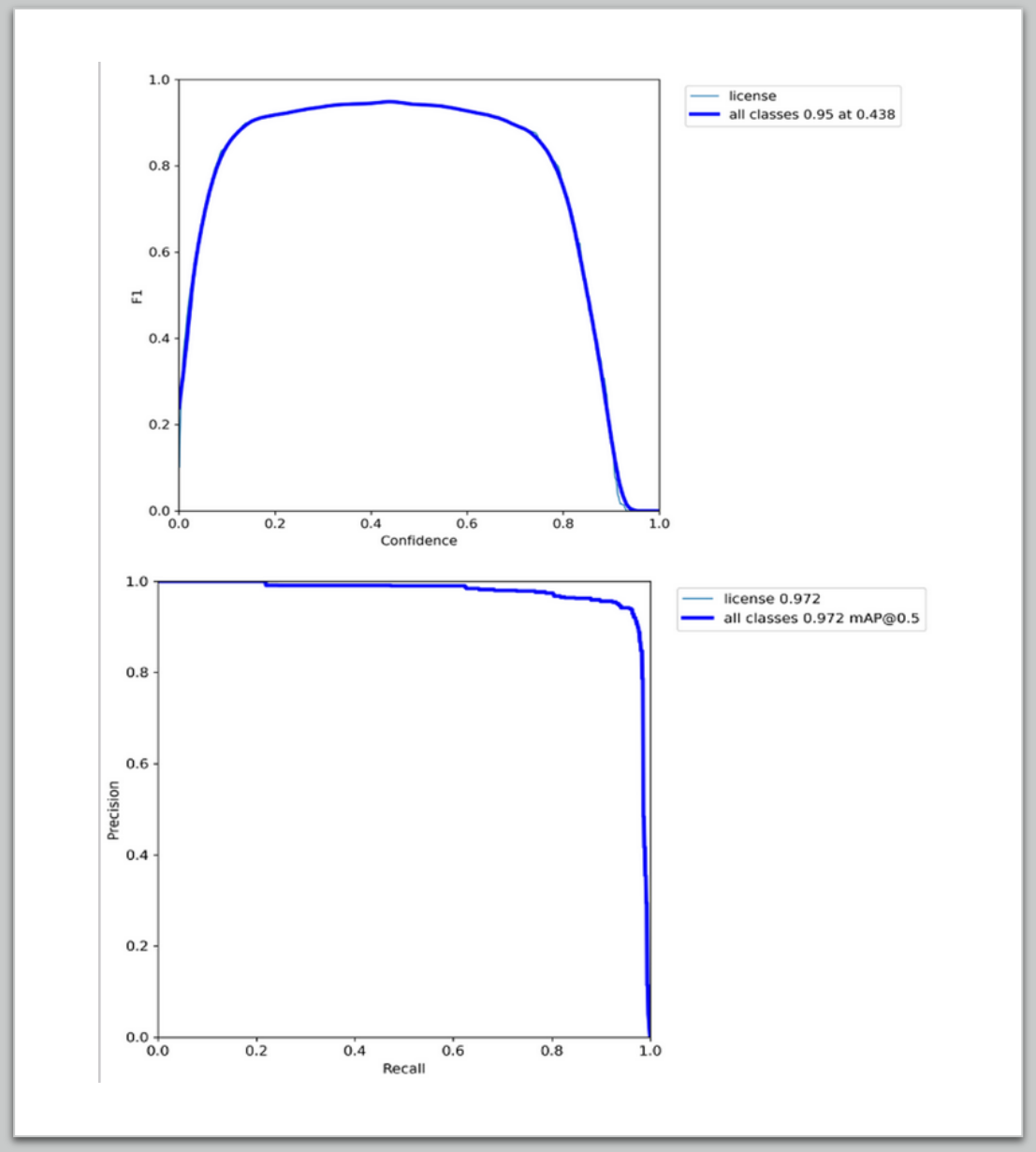


Why YOLOv5?

