Traffic Accident Detection and Localization Using Computer Vision

Team 574/692 - Computer Vision
Gagandeep Kaur, Channing Tan and Viwesh Gupta

***Abstract*—Traffic accident detection from video is a challenging computer vision problem because accidents are short, rare, and motion-dependent events. A single image frame may show vehicles, roads, or traffic conditions, but it may not provide enough information to decide whether a crash has actually occurred. For this reason, accident detection requires both spatial understanding and temporal understanding. Spatial understanding helps the model recognize vehicles, road scenes, and crash-related visual cues, while temporal understanding helps the model learn how objects move and interact across time.**

In this project, we developed a multi-stage pipeline for traffic accident detection and localization. The goal of the system is not only to classify the accident type, but also to identify when the accident occurs and where the crash region is located in the frame. We first implemented a simple 2D Convolutional Neural Network (CNN) as a frame-level baseline. Then, we developed a 3D CNN model that processes 16-frame video clips to capture appearance and motion together. Finally, we added a temporal and spatial localization stage using YOLO-based vehicle detection and bounding-box dynamics.

The final system shows a clear progression from simple classification to accident understanding. The 2D CNN provides a useful baseline but is limited because it does not directly model motion. The 3D CNN improves the system by learning temporal crash patterns. The localization stage makes the output more actionable by estimating the accident time and crash position. Overall, this project demonstrates that temporal context and interpretability are essential for building useful video-based accident detection systems.

I. Introduction

Traffic accidents are a serious safety issue and can cause injuries, fatalities, traffic delays, and economic loss. In real traffic monitoring systems, fast accident detection can help emergency services respond sooner and can help traffic management systems reduce secondary collisions or congestion. Because many roads and intersections already use cameras, video-based accident detection is an important application of computer vision.

However, traffic accident detection is more difficult than normal image classification. In many image classification tasks, a single image is enough to identify the object or class. For example, a model can classify a car, a dog, or a traffic sign from one image. Accident detection is different because an accident is not just an object; it is an event. An event happens over time and depends on motion. Two vehicles may look close to each other in one frame, but without looking at the frames before and after, it is hard to know whether they are simply driving near each other, stopping at traffic, or actually colliding.

This creates an important challenge: the model must understand both *what* is in the scene and *how* the scene changes over time. The "what" part is spatial information, such as vehicle shape, road layout, and crash appearance. The "how" part is temporal information, such as vehicle movement, sudden stops, direction changes, and impact. A model that only uses spatial information may miss the motion cues that define the crash.

The objective of this project is to build a pipeline that answers three questions:

- What type of accident happened?
- When did the accident happen?
- Where did the accident happen in the frame?

To answer these questions, our project follows a staged design. The first stage uses a simple 2D CNN to classify accident frames. This gives a baseline and helps us understand what can be learned from a single frame. The second stage uses a 3D CNN to process short video clips. This allows the model to learn motion and temporal patterns. The final stage uses object detection and bounding-box analysis to estimate accident timing and location.

This step-by-step design is useful because each stage adds a capability that the previous stage does not have. The simple CNN adds basic spatial recognition. The 3D CNN adds motion awareness. The localization stage adds interpretability by giving the time and position of the accident.

II. Technical Approach

The technical approach is designed as a complete video understanding pipeline. Instead of treating accident detection as only a classification problem, we treat it as a combination of classification, temporal localization, and spatial localization.

A. Overall Pipeline

The full pipeline consists of the following stages:

1) Load traffic accident videos and annotation files.
2) Extract and preprocess video frames.
3) Train or run a 2D CNN baseline on individual frames.
4) Train or run a 3D CNN on 16-frame clips.
5) Detect vehicles using YOLO.
6) Analyze bounding-box changes over time.
7) Estimate accident time and accident location.

This pipeline is important because accident detection requires more than one type of information. Frame classification gives a basic prediction, clip classification captures motion, and localization gives an interpretable output.

B. Stage 1: Simple CNN for Frame-Level Classification

The first model is a simple 2D CNN. The input to this model is one RGB frame resized to 224×224. The frame has three channels: red, green, and blue. The purpose of this model is to learn spatial crash cues from individual frames.

The CNN architecture uses multiple convolution blocks. The number of filters increases from 32 to 64, then 128, and finally 256. This structure allows the model to learn simple features in the early layers and more complex features in the deeper layers. Early layers may learn edges, corners, and textures. Deeper layers may learn vehicle shapes, road context, damage patterns, or scene-level accident cues.

