Distracted Driving Detection Using Convolutional Neural Networks

Trevor Rrizzo
Robotics Engineering
Worcester Polytechnic Institute
Worcester, USA
tarizzo@wpi.edu

***Abstract*—Distracted driving is a leading cause of motor vehicle accidents. One of the most popular driver distractions is cellular phones. This study utilizes convolutional neural networks (CNNs) and time series data to predict if a driver is distracted. The features used will include sensor data and user input data from a automotive driving simulator. This paper will also detail the data collection techniques utilized in this study.**

I. Introduction

Machine learning and CNNs are the focus of many bodies of work focusing on the automation of driving. The goal of this research is to keep human drivers safer by utilizing CNNs to determine if a driver is distracted. The proposed method utilizes vehicle sensor data, as well as driver inputs, as features. All training data was collected using automotive driving simulation software and a USB steering wheel and pedals. Typing on a cellular while using the simulator was used to emulate distracted driving.

A. Distracted Driving

Driving a motor vehicle requires the constant coordination of a driver's vision, sensory perception, cognitive processing, and motor function. If any one of these factors is impaired, a driver's ability to safely operate a motor vehicle is reduced. Many ordinary items and events can draw a driver's focus away from the act of driving and serve as a mode of impairment. Certain distractions can impact a driver's abilities by drawing their hands and/or feet away from the control surfaces. Distracted driving contributes to motor vehicle collisions in drivers of all ages, many of which can be contributed to cellphone usage while driving [1].

Using road traffic data collected in Spain along with Bayesian networks, García-Herrero *et al.* show that technology-based distracted driving has a strong correlation to speeding infractions. The data also showed a correlation between speeding infractions and the severity of traffic accidents [2].

B. Time Series Convolutional Neural Networks

The use of convolution allows neural networks to be applied to applications with time series data. The input data for these CNNs is a series of data points over time, often with a constant time step. The convolutional layers are used to convolve for each field over time and extract any temporal relationship between the input data and the output truth. The applications of using CNNs with time series data spans many fields including finance, robotics, and the medical field. Time series data and CNNs can be used in both regressing and classification use cases.

Liu *et al.* propose a general architecture for time series classification. The proposed architecture is designed for multivariate time series data. The proposed architecture performs uni-variate convoltution, convolving over time for each variable. Next, multi-variate convolution is performed on all the features. Finally, the feautures are fed to a fully connected layer. The proposed architecture uses a cross-entropy loss function. To validate the architecture, it is trained to predict the performance of equipment based on previous maintenance. The results of the experiments show that the proposed architecture outperforms other architectures including a more conventional CNN and gradient boosted trees [3].

Using stock market data Sezer *et al.* developed a CNN to trade stocks on the stock market. The CNN convolves over the last 15 days of samples containing technical indicators and classifies the stock a buy, sell, or hold. The proposed methodology used 2D convolution techniques commonly used in image classification to convolve 15 days of 15 data points. The convolved data is then passed to a max polling layer before it is finally passed to two fully connected layers, including the output layer. This technique was able to outperform the standard "buy and hold" strategy on 86% of the analyzed stock over a full year of test data [4].

Xia *et al.* successfully use a time series convolutional network to predict the the force applied by microsurgery robots in real time. This research uses 3 time series images of the microsurgery robots as inputs with the intent of using the deformation of the surgical tool over time to calculate the applied torque. This model has individual convolutional and fully connected layers for each image. The feature vectors of all the individual fully connected layers are concatenated before passing through a final fully connected layer. This model utilizes a linear activation function. This model was able to extract the applied torque from the time series imagery with R^2 values above 0.92.

II. Related Work

This section will discuss existing works focused on identifying when a driver is distracted.

Nakano *et al.* use deep learning to detect if a driver is distracted. An automotive driving simulator was used to collect time series data from participants. The collected features include steering angle, steering torque, brake stroke, car speed, and engine speed. Data was collected in scenarios that simulate both rural and urban driving. Data was collected while the participants were in 1 of 3 states of cognitive impairment; 1) not impaired, 2) impaired by conversation, 3) impaired by arithmetic problem. The data set was used to train 3 networks a k-nearest-neighbor (kNN) classifier with dynamic time warping, a kNN classifier with Euclid distance, and a 1D CNN. Both kNN classifiers were more accurate than the CNN at predicting the drivers state. Data from each participant was used to train participant-specific networks, as well as a network trained on all the participants' data, for each network architecture. The authors concluded that due to the varying driving skills and techniques of each driver, the user-specific models outperform models that are trained and evaluated on many users [5].

