NYU | TANDON SCHOOL OF ENGINEERING

Big Data

CS-GY 6513, Spring 2024

VigilEye

Project Report

Submitted by:

Pratham Shah	ps4896
Jay Desai	jd5558
Jugal Pumbhadia	jp6988
Samkit Shah	ss17651

Submitted Under:

Prof. Juan Rodriguez

Table of Contents

Introduction

In our project, we aim to develop a system capable of detecting criminal or violent activities in real-time video streams. This endeavor addresses a pressing need for advanced surveillance and security measures in today's society. To achieve this goal, we employ a sophisticated approach utilizing Big Data technologies such as Apache Spark and Apache Kafka.

Firstly, we compile a diverse dataset comprising training videos depicting various criminal behaviors, including protests and arson. Apache Spark is instrumental in processing this vast amount of video data efficiently, converting it into a format suitable for analysis. By breaking down the videos into images, we pave the way for our system to recognize patterns and anomalies effectively.

Next, we leverage a deep learning model to analyze the images and identify potential threats in real-time. These models are fine-tuned using the training dataset, enhancing their accuracy in detecting anomalous behaviors. To assess the system's performance, we employ a variety of metrics, providing insights into its ability to distinguish between normal and abnormal activities. Finally, we deploy our solution using Apache Kafka, enabling seamless streaming of live video feeds from platforms like YouTube. This real-time monitoring allows for swift detection and response to security threats, ultimately contributing to enhanced public safety.

Why is this Big Data?

The utilization of big data principles and technologies in this scenario is evident across various dimensions, including the dataset size, data processing requirements, and technological infrastructure. Here's a detailed explanation of why this project falls under the realm of big data:

Volume of Data:

The dataset comprises 140 hours of surveillance videos covering 14 different anomalies, along with the addition of a new class, protests, consisting of 100+ videos. This sheer volume of video data translates to a massive amount of raw information.

Converting these videos into images results in over 50,000+ individual images, significantly increasing the volume of data to be processed and analyzed.

Variety of Data:

The dataset encompasses diverse types of surveillance videos capturing various real-world anomalies, ranging from abuse and assault to robbery and vandalism. Additionally, the inclusion of protests introduces a new category, adding to the dataset's diversity.

Each video is likely to differ in terms of content, duration, lighting conditions, camera angles, and other factors, contributing to the variety of data to be handled.

Velocity of Data:

The use of Apache Kafka for creating a pipeline of live stream videos suggests a continuous influx of new data in real-time. This live stream of video data adds a velocity dimension to the dataset, requiring rapid processing and analysis to keep up with the incoming data flow.

Additionally, the process of converting live video streams into image frames further emphasizes the need for real-time or near-real-time data processing capabilities.

Data Processing Requirements:

The utilization of Apache Spark for parallelizing and efficiently processing every 10th frame from video files highlights the need for distributed computing frameworks to handle data-intensive tasks effectively.

Processing large-scale video data involves computationally intensive operations such as frame extraction, feature extraction, and model inference, necessitating scalable and high-performance computing infrastructure.

Technological Infrastructure:

The adoption of Apache Spark and Apache Kafka underscores the reliance on advanced big data technologies to manage, process, and analyze large volumes of data efficiently.

Apache Spark's ability to distribute data processing tasks across multiple nodes in a cluster enables parallelization and scalability, making it well-suited for handling big data workloads.

Similarly, Apache Kafka facilitates the creation of robust, fault-tolerant data pipelines for streaming data ingestion and processing, supporting real-time data processing requirements.

In summary, the combination of a large volume of diverse data, real-time data streams, and the need for scalable data processing and analysis underpins the classification of this project as a big data initiative. The utilization of advanced technologies like Apache Spark and Apache Kafka further solidifies its classification as a big data project, highlighting the significance of big data principles in addressing complex data challenges.

Dataset

The dataset utilized for our project is the UCF Crime Dataset, comprising lengthy, untrimmed surveillance videos spanning over 140 hours. These videos encapsulate 13 real-world anomalies, including Abuse, Arrest, Arson, Assault, Road Accident, Burglary, Explosion, Fighting, Robbery, Shooting, Stealing, Shoplifting, and Vandalism, chosen for their significant impact on public safety. Additionally, the dataset contains normal scenario videos.

