# Predicting Vehicle Collisions from Dashcam Video

Can we predict crashes before they happen? Spoiler: we can’t.

# Today's Presenters

Adam
Stein

Hung
Tran

Sean
Morris

David
Corcoran

# Agenda

1. Introduction
2. Data Collection and Preprocessing
3. Model Architectures
4. Model Results
5. Conclusion, Limitations, and Future Work

# Introduction

# Introduction

- Enhancing road safety through early accident prediction
- Supports autonomous vehicles and advanced driver assistance systems (ADAS)
- Our Task: Analyze dashcam footage to predict vehicle collisions before they happen
- Challenges: Real-world complexity including:
  - Varying weather conditions
  - Visual occlusions
  - Unpredictable road events

**Goal: Build a neural network model that accurately classifies whether a dashcam video contains a collision**

# Data Collection and Preprocessing

Raw dashcam videos

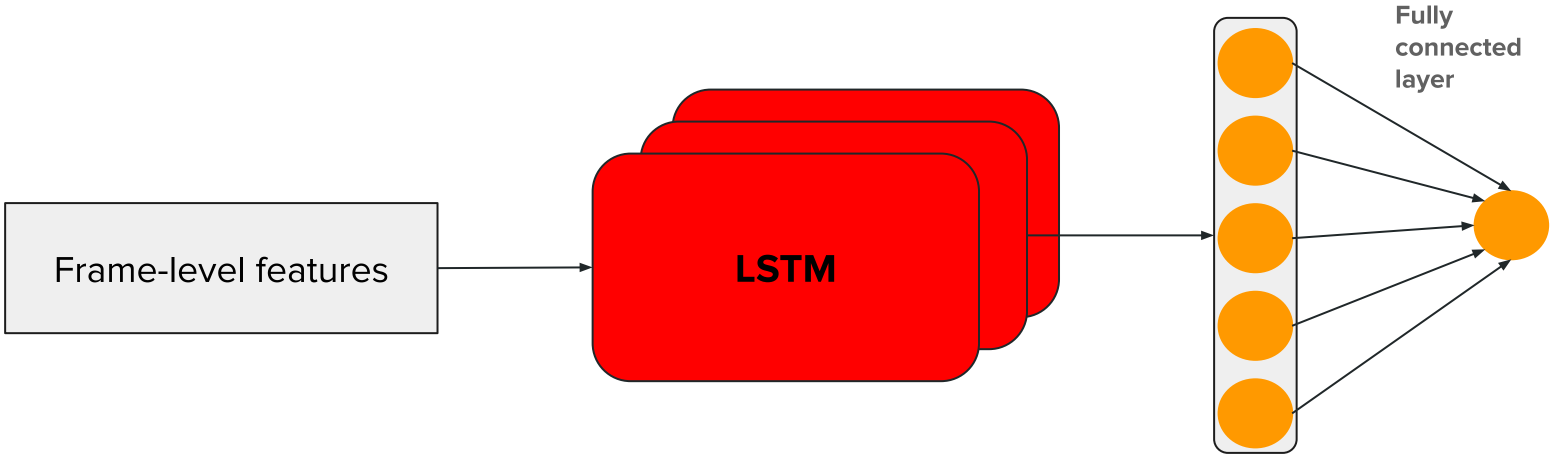
- 1500 annotated video clips ~40s in length & 30 fps
- Represent diverse range of real-world conditions
- Labeled as collision vs normal driving

- Extracted frames of each video at 1 frame per second
- ~60,000 total frames extracted

- Extracted object-level spatial features per frame using YOLOv8 (You Only Look Once)
- Detected bounding boxes, object classes, and confidence scores

Fed a **sequence of frame-level features** into an LSTM, GRU, and Transformer

# Model Architectures - LSTM




- RNN designed to better handle long-term dependencies
- **Cell state** flows through time with three gates controlling the flow of information:
  - **Forget Gate**: Decides what information to discard
  - **Input Gate**: Decides what new information to add
  - **Output Gate**: Decides what information to output
- **Preserves gradients** over long sequences

