Anomaly Detection

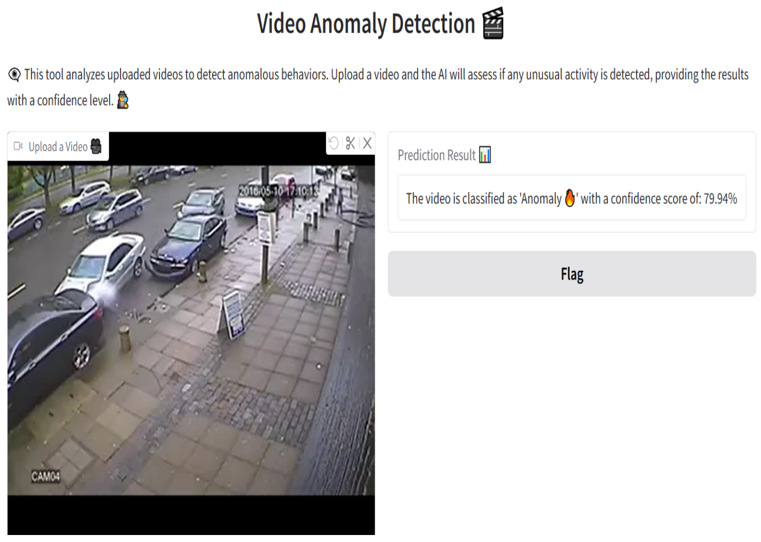
Eyad Alatifi, Mohammed Saati, Abdullah Alharbi, Mohammed Almufarriji
Department Computer Science and Artificial Intelligence
Umm Al-Qura University, Makkah
Computer Vision

Abstract

The growing use of surveillance cameras for security has created a need for automated systems that can detect unusual activities. This project uses deep learning to analyze surveillance videos and spot events that differ from normal behavior. It works by extracting frames from videos and processing them with a 3D ResNet-18 neural network, which learns both the spatial details and the motion patterns in the video. Each video is broken down into a fixed number of frames, transformed into useful features, and then used to train a model that distinguishes between normal and abnormal events. The model achieves an accuracy of 66.8%, with a precision of 67% and a recall of 43.5%, showing that it can effectively tell normal activities from anomalies. To evaluate its performance, standard metrics such as confusion matrices and F1-scores are used. The project is implemented using PyTorch, OpenCV, and other deep learning libraries, and it also uses techniques like data augmentation and frame-based segmentation to improve detection accuracy. Overall, the results suggest that this approach is a promising tool for real-world security applications in automated surveillance systems. Figure[1].

1. Introduction

In today's security-driven world, surveillance systems play a crucial role in ensuring public safety and monitoring critical environments such as airports, shopping malls, hospitals, and streets. Traditional surveillance methods rely on human operators to monitor live feeds, which is labor-intensive, prone to fatigue, and inefficient in detecting subtle anomalies in realtime. With the increasing volume of surveillance footage, there is a pressing need for automated solutions capable of identifying anomalous activities with high accuracy and minimal human intervention. Anomaly detection in surveillance videos refers to the process of identifying behaviors or events that deviate from expected norms. These anomalies can range from suspicious movements, unauthorized access, theft, violence, and other security threats. Unlike standard object detection tasks, anomaly detection is challenging because abnormal events are rare, diverse, and context-dependent, making it difficult to create a comprehensive dataset covering all possible scenarios. In our project, we present an anomaly detection model that can handle 13 types of anomaly events effectively and can be used for camera surveillance in any public place to ensure safety. Our approach leverages 3D convolutional neural networks (CNNs) to learn spatial and temporal patterns in video sequences, improving detection accuracy. Additionally, data augmentation techniques are incorporated to enhance model robustness against environmental variations such as lighting changes and camera angles. By integrating this system into existing surveillance infrastructure, authorities can automate real-time threat detection, reducing response times and enhancing public security.



Video Anomaly Detection 🎬

This tool analyzes uploaded videos to detect anomalous behaviors. Upload a video and the AI will assess if any unusual activity is detected, providing the results with a confidence level.

The video is classified as 'Anomaly 🔥' with a confidence score of: 79.94%

Figure 1. User interface for anomaly detection, where users can upload surveillance videos for analysis.

2. Literature Review

In recent years, video anomaly detection has become a critical area of research, particularly in the field of surveillance and security applications. Traditional approaches have primarily focused on modeling normal behaviors and flagging deviations as anomalies. However, defining anomalies remains a challenge, requiring more robust and adaptive detection frameworks.

