AUTOMATIC ACCIDENT DETECTION USING 3DCNN MODEL AND ANOMALY DETECTION ALGORITHMS

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE300-MINI PROJECT

Submitted by

RAJOLI PAVANI REDDY
(Reg No : 225003101)
KANDULA VENKATA PADMAJA
(Reg No : 225003166)
SETTIPALLI MOUNIKA
(Reg No : 225003191)

May 2024

Department of Computer Science and Engineering

SRINIVASA RAMANUJAN CENTRE

Kumbakonam, Tamil Nadu, INDIA - 612 001

Department of Computer Science and Engineering

SRINIVASA RAMANUJAN CENTRE

Kumbakonam, Tamil Nadu, INDIA - 612 001

May 2024

Bonafide Certificate

This is to certify that the report titled **" Automatic Accident Detection using 3DCNN model and Anomaly Detection Algorithms"** submitted as a *requirements for the award of the degree of B.Tech.,* Computer Science and Engineering to the SASTRA Deemed to be University, is a bonafide record of the work done by **Ms. Rajoli Pavani Reddy (225003101) , Ms. Kandula Venkata Padmaja (225003166) , Ms. Settipalli Mounika (225003191)** during the final semester of the academic year 2023-2024, in the Srinivasa Ramanujan Centre, under my supervision. This project report has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

Signature of Project Supervisor :

Name with Affiliation :

Date :

Main Project *Viva voc*e held on _____

Examiner 1 **Examiner 2**

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

Department of Computer Science and Engineering

SRINIVASA RAMANUJAN CENTRE

Kumbakonam - 612 001

Declaration

We declare that the project report titled **"Automatic Accident Detection using 3DCNN model and Anomaly Detection Algorithms"** was submitted by us in an original work done by us under the guidance of **Dr .A. Menaga , Department of CSE, SRC, SASTRA Deemed to be University, Kumbakonam,** during the sixth semester of the academic year 2023-24, in the **Srinivasa Ramanujan Centre**. The work is original and wherever we have used the materials from other sources, we have given due credit and cited them in the text of the project report. This project report has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

Signature of the candidates :1.

2.

3.

Name of the candidates : 1.Rajoli Pavani Reddy

2.Kandula Venkata Padmaja

3.Settipalli Mounika

Date :

Acknowledgements

We would like to thank our Honorable Chancellor Prof.R.Sethuraman for providing us with an opportunity and necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam and Dr. S. Swaminathan,** Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli,** Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. Ramaswamy,** The Dean and **Dr. A. Alli Rani,** Associate Dean, Srinivasa Ramanujan Centre for their constant support and suggestions when required without any reservations.

We express our sincere gratitude to **Dr. V. Kalaichelvi,** Associate Professor, Department of Computer Science and Engineering for her unconditional support and encouragement for completing the project.

Our guide **Dr.A.Menaga,** Assistant Professor **,** Department of Computer Science & Engineering, Srinivasa Ramanujan Centre was the driving force behind this whole idea from the start her deep insight in the field and invaluable suggestions helped me in making progress throughout our project work.

We also thank the Project Coordinator **Dr. S. Priyanga,** Assistant Professor, Department of Computer Science & Engineering, Srinivasa Ramanujan Centre for her ever-encouraging spirit and meticulous guidance for the completion of the project. We also thank the panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the Srinivasa Ramanujan Center who have either directly or indirectly helped us in the completion of the project. We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the completion of this project. We thank you all for providing us with an opportunity to showcase my skills through the project.

LIST OF FIGURES

Fig.no	Title	Page.No
1.1	CNN Model Architecture	5
1.2	LSTM Model Architecture	6
1.3	Isolation Forest Architecture	6
1.4	Proposed Architecture	7
1.5	Solution approach	7
2.1	LSTM Accuracy and Loss	17
2.2	Mean pixel value using Box Plots for Outliers for LSTM	18
3.1	Isolation Forest Classification Table	19
3.2	Distribution of Anomaly scores	19

ABBREVATIONS

CNN	Convolution Neural Networks
ReLU	Rectified Linear Activation.Function
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
resnet	Residual Network
PIL	Python Imaging Library
GANs	Generative Adversarial Networks

ABSTRACT

Accident detection in video surveillance systems is crucial for prompt response and minimizing harm. This study develops a robust accident detection system using advanced machine learning techniques, combining a 3DCNN for video classification with LSTM for temporal anomaly detection and Isolation Forest for spatial anomaly detection. The system attains a high level of accuracy by leveraging the Car Crash Dataset (CCD), which consists of 500 videos depicting accidents and 500 videos without accidents, amounting to a total of 50,000 frames. The 3D CNN captures spatiotemporal features to classify video frames, while the LSTM enhances accuracy by analyzing temporal sequences for anomalies. Isolation Forest effectively distinguishes accidents from normal scenes based on anomalous spatial patterns. The proposed system offers scalability and robust performance, suitable for real-world surveillance applications.

Key Words:

Traffic accident detection, temporal learning, Neural Networks, 3DCNN model, Anomaly Detection Algorithms, machine learning.

Specific Contribution:

Rajoli Pavani Reddy - Data Analysis on CCD dataset and 3DCNN Model Training

Kandula Venkata Padmaja – Implementation of Isolation Forest for Anomaly Detection

Settipalli Mounika – Implementation of LSTM for Anomaly Detection

Technical Limitations & Ethical Challenges faced:

- Training the model using 1000 videos (50000 frames) is very difficult
- The installation of camera is very difficult.

Table of Contents

Title		PageNo.
Bonafide Certificate		ii
Acknowledgements		iv
List of Figures		v
Abbreviations		vi
Abstract		vii
1.	Summary of the base paper	1
2.	Merits and Demerits of the base paper	10
3.	Source code	12
4.	Snapshots	17
5.	Results and discussions	20
6.	Conclusions and Future plans	21
	References	23
7.	Appendix A Base Paper	24

CHAPTER 1

SUMMARY OF THE BASE PAPER

Title: TempoLearn Network: Leveraging Spatio-Temporal Learning for Traffic Accident Detection

Journal Name: Digital Object Identifier –IEEE Access

Authors: Soe Sandi Htun,Ji Sang Park,Kang Woo Lee, Ji Hyeong Han

Year: 2023.

1.1 SUMMARY:

- The project aims to develop a comprehensive system for video analysis, with a primary focus on accident detection using 3D CNNs and anomaly detection.
- Constructed a 3D CNN architecture consisting of Conv3D, MaxPooling3D, Flatten, Dense, and Dropout layers tailored for precise accident detection.
- Utilized a pre-trained ResNet50 model to extract spatial features from videos, facilitating anomaly detection.
- Trained an Isolation Forest model on the spatial features to effectively detect anomalies.
- Designed an LSTM architecture incorporating TimeDistributed layers to process spatial features and LSTM layers for temporal modeling.
- Trained the LSTM model to detect anomalies by analyzing temporal patterns in video sequences.
- The proposed system given in base paper has successfully detected accidents using deep learning algorithms

1.2 BACKGROUND

Below are some of the key concepts that are useful for the implementation of the project and the reason behind usage of the concepts are described

1.3 3D CONVOLUTION NETWORKS

A 3D Convolutional Neural Network (3DCNN) is an advanced version of the Convolutional Neural Network (CNN) tailored for processing volumetric data like videos. Unlike traditional CNNs, which operate in two dimensions (width and height), 3DCNNs process data across three dimensions—width, height, and depth. This allows them to capture spatial details as well as temporal changes inherent in the data. For example, in video analysis, 3DCNNs can analyze changes occurring over successive frames, enabling tasks like action recognition or event detection. The basic operations of 3DCNNs include convolution, pooling, and activation functions, similar to

2D CNNs. These operations extract features, reduce spatial dimensions, and introduce non-linearity, respectively. Overall, 3DCNNs excel in capturing both spatial and temporal dependencies in volumetric data, making them suitable for tasks requiring understanding of three-dimensional structures or sequences of data over time.

1.4 INTRODUCTION

The purpose of this 3DCNN model is that it will learn to detect the accidents using the deep learning methodology and techniques which increase the performance of the model to clone the behavior of the user and implement proper environment.

1. Convolutional Layer:
The convolutional layer is crucial in altering the input data by utilizing a set of connected neurons from the previous layer. It computes the dot product between regions of neurons in the input layer and locally connected weights in the output layer, resulting in the final output volume for the layer. This process is fundamental to convolutional neural networks.

2. Convolution:

Convolution is a computational procedure that defines how two sets of data will merge. In a CNN, a feature detector applies a convolution kernel to the input, yielding a feature map as output. This is achieved by sliding the kernel across the input image and multiplying it with the data segment within its confines to generate a single feature map. Finally, each filter's activation map is stacked along the depth dimension to create a 3D output. Parameter optimization is typically conducted via gradient descent.

3. Hyperparameters:

Hyperparameters play a crucial role in determining the spatial organization and size of a convolutional layer's output volume. Some of the most important hyperparameters include:

- **Filter Size:** Filters are typically small in size and have three dimensions: width, height, and color channel.
- **Output Depth:** Determines how many neurons in the CNN layer are connected to the same input volume regions.
- **Stride:** Specifies the filter's sliding speed for each application. The stride value is inversely proportional to the depth of the output volume.
- **Zero Padding:** Useful for determining the spatial size in the output volume, particularly when maintaining the spatial size of the input volume in the output volume is preferred.

4. Pooling Layer:

Pooling layers are instrumental in gradually reducing the spatial size of data representations, thereby preventing overfitting on training data. Typically placed between convolutional layers,

pooling layers employ operations like max pooling to spatially resize the input data. Pooling layers have no learnable parameters and are often zero-padded.

5. Fully Connected Layer:

The fully connected layer serves as the network's output layer, with an output dimension usually represented as [1 * 1 * N], where N denotes the number of output classes to be evaluated. In fully connected layers, neural network parameters and hyperparameters are presented.

RESNET50:

ResNet-50 is a convolutional neural network architecture that belongs to the ResNet (Residual Network) family. ResNet-50 specifically consists of 50 layers, including convolutional layers, pooling layers, fully connected layers, and shortcut connections known as skip connections or identity mappings.One of the key innovations of ResNet-50 is the introduction of residual blocks, which help address the vanishing gradient problem commonly encountered in deep neural networks. These residual blocks allow the network to learn residual mappings, making it easier to train very deep networks.ResNet-50 has been widely used for various computer vision tasks such as image classification, object detection, and image segmentation. It has achieved state-of-the-art performance on benchmark datasets like ImageNet.

ANOMALY DETECTION TECHNIQUES

Anomaly detection encompasses training algorithms to recognize irregular patterns within datasets and subsequently identify instances that significantly deviate from these established norms. This task holds paramount importance across diverse domains, as anomalies often signify critical events, errors, or rare incidents demanding special attention. While supervised methods involve training models on labeled datasets where anomalies are explicitly marked, enabling algorithms to differentiate between normal and anomalous instances based on extracted features, unsupervised techniques operate on unlabeled data to detect deviations from expected patterns without prior anomaly knowledge. Time-series data poses unique challenges, necessitating specialized algorithms like Autoencoders or LSTM networks to effectively capture temporal patterns and anomalies within sequential data. Ensemble methods further enhance detection accuracy and robustness by combining multiple anomaly detection techniques.

ISOLATION FOREST

Isolation Forest, an anomaly detection algorithm, focuses on isolating anomalies rather than modeling normal data points. It constructs a set of decision trees, randomly selecting features and split values within the dataset's range during each tree's recursive building process. Anomalies, being typically isolated instances, are expected to have shorter paths from the root of the tree compared to normal instances, which are more densely clustered. An anomaly score is then calculated for each data point based on its average path length across all trees. Isolation Forest is efficient for high-dimensional datasets, less sensitive to irrelevant features, and has low computational complexity, making it suitable for large-scale datasets.

Contamination: This hyperparameter specifies the expected proportion of outliers in the data. In the provided code, it is set to 0.1, indicating that approximately 10% of the data is considered outliers. Adjusting this parameter allows you to control the sensitivity of the model to anomalies.

Number of Estimators (n_estimators): This parameter determines the number of isolation trees in the forest. Increasing the number of estimators can improve the model's ability to detect outliers but may also increase computation time.

Maximum Samples (max_samples): It specifies the maximum number of samples to be used for constructing each isolation tree. Reducing this parameter can speed up the training process but may also decrease the model's effectiveness, especially for datasets with high-dimensional features.

Maximum Features (max_features): This parameter controls the maximum number of features to consider when splitting a node in the isolation tree. Choosing a smaller value can reduce overfitting, especially in high-dimensional datasets.

Bootstrap: This Boolean parameter indicates whether to use bootstrap sampling when constructing each isolation tree. Bootstrap sampling can introduce randomness and diversity into the trees, potentially improving the model's robustness.

LONG SHORT TERM MEMORY(LSTM)

LSTM, a variation of recurrent neural networks (RNNs), is designed to capture extended dependencies in sequential data. In contrast to conventional RNNs, which often encounter gradient vanishing problems, LSTMs incorporate memory cells and gating mechanisms to manage the flow of information within the network. These gates, comprising input, forget, and output gates, enable the selective processing and retention of information across time steps. Employed as part of anomaly detection, LSTM networks excel at recognizing temporal patterns and deviations within sequential data. Their capacity to model long-term dependencies renders them particularly adept at identifying anomalies in time-series data, where deviations from expected patterns evolve over time. By harnessing the inherent memory cells and gating mechanisms, LSTM networks learn to discern between normal and anomalous sequences, detecting subtle irregularities and variations within the data. This pivotal capability enables LSTM networks to serve as valuable tools for anomaly detection across various domains.

