3D - Driver Distraction Detection

Francesco Zampirollo
271664@studenti.unimore.it

Stefano Politanò
267705@studenti.unimore.it

Vincenzo Macellaro
322350@studenti.unimore.it

1 Abstract

Driver distraction is a significant contributor to road accidents and poses a substantial threat to road safety. In this project, we aim to address this critical issue by leveraging advanced technologies, including Computer Vision and Machine Learning algorithms, to detect and mitigate driver distraction effectively. Our system analyzes driving behaviors and assesses the level of risk involved, with the ultimate goal of reducing accidents caused by distracted driving.

We introduce a comprehensive approach that combines pose estimation and distraction source detection to create a robust driver monitoring system. Pose estimation, powered by the MediaPipe framework, provides insights into the driver's posture and behavior, enabling us to identify signs of distraction. Simultaneously, a fine-tuned YOLOv7 Convolutional Neural Network (CNN) model is trained to detect distraction sources such as "drinks" and "smartphones" within the vehicle.

The seamless integration of our components facilitates precise driver behavior classification into five risk categories, spanning from safe driving to high-risk distraction scenarios. Our custom Graph Convolutional Network (GCN) model processes input data from both YOLO detection and MediaPipe, enabling the classification of driver behavior across a spectrum of five distinct classes, ranging from 'zero-risk' to 'very-high-risk'.

Furthermore, we have implemented a retrieval algorithm using Faiss library and K-nearest neighbors (KNN) technique to retrieve the K-most similar images in our dataset when given a new test image. This allows us to enhance the understanding of driver behavior and further refine our risk assessment.

Our preliminary results show promising accuracy, with an average of 90% in our driver distraction detection task, indicating the effectiveness of our system. By alerting drivers or initiating corrective measures when distraction is detected, our system has the potential to significantly enhance road safety and save lives.

This project demonstrates the feasibility of using advanced technology to combat driver distraction, providing a foundation for future research and development in this critical area of road safety.

2 Introduction

Driver distraction poses a grave threat to road safety, leading to frequent accidents. In response, our project harnesses the power of Computer Vision and Machine Learning to detect and mitigate distractions effectively, thereby enhancing overall safety.

Distracted driving, encompassing activities from texting to in-car adjustments, demands a vigilant solution. Our integrated system employs Computer Vision techniques, including pose estimation and object detection. MediaPipe's pose estimation provides crucial insights into the driver's posture and behav-

ior, enabling the detection of signs of distraction or fatigue. Simultaneously, our fine-tuned YOLOv7 model adeptly identifies key distractions, focusing on "drinks" and "smartphones."

A standout feature of our system is the custom Graph Convolutional Network (GCN). This GCN integrates YOLO detection and pose estimation data, allowing the classification of driver behavior into five distinct categories. To delve deeper into understanding driver behavior, we utilize a retrieval algorithm with Faiss and K-nearest neighbors (KNN).

Our preliminary results showcase promising accuracy, with ongoing refinements expected. The project not only underscores the role of technology in road safety but also lays the foundation for future research in this critical domain.

In subsequent sections, we delve into our methodology, present experimental results, discuss challenges and limitations, and outline future directions.

3 Related Work

Distracted driving is a major safety concern, and researchers have developed a variety of machine learning-based approaches to detect and warn drivers who are distracted. One common approach is to use convolutional neural networks (CNNs) to analyze video footage of the driver's face and hands to identify distracted behaviors such as talking on a cell phone, eating, or looking away from the road.

[3] Baheti et al. proposed a CNN-based system for detecting distracted drivers that achieves an accuracy of 96.31%. Their system uses a modified VGG-16 architecture and various regularization techniques to improve performance. They also studied the effect of different dropout, L2 regularization, and batch normalization parameters on the performance of the system.

[5] Liang et al. developed a real-time approach for detecting driver cognitive distraction using support vector machines (SVMs) to analyze eye movements and driving performance data. Their system achieved an average accuracy of 81.1%, with the best performing model achieving an accuracy of 96.1%.

[1] Abouelnaga et al. presented a new dataset for distracted driver posture estimation and a novel system that achieves 95.98% driving posture estimation classification accuracy. Their system consists of a genetically-weighted ensemble of CNNs. They also studied the effect of different visual elements (i.e. hands and face) in distraction detection and classification.

