Drowsiness Detection from Labelled Recorded Videos Using Computer Vision

Problem Statement: Develop a model for detecting if the driver is alert while driving or not and give the signal.

Understanding: While driving many times it happens that due to various reasons the drivers got distracted or due to fatigue, they lose their alertness. The goal is to take the video from camera in real time and predict/classify from the frames (*considering as a time series of frames or individual frame one at a time*) that the driver is distracted or not alert while driving using new technologies like deep learning, Convolutional Neural Network etc.

Description of Data: The DMD dataset comprises extensive recorded videos featuring diverse drivers in various scenarios, encompassing both authentic car interiors and simulated environments. During a specific timeframe, video footage was simultaneously captured from three distinct perspectives: 1. A frontal view of the driver's face, 2. A rear view capturing the driver's hands, and 3. A comprehensive view of the entire body. Each camera angle produced three corresponding types of videos: 1. Frames containing RGB channels, 2. Infrared (IR) frames, and 3. Depth data.

For training purposes, a representative video showcasing a complete body with an RGB frame is employed. The annotation process involves JSON files detailing frame ranges and corresponding labels for different driving scenarios. These labels correspond to the actions undertaken by the driver during specific frame intervals.

Figure 1: Sample Photo Frames

Preprocessing Labels and EDA: The videos have been transformed into individual frames. The frame ranges specified in the JSON files have been restructured into a tabular layout, with columns representing frame IDs and various actions. This information is represented in binary form. An illustrative table is provided below, wherein the classes, labeled as c0, c1, and so forth, correspond to the distinct classes derived from the JSON files. Notably, the labels within these different classes are generally well-distributed. The frequency diagram is visually depicted underneath, offering insights into the distribution of these labels.

Class	Actual Case
c0	drinking
c1	hair and makeup
c2	operating the radio
c3	reaching behind
c4	safe driving
c5	talking on the phone - left
c6	talking on the phone - right
c7	talking to passenger
c8	texting - left
c9	texting - right

Table 1: Class Mapping

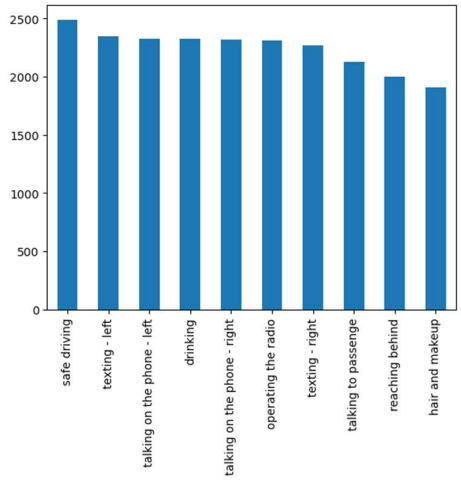

Figure 2: Class Distribution in Training Data

frameid	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
987	1	0	0	0	0	0	0	0	0	0
988	1	0	0	0	0	0	0	0	0	0
989	1	0	0	0	0	0	0	0	0	0
990	1	0	0	0	0	0	0	0	0	0
1256	0	0	0	1	0	0	0	0	0	0
1257	0	0	0	1	0	0	0	0	0	0
1258	0	0	0	1	0	0	0	0	0	0
1259	0	0	0	1	0	0	0	0	0	0
1260	0	0	0	1	0	0	0	0	0	0
1261	0	0	0	1	0	0	0	0	0	0
1262	0	0	0	1	0	0	0	0	0	0
1821	0	0	0	0	0	0	1	0	0	0
1822	0	0	0	0	0	0	1	0	0	0

Table 2: Sample Tabular Labels

Solution Approach: This use case of deep learning can be solved in four different approaches.

I. 3-Dimensional Convolutional Neural Network over the time dimension.
II. Traditional 2-Dimensional Convolutional Neural Network without time dimension.
III. Recurrent Neural Network (LSTM, GRU or any RNN) for time series analysis.
IV. Depth and Point wise Convolutional Neural Network.