In an extension to the existing categories, we incorporated a new class of videos: Protests, totaling over 100 videos. This addition was motivated by the prevalence of protests in both urban areas and university campuses.

In total, this amalgamation of data surpasses 100 GB in size, and has 15 different classes.

Fig 1: Sample Frames from across different classes like protest, shoplifting, vandalism, accident

Data Preprocessing

1. **Environment Setup:**

The first phase of our data preprocessing involved setting up the necessary environment to facilitate efficient data handling. This step encompassed the installation of essential software packages, including the Java Development Kit (JDK), Apache Spark, and Python libraries such as findspark, pyspark, and py4j. These tools provided the foundational framework for subsequent data processing tasks.

2. **Frame Extraction:**

In this phase, we extracted frames from the video files within our dataset to facilitate further analysis. Leveraging a custom Python function, we selectively captured every 10th frame from each video. This approach balanced processing efficiency with information retention, allowing us to convert the video data into a more manageable format conducive to subsequent analysis.

3. **Folder-Specific Processing:**

Our dataset was organized into multiple folders, each representing distinct activity types such as protests, arson, and assault. We conducted folder-specific processing, wherein each folder underwent individual frame extraction. This meticulous approach ensured that preprocessing was tailored to the unique characteristics of each activity type, facilitating accurate analysis and model training.

4. **Parallel Processing with Apache Spark:**

To expedite the preprocessing task and handle our extensive video dataset efficiently, we leveraged the parallel processing capabilities of Apache Spark. By distributing the workload across multiple computing resources, Spark enabled simultaneous processing of multiple folders, significantly reducing processing time. This parallelization ensured efficient handling of our data, laying the groundwork for timely analysis and insights extraction.

In summary, the data preprocessing phase played a pivotal role in preparing our dataset for subsequent analysis. Through environment setup, frame extraction, folder-specific processing, and parallelization with Apache Spark, we transformed raw video data into a structured format amenable to analysis, setting the stage for meaningful insights into anomalous behaviors in video streams.

Model Training

The model architecture presented here is designed for robust multiclass prediction, particularly tailored for the classification of 15 distinct labels, including a variety of real-world anomalies and the addition of a new class: Protests.

1. **FeatureExtractor**: At the core of the architecture lies the FeatureExtractor module, which leverages the DenseNet121 architecture pre-trained on a large-scale image dataset. DenseNet121 stands out for its dense connectivity pattern, where each layer receives feature maps from all preceding layers, promoting feature reuse and enhancing model efficiency. By removing the final fully connected layers, the FeatureExtractor isolates the convolutional backbone, enabling it to extract high-level features from input images efficiently.

2. **Classifier**: Following the feature extraction stage, the Classifier component takes over to process the extracted features and make predictions. It comprises a sequence of fully connected layers, each followed by a rectified linear unit (ReLU) activation function to introduce non-linearity. These layers are crucial for transforming the high-dimensional feature representations into a compact feature space that can be effectively classified. To mitigate overfitting, dropout layers are incorporated, which randomly deactivate a fraction of neurons during training, preventing excessive reliance on specific features.

3. **FinalModel**: The FinalModel encapsulates both the FeatureExtractor and Classifier modules, forming an end-to-end trainable architecture. This integration facilitates seamless feature extraction and classification in a unified framework. During inference, input images are first passed through the FeatureExtractor to obtain informative feature representations. These features are subsequently fed into the Classifier, which produces class predictions based on learned patterns.

In addition to the model architecture, the training process involves optimizing the model parameters using the stochastic gradient descent (SGD) optimizer with a specified learning rate

(LR). The choice of optimizer and learning rate plays a crucial role in determining the convergence speed and final performance of the model. Furthermore, the cross-entropy loss function is employed to quantify the disparity between predicted class probabilities and ground truth labels, guiding the model towards accurate classification.

Overall, this meticulously designed architecture, combining the power of DenseNet121 for feature extraction with a tailored classifier for multiclass prediction, showcases a sophisticated approach to anomaly detection and event classification in surveillance videos, ensuring enhanced public safety monitoring and management.

