A logo with text on it

Description automatically generated

**Artificial Intelligence**

**Project Report**

**Driver’s Drowsiness Detection**

**Submitted By:**

Ahmed Bilal – 210201096

Ibraheem Lughmani – 210201044

**Submitted To:**

Ma’am Tayaba Arshad

1. **Objectives**

The primary objective of this project is to alert the driver if he is drowsy while driving using advanced artificial intelligence techniques. The goals include:

* Develop an algorithm to detect when the user's eyes are closed with high accuracy.
* Define and implement a threshold for eye closure duration to determine when the user is drowsy.
* Ensure the system activates an alarm when the user is detected to be drowsy and keeps it active until the user opens their eyes.
* Achieve real-time processing capabilities to monitor and respond to drowsiness instantly.

1. **Methodology**

**2.1 Data Collection and Preprocessing**

The dataset for this project contains images of closed and opened eyes. The preprocessing steps include:

* Data Cleaning: Remove any blurry, irrelevant images and frames to ensure the dataset contains only high-quality, useful data.
* Normalization: Standardize pixel values to a consistent range (e.g., 0-1) to improve model performance and convergence during training.
* Resizing: Detected eye regions are resized to 80x80 pixels.
  1. **Algorithm Implementation**

We implemented several machine learning models for this project:

* **Haar Cascade Classifier:**
  + **Face and Eye Detection:**
    - Uses pre-trained Haar cascade classifiers from OpenCV to detect faces and eyes in video frames.
    - Applies the classifiers to grayscale images to identify and locate faces and eyes.
* **Convolutional Neural Network (CNN) Prediction:**
  + **Model Loading and Prediction:**
    - A CNN model (‘model.keras’) is loaded to classify eye states.
    - The preprocessed eye image is fed into the model, which outputs a probability for the eye being open or closed.

**2.3 Model Training**

1. **Convolutional Neural Network (CNN):**
   * CNN architecture implemented using TensorFlow and Keras, processes image data effectively.
2. **Image Data Generator:**

* Keras' Image Data Generator augments training and validation datasets, enhancing model robustness.

1. **Model Architecture:**

* The CNN consists of convolutional layers followed by max-pooling layers, ending with a fully connected layer and softmax activation.

1. **Loss Function and Optimizer:**

* Categorical cross-entropy loss function and Adam optimizer are used for model compilation.

1. **Callbacks:**

* Callbacks like ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau optimize training and monitor model performance.

1. **Training Loop:**

* The fit() method trains the model on the dataset with specified parameters.

**3. Implementation**

* 1. **Model Development**
* **Data Preparation:**

1. Organize eye images into "Close" and "Open" directories.
2. Apply data augmentation for diversity.

* **Model Architecture:**

1. Design a CNN using TensorFlow and Keras.
2. Include convolutional and max-pooling layers.
3. Add dropout layers for regularization.
4. Utilize softmax activation for classification.

* **Compilation:**

1. Compile the model with Adam optimizer and categorical cross-entropy loss.

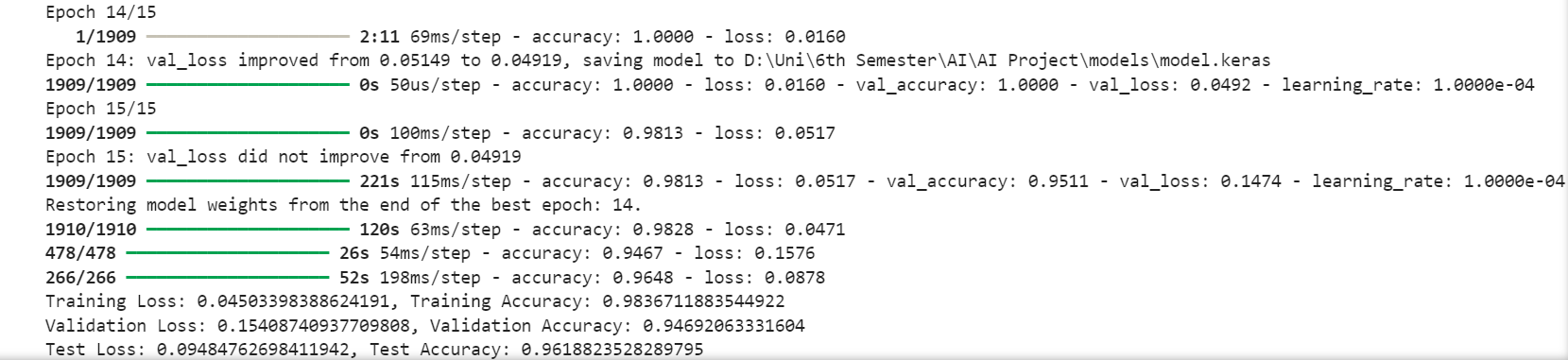
* **Training:**

1. Train the model on augmented data.
2. Specify batch size, epochs, and callbacks for optimization.

**3.2 Model Training**

1. **Data Preparation**:
   * Organize eye images into "Close" and "Open" directories.
   * Apply data augmentation techniques (e.g., rotation, shearing) to increase dataset diversity.
2. **Model Architecture Design**:
   * Design a Convolutional Neural Network (CNN) using TensorFlow and Keras.
   * Configure convolutional layers for feature extraction and max-pooling layers for dimensionality reduction.
   * Incorporate dropout layers to prevent overfitting.
   * Use softmax activation for classifying eye states.
3. **Compilation**:
   * Compile the model using the Adam optimizer for gradient descent and categorical cross-entropy loss function for multi-class classification.
4. **Training**:
   * Train the compiled model on the augmented dataset.
   * Specify training parameters such as batch size, number of epochs, and callbacks (e.g., ModelCheckpoint, EarlyStopping) for optimization.
   * Execute the training loop to update model parameters iteratively.
5. **Evaluation**:
   * Evaluate the trained model on separate training, validation, and test datasets.
   * Assess performance using metrics like loss and accuracy to gauge model effectiveness.
6. **Deployment**:
   * Once trained and evaluated, deploy the model for real-time eye state detection in the drowsiness detection system.

**3.3 Evaluation and Results**



**3.4 Deployment**

When the project runs, users see themselves on the screen. The project draws boxes around their face and eyes to show where it's looking. If it notices they're getting drowsy (like closing their eyes or looking down), it makes a sound or shows a warning on the screen to let them know. This helps users stay awake and pay attention.

**4. Conclusion**

In conclusion, this project demonstrates the effective use of computer vision and machine learning techniques for real-time drowsiness detection. By leveraging a Convolutional Neural Network (CNN) model trained to classify eye states, the system can accurately identify signs of drowsiness, such as closed eyes or a downward-facing face, in live video streams from a webcam. The integration of Haar cascade classifiers for face and eye detection ensures efficient preprocessing of input data, while the implementation of data augmentation techniques during model training enhances the robustness and generalization capability of the CNN. Through the deployment of an intuitive user interface, users receive timely alerts when drowsiness is detected, enabling them to take necessary actions to maintain alertness and prevent potential accidents. Moving forward, further refinements in model architecture, optimization of training parameters, and continuous monitoring of system performance will contribute to the ongoing improvement and effectiveness of the drowsiness detection system in real-world applications, particularly in safety-critical contexts such as driver monitoring systems.