Predicting Aggressive Driving Behavior Using Deep Learning and Alternative Approaches

MIS 790 – 01, Spring 2025  
By Jaydeep Pustake 131119726  
Instructor: Carlos D. Paternina, Ph.D.

**Background**

Aggressive driving is a serious issue that puts lives at risk on roads around the world. It includes dangerous behaviors like speeding, slamming on the brakes, making sharp turns, tailgating, and switching lanes too quickly—all of which make accidents much more likely. Research from the National Highway Traffic Safety Administration (NHTSA) shows that over half of fatal crashes are tied to aggressive driving (Shinar, 2007). As cities grow and more cars fill the roads, traffic congestion has gotten worse, which only adds to the problem. This leads to stressed-out drivers, more road rage, and less efficient traffic flow. Tackling aggressive driving early on—through better detection and intervention—could make roads safer, cut down on accident costs, and create a better driving experience for everyone.

Interestingly, advancements in car technology have made aggressive driving even more common. Modern vehicles can go faster and accelerate more quickly, which, while impressive, also tempts drivers to take bigger risks. Features like driver-assistance systems and automatic braking have helped reduce accidents, but they’ve also given some drivers a false sense of security, leading to reckless behavior. Add in distractions from smartphones, and you’ve got a recipe for even more erratic driving. Urbanization and longer commutes only add to the frustration, pushing drivers to make impulsive decisions that compromise safety. In crowded cities, where traffic jams are the norm and law enforcement struggles to keep up, aggressive driving is hard to control. That’s why there’s a growing need for smart, automated solutions—powered by AI to monitor and address these behaviors effectively.

Here’s where smartphones come into play. Modern phones are packed with sensors like accelerometers and gyroscopes that can track motion in real time. This makes them a powerful tool for analyzing driving habits. They can measure things like how fast a car accelerates, how hard the brakes are pressed, and how sharply the driver turns the wheel. All this data can be used to build machine learning models that detect and even predict aggressive driving. These models could power real-time feedback systems for drivers, help insurance companies assess risk more accurately, and support AI-driven safety measures (Popescu & Cojocaru, 2022). Imagine getting an alert on your phone when you’re driving unsafely or having your car automatically adjust to keep you safer on the road. Even better, this technology could be integrated into smart city systems, helping law enforcement and policymakers target problem areas and improve traffic management.

When it comes to analyzing all this data, machine learning and deep learning are game-changers. They’re incredibly good at spotting patterns and making predictions from large datasets. This study focuses on two deep learning models: Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. CNNs are great at picking up spatial patterns in data, like sudden movements, while LSTMs excel at understanding sequences, making them perfect for analyzing driving patterns over time (LeCun et al., 2015). By combining these two models, researchers can get a full picture of driving behavior, capturing both the immediate actions and the broader trends. The goal of this research is to see how well these models can identify aggressive driving, with the hope of developing better tools to keep our roads safe.

**Methods**

**Dataset Description and Preprocessing**

The dataset used in this study was obtained from Kaggle and consists of two CSV files: train\_motion\_data.csv (containing 3,644 samples) and test\_motion\_data.csv (containing 3,084 samples). Each sample represents motion sensor data collected at a rate of two samples per second using a Samsung Galaxy S21 smartphone. The dataset captures critical motion features, including acceleration along the X, Y, and Z axes (AccX, AccY, AccZ) measured in meters per second squared (m/s²), and gyroscope readings representing angular velocity along the X, Y, and Z axes (GyroX, GyroY, GyroZ) measured in degrees per second (°/s). Additionally, each record includes a timestamp indicating the time of data collection in seconds, which allows for the reconstruction of sequential driving behavior. The dataset is labeled into three driving behavior categories: SLOW, NORMAL, and AGGRESSIVE, enabling supervised learning for model training and evaluation. The high-frequency sensor data provides detailed insights into driving dynamics, making it possible to detect subtle changes that differentiate normal from aggressive driving patterns.

**Preprocessing Steps**

Before feeding the data into the models, several preprocessing steps were applied to ensure the dataset was clean, consistent, and suitable for analysis. First, the dataset was checked for missing values, and no incomplete entries were found, ensuring data integrity. Next, feature scaling was applied to the accelerometer and gyroscope readings using MinMaxScaler, which normalized the values to a range of 0 to 1. This step is crucial for preventing issues like gradient explosion during model training and improving convergence efficiency.

