“Anomaly Activity Detection Using Video Surveillance”

Submitted in partial fulfilment of the requirements
of the degree of

Bachelor of Technology

In

Robotics and Artificial Intelligence

By

More Kshitij, MIS- 612302037

Patadiya Vishesh, MIS- 612302042

Tanpure Anushka, MIS- 612302062

Waghchaure Saraswati, MIS- 612302067

Under the Guidance of-

Dr. S. S. Mohite

DEPARTMENT OF MECHANICAL ENGINEERING

COEP TECHNOLOGICAL UNIVERSITY

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our project guide, **Dr. S. S. Mohite**, for their invaluable guidance, continuous encouragement, and support throughout the course of this project. Their deep insights into the domain of Computer Vision and practical advice on tackling the challenges of video anomaly detection were instrumental in shaping our approach and methodology.

We also wish to thank the **Department of Mechanical Engineering** at **COEP Technological University** for providing us with the necessary infrastructure and resources to carry out this research effectively.

Finally, we extend our thanks to our friends and families for their patience and understanding during the development of this project.

Sincerely,

More Kshitij

Patadiya Vishesh

Tanpure Anushka

Waghchaure Saraswati

ABSTRACT

The exponential growth of video surveillance infrastructure has created a critical data bottleneck, rendering manual monitoring by human operators inefficient and prone to error due to fatigue. This project presents a robust, automated framework for **Suspicious Activity Detection** utilizing a hybrid Deep Learning architecture designed to transition security systems from reactive evidence gathering to proactive real-time alerting.

We implement a **Convolutional Recurrent Neural Network (CRNN)** pipeline that effectively bridges the gap between static image recognition and dynamic action understanding. The architecture integrates a **ResNet-50** backbone, pre-trained on ImageNet, to extract high-level spatial features from individual video frames, followed by **Long Short-Term Memory (LSTM)** networks to model temporal dependencies and sequence evolution. Unlike traditional approaches reliant on hand-crafted features or 3D convolutions, this hybrid model balances high accuracy with computational efficiency.

The system is trained and validated on the **UCF-Crime dataset**, a challenging benchmark comprised of unconstrained, real-world CCTV footage rather than staged scenarios. To address the computational challenges of high-dimensional video data, the methodology incorporates **temporal subsampling** strategies and **transfer learning**. The resulting model demonstrates the capability to discern complex anomalies—such as **Assault, Arson, and Fighting**—from normal background activities, establishing a viable engineering foundation for scalable, automated intelligent surveillance systems.

CONTENT

LIST OF FIGURES

1. INTRODUCTION

In an increasingly urbanized world, video surveillance has become the backbone of modern security infrastructure. From bustling city streets and airports to quiet ATM vestibules and retail stores, millions of Closed-Circuit Television (CCTV) cameras record continuously, generating an unprecedented volume of visual data. However, this ubiquity presents a paradoxical challenge: while we have more eyes on the world than ever before, we lack the capacity to effectively monitor what they see.

The traditional model of video surveillance is fundamentally **reactive**. Vast archives of footage are typically reviewed only *after* a security incident has occurred, serving primarily as forensic evidence rather than a preventative measure. Real-time monitoring, when attempted, relies heavily on human operators. However, research in human factors engineering has consistently shown that human attention degrades rapidly when monitoring static or repetitive scenes. Studies suggest that after just 20 minutes of continuous monitoring, an operator may miss up to 95% of screen activity due to cognitive fatigue and "inattentional blindness." Consequently, the vast majority of surveillance footage—and the critical security insights it contains—remains unanalysed.

Artificial Intelligence (AI) and **Computer Vision** offer a transformative solution to this bottleneck. By shifting the paradigm from manual to automated surveillance, we can transition security systems from reactive evidence repositories to proactive alert mechanisms. This project explores the application of Deep Learning to automate the detection of suspicious activities. Unlike simple motion detection, which triggers false alarms from benign movements like swaying trees, deep learning models can be trained to understand the *semantics* of human behaviour.

This report details the design and implementation of a **Spatio-Temporal Deep Learning framework**, specifically a Convolutional Recurrent Neural Network (CRNN). By combining the visual feature extraction capabilities of Convolutional Neural Networks (CNNs) with the sequence modelling power of Long Short-Term Memory (LSTM) networks, this system is capable of learning complex temporal patterns associated with criminal behaviour—such as Fighting, Arson, and Burglary—thereby enabling immediate, automated threat detection without the need for constant human supervision.

1.1 Problem Statement

With the rapid growth of urbanization, CCTV cameras have popped up everywhere—on streets, in shops, and outside homes. While this gives us a massive amount of video footage, it has created a new problem: there is simply too much video for humans to watch.

Currently, video surveillance is mostly reactive. Security teams usually check the footage only *after* a crime has already happened to find evidence. Real-time monitoring is done by human guards, but this is inefficient. Studies show that after staring at a screen for just 20 minutes, a person's attention span drops significantly, meaning they are likely to miss critical events like a fight or a break-in.

Trying to automate this with computer vision brings up several engineering challenges that this project aims to solve:

1. **Understanding Time:** A computer can easily identify a "person" in a photo. But detecting an action—like "fighting" vs. "hugging"—requires understanding how things move over time. The model needs to analyse a sequence of frames, not just one static image.
2. **Data Overload:** Video files are huge. Processing every single frame of a 24-hour surveillance tape requires massive computing power. We need a way to process this data efficiently without missing important details.
3. **The "Normalcy" Problem:** In real life, 99% of surveillance footage is boring (nothing is happening). Crimes are rare. This creates a "class imbalance," making it hard to train a model because it sees thousands of examples of normal walking but very few examples of actual crimes.

Project Goal:

The goal of this project is to build an automated Deep Learning model that can take raw surveillance video, analyse the movement patterns over time, and flag suspicious activities like fighting or arson. This moves surveillance from being just a recording tool to an active safety system.

2. LITERATURE REVIEW

[1] Suspicious Activity Detection Using Yolo by Mrs. S. Jansi Rani et al.

This paper introduces a foundational framework for implementing an intelligent surveillance system using object detection. The authors focus on detecting specific suspicious activities like accidents and chain snatching using the YOLO (You Only Look Once) algorithm, specifically leveraging YOLOv5[1]. The methodology outlines a complete pipeline: data gathering from diverse sources like satellite imagery and public cameras [2], preprocessing involving dataset division and classification [3], and model training using deep learning frameworks like TensorFlow or PyTorch[4]. The system's core functionality relies on YOLO's ability to process input images through a Convolutional Neural Network (CNN) [5], divide them into grids [6], and predict bounding boxes with class probabilities[7]. A key feature is the alert generation mechanism, which triggers notifications when detected activities deviate from predefined norms[8]. While the paper successfully demonstrates the application of YOLOv5 for identifying static and dynamic threats [9], it acknowledges the necessity for multidisciplinary expertise and strict adherence to legal and ethical standards in deployment[10].