LSTM Layer:

Long Short-Term Memory (LSTM) layers are employed to capture temporal dependencies in sequential data, such as video frames. LSTMs are a type of recurrent neural network (RNN) that can retain information over time steps, making them suitable for tasks involving sequential data processing.

Fully Connected Layer:

The final fully connected layer integrates the features extracted by convolutional and LSTM layers to make predictions. This layer typically employs a sigmoid activation function to produce binary classification outputs.

1.5 PROBLEM DEFINITION

Detecting accidents swiftly and accurately in video surveillance systems is crucial for minimizing harm. This study aims to develop a robust accident detection system by leveraging state-of-the-art machine learning techniques. Our method integrates various components to effectively analyze video data and identify anomalies indicative of accidents. We utilize a three-dimensional Convolutional Neural Network (3D CNN) to process video frames, capturing spatiotemporal features effectively. This model is trained on a dataset comprising labeled frames, where accidents are appropriately identified. Moreover, a pretrained Residual Network (ResNet50) is employed to extract key features from the video frames. Isolation Forest is utilized to detect spatial anomalies associated with accidents, while Long Short-Term Memory (LSTM) networks analyze temporal sequences for abnormal patterns. This component complements the 3D CNN by providing additional insights into spatial irregularities associated with accidents. This comprehensive approach enhances accuracy by considering both spatial irregularities and temporal dynamics related to accidents.

- To conduct a comprehensive review of existing methodologies for accident detection in video surveillance systems, with a particular emphasis on machine learning approaches.
- To describe the algorithmic framework of the proposed accident detection system, detailing the integration of components such as 3D CNN for video classification, LSTM for temporal anomaly detection, and Isolation Forest for spatial anomaly detection.
- To develop the architecture for the accident detection system, considering classifier fusion methods to optimize accuracy and performance.
- To analyze the accuracy of decision-making processes based on each aggregation method employed, conducting experiments to evaluate the effectiveness and robustness of the system in detecting accidents.

1.6 PROPOSED METHODOLOGY:

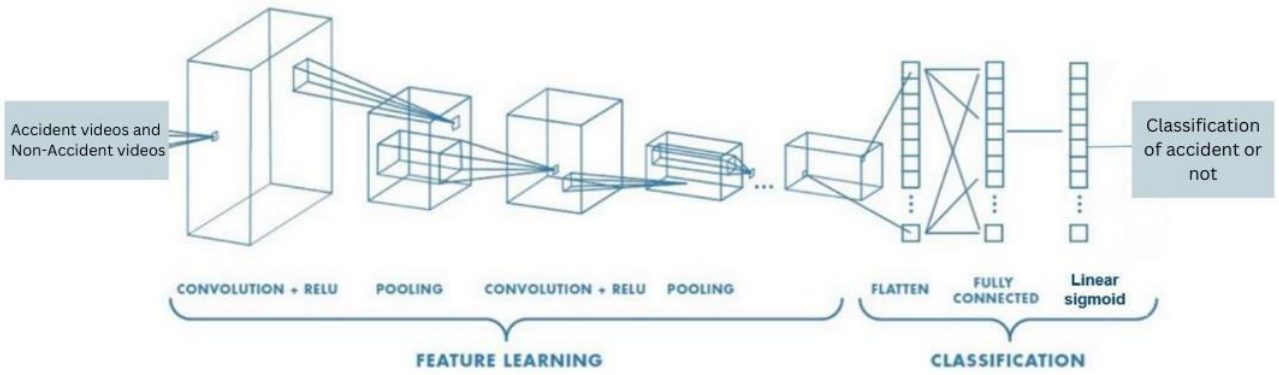


Fig 1.1 . CNN Architecture

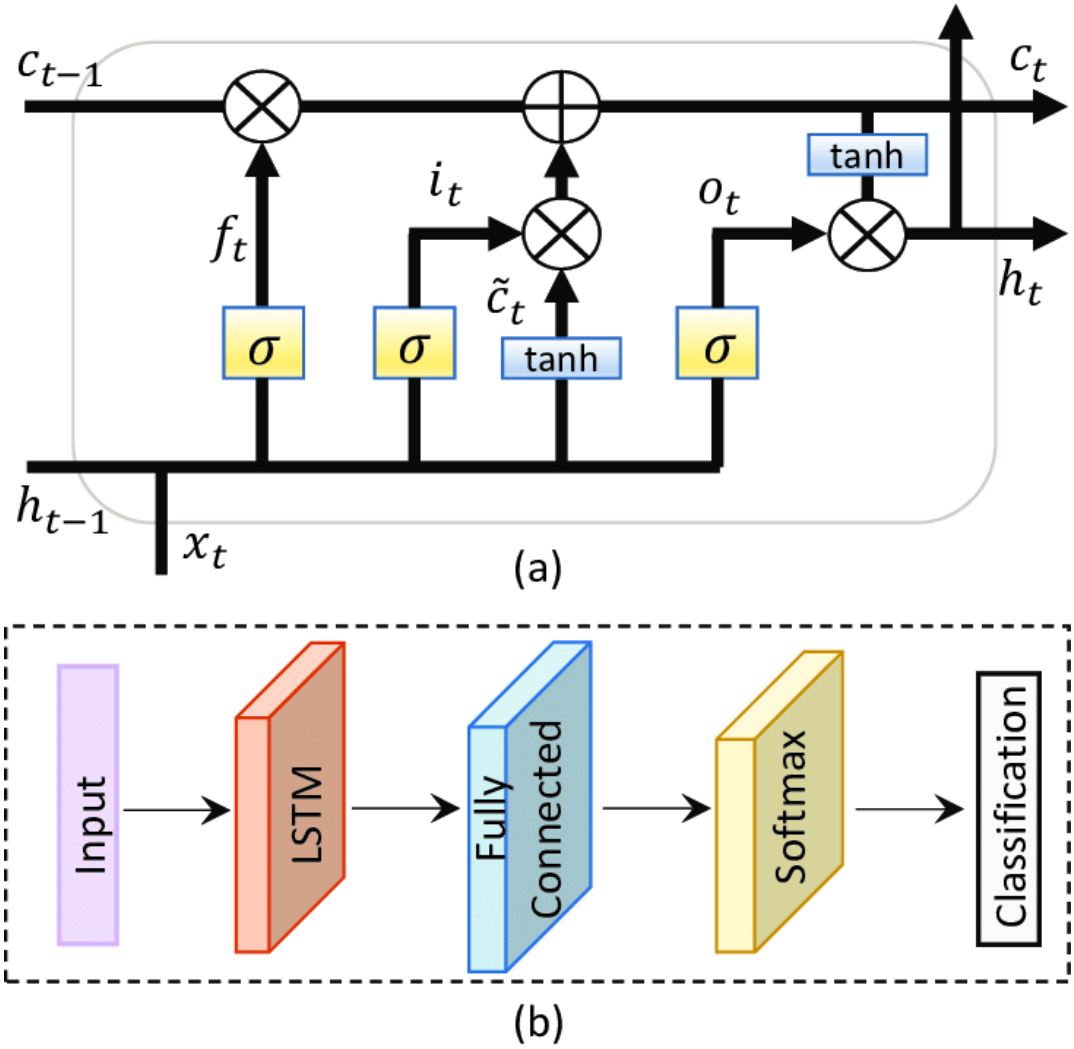


Fig 1.2 LSTM Architecture

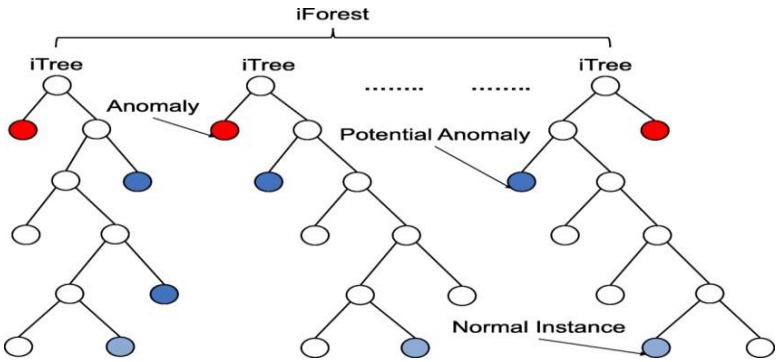


Fig 1.3 Isolation Forest Architecture

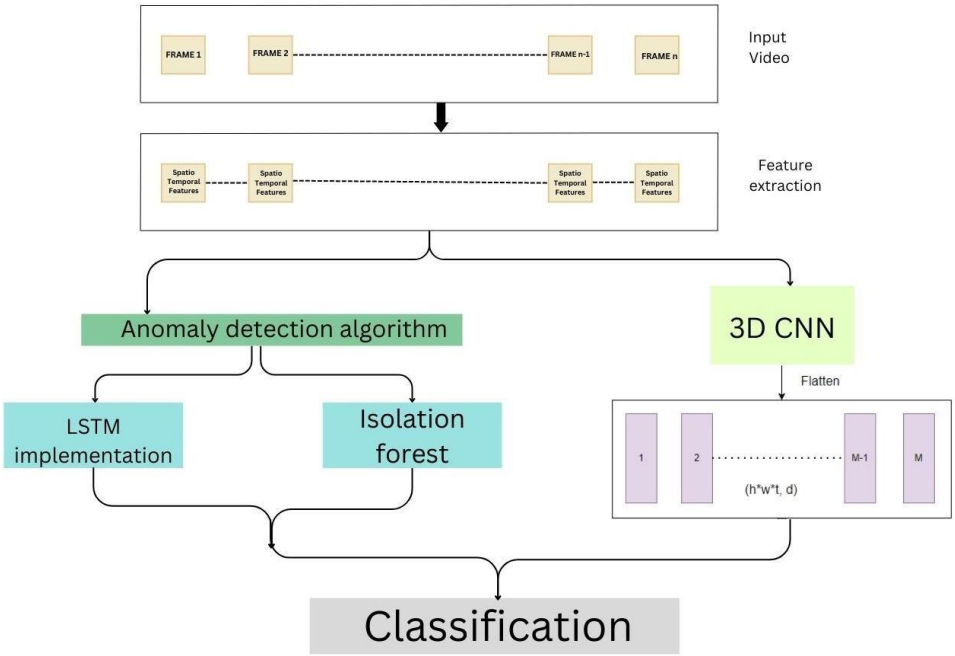


Fig 1.4 Proposed Architecture

1.7 SOLUTION APPROACH

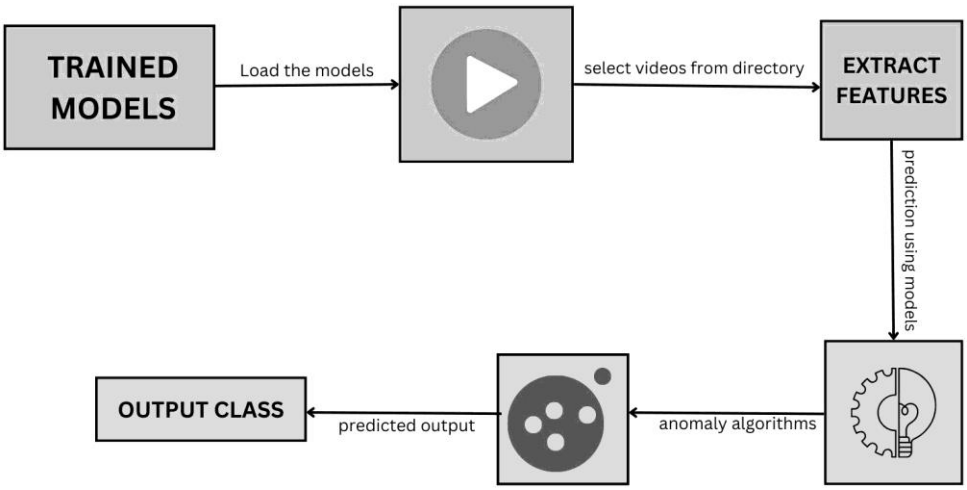


Fig 1.5 solution approach

The solution for the above approach is:

- Utilize the Car Crash Dataset (CCD) to extract features from both accident and non-accident videos.
- Train the integrated model, comprising a 3D CNN, LSTM, and Isolation Forest, using the extracted features.
- Load the trained model into the program for accident detection.
- Input surveillance videos into the program for real-time accident detection.
- For each frame of the video, analyze spatiotemporal features using the 3D CNN and LSTM to detect anomalies indicative of accidents.
- Utilize Isolation Forest to distinguish between normal and abnormal spatial patterns.
- Display the detected accidents with appropriate annotations on the surveillance video for prompt response and analysis.

DATASET

The dataset used for the project is CCD(Car Crash Dataset)face dataset. It has 500 accident videos and 500 non accident videos .For each video the label is given for the 50 frames . The labels are given in a annotation file .

1.8 TECHNOLOGIES USED

To implement this project, several technologies and libraries were utilized, each serving a specific purpose:

TensorFlow and Keras: TensorFlow, serving as the foundation for machine learning tasks, partners with Keras, a user-friendly interface built upon TensorFlow, streamlining neural network development.

NumPy: Empowers efficient mathematical operations and data manipulation, offering support for essential data structures like arrays and matrices, pivotal for neural network computations.

Scikit-learn: Furnishes a rich toolkit for machine learning endeavors, encompassing preprocessing utilities, model training algorithms, and performance evaluation metrics, enriching project efficacy.

OpenCV: Vital for computational optimization, especially in practical applications, aids in various image preprocessing tasks, ensuring swift and efficient processing.

Tqdm: Streamlines progress monitoring with clear and concise progress bars, enhancing visibility into tasks like data loading and video processing.

PIL (Python Imaging Library): Supports a spectrum of image processing tasks, including image loading, manipulation, and storage, frequently employed for preprocessing image datasets.