- Although the mean Average Precision (mAP) of Fastest R-NN is higher, YOLOv5 is much better since the frames per second, which is the detection speed, is nearly seven to eight times faster than Faster R-NN.
- Also, the size of the YOLOv5 model is much smaller than its predecessors, therefore making it much simpler to deploy in embedded devices.

Machine Learning Results

The values of precision parameter and recall for YOLOv5s are 0.902 and 0.99. The F1-score value is good at 0.95, respectively.



Accuracy of the Model for Image Dataset

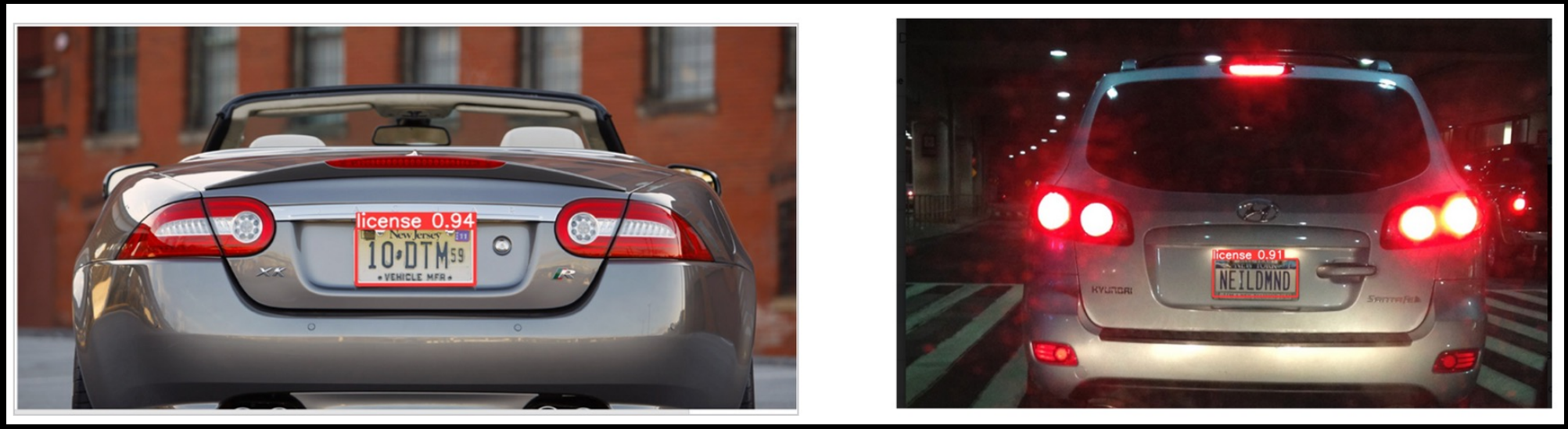
~ 94%

Accuracy of the Model for Video Dataset

~ 92%

Predicted Output

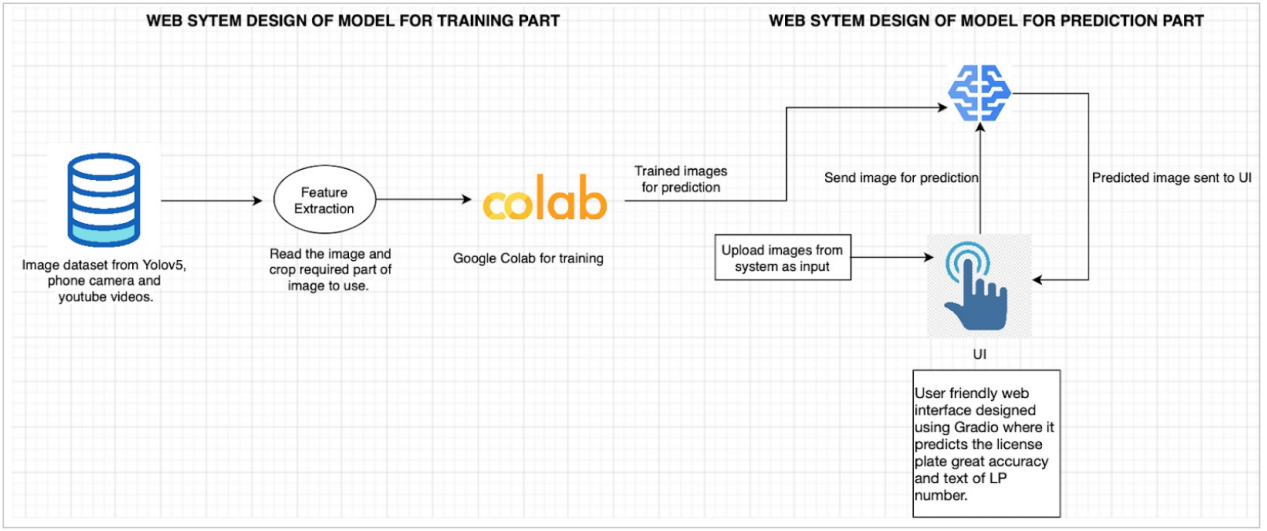
When the trained model runs on an image, the predicted output consists of a new image getting generated with a bounding box (if the object is detected) with class name i.e., plate, along with its class confidence value.