Ahmed *et al.* explored using accelerometer and gyroscope data from cellular phones to detect distracted driving. This study sampled accelerometer and gyroscope data from cellular phones at 200Hz while the participants drove the simulator. After a 2 minute warm up in the simulator, the participants interacted with the phone to text and make calls while driving the simulator. Additional features were extracted from each group of n samples, including maximum acceleration, minimum acceleration, and square sum variance of gyroscope. The engineered features were used to train a random forest regressor. Participant-specific models and a cross-participant model were trained with the dataset. With single participant data 85% precision was achieved, with cross-participant data 80% accuracy was achieved [6].

Tran *et al.* use images of drivers and deep learning to identify distracted drivers. This study also utilized a driving simulator to collect training data. Images of the participants were capture continuously while they drove the simulator. While driving, the participants performed 10 different actions that the researchers identified as actions that distract from driving. The actions include manual distractions, visual distractions, and cognitive distractions. The images collected from the data collection were used to train 4 different classifiers. The researchers utilized transfer learning for the 4 classifiers, using existing architectures from other bodies of work. The 4 models utilized are: 1) VGG-16, 2) AlexNet, 3) GoogleNet, 4) ResNet. These model were able to achieve accuracies of 79%, 81%, 83%, and 88% respectively [7].

III. Data Collection

This section will discuss the methods used for data collection for this study, including the technologies and methods used to collect the data as well as which features were recorded.

Fig. 1. Driving simulator first person view.

Fig. 2. Thrustmaster T150 USB steering wheel and pedals [9].

A. Driving Simulator

An automotive driving simulator was utilized to collect the data for this study. The software used for the simulator is BeamNG.drive, a driving simulator built around soft-body physics engine that accurately models the kinematic properties of vehicles [8]. The Thrustmater T150 USB steering wheel and pedals, seen in Fig. 2, was used as user inputs for the simulator. The Thrustmater T150 contains a force-feedback motor that allows BeamNG.drive to apply a torque to the steering wheel that matches the torque in the physics engine. Force feedback provides a more realistic driving experience and contributes to more data the more closely resembles the data that could be collect from a real automobile. Fig. 1 shows the cockpit view within the driving simulator, which gives a first person driving perspective to the participant. Data collection was achieved by editing the Lua source code of BeamNG.drive to publish the desired data to a UDP socket. A python script was written to interface with BeamNG.drive via the UDP socket, in order to collect time series data. Data from the simulated car's sensors and the user inputs were sampled at 10 Hz and recorded to a CSV file.

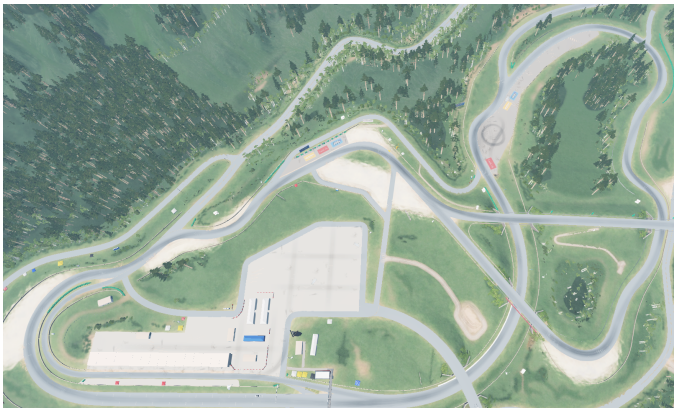

Fig. 3. Closed course driving scenario.

B. Driving Scenarios

Two driving scenarios were chosen to emulate the various conditions found in real-life driving. A closed course with no traffic was used to emulated rural roads with many curved sections and only short continuous straight sections. A highway with AI vehicle traffic was used to emulate urban driving. Fig. 3 shows an overhead view of the closed course driving scenario.

C. Distraction

The distraction chose for this study is typing on a cellphone. This was chosen because it is a very common distraction and it falls under all three categories of distraction. Typing on a cellphone is a visual distraction because the user must look at the cell phone, it is a cognitive distraction because the user has to think about using the cellphone and what they are typing, and it is a physical distraction because the user must move at least one hand off the controls to type on a cellphone. Speed Typer, a typing test app, was used to simulate texting a cell phone. This app requires the user to correctly type a continuous stream of random words for a predetermine amount of time. Users used their own cellular phones durring the data collection to eliminate the possible cognitive load caused by using an unfamiliar device.

D. Features

The features captured during the data collection process are features that encompass the participant driving inputs as well as the behavior of the car that result from those inputs. The features that encompass the driver's input are 1) throttle input 2) brake input 3) steering input angle. . The features that encompass the car's state are 1) anti-lock braking system status 2) traction control system status 3) vehicle airspeed 4) vehicle wheel speed 5) engine speed 6) vehicle throttle position 7) vehicle brake throw 8) vehicle steering position 9) currently engaged transmission gear. When analyzed over time, these features have the potential to reveal if sudden inputs were applied and if the vehicle's state matched the intended driver inputs. The feature used as the truth feature in the data set was the drivers state of distraction. This feature was set to 1 for the distracted data collections and 0 for non-distracted data collections.