Tkinter: Empowers the creation of intuitive graphical user interfaces (GUIs) in Python, likely utilized for crafting the user interface of the accident detection system.

The machine used for this project is Visual Studio , Kaggle and laptop with following specifications.

• Processor: Intel i3 Core -2100CPU @ 3.10GHz

• Random Access Memory (RAM) : 8GB

• Operating System : Windows 11 64-bit

1.9 Functions used in source code

CONV3D():

conv3d() is a key function in deep learning for processing 3D data. It applies filters to a 3D input volume, extracting features by sliding over it. This produces multiple feature maps, each highlighting different aspects of the input. Parameters like stride and padding control the filter movement and output size. With an activation function like ReLU, non-linearity is introduced to the output. It forms the backbone of 3D convolutional neural networks, enabling the analysis of volumetric data in various applications.

Maxpooling3d():

MaxPooling3d() is a function used in deep learning to downsample 3D data by selecting the maximum value from smaller sub-volumes. It reduces the spatial dimensions of each feature map while retaining important information. This operation aids in dimensionality reduction and feature selection in 3D convolutional neural networks, enhancing computational efficiency and reducing overfitting.

Dense():

Dense function basically used to creates nodes or neurons in the respectively layer. .Range is specified in the dense function. At last only one output function as dense(1). Dense=activation(dot(input ,kernel)+bias)

Lstm():

lstm() stands for Long Short-Term Memory, which is a type of recurrent neural network (RNN) layer. It's used for processing sequential data, like time series or natural language, by retaining long-term dependencies. LSTMs have internal mechanisms called gates that regulate the flow of information, making them capable of capturing long-range dependencies and avoiding the vanishing gradient problem.

TimeDistributed():

TimeDistributed() is a wrapper function commonly used in deep learning frameworks like Keras. It's often applied to layers such as lstm() to apply them across multiple time steps of input data. This is particularly useful in scenarios where the input has a time dimension, like time series data.

CHAPTER 2

MERITS AND DEMERITS OF BASE PAPER

2.1 MERITS:

- Integrates multiple deep learning models (3D CNN, LSTM, ResNet) for comprehensive analysis of video data, capturing both spatial and temporal features effectively.
- Utilizes advanced feature extraction methods from deep learning architectures like ResNet50 to extract high-level spatial features, enriching the representation of video data.
- Incorporates Isolation Forest algorithm and LSTM model for anomaly detection, enabling the identification of abnormal spatial patterns indicative of accidents with high accuracy.
- Applies cross-validation methodologies to evaluate model performance and generalization across diverse datasets, ensuring reliable accident detection in various scenarios.
- Optimizes model hyperparameters such as batch size, learning rate, and dropout rates to enhance model convergence and performance.
- Designed for scalability, allowing for efficient processing of large volumes of video data in real-time surveillance applications without compromising performance.
- Utilizes parallel processing capabilities for concurrent execution of preprocessing tasks and model predictions, maximizing hardware utilization and accelerating inference speed.
- Implements dynamic batch sizing strategies during training to adaptively adjust batch sizes based on available memory resources, maximizing GPU utilization and minimizing training time without compromising accuracy.
- Integrates multi-modal fusion techniques to combine information from diverse sources such as spatial, temporal, and spectral features, enhancing the discriminative power of the model and enabling robust accident detection in complex scenarios.

2.2 DEMERITS

- Environmental noise, compression artifacts, or image distortions in surveillance footage may introduce false signals or anomalies, leading to incorrect accident detection by the system.
- Objects obstructing the view or partial occlusion of the scene within the video frames could obscure critical information relevant to accident detection, introducing challenges for the system.
- Variations in camera angles and perspectives may influence the quality and coverage of surveillance footage, affecting the system's ability to detect accidents from different viewpoints.
- Integration of multiple deep learning models and algorithms increases the overall complexity of the system, making it harder to interpret and debug, and potentially requiring substantial computational resources for training and inference.

CHAPTER 3

SOURCE CODE

3.1 ANALYSIS:

3.1.1 Extract frames and labels:

Preprocessed the frames to reduce noise.

- Resize
- Normalize
- Blur

Saved labels and frames of each video as numpy arrays.

```python
def preprocess_video(video_path, target_shape=(64, 64)):
    try:
        video_cap = cv2.VideoCapture(video_path)
        frame_count = int(video_cap.get(cv2.CAP_PROP_FRAME_COUNT))
        preprocessed_frames = []
        while True:
            success, frame = video_cap.read()
            if not success:
                break
            resized_frame = cv2.resize(frame, target_shape[::-1])
            blurred_frame = cv2.GaussianBlur(resized_frame, (5, 5), 0)
            normalized_frame = blurred_frame.astype(np.float32) / 255.0
            preprocessed_frames.append(normalized_frame)
        video_cap.release()
        return preprocessed_frames, frame_count
    except Exception as e:
        logger.error(f"Error preprocessing video: {e}")
        return None, None
```

```python
for video_folder, annotation_file, label_type in video_folders:
    with open(annotation_file, 'r') as file:
        lines = file.readlines()
    frames_list = []
    labels_list = []
    for line in tqdm(lines):
        parts = line.strip().split(',')
        vidname = parts[0]
        labels_str = ','.join(parts[1:])
        labels = [int(label) for label in labels_str[1:-1].split(',')]
        video_path = os.path.join(video_folder, f"{vidname}.mp4")
        preprocessed_frames, num_frames = preprocess_video(video_path)
        if preprocessed_frames is not None:
            frames_list.append(preprocessed_frames)
            labels_list.append(labels)
    frames = np.concatenate(frames_list, axis=0)
    labels = np.concatenate(labels_list, axis=0)
    print(f"Total number of {label_type} frames: {len(frames)}")
    print(f"Total number of {label_type} labels: {len(labels)}")
    np.save(os.path.join(output_folder, f"{label_type}_preprocessed_frames.npy"), frames)
    np.save(os.path.join(output_folder, f"{label_type}_labels.npy"), labels)
```

OUTPUT:

```
debugpy\launcher' '53236' '--' 'c:\Users\pavan\OneDrive\Desktop\miniproject\01datapreprocessing.py'
100%|                                                                                | 500/500 [01:55<00:00,  4.35it/s]
Total number of non_accident frames: 25000
Total number of non_accident labels: 25000
100%|                                                                                | 500/500 [01:42<00:00,  4.87it/s]
Total number of accident frames: 25000
Total number of accident labels: 25000
```

Extracting features:

Extracted features using Resnet50 which is trained with a dataset imagenet.

- Resnet50 is a pretrained model

```
model = ResNet50(weights='imagenet', include_top=False, pooling='avg')
def extract_spatial_features(frame):
    preprocessed_frame = preprocess_frame(frame)
    preprocessed_frame = np.expand_dims(preprocessed_frame, axis=0)
    spatial_features = model.predict(preprocessed_frame)
    return spatial_features
```

3.2 Training 3dcnn model

- Defined 3DCNN architecture using cov3d layers,MaxPooling layer,Flatten, Dense, Dropout layers

```
Click here to ask Blackbox to help you code faster
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, models
import os

output_folder = r"C:\Users\pavan\OneDrive\Desktop\miniproject\output"
accident_train_frames = np.load(os.path.join(output_folder, "accident_preprocessed_frames.npy"))
accident_train_labels = np.load(os.path.join(output_folder, "accident_labels.npy"))
non_accident_train_frames = np.load(os.path.join(output_folder, "non_accident_preprocessed_frames.npy"))
non_accident_train_labels = np.load(os.path.join(output_folder, "non_accident_labels.npy"))
train_frames = np.concatenate([accident_train_frames, non_accident_train_frames], axis=0)
train_labels = np.concatenate([accident_train_labels, non_accident_train_labels], axis=0)
train_frames, val_frames, train_labels, val_labels = train_test_split(train_frames, train_labels, test_size=0.2,
print(f"Number of training frames: {len(train_frames)}")
print(f"Number of training labels: {len(train_labels)}")
print(f"Number of validation frames: {len(val_frames)}")
print(f"Number of validation labels: {len(val_labels)}")
print(f"Shape of training frames: {train_frames.shape}")
train_frames = np.expand_dims(train_frames, axis=-1)
val_frames = np.expand_dims(val_frames, axis=-1)
```

```
model = models.Sequential([
    layers.Conv3D(32, kernel_size=(3, 3, 3), activation='relu', input_shape=train_frames.shape[1:], padding='same
    layers.MaxPooling3D(pool_size=(2, 2, 2), padding='same'),
    layers.Conv3D(64, kernel_size=(3, 3, 3), activation='relu', padding='same'),
    layers.MaxPooling3D(pool_size=(2, 2, 2), padding='same'),
    layers.Conv3D(128, kernel_size=(3, 3, 3), activation='relu', padding='same'),
    layers.MaxPooling3D(pool_size=(2, 2, 2), padding='same'),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(train_frames, train_labels, batch_size=32, epochs=10, validation_data=(val_frames, val_labels
test_loss, test_accuracy = model.evaluate(val_frames, val_labels)
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')
model_path = os.path.join(output_folder, "3d_cnn.h5")
model.save(model_path)
print("Model saved successfully at:", model_path)
```

3.3 LSTM Model Training:

Trained LSTM with the Time Distributed layers,LSTM layers.

```
model = Sequential([
    TimeDistributed(Conv2D(16, (3, 3), activation='relu', padding='same'), input_shape=(num_frames, height, width
    TimeDistributed(MaxPooling2D((2, 2), padding='same')),
    TimeDistributed(Conv2D(8, (3, 3), activation='relu', padding='same')),
    TimeDistributed(MaxPooling2D((2, 2), padding='same')),
    TimeDistributed(Conv2D(8, (3, 3), activation='relu', padding='same')),
    TimeDistributed(MaxPooling2D((2, 2), padding='same')),
    TimeDistributed(Flatten()),
    LSTM(32, return_sequences=True),
    LSTM(32),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_split=0.2)
```

ISOLATION FOREST ALGORITHM

```

if all_features and all_labels:
    all_features = np.concatenate(all_features, axis=0)
    all_labels = np.concatenate(all_labels, axis=0)
else:
    print("No spatial features and labels found. Please check the data paths.")

print("Shape of all_features:", all_features.shape)
print("Shape of all_labels:", all_labels.shape)
if all_features.shape[0] > 0 and all_labels.shape[0] > 0:
    X_train, X_test, y_train, y_test = train_test_split(all_features, all_labels, test_size=0.2, random_state=42)
else:
    print("Insufficient data for training and testing.")
X_train = X_train.reshape(X_train.shape[0], -1)
X_test = X_test.reshape(X_test.shape[0], -1)
if 'X_train' in locals() and X_train.shape[0] > 0 and 'X_test' in locals() and X_test.shape[0] > 0:
    model = IsolationForest(contamination=0.1)  # Adjust contamination parameter as needed
    model.fit(X_train)
    joblib.dump(model, model_save_path)
    print("Model saved successfully at:", model_save_path)
    y_pred = model.predict(X_test)
    y_pred_binary = np.where(y_pred == 1, 0, 1)
    accident_indices = np.where(y_test == 0)[0]
    non_accident_indices = np.where(y_test == 1)[0]
    accuracy_accident = accuracy_score(y_test[accident_indices], y_pred_binary[accident_indices])
    accuracy_non_accident = accuracy_score(y_test[non_accident_indices], y_pred_binary[non_accident_indices])
    print("Accuracy for Accident class:", accuracy_accident)
    print("Accuracy for Non-Accident class:", accuracy_non_accident)
```

3.4 Prediction of Accident

Preprocessed according to each model expected input shape and calculated threshold for each model.

```python
def preprocess_video_cnn(video_path, target_shape=(64, 64)):
    try:
        video_cap = cv2.VideoCapture(video_path)
        frame_count = int(video_cap.get(cv2.CAP_PROP_FRAME_COUNT))
        preprocessed_frames = []

        while True:
            success, frame = video_cap.read()
            if not success:
                break
            resized_frame = cv2.resize(frame, target_shape[::-1])
            blurred_frame = cv2.GaussianBlur(resized_frame, (5, 5), 0)
            normalized_frame = blurred_frame.astype(np.float32) / 255.0
            preprocessed_frames.append(normalized_frame)

        video_cap.release()
        preprocessed_frames = np.expand_dims(preprocessed_frames, axis=-1)
        return np.array(preprocessed_frames), frame_count

    except Exception as e:
        print(f"Error preprocessing video: {e}")
        return None, None
```

```python
def preprocess_video_resnet(video_path, target_shape=(224, 224)):
    try:
        video_cap = cv2.VideoCapture(video_path)
        frame_count = int(video_cap.get(cv2.CAP_PROP_FRAME_COUNT))
        preprocessed_frames = []

        while True:
            success, frame = video_cap.read()
            if not success:
                break
            resized_frame = cv2.resize(frame, target_shape[::-1])
            normalized_frame = resized_frame.astype(np.float32) / 255.0
            preprocessed_frames.append(normalized_frame)

        video_cap.release()
        preprocessed_frames = np.array(preprocessed_frames)
        preprocessed_frames = preprocess_input(preprocessed_frames)
        return preprocessed_frames, frame_count

    except Exception as e:
        print(f"Error preprocessing video: {e}")
        return None, None
```

```python
def preprocess_video_lstm(video_path, target_shape=(64, 64), num_frames=16):
    try:
        video_cap = cv2.VideoCapture(video_path)
        frame_rate = int(video_cap.get(cv2.CAP_PROP_FPS))
        frame_count = int(video_cap.get(cv2.CAP_PROP_FRAME_COUNT))
        target_frames = min(num_frames, frame_count)
        preprocessed_frames = []
        for _ in range(target_frames):
            success, frame = video_cap.read()
            if not success:
                break
            resized_frame = cv2.resize(frame, target_shape[::-1])
            blurred_frame = cv2.GaussianBlur(resized_frame, (5, 5), 0)
            normalized_frame = blurred_frame.astype(np.float32) / 255.0
            preprocessed_frames.append(normalized_frame)

        video_cap.release()
        while len(preprocessed_frames) < num_frames:
            preprocessed_frames.append(np.zeros(target_shape[::-1] + (3,)))

        preprocessed_frames = np.array(preprocessed_frames)
        preprocessed_frames = np.expand_dims(preprocessed_frames, axis=0)
        return preprocessed_frames, frame_count

    except Exception as e:
        print(f"Error preprocessing video: {e}")
        return None, None
```

```python
def test_video(video_path):
    preprocessed_frames_cnn, num_frames_cnn = preprocess_video_cnn(video_path)
    cnn_predictions = cnn_model.predict(preprocessed_frames_cnn)
    cnn_accident = np.any(cnn_predictions > 0.5)
    if cnn_accident :
        result_label.config(text="Accident Detected", fg="red", font=("Helvetica", 16, "bold"))
    else:

        preprocessed_frames_resnet, num_frames_resnet = preprocess_video_resnet(video_path)
        resnet_features = resnet_model.predict(preprocessed_frames_resnet)
        preprocessed_frames_lstm, num_frames_lstm = preprocess_video_lstm(video_path)
        lstm_prediction = lstm_model.predict(preprocessed_frames_lstm)
        resnet_predictions = iso_forest_model.predict(resnet_features)

        lstm_accident = lstm_prediction > 0.5
        resnet_accident = -1 in resnet_predictions
        if (lstm_accident and resnet_accident):
            result_label.config(text="Accident Detected", fg="red", font=("Helvetica", 16, "bold"))
        else:
            result_label.config(text="No Accident Detected", fg="green", font=("Helvetica", 16, "bold"))

