Accident Detection using Deep Networks

Saideep Arikontham
Master of Science in Data Science
Northeastern University
Portland, Maine
arikontham.s@northeastern.edu

***Abstract*—This project addresses the critical issue of timely accident detection from CCTV footage to help victims. The system will use image classification with deep learning frameworks to quickly identify accident scenes. The project will use publicly available traffic and accident image datasets from Kaggle. The core of the system will involve the training of convolutional neural networks (CNN) and residual networks (ResNet). These models will be fine-tuned using different methods to build a best-performing accident recognition model that has high performance on different classification metrics. A successful outcome will be a system that provides accurate detection of accidents in the CCTV video feed, demonstrating the potential of computer vision to improve emergency response and public safety.**

***Index Terms*—CNN, ResNet**

I. Introduction

This project focuses on the development and evaluation of deep learning models for the live detection of accidents in CCTV footage. The primary goal is to create a system capable of accurately and reliably identifying accidents, which can be critical to improving surveillance and enabling rapid response, a result of which I hope to save lives in critical cases.

The project explores two main deep learning architectures: Convolutional Neural Networks (CNNs) and ResNet, a type of Residual Network. Various configurations and training and fine-tuning techniques are investigated to optimize their performance on a dataset of CCTV accident videos and finally select the best performing model for live accident detection from CCTV footage.

II. Previous Work

Real-time accident detection has been the subject of numerous studies in the field of computer vision and deep learning, especially as urban traffic monitoring grows increasingly complex.

Machhi et al. [1] developed a CNN-based system that detects accidents from images and sends alerts via email to nearby hospitals. Their TensorFlow implementation achieved 84% accuracy. However, their approach did not involve live video stream classification nor did it explore advanced architectures or hyperparameter optimization.

Ghosh et al. [2] introduced a CNN-LSTM hybrid model using Inception v3 to classify accident versus non-accident frames in CCTV footage. Their system was deployed on Raspberry Pi for real-time detection with up to 95% accuracy. While the model integrated temporal dynamics using LSTM, it lacked architectural adaptability and efficient tuning mechanisms.

In a more recent work, Sabitha et al. [3] proposed a ResNet-based system that classifies accidents as either major or minor using live video feeds. The system includes automated alerting and proposes the integration of SSD-MobileNet for human presence detection. Their method highlights the advantages of ResNet's deep feature extraction capability for real-time classification.

In contrast, my work employs both custom CNN and ResNet architectures, and introduces hyperparameter optimization using Optuna to fine-tune model performance. Furthermore, O conduct comparative evaluations on live video streams, thereby identifying the most reliable architecture for real-time deployment in accident detection systems taking inspiration from the above references. Though not all of these are implemented in this project, the goal was to build a model good enough to identify accidents efficiently through image classification.

For future scope, I would like to implement something similar to what [4], Ghahremannezhad et al. proposed, which is a hierarchical framework utilizing YOLOv4 for object detection, Kalman filtering and the Hungarian algorithm for object tracking, and trajectory conflict analysis to detect accidents. Although their approach demonstrated robustness in varying lighting and occlusion scenarios, it relied heavily on heuristic trajectory analysis rather than deep learning classification models. This can be considered as a future scope as it was not possible to implement due to time constraints.

III. Data Collection

The dataset required to perform this task must be an image data set containing a training set, a validation set, and a test set. Each of these sets will have images for respective classes, Accident and Non-Accident images. Such data is publicly available at Kaggle. The train set has 369 images for accident data and 422 images for non-accident data. This data will be used for training the deep networks, meaning the models will learn visual patterns from this data. The validation set has 46 images for accident data and 52 images for non-accident data. This data will be used to test the performance of the models after training. The test set has 47 images for accident data and 53 images for non-accident data. This data will be used to test the fully fine-tuned models and promote one model for live CCTV footage accident detection.