These studies show that machine learning can effectively detect distracted drivers. However, improvements are needed in accuracy, robustness, and real-time performance. Future research might explore detecting a broader range of distracted behaviors, improving robustness to noise and lighting variations, and creating real-time, low-power hardware systems for deployment in vehicles.

4 Data Preparation

This section outlines the data preparation process for both the YOLO model and the Graph Convolutional Network (GCN) model, utilizing the "State Farm Distracted Driver Detection" dataset [2]. This dataset contains images categorizing drivers into ten behavior classes, from "safe driving" to distractions like texting or phone use.

For image quality, we applied pre-processing using Contrast Limited Adaptive Histogram Equalization (CLAHE) to enhance contrast and improve visibility of critical details. This improved dataset was used to train our fine-tuned YOLO model for accurate detection of "drinks" and "phones" in vehicles.

For the GCN model, we utilized the MediaPipe Pose framework, a real-time human pose estimation solution. This framework accurately detects and tracks 33 3D keypoints associated with a person's body, providing crucial spatial information for precise 3D pose estimation [1].

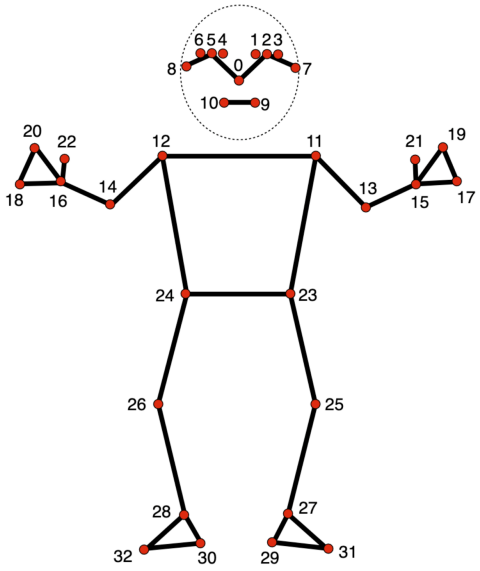


Figure 1: MediaPipe Pose 3D Landmarks

Incorporating insights from our YOLO model, we enriched spatial data, resulting in 100-length feature arrays for each image. These arrays were saved as CSV files, forming a comprehensive dataset.

While NetworkX aided in visualizing data conceptually (see Figure 2), its primary role was for visualization purposes. We used it to represent landmarks as nodes, connecting edges based on the MediaPipe Pose scheme, transforming the original 100-dimensional array into a graph structure.

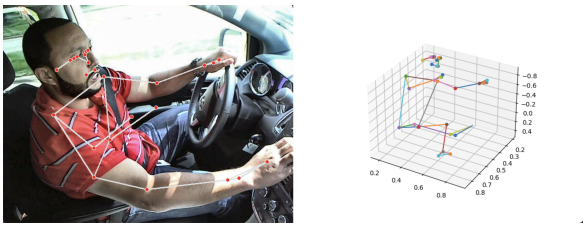


Figure 2: Graph Visualization

For GCN model training data preparation, we conducted a detailed analysis, examining geometric relationships, calculating relative coordinates, and determining angles between nodes and their neighbors. These steps allowed us to construct [33, 15] tensors as a fundamental reference for the subsequent GCN model construction, ensuring alignment with specific data requirements.

Subsequent sections delve into YOLO model training for object detection and GCN model training for behavior classification.

5 Methodology

5.1 Object Detection

Our fine-tuned YOLO model precisely identifies 'drinks' and 'phones' within images, contributing categorical labels (0 for drinks, 1 for phones) and bounding box coordinates. When no object is detected, a label of -1 seamlessly integrates into our graph and tensor creation process, enriching our data representation.

5.2 Pose Estimation

Pose estimation is crucial for driver behavior analysis, extracting 3D landmark coordinates from images to understand posture and detect distraction signs. We use the MediaPipe Pose framework to track essential body parts, offering insights into attentiveness and engagement. For example, a turned head or hands off the steering wheel may suggest distraction or fatigue. This detailed anatomical information forms the foundation for behavior analysis.

5.3 Integration

The integration of data for our system is a meticulously crafted process that seamlessly merges insights from both the MediaPipe Pose and YOLO object detection components. This pipeline is designed to extract and fuse critical information, enabling robust behavior classification. The true power of our data integration lies in the fusion of MediaPipe Pose and

YOLO-derived insights within a unified graph structure. Here's how this integration process unfolds:

1. **Node Creation**: Each 3D landmark from MediaPipe Pose is a node in the graph, initially featuring fundamental spatial coordinates (x, y, z). Nodes are enhanced with attributes, including YOLO detection labels (indicating drink or phone presence), relative coordinates with neighbors, and Euclidean distance and angular information. This comprehensive feature set enriches node representations.

