AI-Based Car Crash Detection Using Deep Learning

Shiyi Chen, Yuanchen Yin, Yuchen Li

April 6, 2025

Project Repository: `https://github.com/Nikki001021/DS542_Final_Project`

Abstract

Car crash detection plays a crucial role in intelligent traffic surveillance systems by enabling instance response and accident analysis. This project presents a deep learning pipeline that detects car crashes in video sequences using a hybrid `ResNet-18` and `BiLSTM` architecture. Instead of relying on object detection or per-frame classification, we analyze entire video clips to model spatial and temporal dynamics. We train our model using the DoTA dataset for crash scenarios and BDD100K for normal driving, combining them to provide a balanced training set. For testing, we introduce a third dataset: Cogito Car Crash Dataset. Model performance is evaluated using precision, recall and F1-score, with threshold tuning applied for optimal classification.

Introduction

Car crashes are a major cause of fatalities and congestion on roads, requiring rapid detection and response. Traditional traffic monitoring relies on manual observation, which is inefficient and prone to delays. AI-powered car crash detection can significantly improve response times by automatically identifying crash events in real time. This project explores an AI-based crash detection model that can process dashcam video to automatically classify driving events as crash or non-crashes.

We adopt a temporal classification approach using a hybrid `ResNet-18` and `BiLSTM` model. Each frame is first processed by a pretrained `ResNet-18` to extract spatial features. These per-frame features are then passed to a Bidirectional LSTM (`BiLSTM`), which captures both past and future temporal dependencies. This is particularly well-suited for our task, where video-level binary classification benefits from understanding the full sequence context, not just forward transitions.

Our training data combines crash videos from DoTA and normal driving clips from BDD100K, with consistent preprocessing for both datasets. For evaluation, we incorporate a new dataset–Cogito Car Crash Dataset–as our held-out test set, enabling rigorous assessment of model generalizability.

Related Work

Recent advances in deep learning have significantly improved traffic anomaly detection, particularly in crash prediction and event anticipation. Prior work mainly focus on these areas: datasets for traffic anomalies, object detection vs. temporal modeling approaches, and motion-based crash detection. Several studies have explored the use of deep learning for traffic anomaly detection.

Xu et al. [7] introduced the DoTA dataset, which includes annotated dashcam footage of traffic anomalies. Their work highlighted the importance of temporal localization in crash anticipation, as anomalies often depend on contextual motion patterns rather than single-frame features. Coursey et al. [2] released the FT-AED dataset to demonstrate the value of diverse data sources in training anomaly detection systems. Sultani et al. Their findings support our decision to combine multiple datasets (DoTA, BDD100K, and Cogito) to improve generalization. [8] Sultani et al. proposed a weakly supervised anomaly detection framework for surveillance videos using 3D CNNs.

Existing crash detection systems often prioritize real-time object localization over temporal dynamics. Rosales et al. [4] applied YOLOv8 for crash detection and reported strong results using object detection-based methods. However, our approach builds upon the idea that temporal sequence modeling—specifically, `ResNet + BiLSTM` architectures identify crash dynamics over time can more effectively.

Optical flow and motion features have long been used to identify collisions. Michalke et al. [3] further demonstrated the value of motion analysis in detecting side-collisions using optical flow, which motivates the temporal modeling aspect of our system.

Datasets

- **Detection of Traffic Anomaly (DoTA):** Includes 4,677 videos with annotated anomalies. We will only focus on car-to-car crash-related footage.
- **Normal Driving Footages (BDD100K):** Provides normal driving videos for comparison to derive insights for binary classification.
- **Cogito Car Crash Dataset (CCD):** Includes 1500 car crash videos and 3000 normal driving videos, used for an independent test set to evaluate model generalizability.

Methodology

Dataset Preparation

1.1 DoTA Dataset: The DoTA dataset contains video frames and associated label files. It also provides scripts to convert videos into frames and extract anomaly frames. Since we are only interested in crashes between two cars, we filter out anomaly classes such as pedestrian and obstacle-related labels.