Predicted Output – With OCR

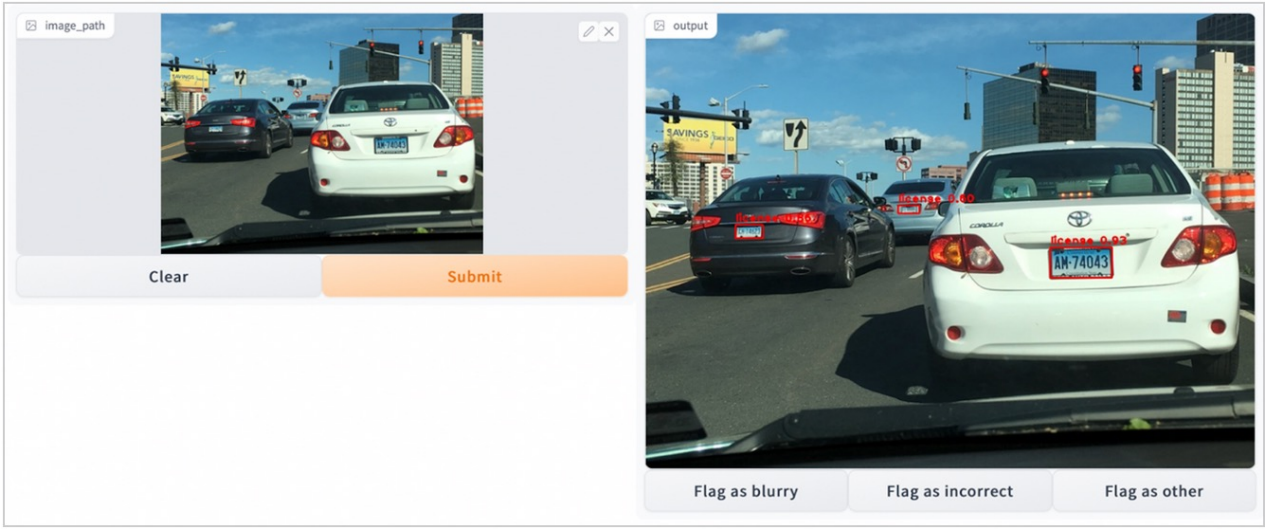
Web-based System Design

The web page has been developed using the "Gradio" library which is the fastest way to present our machine learning model with a user-friendly web interface so that anyone can use it and navigate it.



Web-based GUI

In the input we need to feed various images of vehicle license plates (day/night-time) or video containing cars with license plates and in output we get the license plate predicted with a bounding box with great accuracy.