2.1. Real-World Anomaly Detection in Surveillance Videos Using MIL

Problem of finding unusual events in surveillance videos—such as accidents or fights—without needing detailed, time-consuming annotations. Instead of marking exactly where an anomaly happens, the method only uses simple video-level labels (normal or abnormal). The authors also created a new, large dataset with 1,900 long surveillance videos covering 13 types of real-world anomalies, which is much larger and more varied than previous collections.

The approach works by splitting each video into smaller segments and treating each segment as an "instance" within a larger "bag" that represents the whole video. Using a technique called Multiple Instance Learning (MIL), the method learns to give high anomaly scores to segments that likely contain unusual events. To improve accuracy, the method adds rules that keep the scores smooth over time (so they don't jump around too much) and sparse (so only a few segments in a video are marked as anomalous). The system uses features extracted from a pre-trained 3D convolutional network to help recognize motion and appearance patterns in the video segments.

The experimental results show that this method works better than previous techniques. It achieves higher true positive rates (correctly identifying anomalies) and lower false alarm rates (fewer normal events mistakenly marked as anomalies). Additionally, when used to classify different types of anomalous activities, the dataset proved to be challenging, highlighting the need for further improvements in video analysis methods. Overall, the approach demonstrates an effective and practical way to detect anomalies in real-world surveillance videos.

2.2. Detecting Anomalies in Security Cameras with 3DCNN and ConvLSTM

This paper presents a new deep learning method for detecting anomalies in security camera videos. The approach focuses on spotting unusual events—such as fights, robberies, and accidents—by analyzing both spatial and temporal information from video sequences. To tackle the challenge of limited resolution and intermittent video quality, the authors combine two types of neural networks: 3D Convolutional Neural Networks (3DCNN) for short-term feature extraction and Convolutional Long Short-Term Memory (ConvLSTM) networks for capturing long-term temporal dependencies. The method was evaluated on several large-scale real-world datasets, including UCF-Crime, XD-Violence, UBI-Fights, CCTV-Fights, and UCF-101.

The proposed architecture starts by processing a stack of consecutive video frames through a series of 3DCNN layers that extract spatio-temporal features. These layers use different filters to capture various aspects of motion and appearance, and are followed by activation, pooling, and dropout layers to ensure robustness. Next, a ConvLSTM layer is applied to model the temporal evolution of these features, effectively learning long-term dependencies in the video data. Finally, a fully connected layer with a softmax activation function classifies the sequence into one of the anomaly classes by aggregating the per-frame predictions using a majority voting scheme.

Experimental evaluations on five large datasets demonstrate that the combined 3DCNN and ConvLSTM model achieves high classification accuracy and excellent Area Under the Curve (AUC) performance. For instance, on the UCF-101 dataset, the model reached 100% accuracy after 50 training iterations, while it achieved competitive accuracy on other datasets (98.5% on UCF-Crime and 99% on CCTV-Fights). Compared to recent state-of-the-art methods, the proposed approach shows lower false positive rates and better overall detection performance, proving its effectiveness in both indoor and outdoor scenarios for various types of anomalous events.

2.3. Anomaly Recognition from Surveillance Videos Using 3D Convolutional Neural Networks

This paper presents a method for recognizing different types of anomalous activities—such as abuse, fighting, road accidents, and more—from surveillance videos. The approach focuses on addressing the challenge of sparse and diverse anomalies by using a large-scale dataset (the UCF Crime dataset) that contains long, untrimmed videos with multiple anomaly types. To better learn the differences between normal and abnormal events, the study provides frame-level annotations and employs spatial augmentation, which helps the model generalize across various real-world conditions.

The proposed system begins by converting each video into individual frames and then grouping consecutive frames into 3D cubes, preserving both spatial and temporal information. To boost the amount of training data, spatial augmentation techniques (horizontal and vertical flips) are applied to the frames. A fine-tuned, pre-trained 3D Convolutional Neural Network (3D ConvNets) is then used to extract spatiotemporal features from these cubes. The network architecture consists of several convolutional layers with batch normalization and pooling, followed by fully connected layers, ending with a SoftMax classifier for multiclass anomaly recognition. Training is carried out in a semi-supervised manner using the detailed frame-level annotations.

