Mid-term Demo

Real-time Drowsiness Detection using transfer learning

Presented By:

Venkatesh Dommata
b00116626
Oklahoma City University

A Stepping Stone to Safer Roads

Why transfer learning ?

Transfer learning applies knowledge from models trained on large datasets to new problems, reducing computational costs and improving performance even with limited training samples.

Mission

As a part of my Masters project, I developed a real-time drowsiness detection system using the MRL Dataset and the MobileNet architecture (transfer learning). The system is designed to detect drowsiness in drivers in real-time, providing immediate alerts when signs of drowsiness are detected. This project focused on implementing an efficient and accurate detection system that can work in varying lighting conditions and with different drivers.

Methodology

Data Collection, Preprocessing and Feature extraction

Collected MRL (Media Research lab) Eye dataset with 84.9k images from 37 individuals (33 men, 4 women), split into training (40.4k closed, 41.3k open eyes) and testing sets (1566 closed, 1657 open eyes).

Model Training

Trained a MobileNet-based transfer learning model on the MRL Eye dataset, fine-tuning the pre-trained network with our drowsiness detection data to achieve accurate classification between open and closed eyes for real-time detection.

Performance Evaluation

Tested the MobileNet model for real-time drowsiness detection using both test data and random eye images from external sources. The model showed accurate results, even on images not included in the original dataset.

Real time detection

After achieving satisfactory accuracy, we'll implement real-time detection using the webcam. With Haar Cascade eye classifier in OpenCV, we'll detect eyes and use the trained model to predict if they're open or closed, triggering an alarm if closed for too long.

Data Collection and Preprocessing

Train Data along with labels (open/closed)

```
[8]: df_train
```

[8]:

	image_path	label
0	../input/dataset/data/train/close eyes/s0007_0...	close eyes
1	../input/dataset/data/train/close eyes/s0037_0...	close eyes
2	../input/dataset/data/train/close eyes/s0031_0...	close eyes
3	../input/dataset/data/train/close eyes/s0037_0...	close eyes
4	../input/dataset/data/train/close eyes/s0018_0...	close eyes
...	...	...
81670	../input/dataset/data/train/open eyes/s0028_00...	open eyes
81671	../input/dataset/data/train/open eyes/s0019_04...	open eyes
81672	../input/dataset/data/train/open eyes/s0032_01...	open eyes
81673	../input/dataset/data/train/open eyes/s0036_04...	open eyes
81674	../input/dataset/data/train/open eyes/s0014_07...	open eyes

81675 rows × 2 columns

Test Data along with labels (open/closed)

```
df_test
```

[9]:

	image_path	label
0	../input/dataset/data/test/close eyes/s0002_00...	close eyes
1	../input/dataset/data/test/close eyes/s0002_00...	close eyes
2	../input/dataset/data/test/close eyes/s0003_00...	close eyes
3	../input/dataset/data/test/close eyes/s0002_00...	close eyes
4	../input/dataset/data/test/close eyes/s0002_00...	close eyes
...	...	...
3218	../input/dataset/data/test/open eyes/s0001_024...	open eyes
3219	../input/dataset/data/test/open eyes/s0001_023...	open eyes
3220	../input/dataset/data/test/open eyes/s0001_031...	open eyes
3221	../input/dataset/data/test/open eyes/s0001_024...	open eyes
3222	../input/dataset/data/test/open eyes/s0001_029...	open eyes