Table 2: Traffic anomaly categories in the DoTA dataset

ID	Short	Anomaly Categories
1	ST	Collision with another vehicle which starts, stops, or is stationary
2	AH	Collision with another vehicle moving ahead or waiting
3	LA	Collision with another vehicle moving laterally in the same direction
4	OC	Collision with another oncoming vehicle
5	TC	Collision with another vehicle which turns into or crosses a road
6	VP	Collision between vehicle and pedestrian
7	VO	Collision with an obstacle in the roadway
8	OO	Out-of-control and leaving the roadway to the left or right
9	UK	Unknown

Figure 1: Types of anomaly in DoTA dataset

Furthermore, DoTA annotations include labels for both `ego:<type>` and `other:<type>`, indicating whether the dashcam vehicle was involved or witnessed the crash. Since our task is binary classification, we merge these into generalized types (e.g., `lateral`).

1.2 BDD100K Dataset: Using an adapted `video2frame.py` script, we extract 10 fps frames and assign each video the following label format:

```
["video_id"]: {
    "video_start": 0,
    "video_end": num_frames - 1,
    "anomaly_start": None,
    "anomaly_end": None,
    "anomaly_class": "normal",
    "num_frames": int,
    "subset": "train" or "val"
}
```

Model Framework

2.1 Model Description

Our pipeline is designed as a sequence-level binary classification model to detect whether a given driving video contains a car crash. The process involves four core stages:

1. **Image Preprocessing:** Each frame is resized to 224×224 and undergoes **aggressive augmentation**, including:
    - random cropping (`RandomResizedCrop`),
    - horizontal flipping (`RandomHorizontalFlip`),
    - color jittering (`ColorJitter`),
    - perspective distortion (`RandomPerspective`),
    - Gaussian blur (`GaussianBlur`),
    - automated policy-based augmentation (`AutoAugment`),
    - and random erasing (`RandomErasing`).

    This strengthens generalization and prepares the data for robust training.
2. **Frame-wise Feature Extraction (`ResNet-18`):** A pretrained `ResNet-18` is used to extract spatial features from each frame. We remove the final fully connected layer to obtain a **512-dimensional** feature vector per frame.
3. **Temporal Modeling (`BiLSTM`):** The sequence of frame-level features is passed into a **Bidirectional Long Short-Term Memory** network. Unlike standard LSTM, `BiLSTM` processes the video in both forward and backward directions, allowing it to capture full temporal context—critical for video-level binary classification.
4. **Classification:** The final hidden state from the `BiLSTM` is passed through a fully connected layer followed by a sigmoid activation to produce a crash probability score.

2.2 Why We Choose ResNet + BiLSTM over YOLO

In the early phase of our project, we considered using **YOLOv8**, a state-of-the-art object detection model, for car crash detection. YOLO is extremely effective for **real-time object localization** – identifying and drawing bounding boxes around cars, pedestrians, and other objects in individual frames. However, as we refined our project goals, we realized that our task is fundamentally different from object detection.
Our objective is to perform **binary classification** at the video clip level – determine whether a given dashcam video segment contains a car-to-car collision or not. This requires modeling not only what objects are present in individual frames, but also how those objects interact over time, such as:

- A sudden deceleration of a vehicle,
- Two cars approaching and abruptly stopping,
- A visual blur or distortion suggesting a crash event.

YOLO alone operates frame-by-frame and does not natively capture these temporal patterns. Therefore, we transitioned to a hybrid architecture that combines

- `ResNet18` for **spatial feature** extraction from individual frames, and
- `BiLSTM` for capturing **motion dynamics** and **temporal context** across the frame sequence.

This `ResNet + BiLSTM` approach is significantly more suited to our binary classification task. It enables the model to learn patterns of change over time, which are crucial to distinguishing crash events from normal driving behavior.

2.3 Training

Our model was trained using the following setup:

- **Loss Function:** Binary Cross-Entropy (`nn.BCELoss`) for binary classification.
- **Optimizer:** Adam with a learning rate of $1e-4$ to balance the convergence speed and stability.
- **Batch Size**: 8 (Due to GPU memory constraints).
- **Framework:** All training metrics were logged using Weights & Biases (`wandb`) for real-time visualization and monitoring.