Fig. 1. Sample Train set images for Accident class

Fig. 2. Sample Train set images for Non-Accident class

IV. Methodology

A. Data Preprocessing

To prepare the CCTV images for training the accident detection model, several preprocessing steps were applied. First, all images were resized to a fixed dimension of 224×224 pixels to ensure a consistent input size for the convolutional neural network. The images were then converted into PyTorch tensors, enabling them to be efficiently used during model training. Following this, normalization was applied using the standard ImageNet mean and standard deviation values (mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]) to scale the pixel values and align them with the distribution expected by commonly used pretrained models. This is done for CNN models as well as ResNet models for simplicity. In this project, 'Accident' was labeled as 0 and 'Non Accident' as 1 as target labels across the dataset.

B. Training Convolutional Neural Networks

First, a random grid of 50 convolutional neural network architectures are selected and trained to identify the best performing networks. Below are the top 5 best results based on validation set accuracy score obtained from the initial training.

	conv_layers	linear_layers	dropout_rate	hidden_units	batch_size	learning_rate	epochs	val_acc	time_sec
42	1	4	0.5	512	96	0.0001	20	93.877551	784.84
17	5	2	0.4	512	96	0.0010	15	93.877551	558.75
25	3	1	0.2	384	32	0.0001	15	92.857143	464.71
36	2	3	0.4	320	32	0.0010	10	91.836735	255.16
38	1	4	0.5	512	96	0.0010	20	91.836735	752.39

Fig. 3. Random architecture results for CNN

C. Fine-tuning CNN using Optuna

To optimize the CNN architecture for accident detection, I used Optuna, a hyperparameter tuning framework that efficiently searches for the best model configuration. Optuna works by using a trial-based optimization approach where it suggests hyperparameter values, evaluates model performance, and intelligently selects new values based on past results using techniques like Tree-structured Parzen Estimator (TPE) to efficiently explore the search space.

The model architecture was dynamically built by varying the number of convolutional and linear layers, dropout rate, hidden units, and batch size. Each candidate model was trained on preprocessed CCTV images—resized, normalized, and tensor-converted—and evaluated using validation accuracy. The best-performing model's weights and configuration were saved, enabling reproducibility and ensuring optimal performance for the task. Fine-tuning has been performed for 40 trials and below is an image of the best trial.

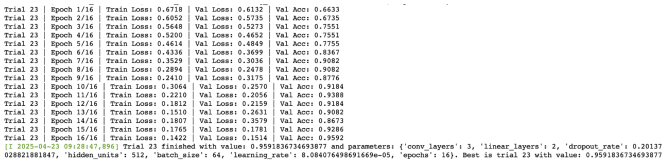

Fig. 4. Trial-23 out of 40 trials for fine-tuning using Optuna

D. Selecting appropriate ResNet architecture

Using similar preprocessing technqiues as specified above, the train and validation data is used to train and evaluate and select best ResNet architecture from ResNet18, ResNet34 and ResNet50. Each architecture is trained for 5 epochs and with a learning rate of 0.001. The best model was found to be the ResNet18 architecute which was select for further fine-tuning.

Model	Epoch	Train Loss	Val Loss	Val Accuracy	Precision	Recall	F1 Score
resnet18	5	0.231073	0.194317	0.918367	0.866667	1.000000	0.928571
resnet34	5	0.325543	0.275991	0.826531	0.843137	0.826923	0.834951
resnet50	5	0.586999	0.508631	0.663265	0.630137	0.884615	0.736000

Fig. 5. ResNet18, ResNet34 and ResNet50 model performance

For each model, the best epoch is selected based on least validation loss and the validation accuracy for that epoch. This score is recorded and used for final selection.

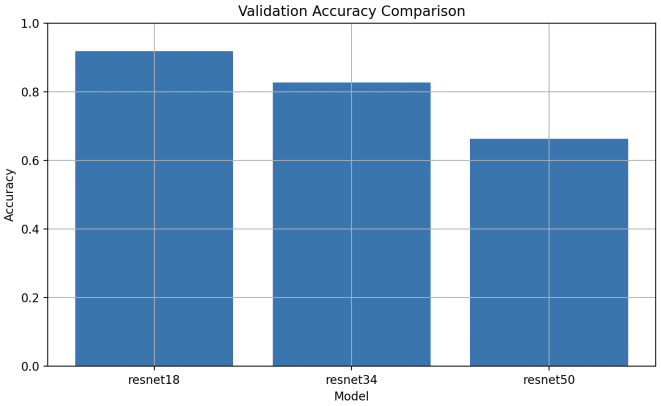

Fig. 6. ResNet18, ResNet34 and ResNet50 performance plot

Therefore, we can conclude that ResNet18 architecture might be the best for our usecase and the other architectures might be more complex for this problem. Hence, we choose to fine-tune ResNet18 architecture.