## Hyperparameter Tuning Results

**Hidden Layers:** 3

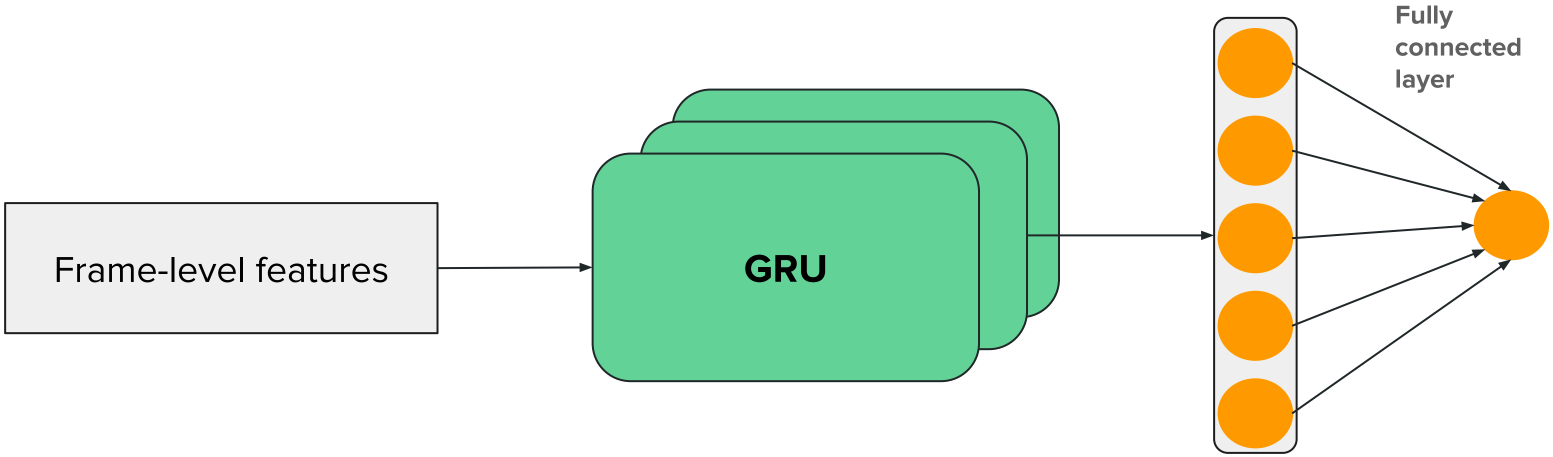
**Optimizer:** Adam

**Layer Size:** 128

**Learning Rate:** 0.001

**Dropout:** 0.2

# Model Architectures - GRU




- **Do not use a separate cell state** — they rely solely on the hidden state to store and transfer information
- Two gates instead of three:
  - **Update Gate**: Controls what previous information to retain and how much of the new input to use
  - **Reset Gate**: Decides how much of the past information to forget
- Less complex and typically train faster than LSTMs