Demo Video

Crime Audio Detection and Classification Model

Data Engineering

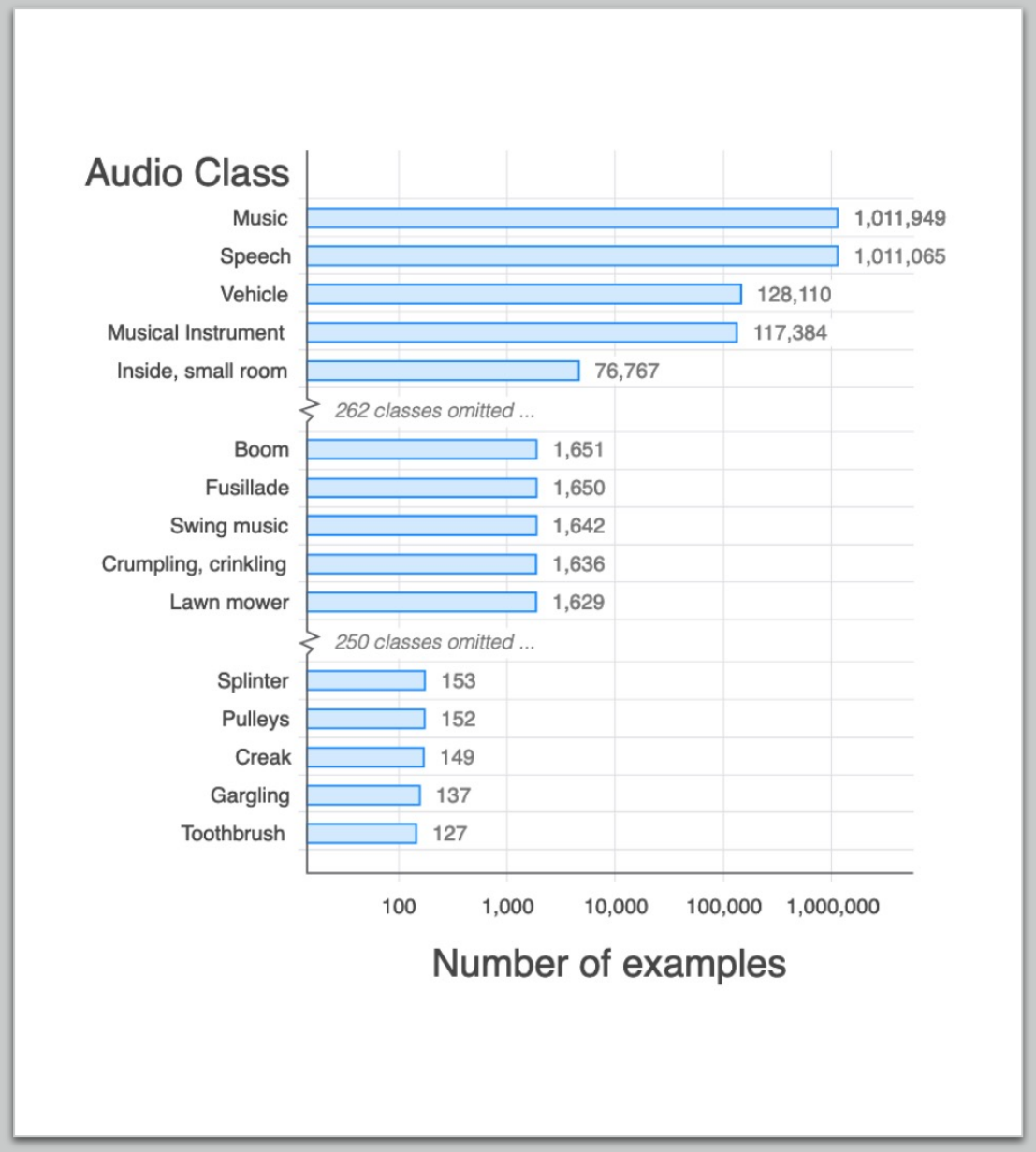
Machine Learning Model

Machine Learning Results

Demo Video

Dataset

- Previous audio detection models worked on specific datasets such as UrbanSound8K or ESC-50 Dataset all of which are limited in terms of the number of audio classes that can be recognized or detected.
- The model used for this project is a pre-trained YAMNet model with MobileNetV1 architecture that has been trained using the Google Audioset.