The experimental evaluation on the UCF Crime dataset shows that the proposed method achieves significantly better performance in multiclass anomaly recognition compared to previous approaches. Specifically, the model reached an overall accuracy of 45%, outperforming earlier methods like C3D (23%), TCNN (29%), and a motion-based TCNN approach (31%). Performance metrics such as the AUC-ROC, precision, recall, and F-measure indicate that the model can effectively distinguish between several anomaly classes. However, some classes (e.g., arrest, burglary, and shooting) still pose challenges due to high intra-class similarities. Overall, the findings demonstrate that using a fine-tuned 3D ConvNets model with spatial augmentation and frame-level labels can improve the recognition of complex anomalous activities in surveillance videos.

3D Convolutional Neural Networks (3DCNNs) play a pivotal role in anomaly detection by simultaneously capturing spatial and temporal information in video sequences. Unlike traditional 2D CNNs, which process individual frames, 3DCNNs operate on volumetric data, allowing them to learn motion dynamics and scene transitions. This makes them particularly effective for distinguishing normal activities from subtle, anomalous patterns that evolve over time.

3. Methodology

3.1. Tools and Technologies

To develop an efficient anomaly detection system, we utilize a combination of deep learning, computer vision, data processing, and visualization tools. The selected technologies facilitate model training, data handling, and performance evaluation.

3.1.1. Development Environment

Google Colab: Provides a cloud-based platform with access to GPUs, significantly reducing training time compared to local setups. It integrates seamlessly with Google Drive for efficient dataset management.

3.1.2. Computer Vision and Image Processing

OpenCV (cv2):

- Reading and handling video files using `cv2.VideoCapture`.
- Extracting and saving frames with `cv2.imwrite`.

PIL (Pillow):

- Loading and manipulating images from disk.

3.1.3. Data Handling and Preprocessing

pandas:

- Loading and preprocessing video labels and metadata.
- Merging, filtering, and structuring data for training.

numpy:

- Handling vectorized computations for deep learning.
- Performing matrix operations for feature extraction.

torchvision.transforms:

- Resizing images to a standard format.
- Normalizing pixel values to enhance deep learning performance.
- Applying augmentations like flipping, rotation, and brightness adjustments to improve model generalization.

3.1.4. Deep Learning Frameworks and Models

PyTorch:

- Creation of custom deep neural networks.
- Efficient training and evaluation using GPU acceleration.
- Real-time inference for anomaly detection in surveillance footage.

torchvision.models.video (3D ResNet-18):

- A pre-trained model, fine-tuned on our dataset to classify normal vs. anomalous events.

3.1.5. Visualization and Performance Monitoring

matplotlib.pyplot:

- Plotting loss curves, accuracy trends, and confusion matrices.
- Displaying sample frames from surveillance videos.

tqdm:

- Tracking dataset preprocessing, such as video frame extraction.
- Monitoring model training progress through real-time epoch updates.

3.2. Dataset

The DCSASS (Detection of Criminal and Suspicious Activities in Surveillance) dataset is a comprehensive collection of surveillance videos designed to facilitate research in anomaly detection. It comprises 16,853 videos categorized into 13 distinct classes, each representing different scenarios of interest (Figure 2).

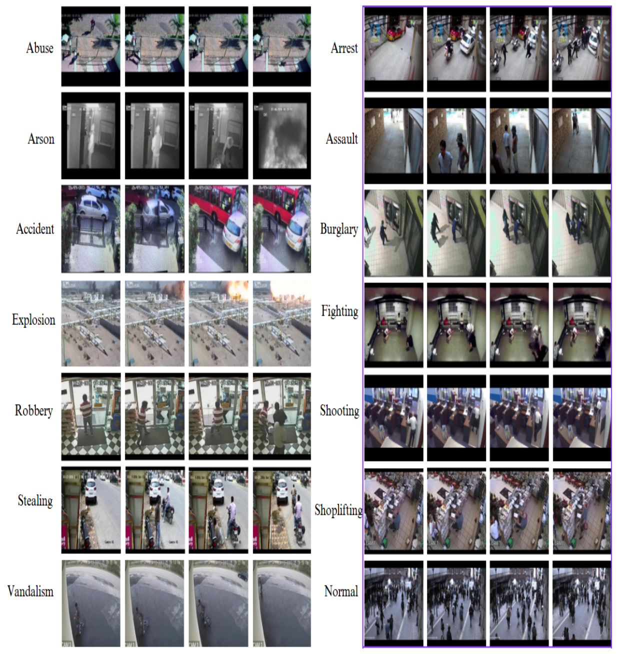


Figure 2. Examples of different anomalies from the training and testing videos in our dataset.