```

CHAPTER 4

SNAPSHOTS

Accuracy of 3DCNN Model:

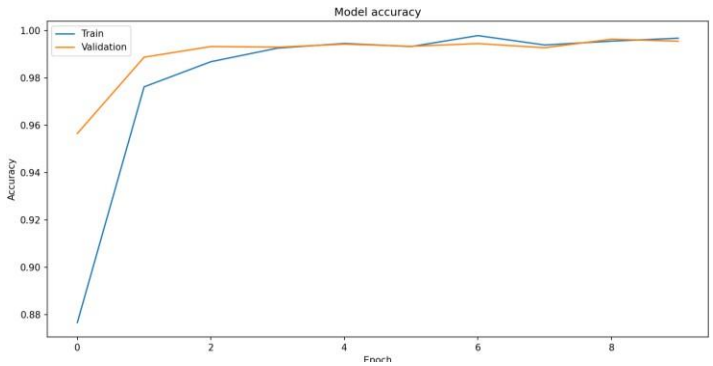
Achieved 96.82% accuracy

```
500/500 [==============================] - 163s 327ms/step - loss: 0.1033 - accuracy: 0.9606 - val_loss: 0.2840
 val_accuracy: 0.8535
Epoch 8/10
500/500 [==============================] - 168s 336ms/step - loss: 0.0849 - accuracy: 0.9686 - val_loss: 0.2625
 val_accuracy: 0.8733
Epoch 9/10
500/500 [==============================] - 168s 337ms/step - loss: 0.0764 - accuracy: 0.9722 - val_loss: 0.2955
 val_accuracy: 0.8705
Epoch 10/10
500/500 [==============================] - 169s 338ms/step - loss: 0.0607 - accuracy: 0.9769 - val_loss: 0.3487
 val_accuracy: 0.8805
625/625 [==============================] - 61s 97ms/step - loss: 0.1001 - accuracy: 0.9682
Test Accuracy: 0.9682499766349792
```

Accuracy of LSTM Model:

Achieved 99.48% accuracy

```
Epoch 4/10
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 31s 31ms/step - accuracy: 0.9944 - loss: 0.0200 - val_accuracy: 0.9934 - val_loss:
 0.0213
Epoch 5/10
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 32s 32ms/step - accuracy: 0.9936 - loss: 0.0192 - val_accuracy: 0.9958 - val_loss:
 0.0142
Epoch 6/10
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 34s 33ms/step - accuracy: 0.9965 - loss: 0.0125 - val_accuracy: 0.9951 - val_loss:
 0.0168
Epoch 7/10
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 33s 33ms/step - accuracy: 0.9963 - loss: 0.0116 - val_accuracy: 0.9958 - val_loss:
 0.0168
Epoch 8/10
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 31s 31ms/step - accuracy: 0.9940 - loss: 0.0193 - val_accuracy: 0.9937 - val_loss:
 0.0199
Epoch 9/10
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 24s 24ms/step - accuracy: 0.9978 - loss: 0.0072 - val_accuracy: 0.9975 - val_loss:
 0.0098
Epoch 10/10
1000/1000 ━━━━━━━━━━━━━━━━━━━━ 19s 19ms/step - accuracy: 0.9976 - loss: 0.0077 - val_accuracy: 0.9969 - val_loss:
 0.0165
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9948 - loss: 0.0152
Test Loss: 0.015865115448832512
Test Accuracy: 0.9948999881744385
```



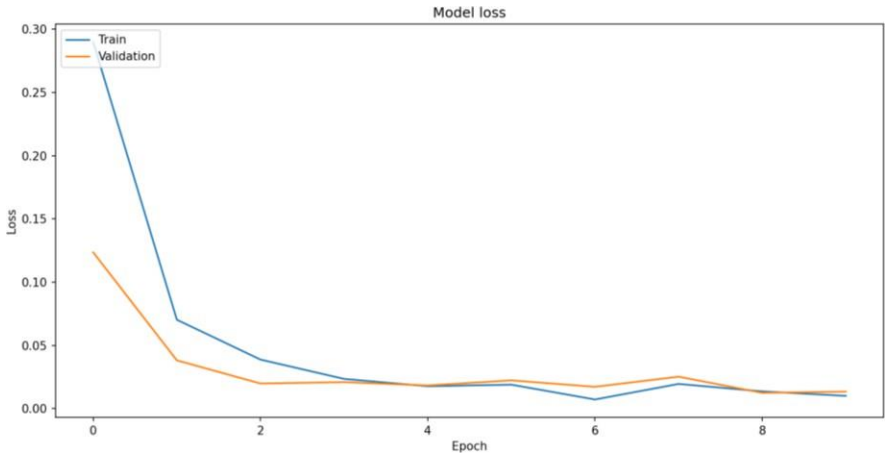


Fig 2.1 LSTM accuracy and loss

Mean pixel value using Box Plots for Outliers for LSTM :

Box plot for the anomalies for the given data

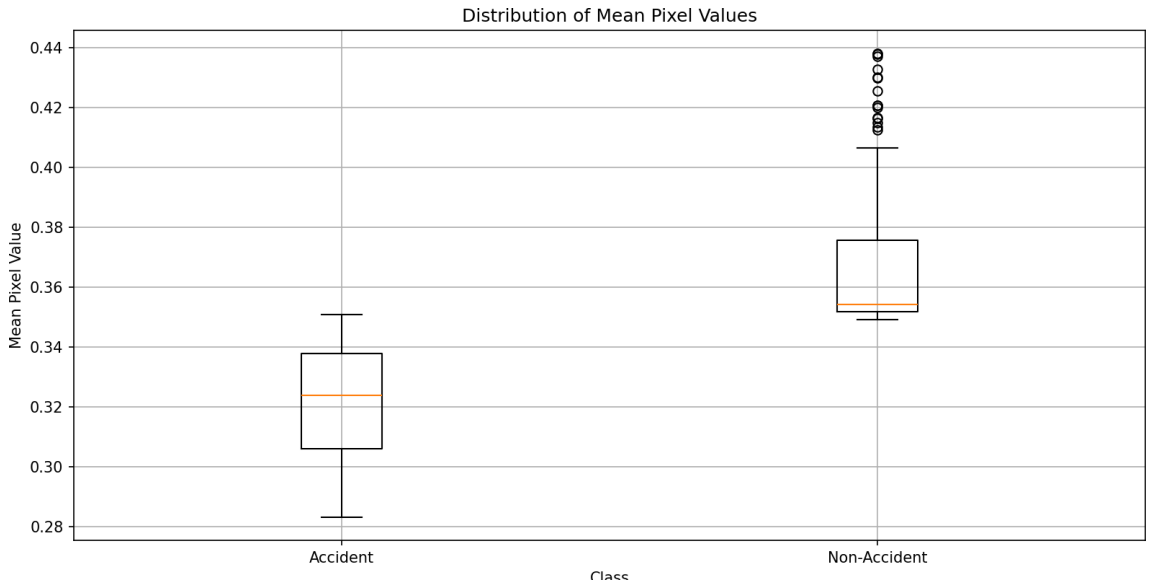


Fig 2.2 Distribution of Mean Pixel Values

CLASSIFICATION TABLE FOR ISOLATION FOREST:

Classification table for the isolation forest and achieved accuracy of 90% for the accident videos and 10% for the non-accident videos.

```
Shape of all_features: (50000, 1, 2048)
Shape of all_labels: (50000,)
Model saved successfully at: C:\Users\pavan\OneDrive\Deskto
h5
Accuracy for Accident class: 0.9064300530076055
Accuracy for Non-Accident class: 0.10287443267776097
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      8678
           1       0.14      0.10      0.12      1322

    accuracy                           0.80     10000
   macro avg       0.51      0.50      0.50     10000
weighted avg       0.77      0.80      0.79     10000
```

Fig 3.1 Classification table for I.S

Anomaly score distribution for the features

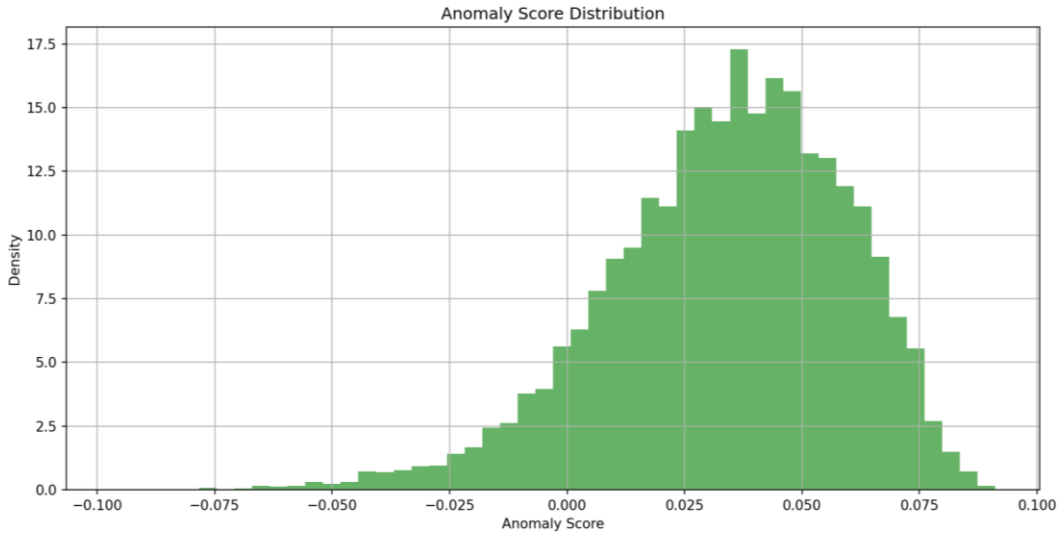


Fig 3.2 Anomaly score distribution for I.S

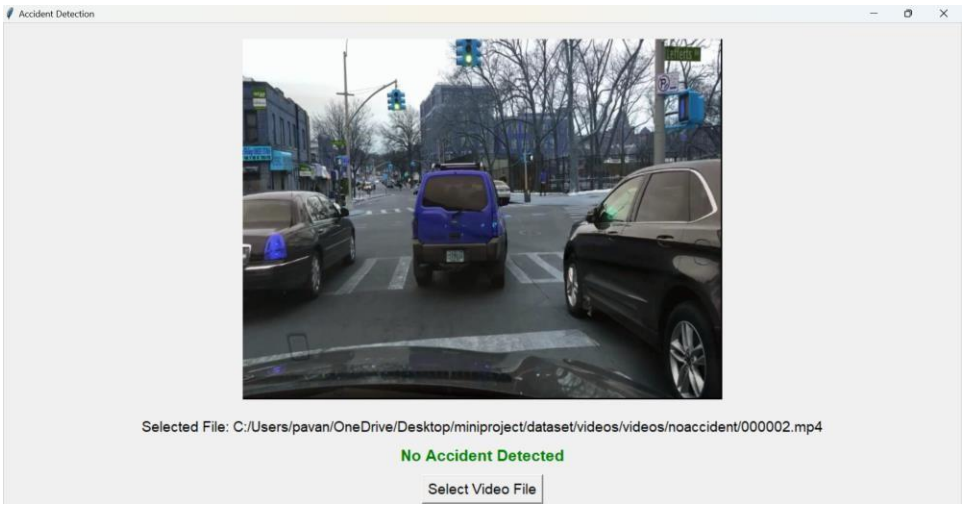
CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Result Analysis:

- Utilizes binary cross-entropy for binary classification, distinguishing between accident and non-accident frames.
- Uses sigmoid activation function during training for binary classification.
- Employs binary cross-entropy for anomaly detection.

5.2 Results:



CONCLUSION AND FUTURE PLANS

6.1 CONCLUSION :

In conclusion, the accident detection system developed through this project utilizes advanced machine learning techniques to effectively analyze video data and detect anomalies indicative of accidents. The system integrates multiple components, including a 3D CNN for video classification, a pretrained ResNet50 for extracting spatial features, an LSTM network for analyzing temporal sequences, and an Isolation Forest algorithm for anomaly detection.