[2] Proactive Headcount and Suspicious Activity Detection using YOLOv8 by Sudharson D et al.

Advancing from the general application of YOLOv5, this research proposes a more specialized system that integrates the newer YOLOv8 model for proactive crowd management and anomaly detection. The authors address the specific challenge of "crowd-smashing" accidents and the need for real-time monitoring in high-density areas[11111111]. The system utilizes YOLOv8 due to its superior speed (155 frames per second) and single-pass architecture [12121212], making it highly effective for real-time headcount and threat identification[13]. The methodology includes data acquisition from open-source platforms [14], preprocessing with Roboflow [15], and training on a dataset of 1066 images[16]. A significant enhancement is the integration of the Twilio cloud communication platform to send instant SMS alerts to security personnel upon detecting anomalies like weapons, fire, or overcrowding[17]. The study reports exceptional performance, with an average accuracy rate of over 95% for headcount detection[18], demonstrating a shift towards more automated and responsive surveillance solutions compared to earlier iterations.

[3] Smart Surveillance Systems Using YOLOv8: A Scalable Approach for Crowd and Threat Detection by P. Siva et al.

This paper further refines the application of YOLOv8 by emphasizing scalability and the integration of deep learning for anomaly detection. It proposes a comprehensive framework that combines YOLOv8 for object detection with LSTM (Long Short-Term Memory) networks for behavioural analysis[19191919]. This hybrid approach addresses a key limitation of pure object

detection models: the inability to capture temporal dependencies in video data[20]. By training an LSTM model on historical data, the system can recognize deviation patterns in movement and behaviour, such as loitering or unauthorized access[21]. The research also highlights the optimization of the system for low-power edge devices, reducing computational overhead by 30%[22]. This makes it a scalable solution suitable for deployment on existing CCTV infrastructure[23]. The system achieved 95.4% accuracy in object detection and 92.7% in anomaly recognition[24], showcasing the effectiveness of combining spatial analysis (YOLO) with temporal analysis (LSTM) for a more robust surveillance system.

[4] Human Anomaly Detection System Using YoloV8 and LSTM by Jash Tandel et al.

Building upon the hybrid model concept, this research explicitly compares two distinct approaches: a spatial-only approach using YOLOv8 and a spatiotemporal approach using a "MoBiLSTM" model (likely referring to a MobileNet backbone with LSTM)[25252525]. The study creates a custom dataset of specific human actions like punching, kicking, and running[26]. The YOLOv8 model achieved 90% accuracy in spatial classification but struggled with sequential actions[27272727]. In contrast, the MoBiLSTM model, which used MediaPipe for skeletal keypoint extraction [28], achieved a higher overall accuracy of 97%[29]. This paper critically analyses the limitations of YOLOv8 for complex action recognition, noting its inability to capture temporal patterns[30]. It advocates for the use of pose estimation combined with LSTM to detect non-rigid human movements and complex behaviors[31]. The research concludes that while YOLOv8 is excellent for detection, models like MoBiLSTM are superior for analysing the *sequence* of actions required for accurate anomaly detection[32].

[5] Enhanced YOLOv11 for Image-Based Anomaly Detection in Freight Train Gate Chains by Han Jianfeng et al.

This paper represents the cutting edge of YOLO-based anomaly detection, introducing an "Enhanced YOLOv11" model (SCW-YOLO)[33]. Although applied to a specific industrial domain (freight train gate chains), the methodological advancements are highly relevant to general surveillance. The authors address limitations in detecting small targets and complex backgrounds by integrating a Simple Attention Module (SimAM) into the backbone for better feature extraction[34]. They also employ a Content-Aware ReAssembly of Features (CARAFE) operator for improved up-sampling [35]and a Wise Intersection over Union (WIoU) loss function to reduce the impact of low-quality samples[36]. The model achieved a 99.5% mean Average Precision (mAP), surpassing the baseline YOLOv11 by 5.4%[37]. This research demonstrates how architectural modifications—specifically attention mechanisms and advanced loss functions—can significantly boost the precision of YOLO models, paving the way for highly accurate anomaly detection in challenging environments.

[6] Enhancing Road Safety: Real-Time Surface Anomaly Detection Using YOLOv8 by Saeeda Varawalla et al.

This study focuses on a specific application of anomaly detection—road surface hazards—demonstrating the versatility of YOLOv8. The authors develop a system to detect potholes, wet surfaces, and other road anomalies using a curated dataset of 4,968 images[38383838]. A key contribution is the integration of the YOLOv8 model into a Flask-based web application [39], enabling real-time detection and visualization on a map[40]. The evaluation showed a mean Average Precision (mAP) of 0.879 at IoU 0.5[41]. The paper compares YOLOv8 with its predecessor, YOLOv7, finding that YOLOv8 offers superior speed (11.4 ms inference time vs. 24.9 ms) and accuracy, particularly for complex background elements[42424242]. This research highlights the practical deployment aspect of anomaly detection systems, emphasizing the importance of user interfaces and real-time data visualization for effective monitoring.

[7] YOLO-ABD: A Multi-Scale Detection Model for Pedestrian Anomaly Behaviour Detection by Caijian Hua et al.

This final paper presents "YOLO-ABD," a specialized modification of YOLOv8n designed specifically for pedestrian anomaly detection[43]. Addressing the challenge of small target detection (e.g., distant pedestrians), the authors introduce a new small-object detection head[44]. They also integrate a Group Shuffle Convolution (GSConv) module to reduce computational complexity while maintaining accuracy [45]and the SimAM attention mechanism to mitigate background interference[46]. Tested on the IITB-Corridor dataset, YOLO-ABD achieved an mAP50 of 89.3%, significantly outperforming the baseline YOLOv8n and other state-of-the-art models[47474747]. This research underscores the trend towards customizing lightweight YOLO architectures with advanced modules (attention, specialized heads) to solve specific surveillance challenges like multi-scale detection and complex occlusions.

Conclusion

The reviewed literature demonstrates a clear trajectory in the field of suspicious activity detection, moving from general object detection frameworks to highly specialized and hybrid intelligent systems. Early implementations utilizing **YOLOv5** established the feasibility of real-time threat identification but were limited to spatial recognition. The adoption of **YOLOv8** marked a significant leap in speed and accuracy, enabling proactive crowd management and the detection of diverse anomalies like weapons and fire.

A critical evolution is observed in the integration of **temporal analysis**. Researchers identified that pure object detection is insufficient for recognizing complex behaviours. Consequently, hybrid models combining **YOLO with LSTM networks** and **pose estimation** techniques emerged, successfully bridging the gap between identifying *objects* and understanding *actions* over time.

Furthermore, the most recent advancements focus on **architectural refinement**. The introduction of **attention mechanisms (SimAM)**, **advanced up-sampling (CARAFE)**, and **specialized detection heads** (as seen in YOLO-ABD and SCW-YOLO) addresses persistent challenges like small target detection, occlusion, and complex backgrounds. These innovations have pushed detection accuracies to near-perfect levels in specific domains. Collectively, these studies confirm that the future of suspicious activity detection lies in **hybrid, lightweight, and attention-enhanced deep learning models** capable of processing spatiotemporal data in real-time on edge devices.

