
Driver Alertness
Detection
System
Group 7

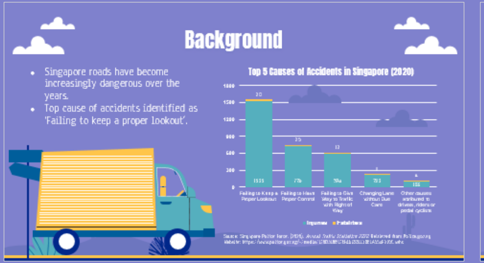
Project Team

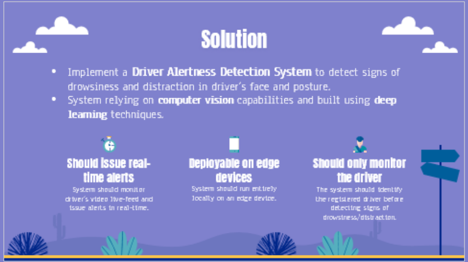
Hwang Sion
A0249263Y

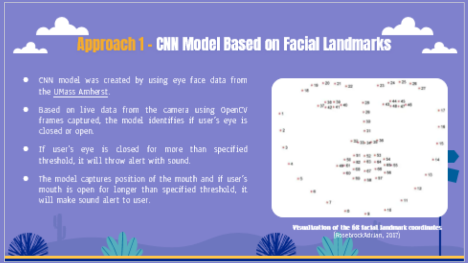
Prerak Agarwal
A0116711R

Santi
A0249294R

Zhang Junfeng
A0249266U

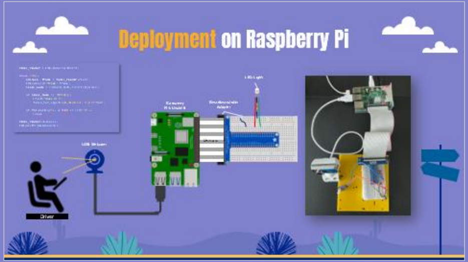

Background
Singapore roads have become increasingly dangerous over the years.
Top cause of accidents identified as 'Failing to keep a proper lookout'.
Top 5 Causes of Accidents in Singapore (2020)
Failing to Keep a Proper Lookout
Failing to Have Proper Control
Failing to Give Way to Traffic with Right of Way
Changing Lane without Due Care
Other causes attributed to drivers, riders or pedal cyclists

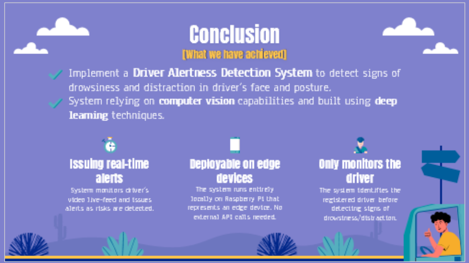

Solution
Implement a Driver Alertness Detection System to detect signs of drowsiness and distraction in driver's face and posture.
System relying on computer vision capabilities and built using deep learning techniques.
Should issue real-time alerts
System should monitor driver's video live-feed and issue alerts in real-time.
Deployable on edge devices
System should run entirely locally on an edge device.
Should only monitor the driver
The system should identify the registered driver before detecting signs of drowsiness/distraction.


Approach 1 - CNN Model Based on Facial Landmarks
CNN model was created by using eye face data from the UMass Amherst.
Based on live data from the camera using OpenCV frames captured, the model identifies if user's eye is closed or open.
If user's eye is closed for more than specified threshold, it will throw alert with sound.
The model captures position of the mouth and if user's mouth is open for longer than specified threshold, it will make sound alert to user.
Visualization of the 68 facial landmark coordinates
(RosebrockAdrian, 2017)


Approach 2 - CNN based distraction detection
[Image Dataset]
Front View
Side View
Alert
Drowsy
Data Source
Description
Type
Data Size (No. of Samples)
Data Size


Approach 3 - CNN+RNN
[Architecture]
ConvLSTM
Convolutional Long Short-Term Memory
LRCN
Long-term Recurrent Convolutional Network

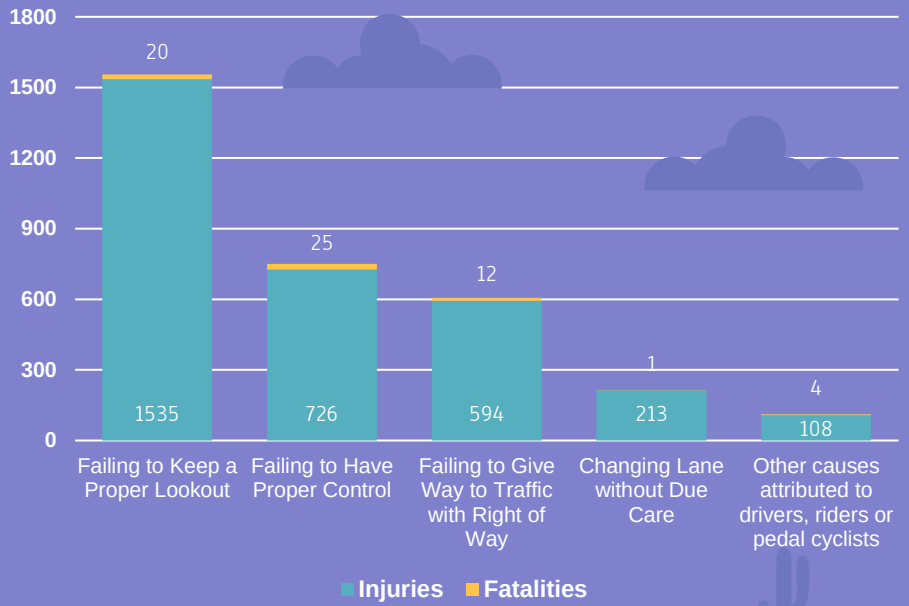

Deployment on Raspberry Pi
Driver


Conclusion
[What we have achieved]
Implement a Driver Alertness Detection System to detect signs of drowsiness and distraction in driver's face and posture.
System relying on computer vision capabilities and built using deep learning techniques.
Issuing real-time alerts
System monitors driver's video live-feed and issues alerts as risks are detected.
Deployable on edge devices
The system runs entirely locally on Raspberry Pi that represents an edge device. No external API calls needed.
Only monitors the driver
The system identifies the registered driver before detecting signs of drowsiness/distraction.

Background

- Singapore roads have become increasingly dangerous over the years.
- Top cause of accidents identified as ‘Failing to keep a proper lookout’.

Top 5 Causes of Accidents in Singapore (2020)



Source: Singapore Police Force. (2020). *Annual Traffic Statistics 2020.* Retrieved from Police.gov.sg Website: https://www.police.gov.sg/-/media/170D31BB17EF441881138E1A556F210C.ashx

Problem Statement

How can we implement a Pattern Recognition System to improve the safety of drivers on the road?

Solution

- Implement a **Driver Alertness Detection System** to detect signs of drowsiness and distraction in driver's face and posture.
- System relying on **computer vision** capabilities and built using **deep learning** techniques.

Should issue real-time alerts

System should monitor driver's video live-feed and issue alerts in real-time.

Deployable on edge devices

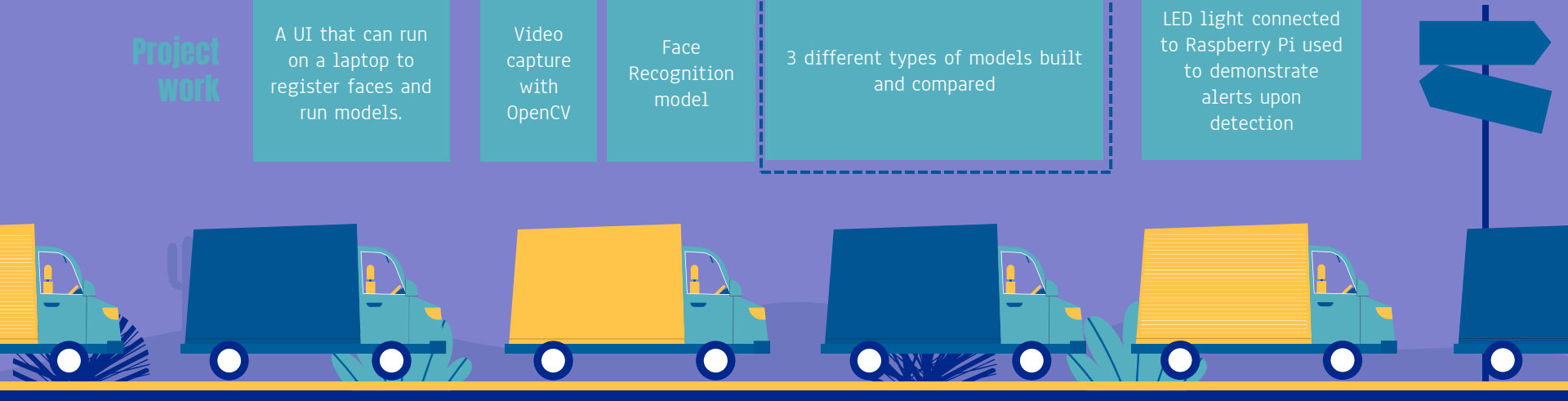
System should run entirely locally on an edge device.