No. of Audio Classes

527

Total Number of Videos Used for Training

2.1 Million

Total Number of Main Classes

7

Number of Main Sub-Classes

43

Ontology of Google AudioSet

Human sounds

- Human voice
- Whistling
- Respiratory sounds
- Human locomotion
- Digestive
- Hands
- Heart sounds, heartbeat
- Otoacoustic emission
- Human group actions

Source-ambiguous sounds

- Generic impact sounds
- Surface contact
- Deformable shell
- Onomatopoeia
- Silence
- Other sourceless

Animal

- Domestic animals, pets
- Livestock, farm animals, working animals
- Wild animals

Sounds of things

- Vehicle
- Engine
- Domestic sounds, home sounds
- Bell
- Alarm
- Mechanisms
- Tools
- Explosion
- Wood
- Glass
- Liquid
- Miscellaneous sources
- Specific impact sounds

Music

- Musical instrument
- Music genre
- Musical concepts
- Music role
- Music mood

Natural sounds

- Wind
- Thunderstorm
- Water
- Fire

Channel, environment and background

- Acoustic environment
- Noise
- Sound reproduction

Data Pre-processing – Audio Extraction

We copy paste the URL of the video which we wish to extract and recognize the audio from. We can also save any video to our Google Drive and provide the path of the video for audio extraction.

```
[ ]  import moviepy.editor as mp

     #Please enter the name of the video file that is downloaded into the drive

     my_clip = mp.VideoFileClip(r"woman_screams.webm")
     my_clip

     <moviepy.video.io.VideoFileClip.VideoFileClip at 0x7fc8a2884110>

     # Extracted the audio from the video and writing it into the file

     my_clip.audio.write_audiofile(r"ws_audio.wav") # Give a name for the audio file

     [MoviePy] Writing audio in ws_audio.wav
     100%|██████████| 1843/1843 [00:01<00:00, 1796.28it/s][MoviePy] Done.
```

Formatting the Audio File

The extracted audio is then converted from stereo to audio as the audio input requirement for the YAMNet model is it should be a mono WAV format with a sample rate of 16kHz. The audio file is then displayed to be played.

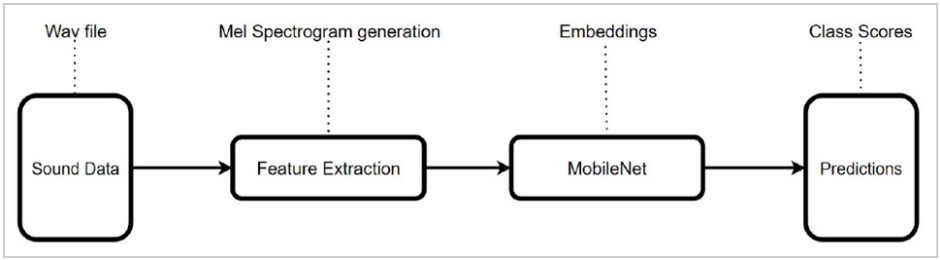
.

```
import scipy
#import scipy.io.wavfile
#wav_file_name = 'speech_whistling2.wav'
wav_file_name = 'ws_audio_mono.wav'
sample_rate, wav_data = wavfile.read(wav_file_name, 'rb')
sample_rate, wav_data = ensure_sample_rate(sample_rate, wav_data)

# Show some basic information about the audio.
duration = len(wav_data)/sample_rate
print(f'Sample rate: {sample_rate} Hz')
print(f'Total duration: {duration:.2f}s')
print(f'Size of the input: {len(wav_data)}')

# Listening to the wav file.
Audio(wav_data, rate=sample_rate)
```

```
Sample rate: 16000 Hz
Total duration: 83.56s
Size of the input: 1336960
```