E. Procedure

All participants used for data collection were licensed drivers familiar with driving real automobiles. Additionally, the participants were given an hour to become familiar with the driving simulator. After the warm-up session, multiple data collection sessions were performed for each participant, each totaling 5 minutes. For each 5 minute data collection, the participant would alternate between distracted and non-distracted driving. Consistently alternating means that the participants increase in driving skill over time does not effect the data. All of the data collection was performed with the same automobile to ensure the driving characteristics and input response of the vehicle was consistent across all data collections. Additionally, the chosen vehicle was a 4-door sedan with a typical level of power and torque and an automatic transmission. This vehicle was chosen because it is a generalized representation of a vehicle a typical driver would encounter in a real-world scenario.

IV. Methods

This section will details the methods used to process the collected data, design the CNN, and train the CNN. The goal of the proposed CNN is to take a time series data set from the data collection script and classify the driver as being distracted or not distracted.

A. Data Processing

The first step in training a distracted driving classifier was to process the collected data. First, the 'currently engaged transmission gear', the only categorical feature was one hot encoded with a column reserved for each possible gear selections in the simulator. Next all the numerical features were normalized to their maximum possible value in the simulator. For example, the maximum possible engine speed of the chosen vehicle is 7000 RPM. Therefore, all engine speed data points were normalized to 7000, even if no data points in the sample reached 7000 RPM. A consequence of the data collection processed used is that the beginning and ending of the data recordings contain data points where the vehicle is not in motion. Vehicle data with no motion cannot reveal if a driver is distracted and could unfavorably effect the weights of the trained network. As such, the data recordings were trimmed such that the first data point became, the first data point with a non-zero vehicle wheel speed, and the last data point became the last data point with a non-zero wheel speed.

B. Network Architecture

The proposed network utilizes convolution layers to extract features from the time series data. The first three layers of the proposed CNN are 1 dimensional convolutions layers that convolve each feature in the time dimension. The convolutional layers increase in stride, kernel size, and output layers as the network progresses. The stride and kernel size were chosen

such that no features are step over by the kernel. Additionally, the padding for the convolution layers was calculated with the function $floor(KernelSize/2)+1$. Each convolutional layer utilizes a ReLU activation function and is regularized using dropout.

Four fully connected layers, including the output layer, are utilized in the proposed architecture. The output of the third convolutional layer is flattened and passed to a fully connected layer. The output size of the fully connected layers decrease as the network progresses, with the final fully connected layer have one output node. Each fully connected layer utilizes a ReLU activation function and is regularized using dropout. The output of the final fully connected layer is passed to a signmoid function to bound the output between 0 and 1. The full network architecture can be seen in Table I.

Layer (type)	Output Shape
Conv1d-1	[-1, 30, 602]
ReLU-2	[-1, 30, 602]
Dropout-3	[-1, 30, 602]
Conv1d-4	[-1, 90, 302]
ReLU-5	[-1, 90, 302]
Dropout-6	[-1, 90, 302]
Conv1d-7	[-1, 1000, 61]
ReLU-8	[-1, 1000, 61]
Dropout-9	[-1, 1000, 61]
Flatten-10	[-1, 61000]
Linear-11	[-1, 1800]
ReLU-12	[-1, 1800]
Dropout-13	[-1, 1800]
Linear-14	[-1, 200]
ReLU-15	[-1, 200]
Dropout-16	[-1, 200]
Linear-17	[-1, 30]
ReLU-18	[-1, 30]
Dropout-19	[-1, 30]
Linear-20	[-1, 1]
Sigmoid-21	[-1, 1]

TABLE I
CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE

C. Training

The first step in training the network was to split the data into a training set, a validation set, and a testing set. The data was split 80%, 10%, and 10% respectively. The validation set was used to show that the network was not being over-trained to the training set and the test set was used to analyze final network performance. During the process of developing and training the network, many parts of the architecture were changed to improve the network's performance. The biggest adjustment were made to the number of output channels of the convectional and fully connected layers. The number of output channels were increased until an additional increase made no significant impact on the performance of the network. Additionally, the length of each set of time series data was used as a training hyperparameter. During the training process this parameter ranged from 30 seconds of data to 120 seconds of data. Changing the length of the time series data also required change the number of input channels of the first fully connected layer. The network architecture and hyperparameter changes were made in a manner that minimized loss and maximized accuracy on the training and validation set.