Through extensive preprocessing of video data and training of the various models, the system demonstrates promising results in accurately identifying accidents within surveillance videos. By leveraging spatial and temporal information, the system achieves robust performance in distinguishing between normal and abnormal events, thus enabling prompt response and mitigation of potential harm.

The combination of these machine learning techniques offers scalability and adaptability, making the system suitable for real-world surveillance applications where timely accident detection is crucial for ensuring safety and security. Further refinements and optimizations could enhance the system's performance and applicability in various surveillance scenarios, contributing to improved accident prevention and response measures.

6.2 FUTURE PLAN:

For future development in accident detection, several avenues can be explored to enhance the system's effectiveness and expand its applicability:

Integrating accident detection systems with Internet of Things (IoT) devices and sensors can enhance their capabilities. Data from traffic cameras, GPS systems, vehicle sensors, and other sources can provide additional context for accident detection and improve accuracy.

Continuing research into advanced anomaly detection techniques, such as deep learning models like Generative Adversarial Networks (GANs) or reinforcement learning algorithms, can help identify subtle or complex accident scenarios that may be missed by traditional methods.

Integrating multiple data sources, such as video feeds, audio recordings, and environmental sensors, can provide a more comprehensive understanding of the accident scene and improve the accuracy of accident detection algorithms.

Collaborating with emergency services and local authorities can facilitate the integration of accident detection systems with existing emergency response systems, enabling faster and more coordinated responses to accidents.

REFERENCES

[1] W. Sultani, C. Chen, and M. Shah, ‘‘Real-world anomaly detection in surveillance videos,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.

[2] Y. Yao, X. Wang, M. Xu, Z. Pu, Y. Wang, E. Atkins, and D. J. Crandall, ‘‘DoTA: Unsupervised detection of traffic anomaly in driving videos,’’ IEEE Trans. Pattern Anal. Mach

[3] Z. Zhou, X. Dong, Z. Li, K. Yu, C. Ding, and Y. Yang, ‘‘Spatio-temporal feature encoding for traffic accident detection in VANET environment,’’ IEEE Trans. Intell. Transp.

[4] C. Lu, J. Shi, and J. Jia, ‘Abnormal event detection at 150 FPS in MATLAB,’ in Proc. IEEE

[5] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, ‘‘Learning temporal regularity in video sequences,” in Proc. IEEE

[6] J.-X. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li, ‘‘Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection,’ in Proc. IEEE

[8] J. Zhang, L. Qing, and J. Miao, ‘‘Temporal convolutional network with complementary inner bag loss for weakly supervised anomaly detection,’’ in Proc. IEEE

CHAPTER 7

APPENDEX A

BASE PAPER

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 October 2023, accepted 6 December 2023, date of publication 15 December 2023,
date of current version 20 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343410

RESEARCH ARTICLE

TempoLearn Network: Leveraging Spatio-Temporal Learning for Traffic Accident Detection

SOE SANDI HTUN[1], JI SANG PARK[2], KANG-WOO LEE[2], AND JI-HYEONG HAN[1]
[1]Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, South Korea
[2]Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea

Corresponding author: Ji-Hyeong Han (jhhan@seoultech.ac.kr)

This work was supported by the Electronics and Telecommunications Research Institute (ETRI) Grant funded by the Korean Government (Development of Technology to Reproduce and Analyze the Real-World Cities Into Digitalized Intelligent Urban Spaces) under Grant 22ZR1210.

ABSTRACT Recognizing traffic accident events in driving videos is a challenging task and has emerged as a crucial area of interest in autonomous driving applications in recent years. To ensure safe driving alongside human drivers and anticipation of their behaviors, methods to efficiently and accurately detect traffic accidents from a first-person viewpoint must be developed. This paper proposes a novel model, named the *TempoLearn* network, which leverages spatio-temporal learning to detect traffic accidents. The proposed approach incorporates temporal convolutions, given their effectiveness in identifying abnormalities, and a dilation factor for achieving large receptive fields. The TempoLearn network has two key components: accident localization, for predicting when the accident occurs in a video, and accident classification based on the localization results. To evaluate the performance of the proposed network, we conduct experiments using a traffic accident dashcam video benchmark dataset, i.e., the detection of traffic anomaly (DoTA) dataset, which is currently the largest and most complex traffic accident dataset. The proposed network achieves excellent performance on the DoTA dataset, and the accident localization score, measured in terms of AUC, is 16.5% higher than that of the existing state-of-the-art model. Moreover, we demonstrate the effectiveness of the TempoLearn network through experiments conducted on another benchmark dataset, i.e., the car crash dataset (CCD).

INDEX TERMS Traffic accident detection, temporal learning, segment proposal, transformer, dashcam videos.

I. INTRODUCTION

Self-driving vehicles have led to significant advancements in economy, mobility, and society at large. Such autonomous systems must be able to detect, classify, and assign appropriate weights to various road objects and use this information to identify the most reliable and safe path. Despite the rapid development in self-driving technologies, several problems remain to be addressed, for example, reducing the risks associated with the behaviors of human drivers and responding safely to a wide range of potential accident scenarios.

Recent computer vision studies have attempted to address these problems by training models to detect traffic accidents by applying deep learning approaches to footage captured by dashboard-mounted cameras, commonly known as dashcam. The existing traffic accident detection models focus on detecting when anomalous events begin and end in videos, implicitly considering the spatial context of the frames to identify the participants in the accident scene. General video-level action recognition approaches for traffic accident detection train models to classify the normal and abnormal video instances including accidents. Notably, these

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

techniques assume that all normal instances belong to the same category, and a video with any deviation is considered an abnormal instance. However, real-world dashcam videos that capture vehicle accidents are highly diverse and feature various combinations of semis, cars, motorcycles, mopeds, bicyclists, and pedestrians. Moreover, accidents typically occur suddenly and occupy only a brief part of a lengthy video. Considering these characteristics, several researchers have proposed weakly-supervised frameworks based on multiple instance learning (MIL) to detect distributed long-tailed traffic accidents in long videos [1], [2]. These techniques directly learn the anomaly class score using the high-level class labels for each video segment.

Furthermore, frame-level traffic accident detection approaches have been proposed, which focus on the object overlaps in key frames, learn the temporal information across frames using long short-term memory (LSTM) networks, and construct future frames using unsupervised and weakly supervised methods. For example, Yao et al. [3] proposed a method that combines bounding box trajectories and margin learning to increase the distance between abnormal and normal features. Similarly, Zhou et al. [4] developed a frame-level coarse-fine detection approach by adopting k-means clustering to cluster normal and abnormal frames. Overall, frame-level traffic accident detection methods typically focus on either appearance or object motion, which may lead to detection failure when the model encounters significant differences in lighting conditions or overlapping trajectory predictions of objects crossing the path. Thus, to ensure compatibility with self-driving systems, it is necessary to build a model that can learn both the spatial appearance of each frame and the temporal context, which corresponds to drastic changes in spatial features across frames.

Considering these aspects, this paper proposes an anchor-offset based segment-level detection approach for traffic accident detection from dashcam videos. The proposed approach introduces a temporal convolutional network that can not only learn temporal context but also detect accidents. Based on the temporal and spatial context-aware features extracted by the temporal learning module, the segment proposal generation (SPG) network detects traffic accident segments, and then a transformer performs accident classification. The key contributions of this study can be summarized as following:

1) A novel traffic accident detection model named *TempoLearn* is developed, which can provide information regarding the frames in which an accident starts and terminates, along with the traffic accident confidence scores and categories of the accident that occurs during the segment.
2) The TempoLearn model applies temporal convolution, which can facilitate temporal contextual information learning, allowing the model to better understand the context before and after the accident occurs.
3) The applicability of the proposed model is extensively examined on benchmark traffic-accident datasets, i.e., the detection of traffic anomaly (DoTA) dataset and car crash dataset (CCD).

The remaining paper is organized as follows: Section II reviews the existing research relevant to the proposed model. Section III introduces the TempoLearn model, which consists of the temporal convolution network, SPG network, and transformer, and detect traffic accidents from dashcam videos. Section IV describes the experimental environment and discusses results of application of the TempoLearn network. Section V presents the concluding remarks.

II. RELATED WORK

Compared with typical video classification tasks, traffic accident detection remains a challenging task in the field of computer vision owing to the sparse, diverse and ambiguous nature of the participants and situations involved in traffic accidents. In this section, we introduce the research related to traffic accident detection.

A. TRAFFIC ACCIDENT DETECTION

Because the events and dynamic scenes of traffic accidents are difficult to understand, even for human annotators, most of the research in this domain has been conducted based on unsupervised and weakly supervised algorithms. Unsupervised algorithms [5], [6] have been used to establish reconstruction approaches and to predict the future locations of potential traffic accident participants [3]. Zhong et al. [7] and Zhang et al. [8] considered the task as an MIL problem, and Zhang et al. [8] defined an inner bag loss focused on training both positive and negative videos. In contrast to the existing unsupervised and weakly supervised approaches, the proposed model is based on a fully supervised method that applies visual feature encoding and temporal context learning to learn from all of the training data. Zhou et al. [4] proposed a spatio-temporal traffic accident detection approach that clusters normal and abnormal features encoded with a multi-layer neural network. The approach involves extracting the motion feature vectors using histogram optical flow, and the self-representation constrained low-rank representation algorithm is used to cluster the potential traffic accident frames. Notably, because the coefficient matrix that satisfies the self-representation constraint has limited scope criteria for clustering the potential accident frames, this method is often ineffective for accident classification tasks involving a wide range of factors, such as traffic flow, weather conditions, road infrastructure, and historical accident data. Yao et al. [9] attempted to predict future object bounding boxes from past time steps using recurrent neural network (RNN) encoder-decoders, in which the standard deviation of predictions serves as the anomaly score. This model learns the position and motion of accident participants from location and odometry encoded features. However, similar to the work of Zhou et al., the RNN model [9] also requires precomputation

of the optical flow. In contrast, the proposed model uses an end-to-end trainable approach that eliminates the need to compute optical flow data to learn temporal information. The proposed model not only learns the spatial information of the frame, but also encodes features with pixel-level attention to highlight the notable factors related to the participants and accidents scene conditions. The temporal contextual learning module of the proposed model takes the encoded features as the input and yields sequential information of variations in the features over time during the accident duration. To the best of our knowledge, the proposed model represents the first successful attempt at realizing both accident detection and traffic accident classification.

B. TEMPORAL CONTEXT LEARNING AND VIDEO SEGMENTATION

Understanding the temporal context before and after the accident in a video provides crucial clues for determining the types of event that occur within the accident duration. Zhou et al. [10] demonstrated that context-aware feature encoding outperforms context-free networks by using bidirectional LSTM (Bi-LSTM) for frame-wise context feature extraction. However, passing the features in the LSTM in a frame-wise manner and concatenating them with ResNet features are computationally expensive processes. To overcome the disadvantage of the LSTM in encoding temporal context features, Chao et al. [11] addressed the task of contextual feature learning by doubling the dilation rate of convolution layers, allowing the receptive fields to cover the two segments before and after the anchor. In addition, Zhou et al. [12] obtained context information by encoding the frames using a self-attention mechanism, enabling the learning of potential dependencies among distant frames. Fang et al. [13] proposed a method to learn the motion and the context relation consistency within consecutive frames based on a generative adversarial network (GAN) [14]. This model predicts the visual scene context in driving scenarios and detects traffic accidents based on temporal frame consistency, temporal object location consistency, and the spatial-temporal relation consistency of road participants.

In anchor-offset based segment-level detection in video datasets, the input of the SPG layer is the feature map output by the convolutional neural network, which captures only the spatial information of the frames. However, for accurate localization of event boundaries, it is necessary to incorporate the temporal context information into the segment-level localization pipeline. To address temporal modeling tasks, a temporal convolutional network (TCN) was designed, which included causal and inflation convolution layers as well as residual modules [15].

Several studies have demonstrated that TCN-based models outperform RNN-based models in not only video segmentation tasks, but also for detecting anomalous events in sequential data. Unlike recurrent architectures such as RNNs, TCNs incorporate a backpropagation path that lies in a direction different from the temporal direction of the sequence. Consequently, TCNs can avoid the problems of exploding and vanishing gradients, which is a major issue for RNNs and led to the development of LSTM and GRU frameworks. Owing to their clarity and simplicity, TCNs [15] were highlighted as promising natural starting points and powerful toolkits for sequence modeling. Notably, in traffic accident detection tasks, accidents are sparse and occur only in short durations within long input sequence. Thus, LSTMs and GRUs may consume considerable memory to store partial results for multiple cell gates. In contrast, in a TCN, the filters are shared across a layer, and the backpropagation path depends on only the network depth. Therefore, in practice, TCNs are more suitable for accident/abnormality detection tasks than the sophisticated gated RNNs, which have significantly higher memory requirements. Lea et al. [16] captured the long-range temporal information between video frames by introducing layers of temporal convolutions. Specifically, the authors applied an encoder-decoder architecture with downsampling and upsampling using pooling operations for segmentation. Bai et al. [15] proposed the introduction of residual connections between dilated causal temporal convolution blocks to achieve longer effective memory and prevent information leakage to subsequent layers. Farha and Gall [17] introduced a multistage architecture with a large temporal receptive field and fewer parameters by eliminating the temporal downsampling operation. He and Zhao [18] applied a TCN for anomaly detection in time series. To encode the temporal context feature for video anomaly detection, Zhang et al. [8] used a traditional TCN as a temporal modeling module with temporal pooling and causal dilated convolutions. The method proposed in the current study leverages the advantages of TCNs in learning temporal contextual information, based on empirical evidence of their efficiency in traffic accident detection and their ability of effectively learning from large historical datasets. Similar to an existing framework [17], TempoLearn excludes the downsampling operation in the temporal contextual learning stage and instead expands the wide receptive field using dilation factors.