We initially trained the model for 30 epochs. The best performance was observed at Epoch 29, with:

- **Train F1:** 0.9904, **Train Accuracy:** 98.80%
- **Validation F1:** 0.9839, **Validation Accuracy:** 97.97%
- **Validation Loss:** 0.0608

Although this performance was strong, we extended the training to 35 epochs to explore further improvements. Based on our observations, we shifted our early stopping criteria from validation loss to validation F1 score, as it better aligned with our binary classification goal and helped mitigate overfitting. This ensured that we captured the best balance between precision and recall during training. Here are the training results we got from `wandb`: From the extended 35-epoch run, the best-performing model was saved at **epoch 30**, where we achieved:

- **Train F1:** 0.9882, **Train Accuracy:** 98.52%
- **Validation F1:** 0.9898, **Validation Accuracy:** 98.71%
- **Validation Loss:** 0.0501

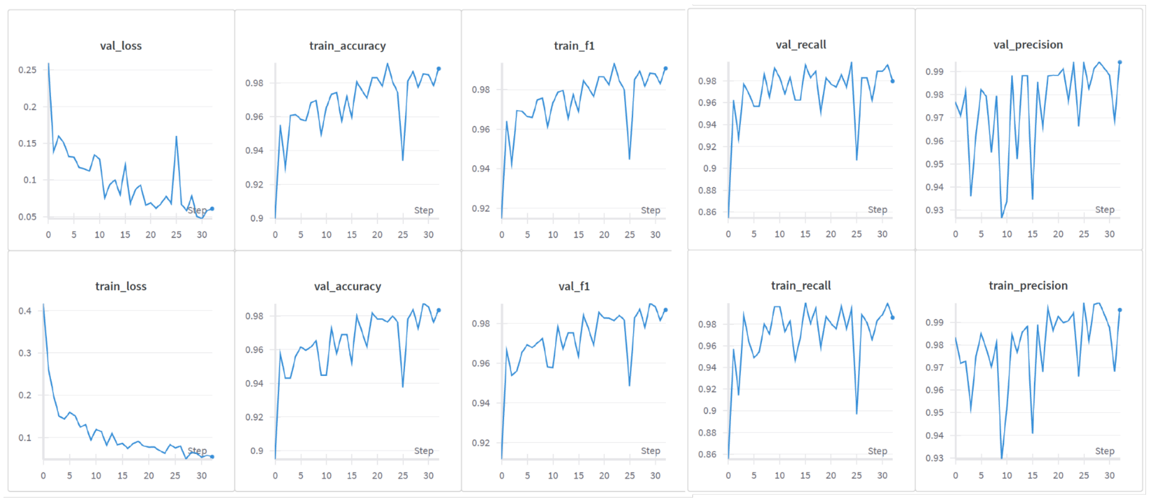

Figure 2: Training Results

In addition, the training and validation curves follow a similar trend in accuracy, F1 Score, recall, and precision, indicating strong generalization with no signs of overfitting or underfitting.

Evaluation Result

For model evaluation, we conduct the following steps:

1. **Test Set Preprocessing:** The third-party Cogito dataset (1500 crash + 3000 normal videos) was used as the test set. Videos were first converted to image frames at 10 fps, and then processed using the same augmentation and frame sampling strategy used during training.
2. **Model Inference:** The best saved model (`best_model.pth`) was loaded to make predictions over the entire test set.
3. **Threshold Tuning:** Initially, the model achieved the following metrics:
    - **Accuracy:** 77.33%
    - **Precision:** 0.9800
    - **Recall:** 0.3267
    - **F1-Score:** 0.4900

   While the precision was remarkably high, the recall was significantly low, indicating the model was highly confident when predicting a crash but often failed to detect many actual crashes. This imbalance is concerning in real-world applications, where missing a crash is far more dangerous than raising a false alarm.

Upon further analysis, we attributed this issue to **class imbalance**—especially in the test set (1500 crash vs. 3000 normal), and to a lesser extent in training (1706 crash vs. 1000 normal). To address this, we implemented **threshold tuning**, adjusting the decision boundary to find the best trade-off between precision and recall. As shown in *Table 1*, the best threshold of **0.1** yielded a significantly improved F1-score of **0.795**, though the underlying imbalance challenge remains.