Batch normalization is used to stabilize training. Max pooling is applied after convolution blocks to reduce the feature map size and keep important information. After the convolution blocks, the feature maps are flattened and passed through a fully connected layer with 1024 units before the final class logits are produced.

The main strength of this model is simplicity. It is fast, easy to implement, and useful as a baseline. However, the main limitation is that it predicts frame by frame. Since it does not directly use previous or future frames, it cannot fully understand motion. This means that the 2D CNN may detect visible crash cues, but it may miss accidents where the important evidence is movement rather than appearance.

C. Simple CNN Inference Pipeline

The inference pipeline for the simple CNN is designed to apply the trained model to video frames. First, a video is opened using OpenCV. Each frame is resized to 224×224. The frame is normalized and converted from height-width-channel format to channel-height-width format so that it can be used by the neural network. The trained model then produces softmax probabilities, and the predicted label is overlaid on the video frame.

This approach is useful for visualizing model predictions, but it still has an important weakness. Since each frame is processed independently, the model may produce unstable predictions across time. For example, one frame may be predicted as accident and the next frame may not, even though they belong to the same event. This motivates the need for a temporal model.

D. Stage 2: 3D CNN for Clip-Level Accident Classification

The second model is a 3D CNN. Instead of using a single frame, this model uses a sequence of 16 frames as input. The input size is therefore represented as 16 frames, each with height, width, and three color channels.

The 3D CNN is more suitable for accident detection because it can learn both appearance and motion. A 2D convolution only moves across height and width, but a 3D convolution also moves across time. This means the model can learn how visual patterns change from one frame to the next.

Our 3D CNN uses Conv2Plus1D blocks. This method separates spatial learning and temporal learning. First, spatial convolution learns features within each frame. Then, temporal convolution learns how those features change over time. This is useful because accident videos contain both visual information and motion information. Separating these two types of learning can make the model easier to train compared to using full 3D convolution directly.

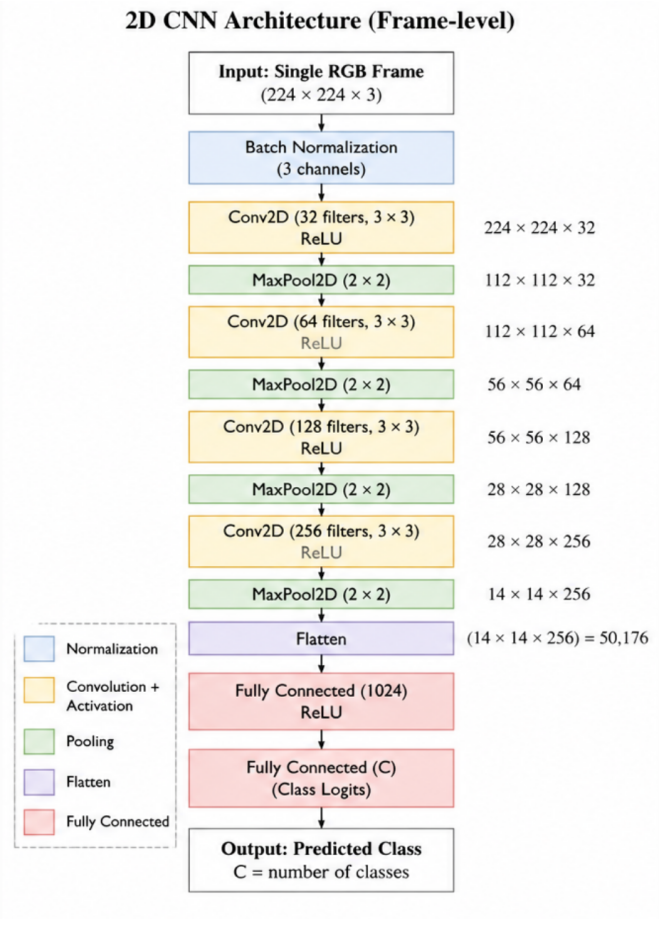

Fig. 1. Simple CNN architecture used as the frame-level baseline. The model learns spatial features from individual frames but does not directly model motion.

The model also uses residual blocks, Layer Normalization, ReLU activation, downsampling stages, global average pooling, dropout, and a dense classification layer. The final output predicts one of the accident classes: head-on, rear-end, sideswipe, single vehicle, or T-bone.