## Hyperparameter Tuning Results

**Hidden Layers:** 2

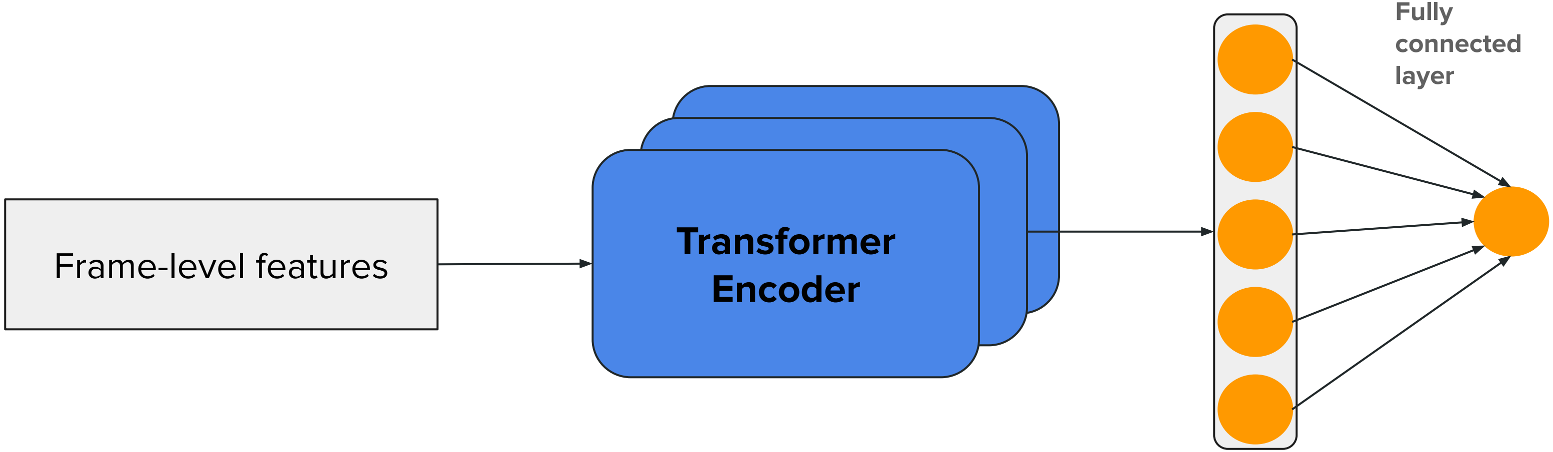
**Optimizer:** Adam

**Layer Size:** 128

**Learning Rate:** 0.001

**Dropout:** 0.2

# Model Architectures - Transformer




- Unlike RNNs, **Transformers** do not process data sequentially
  - No hidden or cell state – use **self-attention** to dynamically relate all time steps to one another
- **Self-Attention Layer**:
  - Allows model to focus on different parts of the input sequence when encoding a particular time step
- **Positional Encoding:**
  - Positional signals (sine in this case) to preserve sequence structure
- **Layer Normalization & Skip Connections:**
  - Used to stabilize training and improve gradient flow

## Hyperparameter Tuning Results

**Hidden Layers:** 2 (Encoders)

**Optimizer:** Adam

**Layer Size:** 128

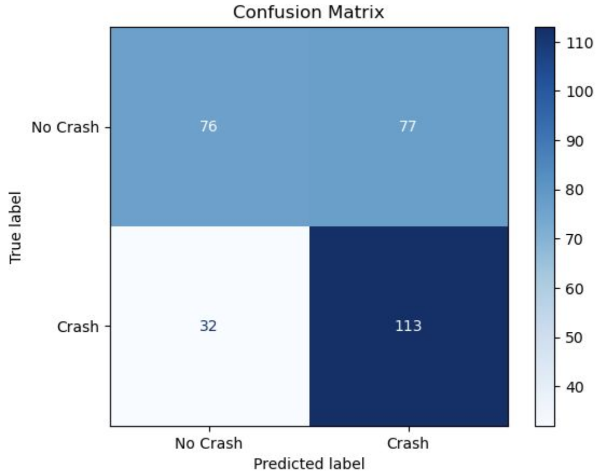
**Learning Rate:** 0.001

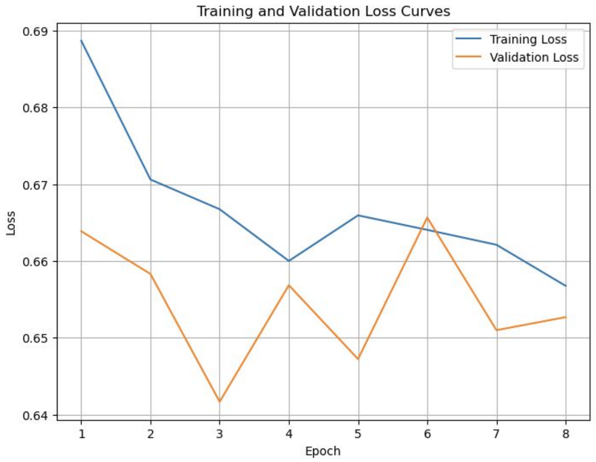
**Dropout:** 0.1

# Model Results - LSTM

## Evaluation Metrics

| | |
|---|---|
| Accuracy | 63% |
| Precision | 59% |
| Recall | 78% |
| F1 Score | 67% |