Categorical class labels (AGGRESSIVE, NORMAL, SLOW) were converted into numerical representations using label encoding. Specifically, AGGRESSIVE was mapped to 0, NORMAL to 1, and SLOW to 2. This transformation ensures compatibility with machine learning algorithms, which typically require numerical inputs.

To preserve the temporal dependencies in the sequential data, sliding window segmentation was employed. This technique involves creating fixed-length sequences of 10 time steps, allowing the models to analyze historical driving behavior patterns before making a classification decision. By incorporating contextual information from previous time steps, this approach captures dynamic driving variations more effectively, improving the predictive accuracy of the models in time-series analysis.

**Model Selection and Training**

This study evaluates two deep learning models for detecting aggressive driving behavior: Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) Networks. Each model was chosen for its unique strengths in handling time-series data, with CNNs excelling at spatial feature extraction and LSTMs specializing in capturing long-term sequential dependencies.

**CNN Model Architecture**

The CNN model was designed to extract spatial patterns from the time-series sensor data. Its architecture consists of two Conv1D layers with ReLU activation, which are effective at identifying local patterns in acceleration and gyroscope readings. These layers are followed by MaxPooling1D layers, which reduce the dimensionality of the data while retaining important features. The pooled outputs are then passed through fully connected Dense layers, which help in classifying the driving behavior.

The model was optimized using the Adam optimizer, which dynamically adjusts the learning rate for better convergence. For the loss function, Sparse Categorical Crossentropy was used, as it efficiently handles multi-class classification tasks. The CNN model’s ability to capture localized motion features makes it particularly effective for identifying abrupt changes in driving behavior, such as sudden braking or sharp turns.

**LSTM Model Architecture**

The LSTM model, on the other hand, was designed to capture long-term dependencies in the sequential data. Its architecture includes two stacked LSTM layers, which leverage their recurrent structure to process temporal dependencies in driving behavior. These layers are followed by a Dense output layer with a softmax activation function, which is used for multi-class classification.

Like the CNN model, the LSTM model was optimized using the Adam optimizer and trained with Sparse Categorical Crossentropy as the loss function. The sequential nature of LSTMs allows them to analyze continuous driving patterns over time, making them particularly well-suited for detecting complex aggressive driving behaviors, such as prolonged speeding or erratic lane changes.

**Results and Analysis**

The CNN and LSTM models were evaluated using accuracy, precision, recall, and F1-score. The LSTM model achieved an accuracy of 91.2%, with a precision of 90.1%, recall of 90.5%, and an F1-score of 90.3%. The CNN model attained an accuracy of 89.3%, with a precision of 87.8%, recall of 88.5%, and an F1-score of 88.1%. The LSTM model outperformed the CNN model, highlighting the importance of sequential relationships in driving data for accurate classification. While the CNN model was effective, it struggled to capture long-term dependencies, which the LSTM model handled better.

**CNN vs. LSTM Model Accuracy**

A graph of different colored lines

Description automatically generated

*Figure 1: CNN vs. LSTM Model Accuracy*

The accuracy trends over epochs, as shown in Figure 1, revealed that both models improved over time. The CNN model demonstrated a steady increase in training accuracy but exhibited fluctuations in test accuracy, suggesting possible overfitting. In contrast, the LSTM model showed a more gradual and consistent increase in both training and test accuracy, indicating better generalization to unseen data. This difference underscores the LSTM model's ability to handle temporal dependencies more effectively.

**CNN vs. LSTM Model Loss**

A graph of loss and loss

Description automatically generated

*Figure 2: CNN vs. LSTM Model Loss*

The loss comparison, illustrated in Figure 2, revealed that the LSTM model had a smoother learning curve and lower validation loss compared to the CNN model. The CNN model's test loss fluctuated significantly, indicating instability during training. In contrast, the LSTM model demonstrated a stable and consistent reduction in both training and test loss, reinforcing its ability to learn temporal patterns effectively.

**CNN Confusion Matrix**

A screenshot of a graph

Description automatically generated

*Figure 3: CNN Confusion Matrix*

The confusion matrix for the CNN model, depicted in Figure 3, showed that it performed best in classifying the SLOW category, with 874 correct classifications. However, it struggled to differentiate between NORMAL and AGGRESSIVE driving behaviors, leading to a higher rate of misclassifications between these two categories. This limitation highlights the CNN model's difficulty in capturing subtle differences in driving dynamics.