Table 1: Threshold Tuning Results on Test Set

Threshold	**Accuracy**	**Precision**	**Recall**	**F1-Score**
0.10	0.8829	0.9542	0.6813	0.7950
0.15	0.8627	0.9594	0.6140	0.7488
0.20	0.8458	0.9643	0.5580	0.7069
0.25	0.8302	0.9670	0.5080	0.6661
0.30	0.8178	0.9709	0.4673	0.6310
0.35	0.8071	0.9773	0.4313	0.5985
0.40	0.7964	0.9787	0.3980	0.5659
0.45	0.7856	0.9785	0.3647	0.5313
0.50	0.7764	0.9805	0.3360	0.5005
0.55	0.7660	0.9806	0.3040	0.4641
0.60	0.7549	0.9806	0.2700	0.4234
0.65	0.7467	0.9865	0.2433	0.3904
0.70	0.7376	0.9848	0.2160	0.3543
0.75	0.7282	0.9860	0.1873	0.3148
0.80	0.7191	0.9836	0.1600	0.2752
0.85	0.7096	0.9949	0.1293	0.2289
0.90	0.6971	1.0000	0.0913	0.1674

Best Threshold: 0.1 **Best F1-Score:** 0.7950

Future Improvement

To mitigate the low recall and further enhance real-world robustness, we propose the following next steps:

Short-Term:

- **Integrate Optical Flow:** Incorporate motion cues to better distinguish crash dynamics from normal driving.

- **Accident Type Classification:** Move beyond binary classification to categorize types of crashes (e.g., rear-end, lateral, pedestrian).
- **Experiment with Vision Transformers (ViT):** Explore ViTs as spatial encoders to potentially outperform `ResNet-18` in complex visual scenes.

Long Term:

- **Edge Deployment:** Optimize and deploy the model on NVIDIA Jetson for real-time, on-device crash detection.
- **Multi-tasking Learning:** Extend the system to not only detect crashes but also localize the collision in the frame and estimate severity.

Conclusion

This project developed a deep learning system for detecting car crashes from dashcam video using a `ResNet-18 + BiLSTM` architecture. We trained the model on a combined dataset of annotated crash clips from **DoTA** and normal driving videos from **BDD100K**, and evaluated it on the **Cogito Car Crash Dataset**.

While the model performed well on the training and validation sets (`Val_F1 = 0.9839`), initial test results revealed high precision but low recall due to class imbalance. We addressed this issue through **threshold tuning**, improving the test F1-score from **0.49** to **0.795**.

Overall, our model effectively captures spatiotemporal crash patterns. Future improvements include incorporating motion features (e.g., optical flow), exploring Vision Transformers (ViT), and deploying the model to edge devices for real-world traffic safety applications.

References

[1] Bao, W., Yu, Q., & Kong, Y. (2020). *Uncertainty-Based Traffic Accident Anticipation with Spatio-Temporal Relational Learning.* In Proceedings of the ACM Multimedia Conference. `https://doi.org/10.1145/3394171.3413694`

[2] Coursey, A. (2023). *FT-AED: Freeway Traffic Anomaly Detection Dataset.* Retrieved from `https://acoursey3.github.io/ft-aed/`

[3] Michalke, T. P., Baltzer, M., & Maurer, M. (2011). *Optical Flow-Based Detection of Vehicle Side Collisions.* In Proceedings of IEEE Intelligent Vehicles Symposium. `https://doi.org/10.1109/IVS.2011.5940446`

[4] Rosales, J. A., et al. (2023). *Motor Vehicle Crash Detection Using YOLOv8 Algorithm.* In IEEE Intelligent Vehicles Symposium. `https://doi.org/10.1109/IVS.2023.10420786`

[5] Singh, H., Hand, E. M., & Alexis, K. (2020). *Anomalous Motion Detection on Highways Using Deep Learning.* arXiv preprint, arXiv:2006.08143. `https://arxiv.org/abs/2006.08143`

[6] Sultani, W., Chen, C., & Shah, M. (2018). *Real-world Anomaly Detection in Surveillance Videos.* In Proceedings of CVPR, 6479–6488. `https://doi.org/10.1109/CVPR.2018.00679`

[7] Xu, M., et al. (2020). *When, Where, and What? A New Dataset for Anomaly Detection in Driving Videos.* arXiv preprint, arXiv:2004.03044. `https://arxiv.org/abs/2004.03044`

[8] Sultani, W., Chen, C., Shah, M. (2018). *R*eal-world Anomaly Detection in Surveillance Videos. CVPR.`https://ieeexplore.ieee.org/document/8578776`