Predicting Vehicle Collisions from Dashcam Video Using Neural Network Models

Adam Stein, Hung Tran, Sean Morris, and David Corcoran

Introduction

In this project, we addressed the challenge of real-time collision detection and classification from dashcam videos, a task with significant implications for road safety and post-incident analysis. Our goal was to develop deep learning models capable of classifying video segments as either involving a collision or near miss, or no risk of collision at all. To achieve this, we extracted bounding box features from each frame of all provided dashcam videos using a YOLO object detection model. These raw bounding box and class features were then used to calculate additional attributes, such as changes in position and relative size over time, to better capture motion patterns and object interactions as frames progress through each video. Using these enhanced time-series representations, we evaluated the effectiveness of three sequence modeling architectures: Gated Recurrent Units (GRU), Long Short-Term Memory networks (LSTM), and Transformers. Each model was trained to recognize temporal patterns indicative of potential or ongoing collisions, supporting applications in real-time driver assistance systems and automated incident detection.

Literature Review

Advancements in autonomous driving and intelligent safety systems have led to increased research in collision detection using computer vision and deep learning. Dashcam-based crash prediction is particularly relevant for real-time driver assistance, auto insurance evaluation, and autonomous navigation. This section reviews recent studies that integrate object detection and sequence modeling for collision analysis.

Nguyen et al. (2023) proposed a pipeline combining YOLOv5 and Mask R-CNN for object and lane detection in dashcam footage, tailored for auto insurance scenarios. Their hybrid approach improved detection accuracy in complex scenes through pre-training and fine-tuning on diverse driving conditions. Fernandes et al. (2018) developed a CNN-RNN system for road anomaly detection, emphasizing how sequential frame dependencies can enhance event recognition, laying groundwork for models like GRUs and LSTMs.

Saini et al. (2022) focused on fast, lightweight ML models for collision detection on embedded devices, achieving over 90% accuracy using sensor data and basic vision techniques. Though they avoided models like YOLO, their work underscores real-world constraints such as speed and resource limits.

Wee et al. (2022) presented a deep learning pipeline for forward collision warning, using YOLACT to balance detection accuracy and inference speed. Their system robustly handled challenging conditions like nighttime and rain, and their benchmark comparisons highlighted YOLACT as a practical compromise between performance and efficiency.

Across these works, a common theme emerges: combining robust object detection (e.g., YOLO, Mask R-CNN) with temporal or contextual modeling enhances collision prediction. Key challenges include occlusion handling, real-time processing, and robustness under diverse conditions. These insights directly shaped our approach, where YOLOv8 preprocessing feeds into sequential models for crash prediction.

Data & Feature Engineering

Dataset:

This study utilized 1,500 labeled dashcam videos sourced from the Nexar Dashcam Crash Prediction Challenge on Kaggle. Each video clip is roughly 40 seconds and categorized into one of two classes: collision/near miss or non-collision, with an equal distribution of 750 videos per class. The dataset reflects real-world driving conditions and includes a diverse range of environments and traffic scenarios, making it suitable for training temporal models for collision detection. The dataset was partitioned into training, validation, and testing sets using an 80/10/10 split. This ensures sufficient data for model training while reserving representative samples for hyperparameter tuning and final performance evaluation.

Preprocessing:

To convert raw video data into a format suitable for sequence modeling, we performed several preprocessing steps. First, frames were extracted at a rate of 1 frame per second using OpenCV, resulting in a relatively consistent 40 frames per video. Each frame was then passed through a pre-trained YOLOv8 object detection model to identify and localize entities of interest at every time step/frame. The YOLOv8 model is configured to detect around 80 object classes, including many that would appear in dashcam footage, like cars, buses, trucks, pedestrians, and road signs. For each video, a structured “detections.csv” file was generated, logging detected objects and their characteristics. The raw features appearing in each “detections.csv” file include frame number, bounding box size variables (x_min, y_min, x_max, y_max), class, and confidence score. From these raw detections, we constructed and appended a suite of per-frame feature vectors by first aggregating and normalizing all of the object-level data over each frame. For each frame, we grouped detections by class and computed their summary statistics, including the count of classes, their mean & variance, and the mean area and aspect ratio of each class’s bounding box. Next, we linked objects across consecutive frames by constructing an IoU-based tracker to estimate each object’s centroid’s displacements, which we use as a proxy for speed and acceleration of different objects over time. To encode short-term temporal context, we also append delta-features (i.e. the frame-to-frame changes in object coins and motion metrics). Finally, all features were standardized using the global mean and standard deviation computed across the training set. All sequences include zero-padding to ensure a uniform length across inputs before feeding into our models.