Classes Included:

- Abuse
- Arrest
- Arson
- Assault
- Accident
- Burglary
- Explosion
- Fighting
- Robbery
- Shooting
- Stealing
- Shoplifting
- Vandalism

Each class contains a varying number of video clips, typically ranging from 50 to 150 per class, ensuring a balanced representation across different types of activity. The dataset was developed based on the UCF Crime Dataset but refines certain limitations by providing more focused and clearly annotated short clips (4–5 seconds), labeled as either normal (0) or abnormal (1).

3.3. Data Preprocessing

- **Frame Extraction:** Each video is converted into a sequence of 32 fixed frames using OpenCV.
- **Frame Resizing & Normalization:** Frames are resized to 112×112 pixels, and pixel values are normalized to the [0,1] range.
- **Handling Missing Values:** Missing values are addressed using forward filling and mode replacement for categorical features to ensure data integrity.
- **Data Augmentation:** Horizontal flipping, random cropping, and brightness adjustments are applied to increase dataset diversity.
- **Dataset Splitting:** The dataset is divided into training (60%), validation (20%), and test (20%) sets.

3.3.1. Dataset Preparation

- **Frame Loading and Processing:** Frames are extracted from subvideo folders, normalized to [0,1], and resized to 112X112.
- **Fixed Frame Count Handling:** Videos with fewer than 32 frames are padded by repeating the last frame, while videos with more than 32 frames are truncated.
- **Data Labeling:** A mapping dictionary correlates each subvideo name with its corresponding label (0 or 1).
- **Data Structuring:** The dataset is structured into a list of tuples, each containing a video tensor and a label for downstream training.

  Below is an example of a structured data instance:

  (torch.Size([32, 32, 3, 112, 112]),

  tensor([0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1]))

3.4. Model Architecture

Overview: The model architecture for anomaly detection in surveillance videos is based on 3D convolutional neural networks (3D CNNs), specifically ResNet-18 (3D ResNet). This architecture is well-suited for spatiotemporal feature extraction from video frames.

Input Shape: (B, C, F, H, W)

- B = Batch size (32 videos per batch)
- C = Number of channels (RGB $\rightarrow$ 3)
- F = Number of frames per video (32)
- H = Frame height (112 pixels)
- W = Frame width (112 pixels)

3D ResNet-18: We start by loading the `r3d_18(pretrained=True)` model from `torchvision.models.video`, which extends 2D ResNet-18 into 3D convolutions for capturing both spatial and temporal dimensions. We modify the final fully connected layer to match our binary classification (normal vs. anomaly). The Adam optimizer and cross-entropy loss are used during training, and the model is deployed to a GPU if available.

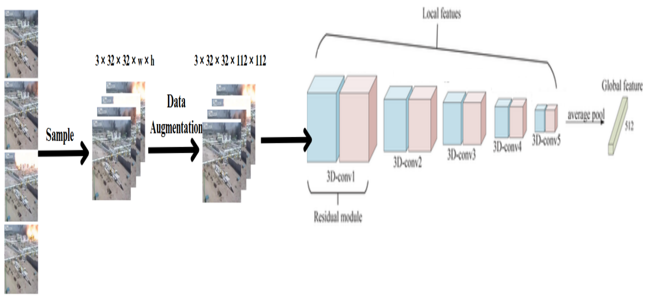


Figure 3. The process involves data augmentation, multiple 3D convolutional layers for local feature extraction, and global feature generation through average pooling, resulting in a 512-dimensional feature vector.

3.4.1. Device Configuration

The model is deployed to a GPU if available; otherwise, it defaults to the CPU.

3.4.2. Optimizer and Loss Function

Adam is used with a learning rate of 0.001, and cross-entropy handles the binary classification task.

3.4.3. Training Loop

- **Data Permutation:** Videos are reshaped to [BatchSize, Channels, Time, Height, Width].
- **Loss Computation and Backpropagation:** Cross-entropy loss is calculated, and gradients are backpropagated to update weights. Gradient clipping is employed to prevent exploding gradients.
- **Performance Tracking:** Training loss and accuracy are calculated per epoch.