C. TRAFFIC ACCIDENT DATASET

In this study, DoTA dataset [3] and CCD [19] are used owing to their comprehensive spatio-temporal annotations and inclusion of diverse accident categories in different environmental conditions. With recent research attention being focused on accident detection in egocentric traffic videos, researcher have introduced large-scale datasets. For example, the street accident dataset [20] contains 620 video clips of on-road accidents captured from dashboard cameras. The dataset is annotated with object bounding boxes for each frame and a Boolean value indicating whether an accident occurred in the frame. The last 10 frames of each clip are annotated as anomalous. Although this dataset involves both spatial and temporal annotations of road scenes, it lacks

categorical information of the accidents. Fang et al. [21] proposed a large-scale dataset, DADA, in which 2,000 video sequences are annotated with accident participants and categories for driver attention prediction. The A3D dataset [9] is another large-scale dataset, which contains 1,500 driving videos with diverse accidents. moreover, the temporal information is annotated with start and end times.

In addition to the above described datasets, dashcam videos are utilized in other areas such as traffic risk assessment and identifying dangerous vehicles. The traffic risk assessment dataset [22] is designed for traffic risk assessment, which contains four risk levels annotations according to the possibility of traffic accidents by considering actual driving experience and scenario complexity. The GTACrash dataset [23] focuses on identifying dangerous vehicles. This dataset consists of 10,381 scenes and each scene contains 20 frames of images. Among these scenes, 3,661 depict non-accident scenarios, providing the normal flow of traffic and 7,720 scenes capturing accident scenes, portraying the moments of potential danger. Yoo and Han [24] proposed a traffic accident detection benchmark, which consists of 1,935 videos of traffic accidents captured by dashcams. This dataset is annotated 18 semantic labels related to the causes of accidents and 7 semantic labels related to the accident effects.

III. PROPOSED TEMPOLEARN NETWORK

This section describes the overall framework of the TempoLearn network. Fig. 1 shows the architecture of TempoLearn. The proposed model is composed of four parts: feature extraction and encoding, temporal contextual learning from the encoded features, SPG from the context-aware feature map, and transformer classifier for accident classification based on the proposed segments. The feature extraction and encoding module includes ResNet-101 [25] as the backbone of the visual feature extractor, along with spatial self-attention. The temporal context learning module operates on the encoded visual features to enhance the awareness of the temporal context. This step ensured that the features capture the temporal dependencies before the segment proposals are generated. The SPG module uses the features with temporal context as the input and outputs the event segment proposals. To classify the current segment proposal, element-wise multiplication of the temporal context features and segment proposal regions is performed to mask out the outer region of the segment proposal in the sequence. Then, the transformer classifies the accident classes. the following sections describe each component of TempoLearn.

A. FEATURE EXTRACTION AND ENCODING

For a video V with T frames $V = \{x_1, x_2, x_3, \ldots, x_T\}$, we extract the frame-level features using a two-dimensional (2D) ResNet-101 [25] that is pretrained on the ImageNet-1K dataset [26]. Although ImageNet is a large dataset, the egocentric viewpoints of dashcam traffic accident videos are inconsistent and divergent compared with the images from the ImageNet dataset. To adapt the model to this distinct problem domain, we remove the flattened classifier FC layer of the ResNet-101 model and finetune the last layer, i.e., layer 4 containing 2048 channels of convolutions, using the DoTA dataset.

Subsequently, the output feature map is encoded using self-attention on the pixel level as follows:

$$F_t = FC_{512}(ResNet_{101}(x_t)) \tag{1}$$

$$A_{ij} = \frac{exp(Q_i \cdot K_j^T)}{\sum_{j=1}^{N} exp(Q_i \cdot K_j^T)} \tag{2}$$

$$z_i = \sum_{j=1}^{N} A_{ij} \cdot V_j \tag{3}$$

$$z_t = Reshape(Concat(z_1, \ldots, z_N)) \tag{4}$$

where feature F_t is the primary feature map previously extracted from ResNet-101; i and j denote the location of the mean pixel; Q_i, K_j, and V_j are the query, key, and value vectors of the i^{th} pixel and j^{th} pixel; N is the number of pixels, i.e., $N = w \times h$; A_{ij} is the SoftMax attention score for Q_i according to K_j and V_j; z_i is the self-attention output for the i^{th} pixel; and z_t is the final self-attention output for F_t. Spatial self-attention is performed on the feature F_t, and it is then projected onto the query $Q = W_q F_t$, key $K = W_k F_t$, and value $V = W_v F_t$ vectors. All three projected matrices have the same spatial dimension as that of the feature $F_t \in \mathbb{R}^{w \times h \times d_{model}}$, where d_{model} denotes the output feature dimension of the model. For enhancing the sensitivity of feature representations to abnormal positions, we perform element-wise multiplication between $A \in \mathbb{R}^{N \times N}$ and $V \in \mathbb{R}^{N \times d_{model}}$ and reshape the results of z_t to $\mathbb{R}^{w \times h \times d_{model}}$.

B. TEMPORAL CONTEXT LEARNING

We denote the feature output from the feature extraction and encoding module as Z with T timesteps $Z = \{z_1, z_2, z_3, \ldots, z_T\}$. The dilated acausal TCN is applied for extracting the context-aware features. The dilated convolution is a type of convolution that the convolution kernel is expanded by inserting holes between kernel elements. Unlike the dilated causal convolutional layers of WaveNet [27], the temporal learning module in the proposed framework is adapted using dilated acausal convolutions to provide the network with both preceding and future context information of the traffic accident to facilitate detection. In the proposed network, we use a dilation factor with 2 powered values of the dilation layer with equal kernel size 3 and total $L = 4$ layers of the dilated convolution with residual connection is applied. Each dilated convolution layer with residual connections can be described as follows:

$$F(Z)' = Dropout(ReLU(f_1(k) *_d Z) * f_2(1)) \tag{5}$$

$$F(Z) = Z + F(Z)' \tag{6}$$

where $f_1(k)$ and $f_2(1)$ are convolution filters with kernel sizes k and 1, respectively; d is the dilation factor; and $*$ and $*_d$ represent the convolution and dilated convolution

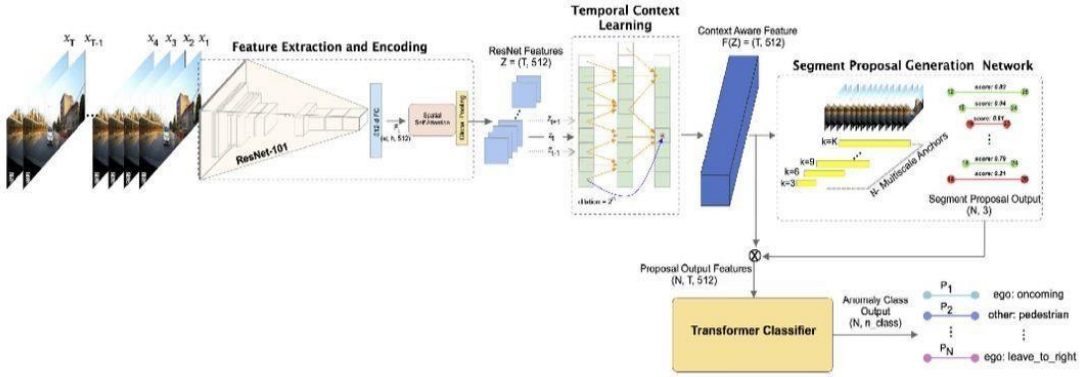


FIGURE 1. Overview of the proposed TempoLearn network architecture. The *Feature Extraction and Encoding* module encodes the visual features of each frame with self-attention. After $ResNet_{101}$, fully-connected (FC) layers *512-d FC* are introduced to reduce the $d_{model} = 2048$ dimension to $d_{model} = 512$. At the end of the *Feature Extraction and Encoding* stage, *global pooling* is performed on the *width* and *height* dimension of the output feature map of each frame to facilitate conversion from $z_t \in \mathbb{R}^{w \times h \times d_{model}}$ to $z_t \in \mathbb{R}^{d_{model}}$, such that for all frames, $Z \in \mathbb{R}^{T \times d_{model}}$. Each frame encoded with self-attention feature map is input to the *Temporal Context Learning* module for learning the temporal contextual feature before the segment proposals are generated. The *Segment Proposal Generation* module outputs N proposals that are trained on K explicit anchors with different kernel sizes k. The output proposals are composed of traffic accident start and end frame indices with the confidence score. The *Transformer Classifier* identifies the label of accident class for each proposal.

operations, respectively. The dilated convolution process based on aforementioned equations can be summarized as follows: The input Z is subjected to dilated convolution with filter $f_1(k)$ with a kernel size of 3 ($k = 3$), and then the output is subjected to the rectified linear unit (ReLU) activation function. One-dimensional (1D) convolution is introduced before the residual connection, with reference to [17].

By applying dilated convolutions instead of convolutions, the proposed TempoLearn network achieves extremely large receptive fields in few layers, which can cover the complete sequence:

$$r(L) = 2^{L+1} - 1 \tag{7}$$

where L is the number of dilated residual layers and $r(L)$ denotes the receptive field that covers the range of sequence. The output from temporal context learning module is a context aware feature $F(Z) \in \mathbb{R}^{T \times d_{model}}$ that is the same dimension with the input to the module.

C. SPG FOR ACCIDENT LOCALIZATION

The SPG module in the proposed TempoLearn network is inspired by ProcNets [10], which demonstrated state-of-the-art performance in generating long dense segment proposals. We adopt the concept of an anchor-offset mechanism with k kernel sizes for K explicit anchors. The proposal decoder in [10] is constructed by 1D temporal convolutions, and a stride factor of 1 is assigned for each kernel size, allowing the model to generate dense proposals. The SPG in the proposed TempoLearn network is more similar to that in [12]. In this framework, the stride factor is defined according to the kernel size, and the sequential prediction module of [10] is eliminated because the events in a video are typically sparse and not closely coupled. We modify the existing model [12] by splitting the regressor and classifier following the base convolution layer. Fig. 2 illustrates the SPG module in the proposed TempoLearn network.

The start frame index S_p and end frame index E_p of the proposed segment based on multiscale anchors are calculated as follows:

$$c_\beta = \frac{c_g - c_a}{w_a} \quad w_\beta = log(\frac{w_g}{w_a}) \tag{8}$$

$$c_p = c_a + \hat{c_\beta} w_a \quad w_p = w_a \exp(\hat{w_\beta}) \tag{9}$$

$$S_p = c_p - \frac{w_p}{2} \quad E_p = c_p + \frac{w_p}{2} \tag{10}$$

where c_g, c_β, and c_a are centers of the ground-truth, ground-truth offset, and anchor, respectively; w_g, w_β, and w_a are widths of the ground-truth, ground-truth offset, and anchor, respectively; $\hat{c_\beta}$ and $\hat{w_\beta}$ are the predicted offset center and width, respectively, which are the outputs of the SPG module; and c_p and w_p are the center and width of the proposed segment, respectively. The SPG module inputs both the context-aware feature map $F(Z)$ output by the temporal contextual learning module and predefined anchors. The predefined anchor width w_a and anchor center c_a are parameterized to obatin the target offset center c_β and width w_β, based on the ground-truth center c_g and width w_g in Equation 8. Then, we decode those offsets at the end of the module to determine the center and width of the proposed segment, i.e., c_p and w_p, respectively, based on the predicted offset center $\hat{c_\beta}$ and width $\hat{w_\beta}$ using Equation 9. To pass the proposed segment to the classifier in the next step, we simply derive the start frame index S_p and end frame index E_p of the proposed segment, as indicated in Equation 10.

D. TRANSFORMER CLASSIFIER

To classify the proposed segment into the traffic accident categories, we apply an existing transformer model based

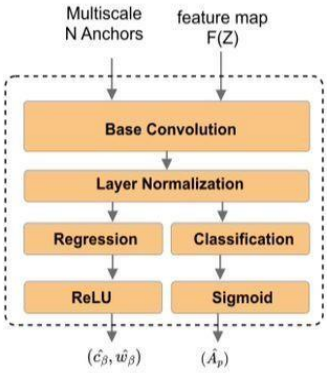


FIGURE 2. Segment proposal generation module in TempoLearn network. $\hat{c_\beta}$ and $\hat{w_\beta}$ are the predicted offset center and width, respectively, and $\hat{A_p}$ is the confidence score of the proposed segment.

on multi-head scaled-dot product attention mechanism [28], which achieved state-of-the-art performance on various tasks. The original encoder-decoder based transformer model [28] was designed for the machine translation task. This model applied self-attention mechanisms to capture relationships between different positions within a single sequence without using sequence aligned RNNs or convolutions. In the TempoLearn network, we adapt the transformer for classification tasks. In general, the decoder is responsible for predicting words at each time-step until a complete sentence is generated. However, in the task considered in this study, only a single output is needed for classification instead of a sequential output. Thus, we apply only the encoder block of the transformer.