Final Input Format:

The final model input consists of a sequence of feature vectors derived from the object detection outputs for each frame. Each sequence has a fixed length of 40 time steps (one per second), with features aggregated or encoded per frame. Sequences are padded and packed to ensure compatibility with PyTorch’s LSTM and GRU modules for batch processing. Each feature vector includes both raw object-level attributes and engineered temporal features that capture inter-frame changes, like shifts in

position and relative size. To standardize the input across videos, we used StandardScaler from scikit-learn. This scaler was fit using all feature vectors extracted from the training set. For each video, features were computed frame by frame, and deltas were calculated for a subset of the features by comparing each frame with its preceding frame, while categorical or static features were excluded from delta computation. The final input representation for each frame combined the raw and delta features into a vector, which was then normalized using the fitted scaler before being passed to the models. The dimension of each final input vector varied, as feature importances were being experimented with. The input vectors contained 7, 16, and 28 features for the LSTM, GRU, and Transformer models, respectively.

Methods

We implemented three neural network architectures for binary classification: LSTM, GRU, and Transformer models. All three models are designed to capture temporal dependencies within video frame sequences by processing frame-level features sequentially. Each model takes a sequence of frame features as input, with dimensions corresponding to the number of frames (time steps) and feature size per frame.

Long Short-Term Memory:

LSTM networks are a type of recurrent neural network (RNN) designed to address the problem of vanishing gradients, which can occur when training standard RNNs over long sequences. LSTMs use gates (input, forget, and output) to control the flow of information, allowing the network to retain long-term dependencies and forget irrelevant past information. This makes LSTMs especially suitable for tasks where long-range temporal dependencies are important, such as detecting collisions in dashcam footage where the current frame might be influenced by the context of previous frames. In collision detection, the model might need to remember several frames of data to predict whether a collision occurs. Frame-by-frame movements or behaviors of objects, such as the change in position of a vehicle, are better captured using sequential models like LSTM. The LSTM model used the following features as inputs: average bounding box area, the change in average bounding box area, the minimum distance to the closest object to the dashcam car, the change in the minimum distance to the closest object in the frame, the area of the largest bounding box, the change in area of the largest bounding box, and the count of objects within a designated “danger zone”.

Gated Recurrent Unit:

GRUs are a simpler alternative to LSTMs, designed to achieve similar performance while having fewer parameters. Unlike LSTMs, which use three gates, GRUs rely on only two gates: the reset gate and the update gate. This simpler architecture makes GRUs computationally more efficient, often resulting in faster training times and lower memory usage. Despite their simplicity, GRUs can perform comparably to LSTMs in many sequence modeling tasks, especially when the sequences are of moderate length or when the task does not demand long-term memory. Additionally, GRUs are less prone to overfitting on smaller datasets due to their reduced complexity, making them an ideal choice in scenarios where interpretability and training speed are important.

To identify the optimal configuration for both the LSTM and GRU models, we performed a grid search over a defined hyperparameter space including hidden_size, num_layers, dropout, learning_rate, and

optimizer type. Each combination of hyperparameters was used to initialize and train an instance of the models. During training, early stopping based on validation loss was employed using a patience parameter to prevent overfitting. Both the LSTM and GRU models use Binary Cross Entropy (BCE) as the loss function because BCE encourages the model to output probabilities near 1 for positive examples and near 0 for negative ones. The training and validation losses were tracked for each configuration, and the model achieving the lowest final validation loss was selected as the best. This thorough approach helped identify the best-performing model configuration for our video classification task.

Transformer:

Transformer models, first introduced in 2017 by Vaswani et al, take a different approach to sequence modeling by replacing recurrence with self-attention mechanisms. Transformers excel in handling input sequence lengths of different sizes. They are also capable of parallelizing computation across multiple positions, and can extract rich contextual information without running into the vanishing-gradient issues that are common to RNNs. Transformers are notoriously difficult to train, so we knew that having an environment that could facilitate high-speed training was important. With this in mind, we decided to host our transformer on a LambdaLabs instance with a Nvidia A10 GPU to increase training efficiency and allow for quicker model iterations.