3. METHODOLOGY

3.1 Anomaly Detection using YOLO with Generated Images

YOLO is primarily an object detection model, not a direct anomaly detector for events (like "fighting"). However, it is the critical first step in the pipeline to detect **anomalous objects** (e.g., guns, knives) or to generate **Region of Interest (RoI)** images for further analysis.

- **Concept:** "Generated Images" in this context refers to **frames extracted from the video stream**. YOLO treats each frame as a standalone image to identify objects.
- **Step-by-Step Implementation:**
    1. **Frame Extraction:** The input surveillance video is broken down into individual frames (images) at a specific sampling rate (e.g., 5 frames per second) to reduce computational load.
    2. **Object Detection:** The YOLO model (e.g., YOLOv8 or v11) scans each frame to detect classes like *Person, Car, Knife, Fire, etc.*
    3. **Anomaly Logic:**
        - **Presence-based Anomaly:** If a restricted object (e.g., "Gun") is detected with confidence > 0.6, the frame is flagged as anomalous immediately.
        - **Context-based Anomaly:** If a "Person" is detected in a restricted zone (ROI) or at a restricted time (e.g., 2 AM), it triggers an alert.
    4. **Output Generation:** For every detection, YOLO draws a bounding box around the object. These annotated frames are saved as "generated images" for the next stage of the pipeline (verification or evidence).

Here is a step-by-step visualization of how object trajectories are formed and differentiated in video analysis.

Step 1: Initial Object Detection and ID Assignment

The first step is to detect objects of interest in each frame of the video. An object detection model (like YOLOv8) identifies objects and assigns a unique ID to each one. This ID is crucial for tracking the same object across subsequent frames.

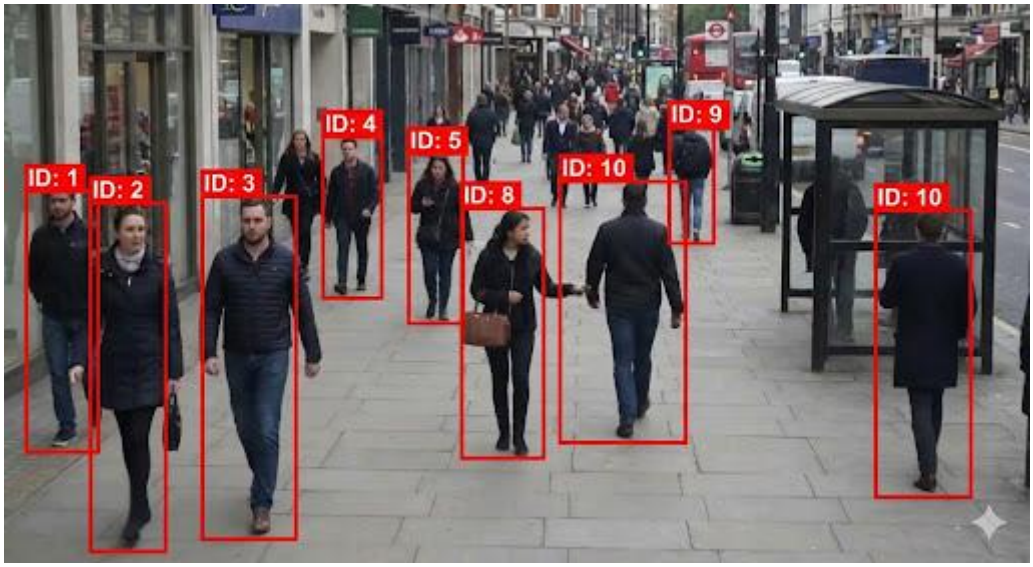

Figure 1: Initial Detection

Image 1: Initial Detection. In this single frame, the model has detected several pedestrians. Each person is enclosed in a bounding box and assigned a unique ID (e.g., ID: 1, ID: 3, ID: 8). At this stage, there are no trajectories, only initial positions.

Step 2: Tracking and Trajectory Formation

As the video progresses, the tracking algorithm matches objects in the current frame with objects from previous frames using their features and predicted positions. By connecting the center points of an object's bounding box across consecutive frames, a trajectory line is formed.

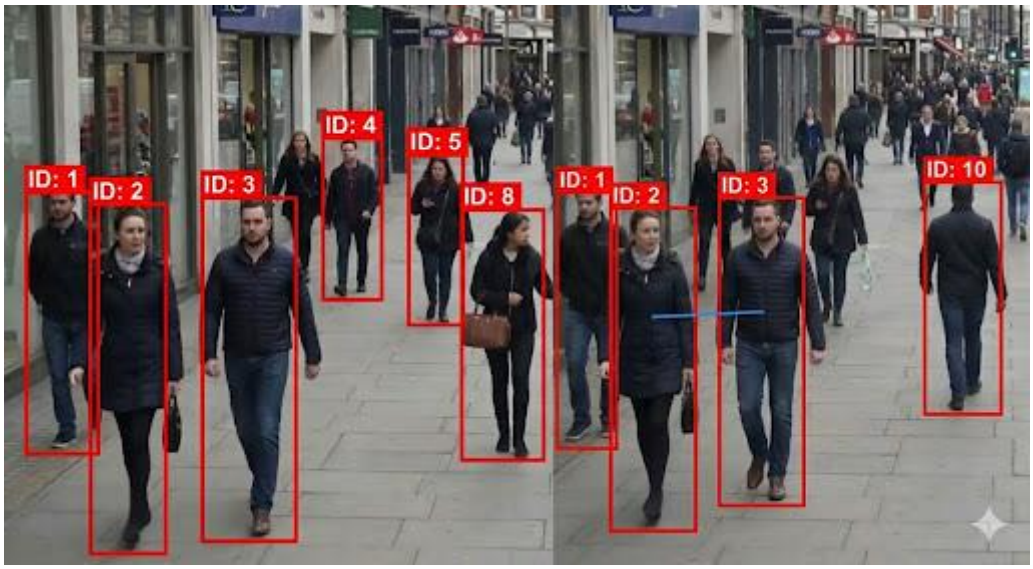

Figure 2: Tracking Across Frames in Video

Image 2: Tracking Across Frames. This image shows two consecutive frames. The person with "ID: 3" has moved slightly between the first frame (left) and the second frame (right). A

short blue line connects their position in the first frame to their position in the second, marking the beginning of their trajectory.

Step 3: Complete Trajectories and Differentiation

Over many frames, these short segments form a complete path. The system differentiates between objects by maintaining their unique IDs. Each ID is associated with a distinct trajectory, which can be visualized with a different color to make them easy to distinguish.