2. **Graph Edges**: Graph edges signify anatomical connections between body landmarks, adhering to human anatomy and the MediaPipe Pose scheme.

3. **Integrated Graph Insight**: The graph unifies driver 3D body orientation with distraction information. The GCN model interprets posture and identifies distractions using the graph and its adjacency matrix. This matrix, crafted through degree matrix calculation, symmetric normalization, and self-connections, enhances input for our neural network-based classification system. This unified pipeline efficiently captures and analyzes driver behavior, seamlessly integrating pose estimation and object detection for comprehensive insights.

6 System Architecture

6.1 Fine-Tuned YOLOv7 Model

Our system integrates a fine-tuned YOLOv7 model, a renowned real-time object detection framework [6]. We initialized the model with preset weights and conducted focused re-training on our State Farm Distracted Driver Detection dataset, enhancing its proficiency in identifying distractions like "drinks" and "phones."

The YOLOv7 model provides crucial insights with labels (0 for drinks, 1 for phones) and bounding box coordinates. Label assignment, including a special case (-1) for no detections, seamlessly integrates into subsequent graph and tensor creation, enriching our data representation.

This fine-tuned model significantly improves our system's ability to detect and classify distractions, establishing YOLOv7 as a pivotal component in our comprehensive driver behavior analysis.

6.2 GATNet

The GCN model, named GATNet (Graph Attention Network), is designed to process 3D graph data representing human body landmarks and object detection extracted from images. It aims to classify driver behavior into one of five distinct risk classes: "ZERO", "LOW, "MEDIUM", "HIGH", and "VERY_HIGH". It comprises the following key components:

- **Graph Convolutional Layers**: GATNet utilizes two Graph Attention Network convolutional layers (GATConv) to capture complex graph relationships. GATConv excels in learning intricate graph structures by assigning varying levels of attention to neighboring nodes during convolution.

- **Skip Connections**: GATNet enhances feature propagation with skip connections, employing additional GATConv layers. These connections facilitate effective information and gradient flow within the network.

- **Readout Layer**: Following graph convolutional layers, a readout layer aggregates node-level information to produce the final classification output. This process involves linear transformations and a Rectified Linear Unit (ReLU) activation function for valuable global feature extraction.

- **Dropout Layers**: To prevent overfitting, dropout layers with a 0.2 dropout probability are inserted after each graph convolutional layer. Dropout randomly deactivates network neurons during training, promoting generalization.

The model is configured as follows:

- **Input Dimension:** 15, as seen in Section 4:
- **Hidden Dimension:** 256, balanced for model complexity and performance.
- **Output Dimension:** 5, aligning with behavior classes.
- **Number of Attention Heads:** 6, empirically determined for optimal performance.

7 Experimental Results

7.1 Fine-Tuned YOLOv7 Model

Performance metrics for the YOLO model training are available in the Appendix (see Figure 5). The model exhibits an average accuracy of 0.99 in object detection - 1.00 for the "drink" class and 0.98 for the "phone" class - contributing to the real-time effectiveness of the system.

7.2 GATNet

7.2.1 Loss Function

GATNet employs Focal Loss for training, which effectively addresses class imbalance by prioritizing challenging examples. We explored alternative loss functions like CrossEntropy and WeightedCrossEntropy but found Focal Loss consistently outperformed them. Our training strategies encompassed two approaches: one using CrossEntropy on a balanced dataset and another with Focal Loss on an imbalanced dataset. The latter consistently yielded superior results.

7.2.2 Training Procedure

The GATNet model was trained using the following configuration:

- **Optimizer**: Adam optimizer with a learning rate of 0.001 and weight decay of 1×10^{-7}. These hyperparameter values were carefully selected through extensive research and experimentation. Deviating from these values, either higher or lower, resulted in a decrease in overall model performance.
- **Batch Size**: A batch size of 64 was utilized during training.
- **Training Dataset**: The following table shows the distribution of samples in each class for the training dataset:

Class ID	Samples
0	7728
1	22246
2	11376
3	13011
4	14488

- **Validation Dataset**: The following table shows the distribution of samples in each class for the validation dataset:

Class ID	Samples
0	637
1	1942
2	752
3	1021
4	1235

- **Training Strategy**: Training was configured to run for a maximum of 1000 epochs. However, early stopping was implemented with a patience value of 10, leading to early termination around the 16th epoch. No specific learning rate schedules were applied as they did not provide meaningful improvements for our problem.
- **Performance Metrics**: Model performance was evaluated using the Focal Loss as the primary loss function and accuracy as a metric for classification performance.