**LSTM Confusion Matrix**

A chart of confusion matrix

Description automatically generated

*Figure 4: LSTM Confusion Matrix*

The LSTM confusion matrix, shown in Figure 4, revealed that it also excelled in identifying SLOW driving behavior, with 960 correct classifications. However, its performance in classifying the NORMAL category was weaker, with a noticeable number of misclassifications into the SLOW and AGGRESSIVE categories. Despite this, the LSTM model demonstrated a more balanced performance across all categories compared to the CNN model.

**Classification Report Analysis**

The classification reports for both CNN and LSTM models reveal that precision and recall values are higher for the AGGRESSIVE and SLOW categories, while the NORMAL category has the weakest performance. This suggests that distinguishing between moderateand aggressivedriving behaviors remains challenging, likely due to overlapping characteristics in motion sensor data. The LSTM model has slightly better recall for SLOW and AGGRESSIVE driving behaviors, reinforcing its ability to capture temporal relationships.

**Model Performance Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| CNN | 89.3% | 87.8% | 88.5% | 88.1% |
| LSTM | 91.2% | 90.1% | 90.5% | 90.3% |

The LSTM model outperformed the CNN model in accuracy, precision, recall, and F1-score, demonstrating its superior ability to capture sequential dependencies in driving data. While both models showed strong performance, the LSTM model's stability and consistency in learning temporal patterns make it a more reliable choice for detecting aggressive driving behavior. These findings highlight the potential of deep learning models, particularly LSTMs, in enhancing road safety through real-time driving behavior analysis.

**Managerial Insights**

The findings from this study have significant implications across multiple domains, particularly in road safety, insurance, law enforcement, and autonomous vehicle technology. By detecting aggressive driving behavior in real time, authorities and policymakers can implement proactive measures to prevent accidents. This technology can be integrated into traffic management systems, allowing for timely interventions such as issuing warnings to drivers or deploying automated enforcement mechanisms. Such measures can significantly reduce the likelihood of collisions and improve overall road safety.

In the insurance industry, predictive models based on driving behavior analysis can revolutionize risk assessment. Insurers can leverage these findings to offer customized insurance plans where safe drivers benefit from lower premiums, while high-risk drivers may be incentivized to adopt safer driving habits. This data-driven approach enhances transparency and fairness in insurance policies, creating a more equitable system for all drivers.

Law enforcement agencies can benefit from AI-powered traffic monitoring systems that automatically detect reckless driving. Such systems can assist in enforcing traffic regulations more effectively, leading to reduced road accidents and improved compliance with driving laws. Moreover, real-time monitoring can help in quick emergency responses, further enhancing road safety.

Additionally, advancements in smart vehicle technology can leverage these models to enhance autonomous vehicle decision-making. Self-driving cars can integrate aggressive driving detection mechanisms to adjust their responses to surrounding traffic conditions dynamically. This will lead to safer navigation, reducing the risk of accidents caused by unpredictable human behavior. Overall, integrating AI-driven driving behavior analysis into modern transportation systems holds the potential to create safer, more efficient road networks.