For the goal of detecting collisions through dashcam footage, we built out a classification pipeline using a transformer paired with a BCE loss function to capture the temporal components of our video frame inputs. The process begins by projecting each frame's raw feature vector into a fixed 128-dimensional embedding space via a linear layer. After this, we apply sinusoidal positional encodings into our feature vector to try and bring back the sequential structure that underlies our raw video data. A LayerNorm preprocessor then stabilizes the embeddings before they are fed into a two layer encoder layer - where each layer uses multi headed attention, and a 512-dimensional feed-forward network using a dropout rate of 0.1 to prevent overfitting. After the encoder, we apply another LayerNorm to prepare our sequence for global aggregation. In this step, we add mask padded positions, sum the amount of valid token embeddings, and divide the sequence lengths to create a single 128-dimensional summary vector per video. For the final step, this vector gets fed into a MLP. For this architecture, we prepare two hidden layers [256, 64], each paired with BatchNorm, ReLU activation, and 0.1 dropout. The output layer consists of two nodes, using sigmoidal activation to yield our final binary classification vector.

Results

Long Short-Term Memory:

As seen in Figure 1, the LSTM model showed moderate performance, achieving a classification accuracy of approximately 66%. This means that just under two-thirds of the dashcam videos were correctly classified. With a precision of about 70.6%, the model was reasonably accurate when predicting crashes, which is beneficial in minimizing false alarms in safety-critical applications. However, its recall was lower at 51.4%, indicating that nearly half of the actual crash videos went undetected. This shortfall is significant, as failing to recognize real crashes poses serious risks in real-world deployment. The F1 score of 59.5% reflects the imbalance between precision and recall, suggesting that while the model avoids over-predicting crashes, it struggles to consistently detect them. Despite these limitations, the LSTM

model demonstrates reasonable promise, but improving its sensitivity would be a necessary task for deployment in scenarios where missed detections can have significant consequences.

The training process involved monitoring both training and validation loss curves over multiple epochs to optimize the model. The best-performing model leveraged the following hyperparameters: a hidden size of 128, a dropout rate of 0.2, a learning rate of 0.001, three hidden layers, and the Adam optimizer.

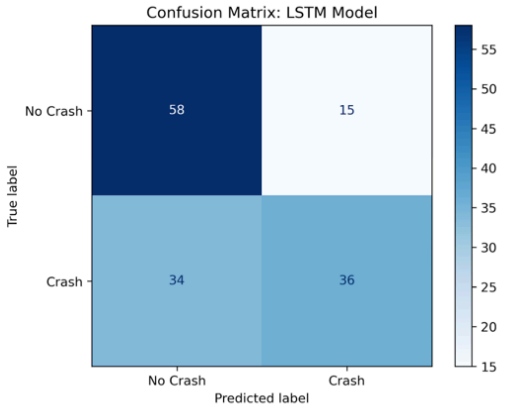


Figure 1: Confusion matrix for the LSTM binary classification model

Figure 2: Training and validation losses for best LSTM Model

Gated Recurrent Unit:

Our GRU model delivered the strongest performance among all models tested in this study. As seen in Figure 4 (confusion matrix), it delivered a moderate performance in the classification task. It achieved an accuracy of 68%, indicating that just over two-thirds of all video segments were correctly labeled. The model also produced a precision of 73%, suggesting that when it predicted a crash, it was correct nearly three-quarters of the time. This high precision is important in safety-critical systems, where false positives can trigger unnecessary interventions. However, the recall rate of 66% reveals a key limitation: the model failed to detect approximately one-third of actual crash cases, which could be critical in real-world deployment where missing a true collision has severe consequences. The F1 score of 69% reflects a reasonable trade-off between precision and recall, signaling moderate overall effectiveness.

Figure 3: Training & Validation Loss Curves for best GRU model

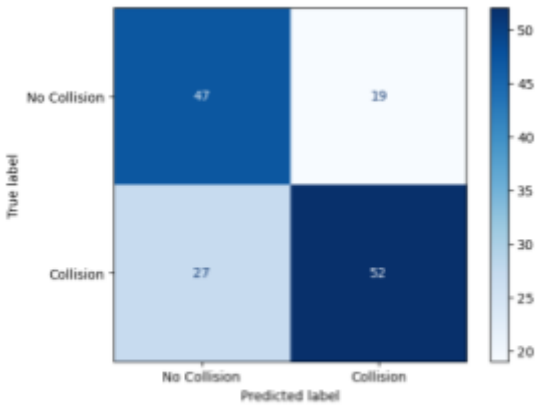


Figure 4: Confusion Matrix for GRU model

Compared to the Transformer and LSTM models, GRU's gated architecture may have been better suited for modeling the temporal dependencies in our sequential YOLO-extracted object data. Its performance indicates that while recurrent structures like GRU can capture meaningful dynamics in video sequences, there is still room to enhance recall. Overall, the GRU's results demonstrate promise for sequence-based collision prediction tasks, but they also highlight the importance of improving sensitivity to ensure critical crash events are not missed.