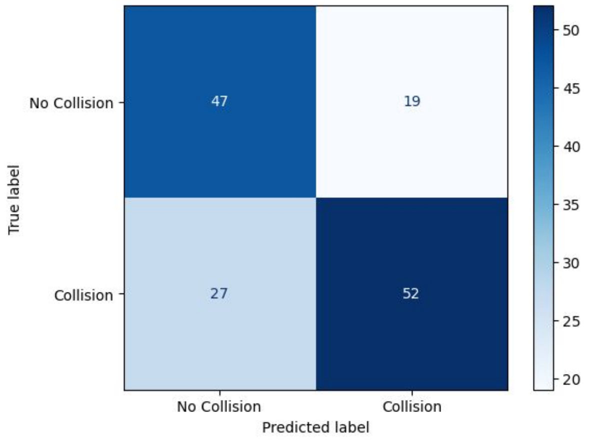


# Model Results - GRU

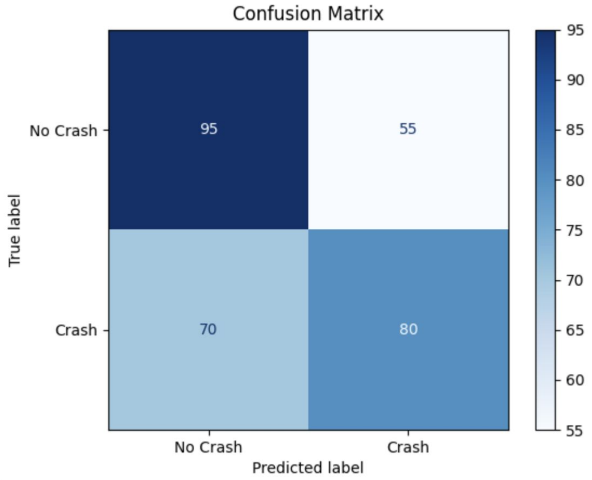
## Evaluation Metrics

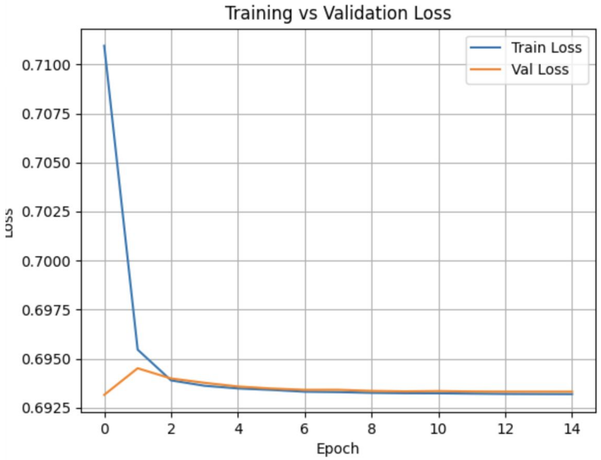
| | |
|---|---|
| Accuracy | 68 % |
| Precision | 73 % |
| Recall | 66 % |
| F1 Score | 69 % |




# Model Results - Transformer

| Evaluation Metrics | |
|---|---|
| **Accuracy** | 58% |
| **Precision** | 59% |
| **Recall** | 53% |
| **F1 Score** | 56% |



# Conclusion, Limitations, and Future Work

Limitations:

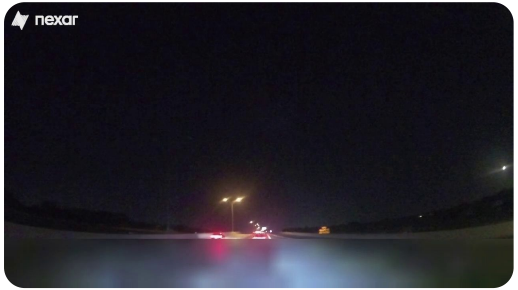
- YOLO captures spatial features (per frame) and LSTM models temporal changes

Future Work:

- Train a **3D** convolutional neural network (CNN) to extract **spatiotemporal features**
  - 3D convolution would **apply a filter** that slides not only **left-right and up-down**, but also **forward-backward through the frames**
  - Model would learn to distinguish between “collision” and “normal” driving
- Extend the model to incorporate additional sensor inputs, such as GPS, LiDAR, accelerometers, gyroscopes, or vehicle telemetry

# Data and YOLO Limitations

## Camera Angles

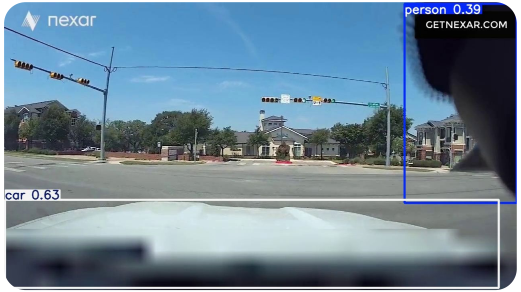




## Weather Conditions

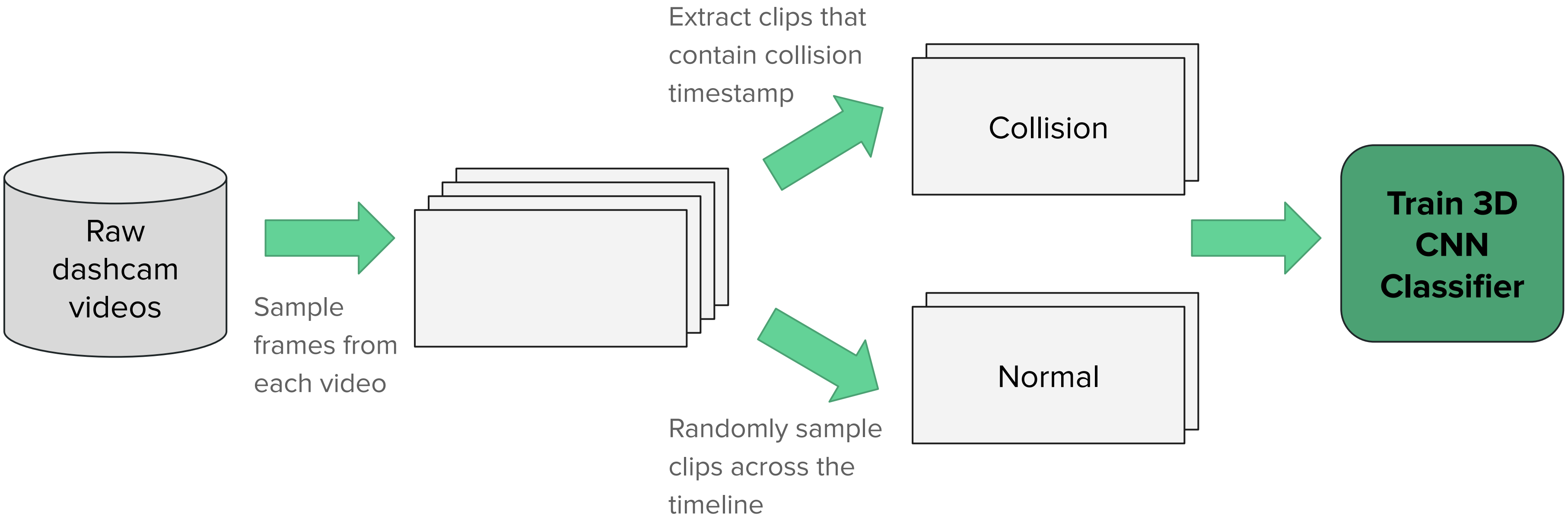


## YOLO Misclassifications




# 3D Convolution for Video Processing

**Goal: Train a binary classifier using a 3D CNN that learns spatiotemporal patterns distinguishing “collision” from “normal” driving**



# References


https://www.kaggle.com/competitions/nexar-collision-prediction

https://arxiv.org/abs/2503.03848

https://www.geeksforgeeks.org/video-classification-with-a-3d-convolutional-neural-network/