- **Training Environment**: The training of GAT-Net was conducted on a 2021 MacBook Pro equipped with the M1 Pro SoC, featuring 8 CPU cores and 14 GPU cores, along with 16GB of RAM. GPU acceleration was utilized during training.

7.2.3 Results

In our most successful training run, we attained an average Epoch Accuracy of 0.9033 and an Epoch Loss of 0.1840. Here are the class-specific accuracy values:

Class	Average Accuracy
0	0.9034
1	0.8965
2	0.9265
3	0.8804
4	0.9156

The confusion matrix, as well as the detailed performance graphs for each class are provided in the Appendix (see Section B.1).

7.3 Retrieval

The final step in our pipeline is the retrieval process, as previously mentioned. Our objective is to leverage the Faiss library [4] to calculate distances between one or more queries and the embeddings generated from the training dataset. The goal is to obtain a classification of the input image based on the classified retrieved image. To create embeddings, we used the output from the GCN, which served as a robust representation of the input data. We preserved this representation along with the labels assigned by the GCN, ensuring the association between the index and the respective embedding is maintained.

Faiss returns two matrices, namely **I**, index matrix, indicating the index of the element most similar to the input query, and **D**, distance matrix, presenting distances from the input query in ascending order.

For each query, we analyze indexes, match them to the corresponding image, and visualize the results. Faiss requires queries and embeddings as NumPy arrays for the search phase. We fed queries into the GCN, using the returned embedding as the query embedding, then generated the query matrix, embedding matrix, and performed the search retrieval.

Faiss library offers exhaustive and approximate index categories. The exhaustive approach precisely calculates distances but lacks compression, making it less optimal for large datasets. The approximate approach involves compression operations, introducing distance approximation.

Given our modest retrieval dataset, we opted for `IndexFlat2D` in the exhaustive category for optimal results. This index operates based on the L2 distance:

$$L2Distance = \sqrt{\sum_{i=1}^{n}(b_i - q_i)^2} \tag{1}$$

After obtaining distances and indexes from Faiss, we return and rank images closest to each query. Next, we implement the K-Nearest Neighbors (KNN) algorithm among the K nearest neighbors (K chosen arbitrarily) to make a majority prediction for the query, identified as the predicted y (Figure 3).

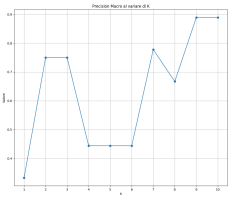
In the final step, individual queries were passed to the GCN to obtain corresponding labels, considered as ground truth. This entire process was repeated for a chosen number of queries, varying K within a range of values.

Precision, accuracy, and recall results were then measured, identifying K=9 as providing the best overall performance (Figure 4).

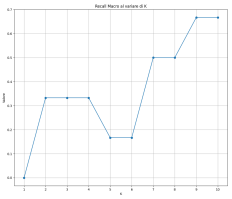
8 Conclusion

In conclusion, our research project has provided satisfactory results throughout the entire pipeline. The accuracy of the system can be further improved by adjusting its intermediate steps and striving to enhance the precision of the components (YOLO and Mediapipe).

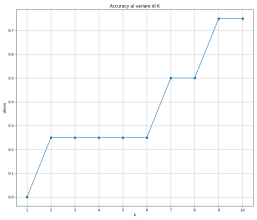
Figure 3: Retrieval Test



(a) Precision



(b) Recall



(c) Accuracy

Figure 4: Retrieval Performance Measures

References

[1] Y. Abouelnaga, H. M. Eraqi, and M. N. Moustafa. Real-time distracted driver posture classification. June 2017. 2

[2] S. T. S. W. K. Anna Montoya, Dan Holman. State farm distracted driver detection, 2016. 2

[3] B. Baheti, S. Gajre, and S. Talbar. Detection of distracted driver using convolutional neural network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1145–11456, 2018. 2

