Student Drowsiness Detection

Vivek Gohel

Preprocessing dataset

- First We have extracted images from the videos using OpenCV library.
- Images were mapped with the labels which were given in the dataset.
- We have also mapped the images in google drive dataset to alert and drowsy as per their labels.
- We have recognised the face and its features like eyes and mouth using dlib and faceutils libraries to mask off other pixels to take only eyes and mouth.
- We did this because on reviewing other papers we found that neural networks like CNN models gets overtrained on this dataset, as it gives more importance to background and content.
- In total we have extracted images from the ultraLDD video dataset apart from the images obtained by the google drive dataset which both combined to 57488 images.

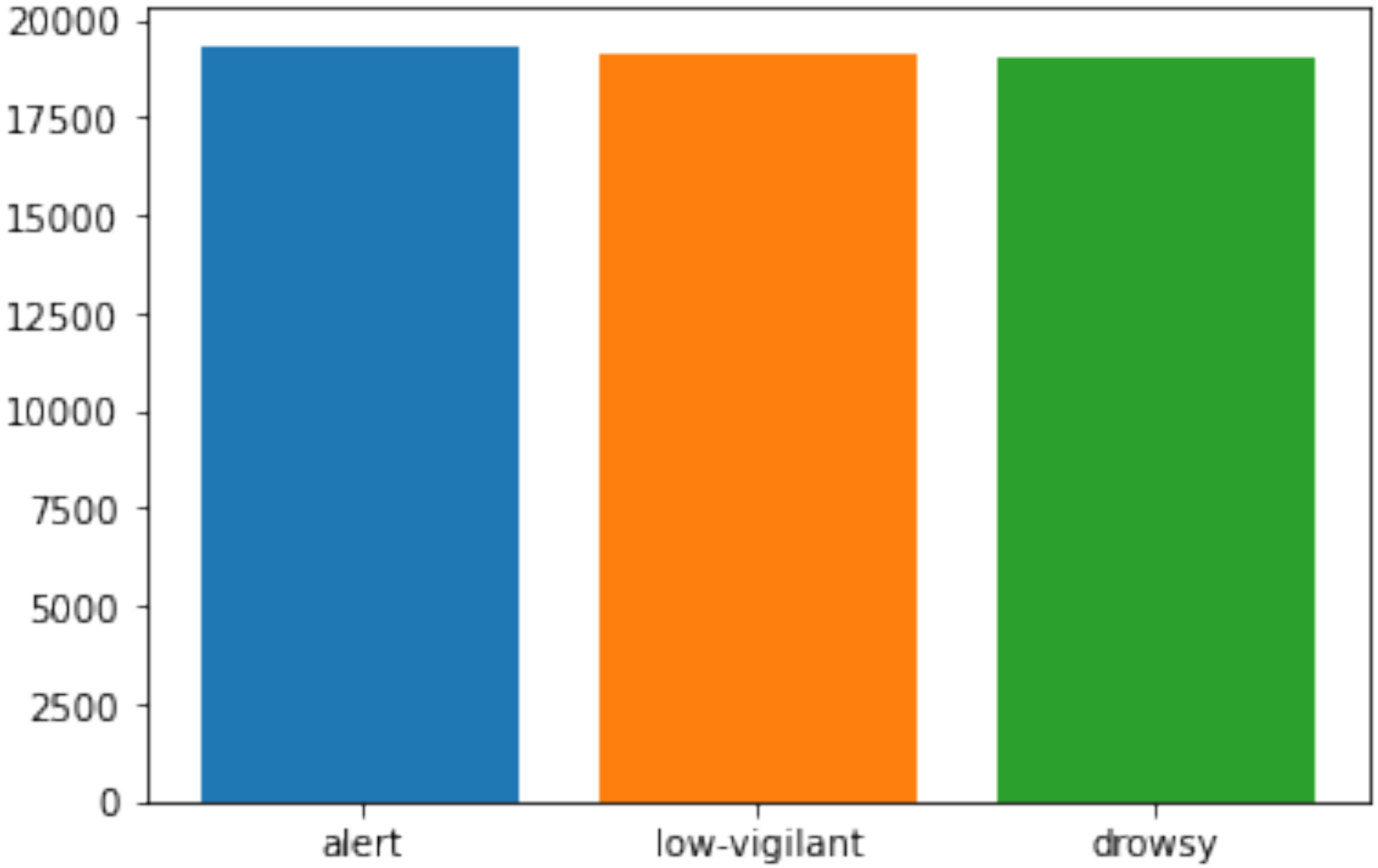
Processed images

Alert

Low Vigilant

Drowsy

Data distribution of frames



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 510, 510, 32)      896
_________________________________________________________________
activation (Activation)      (None, 510, 510, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 255, 255, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 253, 253, 32)      9248
