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 major contributor to road accidents and fatalities worldwide. It includes high-risk behaviors such as excessive speeding, abrupt braking, sharp turns, tailgating, and rapid lane changes, which significantly increase the likelihood of collisions. According to the National Highway Traffic Safety Administration (NHTSA), aggressive driving is responsible for more than 50% of all fatal crashes (Shinar, 2007). With increasing urbanization and rising traffic congestion, instances of aggressive driving have become more frequent, leading to elevated stress levels, road rage, and reduced traffic efficiency. Effectively identifying and mitigating aggressive driving behavior can improve road safety, lower accident-related costs, and enhance the overall driving experience for road users.

The integration of sensor technology in modern smartphones has made it possible to collect real-time motion data to analyze driving behavior. Accelerometers and gyroscopes capture crucial driving dynamics such as acceleration, braking intensity, and steering patterns. These continuous data streams provide a valuable foundation for the development of machine learning models that can detect and predict aggressive driving behavior. Such models have the potential to power real-time driver feedback systems, adaptive insurance risk assessments, and AI-driven transportation safety improvements (Popescu & Cojocaru, 2022).

Machine learning and deep learning techniques have demonstrated exceptional performance in pattern recognition tasks. This study specifically explores two deep learning models—Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks—to analyze sensor data and classify driving behavior into three categories: SLOW, NORMAL, and AGGRESSIVE. CNNs excel at capturing spatial patterns within time-series data, making them effective for extracting localized motion features. In contrast, LSTMs are designed to process sequential dependencies, making them particularly well-suited for analyzing continuous driving patterns over time (LeCun et al., 2015).

**Methods**

**Dataset Description and Preprocessing**

The dataset utilized in this study is sourced 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 readings collected at a fixed rate of two samples per second using a Samsung Galaxy S21 smartphone. The dataset captures key motion features, including acceleration measurements along the X, Y, and Z axes (AccX, AccY, AccZ) in meters per second squared (m/s²) and gyroscope readings representing angular velocity along the X, Y, and Z axes (GyroX, GyroY, GyroZ) in degrees per second (°/s). Additionally, each record includes a timestamp representing the time of data collection in seconds, allowing for the reconstruction of sequential driving behavior. The dataset is labeled into three distinct driving behavior categories: SLOW, NORMAL, and AGGRESSIVE, enabling supervised learning for model training and evaluation. The inclusion of high-frequency sensor data facilitates the detection of subtle changes in driving dynamics, which are critical for differentiating between normal and aggressive driving patterns.

**Preprocessing Steps**

Preprocessing steps included data cleaning, where the dataset was checked for missing values, and no incomplete entries were found, ensuring data integrity and consistency. Feature scaling was applied to accelerometer and gyroscope readings using MinMaxScaler, transforming the values to a normalized range of 0 to 1, which aids in preventing gradient-related issues during model training and enhances convergence efficiency. Label encoding was performed to convert categorical class labels into numerical representations, mapping AGGRESSIVE to 0, NORMAL to 1, and SLOW to 2, ensuring compatibility with machine learning algorithms. To preserve temporal dependencies in the sequential data, sliding window segmentation was employed, where fixed-length sequences of 10 time steps were created, enabling the models to analyze historical driving behavior patterns before making a classification decision. This technique ensures that each instance incorporates contextual information, which is crucial for capturing dynamic driving variations and improving predictive accuracy 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. CNNs are specifically designed for spatial feature extraction in time-series sensor data, making them effective at identifying local patterns in acceleration and gyroscope readings. LSTMs, on the other hand, specialize in capturing long-term sequential dependencies, making them well-suited for analyzing time-series data where past driving behavior influences future predictions.

The CNN model architecture consists of two Conv1D layers with ReLU activation, followed by MaxPooling1D layers to reduce dimensionality and retain important features. These layers are followed by fully connected Dense layers, which help in classification. The model is optimized using the Adam optimizer, which adapts the learning rate dynamically for better convergence, and employs Sparse Categorical Crossentropy as the loss function to handle multi-class classification efficiently.

The LSTM model architecture comprises two stacked LSTM layers that 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 for multi-class classification. Similar to the CNN model, the LSTM model is optimized using Adam and trained with Sparse Categorical Crossentropy to ensure effective classification of driving behaviors. The sequential nature of LSTMs allows them to capture complex patterns in aggressive driving behavior, making them a strong candidate for real-time detection and prediction.

**Results and Analysis**

The models were evaluated based on 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 CNN, indicating that sequential relationships in driving data significantly enhance classification. While the CNN model was effective, it struggled to capture long-term dependencies compared to LSTM.

**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 showed that both models improved over time. The CNN model demonstrated a steady increase in training accuracy but exhibited fluctuations in test accuracy, indicating possible overfitting. In contrast, the LSTM model showed a more gradual but consistent increase in both training and test accuracy, suggesting better generalization on unseen data. *(Figure 1: CNN vs. LSTM Model Accuracy)*

**CNN vs. LSTM Model Loss**

A graph of loss and loss

Description automatically generated

*Figure 2: CNN vs. LSTM Model Loss*

The loss comparison revealed that the LSTM model had a smoother learning curve and lower validation loss compared to CNN. The CNN model's test loss fluctuated significantly, suggesting instability in model training. LSTM, on the other hand, demonstrated a stable and consistent reduction in both training and test loss, reinforcing its ability to learn temporal dependencies effectively. *(Figure 2: CNN vs. LSTM Model Loss)*

**CNN Confusion Matrix**

A screenshot of a graph

Description automatically generated

*Figure 3: CNN Confusion Matrix*

The confusion matrix for the CNN model indicates 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. *(Figure 3: CNN Confusion Matrix)*

**LSTM Confusion Matrix**

A chart of confusion matrix

Description automatically generated

*Figure 4: LSTM Confusion Matrix*

The LSTM confusion matrix shows that it also performed best in identifying SLOW driving behavior, with 960 correct classifications. However, its classification performance for the NORMAL category was weaker, with a noticeable number of misclassifications into the SLOW and AGGRESSIVE categories. Despite this, the LSTM model still demonstrated a more balanced performance across all categories compared to CNN. *(Figure 4: LSTM Confusion Matrix)*

**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% |

**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.

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.

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. Ma, Y., & Wang, Y. (2019). A comparative study of aggressive driving behavior recognition algorithms based on vehicle motion data. IEEE Access, 7, 32134-32144. [<https://doi.org/10.1109/ACCESS.2019.2903036>]
8. 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>]
9. Miyajima, C., Nishiwaki, Y., Ozawa, K., Wakita, T., Itou, K., Takeda, K., & Itakura, F. (2007). Driver modeling based on driving behavior and its evaluation in driver identification. Proceedings of the IEEE, 95(2), 427-437. [<https://doi.org/10.1109/JPROC.2006.888407>]
10. 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>]

**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()”*