Should only monitor the driver

The system should identify the registered driver before detecting signs of drowsiness/distraction.

System Architecture

	Setup	Monitor			Alert
Usage Scenario	Driver sets up mobile app (ride hailing or map apps), and starts driving	Videos streams through phone camera	App detects and locates driver's face	Drowsiness/distraction detected	An alert is triggered (sound and/or lights blinking)
Project work	A UI that can run on a laptop to register faces and run models.	Video capture with OpenCV	Face Recognition model	3 different types of models built and compared	LED light connected to Raspberry Pi used to demonstrate alerts upon detection



Technology Stack

Driver Alertness Detection System

Model Categories

Approach 1 – Facial Landmarks	
Architecture	CNN
Classes	Eyes open/close Mouth open ratio
Number of models built	5

Approach 2 – Body Postures	
Architecture	CNN
Classes	11 Body Postures
Number of models built	5

Approach 3 – Drowsy Face	
Architecture	CNN + LSTM
Classes	Alert Drowsy
Number of models built	2

Face Identification	
Architecture	CNN
Classes	Registered Face detected or not
Number of models built	2

Software

- Tensorflow
- Keras
- Numpy
- Sklearn
- Matplotlib
- Pandas
- Seaborn
- PathLib
- pygame
- Face_Recognition
- Face_Recognition
- imutils
- OpenCV
- Tensorflow-metal for GPU in macOS

- Tflite-runtime
- Numpy
- Facial-recognition
- OpenCV
- Tkinter
- dlib

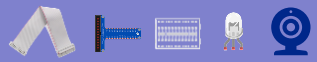
Hardware

Colab Pro+

Mac Studio

Laptop

Raspberry Pi 4 Model B



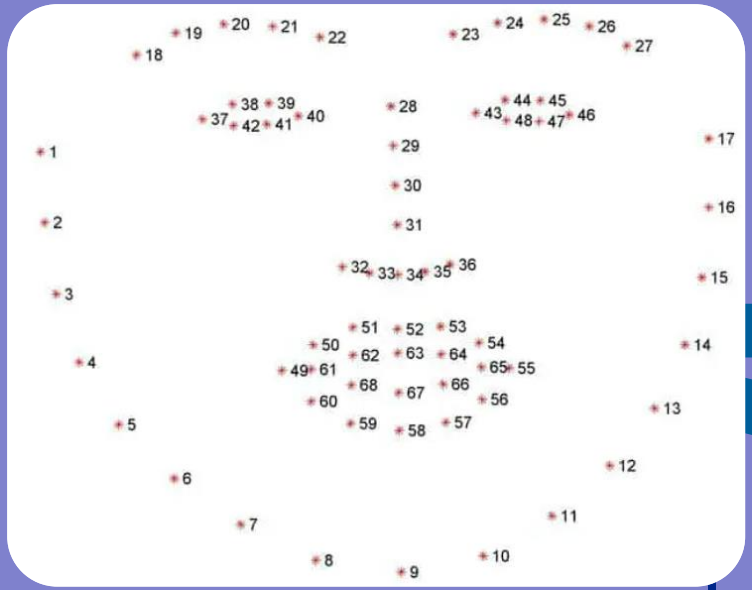
Raspberry Pi Extensions

Training

Deployment

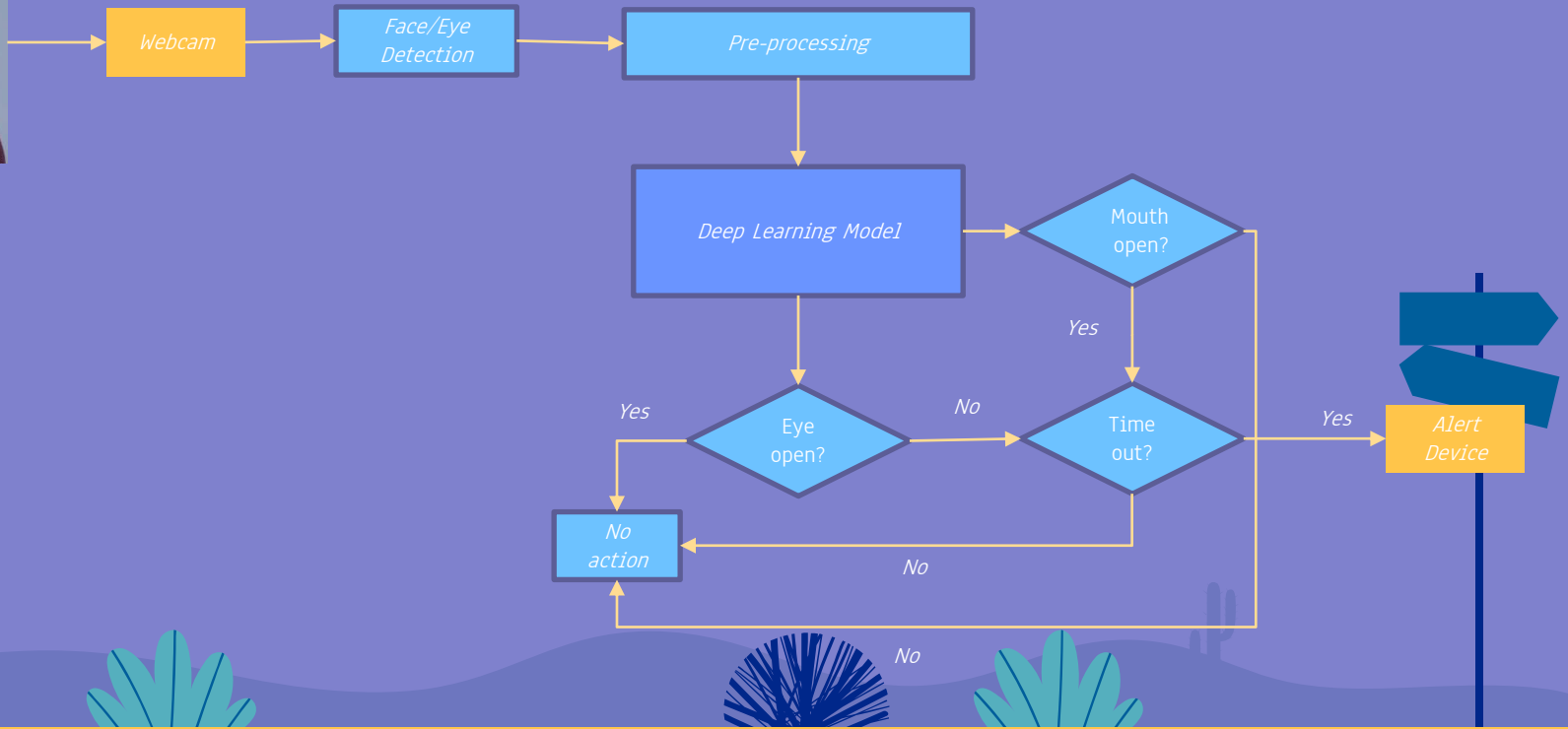
Approach 1 - CNN Model Based on Facial Landmarks

- CNN model was created by using eye face data from the UMass Amherst.
- Based on live data from the camera using OpenCV frames captured, the model identifies if user's eye is closed or open.
- If user's eye is closed for more than specified threshold, it will throw alert with sound.
- The model captures position of the mouth and if user's mouth is open for longer than specified threshold, it will make sound alert to user.



Visualization of the 68 facial landmark coordinates
(RosebrockAdrian, 2017)

Approach 1 - CNN Model Based on Facial Landmarks (Cont.)



Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

Different ways to create the models were experimented to find optimal model

1. *CNN Model* trained from scratch
2. *CNN Model* trained with optimisation (**Batch normalization/drop-out**)
3. Transfer learning with *Xception Model*
4. Transfer learning with *InceptionV3*
5. Transfer learning with *ResNet50*
6. Transfer learning with *MobileNet*

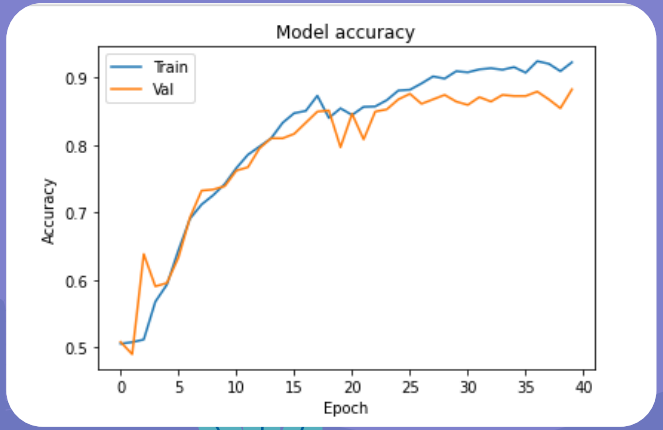
Below dataset is used as common dataset for all training models.

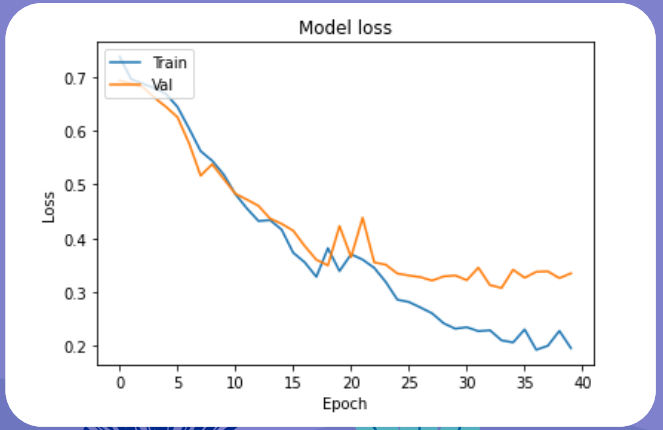
Data Source	Description	Type	Data Size (No. of Samples)	Data Size
UMass Amherst dataset	Open/Closed eye images	Images	1231 open eye Images and 1192 closed eye images with 24x24 pixels.	1MB

Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[CNN Model trained from scratch and optimisation]

- This model has 5 convolutional layers with 'ReLU' activation function for feature extraction and max pooling is applied. Sigmoid activation function is used in last layer as eye open status is either 'Open' = 1 or 'Closed' = 0.'
- This model has proved to have good validation accuracy of 88% to run in Raspberry pi.
- The model didn't have any anomaly or spikes in terms of valuation or loss showing consistent increment of model accuracy.

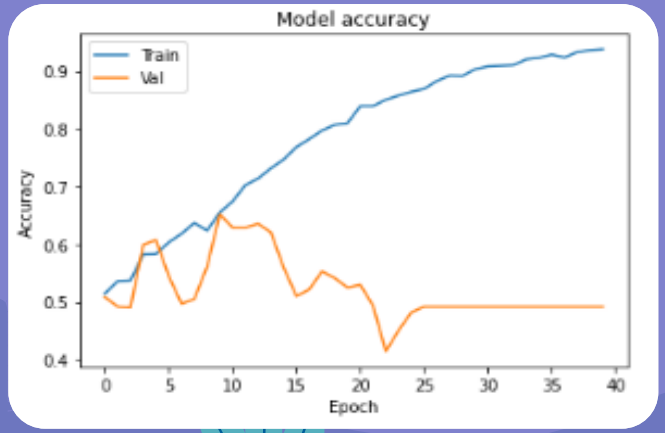


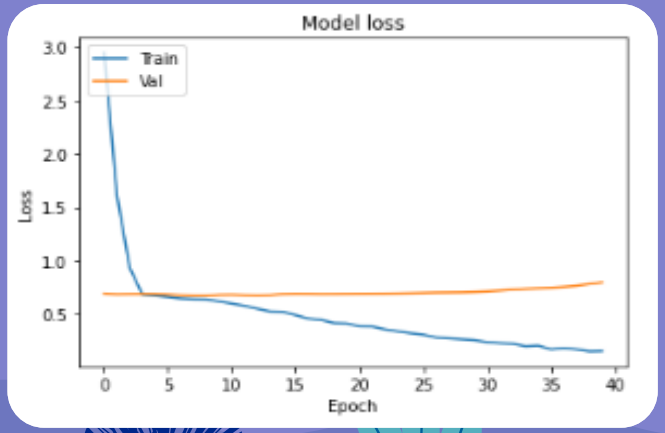


Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[CNN model with Batch Normalization & Drop-out]

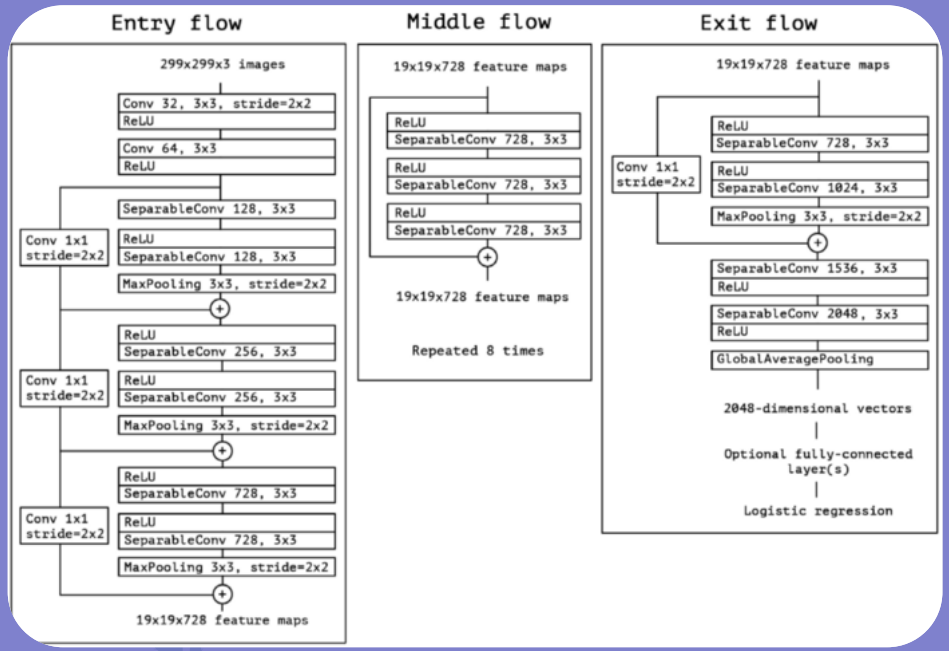
- From the 1st model, batch normalization and dropout are applied to further optimize the model.
- Batch normalization is applied to standardize the layer inputs for every mini-batch. We expected this will stabilize the learning process and reduce the number of epochs required to train deep network.
- Adding batch normalization reduced validation accuracy by 49% and validation accuracy didn't improve even with 40 epoch of training. So, we have concluded that it is case by case to use batch normalization & drop-out to improve existing models.





Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[Xception Model]



Overall Architecture of Xception

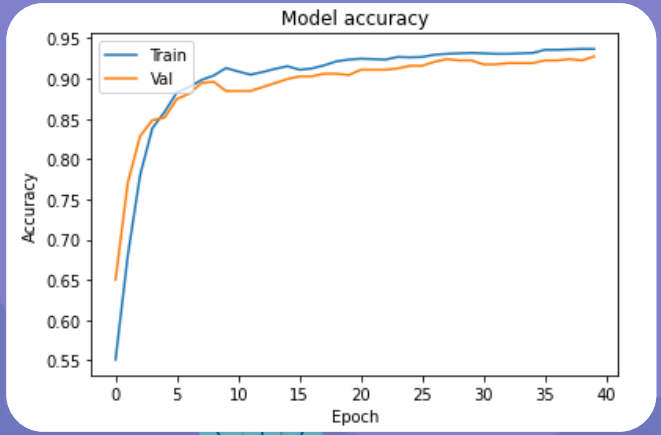
(TsangSik-Ho, Review: Xception – With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification), 2018)

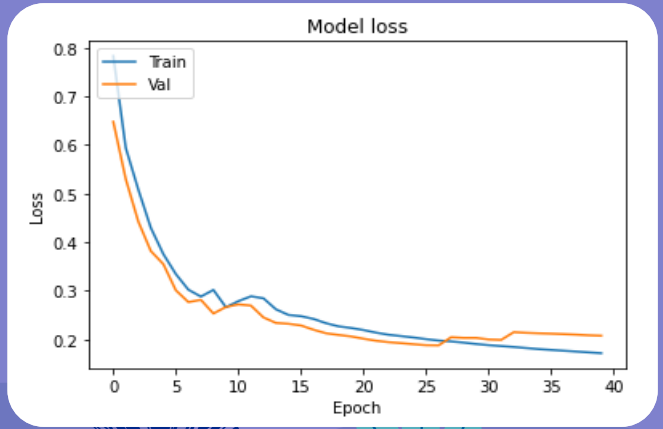
- Xception model which is created by google is abbreviation of 'Extreme version of Inception'.
- SeparableConv is the modified depth-wise separable convolution which performs pointwise convolution first before doing a depth-wise convolution.
- These SeparableConv are used as inception modules throughout the deep learning architecture.

Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[Xception Model]

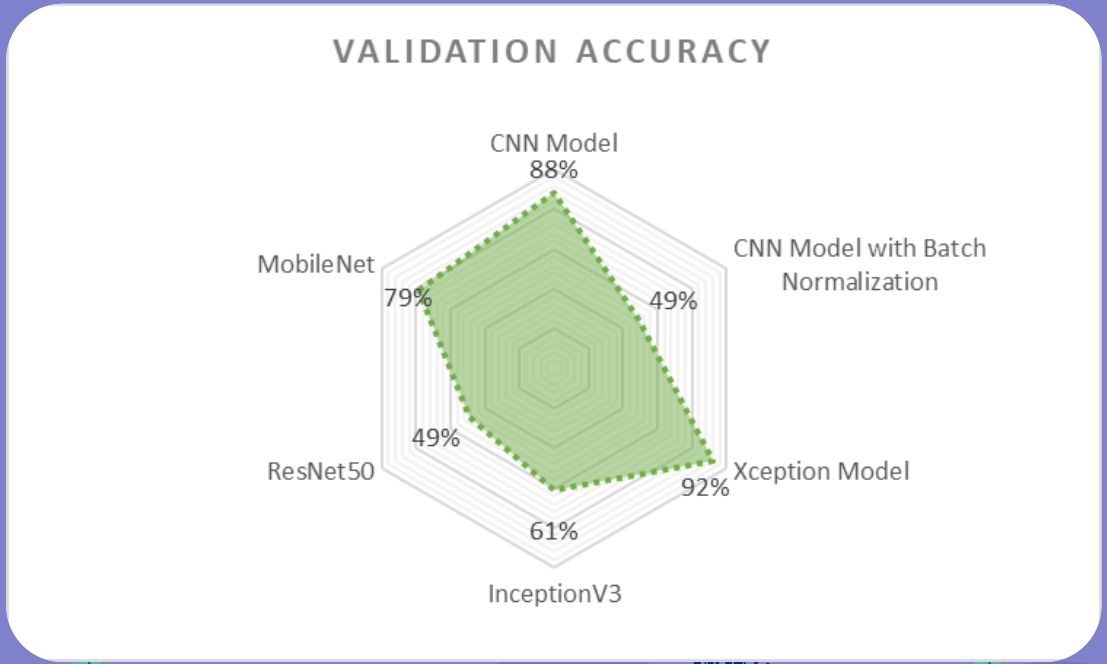
- Total number of parameters was above 20 million and only last few layers added was used as trainable parameters (around 2000 parameters) to utilize weights already learned in this model.
- With the moderate level of depth (81 layers) and parameters (22.9M), this model has managed to achieve best result throughout. This is thanks to efficient architecture which is built with 2 main concepts: 1) Depth-wise Separable Convolution 2) Shortcuts link between Convolution blocks like ResNet implemenation.





Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[Overall Models Assessment]



Approach 2 - CNN based distraction detection

[Image Dataset]

Front View

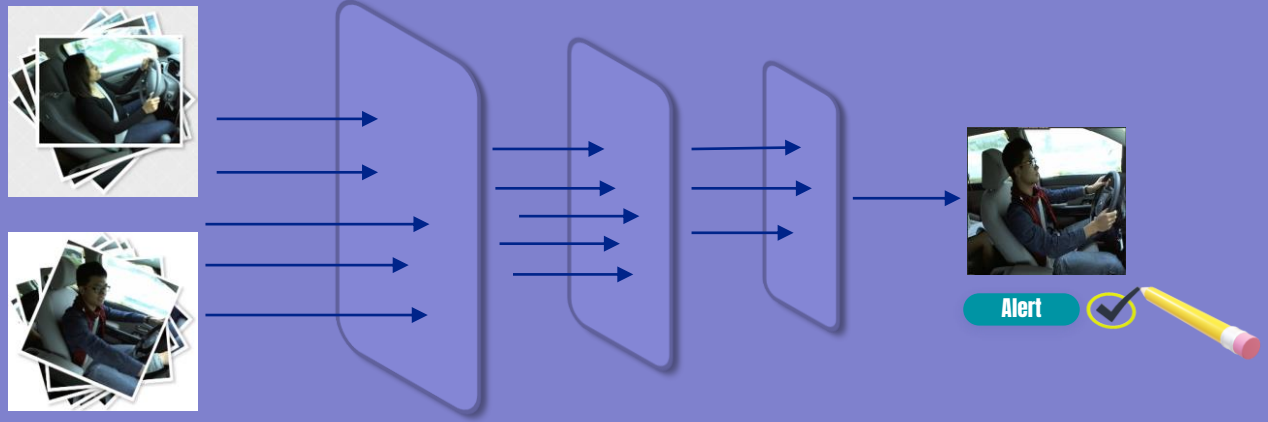
Alert

Drowsy

Side View

Data Source	Description	Type	Data Size (No. of Samples)	Data Size
National Tsing Hua University – Driver Drowsiness Detection Dataset	Dataset with 36 individuals in various simulated driving scenarios, including normal driving, yawning, slow blink rate, falling asleep, etc., under day and night illumination conditions.	Videos	Training + Validation – 90 videos (1-1.5mins each, total 9.5hrs) Testing – 90 videos	± 7.45GB
Kaggle.com - State Farm Distracted Driver Detection Dataset	State-farm distraction dataset. The data consist of 10 classes such as normal driving, talking on the phone, texting, makeup, drinking and others.	Images	22,424 (training + validation) 79,727 (testing – unlabelled)	± 4.52 GB

Approach 2 - CNN based distraction detection

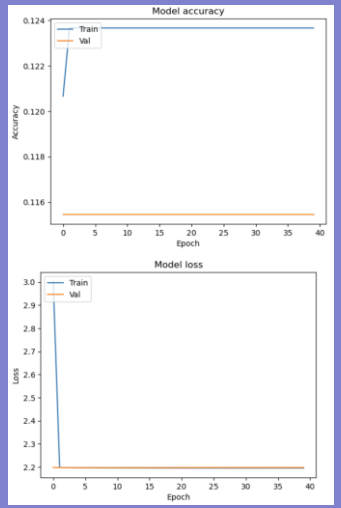


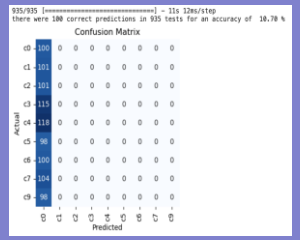
Data preparation	Model technique	Comparison
✓ Resize to 224 *224 ✓ Resize to 32 *32	✓ Simple model creation ✓ Transfer Learning	✓ Training accuracy ✓ Validation Accuracy ✓ Test Accuracy ✓ Testing with real life

Approach 2 - CNN based distraction detection

[Model creation and tuning]

1st creation





Model Tuning:

- ✓ Added depth-wise convolutional
- ✓ Increased dimentionality of the dense layer before last layer

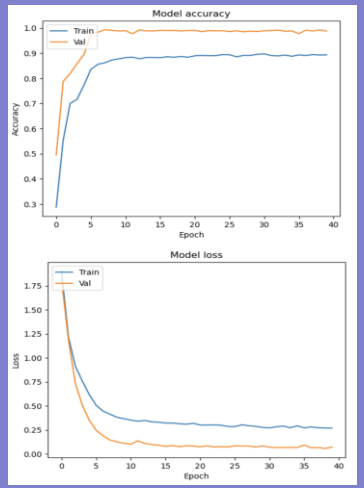
What is solved:

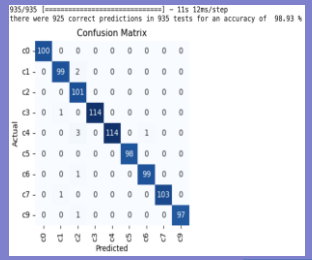
- ✓ Accuracy in training and test

What is not solved:

- ❑ Size of the model

After tuning





Transfer learning and it's Dos and Don'ts

[Tested Architecture]