_________________________________________________________________
activation_1 (Activation)    (None, 253, 253, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 124, 124, 64)      18496
_________________________________________________________________
activation_2 (Activation)    (None, 124, 124, 64)      0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 62, 62, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 246016)            0
_________________________________________________________________
dense (Dense)                (None, 64)                15745088
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
_________________________________________________________________
activation_4 (Activation)    (None, 3)                 0
=================================================================
Total params: 15,773,923
Trainable params: 15,773,923
Non-trainable params: 0
_________________________________________________________________
```

Model

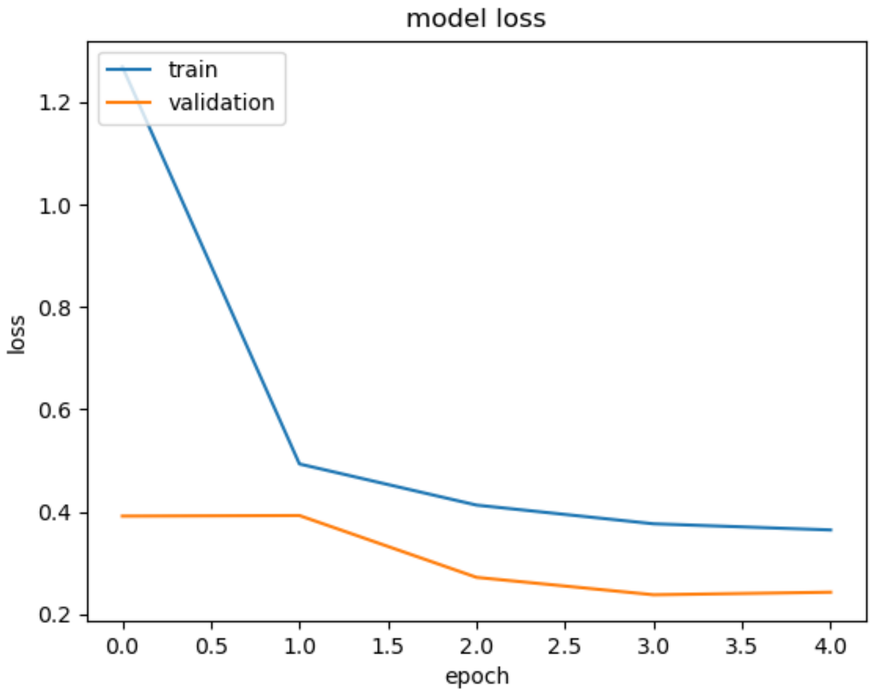
- We have tried to mimic the structure of the VGG network, so that we can get good accuracy using less resources and time.

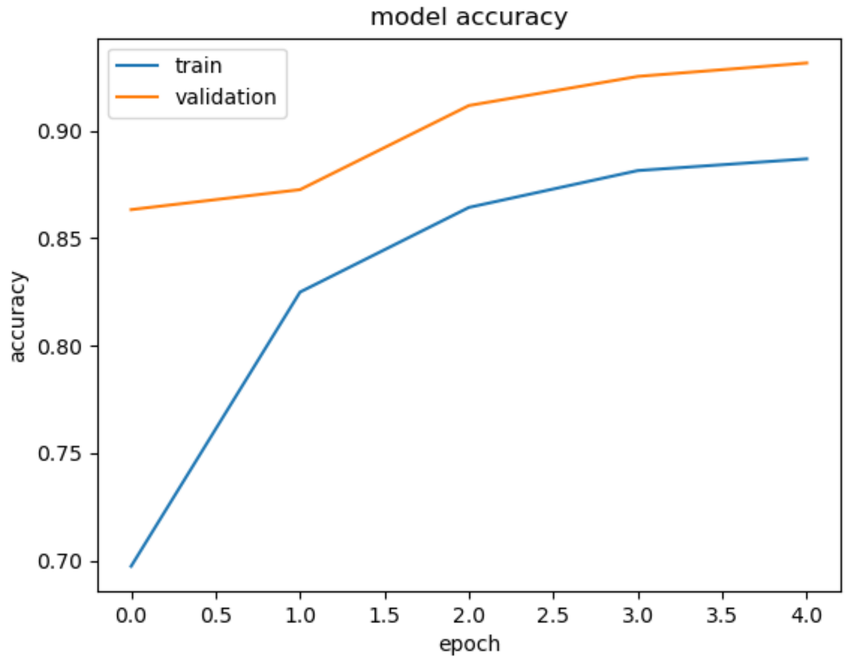