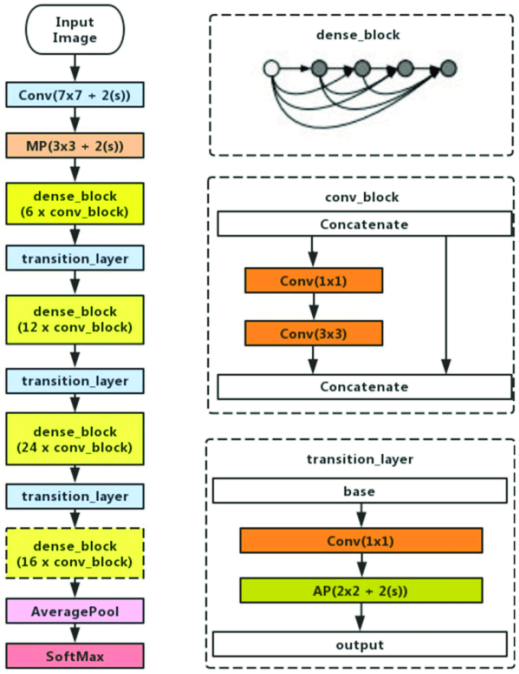

Fig 2: DenseNet121 Architecture

Prediction and Evaluation

The model is designed to predict labels for image data, with a focus on identifying violent activities or specified events within surveillance videos. Evaluation of the model's performance involves rigorous assessment using various metrics, including the AUROC curve, ROC AUC score, accuracy, and classification accuracy per class.

During the prediction process, the model operates in evaluation mode, making predictions on the test dataset. These predictions are then compared against the true labels to evaluate the model's effectiveness.

Key evaluation metrics include:

- **AUROC Curve and ROC AUC Score**: These metrics provide insights into the model's ability to discriminate between positive and negative classes. A higher AUROC score indicates better performance, with the curve's shape and the area under it (AUC) signifying the model's discriminative power.

- **Accuracy**: This metric measures the overall correctness of the model's predictions. While straightforward to interpret, accuracy may not capture performance variations across different classes, especially in the presence of imbalanced class distributions.

- **Classification Accuracy per Class**: Examining accuracy per class reveals the model's strengths and weaknesses for specific event categories. Discrepancies in accuracy across classes highlight areas for improvement or specialization.

Prioritizing the AUROC score over accuracy is justified by its effectiveness in evaluating class-specific performance and resilience to class imbalance.

In interpreting the results, the model demonstrates varying performance across different classes:

- **Strong Performance**: The model excels in identifying certain classes, such as arrest, arson, assault, burglary, shoplifting, and stealing, demonstrating high discriminative power.

- **Moderate Performance**: For classes like abuse, protest, road accidents, robbery, and vandalism, the model performs reasonably well, with room for improvement.

- **Challenges**: Certain classes, including explosion, fighting, and shooting, pose challenges for the model, indicating areas requiring further optimization or additional training data.

Understanding these performance variations is crucial for refining the model and ensuring its effectiveness across all classes in real-world scenarios.