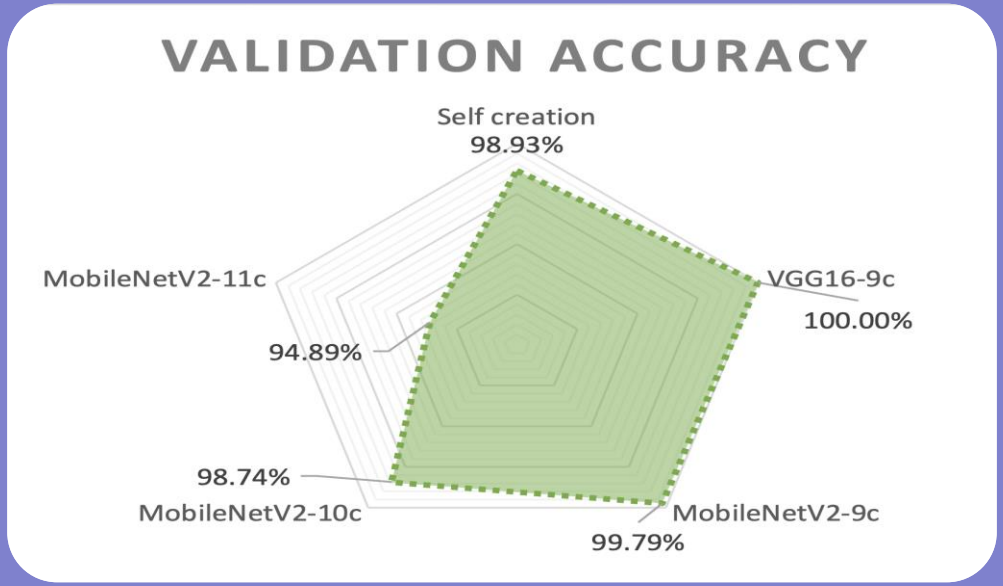
Architecture	Pro	Cons
✓ MobileNetV2	✓ Lightweight, suitable for edge device deployment ✓ Small in size	✓ Lower Accuracy compare with VGG16
✓ VGG16	✓ Suitable for large images ✓ High Accuracy	✓ Huge in size

[Learning]

Learning point of Transfer Learning	Avoid this when using transfer learning
✓ Find suitable architecture with our use case ✓ Training of most parameter with data will give a better accuracy ✓ Some architecture will require a certain input size ✓ Flexibility to add on or remove certain layers	✓ Find the most impressive model without fitting the use case ✓ Removing or adding important layers ✓ Removed trainable parameter too drastic

Approach 2 - CNN based distraction detection

[Overall Models Assessment]

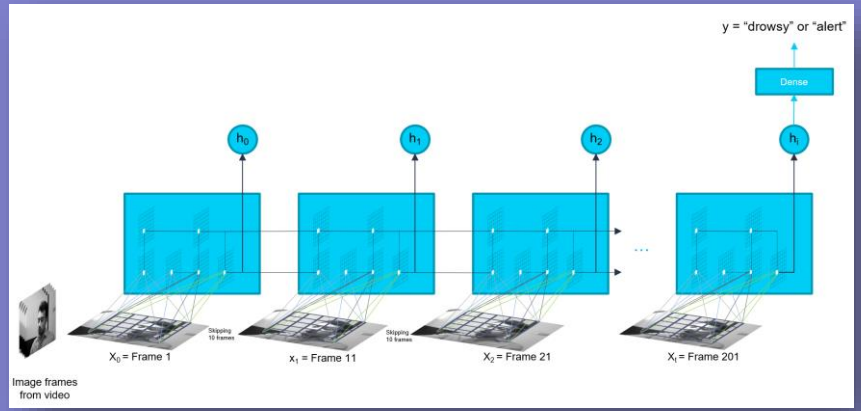


Approach 3 - CNN+RNN

[Architecture]

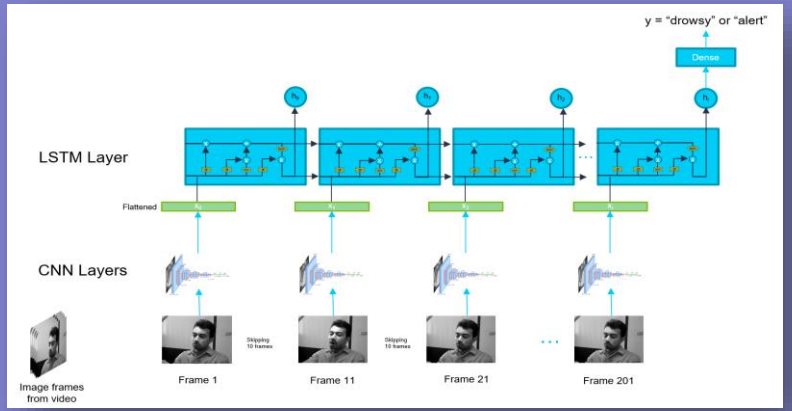
ConvLSTM

Convolutional Long Short-Term Memory



LRCN

Long-term Recurrent Convolutional Network



Approach 3 - CNN+RNN

[Data Prep]

sleepyCombination_drowsiness.avi

005_sleepyCombination_drowsiness.txt

Data Source	Description	Type	Data Size (No. of Samples)	Data Size
National Tsing Hua University - Driver Drowsiness Detection Dataset	Dataset with 36 individuals in various simulated driving scenarios, including normal driving, yawning, slow blink rate, falling asleep, etc., under day and night illumination conditions.	Videos	Training + Validation - 90 videos (1-1.5mins each, total 9.5hrs) Testing - 90 videos	± 7.45GB

Approach 3 - CNN+RNN

[Data Prep]

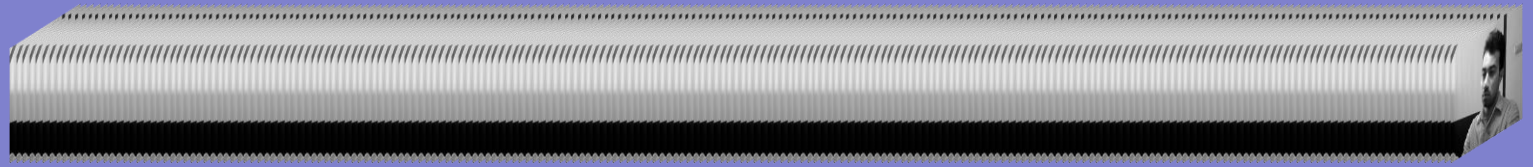
sleepyCombination_drowsiness.avi

005_sleepyCombination_drowsiness.txt

Frame labels

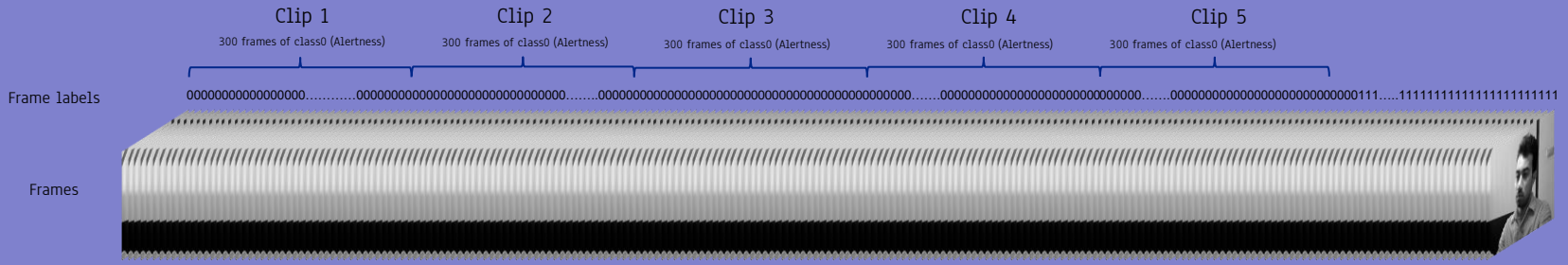
0000000000000000.....0000000000000000000000000000.....0000000000000000000000000000000000000000.....0000000000000000000000000000.....0000000000000000000000000000111.....1111111111111111111111

Frames



Approach 3 - CNN+RNN

[Data Prep]



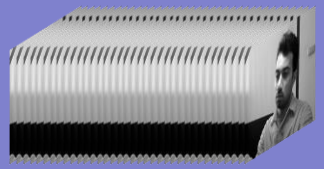
Approach 3 - CNN+RNN

[Data Prep]

Clip 2

300 frames of class0 (Alertness)

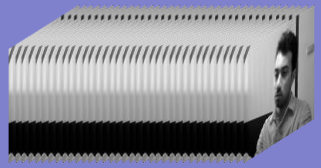
0000000000000000000000.....0000000



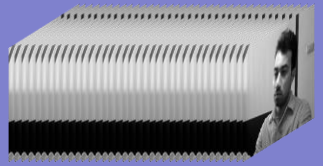
Clip 4

300 frames of class0 (Alertness)