- We have defined our own CNN model with 3 convolution layers with Max pooling and 2 dense layers, here is the structure of the model.

Training Pipeline

- Due to the large size of the dataset we were not able to load all the data into RAM at once.
- Therefore we tried keras' ImageDataGenerator, which loads the images as and when required while training the model, but this approach was very slow.
- The alternative was to make a custom data dataset class using tensorflow.data library to load images into buffer as per the requirement.
- After making the dataset generator, we randomly split it into training and validation datasets(in 80:20 ratio).
- Our model was giving acceptable accuracy in 5 epochs only, as there was adequate ammount of the samples available with us.

Loss and Accuracy



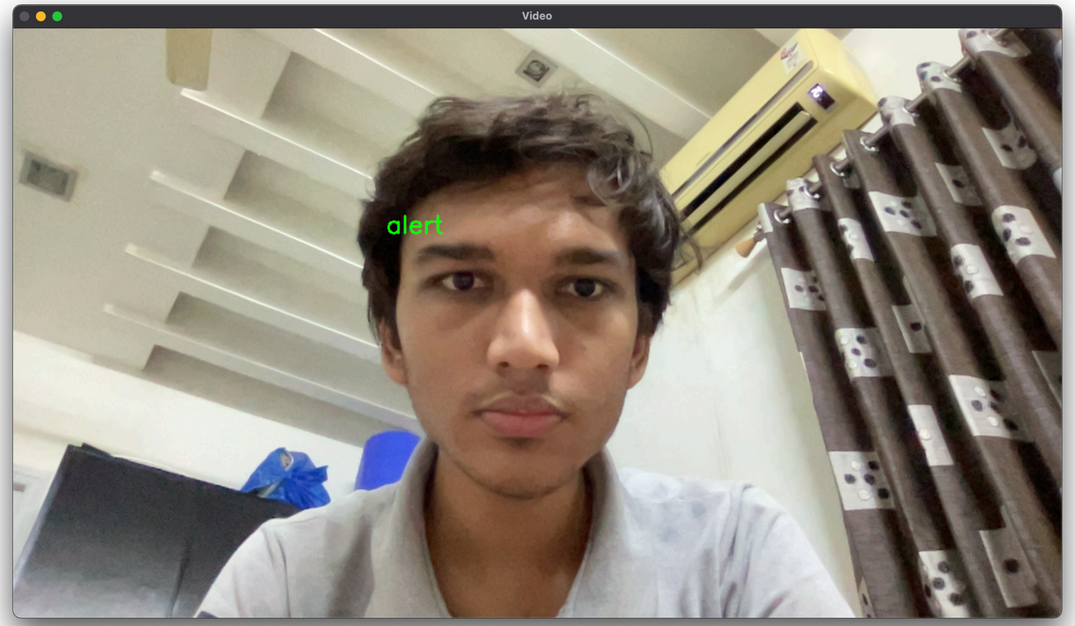


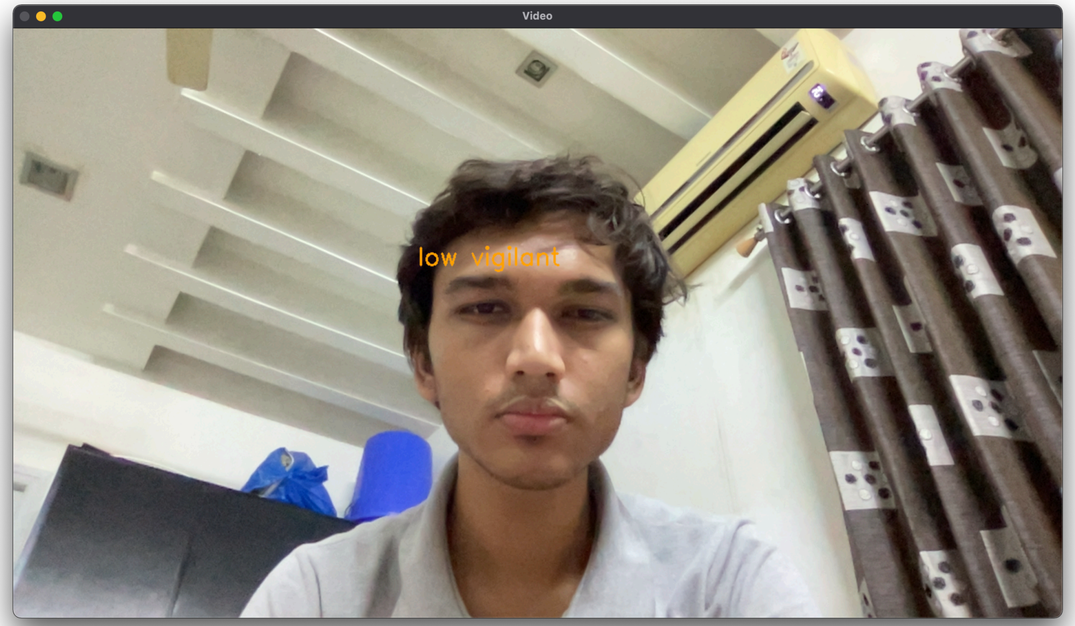
	Loss	Accuracy
Training Dataset	0.2562	92.59%
Validation Dataset	0.2447	93.01%

Prediction Pipeline

- For prediction part, we have 2 options, one is to predict from live camera feed and other is from the saved videos.
- The prediction comes as an overlay in the display window.

Predictions example





Thank you