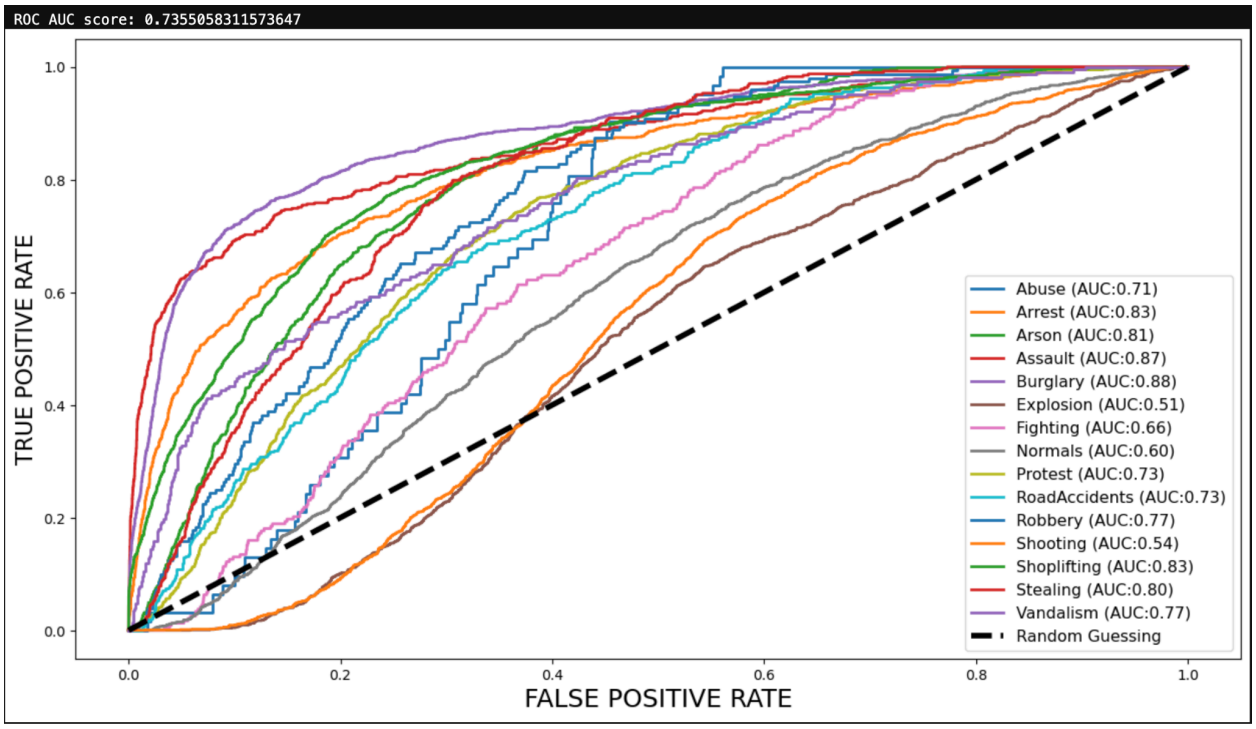

Fig 3: AUROC curve

Streaming Data using Apache Kafka

1. **Kafka Producer Setup:**

The process commences with configuring a Kafka producer tasked with streaming live video data from Bryant Park of New York City. Leveraging the KafkaProducer module, the producer establishes communication with the locally hosted Kafka broker. Continuously monitoring a designated directory for newly created video files via a FileSystemEventHandler, the producer initiates frame extraction upon detection. Utilizing OpenCV's VideoCapture module, every 10th frame from the video is extracted and encoded into JPEG format using cv2.imencode. These encoded frames are subsequently transmitted to the specified Kafka topic, 'frame_topic', for further processing.

2. **Frame Extraction and Transmission:**

Upon the identification of a new video file, the VideoFileHandler class is invoked to manage video processing and frame extraction. Through the process_video method, the VideoCapture module is utilized to iterate through the video frames, extracting every 10th frame. Following extraction, each frame is encoded into JPEG format using cv2.imencode and dispatched to the Kafka broker via the KafkaProducer module. This seamless transmission of frames to the Kafka topic facilitates real-time streaming of video data.

3. **Kafka Consumer Setup:**

Concurrently with the Kafka producer, a Kafka consumer is configured to receive and process the streamed video frames. Subscribing to the 'frame_topic', the consumer initiates a loop to continuously await incoming messages. Upon message reception, the consumer decodes the JPEG-encoded frame, preprocesses the image, and subjects it to inference using a pre-trained deep learning model. The model predicts the activity depicted in the frame, enabling real-time anomaly detection within the video stream.

4. **Model Inference and Prediction:**

Within the Kafka consumer loop, each received frame undergoes preprocessing and inference utilizing a pre-trained deep learning model. Comprising a feature extractor and classifier, the model predicts the class label corresponding to the observed activity in the frame. Leveraging a predefined list of class labels, the predicted label is determined and displayed, facilitating real-time identification of anomalous behaviors.

5. **Real-Time Anomaly Detection:**

Integration of Kafka streaming with the deep learning model empowers real-time anomaly detection within the streamed video data. Through continuous processing and analysis of incoming frames, the system promptly identifies and classifies anomalous activities like protests, arson, assault etc. This real-time detection capability enhances situational awareness and enables timely intervention against potential security threats.