Transformer:

Our transformer model achieved somewhat disappointing results when applied to the crash-no-crash task. For the classification metrics, the model yielded an overall accuracy of 58%, a precision of 59%, a recall of 53%, and an F1 score of 56%. As seen in figure 5, the model was able to better capture instances of non-crash sequences than sequences containing crashes. We believe that this slight under-sensitivity to true crash events may have resulted from the inherent class imbalances in the dataset. As for the model's training dynamics, the whole process ran very smoothly: both the global average train and validation losses seen in figure 6 plummeted in the first two epochs and began to converge to a BCE value of around 0.6931, with almost no gap between them. While training dynamics were healthy, our convergence value corresponds to a binary classification accuracy of around 50% - meaning our model could not improve beyond a random guess. This suggests that despite stable optimization, our current feature representation and model architecture lack the predictive power needed to reliably detect crash events in dashcam video.

Figure 5: Confusion Matrix for Transformer Model

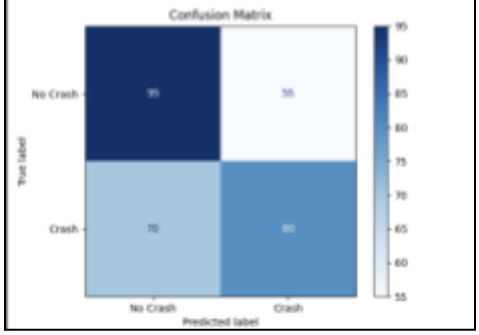
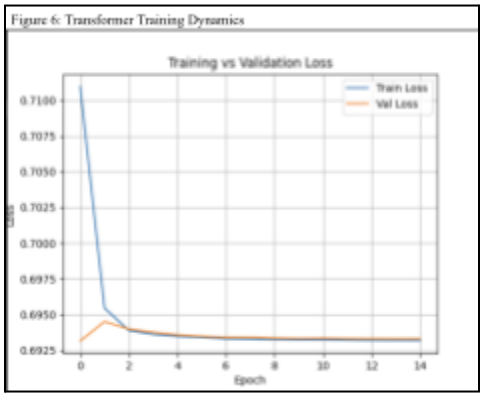


Figure 6: Transformer Training Dynamics



Conclusion & Future Work

In this study, we explored the use of sequential deep learning models, specifically LSTM and GRU networks, to classify driving dashcam footage as either containing a vehicle collision or not. By leveraging temporal patterns across video frames, our models partially captured motion cues and abrupt transitions often associated with collision events. The results demonstrate that sequence-based neural networks are somewhat capable of learning temporal dynamics from raw video data and temporal object features, providing a promising baseline for video-based automotive incident detection. This research contributes to the growing field of AI-assisted driver safety systems by demonstrating that temporal models can meaningfully interpret dashcam footage to detect critical events such as collisions. Real-time classification systems can play pivotal roles in post-accident analysis, autonomous vehicle decision-making, and insurance claim verification.

Limitations:

As reflected in our mostly moderate results, this study encountered several key challenges. One major limitation was visual obstruction in the dashcam footage caused by close-up objects, motion blur, poor camera angles, or climate-related distortions. These issues significantly impaired the performance of the pretrained YOLO object detection model, which ultimately reduced the quality of input data for the sequential neural networks. In some cases, even when there were no physical obstructions, the YOLO model misclassified objects, further degrading the input data quality. Consequently, the system occasionally failed to detect critical objects involved in actual collisions or mistakenly flagged non-threatening scenarios as crashes. Additionally, the majority of the collision videos featured only minor fender benders, meaning the visual cues for collision can be extremely subtle. This made it especially difficult for the model to distinguish low-impact incidents from regular traffic behaviors, such as a vehicle coming to a stop at a red light.

Future Directions:

To address these limitations and push this work toward real-world deployment, several future directions are proposed:

- **3D Motion Modeling:** Integrating optical flow analysis, or estimating object velocities through pixel movement between frames, could provide better motion-based context beyond just features extracted from raw frames. This would hypothetically allow the model to better understand sudden movements and forceful impacts, both characteristic of collisions.

- **Multi-Modal Sensors:** Extending the model to incorporate additional sensor inputs, such as GPS, LiDAR, accelerometers, gyroscopes, or vehicle telemetry, could offer complementary information, improving classification under conditions considered ambiguous with just dashcam footage.