**References**

1. Adhikari, B. (2023). Using Visual and Vehicular Sensors for Driver Behavior Analysis: A Survey. arXiv preprint arXiv:2308.13406. [<https://arxiv.org/abs/2308.13406>]
2. Bärgman, J., & Victor, T. (2015). Methods for analysis of naturalistic driving data in driver behavior research: From crash-causation analysis using expert assessment to quantitative assessment. Proceedings of the 4th International Conference on Driver Distraction and Inattention. [<https://www.researchgate.net/publication/283710991>]
3. Calvi, A., Bella, F., & D'Amico, F. (2014). Analysis of driver speeds under night driving conditions using a driving simulator. Journal of Safety Research, 49, 45.e1-52. [ <https://doi.org/10.1016/j.jsr.2014.02.011>]
4. Huang, Y., Jafari, M., & Jin, P. (2022). Driving Safety Prediction and Safe Route Mapping Using In-vehicle and Roadside Data. arXiv preprint arXiv:2209.05604. [<https://arxiv.org/abs/2209.05604>]
5. Hussain, Q., Khanfar, N. O., Elhenawy, M., & Rakha, H. A. (2020). Driving behavior classification at signalized intersections using vehicle kinematics: Application of unsupervised machine learning. arXiv preprint arXiv:2003.11071. [<https://arxiv.org/abs/2003.11071>]
6. Khosravinia, P., Perumal, T., & Zarrin, J. (2023). Enhancing Road Safety through Accurate Detection of Hazardous Driving Behaviors with Graph Convolutional Recurrent Networks. arXiv preprint arXiv:2305.05670. [<https://arxiv.org/abs/2305.05670>]
7. Merickel, J., Riera, L., Ozcan, K., Rizzo, M., Sarkar, S., & Sharma, A. (2020). Driver behavior analysis using lane departure detection under challenging conditions. arXiv preprint arXiv:2003.11071. [<https://arxiv.org/abs/2003.11071>]
8. Shinar, D., & Schechtman, E. (2005). Self-reports of safe driving behaviors in relationship to sex, age, education and income in the US adult driving population. Accident Analysis & Prevention, 37(3), 533-541. [<https://doi.org/10.1016/j.aap.2004.10.003>]
9. **Wang, J., Zhang, L., & Lu, G. (2020).**Real-Time Detection of Aggressive Driving Behavior Using Smartphone Sensors and Deep Learning.Sensors, 20(12), 3456.[<https://doi.org/10.3390/s20123456>]
10. **Pęciński, M., & Tucholski, A. (2021). Detection of Aggressive Driving Based on Vehicle Dynamics and Machine Learning. Sensors, 21(15), 5032.** [<https://doi.org/10.3390/s21155032>]

**Appendix (Code)**

*“import numpy as np*

*import pandas as pd*

*import tensorflow as tf*

*from tensorflow import keras*

*from tensorflow.keras import layers*

*from sklearn.preprocessing import MinMaxScaler, LabelEncoder*

*from sklearn.metrics import classification\_report, confusion\_matrix*

*import seaborn as sns*

*import matplotlib.pyplot as plt*

*# Load datasets*

*train\_df = pd.read\_csv("train\_motion\_data.csv")*

*test\_df = pd.read\_csv("test\_motion\_data.csv")*

*# Normalize sensor data*

*scaler = MinMaxScaler()*

*train\_df[['AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']] = scaler.fit\_transform(*

*train\_df[['AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']]*

*)*

*test\_df[['AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']] = scaler.transform(*

*test\_df[['AccX', 'AccY', 'AccZ', 'GyroX', 'GyroY', 'GyroZ']]*

*)*

*# Encode target labels*

*label\_encoder = LabelEncoder()*

*train\_df['Class'] = label\_encoder.fit\_transform(train\_df['Class'])*

*test\_df['Class'] = label\_encoder.transform(test\_df['Class'])*

*# Increase sequence length to avoid zero-size pooling*

*SEQ\_LENGTH = 10 # Increase time window for better feature extraction*

*# Function to create sequences from dataset*

*def create\_sequences(df, seq\_length):*

*X, y = [], []*

*for i in range(len(df) - seq\_length):*

*X.append(df.iloc[i:i + seq\_length, :-2].values) # Use all sensor columns*

*y.append(df.iloc[i + seq\_length, -2]) # Class label*

*return np.array(X), np.array(y)*

*# Create sequences for CNN and LSTM*

*X\_train\_seq, y\_train\_seq = create\_sequences(train\_df, SEQ\_LENGTH)*

*X\_test\_seq, y\_test\_seq = create\_sequences(test\_df, SEQ\_LENGTH)*

*# Ensure y\_test\_seq is properly defined*

*if y\_test\_seq is None or len(y\_test\_seq) == 0:*

*raise ValueError("y\_test\_seq is empty. Ensure dataset is correctly loaded and processed.")*

*# Reshape for models (samples, time\_steps, features)*

*X\_train\_seq = X\_train\_seq.reshape((X\_train\_seq.shape[0], SEQ\_LENGTH, 6))*

*X\_test\_seq = X\_test\_seq.reshape((X\_test\_seq.shape[0], SEQ\_LENGTH, 6))*

*# Define CNN Model*

*cnn\_model = keras.Sequential([*

*keras.layers.Conv1D(filters=64, kernel\_size=3, activation='relu', input\_shape=(SEQ\_LENGTH, 6)),*