3.4.4. Validation Loop

- **Evaluation Mode:** Gradients are disabled to save computation and avoid altering model weights.
- **Performance Metrics:** Validation loss and accuracy are computed to monitor generalization.

4. Step-by-Step Implementation

Below, we map our *code* to each major step in the pipeline, aligning with the preceding methodology.

1. **Data Collection and Labeling:**
    - We load CSV annotations (e.g., `Abuse.csv`, `Arson.csv`, etc.) and merge them into a label dictionary mapping subvideo names to {0,1}.
2. **Frame Extraction:**
    - Using OpenCV (`cv2.VideoCapture`), we extract exactly 32 frames per video. If videos have fewer, we repeat the last frame; if they have more, we truncate.
3. **Dataset Splitting:**
    - We split subvideo paths into train/val/test (60%/20%/20%) using `train_test_split`.
4. **Data Transformation and Augmentation:**
    - In PyTorch, we apply `Resize((112,112))`, random horizontal flips, rotations, and color jitter using `torchvision.transforms`.
5. **DataLoader Setup:**
    - We create `Dataset` objects for train, val, and test sets, each yielding a tuple (`video_tensor, label`).
    - `DataLoader` wraps these Datasets with batch size 32, shuffling for train/val.
6. **Model Initialization:**

- Load `r3d_18(pretrained=True)` and replace its FC layer with a `Linear(in_features,2)` for binary classification.

7. **Training Procedure:**
   - For each epoch:
     (a) Forward pass: compute the prediction.
     (b) Cross-entropy loss calculation.
     (c) Backprop + optimizer step.

8. **Validation and Early Stopping:**
   - We compute validation accuracy and loss each epoch.
   - If validation accuracy plateaus or starts decreasing for several epochs, we may halt training to prevent overfitting.

9. **Testing and Final Evaluation:**
   - We run inference on the test set (no gradient mode) and track classification metrics such as accuracy, precision, recall, and F1.
   - We generate a confusion matrix, per-category statistics, and t-SNE visualizations of feature embeddings.

This step-by-step approach ensures reproducibility, as each phase—data preparation, model design, training, validation, and testing—maps directly to the provided Python code blocks.

5. Model Evaluation

Once the model has been trained, a comprehensive evaluation is crucial to verify its effectiveness on unseen data and ensure it generalizes beyond the training set. The evaluation procedure can be broken down into the following steps:

5.1. Inference and Prediction

- The model is switched to evaluation mode (`model.eval()`) to disable operations like dropout.

- Video tensors are permuted into [Batch, Channels, Time, Height, Width] to match the 3D CNN input format.

- Predictions are computed in a no-gradient context (`torch.no_grad()`). The outputs are logits, and the predicted class is determined by the `argmax` function.

5.2. Loss and Accuracy Computation

- The test loss is calculated for each batch to measure how closely predictions match ground-truth labels.

- The average test loss provides a global measure of model fit on unseen data.

- **Accuracy:** Accuracy is computed by comparing predicted labels with ground truth. The number of correct predictions divided by total samples gives overall classification accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.3. Evaluation Metrics

To get deeper insights into model performance, additional metrics beyond accuracy are essential:

- **Precision:** Precision measures how many of the predicted positive cases are actually positive. It is crucial in anomaly detection to avoid false positives (incorrectly detecting an anomaly when there isn't one).

  $$Precision = \frac{TP}{TP + FP}$$

  A high precision indicates that when the model detects an anomaly.

- **Recall:** Recall quantifies how many actual anomalies were correctly identified. It is especially important in **anomaly detection**, where missing an anomaly (false negative) can be critical.

  $$Recall = \frac{TP}{TP + FN}$$

  A high recall means the model **successfully detects most anomalies**, reducing the chances of missing critical events.

- **F1-Score:** The F1-score is the harmonic mean of precision and recall, providing a balanced metric when dealing with imbalanced datasets (which is maybe seen in anomaly detection models).

  $$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

  The F1-score is particularly useful when both **false positives and false negatives** need to be minimized.

- **Importance in Anomaly Detection:**

  In anomaly detection, accuracy alone is misleading since **anomalies are rare**. A model may appear accurate by predicting only normal cases while failing to detect real anomalies. To evaluate performance effectively, we need to metrics like precision, recall, and F1-score. Precision is crucial when false alarms must be minimized, while recall is key in critical applications like medical diagnostics and security, where missing an anomaly is dangerous. F1-score balances both, making it the preferred metric for anomaly detection.

