Project Presentation

Driver Drowsiness Detection

Divit Mittal(229309249)
Date: April 26, 2025

Introduction

Road safety is significantly impacted by driver fatigue, making drowsiness detection a critical area of research and development.

- **Project Goal:** Implement a **Driver Drowsiness Detection** system.
- **Core Technology: Siamese Neural Networks** for similarity learning.
- **Mechanism:** Compare real-time facial features against baseline states.

Introduction: Fatigue Indicators

The system analyzes multiple indicators of fatigue:

- **Eye State:** Prolonged closures, rapid blinking. **Head Pose:** Nodding, slumping.
- **Mouth Movements:** Yawning.

Integrated analysis aims for timely warnings to enhance driver safety.

Features

Key functionalities:

Real-Time Detection: Continuous monitoring via camera.

Eye Aspect Ratio (EAR) Analysis: *(Potential/Planned Feature)* Quantitative measure of eye openness.
Head Pose Estimation: *(Details TBD)* Detects nodding/tilting.

Mouth/Yawning Detection: Identifies yawn characteristics.

Siamese Network Core: Compares image pairs (e.g., current vs. baseline) for state changes.

Technologies Used

Python: 3.8+ **Tensorflow (2.17.0) & Keras (3.6.0):** Core deep learning.

OpenCV: Camera access, image manipulation.

Dlib: *(Inferred)* Facial landmark detection. **Imutils:** *(Inferred)* Image processing utilities.

Numpy: Numerical operations. **Matplotlib:** Data visualization.

OS module: Path/directory operations.

Project Structure

```
.
├── .editorconfig
├── .envrc
├── .gitattributes
├── .gitignore
├── deep learning report.docx
├── flake.lock
├── flake.nix
├── LICENSE
├── pyproject.toml
├── README.md
├── siamese_network.ipynb
├── uv.lock
├── docs/
```

Note: `dataset` directory is missing.

Data Processing: Overview

Siamese Network Requirement: Paired images (similar/dissimilar).

Datasets: Eye state, Yawning detection.

Structure: Anchor, Positive (similar), Negative (dissimilar) samples.

Data Processing: Paths & Setup

Required directory structure (under `dataset/`):

Eyes Dataset:

`eyes/anchor/` `eyes/positive/` `eyes/negative/`

Yawn Dataset:

`yawn/anchor/` `yawn/positive/` `yawn/negative/`

Important: `dataset` directory needs to be populated externally.

Data Processing: Preprocessing Pipeline

TensorFlow (`tf.data`) pipeline steps:

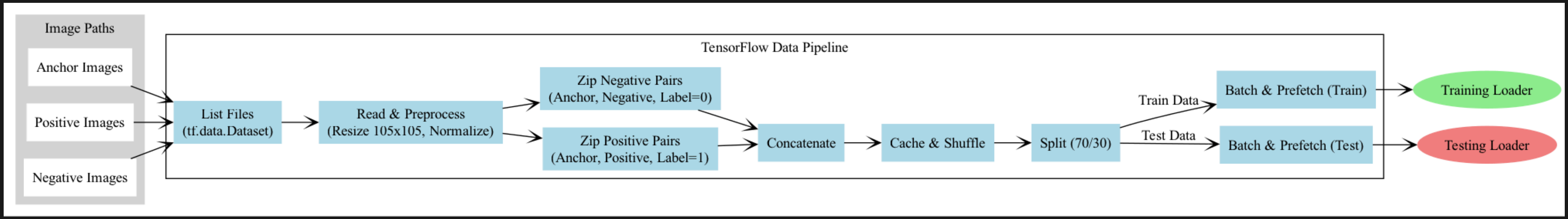
List Files (`.jpg`) Read Files Decode JPEG Resize (105x105) Normalize (0-1)

Pair & Label (Anchor/Positive=1, Anchor/Negative=0) Concatenate Pairs Cache Shuffle

Train/Test Split (70/30) Batch (Size 16) Prefetch

Data Processing: Pipeline Diagram

This diagram illustrates the data flow:



Model Architecture: Siamese Network

- **Purpose:** Similarity comparison (e.g., same state or different state).
- **Core:** Shared CNN acts as an **embedding generator**.

Model Architecture: Embedding Network (CNN)

Input: Image (105x105x3) **Output:** Embedding Vector (4096 dimensions)

Key: Shared weights for both images in a pair.

Architecture:

Conv Block 1 (64 filters, 10x10) -> MaxPool Conv Block 2 (128 filters, 7x7) -> MaxPool

Conv Block 3 (128 filters, 4x4) -> MaxPool Conv Block 4 (256 filters, 4x4) Flatten

Dense (4096 units, Sigmoid) -> Embedding

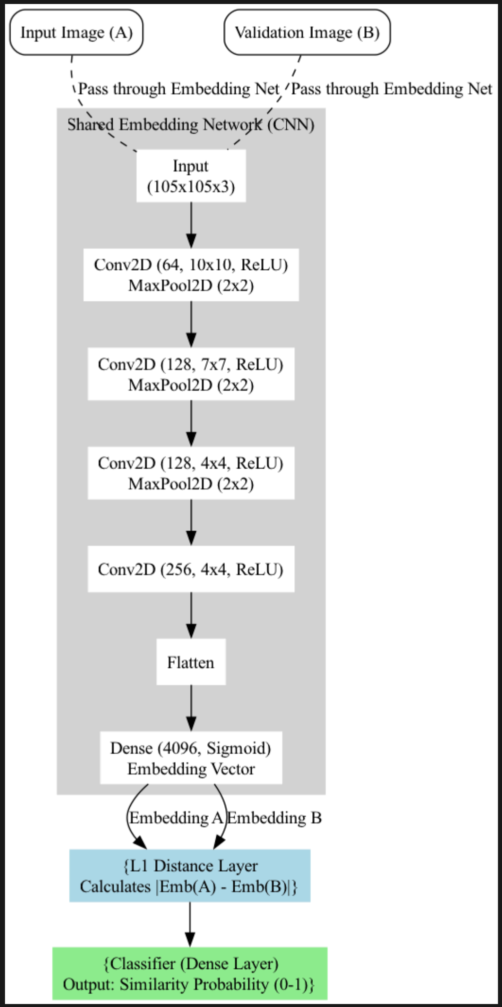
Model Architecture: Siamese Structure

Inputs: `input_image` (A), `validation_image` (B).

Embedding: Both A & B pass through the **shared** Embedding Network -> Emb(A), Emb(B).
Distance: `L1Dist` layer calculates |Emb(A) - Emb(B)|.

Classifier: Dense layer (1 unit, Sigmoid) predicts similarity (0-1).

Model Architecture: Diagram



Training Configuration

Optimizer: Adam (learning rate 1e–4) **Loss Function:** Binary Crossentropy

Training Loop: Custom `@tf.function` loop (`train_step`)

Epochs: Iterates, uses `Progbar` for progress.

Checkpoints: Saves model state every 4 epochs (`./training_checkpoints`).