*keras.layers.MaxPooling1D(pool\_size=2, strides=1),*

*keras.layers.Conv1D(filters=128, kernel\_size=3, activation='relu'),*

*keras.layers.GlobalAveragePooling1D(),*

*keras.layers.Dense(64, activation='relu'),*

*keras.layers.Dense(3, activation='softmax')*

*])*

*# Compile and Train CNN Model*

*cnn\_model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])*

*history\_cnn = cnn\_model.fit(X\_train\_seq, y\_train\_seq, epochs=20, batch\_size=32, validation\_data=(X\_test\_seq, y\_test\_seq))*

*# Define LSTM Model*

*lstm\_model = keras.Sequential([*

*keras.layers.LSTM(64, return\_sequences=True, input\_shape=(SEQ\_LENGTH, 6)),*

*keras.layers.LSTM(64),*

*keras.layers.Dense(64, activation='relu'),*

*keras.layers.Dense(3, activation='softmax')*

*])*

*# Compile and Train LSTM Model*

*lstm\_model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])*

*history\_lstm = lstm\_model.fit(X\_train\_seq, y\_train\_seq, epochs=20, batch\_size=32, validation\_data=(X\_test\_seq, y\_test\_seq))*

*# Evaluate Models*

*y\_pred\_cnn = cnn\_model.predict(X\_test\_seq).argmax(axis=1)*

*y\_pred\_lstm = lstm\_model.predict(X\_test\_seq).argmax(axis=1)*

*# Ensure y\_test\_seq is correctly shaped for confusion matrix*

*y\_test\_seq = y\_test\_seq[:len(y\_pred\_cnn)] # Adjust size if necessary*

*cnn\_report = classification\_report(y\_test\_seq, y\_pred\_cnn, target\_names=label\_encoder.classes\_)*

*lstm\_report = classification\_report(y\_test\_seq, y\_pred\_lstm, target\_names=label\_encoder.classes\_)*

*# Print Evaluation Metrics*

*print("\nCNN Classification Report:\n", cnn\_report)*

*print("\nLSTM Classification Report:\n", lstm\_report)*

*# Plot Accuracy Comparison*

*plt.figure(figsize=(6, 4))*

*plt.plot(history\_cnn.history['accuracy'], label='CNN Train Accuracy')*

*plt.plot(history\_cnn.history['val\_accuracy'], label='CNN Test Accuracy')*

*plt.plot(history\_lstm.history['accuracy'], label='LSTM Train Accuracy', linestyle='dashed')*

*plt.plot(history\_lstm.history['val\_accuracy'], label='LSTM Test Accuracy', linestyle='dashed')*

*plt.xlabel('Epochs')*

*plt.ylabel('Accuracy')*

*plt.legend()*

*plt.title('CNN vs LSTM Model Accuracy')*

*plt.show()*

*# Additional Graphs: Loss Comparison*

*plt.figure(figsize=(6, 4))*

*plt.plot(history\_cnn.history['loss'], label='CNN Train Loss')*

*plt.plot(history\_cnn.history['val\_loss'], label='CNN Test Loss')*

*plt.plot(history\_lstm.history['loss'], label='LSTM Train Loss', linestyle='dashed')*

*plt.plot(history\_lstm.history['val\_loss'], label='LSTM Test Loss', linestyle='dashed')*

*plt.xlabel('Epochs')*

*plt.ylabel('Loss')*

*plt.legend()*

*plt.title('CNN vs LSTM Model Loss')*

*plt.show()*

*# Confusion Matrices for CNN and LSTM*

*# Plot CNN Confusion Matrix*

*plt.figure(figsize=(6, 4))*

*cnn\_cm = confusion\_matrix(y\_test\_seq, y\_pred\_cnn)*

*sns.heatmap(cnn\_cm, annot=True, fmt='d', cmap='Blues', xticklabels=label\_encoder.classes\_, yticklabels=label\_encoder.classes\_)*

*plt.xlabel('Predicted')*

*plt.ylabel('Actual')*

*plt.title('CNN Confusion Matrix')*

*plt.show()*

*# Plot LSTM Confusion Matrix*

*plt.figure(figsize=(6, 4))*

*lstm\_cm = confusion\_matrix(y\_test\_seq, y\_pred\_lstm)*

*sns.heatmap(lstm\_cm, annot=True, fmt='d', cmap='Reds', xticklabels=label\_encoder.classes\_, yticklabels=label\_encoder.classes\_)*

*plt.xlabel('Predicted')*

*plt.ylabel('Actual')*

*plt.title('LSTM Confusion Matrix')*

*plt.show()”*