E. Fine-tuning ResNet18 using Optuna

To fine-tune a pretrained ResNet18 model for accident detection, I employed Optuna to optimize hyperparameters

such as dropout rate, number of frozen layers, learning rate, batch size, and training epochs. The pretrained ResNet18 backbone was customized by freezing a variable number of its initial layers and replacing the final fully connected layer with a dropout layer followed by a two-class output layer. The model was trained on preprocessed CCTV image data, which was resized, normalized, and converted into tensors.

During each trial, only the unfrozen parameters were updated using the Adam optimizer and cross-entropy loss. The validation accuracy was used to guide Optuna's search, and the best-performing model (based on lowest validation loss) was saved for each trial. This approach enabled efficient exploration of the model's performance under different configurations, helping identify the optimal settings for transfer learning in the accident detection task.

```
Trial 6 | Epoch 1/6 | Train Loss: 0.6113 | Val Loss: 0.4201 | Val Acc: 0.8163
Trial 6 | Epoch 2/6 | Train Loss: 0.4462 | Val Loss: 0.3134 | Val Acc: 0.8673
Trial 6 | Epoch 3/6 | Train Loss: 0.3158 | Val Loss: 0.2378 | Val Acc: 0.9388
Trial 6 | Epoch 4/6 | Train Loss: 0.2418 | Val Loss: 0.1845 | Val Acc: 0.9694
Trial 6 | Epoch 5/6 | Train Loss: 0.1937 | Val Loss: 0.1740 | Val Acc: 0.9490
Trial 6 | Epoch 6/6 | Train Loss: 0.1375 | Val Loss: 0.1509 | Val Acc: 0.9490
[I 2025-04-23 16:29:12,459] Trial 6 finished with value: 0.9489795918367347 and parameters: {'dropout_rate': 0.44745999992669554, 'freeze_layers': 5, 'learning_rate': 1.7448710041468407e-05, 'batch_size': 16, 'epochs': 6}. Best is trial 6 with value: 0.9489795918367347.
```

Fig. 7. Enter Caption

V. Results

Once the best model from each of CNN architecture and ResNet18 architecture are fine-tuned, they are used to classify test set images and the performance is analyzed.