000000.....00000000000000000000000



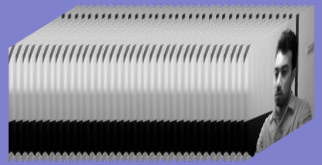
0000111.....11111111111111111111111



Clip 1

300 frames of class0 (Alertness)

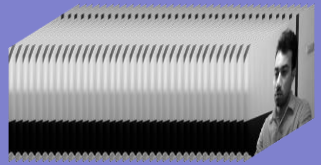
00000000000000000.....00000000



Clip 3

300 frames of class0 (Alertness)

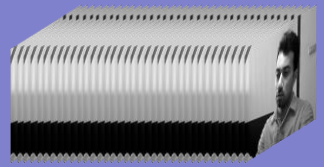
000000000000000000000000000000000



Clip 5

300 frames of class0 (Alertness)

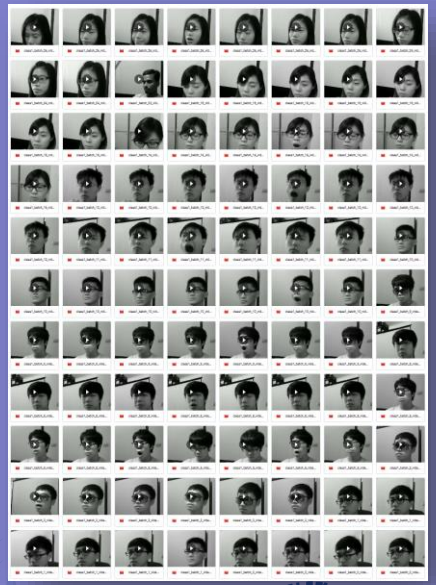
000000.....00000000000000000000000



Approach 3 - CNN+RNN

[Data Prep]

Class 1 - Drowsiness - 88 x 10-second videos



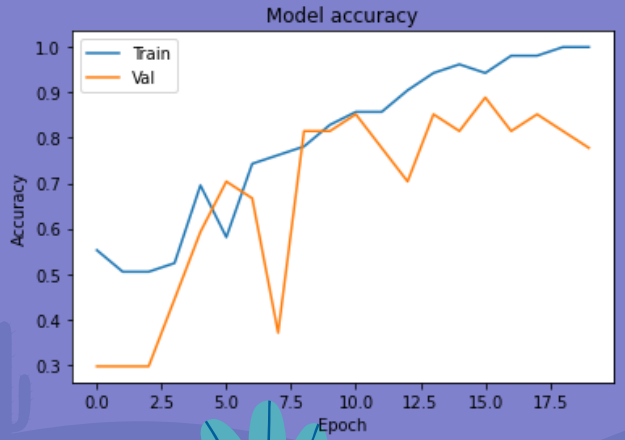
Class 0 - Alertness - 88 x 10-second videos

Approach 3 - CNN+RNN

[Performance]

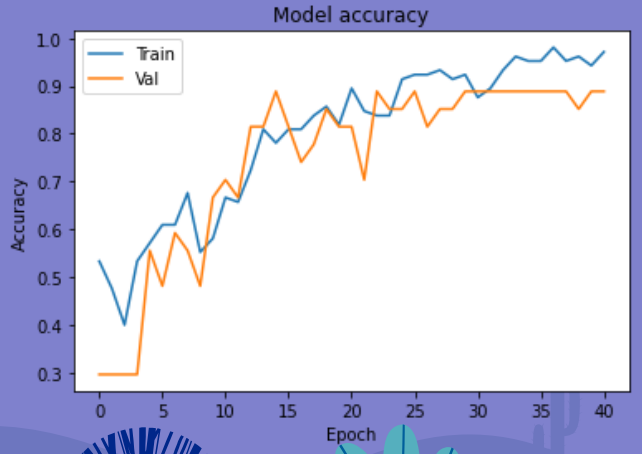
ConvLSTM

Approach 3 - ConvLSTM	
Sequence Length	20 out of 200 frames
Training Speed	438ms/step at batch_size = 4
Test Accuracy	93% (on test data)



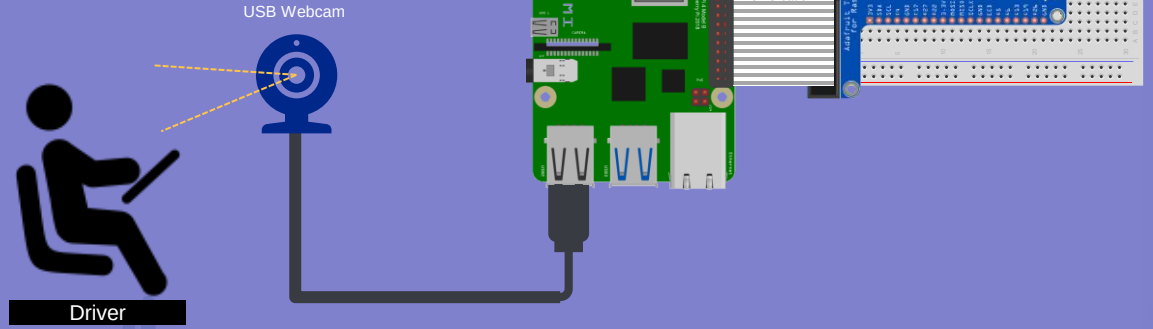
LRCN

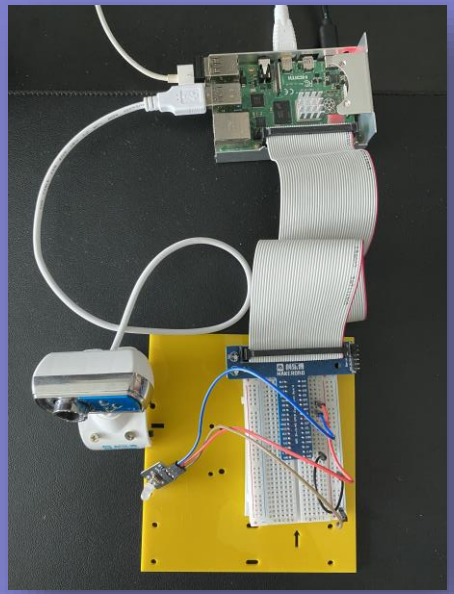
Approach 3 - LRCN	
Sequence Length	20 out of 200 frames
Training Speed	29 ms/step at batch_size = 4
Test Accuracy	84% (on test data)



Deployment on Raspberry Pi

```python
video_reader = cv2.VideoCapture(0)

while True:
    success, frame = video_reader.read()
    cv2.imshow("frame",frame)
    class_name = predict_with_ConvLSTM_model()

    if class_name == "Drowsy":
        print("wake up!")
        flash_red_light(num_times=3, color="red")

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_reader.release()
cv2.destroyAllWindows()
```





