Project Proposal
Driver Attention Detection System using CNN

Team - Phoenix

Arpitha Gurumurty, Gayathri Pulagam, Surabhi Govil

Abstract

Driver Attention Detection: Distraction behind the wheel has increased accidents and traffic congestion very much in recent times . There have been 72000 reported crashes on the national highway in 2013. A number of people have been injured due to the driver's distraction.

Driver distraction can be defined as the driver not paying attention to the road and instead doing activities like drinking, talking on the phone, texting or talking to a fellow passenger.

We are not just pointing out the physical state of distraction but also the brief moments where one might lose attention, for example turning behind for a moment. To tackle this problem we aim to create a solution to detect driver attention through objects in the image and prevent accidents in the early stages.

Objective

To classify the user's attention level into one of these 10 categories distraction based on the driver's activity in the image. To do this, we plan on using Keras with CNN based architecture.

Datasets

We plan to use State Farm data which consists of 100k images split between train and test sets. The images show drivers doing activities like texting on the phone, talking on the phone or talking to a fellow passenger.

The images are divided into 10 classes, each class represents the activity of the driver in the image.

Expected results

1. By the end of the project, we expect to have a model which is able to classify a driver's attention level into 10 categories based on their facial expressions.
2. The model should be deployed to an end-point using **Airflow** and should be fully functional to get inferences from the trained model.

Metrics and visualizations

1. AUC-ROC curve to evaluate the classification of the model. Because our problem is a multi-class, we are going to use **One vs All** technique[4].
2. Precision & Recall scores (using micro and macro averages)
3. Confusion matrix to show the combination of actual and predicted classes.
4. Cross-entropy to measure the match of actual data to the predicted probabilities.

5. Class prediction error charts to visualize the misclassified classes as a stacked bar[5].
6. To visualize the metrics and track the performance of our model we will use **TensorBoard**.

Plan for Demo (Deployment of the model)

We plan to use Apache Airflow to orchestrate our machine learning pipeline to deploy our model and demonstrate it in real-time. We plan to deploy it as a web app using flask.

Team Member Roles

1. **Arpitha Gurumurthy** - Model Deployment to Airflow, Data exploration
2. **Gayathri Pulagam** - Data preparation, feature engineering, data ingestion methods, model evaluation and visualizations
3. **Surabhi Govil** - Feature engineering, model building, research, TensorBoard setup, visualizations, model evaluation

End deliverables

1. Deployed model's endpoint for testing the model in real-time.
2. Team git repository's URL with the entire code used in the project
3. Model's metrics and visualizations (TensorBoard)
4. Project Report providing the details of implementation, methodologies used, evaluations and challenges encountered during the project.
5. Presentation slide deck which will be used by the team in class to walk through the project.

References

1. https://www.cdc.gov/sleep/features/drowsy-driving.html#:~:text=An%20estimated%201%20in%2025,in%20the%20previous%2030%20days.&text=The%20National%20Highway%20Traffic%20Safety,and%20800%20deaths%20in%202013.
2. https://sites.google.com/view/utarldd/home
3. https://arxiv.org/pdf/2001.05137.pdf
4. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/
5. https://medium.com/apprentice-journal/evaluating-multi-class-classifiers-12b2946e755b
6. https://arxiv.org/pdf/1904.07312.pdf
7. https://www.hindawi.com/journals/cin/2020/7251280/