6. Challenges and Their Mitigation

Despite the straightforward training pipeline, several challenges emerged during development:

6.1. Large Dataset and Memory Constraints

Challenge: Handling a dataset with over 16,000 short videos for frame extraction, storage, and model training can be demanding in terms of both memory and storage I/O. **Mitigation:**

- Leveraging Google Colab's GPU and higher RAM capacity.
- Extracting and storing only a fixed number of frames (32) per video reduces overhead.

6.2. Class Imbalance and Rare Anomalies

Challenge: Some anomalies (*Arson*, *Explosion*) have fewer anomaly samples than others, making it harder for the model to learn these categories. **Mitigation:**

- Applying data augmentation to increase sample diversity.
- Using carefully stratified train-test splits to preserve distribution.

6.3. Temporal Modeling Complexity

Challenge: Capturing spatiotemporal patterns is more complex than static image classification, and naive 2D CNNs do not account for motion. **Mitigation:**

- Implementing 3D ResNet for joint spatial-temporal learning.
- Ensuring that consecutive frames are processed as a single clip for more robust temporal features.

6.4. Convergence and Overfitting

Challenge: Training deep 3D networks can lead to overfitting, especially with smaller subvideo clips. **Mitigation:**

- Using Dropout layers and data augmentation to reduce overfitting.
- Employing validation checks and early stopping if validation accuracy declines or plateaus.

7. Results

7.1. Quantitative Performance

Training was conducted over 20 epochs with a batch size of 32. Table 1 summarizes the training and validation metrics at each epoch, highlighting the training loss, training accuracy, validation loss, and validation accuracy.

Table 1. Training and Validation Metrics by Epoch

Epoch	Train Loss	Train Acc	Val Loss	Val Acc
1	0.9317	0.5942	0.6573	0.6241
2	0.6933	0.5829	0.8201	0.4723
3	0.6924	0.5748	3.6548	0.5181
4	0.7019	0.5481	0.9184	0.5325
5	0.7008	0.5764	0.9885	0.6096
6	0.7299	0.5691	0.7119	0.5807
7	0.6939	0.5950	0.6222	0.6506
8	0.6752	0.6217	0.7106	0.5494
9	0.6706	0.6128	0.6783	0.5880
10	0.7129	0.5974	0.6782	0.5157
11	0.6944	0.5918	3.3523	0.5783
12	0.7161	0.5918	0.6595	0.6000
13	0.6882	0.6257	0.6467	0.6361
14	0.6756	0.6362	2.5122	0.5060
15	0.6621	0.6435	0.6212	0.6554
16	0.6448	0.6580	0.6077	0.6699
17	0.6466	0.6500	0.6594	0.6410
18	0.6084	0.6686	0.5818	0.7060
19	0.6094	0.6718	1.4786	0.5277
20	0.6060	0.6799	0.5932	0.6723

Overall, the training loss decreases slightly with each epoch, although **validation loss** shows some gaps and instabilty (Epochs 3, 11, and 14). The **highest** validation accuracy recorded is around **70.60%** (Epoch 18), while the final validation accuracy at Epoch 20 is **67.23%**.

After training, the model was evaluated on the **test set** (20% of the simple). The test loss was **0.6313**, and the overall test accuracy reached **66.83%**. Table 2 summarizes additional classification metrics:

The confusion matrix in Figure 4 shows that the model tends to misclassify a portion of anomalous videos as normal, reflecting a relatively lower recall for anomalies.

7.2. Visualizations

Training and Validation Loss. Figure 5 plots the loss curves over 20 epochs. The training loss remains fairly sta-

Table 2. Test Set Classification Report

Class	Precision	Recall	F1-Score	Support
Normal	0.67	0.84	0.75	238
Anomaly	0.67	0.43	0.52	175
Accuracy		0.6683		
Macro Avg	0.67	0.64	0.63	413
Weighted Avg	0.67	0.67	0.65	413

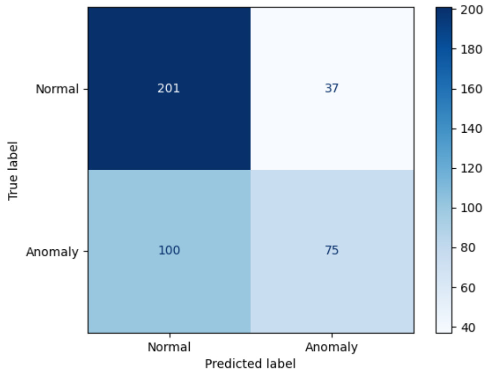


Figure 4. Confusion Matrix illustrating model predictions vs. ground truth on the test set.

ble, while the validation loss has several spikes, suggesting fluctuations in how well the model generalizes at different points during training.