References

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., & Urtasun, R. (2018). Multi-task learning for dangerous object detection in autonomous driving. *Information Sciences, 432–433*, 559–571. https://doi.org/10.1016/j.ins.2017.12.061

Hsu, P.-H., Huang, S.-L., & Han, C.-C. (2020). Collision analysis to motor dashcam videos with YOLO and Mask R-CNN for auto insurance. *International Journal of Computers and Applications*. https://doi.org/10.1080/1206212X.2020.1763237

Moura, Daniel C., Shizhan Zhu, and Orly Zvitia. Nexar Dashcam Crash Prediction Challenge. https://kaggle.com/competitions/nexar-collision-prediction, 2025. Kaggle.

"Video Classification with a 3D Convolutional Neural Network." *Geeksforgeeks*, Sanchhaya Education Private Limited, 1 July 2024, www.geeksforgeeks.org/video-classification-with-a-3d-convolutional-neural-network/.

Saini, R., Rani, R., & Gupta, S. (2022). Collision Detection and Prevention for Automobiles Using Machine Learning. *International Journal of Advanced Computer Science and Applications, 13*(7), 609–616. https://doi.org/10.14569/IJACSA.2022.0130769

Wee, W. G., Woo, J., Lee, D., Jung, J., & Park, K. (2022). Visual-based forward collision warning system using object detection and lane detection. *Sensors, 22*(11), 4095. https://doi.org/10.3390/s22114095

Appendix

Figures A1-A3 below contained dashcam video frames that contributed to the limitations of the YOLOv8 pretrained object detection model. The frames in Figures A1 and A2 cause the YOLOv8 model to miss objects that should have been classified, while the frames in Figure A3 contain objects that the YOLOv8 model misclassified completely. These missed detections and misclassifications reduced the quality of input data for the sequential neural networks, likely causing a decreased performance.

Figure A1: Dashcam video frames with poor camera angles, leading to zero identified YOLO objects.

Figure A2: Dashcam video frames that are affected by weather conditions, resulting in zero identified YOLO objects.

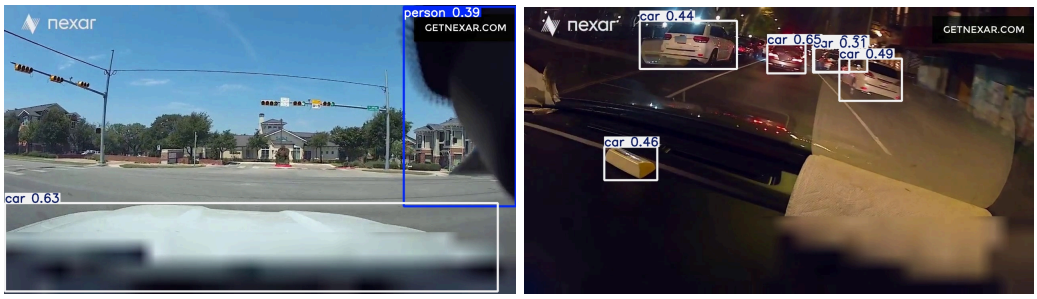


Figure A3: Dashcam video frames that misclassify nearby objects.

Figure A4 depicts the best LSTM model's ROC curve, which illustrates the trade-off between the true positive rate and the false positive rate. The model achieved an area under the curve (AUC) of 0.6916, which indicates moderately better performance than random guessing and reinforces the earlier assessment of the model's reasonable predictive ability.

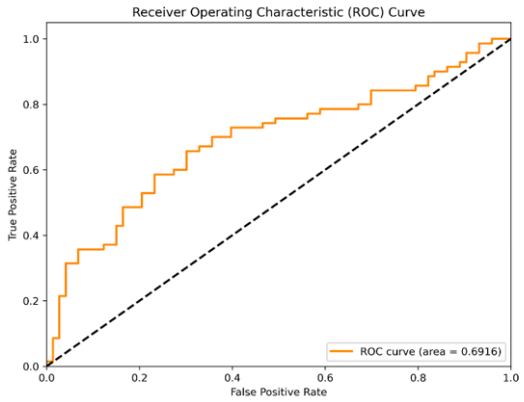


Figure A4: ROC curve for the LSTM binary classification model

Performance Metric	LSTM	GRU	Transformer
Accuracy	66%	68%	58%
Precision	71%	73%	59%
Recall	51%	66%	53%
F1	60%	69%	56%

Figure A5: Summary table of performance metrics for LSTM, GRU, and Transformer models

Hyperparameter	LSTM	GRU	Transformer
Hidden layers	3	2	2 (Encoders)
Layer size	128	128	128
Dropout	0.2	0.2	0.1
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001

Figure A6: Hyperparameter tuning results for each model