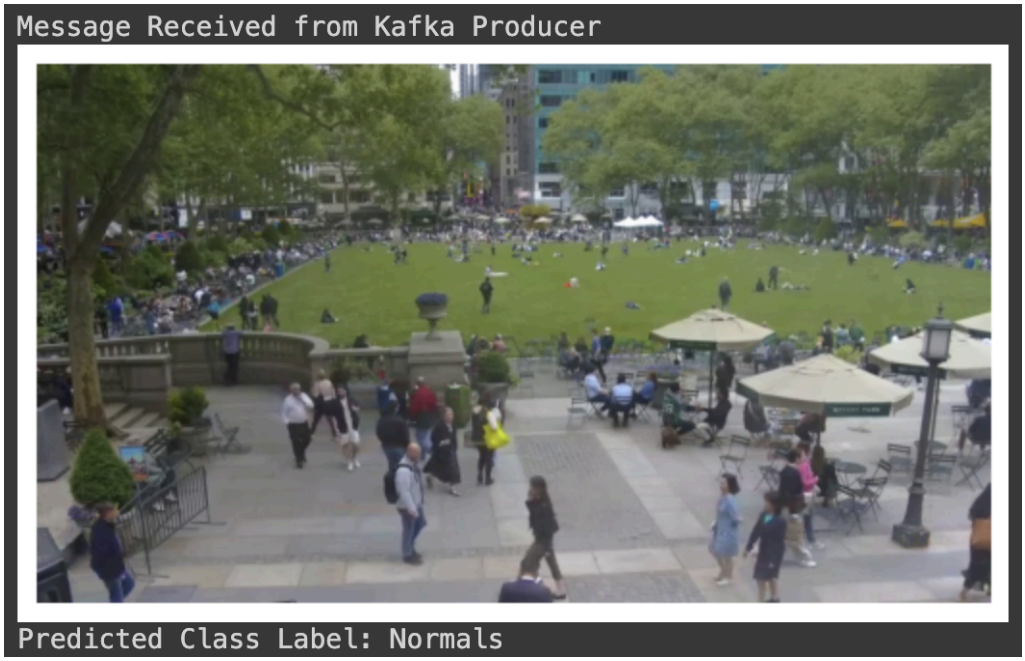

Fig 4: The model predicting the scenario after Kafka creates image frames from incoming livestream video.

Future Work:

As we conclude this project, there are several avenues for future exploration and enhancement to further advance the capabilities and usability of our system.

1. **Training with Large Real-Time Dataset:**

One key area for future work involves enhancing the accuracy and generalization of our anomaly detection model. Currently, our model has been trained on a diverse dataset of criminal and violent activities. However, to improve its effectiveness in real-world scenarios, training the model with a larger, real-time dataset is imperative. By incorporating a continuous stream of updated data, our model can adapt to evolving patterns and nuances, leading to improved accuracy and robustness in anomaly detection.

2. **Development of Comprehensive Software Solution:**

In addition to enhancing the model's training dataset, there is a need to develop a comprehensive software solution encompassing both frontend and backend components. This software would serve to streamline the user experience (UX) and facilitate seamless interaction with our system. The frontend interface would provide intuitive controls and visualizations, allowing users to monitor live video streams and view real-time anomaly alerts. Meanwhile, the backend infrastructure would handle data processing, model inference, and system management tasks, ensuring efficient and reliable operation.

3. **Integration of Alert System:**

To further enhance the usability and effectiveness of our software, an alert system should be integrated. This system would enable real-time notification and response to detected anomalies. Upon identification of a suspicious activity by the anomaly detection model, the software would trigger an alert, notifying designated stakeholders via email, SMS, or push notifications. Additionally, the alert system could include features such as event logging and escalation protocols to ensure timely and appropriate action in response to potential security threats.

By pursuing these future initiatives, we aim to further elevate the performance and usability of our system, ultimately contributing to enhanced public safety and security in real-world applications.

Conclusion:

In conclusion, the VigilEye project represents a significant endeavor in leveraging Big Data technologies for real-time surveillance and security monitoring. By harnessing Apache Spark and Apache Kafka, we have developed a sophisticated system capable of detecting criminal or violent activities in live video streams. Our approach involves preprocessing extensive video datasets, training deep learning models, and integrating real-time streaming capabilities for seamless monitoring.

Through meticulous data preprocessing and model training, we have laid the foundation for accurate anomaly detection and event classification. Our model showcases robust performance across a diverse range of classes, with varying degrees of accuracy. Integration with Apache Kafka enables real-time streaming of video data, facilitating prompt identification and response to security threats.

Looking ahead, future work focuses on enhancing model accuracy through training with larger real-time datasets and developing a comprehensive software solution to enhance user experience. Additionally, the integration of an alert system will enable timely notification and response to detected anomalies, further enhancing public safety and security.

Overall, the VigilEye project embodies the intersection of cutting-edge technology and public safety, paving the way for enhanced surveillance and threat detection in real-world scenarios.