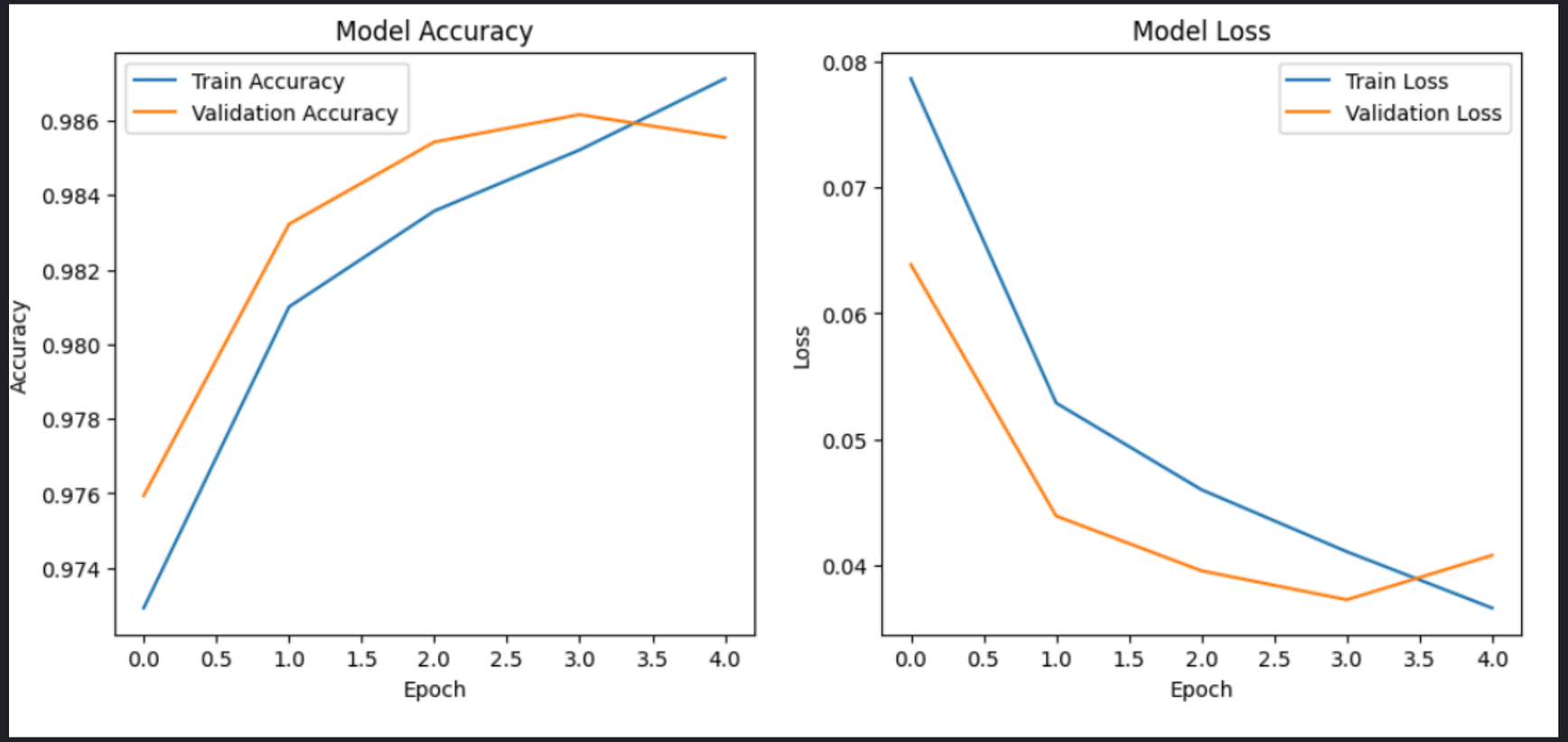
3223 rows × 2 columns

Data Collection and Preprocessing

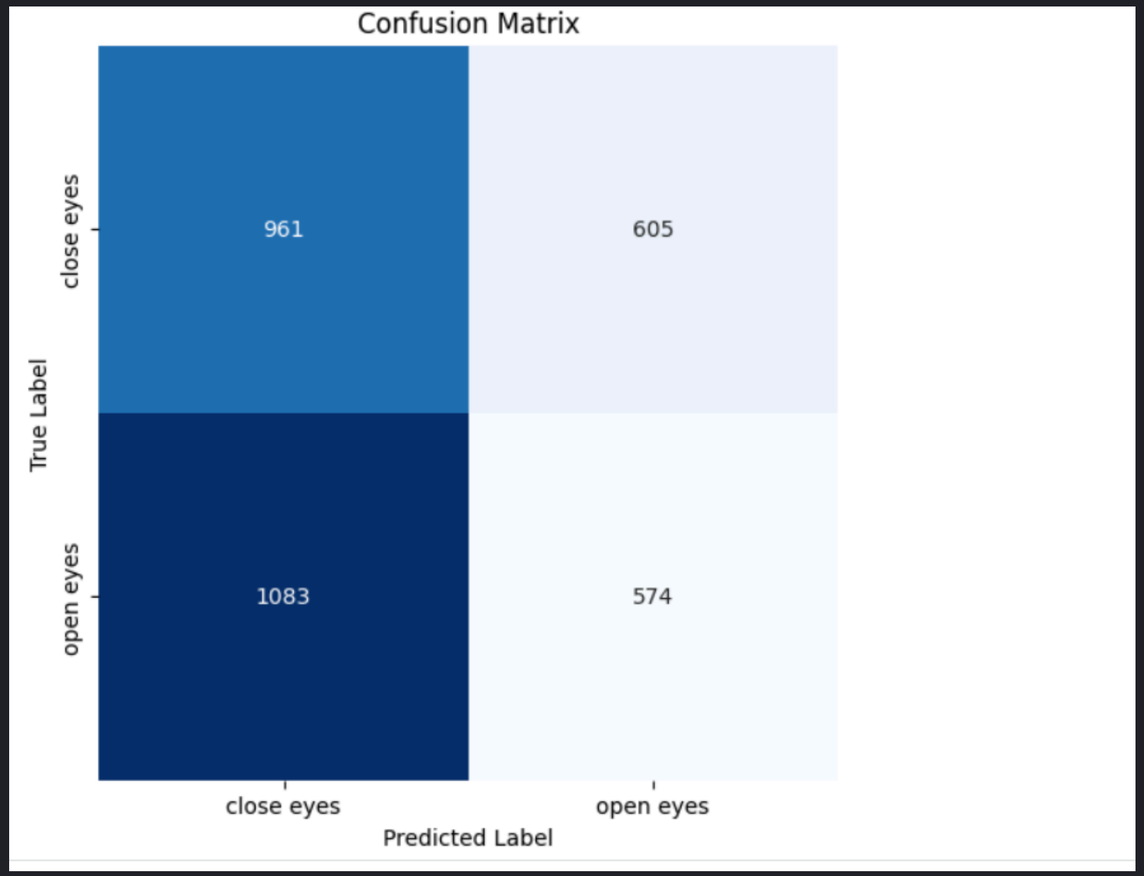
1. **Data Splitting**:
   - The dataset is split into training and validation sets using train_test_split(), where 80% of the data is used for training and 20% for validation. The splitting is done while ensuring the distribution of the labels is preserved using the stratify parameter.
2. **Data Augmentation**:
   - The ImageDataGenerator class is used for data augmentation, which helps improve model generalization. This includes several augmentation techniques:
     - Rotation: Random rotation of images up to 10 degrees.
     - Width and Height Shift: Random shifts in width and height by 10%.
     - Horizontal Flip: Random horizontal flipping of images.
     - Brightness Adjustment: Random changes in brightness within the range [0.8, 1.2].
     - Rescaling: Normalizes pixel values by dividing by 255 (rescaling the pixel values to the range [0, 1]).
     - Fill Mode: Defines how new pixels are filled after transformations.
3. **Initial Image Pixels**:
   - The original images in the dataset may have varying resolutions (e.g., 640x480, 1280x720, etc.). Before feeding them into the model, all images are resized to a consistent size of 224x224 pixels.
4. **Image Preprocessing and Data Loading**:
   - The train_generator, valid_generator, and test_generator are created using flow_from_dataframe(), which loads images from paths stored in the dataframe and applies the corresponding augmentations. The images are resized from their initial resolution (e.g., 640x480) to 224x224 pixels, ensuring consistency in input size for the model.
   - The class_mode is set to 'binary' because the classification problem involves two classes, and the model expects binary labels.

This preprocessing ensures that the images, regardless of their initial resolution, are standardized to 224x224 pixels and undergo transformations that improve the model's robustness and performance.