ML Model – YAMNet with MobileNetV1

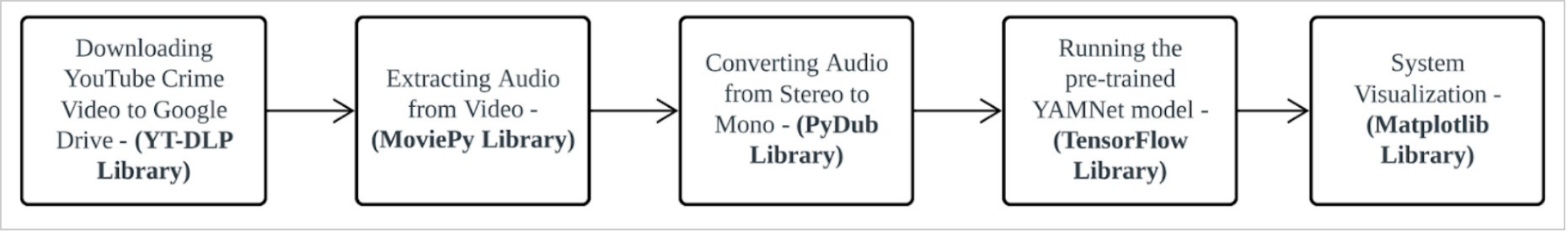
- For our project, a YAMNet pre-trained model has been used for audio recognition and detection. The architecture used within the YAMNet model is MobileNetV1.
- The MobileNetV1 makes use of a newer technology called the Depth wise Separable Convolutional Layers.
- A feature extraction layer has been embedded into the model, therefore eliminating the need for developing a separate feature extraction module.
- The feature extraction layer converts the audio data to spectrograms for audio detection.



ML Model Implementation

- The mono WAV format audio input is then fed into the pre-trained YAMNet with MobileNetV1.
- The model extracts the Mel-Frequency Cepstral Coefficients (MFCCs) of the audio file and calculates the scores of the top 5 classes. It displays the main class under which the audio files.
- The waveform, spectrogram and the score matrix are displayed.

Model Implementation Flow Diagram



Why YAMNet with MobileNetV1?

- MobileNetV1, as stated before, uses newer technology called Depthwise Separable Convolutional Layers.
- Due to the usage of these layers, the model is smaller in size compared to other CNN models which makes them ideal to implement within mobile and embedded systems.
- While ResNet50 has a better accuracy but by increasing the number of training epochs, the accuracy of MobileNetV1 can be improved.

Machine Learning Results

- Two codes have been written, one in which the YAMNet model has been embedded for feature extraction in combination with CNN, where we can see the accuracy is quite high. The model was run for 50 and 100 epochs, and the accuracy of the model improved from 96.66% to 98.33%.
- The inference time was on average 45 seconds.

```
loss, accuracy = my_model.evaluate(test_ds)

print("Loss: ", loss)
print("Accuracy: ", accuracy)
```

```
8/8 [==============================] - 0s 7ms/step - loss: 0.0370 - accuracy: 0.9833
Loss:  0.0369834266602993
Accuracy:  0.9833333492279053
```