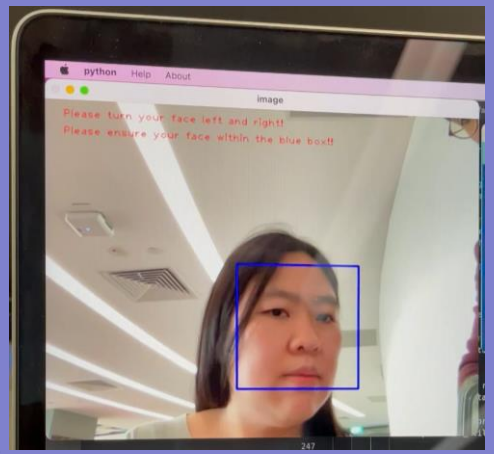
Deployment & Demo

Registration

Mixed

Sleepiness detection with face recognition

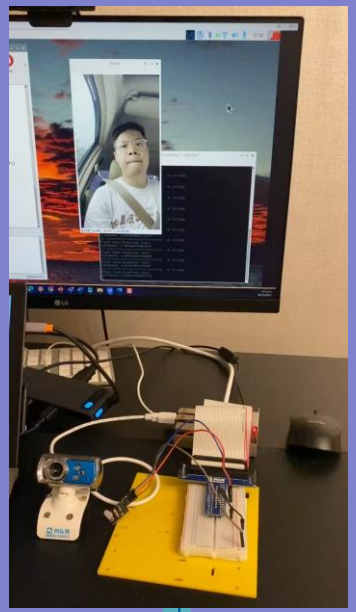
Recognized Face Unrecognize face



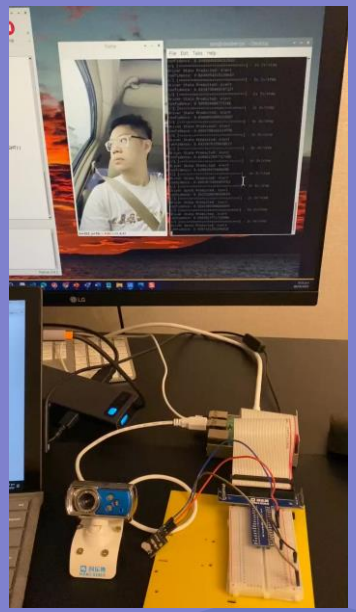
Deployment & Demo

Video File Demo

Alert/Focused

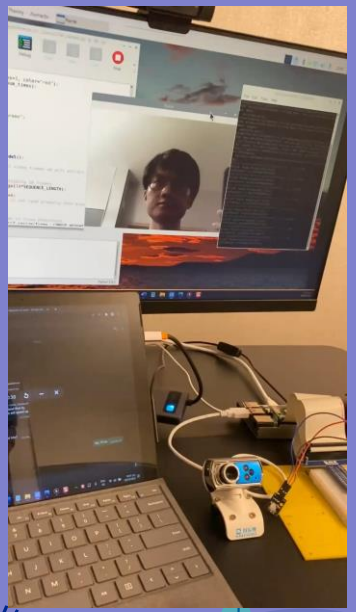


Drowsy/Distracted



Camera Demo

Mixed



Deployment & Demo

```
if is_driver_exists:
    fps_text = 'Current FPS = {0:.1f}. Expect: {1}'.format(fps, _MODEL_FPS)
    text_location = (_LEFT_MARGIN, _ROW_SIZE)
    cv2.putText(image, fps_text, text_location, cv2.FONT_HERSHEY_PLAIN,
                _FONT_SIZE, _TEXT_COLOR, _FONT_THICKNESS)

    time_per_infer_text = 'Time per inference: {0}ms'.format(
        int(time_per_infer * 1000))
    text_location = (_LEFT_MARGIN, _ROW_SIZE * 2)
    cv2.putText(image, time_per_infer_text, text_location,
                cv2.FONT_HERSHEY_PLAIN, _FONT_SIZE, _TEXT_COLOR,
                _FONT_THICKNESS)

    accuracy = 'Accuracy : %.2f%%' % (accuracy_value*100)
    result_text = classification + ' (' + str(accuracy) + ')'
    if classification_cnt > 30 and (accuracy_value*100)> 95 and pre_classification != 'normaldriving':
        pygame.mixer.music.load(constant.sound_focus_alert)
        pygame.mixer.music.play(1)
        classification_cnt = 0
    if classification == pre_classification:
       classification_cnt +=1
    else:
        classification_cnt = 0
    pre_classification = classification
    if constant.is_for_testing and (accuracy_value*100)> 95:
        cv2.imwrite(constant.model_result_model_alertness_storage_dir + constant.model_alertness_detection_split('.')[0] + '-' +
                    result_text + '-' + str(datetime.datetime.now()) + ".jpg", cv2.cvtColor(frame_rgb, cv2.COLOR_RGB2BGR))
```

Approach 2 – CNN Body posture	
Frame size	224 * 224
Classification	9 classes
Test Accuracy	99.79%
Confidence level expected	95% and above
Time to start alert	> 30 classification continuously not normal driving
Expected fps	5

Video File Demo

Turn on radio

Normal Driving

Conclusion

[What we have achieved]

- Implement a **Driver Alertness Detection System** to detect signs of drowsiness and distraction in driver's face and posture.
- System relying on **computer vision** capabilities and built using **deep learning** techniques.

Issuing real-time alerts

System monitors driver's video live-feed and issues alerts as risks are detected.

Deployable on edge devices

The system runs entirely locally on Raspberry Pi that represents an edge device. No external API calls needed.

Only monitors the driver

The system identifies the registered driver before detecting signs of drowsiness/distraction.

Conclusion

[What we have Learned]

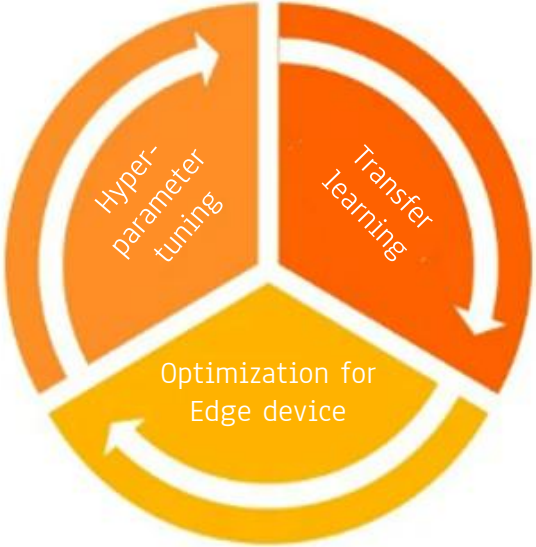
[Data Preparation]

i. Data pre-processing

ii. Data Augmentation

iii. Find optimal number of frames for data feed

[Model Optimization]



[Deployment]

i. Raspberry Pi setup and deployment

ii. Bundling a portable package that can be deployed on other devices

Research & Brainstorm → **Proof of Concept** → **Rollout & Deployment**

Future Improvements

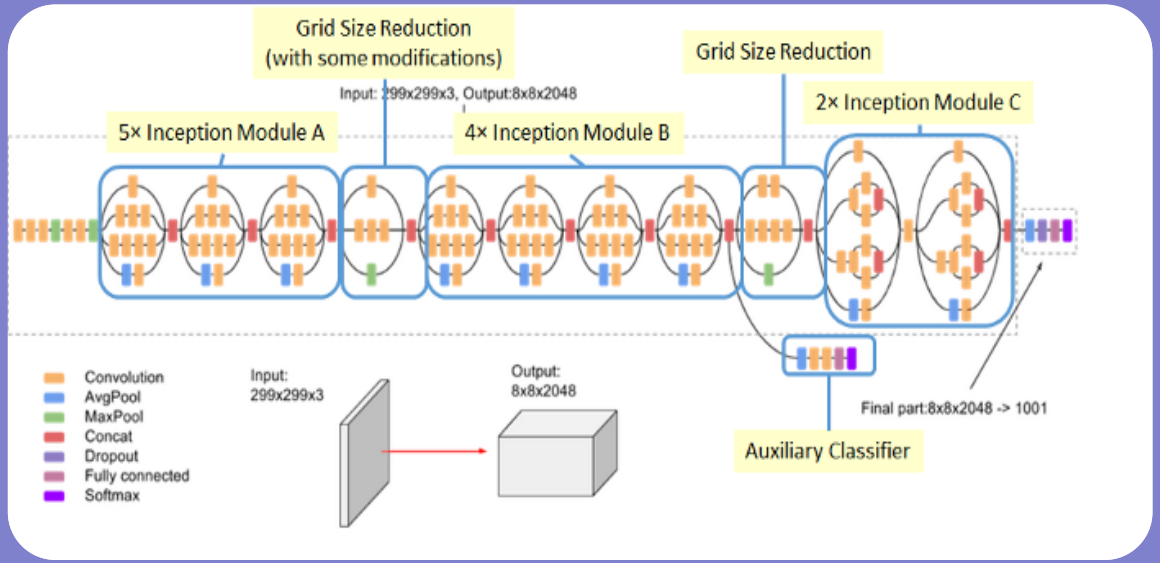
- **Collect more real-world images**
  - Gather images of drivers from different angles to account for the various possible positions of phone placement
- **Improve deployability of models**
  - Optimize and simplify CV models to get more efficient performance on edge devices (faster inference and utilizing lesser computational resources)
- **Integrate more sensor inputs**
  - Make the system more robust by integrating multiple sensors (for instance, GPS sensor could be integrated as an input to determine whether the car is moving or not before determining the alertness of the driver)
  - Explore different model architectures (multi-input, multi-output, shared layers, etc.) using Keras functional API


Thank
you!