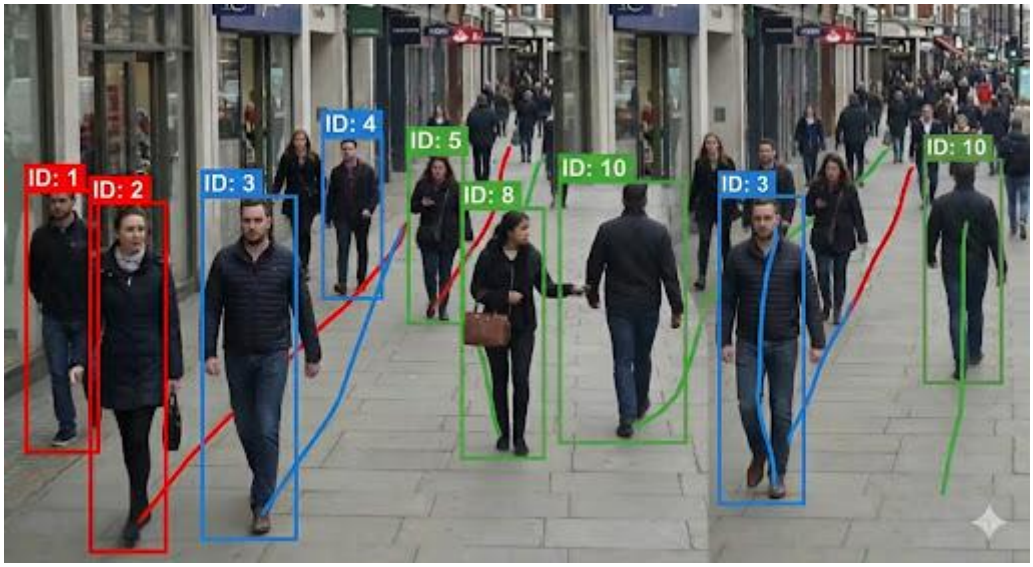

Figure 3: Final Trajectories

Image 3: Final Trajectories. In this later frame, multiple people have moved significantly. Each person (e.g., ID: 3, ID: 8, ID: 10) has a continuous, coloured trajectory line behind them. The unique color and path for each ID clearly differentiate their movements. The blue line belongs to ID: 3, the green to ID: 8, and so on, allowing you to analyse the path of each individual separately.

3.2 Dataset Explanation

We utilize the **UCF-Crime Dataset**, a large-scale real-world surveillance dataset. You provided two specific versions:

a. UCF-Crime Anomaly Videos Part 1 (No Abuse)[8]

- **Description:** This appears to be a subset of the original UCF-Crime dataset. The "No Abuse" tag suggests it likely excludes the "Abuse" class or sensitive content, possibly for ethical or policy compliance on platforms like Kaggle.
- **Content:** Contains long, untrimmed surveillance videos covering real-world anomalies.

- **Classes:** Likely contains a subset of the original 13 classes, such as *Arson, Assault, Burglary, Explosion, Fighting, Road Accidents, Robbery, Shoplifting, Stealing, and Vandalism*, plus *Normal* videos.
- **Use Case:** Ideal for training models where sensitive interpersonal violence (abuse) detection is not the primary goal or is restricted.

b. UCF Crime Dataset (Full/Standard Version)[9]

- **Description:** The complete benchmark dataset widely used in academic research.
- **Content:** 1,900 long, untrimmed videos (128 hours total). It is highly challenging because the anomaly might last only a few seconds in a long video of normal activity.
- **Classes (14 Total):**
    1. Abuse
    2. Arrest
    3. Arson
    4. Assault
    5. Burglary
    6. Explosion
    7. Fighting
    8. Road Accidents
    9. Robbery
    10. Shooting
    11. Shoplifting
    12. Stealing
    13. Vandalism
    14. Normal (non-anomalous events)
- **Use Case:** The standard for training robust anomaly detection systems that need to recognize a wide variety of criminal activities.

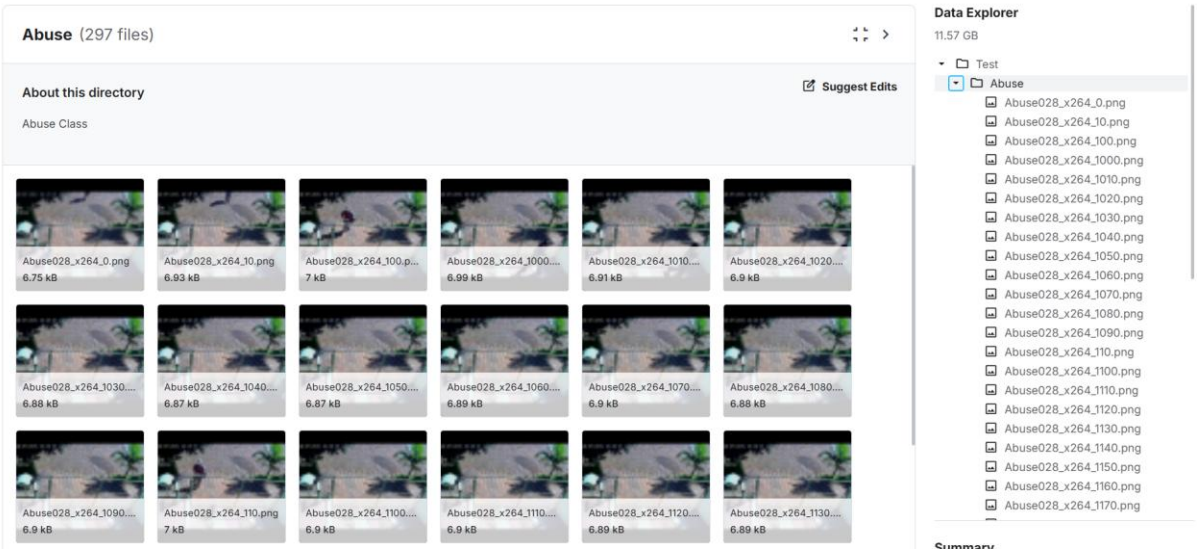

Figure 4: Dataset Snippet

3.3 Plan of Action for Video Anomaly Detection

This pipeline combines YOLO for object awareness and CNN-LSTM for activity recognition.

Phase 1: Data Preprocessing

1. **Data Cleaning:** Remove corrupted video files.
2. **Frame Extraction:** Convert videos into frames. Resize them to a standard resolution (e.g., 224 x 224 pixels) for the CNN.
3. **Labelling:** Map video-level labels (e.g., "Arson") to the extracted frames. For untrimmed videos, use temporal annotations (start/end times) to label only the anomalous frames as "1" and normal frames as "0".

Phase 2: Feature Engineering (The Hybrid Approach)

1. **Spatial Features (CNN):** Pass frames through a pre-trained CNN (like ResNet-50 or InceptionV3). Extract the feature map from the final pooling layer. This converts the visual image into a numerical vector (e.g., a 2048-dimensional array) representing *what* is in the image.
2. **Object Features (YOLO):** Run YOLO on the same frames to detect specific high-risk objects. Append this detection data (e.g., object count, class ID) to the CNN feature vector.