The transformer classifier in TempoLearn inputs both the context-aware feature $F(Z)$ and SPG output (A_p, S_p, E_p) and performs the element-wise multiplication of feature $F(Z)$ according to the proposal S_p and E_p to obtain only the proposed segment features. With N proposed segment features $F(P) \in \mathbb{R}^{N \times T \times d_{model}}$, the operation of the transformer classifier can be expressed as follows:

$$F(P) = (Sp, Ep) \otimes F(Z) \tag{11}$$
$$F(P) = PE(F(P)) + F(P) \tag{12}$$
$$F(E) = Norm(MHA(F(P), F(P), F(P)) + F(P)) \tag{13}$$
$$C = Softmax(FC(F(E)) \tag{14}$$

where PE represents positional encoding, MHA indicates for multi-head self-attention, and FC represents a fully connected layer. After the proposed segment feature $F(P)$ is achieved using Equation 11, positional encoding is applied to $F(P)$ to encode the frame position sequence. Subsequently, the positional encoded feature is input to the multi-head self-attention mechanism and FC layer. The final *softmax* activation function generates the probability for accident classes. The output can be expressed as $C \in \mathbb{R}^{N \times n\_class}$ with respect to the number of traffic accident classes n_class for N segment proposals.

The multi-head self-attention operation for the encoder can be expressed as follows:

$$SA(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_{model}}})V \tag{15}$$
$$MHA(Q, K, V) = Concat(head_1, \ldots, head_H)W^H \tag{16}$$
$$head_i = SA(QW_i^Q, KW_i^K, VW_i^V) \tag{17}$$
$$PE(pos, 2_i) = sin(pos/10000^{2i/d_{model}}) \tag{18}$$
$$PE(pos, 2_{i+1}) = cos(pos/10000^{2i/d_{model}}) \tag{19}$$

where W_i^Q, W_i^K, and W_i^V are the trainable weights of each independent head of $i = \{1, 2, \ldots, H\}$ for query, key, and value, respectively; and W^H is the trainable weight to aggregate multi-head attention. For positional encoding, the same operation as that in the original research [28] is adopted, which is the best fit for the relationship between the input sequence position pos and model dimension d_{model}.

E. MODEL LEARNING

The loss calculation for TempoLearn network training consists of three parts: the binary cross entropy loss L_{cls} for learning the segment proposal confidence score, regression loss L_{reg} for learning the segment proposal boundary offset, and loss L_c for classifying the accident categories. The final loss L is a combination of these three losses, defined as follows:

$$L_{total} = \alpha_1 L_{cls} + \alpha_2 L_{reg} + \alpha_3 L_c \tag{20}$$
$$L_{cls} = BCE(\hat{A_p}, A_p) \tag{21}$$
$$L_{reg} = smooth_{L1}(\hat{c_\beta}, c_\beta) + smooth_{L1}(\hat{w_\beta}, w_\beta) \tag{22}$$
$$L_c = CE(\hat{C_p}, C_p) \tag{23}$$

where α_1, α_2, and α_3 are model hyper-parameters that determine the contribution of each loss. A_p is the ground-truth accident category label of the proposed segment, which indicates whether the proposal contains a traffic accident event; thus, $A_p \in [0, 1]$. For A_p, c_β and w_β denote the ground-truth offset center and length, respectively. $smooth_{L1}$ is the smooth L1 loss from [29], BCE denotes binary cross entropy, and CE represents the traditional supervised cross entropy loss.

The proposed segments are considered positive if they have an overlap of more than 70% with any ground-truth segment and negative if they have an overlap of less than 30% with all ground-truth segments. During the inference stage, redundancy in the overlapping segment proposals is reduced through non-maximum suppression (NMS) [30] on the proposal regions based on their confidence scores. To this end, a fixed intersection of union (IoU) threshold of 0.9 is used during testing. In the traiing phase, for each mini-batch, we randomly select 10 segment proposals from both positive and negative samples, each of which corresponds to one ground-truth segment.

IV. EXPERIMENTS AND ANALYSIS

This section describes the evaluation of the proposed method using various metrics. Section IV-A describes the three benchmark datasets used in the experiments with TempoLearn. Section IV-B presents details of the environment setup and implementation. Sections IV-C and IV-D discuss the performance evaluation results based on three metrics: the area under the ROC curve (AUC) score for traffic accident localization which determines the accident frames from all the frames of a dashcam video, top-1 accuracy for traffic accident classification, and mean average precision (mAP) for both traffic accident localization and classification. Section IV-E describes the validation of the proposed approach over another benchmark dataset and Section IV-F discusses the efficiency of the proposed approach.

A. DATASETS

To evaluate the performance of the proposed framework in accident detection tasks, we conducted experiments on two benchmarks dataset: DoTA [3] and CCD [19]. The other dashcam video datasets for traffic accident detection are summarized in Table 1. The details of DoTA and CCD are presented in the following text.

1) DOTA

The DoTA dataset is a large-scale video benchmark dataset for traffic accident detection, which consists of videos collected from dashcams. To the best of our knowledge, DoTA is the first traffic accident video dataset with extensive spatio-temporal annotations and 18 traffic accident categories. Table 2 presents nine unique traffic accident categories observed from both ego and non-ego viewpoints. The ego refers to accidents involving the ego-vehicle equipped with a dashcam. In the non-ego (marked with *), accidents do not involve the ego-vehicle and they are observed by a dashcam of the ego-vehicle. Each video is annotated with temporal labels and anomalous object bounding box tracklets that cover the duration of the traffic accident.

2) CAR CRASH DATASET

The CCD consists of dashcam videos captured in diverse environmental conditions. The dataset contains 4,500 videos, with 1,500 positive videos (accident videos) and 3,000 negative videos (normal driving car videos). Each video is extracted at a rate of 10 fps and contains 50 frames, i.e., the duration of every video is 5 seconds. For AUC score evaluation, the start and end frame annotations are required to calculate the IoU of the accident segment in the videos. Thus, we conducted experiments using the 1,500 positive videos, for which the the annotations for the accident start and end frame indices are available.

B. IMPLEMENTATION

The proposed method was implemented using PyTorch [31], and all the training and testing experiments were conducted on an NVIDIA GeForce RTX 3090 GPU with 24 GB memory. The ResNet-101 feature output has a channel dimension of 2,048, which was reduced to 512 by passing the features through the FC embedding layer. Identical channel dimensions were used for both the temporal context learning module and segment proposal network module. The hyperparameter α in the loss calculation was set as 1, 10 and 0.25 for α_1, α_2 and α_3, respectively for all benchmark datasets used in the study. During the inference stage, we generated 100 proposals for all benchmark datasets to evaluate the accident detection quality of the proposed network.

For the DoTA dataset, N explicit anchor widths for the SPG module were identified as [1, 2, 3, 4, 5, 7, 9, 11, 15, 21, 29, 41, 57, 71, 111, 161, 211, 251] through experimental validation. To ensure a fair comparison with the experiments conducted in the original research of the DoTA dataset, we excluded videos with unknown category, resulting in 4,585 videos. These data were partitioned into 80% for training and 20% for testing. The network was trained using a learning rate of 0.001 and batch size of 10. The stochastic gradient optimizer (SGD) [32] was used to optimize the network to ensure that it could reach the optimal point within a maximum of 50 epochs.

The configuration for the CCD experiments was similar to that for the DoTA experiments. Specifically, the learning rate, batch size, optimizer, and channel dimensions were the same as those for the DoTA experiments, but the N anchor widths were set as [1, 2, 3, 4, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 35, 39, 41, 45, 49]. The positive videos were split into training (80%) and testing (20%) sets, using the same ratio as that for DoTA experiments.

TABLE 1. Traffic accident dashcam video datasets.

Dataset	#videos	#frames	Annotations
StreetAccident [20]	620	62,000 (20fps)	temporal
CCD [19]	1,500	75,000 (10fps)	temporal
A3D [20]	1,500	128,175 (10fps)	temporal
DADA [21]	2,000	648,476 (30fps)	temporal, spatial(eye-gaze)
DoTA [3]	4,677	731,932 (10fps)	temporal, spatial(tracklets), accident categories

C. EVALUATION METRICS

To perform a comprehensive evaluation, the proposed TempoLearn network was assessed considering three metrics, i.e., AUC for accident localization; top-1 accuracy for accident classification; and mAP for accident detection, encompassing both accident localization and classification tasks.

For accident classification, we used the top-1 accuracy to assess the ability of the TempoLearn network to correctly classify the 18 categories of accidents in the DoTA dataset. Given that segment-level classification models

TABLE 2. Traffic accident categories in the DoTA dataset.

ID	Short	Accident 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

cannot observe the complete video sequence, this task is more challenging than traditional action recognition, in which the models learn from the entire sequential data. As mentioned in Section IV-B, we generated 100 proposals for each video during the inference stage. Thus, to ensure a fair comparison with other state-of-the-art video action recognition models over the DoTA dataset, we calculated the top-1 accuracy score based on only the true positive proposals, as outlined in Algorithm 1.

For the accident localization task, the quality of the output proposals was evaluated using the AUC score. For accident detection, encompassing both accident localization and classification, the accuracy of accident detection for each accident category was evaluated using mAP score.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{24}$$

$$AP = \frac{1}{U}\sum_{j=1}^{U} P_{interpo}(r_j) \tag{25}$$

where N is the number of accident classes; AP_i is the average precision of the i^{th} class; r_j represents the recall for the j^{th} IoU thresholds (tIoU); and U is the number of tIoUs. AP_i is calculated using Equation 25 with different tIoUs. For example, if tIoU ranges from 0.5 to 0.95 with a step of 0.05, then U is 10. $P_{interpo}$ indicates the interpolation to the maximum precision corresponding to recall r_j.

D. COMPARISON WITH STATE-OF-THE-ART MODELS

The results of accident localization, classification, and detection on the DoTA validation set are summarized in Tables 3, 4, and 5, respectively. Table 3 indicates that the proposed TempoLearn network achieves the highest AUC score among the state-of-the-art supervised methods. Among the unsupervised approaches on the DoTA dataset, TAD, TAD+ML, and Ensemble, which are segment-level detection approaches based on localization, can more accurately localize accidents than ConvAE, ConvLSTMAE, AnoPred, AnoPred + Mask, LSTM, and Encoder-Decoder, which are frame-level detection approaches based on the construction error. These results highlight that when using unsupervised models on the DoTA dataset, the introduction of optical flow with segment-level-based detection can yield superior localization results. The proposed TempoLearn network combines the advantages of supervised learning with temporal-contextual-learning-based segment-level detection. In particular, the temporal context learning module in TempoLearn maximizes the benefits of accident localization

Algorithm 1 Calculation of the Top-1 Accuracy for Accident Classification

P : proposal segment list of video V
S_g : accident start frame of ground truth of V
E_g : accident end frame of ground truth of V
C_g : accident class of ground truth of V

S_p : predicted accident start frame of proposal p of V
E_p : predicted accident end frame of p of V
C_p : predicted accident class of p of V
A_p : predicted accident confidence score of p of V
pos_thres : positive threshold
segment_iou : function to calculate IoU between segments

```
correct = 0
total = 0
for p=1 to P do:
    tIoU ← segment_iou([S_g, E_g], [S_p, E_p])
    if (tIoU > pos_thres)and(A_p > pos_thres) then:
        total ← total + 1
        if C_g == C_p then:
            correct ← correct + 1
top1 ← correct / total
return top1
```

TABLE 3. AUC scores for the accident localization task of the DoTA dataset.

Method	Type	Input	AUC
ConvAE [6]	Unsupervised	Gray	64.3
ConvAE [6]		Flow	66.3
ConvLSTMAE [33]		Gray	53.8
ConvLSTMAE [33]		Flow	62.5
AnoPred [34]		RGB	67.5
AnoPred+Mask [3]		Masked RGB	64.8
TAD [9]		Box+Flow	69.2
TAD+ML [3]		Box+Flow	69.7
Ensemble [3]		RGB+Box+Flow	73.0
FC [3]	Supervised	RGB	61.7
LSTM [35]		RGB	63.7
EncoderDecoder [36]		RGB	73.6
TRN [37]		RGB	78.0
VANET [4]		RGB+Flow	79.3
Proposed Approach		RGB	**92.4**

for the SPG module, resulting in the highest AUC score with only RGB input.

Table 4 indicates that the proposed TempoLearn outperforms the existing models in terms of not only the average accuracy but also the per-class accuracy. To demonstrate that TempoLearn can distinguish confusing accident classes, the confusion matrix of the classification results is presented in Fig. 3. The confusion matrix shows that TempoLearn can successfully classify similar accident categories, although it exhibits confusion among the TC, TC*, OC and OO* classes. One possible explanation is the abundance of examples of these four categories in DoTA and the similarity among them.

Table 5 shows that TempoLearn achieves the highest mAP among the compared methods. Notably, although Tables 3 and 4 separately present the quantitative results for accident localization and classification, the mAP was calculated based on both the quality of segment proposals and accuracy of classification of these proposals, as indicated in Equations 24 and 25. The experiments pertaining to Table 5 were conducted with reference to the online action detection experiment table in [3], in which the FC method was trained on a three-layer FC network, and the LSTM approach was trained on a one-layer LSTM classifier for sequential image classification. The results demonstrate that our robust accident detection proposals, learned using the spatio-temporal approach, lead to superior detection than that achieved using conventional classification techniques. Although TempoLearn exhibits lower mAP scores for certain accident classes, such as AH, LA, and TC, it achieves higher mAP scores for the most difficult accident categories, such as ST, ST*, VP, VP*, VO, VO*, and OC*.

Fig. 4 presents a qualitative comparison of the results obtained using the proposed approach and TRN [37], which achieves the high mAP score for accident detection (Table 5) on the DoTA dataset. The gray segments of the bars represent "background" (or "normal") pertaining to the ground truth data, whereas the orange bars represent anomalous segments of the video along with the corresponding accident categories. The green bars indicate the detected accident segments by the proposed model and TRN, and the predicted confidence score and accident class are presented on the bar. The top two rows correspond to ego-involved accidents and the 3rd row corresponds to a non-ego out-of-control accident. In terms of segment localization, TempoLearn detects the accident segment that is the closest to the ground truth. However, similar to other existing models listed in Table 5, TempoLearn finds it challenging to accurately classify the differences between the AH, LA, and TC accident categories. Overall, the following conclusions can be derived: 1) Although the TRN model can provide an indication of the timing and type of accident that may have occurred, it cannot offer precise localization information. Detailed localization, such as the identification of the exact location or region where the accident occurred, requires additional contextual understanding. In this regard, TempoLearn can leverage temporal contextual information, allowing the model to accurately localize of the accident segment. 2) As shown in Fig. 4, the LA accident category is visually similar to the typical AH and TC accident categories.