[4] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 6

[5] Y. Liang, M. L. Reyes, and J. D. Lee. Real-time detection of driver cognitive distraction using support vector machines. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):340–350, 2007. 2

[6] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

A YOLO Metrics

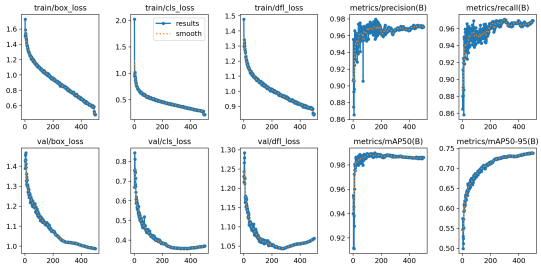


Figure 5: Performance plots

Figure 6: Validation Batch Results

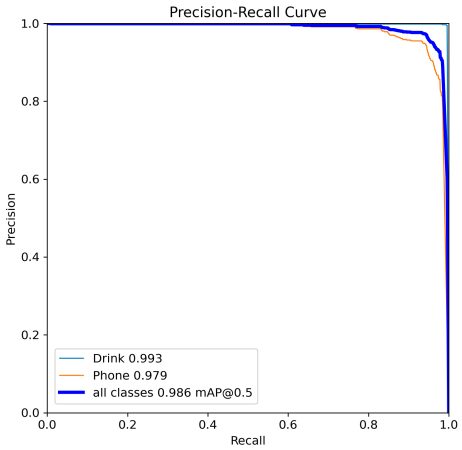
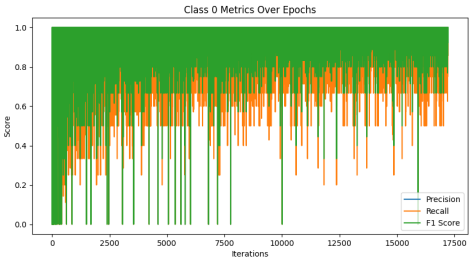
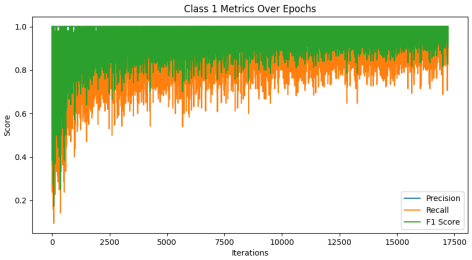


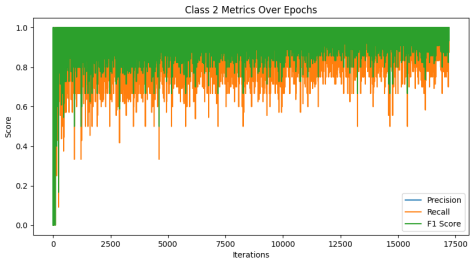
Figure 7: PR Curve

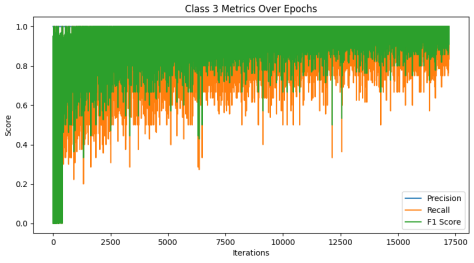
B GATNet Metrics

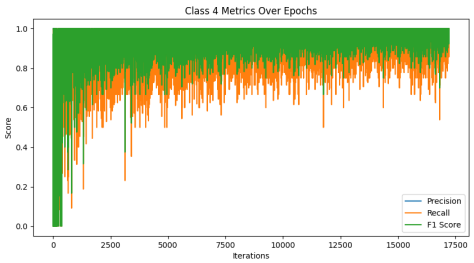
B.1 Precision, Recall, and F1 Score











B.2 Confusion matrix

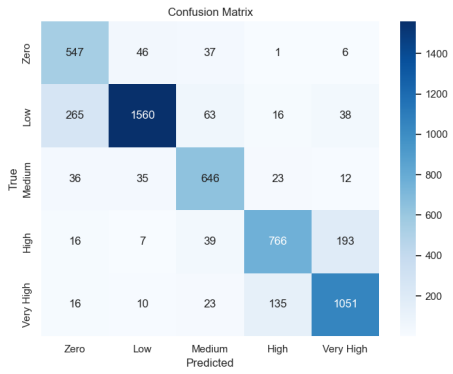


Figure 8: Confusion Matrix