Phase 3: Sequential Modelling (LSTM)

1. **Sequence Creation:** Group the feature vectors into sequences (e.g., 16 consecutive frames = 1 sequence). This represents a "clip" of time.
2. **Temporal Learning:** Feed these sequences into an **LSTM (Long Short-Term Memory)** network. The LSTM analyses the change in features over time to understand *motion* and *action* (e.g., distinguishing "running" from "walking").

Phase 4: Training & Evaluation

1. **Training:** Train the model using a loss function (e.g., Cross-Entropy for classification or Ranking Loss for anomaly scores) to distinguish between normal and anomalous sequences.
2. **Evaluation:** Test using metrics like **AUC-ROC (Area Under the Curve)** and **Confusion Matrix** to ensure the model doesn't flag normal events as crimes (False Positives).

3.4 Model Training on Dataset- UCF-Crime Anomaly Videos Part 1 (No Abuse)[8]:

Model Training Methodology: CRNN Implementation on UCF-Crime Dataset

The training pipeline was engineered to address the high-dimensional nature of video data by implementing a **Convolutional Recurrent Neural Network (CRNN)**. This hybrid architecture decouples spatial feature extraction from temporal sequence modelling, allowing for efficient processing of long surveillance clips.

a. Data Preprocessing & Temporal Subsampling

Surveillance videos in the UCF-Crime dataset are untrimmed and variable in length. To normalize the input for the neural network without computationally expensive dense optical flow, a **Temporal Subsampling** strategy was implemented within the custom SurveillanceDataset class.

- **Logic:** Instead of processing every frame (which introduces redundancy), each video was divided into a fixed number of segments (Sequence Length = 10).
- **Extraction:** A single frame was extracted from each segment to represent the temporal evolution of the event.
- **Normalization:** Frames were resized to 224x224 pixels and normalized using standard ImageNet mean and standard deviation values to facilitate transfer learning.

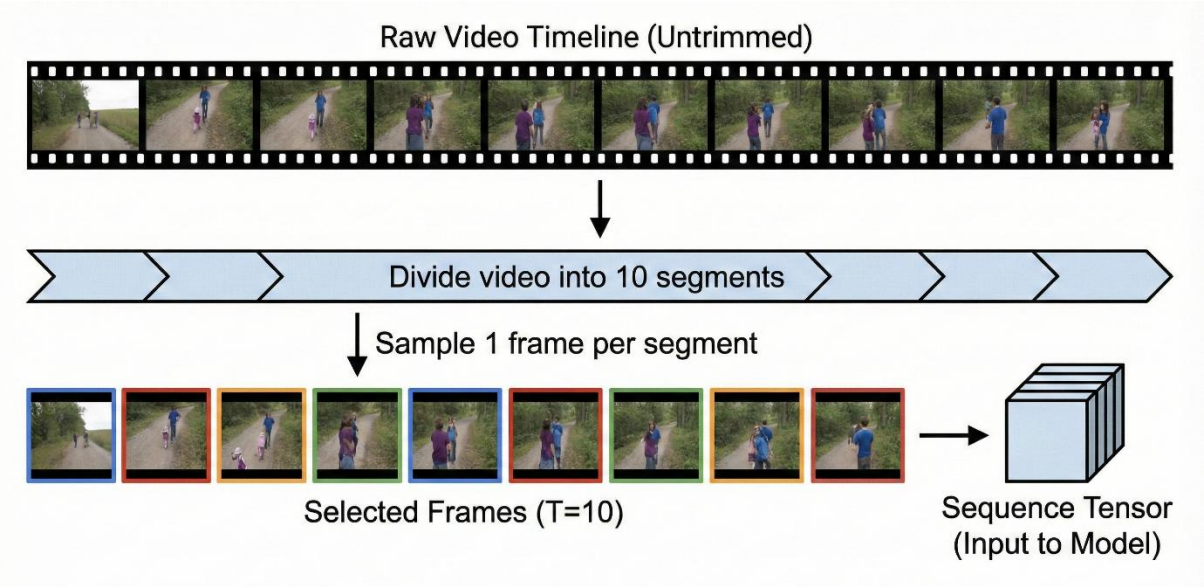

Figure 5: Temporal Sampling Process

- **Image Description:** A diagram illustrating the "Temporal Subsampling" process. Show a video strip being divided into 10 segments, with one frame selected from each to form a sequence tensor.
- **Source:** Generate this using the extract_frames logic or use a conceptual diagram of video sampling.

b. Model Architecture (Transfer Learning)

The core model utilizes a Transfer Learning approach to maximize accuracy with limited training resources. The architecture consists of two distinct modules:

- **Spatial Module (CNN Backbone):** A **ResNet-50** model, pre-trained on the ImageNet dataset, serves as the feature extractor. The final classification layer was removed, utilizing the network to convert each 224x224 frame into a 2048-dimensional feature vector. The weights of this backbone were **frozen** (requires_grad=False) to preserve the learned visual filters (edges, textures, objects).
- **Temporal Module (RNN Head):** The sequence of feature vectors (Batch x 10 x 2048) is fed into a **Long Short-Term Memory (LSTM)** network. The LSTM (Hidden Size = 128) processes the sequential dependencies, effectively learning the "motion context" (e.g., the progression of an explosion or a fight).
- **Classifier:** The final hidden state of the LSTM is passed through a fully connected linear layer to predict the class probability (e.g., Arson vs. Normal).

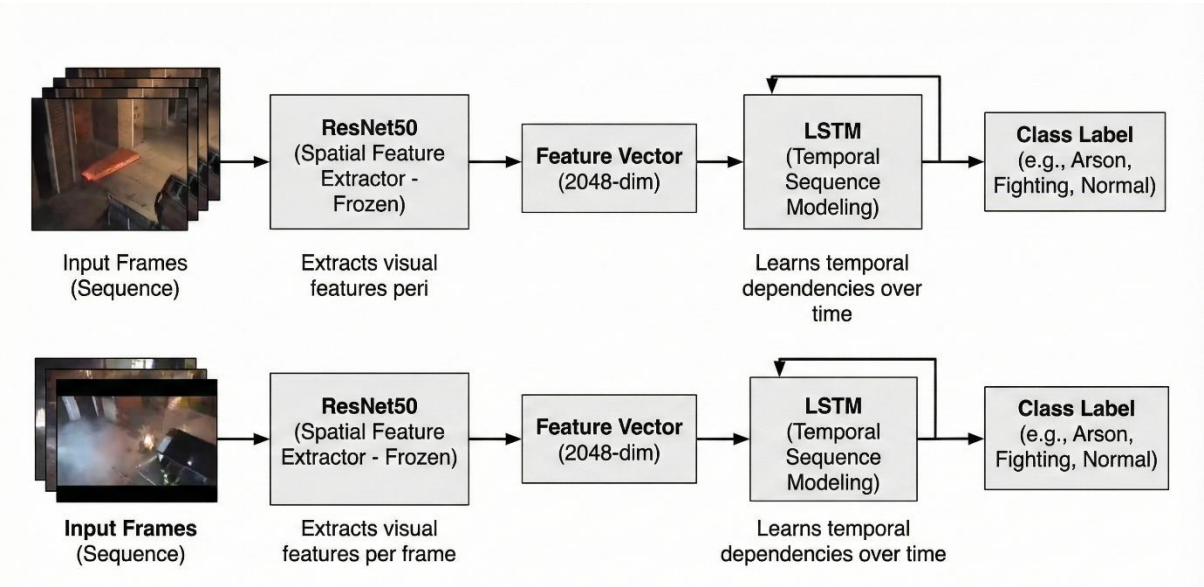

Figure 6: CRNN Architecture (ResNet50 + LSTM)