E. Training Strategy for 3D CNN

The 3D CNN training setup uses 16-frame clips, batch size 4, a maximum of 30 epochs, Adam optimizer, and a learning rate of 1×10^{-4}. Since accident classes can be imbalanced, class weighting and oversampling are used to reduce bias toward the majority classes.

Several training controls are also used. Early stopping monitors validation loss and stops training when performance stops improving. Model checkpointing saves the best model. ReduceLROnPlateau lowers the learning rate when validation

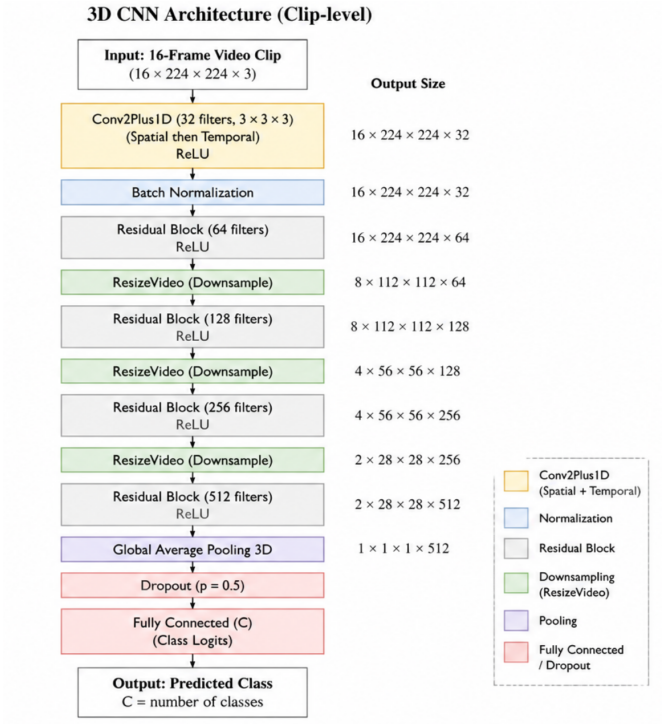

Fig. 2. 3D CNN architecture for clip-level accident classification. The model uses 16-frame clips to learn appearance and motion together.

loss stops improving. These controls help prevent overfitting and improve training stability.

Data augmentation is also applied. Horizontal flipping helps the model generalize to accidents from different directions. Color jitter helps the model handle different lighting and weather conditions. Temporal cropping helps the model learn from different parts of the video sequence.

F. Stage 3: Temporal and Spatial Localization

Classification alone answers what type of accident happened, but it does not answer when or where it happened. For real-world use, timing and location are very important. A traffic monitoring system should ideally show the crash moment and point to the crash region.

For temporal localization, we analyze bounding-box dynamics. Vehicle bounding boxes are detected across frames. For each frame, the system sums the vehicle bounding-box areas. When a crash occurs, the scene usually changes suddenly because vehicles overlap, stop, rotate, or move closer. The frame with the largest bounding-box area change is selected as the predicted accident frame.

For spatial localization, the system uses YOLO detections. At the predicted crash frame, the detected vehicles are compared to find the closest pair. The midpoint between the closest pair is used as the estimated crash location. This gives a simple and interpretable location estimate.

This localization stage is not meant to replace the classifier. Instead, it adds an explanatory layer. The classifier predicts the accident type, and the localization module explains the time and approximate location of the event.

$$A_t = \sum_{i=1}^{n_t} w_{i,t} h_{i,t} \tag{1}$$

In this equation, A_t is the total bounding-box area at frame t, n_t is the number of detected vehicles in that frame, and $w_{i,t}$ and $h_{i,t}$ are the width and height of the i-th detected bounding box.

$$t^* = \arg\max_t |A_t - A_{t-1}| \tag{2}$$

Here, t^* is the predicted accident frame. The system selects the frame where the change in total bounding-box area is largest.

$$p^* = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right) \tag{3}$$

Here, p^* is the predicted accident location, computed as the midpoint between the centers of the closest vehicle pair.

III. Dataset Description

The dataset used in this project is the Kaggle ACCIDENT dataset, which contains synthetic traffic collision videos. Synthetic data is useful in this project because accident data is difficult to collect in the real world, and real accident videos may have privacy or safety concerns. Synthetic simulation provides controlled scenes, labels, accident timing, and object annotations.