To solve this problem, a Depth and Point-wise Convolutional Neural Network (CNN) approach was used. This method achieves similar results while using way fewer computations, which saves a lot of time and computational power. For instance, with the traditional 2D Convolution operation on a 25x25x3 image using a 3x3 filter, and considering 16 such filters, you'd need 270,000 multiplications. However, the Depth-wise convolution first performs operations for each channel, requiring only 16,875 operations. Then, the Point-wise operation with 16 sets of 1x1x3 filters needs 30,000 operations. In total, this adds up to 46,875 operations, much fewer than the standard 2D Convolution.

The model takes one frame at a time as input and does a classification task to assign the frame to a class mentioned in table 2. The model outputs class weights, using a softmax activation at the output layer to assume that each frame belongs to only one class. The model consists of a single input layer designed to take one frame as input.

Model Architecture: The model is divided into two distinct sections. The initial part employs MobileNetV2, a pre-trained model for image classification. The MobileNetV2 model is set to non-trainable, benefiting from its extensive training on vast datasets for classification tasks. However, there's the possibility of fine-tuning it on this specific dataset for enhanced robustness. Following this, a MaxPooling2D layer is integrated to extract pivotal features. Subsequently, the output is transformed into flat tensors, followed by a series of dense layers incorporating ReLU activation functions. The conclusive output layer consists of 10 nodes utilizing softmax activation to classify the images. Refer to Figure 4 for a visual representation of the model's architecture.

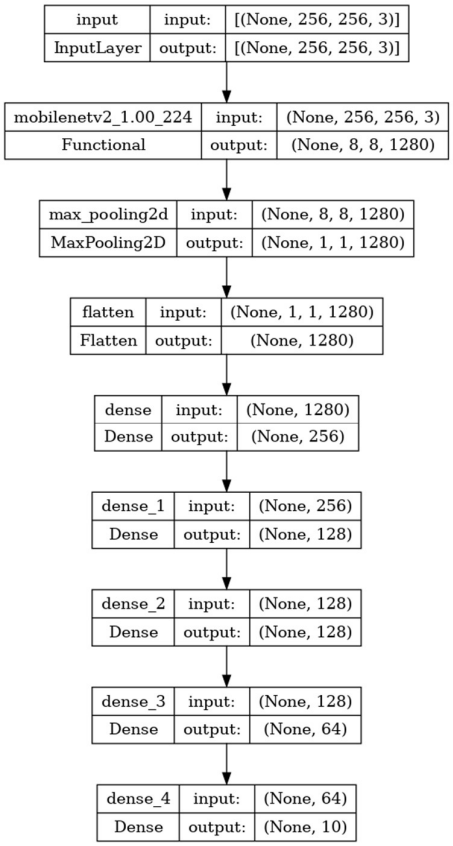

Figure 3: Model Architecture

Image Preprocessing: During the training process, a subset of videos from the body camera angle is exclusively utilized. Specifically, the labels inferred from this body camera angle are employed. The training images undergo preprocessing steps before being input into the model. The sequence of actions is outlined as follows:

1. The images are transformed into a 3-channel image volume, incorporating red, green, and blue channels.

2. These images undergo resizing, adjusting them to a size of (256, 256, 3), given that the base model operates optimally with square images.

3. The pixel intensities are initially ranged between 0 and 255. To expedite the convergence of the model, these intensities are normalized to fall within the range of 0 to 1.

Results: Different training parameters are stated below.

- Optimizer: Adam optimizer
- Learning rate: 1e-5
- Loss: categorical cross entropy
- Metrics: Accuracy, Recall, Precision
- Epochs: 36
- Steps per epoch: 561
- Batch size: 32
- Training data size: 17939 images
- Validation data size: 4485 images

	Training	**Validation**
Categorical cross entropy	0.0420	0.1089
Accuracy	99.46%	96.79%
Precision	0.9971	0.9743
Recall	0.9916	0.9645

Table 3: Training and Validation Results

After 36 epochs of training the final training and validation loss and metric values are shown in table and the plots are shown below.