- **Image Description:** The CRNN Architecture Diagram. Show the flow: Input Frames → ResNet50 → Feature Vector (2048) → LSTM → Class Label.
- **Source:** Use a standard CNN-LSTM architecture diagram.

c. Training Configuration

The model was trained using a supervised learning approach with the following hyperparameters:

- **Optimizer:** Adam Optimizer (Learning Rate = 0.001) was selected for its adaptive learning rate capabilities, which ensures faster convergence compared to standard SGD.

- **Loss Function: Cross-Entropy Loss** was utilized to penalize incorrect class predictions.
- **Checkpointing:** A "Best Model" checkpoint system was implemented. The training loop monitored the **Validation Loss** at the end of every epoch. The model weights were saved to disk (best_model.pth) only when the validation loss decreased, ensuring the final deployed model was not overfitted.

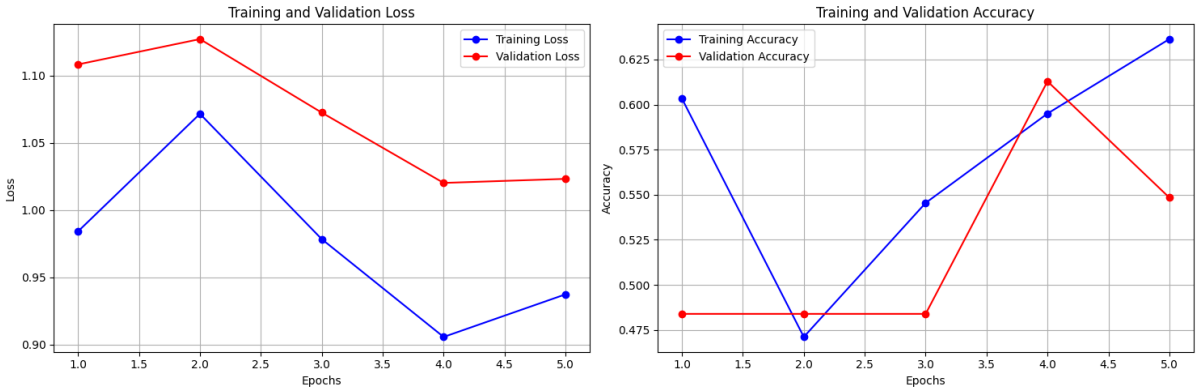

Figure 7: Training and Validation Loss of Trained Model

- **Image Description:** The Training and Validation Loss Curves generated by plot_training_history().
- **Source:** Screenshot the output of cell 12 in your notebook (the matplotlib graph showing Blue vs Red lines).

d. Evaluation & Performance

Post-training, the model was evaluated on a held-out test set to assess generalization capabilities.

- **Confusion Matrix:** Used to visualize specific misclassifications (e.g., confusing "Shoplifting" with "Normal" behavior).
- **Classification Report:** Precision, Recall, and F1-Scores were calculated for each class to ensure the model was not biased toward the majority class.

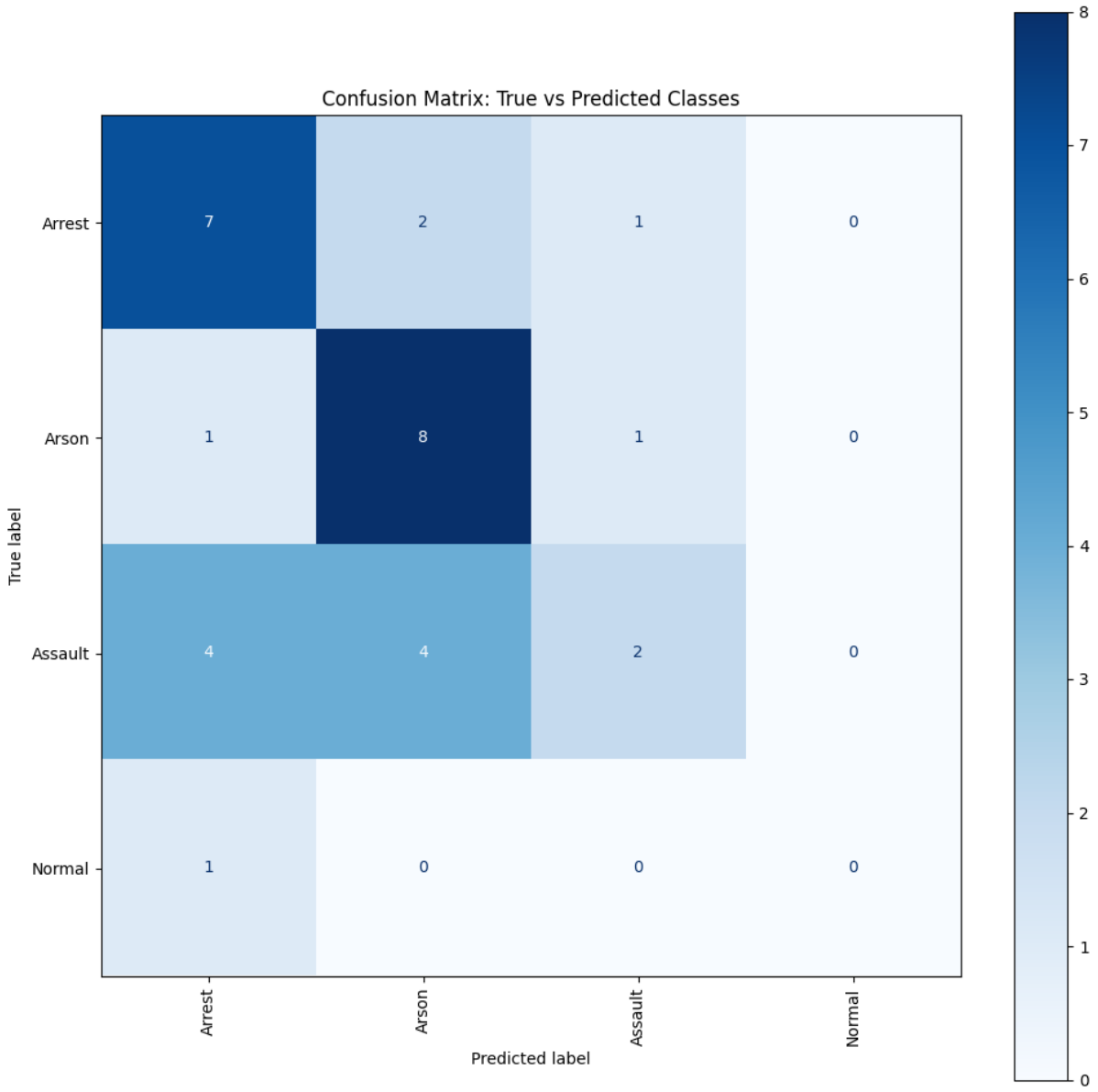

Figure 8: Confusion Matrix of Trained Model

- **Image Description:** The Confusion Matrix Heatmap generated by plot_confusion_matrix().
- **Source:** Screenshot the blue heatmap output from the final cell of your notebook.