A. Dataset Structure

The dataset contains different accident types, including head-on collisions, rear-end collisions, sideswipe accidents, single-vehicle crashes, and T-bone collisions. These classes represent common traffic accident categories and allow the model to learn different crash patterns.

The dataset includes a labels file that works as the central index. This file connects each video path with the accident type, weather, map, accident time, and accident frame. The annotation files also contain different layers of information, including base detections, collision boxes, actor IDs, and sensor metadata.

This structure is useful because the project needs both classification and localization. The label file supports accident-type classification, while the annotation files support temporal and spatial analysis.

B. Accident Timing Distribution

One important dataset observation is that accidents usually occur early in the videos. The mean accident time is 7.61 seconds and the median accident time is 6.90 seconds. This affects model design because temporal windows must cover the crash onset. If the clip window misses the early crash moment, the model may only see normal driving or post-crash frames.

TABLE I
DATASET FIELDS USED IN THE PROJECT.

Field	Purpose
rgb_path	Path to video or image frames
accident type	Class label for model training
weather	Scene condition information
map	Simulation environment information
accident_time	Time of collision in seconds
accident_frame	Frame index of collision
base annotations	Object detections in frames
collision annotations	Collision boxes and actor IDs
sensor metadata	Camera/sensor placement information

This observation supports the use of 16-frame clips and temporal cropping. It also supports the localization stage because the system must focus on detecting the sudden event rather than only classifying the whole video.

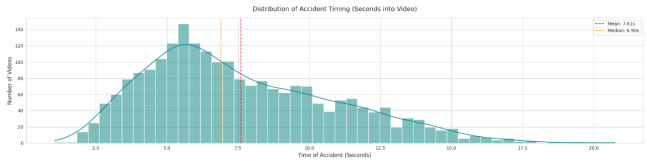

Fig. 3. Accident timing distribution. The mean accident time is 7.61 seconds and the median accident time is 6.90 seconds, showing that most collisions occur early in the video.

C. Dataset Visualizations

The EDA and preview notebooks are important because they help verify the data structure before training. Dataset preview images show sample videos, labels, annotation layers, and timing information. These visualizations help confirm that the dataset supports both accident classification and accident localization.

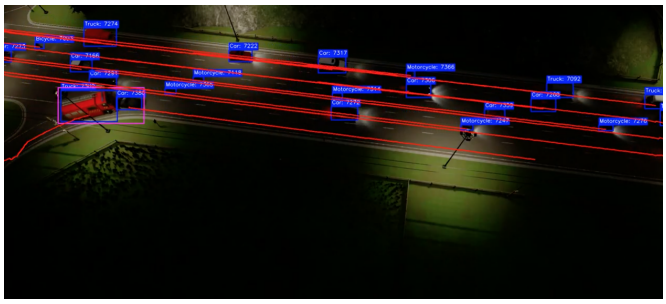

Fig. 4. Synthetic dataset preview showing video samples and annotation structure.

IV. Results and Discussion

The results are best understood as an ablation across three stages. Each stage adds a capability that the previous stage lacked. The simple CNN provides spatial classification. The 3D CNN adds motion-aware classification. The localization stage adds interpretable timing and coordinate outputs.

A. Ablation Across Pipeline Stages

The most important result of this project is the system progression. The 2D CNN is useful as a baseline, but it is limited because it only sees one frame at a time. The 3D CNN improves the model by using 16-frame clips and learning motion. The localization stage makes the result actionable by identifying accident timing and location.

TABLE II
ABLATION SUMMARY OF THE ACCIDENT DETECTION PIPELINE.

Stage	Main Capability	Limitation / Improvement
Simple CNN	Frame-level spatial classification	Fast baseline, but weak temporal context
3D CNN	Motion-aware clip classification	Learns appearance and motion together
Localization	Time and coordinate estimation	Converts classification into actionable output

B. 2D CNN Results

The 2D CNN result visualization shows how the model predicts accident labels on individual frames. This confirms that the frame-level model can detect some spatial crash cues. For example, it can learn vehicle positions, road context, and visible collision evidence.

However, the result also shows why frame-level classification is limited. Accident detection depends heavily on motion. A frame before a crash and a frame during a crash may look similar, but the motion between them is what defines the event. Therefore, the 2D CNN is useful as a baseline but not sufficient as the final system.

