

INTERNSHIP REPORT

Task 3: Drowsiness Detection System

Prepared by: **Tarrush Saxena**

Date: December 2025

Table of Contents

1. Introduction
2. Background
3. Learning Objectives
4. Activities and Tasks
5. Skills and Competencies
6. Challenges and Solutions
7. Outcomes and Impact
8. Conclusion

1. Introduction

This report documents my work on Task 3 of my internship program: developing a **Drowsiness Detection System**. The project aims to create a real-time application that can detect if a person (typically a driver) is drowsy or falling asleep, and also predict their approximate age. This is an important safety application that could help prevent accidents caused by driver fatigue.

The system uses computer vision techniques combined with deep learning models to analyze video feeds from cameras or uploaded images and videos. It processes each frame to detect faces, analyze eye states, and determine if the person appears to be sleeping. Additionally, it estimates the age of detected individuals to provide demographic information.

2. Background

Drowsy driving is a major cause of road accidents worldwide. According to studies, driver fatigue contributes to approximately 20% of all traffic accidents. Developing automated systems that can detect drowsiness in real-time can significantly improve road safety by alerting drivers before accidents occur.

Previous approaches to drowsiness detection have included physiological sensors (like EEG), vehicle-based metrics (lane deviation, steering patterns), and computer vision-based methods. This project focuses on the computer vision approach, which is non-intrusive and can be implemented using readily available cameras.

Technologies and Frameworks Used:

- Python 3.9+ as the primary programming language
- TensorFlow/Keras for deep learning model development
- OpenCV for computer vision and image processing
- Tkinter for the graphical user interface
- NumPy for numerical computations
- Matplotlib and Seaborn for data visualization

3. Learning Objectives

The primary learning objectives for this project were designed to build practical skills in machine learning, computer vision, and software development:

- **Deep Learning Fundamentals:** Understanding Convolutional Neural Networks (CNNs) and their application to image classification and regression tasks.
- **Computer Vision Techniques:** Learning face detection methods including Haar Cascades and DNN-based detectors, as well as eye detection algorithms.
- **Model Training Pipeline:** Gaining experience in data preprocessing, augmentation, model architecture design, training, and evaluation.

- **GUI Development:** Creating user-friendly interfaces using Tkinter to make the application accessible and easy to use.
- **Real-time Processing:** Implementing efficient video processing pipelines that can handle real-time camera feeds.
- **Documentation and Analysis:** Creating comprehensive Jupyter notebooks to document the analysis process and model evaluation.

4. Activities and Tasks

4.1 Data Collection and Preparation

The project uses two primary datasets: the MRL Eye Dataset for sleep detection (containing images of open and closed eyes) and the UTKFace Dataset for age prediction (containing facial images with age labels). I developed data preprocessing pipelines to organize, clean, and augment the training data.

4.2 Sleep Detection Model Development

For the sleep detection component, I designed and trained a CNN with three convolutional layers. The model takes 24x24 grayscale eye images as input and classifies them as either 'Open' or 'Closed'. Data augmentation techniques including rotation, shifting, and zoom were applied to improve model robustness.

4.3 Age Prediction Model Development

The age prediction model uses a deeper CNN architecture with four convolutional layers and BatchNormalization for better training stability. The model accepts 200x200 RGB face images and outputs a continuous age prediction. I used Mean Absolute Error (MAE) as the loss function since this is a regression task.

4.4 Detection Engine Implementation

I implemented a multi-method face detection system that first tries a DNN-based detector (SSD with ResNet) for accurate detection, then falls back to Haar Cascade classifiers if needed. The engine processes video frames in real-time, detecting faces, analyzing eye states, and predicting ages for individuals deemed to be sleeping.

4.5 GUI Application Development

The graphical interface allows users to upload images, videos, or use a live webcam feed. It displays processed frames with bounding boxes (green for awake, red for sleeping) and shows age predictions for sleeping individuals. A summary report is generated at the end of processing.

4.6 Jupyter Notebook Analysis

I created comprehensive Jupyter notebooks documenting the model training process, including dataset exploration, training visualizations (loss/accuracy plots), confusion matrices, classification

reports, and sample prediction visualizations.