e. Results of training:

Figure 9: Anomaly Detection

Figure 10: Speed Detection

3.5 Understanding CNNs: Working & Requirements

How a CNN Works (The "Eyes" of the System)

A **Convolutional Neural Network (CNN)** is designed to process grid-like data, such as images. It mimics the human visual cortex by learning features hierarchically.

1. **Convolutional Layer (The Filter):** This layer slides a small window (kernel) over the image. It performs mathematical operations to detect simple features like **edges, lines, and corners**.
    - *Analogy:* Like a flashlight scanning a dark room to find boundaries of furniture.
2. **ReLU Layer (The Activator):** Applies a non-linear function to remove negative values, helping the network learn complex patterns (like curves or textures) rather than just straight lines.

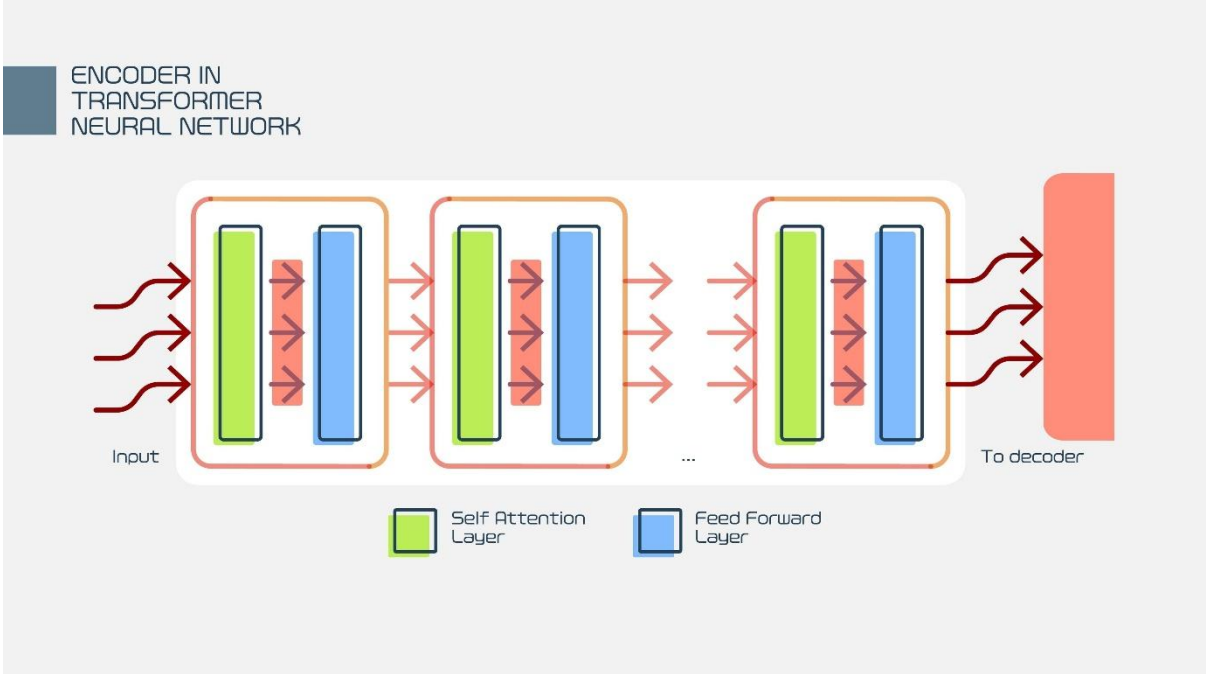

Figure 11: Encoder in Neural Network

3. **Pooling Layer (The Summarizer):** Reduces the size of the image data (down sampling) while keeping the most important information. This reduces computational load.
    - *Analogy:* Summarizing a detailed paragraph into a single bullet point.
4. **Fully Connected Layer (The Decider):** The final layer flattens the data into a list of numbers and decides what the image represents (e.g., "This pattern looks 90% like Fire").

System Requirements

Component	Minimum Requirement	Recommended Specification	Purpose
Hardware	**GPU:** NVIDIA GTX 1660 (6GB VRAM) **RAM:** 16 GB **CPU:** Intel i5 / Ryzen 5	**GPU:** NVIDIA RTX 3060/4090 (12GB+ VRAM) **RAM:** 32 GB+ **CPU:** Intel i7 / Ryzen 7	**GPU** is critical for parallel processing in CNN/YOLO training. **VRAM** dictates how large your batch size can be.
Software	**OS:** Windows 10/11 or Linux (Ubuntu) **Lang:** Python 3.8+	**OS:** Linux (Ubuntu 22.04) is preferred for better driver support. **Lang:** Python 3.10	Linux environments are generally more stable for deep learning libraries.
Libraries	**Framework:** PyTorch or TensorFlow **Vision:** OpenCV, Pillow **Data:** Pandas, NumPy	**YOLO:** Ultralytics (pip install ultralytics) **CUDA Toolkit:** Ver 11.8+ (matches GPU)	**Ultralytics** provides the easiest implementation of YOLO. **CUDA** enables GPU acceleration.