TABLE III
DETAILED PER-CLASS PERFORMANCE METRICS

Class	Precision	Recall	Average Precision (AP)
Non-Accident	0.5520	0.7619	0.5955
Head-on	0.7119	0.5468	0.7034
Rear-end	0.7177	0.4731	0.7034
Sideswipe	0.5977	0.3367	0.4881
Single	0.8513	0.4702	0.7908
T-bone	0.7123	0.5784	0.7032

C. 3D CNN Results

The 3D CNN result visualization shows what the model learns in the final layer. Since the model receives 16-frame clips, it can learn temporal accident patterns instead of relying

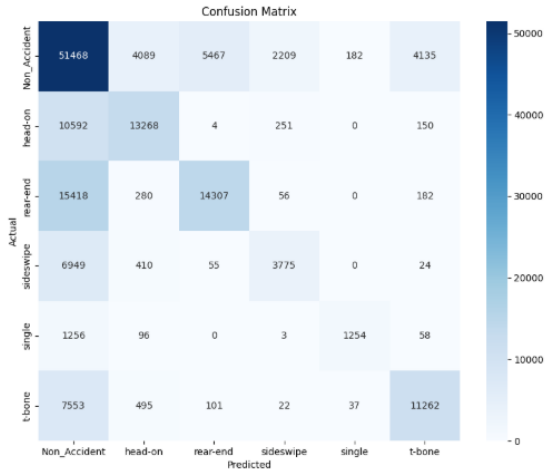

Fig. 5. 2D CNN result visualization. The model predicts accident labels frame by frame, which is useful for spatial cues but limited for motion understanding.

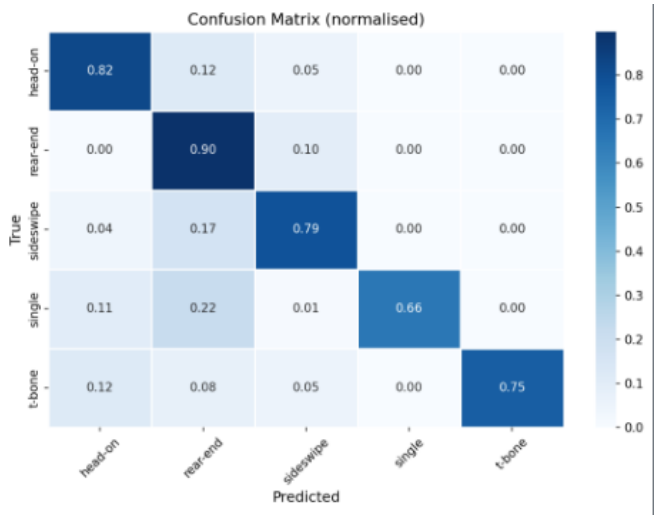

Fig. 6. Confusion matrix for the 3D CNN model. This figure should be generated from the evaluation notebook and uploaded to Overleaf.

on isolated frames. This makes the model more appropriate for accident detection because collisions are defined by motion.

The 3D CNN evaluation includes test evaluation, precision, recall, average precision, classification report, confusion matrix, and CAM visualization. The confusion matrix helps identify which accident types are easier or harder to classify. The CAM visualization helps show which spatial regions the model focuses on when making predictions.

TABLE IV
3D CNN PER-CLASS PERFORMANCE METRICS

Class	Precision	Recall	Avg Precision (AP)
Head-on	0.878	0.822	0.923
Rear-end	0.807	0.899	0.960
Sideswipe	0.639	0.787	0.830
Single	1.000	0.658	0.930
T-bone	0.997	0.745	0.987
Mean AP			**0.926**

Compared with the 2D CNN baseline, the 3D CNN achieves stronger per-class performance because it learns temporal motion patterns from frame sequences rather than relying only on spatial cues from individual frames.

D. Localization Results

The localization stage produces the most interpretable result. In the example visualization, the predicted accident time is 7.34 seconds. The red dot marks the predicted crash location, and the bounding-box dynamics curve shows how the system selected the crash moment.

This is important because classification alone is not enough for real-world systems. A traffic operator needs to know not just that an accident happened, but where to look and when the event occurred. The localization module answers this by combining temporal bounding-box changes with YOLO vehicle detections.