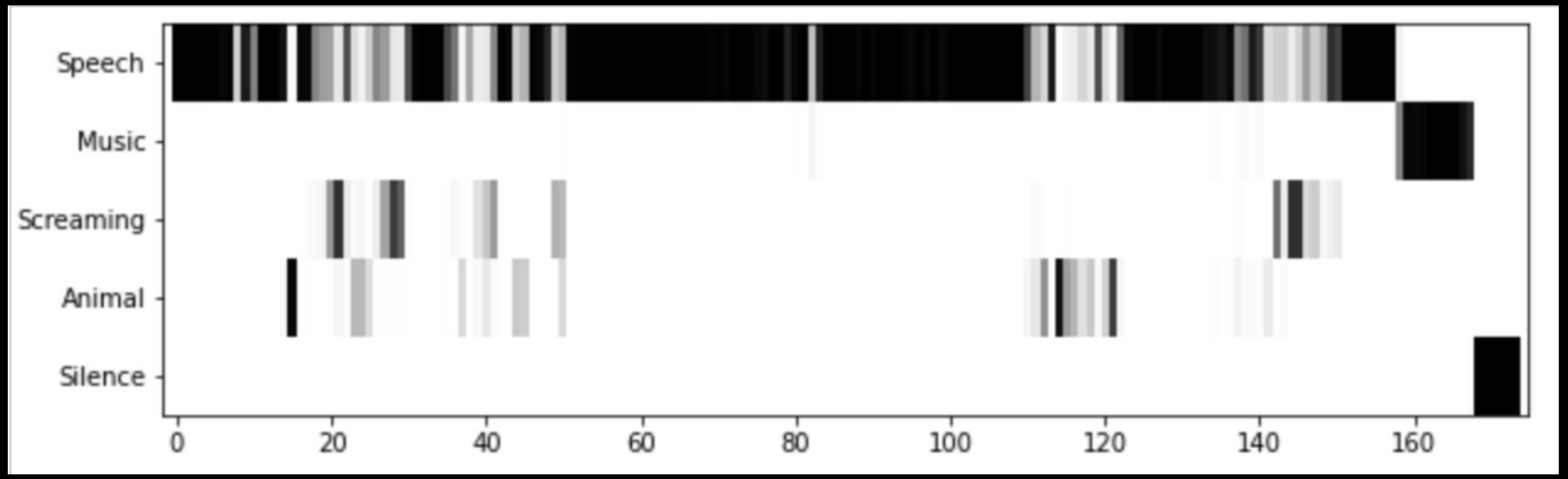
Predicted Output – Inferred Class

The main class and subclasses should be identified despite background noise.

```
[ ]  scores_np = scores.numpy()
     spectrogram_np = spectrogram.numpy()
     infered_class = class_names[scores_np.mean(axis=0).argmax()]
     print(f'The main sound is: {infered_class}')

     The main sound is: Speech
```

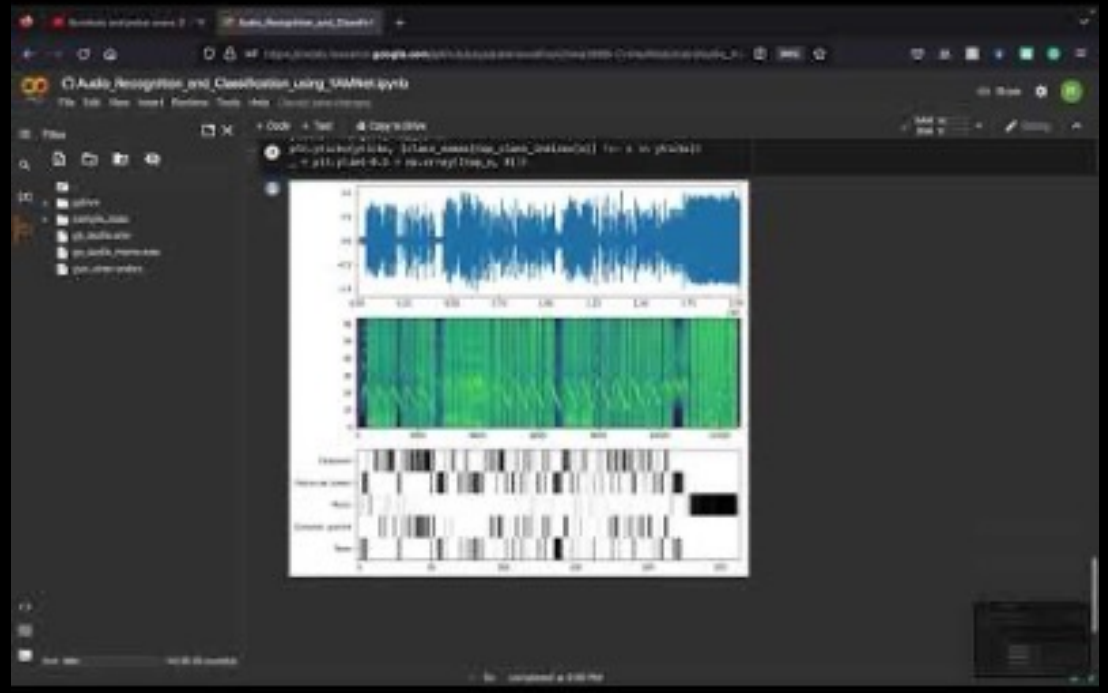
Predicted Output – Score Matrix

In the second code, we have used the pre-trained YAMNet model to detect and classify the audio and we are able to identify the top 5 scores with an accuracy of 95%.



Demo Video


ring.com
INSIDE
Woman Screams for Her Life on Doorbell Camera Footage
Subscribe



Thank You