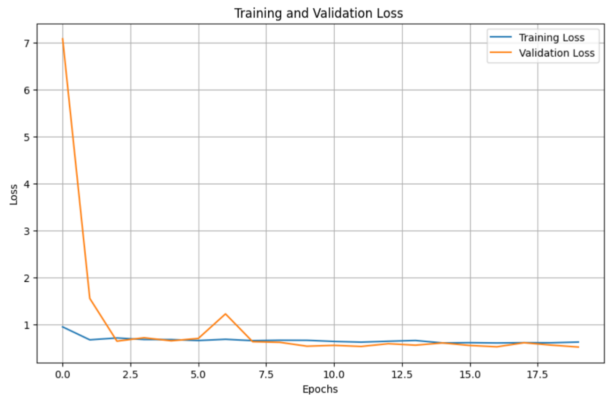


Figure 5. Training vs. validation loss across epochs.

t-SNE Embeddings. We extracted penultimate-layer features and used **t-SNE** to project them into 2D space (Figure 6). Each point represents one test video, colored by its ground-truth label (blue = Normal, orange = Anomaly). The resulting plot shows partial clustering of each class, although significant overlap remains.

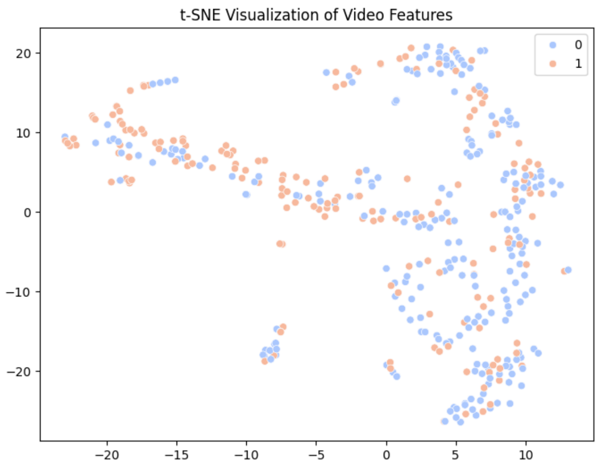


Figure 6. t-SNE visualization of extracted 3D ResNet-18 features, colored by class label.

Per-Category Accuracy. We computed accuracy for each anomaly class (*Fighting, Shooting, RoadAccidents*). Figure 7 shows a bar chart of category-wise accuracy. Some categories (*Fighting, Shooting*) exhibit higher accuracy, whereas others (*Burglary, Arson, Stealing*) are more challenging.

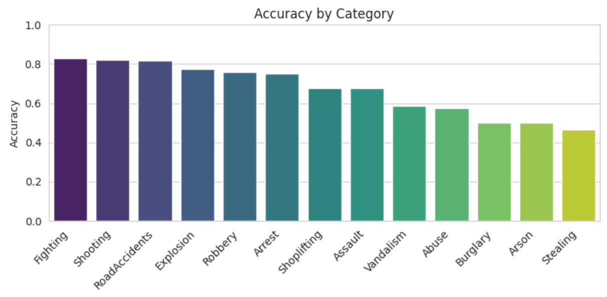


Figure 7. Accuracy by Category for the 13 anomaly classes in the test set.

Misclassifications by Category. Figure 8 illustrates how each category was misclassified as normal (Misclassified_As_0) or anomalous (Misclassified_As_1). Burglary, for example, shows a higher rate of being incorrectly predicted as normal, consistent with the confusion matrix trends.

8. Discussion

Overall Performance. The final test accuracy of **66.83%** demonstrates that our 3D ResNet-18-based approach can identify anomalies to a moderate degree, but there is room for improvement. While the macro-averaged F1-score (0.63) indicates a fair balance across classes, the **recall** for anomalies (0.43) which is low to detect a correct anomaly, suggests

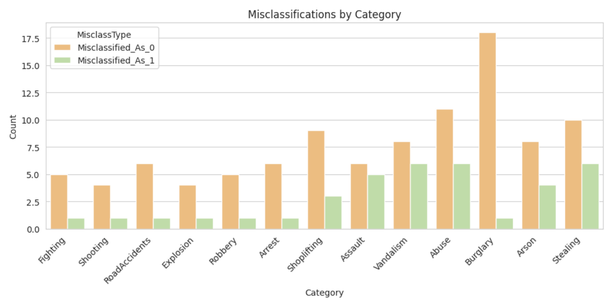


Figure 8. Misclassification counts by category and predicted label.

that the model often fails to recognize all abnormal events. A big shortcoming in safety-critical applications.