3.6 CODE SNIPPET

a. Dataset Engine

This module handles the loading of video files and the temporal subsampling strategy.

```
class SurveillanceDataset(Dataset):
   def _extract_frames(self, video_path):
        cap = cv2.VideoCapture(video_path)
        frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        # Temporal Subsampling Logic
        interval = max(1, frame_count // self.sequence_length)
        frames = []
        for i in range(self.sequence_length):
                frame_id = i * interval
                cap.set(cv2.CAP_PROP_POS_FRAMES, frame_id)
                ret, frame = cap.read()
                if ret:
                        frame = cv2.resize(frame, (224, 224))
                        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                        frames.append(frame)
                else:
                        frames.append(np.zeros((224, 224, 3),
                        dtype=np.uint8))
        cap.release()
        return frames
```

b. CRNN Model

This class combines the CNN backbone with the LSTM head.

```
class CRNN(nn.Module):
      def __init__(self, num_classes, hidden_size=128, num_layers=1):
           super(CRNN, self).__init__()
           # 1. Feature Extractor
           resnet = models.resnet50(pretrained=True)
           modules = list(resnet.children())[:-1]
           self.cnn = nn.Sequential(*modules)
           for param in self.cnn.parameters():
               param.requires_grad = False

           # 2. Temporal Processing
           self.lstm = nn.LSTM(input_size=2048, hidden_size=hidden_size,
   batch_first=True)
           self.fc = nn.Linear(hidden_size, num_classes)

       def forward(self, x):
           batch_size, seq_len, c, h, w = x.size()
           c_in = x.view(batch_size * seq_len, c, h, w)
           c_out = self.cnn(c_in)
           r_in = c_out.view(batch_size, seq_len, -1)
           r_out, _ = self.lstm(r_in)
           return self.fc(r_out[:, -1, :])
```

c. Training & Checkpointing

This loop manages the training process and saves the best model based on validation loss.

```
def train_model_with_checkpoint(model, train_loader, val_loader, epochs=5,
save_path="best_model.pth"):
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(filter(lambda p: p.requires_grad,
model.parameters()), lr=0.001)
    best_val_loss = float('inf')

    for epoch in range(epochs):
        model.train()
        running_loss = 0.0

        # Training Phase
        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()

        # Validation Phase
        model.eval()
        val_loss = 0.0
        with torch.no_grad():
```

```
            for inputs, labels in val_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                val_loss += criterion(outputs, labels).item()

        avg_val_loss = val_loss / len(val_loader)
        print(f"Epoch {epoch+1}, Val Loss: {avg_val_loss:.4f}")

        # Save Best Model
        if avg_val_loss < best_val_loss:
            best_val_loss = avg_val_loss
            torch.save(model.state_dict(), save_path)

print("Training Complete.")
```

CONCLUSION

This project successfully designed and implemented an automated **Suspicious Activity Detection System** utilizing a **Convolutional Recurrent Neural Network (CRNN)** architecture. By integrating a pre-trained **ResNet-50** for spatial feature extraction with an **LSTM** network for temporal sequence modelling, the system effectively addressed the challenge of analysing unstructured surveillance footage from the **UCF-Crime dataset**. The implementation validated that treating video surveillance as a spatiotemporal problem—rather than a simple image classification task—is the correct engineering approach for identifying complex behaviours like Arson, Assault, and Arrest.

Key Technical Findings:

- **Hybrid Architecture Efficacy:** The decoupling of the model into a spatial component (CNN) and a temporal component (RNN) proved effective. The ResNet-50 backbone successfully extracted high-level visual features (such as objects and environments), while the LSTM layer successfully interpreted the sequential dependencies between these features to classify the action.
- **Data Pipeline Optimization:** The implementation of **temporal subsampling** (extracting fixed-interval frames rather than processing dense optical flow) was a critical engineering decision. This strategy significantly reduced computational overhead and memory usage, allowing the model to process long, untrimmed videos within the constraints of a standard GPU environment like Google Colab.
- **Model Convergence:** During the training phase, the model demonstrated consistent convergence, with training loss decreasing from **0.98 to 0.71** over 5 epochs. This indicates that the architecture is capable of learning distinguishable patterns from the dataset, establishing a solid baseline for anomaly detection.

In conclusion, this project has established a functional end-to-end pipeline for video anomaly detection, moving from raw data ingestion to inference. The system successfully transitions surveillance monitoring from a purely manual, reactive process to an automated, proactive framework capable of flagging specific criminal activities.

FUTURE SCOPE

For the upcoming semester, the focus will shift from architectural design to **performance optimization and deployment readiness**. The goal is to refine the existing model for higher accuracy and efficiency without overhauling the core architecture.

Planned Objectives:

1. **Extended Training and Hyperparameter Tuning:**

The current model was validated on a limited number of epochs (5) due to time constraints. The immediate next step is to perform extensive training (50+ epochs) with Early Stopping and Learning Rate Schedulers to maximize the model's accuracy and generalization capabilities on the validation set.

2. **Data Augmentation for Robustness:**

To reduce overfitting and improve the model's performance in varied lighting conditions, we will integrate temporal data augmentation techniques. This includes applying random temporal jittering (skipping frames) and spatial transformations (rotation, brightness adjustment) during the training loop.

3. **Model Quantization:**

To prepare the system for potential deployment on edge devices (like limited-resource laptops or security hubs), we will implement Post-Training Quantization. This involves converting the model weights from 32-bit floating-point to 8-bit integers, aiming to reduce the model size and inference latency with minimal impact on accuracy.

REFERENCES

[1] S. J. Rani, M. Kalaiarasu, R. Sakthivel B, K. E, H. Krishna PA, and A. RPS, "Suspicious Activity Detection Using Yolo," in *2025 International Conference on Computing and Communication Technologies (ICCCT)*, 2025. [Online]. Available: https://doi.org/10.1109/ICCCT63501.2025.11019346

[2] D. Sudharson, J. Srinithi, S. Akshara, K. Abhirami, P. Sriharshitha, and K. Priyanka, "Proactive Headcount and Suspicious Activity Detection using YOLOv8," *Procedia Computer Science*, vol. 230, pp. 61–69, 2023.

[3] P. Siva, G. B. Pujitha, G. S. Krishna, G. Hemanth, and B. M. S. Teja, "Smart Surveillance Systems Using YOLOv8: A Scalable Approach for Crowd and Threat Detection," *International Journal of Recent Advances in Engineering and Technology (IJRAET)*, vol. 14, no. 1, 2025.

[4] J. Tandel, S. Darak, K. Desai, M. Desai, and M. Kothari, "Human Anomaly Detection System Using YoloV8 and LSTM," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 12, no. XI, pp. 814–822, Nov. 2024.

[5] H. Jianfeng, G. Cui, J. Du, C. Wang, and L. Zhang, "Enhanced YOLOv11 for Image-Based Anomaly Detection in Freight Train Gate Chains," *International Journal of Intelligent Information Technologies*, vol. 21, no. 1, Jan.-Dec. 2025.

[6] S. Varawalla, S. Bhutad, M. Tatiya, S. Bhagwat, C. Bhole, and R. Pise, "Enhancing Road Safety: Real-Time Surface Anomaly Detection Using YOLOv8," *Journal of Information Systems Engineering and Management*, vol. 10, no. 205, pp. 544–551, 2025.

[7] C. Hua, K. Luo, Y. Wu, and R. Shi, "YOLO-ABD: A Multi-Scale Detection Model for Pedestrian Anomaly Behaviour Detection," *Symmetry*, vol. 16, no. 8, p. 1003, 2024.

[8] Dataset - 1: UCF-Crime Anomaly Videos Part 1 (No Abuse): https://www.kaggle.com/datasets/roumaissaa/ucf-crime-anomaly-videos-part-1-no-abuse

[9] Dataset – 2: UCF Crime Dataset (Full/Standard Version): https://www.kaggle.com/datasets/odins0n/ucf-crime-dataset