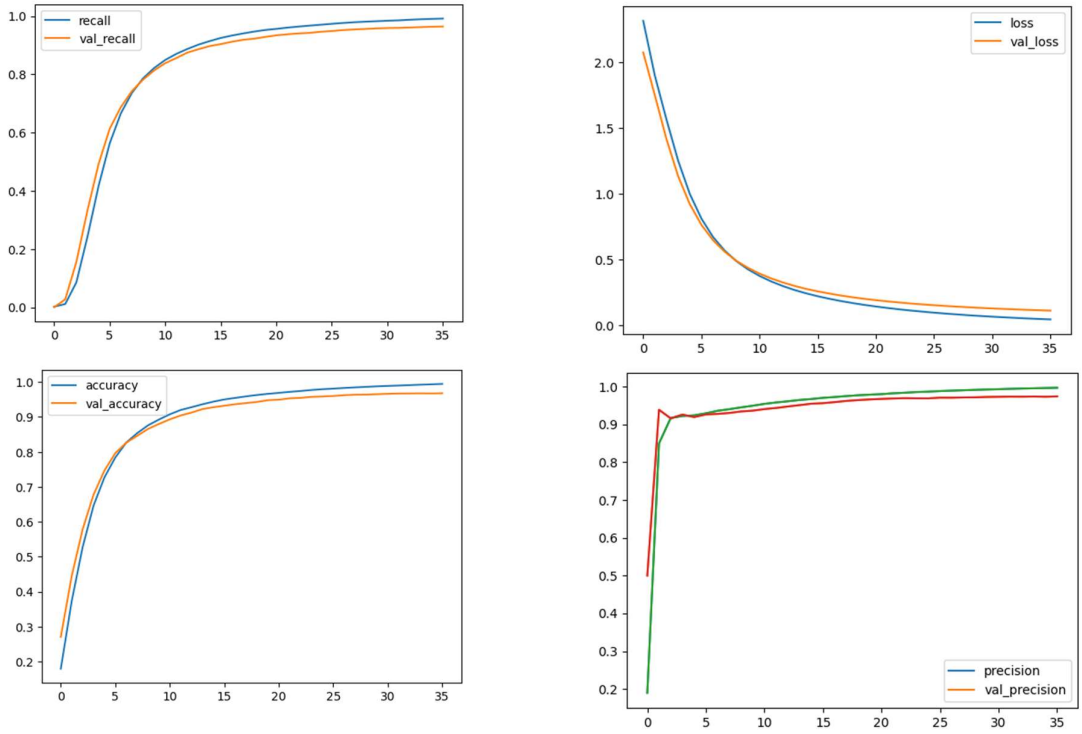

Figure 4: Loss and Metrics Plot Over Epochs

Test Results: The model is then tested with unseen data. The images are preprocessed as of the training images and then it is used for prediction. The predicted values are post processed and mapped back to the actual textual classes from the mapping object. Some sample test images with predicted labels are shown below.

Figure 5: Test Predictions On Unseen Data

Realtime Use: The foundational component of this model heavily relies on the MobileNetV2 architecture, originally designed for mobile devices, enabling its effective deployment even with limited computational resources. Employing this trained model involves a sequential process: real-time frames need to undergo initial preprocessing before being fed into the model. The resulting predicted values are subsequently mapped back to their corresponding actual actions. Alternatively, based on these predicted values, an alerting mechanism could be implemented. This might involve triggering sound alerts or notifying designated emergency contacts, contributing to an enhanced safety aspect.

Future Scope: Certain assumptions underlie the functionality of this model. One such assumption is that a driver engages in a singular action at any given time—such as drinking or hair-makeup—but not a combination of two categories within a single frame. Additionally, this assumption holds true for individual frames, as the model exclusively performs classification on whole-body videos. However, there exist several avenues for potential improvement:

- Multi-Action Frames: The current model design can be enhanced to accommodate scenarios where a single frame might encompass multiple actions. This would require modifying the model's architecture to handle such complexities.
- Temporal Convolution: Introducing convolutional operations along the time dimension can facilitate processing sequences of frames, allowing the model to classify entire videos instead of individual frames.
- Data Augmentation: Employing data augmentation techniques can contribute to the creation of a more robust model, capable of handling diverse real-world scenarios.
- Facial Analysis: A separate training approach could be applied to analyze facial expressions and actions, enabling predictions related specifically to face-related classes.

By addressing these aspects, the model's capabilities could be expanded and refined to provide more accurate and adaptable results.