Appendix

Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[Inception-v3 Model]



Inception-v3 Architecture

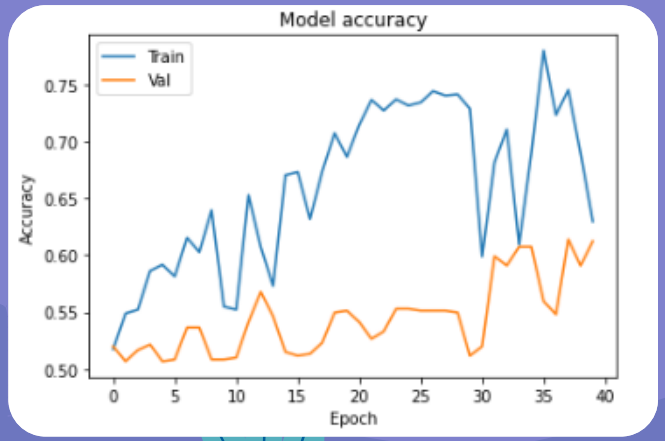
(TsangSik-Ho, Review: Inception-v3 – 1st Runner Up (Image Classification) in ILSVRC 2015, 2018)

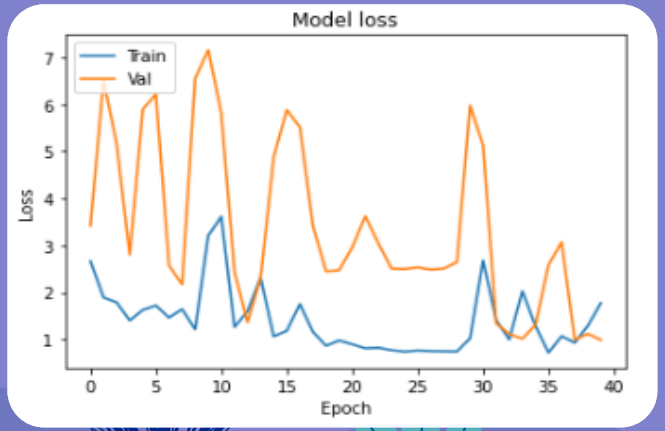
- Inception-v3 is created by considering below main concepts
  - Factorized Convolutions
  - Smaller Convolutions
  - Asymmetric convolutions
  - Auxiliary classifier
  - Grid size reduction

Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[Inception-v3 Model]

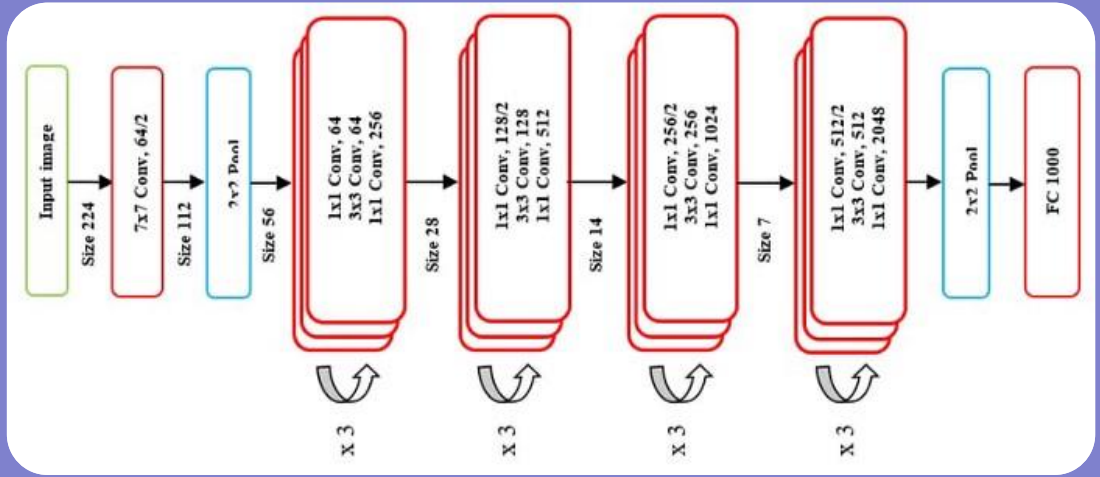
- Total number of parameters was above 21 million and only last few layers added was used as trainable parameters to utilize weights already learned in this model. Batch Normalization and ReLU are used after Convolutional layers.
- The model has managed to achieve low validation accuracy of 61% which was significantly lesser than CNN model created by the team (88% validation accuracy) or Xception model (92% validation accuracy).
- InceptionV3 model validation accuracy gradually increased but not so much improvement and huge spike of loss in between the epochs showing that this model is not so suitable for our use case.





Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[ResNet50 Model]



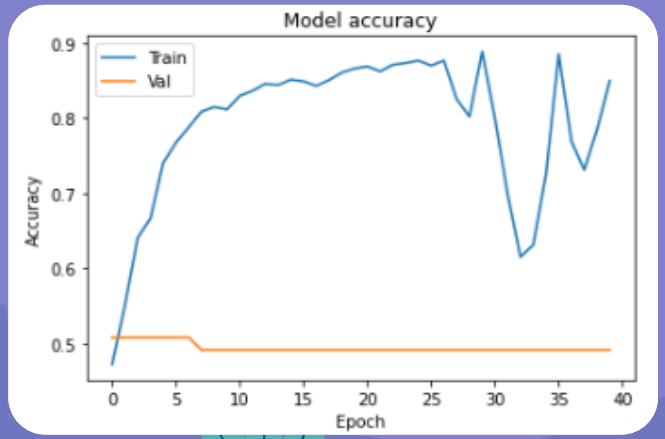
ResNet-50 Architecture
(MukherjeeSuvaditya, 2022)

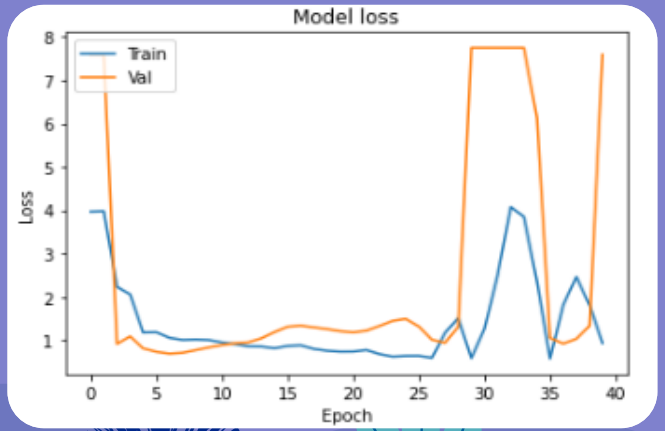
- ResNet solves "Vanishing Gradient" problem with "Skip Connections".
- ResNet piles up multiple convolution layers, and skips those layers, and use the activations of the previous layer again.
- Skip connections increase initial training speed by compressing deep neural network into few layers.

Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[ResNet50 Model]

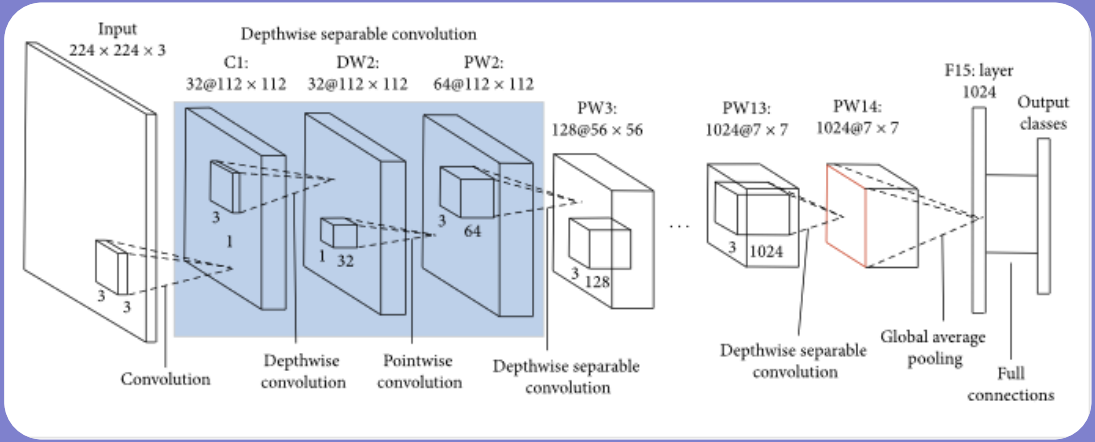
- Total number of parameters was above 23 million and only last few layers added was used as trainable parameters to utilize weights already learned in this model.
- ResNet-50 achieved very low validation accuracy of 49% which was worst of all transfer learning models tried.
- Although training accuracy has improved over time to achieve up to 88%, validation accuracy never improved and remained around 49% which is clear indicator that this model was not suitable for our use case.
- Also, model loss has increased drastically between 31st to 35th epoch which makes the model not so reliable for usage.





Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[MobileNet Model]



ResNet-50 Architecture
(MukherjeeSuvaditya, 2022)

- MobileNet is created to be deployed into mobile applications. MobileNet uses depth-wise segregation of convolutions.
- This model drastically decreased number of parameters to train, resulting in lightweight deep neural network.

Approach 1 - CNN Model Based on Facial Landmarks (Cont.)

[MobileNet Model]

- Total number of parameters was above 3 million and only last few layers added was used as trainable parameters to utilize weights already learned in this model.
- MobileNet achieved moderate validation accuracy of around 80% which was slightly lower than CNN model created by the team (88% validation accuracy).
- One of the observations from the model training is that training and validation accuracy goes through lots of ups and downs throughout the epoch. This model seemed little inconsistent to be utilized on our use case.