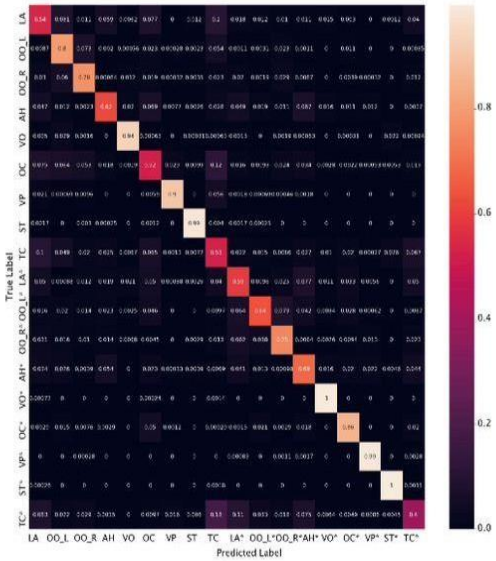


FIGURE 3. Confusion matrix for the accident classification task on the DoTA dataset. Traffic accident class OO is presented separately as OO-L and OO-R, corresponding to "out-of-control" and "leaving the roadway to the left or right" classes, respectively. OO-L* and OO-R* denote the same traffic accident category for the non-ego accident class.

FIGURE 4. Qualitative results of accident detection on the DoTA datset.

TABLE 4. Top-1 accuracy for accident classification on the DoTA dataset.

Method	ST	AH	LA	OC	TC	VP	VO	OO	ST*	AH*	LA*	OC*	TC*	VP*	VO*	OO*	AVG
TSN	18.2	67.2	52.9	53.8	71.0	0.0	0.0	61.6	0.0	14.7	25.3	6.7	48.1	9.5	0.0	53.4	30.2
C3D	25.5	61.8	43.9	47.8	57.9	3.3	4.4	52.9	1.2	18.4	36.0	6.7	55.9	8.6	6.0	33.2	29.0
I3D	10.0	62.4	45.8	45.8	62.2	2.8	6.9	66.6	2.4	28.1	24.5	4.7	60.3	9.5	5.0	37.6	29.7
R3D	0.0	56.5	49.6	49.8	66.6	4.4	6.2	47.7	1.8	17.6	32.2	1.0	48.3	15.2	6.5	48.0	28.2
MC3D	6.4	62.9	40.1	**57.7**	64.5	16.7	0.0	61.5	2.4	18.1	20.2	4.0	62.2	4.8	6.5	45.6	29.6
R(2+1)D	4.5	64.7	42.8	47.6	**68.7**	25.6	5.6	64.4	9.4	14.3	24.3	2.3	**64.7**	9.5	0.0	47.8	31.0
SlowFast	0.0	**70.0**	46.0	48.9	67.2	5.6	13.1	68.3	5.9	24.9	37.2	3.3	64.0	0.0	0.0	41.3	31.0
Our Approach	**98.8**	62.2	**53.9**	52.1	52.8	**90.2**	**93.5**	**78.5**	**99.7**	**68.6**	**59.2**	**85.7**	39.7	**99.3**	**99.7**	**69.2**	**75.2**

TABLE 5. *m*AP scores for accident detection on the DoTA dataset.

Method	ST	AH	LA	OC	TC	VP	VO	OO	ST*	AH*	LA*	OC*	TC*	VP*	VO*	OO*	mAP
FC	2.5	13.9	10.6	6.2	16.3	0.8	1.2	21.0	0.6	2.9	3.0	0.6	8.0	1.2	0.7	7.6	9.9
LSTM	0.6	19.9	15.1	9.2	25.3	2.4	0.6	34.3	0.6	3.8	5.0	1.5	11.0	1.2	0.5	13.3	12.9
Encoder-Decoder	0.5	20.1	15.6	10.4	28.1	2.9	0.7	**39.9**	0.8	3.7	7.4	2.5	14.7	1.2	0.5	13.2	14.5
TRN	1.0	**22.8**	**20.6**	15.5	**30.0**	1.5	0.7	32.3	0.7	4.0	10.2	2.9	**17.0**	1.2	0.7	**13.8**	15.3
Our Approach	**27.8**	11.8	12.1	**15.9**	16.4	**16.9**	**16.6**	28.9	**21.7**	**6.8**	**11.8**	**21.2**	12.8	**20.1**	**21.4**	10.1	**16.5**

E. APPLICABILITY TO TRAFFIC ACCIDENT DATASETS

We evaluated the applicability of the proposed model on another dashcam-based traffic accident video dataset, i.e., CCD. Unlike the DoTA dataset, CCD contains videos with 50 frames capture at a frame rate of 10 fps. The current version of CCD provides temporal annotations and environmental attributes for the videos but lacks information regarding the types of traffic accidents. Thus, only accident localization experiments were performed. Specifically, two experiments were conducted, in which the TempoLearn network was trained on the CCD with and without initialization of the pretrained weights on the DoTA dataset. All the hyperparameters were the same, except for the scale of explicit anchors, which were defined considering the average length of videos in the CCD. Fig. 5 shows the AUC scores for the validation set of the CCD for every epoch. The best performance was achieved with $AUC = 89.51$, when the model was initialized with pretrained weights.

F. EFFICIENCY OF THE PROPOSED MODEL

We performed an ablation study to evaluate the performance of the proposed model in two configurations. The results are summarized in Table 6. First, we omitted the temporal context learning module from the model to clarify the importance of temporal contextual understanding in traffic accident detection. We evaluated the performance of the proposed model trained with dense samples including N anchors with different k values in the SPG module. For this experiment, the N anchor widths were set as [1, 2, 3, 4, 5, 7, 9, 11, 15, 17, 19, 21, 25, 29, 35, 41, 57, 60, 71, 80, 90, 111, 161, 211, 251] to enable accurate detection of the accident segment boundaries. The number of trainable parameters and processing time during the inference stage were measured in millions and seconds, respectively. The mAP score was evaluated using the same technique described in Section IV-C. The approach with dense proposal generation achieves the highest mAP of 16.9, and the average inference time for each video is 5.37 seconds. Although the inference time is similar to that of the model without a temporal context learning module, the mAP score indicates that the performance of the latter is deteriorated. This analysis highlights the importance of context-aware feature encoding in accident detection, as it can help capture valuable signals for determining the types of events occurring within the accident duration.

TABLE 6. Efficiency of the proposed model in experiments on the DoTA dataset.

Method	♯Param	mAP	Inference Time(sec)
TempoLearn	42M	16.5	5.21
W/O context Learning	38M	10.4	4.68
With dense proposal generation	42M	16.9	5.37

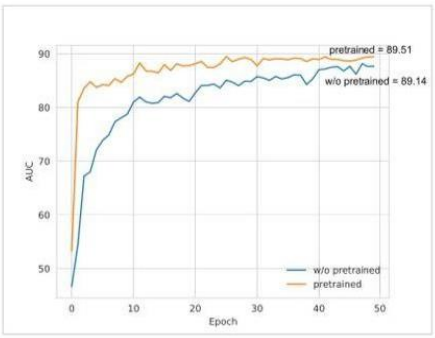


FIGURE 5. AUC score for accident localization on the validation set of the CCD.

V. CONCLUSION

This paper proposes an accident detection model named TempoLearn, which is fully-supervised and based on a segment-level traffic accident detection approach. The

TempoLearn network has three key components: temporal context learning, SPG, and transformer classifier. Thus, the TempoLearn network can predict the temporal positions of traffic accidents in a video along with the confidence scores and identify the type of accident that occurs during the detected segment. By incorporating the wide receptive fields based on temporal convolutions, the TempoLearn network can capture features containing temporal context information, prior to the generation of the traffic accident segment proposals. Owing to these abilities, TempoLearn outperforms the state-of-the-art frameworks in detecting long-tailed, distributed, and heterogeneous traffic accidents. Moreover, to classify different categories of traffic accidents based on the segments, we apply the transformer as a classifier, which outperforms the existing strategies in terms of the top-1 accuracy and mAP score.

In future work, the proposed approach can be extended to provide per-object accident scores by incorporating object features in the learning process. This extension would enable the application of the TempoLearn network to understand traffic scenes for clarifying the relationships among objects involved in the detection accident segment. In this manner, the proposed model would be extremely beneficial for autonomous driving systems.

REFERENCES

[1] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6479–6488.

[2] Y. Zhu and S. D. Newsam, "Motion-aware feature for improved video anomaly detection," 2019, *arXiv:1907.10211*.

[3] Y. Yao, X. Wang, M. Xu, Z. Pu, Y. Wang, E. Atkins, and D. J. Crandall, "DoTA: Unsupervised detection of traffic anomaly in driving videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 444–459, Jan. 2023.

[4] Z. Zhou, X. Dong, Z. Li, K. Yu, C. Ding, and Y. Yang, "Spatio-temporal feature encoding for traffic accident detection in VANET environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19772–19781, Oct. 2022.

[5] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 FPS in MATLAB," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2720–2727.

[6] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 733–742.

[7] J.-X. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li, "Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1237–1246.

[8] J. Zhang, L. Qing, and J. Miao, "Temporal convolutional network with complementary inner bag loss for weakly supervised anomaly detection," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 4030–4034.

[9] Y. Yao, M. Xu, Y. Wang, D. J. Crandall, and E. M. Atkins, "Unsupervised traffic accident detection in first-person videos," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 273–280.

[10] L. Zhou, C. Xu, and J. J. Corso, "Towards automatic learning of procedures from web instructional videos," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 7590–7598.

[11] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster R-CNN architecture for temporal action localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1130–1139.

[12] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8739–8748.

[13] J. Fang, J. Qiao, J. Bai, H. Yu, and J. Xue, "Traffic accident detection via self-supervised consistency learning in driving scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9601–9614, Jul. 2022.

[14] I. Goodfellow, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[15] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1907.10211*.

[16] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1003–1012.

[17] Y. A. Farha and J. Gall, "MS-TCN: Multi-stage temporal convolutional network for action segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3570–3579.

[18] Y. He and J. Zhao, "Temporal convolutional networks for anomaly detection in time series," *J. Phys., Conf. Ser.*, vol. 1213, no. 4, Jun. 2019, Art. no. 042050.

[19] W. Bao, Q. Yu, and Y. Kong, "Uncertainty-based traffic accident anticipation with spatio-temporal relational learning," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 2682–2690.

[20] F.-H. Chan, Y.-T. Chen, Y. Xiang, and M. Sun, "Anticipating accidents in dashcam videos," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 136–153.

[21] J. Fang, D. Yan, J. Qiao, J. Xue, and H. Yu, "DADA: Driver attention prediction in driving accident scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4959–4971, Jun. 2022.

[22] C. Liu, Z. Li, F. Chang, S. Li, and J. Xie, "Temporal shift and spatial attention-based two-stream network for traffic risk assessment," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 12518–12530, Aug. 2022.

[23] H. Kim, K. Lee, G. Hwang, and C. Suh, "Crash to not crash: Learn to identify dangerous vehicles using a simulator," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 978–985.

[24] T. You and B. Han, "Traffic accident benchmark for causality recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 540–556.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[27] A. van den Oord, "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.

[28] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, "Attention is all you need in speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 21–25.

[29] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[30] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2006, pp. 850–855.

[31] A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035.

[32] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.

[33] L. Wang, F. Zhou, Z. Li, W. Zuo, and H. Tan, "Abnormal event detection in videos using hybrid spatio-temporal autoencoder," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 2276–2280.

[34] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection—A new baseline," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6536–6545.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[36] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[37] M. Xu, M. Gao, Y.-T. Chen, L. Davis, and D. Crandall, "Temporal recurrent networks for online action detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5531–5540.

SOE SANDI HTUN received the B.S. degree in computer science and engineering from the University of Information Technology, Yangon, Myanmar, in 2018. She is currently pursuing the M.S. degree in computer science and engineering with the Seoul National University of Science and Technology, Seoul, South Korea. Her research interests include machine learning and computer vision.

JI SANG PARK received the B.A. and M.L.A. degrees in landscape architecture from Kyungpook National University, Daegu, South Korea, in 1995 and 1997, respectively, and the M.S. and Ph.D. degrees in civil and environmental engineering from the University of Wisconsin-Madison, WI, USA, in 2000 and 2005 respectively. Since 2006, he has been with the Electronics and Telecommunication Research Institute, Daejeon, where he is currently a Principal Researcher. His research interests include machine learning, geo-spatial data modeling and analysis, and time-series data analysis.

KANG-WOO LEE received the B.S. and Ph.D. degrees in computer science and statistics from Seoul National University, Seoul, South Korea, in 1993 and 2000, respectively. Since 2000, he has been with the Electronics and Telecommunications Research Institute, Daejeon, where he is currently a Principal Researcher. His research interests include distributed computing systems and data processing systems.

JI-HYEONG HAN received the B.S. and Ph.D. degrees in electrical engineering from KAIST, Daejeon, South Korea, in 2008 and 2015, respectively. From 2015 to 2017, she was a Senior Researcher with the Electronics and Telecommunications Research Institute, Daejeon. Since 2017, she has been with the Seoul National University of Science and Technology, Seoul, South Korea, where she is currently an Assistant Professor. Her research interests include machine learning, human-centered intelligent robotics, and human–robot interaction.