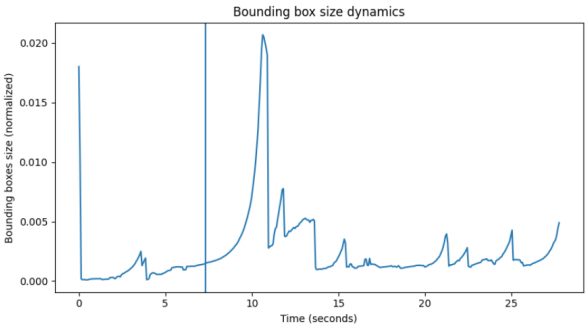

Fig. 7. Localization result showing predicted accident time, predicted crash point, and bounding-box size dynamics curve.

E. Discussion of Findings

The main lesson from the results is that accident detection improves when the model moves from classification to understanding. The simple CNN can learn spatial cues, but it cannot fully understand accidents because accidents are temporal events. The 3D CNN improves this by learning motion from clips. The localization stage further improves usefulness by giving time and position.

Synthetic data is also useful because it provides controlled labels, sensor metadata, collision annotations, and accident timing. However, synthetic data may not perfectly match real-world traffic videos. Real roads include more variation in camera angles, lighting, weather, occlusion, and driver behavior. Therefore, future work should test the pipeline on real-world accident videos.

V. CONCLUSION

This project developed a multi-stage pipeline for traffic accident detection and localization. The system begins with a simple CNN baseline, improves with a 3D CNN for temporal

modeling, and adds a localization stage for accident timing and position estimation.

The main finding is that temporal context matters. Accidents are not isolated images; they are events that happen over time. The 3D CNN is more appropriate than a frame-level model because it can learn motion patterns. Another important finding is that interpretability matters. A model that only predicts an accident type is less useful than a model that also predicts when and where the accident occurred.

The final direction of the system is to classify crash type, identify crash time, and estimate crash location. This makes the system more practical for intelligent traffic monitoring and emergency response applications.

VI. Code and Design

The codebase is organized into separate components for data preparation, exploratory data analysis, simple CNN modeling, 3D CNN modeling, and temporal filtering/localization. This modular structure improves readability and makes it easier to debug or extend the system.

The data preparation code handles loading the dataset, reading labels, extracting frames, and organizing input paths. The EDA notebooks support dataset visualization and timing analysis. The simple CNN code provides a frame-level baseline. The 3D CNN code handles clip preparation, training, evaluation, confusion matrix generation, and CAM visualization. The temporal filtering notebook supports accident timing and localization.

The project uses several tools:

- PyTorch for the simple CNN implementation.
- TensorFlow/Keras for the 3D CNN implementation.
- OpenCV for video loading, frame extraction, and visualization.
- Ultralytics YOLO for vehicle detection.
- pandas for reading labels and organizing metadata.
- matplotlib for plots, graphs, and visual result displays.

From a design perspective, the strongest part of the codebase is that each model stage is separated. This makes the workflow easier to follow: first prepare data, then explore data, then train a baseline, then train a temporal model, and finally run localization. This matches the project goal of moving from frame-level classification to temporal and spatial accident understanding.

Project Code Repository: Traffic Accident Detection and Localization using Deep Learning

VII. Workload Distribution

The project work was divided across model development, data processing, evaluation, and localization.

- **Gagandeep:** Worked on data preparation, exploratory analysis, model implementation for temporal and spatial localization, training workflow, and repository organization.
- **Channing:** Worked on dataset preparation, and 2D-CNN, and evaluation support.
- **Viwesh:** Worked on dataset preparation, and 3D-CNN, and evaluation support.

Acknowledgment

We acknowledge the Kaggle ACCIDENT dataset, the project teammates, the instructor, and the tools used in this project, including PyTorch, TensorFlow/Keras, OpenCV, Ultralytics YOLO, pandas, and matplotlib.

References

[1] Kaggle, "Accident Detection Dataset," Kaggle Competition.

[2] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in Proceedings of the IEEE International Conference on Computer Vision, 2015.

[3] S. Robles-Serrano, G. Sanchez-Torres, and J. Branch-Bedoya, "Automatic Detection of Traffic Accidents from Video Using Deep Learning Techniques."

[4] M. A. K. Rifat, A. Kabir, and A. S. Huq, "An Explainable Machine Learning Approach to Traffic Accident Fatality Prediction."

[5] W. Zhou, "Traffic Accident Severity Prediction Based on Data Cleaning and Machine Learning."