5. Skills and Competencies

Through this project, I developed and strengthened the following skills and competencies:

- **Machine Learning:** Designing CNN architectures, implementing training pipelines, hyperparameter tuning, and model evaluation using metrics like accuracy, precision, recall, F1-score, and MAE.
- **Computer Vision:** Face and eye detection using multiple methods (Haar Cascades, DNN), image preprocessing, color space conversions, and real-time video processing.
- **Python Programming:** Object-oriented design, modular code organization, threading for concurrent processing, and effective use of libraries like TensorFlow, OpenCV, and NumPy.
- **Data Analysis:** Exploratory data analysis, visualization of training metrics, interpretation of confusion matrices and error distributions.
- **Software Engineering:** Creating maintainable code structure, configuration management, and building user-friendly GUI applications.
- **Problem Solving:** Debugging complex issues, optimizing performance, and implementing fallback mechanisms for robustness.

6. Challenges and Solutions

Challenge 1: Face Detection Accuracy

Problem: Initial face detection using only Haar Cascade classifiers was unreliable in varying lighting conditions and face orientations.

Solution: Implemented a hybrid detection approach that prioritizes a DNN-based face detector (SSD with ResNet) and falls back to Haar Cascades. Also tried multiple Haar Cascade variants to improve detection rates.

Challenge 2: Age Prediction Stability

Problem: Age predictions in video feeds fluctuated rapidly between frames, causing a jumpy and inconsistent user experience.

Solution: Implemented a per-face age smoothing algorithm that maintains a history of predictions for each detected face and outputs the moving average, resulting in much smoother and more stable age displays.

Challenge 3: Model Training with Limited Data

Problem: Deep learning models require substantial amounts of training data, and overfitting was a concern with smaller datasets.

Solution: Applied extensive data augmentation (rotation, shifting, flipping, brightness adjustment, zoom) to artificially increase the effective dataset size and improve model generalization.

Challenge 4: Real-time Performance

Problem: Running face detection and two CNN models on each frame created noticeable lag in video processing.

Solution: Optimized the pipeline by only running age prediction when a sleeping person is detected (not for awake individuals), reducing unnecessary computation while maintaining full functionality.

7. Outcomes and Impact

The project resulted in a fully functional drowsiness detection system with the following deliverables and outcomes:

- **Trained Sleep Detection Model:** A CNN model that classifies eye states as open or closed with good accuracy on the test set.
- **Trained Age Prediction Model:** A regression model that predicts ages with reasonable Mean Absolute Error on the UTKFace dataset.
- **Complete GUI Application:** A user-friendly desktop application supporting image upload, video playback, and live webcam processing.
- **Analysis Notebooks:** Two comprehensive Jupyter notebooks documenting the entire training and evaluation process with visualizations.
- **Organized Codebase:** Clean, modular code structure with configuration files, separate modules for engine and GUI, and training scripts.

Potential Real-World Impact: This system could be deployed in vehicles to monitor driver alertness and issue warnings when drowsiness is detected, potentially preventing accidents and saving lives. It could also be used in other contexts like monitoring operators of heavy machinery or workers in safety-critical roles.

8. Conclusion

This internship task provided valuable hands-on experience in applying machine learning and computer vision techniques to a real-world safety problem. I successfully developed a drowsiness detection system that combines face detection, eye state classification, and age prediction into an integrated application.

Key takeaways from this project include the importance of robust preprocessing, the value of multiple fallback mechanisms for critical systems, and the effectiveness of CNN architectures for image-based classification and regression tasks. The experience of building an end-to-end machine learning application—from data preparation through model training to GUI deployment—has significantly enhanced my practical skills in AI development.

Future improvements could include implementing more sophisticated drowsiness indicators (like yawn detection or head pose estimation), using larger and more diverse datasets, and optimizing the system for deployment on edge devices. Overall, this project has been a valuable learning experience that has prepared me well for future challenges in the field of artificial intelligence and computer vision.

Report prepared by Tarrush Saxena