D. Performance Validation

In order to valid the performance of the proposed architecture, the results of the trained network were compared against an existing architecture. A dynamic time warping k-nearest-neighbor classifier was chosen as the validation architecture. This network architecture is commonly used to classify time series data. To compare the two networks, the dynamic time warping k-nearest-neighbor classifier was trained and evaluated on the same data as the proposed network.

V. RESULTS

This section will detail the results of the data collection process and the performance of the proposed network.

A. Data Collection

The data collection process was performed with 3 participants. Each participant attended 3 data collection sessions. The data collected totals about 18 hours of data. The process of continuously alternating between distracted and non-distracted driving resulted in a data set that is 50% distracted driving and 50% non-distracted driving. About 60% of the data collected is from the rural driving scenario and about 40% of the data is from the highway scenario.

B. Network Performance

The results of the training process show that the optimal time series length for distracted driver classification is 60 seconds. The first attempts at training the network combined the data from all the users over both driving scenarios. This resulted in a training accuracy of 85% and and testing accuracy of 82%. Next, the data was divided by driving scenario. Training on the dividing driving scenarios resulted in a 86% training accuracy and 84% testing accuracy for the highway scenario and 82% training accuracy and 81% testing accuracy for the closed-course scenario. In the final experiment, the data set was divided by driver, the best network from this experiment resulted in an 90% training accuracy and an 88% testing accuracy.

A dynamic time warping k-nearest-neighbor classifier was trained on the data from the best single-driver network. The dynamic time warping k-nearest-neighbor classifier achieved a 83% training accuracy and a 79% testing accuracy.

VI. DISCUSSION

The results show that the proposed CNN is able to outperform a dynamic time warping k-nearest-neighbor classifier. The results validate the network architecture and the data collection techniques. Additionally, the results show that the features associated with distracted driving can be driver-specific, but distracted driving can exhibit similar behaviours in multiple driving scenarios. More data collection is need

to confirm the results presented in this paper. It is possible that with more data, collected from more participants, a more accurate generalized model could be trained.

VII. Future Work

Future work for the presented methods is to conduct more data collection events. The presented data collection technique is robust and capable of producing data that is consistent and representative of a participants driving techniques. With more data a more generalized and accurate trained model could be produced.

VIII. Conclusion

This study presented a driving simulator data collection technique as well as an architecture for classifying distracted driving. The proposed architecture resulted in a trained network that outperforms a common time-series classifier. With more data for more users the proposed network has the potential to become more generalized and accurate. Refining the presented network and implementing it in production vehicles could save the lives of distracted drivers and innocent bystanders.

IX. Github

https://github.com/tarizzo/distracted_driving

References

[1] C. Mick, R. A. Young, V. Boccardi, G. Paolisso, J. M. Silver, S. G. Klauer, F. Guo, and B. G. Simons-Morton, "Distracted driving and crash risk," *The New England journal of medicine*, vol. 370, no. 16, pp. 1564–1566, 2014.

[2] S. García-Herrero, J. D. Febres, W. Boulagouas, J. M. Gutiérrez, and M. M. Saldaña, "Assessment of the influence of technology-based distracted driving on drivers' infractions and their subsequent impact on traffic accidents severity," *International journal of environmental research and public health*, vol. 18, no. 13, pp. 7155–, 2021.

[3] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE transactions on industrial electronics (1982)*, vol. 66, no. 6, pp. 4788–4797, 2019.

[4] O. B. Sezer and A. M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied soft computing*, vol. 70, pp. 525–538, 2018.

[5] K. Nakano and B. Chakraborty, "Real-time distraction detection from driving data based personal driving model using deep learning," *International journal of ITS research*, vol. 20, no. 1, pp. 238–251, 2022.

[6] K. Ben Ahmed, B. Goel, P. Bharti, S. Chellappan, and M. Bouhorma, "Leveraging smartphone sensors to detect distracted driving activities," *IEEE transactions on intelligent transportation systems*, vol. 20, no. 9, pp. 3303–3312, 2019.

[7] D. Tran, H. Manh Do, W. Sheng, H. Bai, and G. Chowdhary, "Real-time detection of distracted driving based on deep learning," *IET intelligent transport systems*, vol. 12, no. 10, pp. 1210–1219, 2018.

[8] P. Maul, M. Mueller, F. Enkler, E. Pigova, T. Fischer, and L. Stamatogiannakis, "Beamng.tech technical paper." [Online]. Available: https://beamng.tech/blog/2021-06-21-beamng-tech-whitepaper/bng_technical_paper.pdf

[9] [Online]. Available: https://www.amazon.com/Thrustmaster-Racing-PlayStation4-PlayStation3-playstation-2/dp/B014US02ZA