A. For ResNet

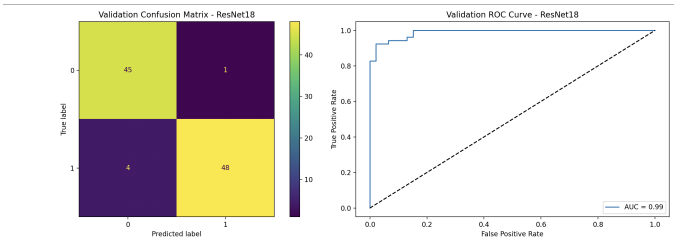

Fig. 8. Confusion matrix and ROC curve for ResNet - validation set

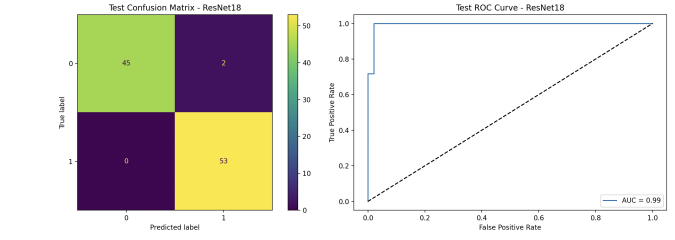

Fig. 9. Confusion matrix and ROC curve for ResNet - test set

B. For CNN

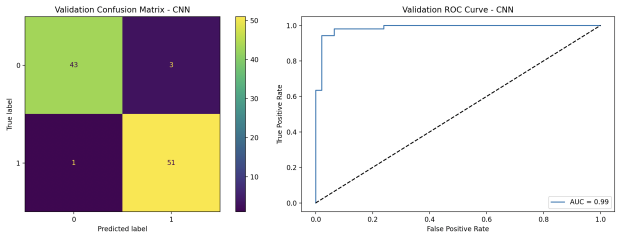

Fig. 10. Confusion matrix and ROC curve for CNN - validation set

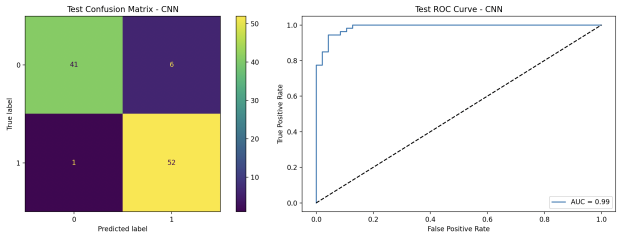

Fig. 11. Confusion matrix and ROC curve for CNN - Test set

Since ResNet18 model has a better generalization compared to CNN model, we promote the ResNet18 architecture to use it for live CCTV footage accident detection.

VI. Real-time accident detection using CCTV footage

A compiled video of CCTV accident footage combined with other accident footage was downloaded from Kaggle. This video will be used to test the fine-tuned ResNet18 model with classification threshold of 0.5. Each frame of the video will be extracted, preprocessed with similar steps and then classified using this fine-tuned model.

When the CCTV accident footage video is given as input, the system will output a captioned video where each frame of the video is labeled with the prediction of whether there is accident or not. Below are few sampled frames of the predictions.

A. Scenario-1

Fig. 12. No Accident Detected

Fig. 13. Accident Detected

B. Scenario-2

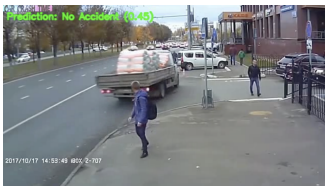

Fig. 14. No Accident Detected

Fig. 15. Accident Detected

C. Mistakes made by the model

The model was unable to identify accidents that happened during nights correctly. This might be due to the video clarity or insufficient image processing during training. There were few frames which were classified as accident even in these night accidents, but it is not consistent. Fine-tuning for more trials might also solve this problem (Initial version had better performance, but code was re-run for less number of trials again due to time constraints).

Fig. 16. Night accident - misclassification case 1

Fig. 17. Night accident - misclassification case 2

VII. Discussion and Summary

This project explored deep learning techniques—CNN and ResNet—for real-time accident detection from CCTV footage. Among the evaluated models, ResNet18 demonstrated the best performance, striking a balance between model complexity and generalization. Deeper architectures like ResNet34 and ResNet50 did not yield significant improvements and introduced unnecessary overhead.

Hyperparameter optimization using Optuna played a crucial role in enhancing model accuracy and robustness. It enabled the identification of optimal configurations for both CNN and ResNet architectures.

The real-time application of the fine-tuned ResNet18 model showed promising results on video frames. However, performance under low-light or night-time conditions was limited, likely due to insufficient training samples from such scenarios.

A. Summary

- Successfully implemented and fine-tuned CNN and ResNet architectures for image-based accident detection.
- ResNet18 emerged as the most reliable model for real-time detection.
- Demonstrated practical application of the model on CCTV footage.
- Identified night-time detection as a key area for future improvement.

This work establishes a solid baseline for real-time accident detection systems and can be extended with multi-frame analysis, object tracking, and lighting-invariant preprocessing in future work.

References

[1] D. A. Machhi, A. S. Pillai, S. S. More, and P. Thirumalaikumar, "Accident detection and reporting using convolution neural network," in *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 2022.

[2] S. Ghosh, S. J. Sunny, and R. Roney, "Accident detection using convolutional neural networks," in *IEEE Conference Proceedings*, 2019.

[3] R. Sabitha, D. Tejashree, and G. Yaashish, "Real-time accident detection and response system using resnet algorithm," in *16th IEEE International Conference on Computational Intelligence and Communication Networks (CICN)*, 2024.

[4] H. Ghahremannezhad, H. Shi, and C. Liu, "Real-time accident detection in traffic surveillance using deep learning," in *arXiv preprint arXiv:2208.06461*, 2022.