Model Training and Evaluation



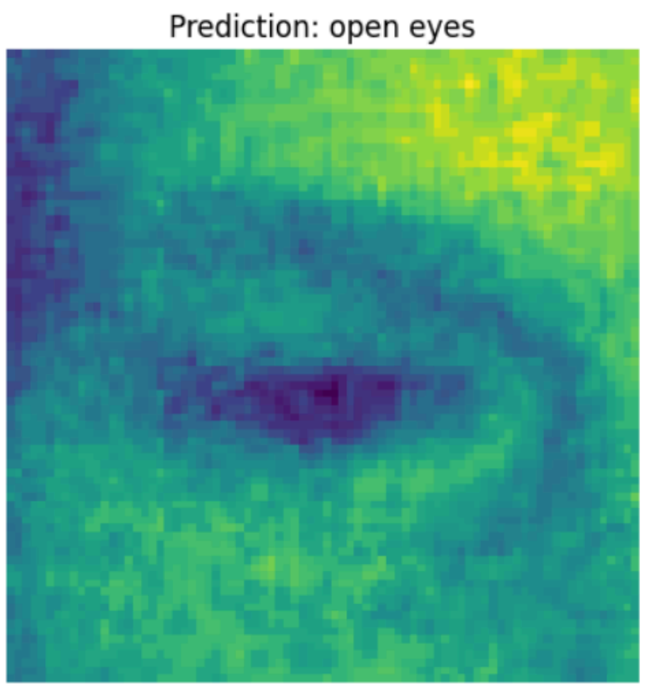
Performance Evaluation



,

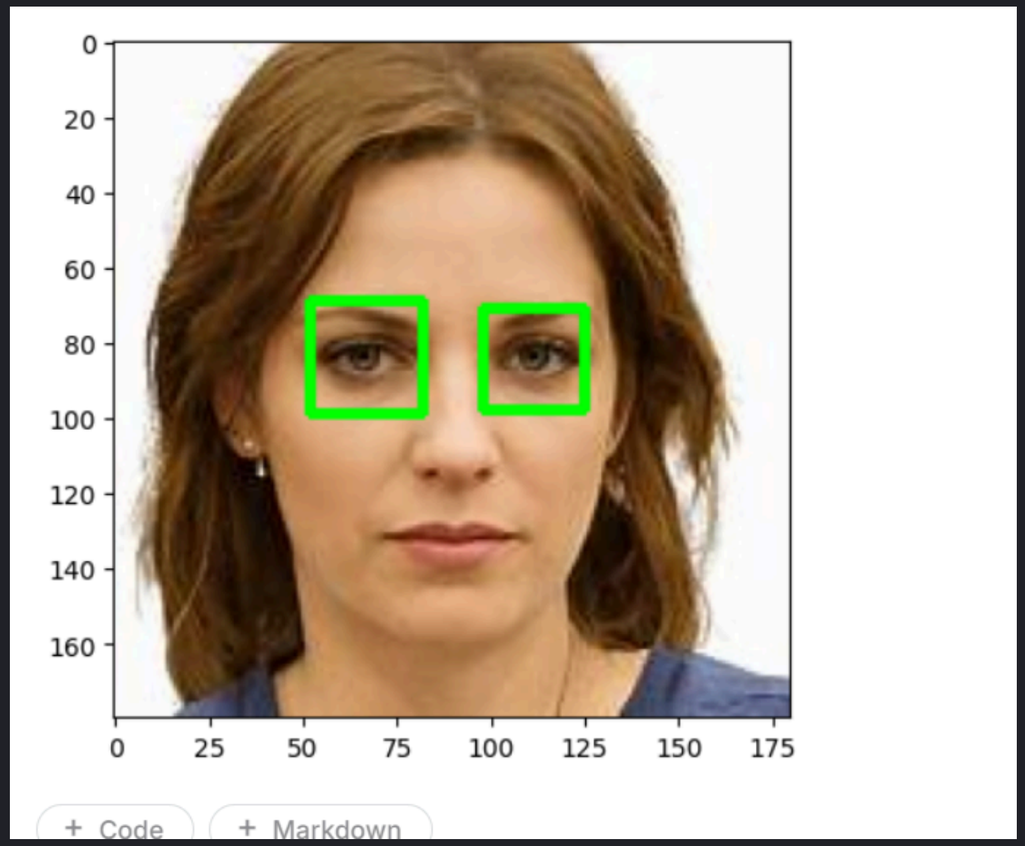
Performance Evaluation

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 639ms/step
Predicted Probabilities: [[0.85977834]]
Predicted Class: open eyes
```



,

Real time detection



After Mid - Evaluation

Real time detection and alarm

Thank you

Oklahoma City University | Venkatesh Dommata | Computer Science - Data Science