Instable validation loss. Spikes in validation loss (e.g., at Epochs 3, 11, and 14) can indicate instability in training, possibly due to sudden overfitting or sensitivity to certain **mini-batches**. This behavior suggests additional regularization, more robust augmentation, or a different learning rate schedule could stabilize training.

t-SNE Interpretations. The t-SNE plot reveals partial separation between normal and anomalous samples. the t-SNE shows that the model not classify the supreted featuere correctly, which may lead to misclassification. More advanced architectures or fine-tuning techniques (adding attention mechanisms or using Transformer-based video models) could improve separability.

Potential Improvements.

- **Longer Clips or Motion Emphasis**: Capturing more temporal context either by increasing frames from 32 to a larger number or incorporating LSTM/GRU modules may help detect prolonged anomalies.
- **Fine-grained Annotation**: If available, more precise labeling could guide the network to learn more key features of the class.
- **Regularization and Hyperparameters**: Introducing stronger regularization (dropout, weight decay) or adopting a cyclical learning rate could help manage overfitting.
- **Multitask Learning**: Simultaneously classifying broader scene categories (indoor vs. outdoor) might offer additional context for identifying unusual events.

In summary, while the model effectively learns certain anomaly classes, **improving recall** for difficult categories remains a key challenge. Further refinements, particularly in handling imbalanced data and stabilizing training dynamics, could lead to more robust performance in real-world surveillance scenarios.

9. Conclusion

This project introduced a deep learning approach for anomaly detection in surveillance videos using a 3D ResNet-18 architecture. By processing fixed-length video clips of 32 frames, the model effectively captured both spatial and temporal features necessary to differentiate normal behavior from unusual activities. The implementation leveraged tools such as PyTorch and OpenCV within a cloud-based environment (Google Colab) to ensure efficient training and evaluation, while data augmentation techniques helped improve model generalization and reduce overfitting. The model achieved an overall test accuracy of approximately 66.83%, with precision around 67% and recall of 43.5%, indicating moderate success in identifying anomalies. Comprehensive evaluations using standard metrics, confusion matrices, and t-SNE visualizations provided insight into the strengths and limitations of the approach. While the spatiotemporal representations learned by the model were effective in capturing dynamic patterns, challenges such as overlapping feature spaces and class imbalance—particularly for rare anomalies—remained significant hurdles. Future improvements could focus on enhancing the temporal context by incorporating longer clips or recurrent modules, addressing class imbalance with class-weighted losses and targeted data augmentation, and refining the model architecture through hybrid designs or attention mechanisms. Additionally, advanced regularization techniques and hyperparameter tuning may further stabilize training and improve anomaly recall. Overall, while the current system demonstrates a promising approach to automated surveillance, further refinements are essential for deployment in real-world security scenarios where precise anomaly detection is critical.

References

[1] R. Maqsood, U. I. Bajwa, G. Saleem, R. H. Raza, and M. W. Anwar, ”Anomaly Recognition from Surveillance Videos using 3D Convolution Neural Network,” *arXiv preprint arXiv:2101.01073*, 2021. Available: `https://arxiv.org/pdf/2101.01073`

[2] W. Sultani, C. Chen, and M. Shah, ”Real-World Anomaly Detection in Surveillance Videos,” Center for Research in Computer Vision, University of Central Florida. Available: `https://www.crcv.ucf.edu/projects/real-world/`

[3] M. Ali, M. A. Shah, and A. M. Khan, ”Anomaly Detection in Surveillance Videos: A Deep Learning Approach,” *Research Square*, 2023. Available: `https://assets-eu.researchsquare.com/files/rs-2524566/v1_covered.pdf?c=1677338975`

[4] Deepchecks, ”Understanding F1 Score, Accuracy,

ROC-AUC, and PR-AUC Metrics for Models," 2024.
Available: https://www.deepchecks.com/f1-score-accuracy-roc-auc-and-pr-auc-metrics-for-models/