AI Model for Detecting Abnormal Behaviour

Detailed System Design and Implementation Report

GROUP 1-C

Name	Position	Email	Phone No
Tran Quoc Dung	Software Development	dungtqswh00420@fpt.edu.vn	0943910306
Pham Hoang Duong	Data Research	duongphswh00843@fpt.edu.vn	0902162467
Nguyen Thai Son	Data Research	sonntswh00509@fpt.edu.vn	0902950403
Duong Quoc Trung	AI Researcher	trungdqswh00902@fpt.edu.vn	0962579876
Do Tuan Dat	Software Development	datdtswh00592@fpt.edu.vn	0865411803

COS40005, Computing Technology Project A, May-2024, 16/07/2024

Table of Contents

DOCUMENT CHANGE CONTROL

Version	Date	Authors	Summary of Changes
1.00	16/07	All	Create an initial draft document.
1.10	27/07	Dũng, Dương, Sơn	Finalizing the document.

DOCUMENT SIGN OFF

Name	Position	Signature	Date
Tran Quoc Dung	Software Developer, Writer		16/07/2024
Pham Hoang Duong	Data Research, Software Dev	Dương	17/07/2024
Nguyen Thai Son	Data Researcher	Sơn	17/07/2024
Duong Quoc Trung	AI Researcher	Trung	17/07/2024
Do Tuan Dat	Software Developer	Dat	17/07/2024

CLIENT SIGN OFF

Name	Position	Signature	Date
Le Van Khang	AI Engineer	Khang	27/07/2024
Organisation			
Quy Nhon AI Creative Alley			

1. Introduction

The AI model that is being created is intended to identify anomalous activity in data sets. Utilizing cutting-edge machine learning algorithms, the system examines data to spot anomalous patterns and offers insightful information for a range of uses, including fraud detection, security monitoring, and quality assurance.

This report serves as a beacon for Fellow Engineers, Project Managers & Stakeholders and Ethics Guardians. For those who thrive on code and matrices, we'll unravel the technical intricacies. We'll address scalability, resource requirements, and risks. How does this AI fit into the grand scheme? Moreover, biases, fairness, and the societal implications of our creation are also analysed.

1.1 Overview

In this report, we dissect the inner workings of our AI model—the gears, the circuits, and the neural pathways that bring it to life. Here's a succinct preview:

- System Architecture: We'll unveil the blueprint—the layers, connections, data flow and how the components contribute to the workflow. From input preprocessing to inference, we'll explore each component.
- Model Components: Convolutional layers, recurrent cells, attention mechanisms - they all play their part. We'll introduce the cast.
- Training Pipeline: How do we train our model? Gradient descent, backpropagation, epochs—it's a dance of optimization. We'll peek behind the curtain.

1.2 Definitions, Acronyms and Abbreviations

- AI (Artificial Intelligence): Simulation of human intellectual processes using machines, especially computer systems.
- ML (Machine Learning): A subset of AI that involves the use of algorithms and statistical models to enable computers to perform specific tasks without explicit instructions.
- CNN: Convolutional Neural Network.
- RNN: Recurrent Neural Network.
- IoU: Intersection over Union (a metric for object detection).
- FPS: Frames Per Second (video processing speed).
- ROI: Region of Interest (relevant area within an image or video frame).

1.3 Assumptions and Simplifications

- Sufficient Training Data: We assume access to diverse, labelled datasets containing both normal and abnormal behaviours. The quality and quantity of training data significantly impact model performance.
- Stationary Environments: While our model can adapt to gradual changes, sudden environmental shifts (e.g., lighting conditions, camera angles) are assumed to be minimal during deployment.
- Real-Time Inference: Our system aims for real-time performance, assuming hardware capable of handling the computational load.

2. System Architecture Overview

The system architecture for the AI Model for Detecting Abnormal Behavior is designed to leverage modern machine learning techniques and robust software engineering principles to ensure scalability, accuracy, and real-time performance. The architecture includes the following layers:

- ***Data ingestion layer***: This layer handles the collection and preprocessing of data from various sources, including sensors, cameras, and logs. It ensures that the data is cleaned, normalized, and converted into a format suitable for analysis

- ***Feature extraction layer***: This layer processes raw data to extract meaningful features that machine learning models can use. This can include statistical features, time series patterns, and contextual information

- ***Model training layer***: This layer is responsible for training machine learning models using historical data. It includes data separation, model selection, hyperparameter tuning, and model evaluation processes

- ***Inference layer***: This layer deploys trained models to detect anomalous behavior in real-time. It processes incoming data streams, applies models, and generates alerts or notifications about detected anomalies.

- ***Database layer:*** This layer stores historical data, feature sets, model parameters, and detection results. It ensures efficient data retrieval and supports analytical queries

- ***User interface layer:*** This layer provides an interface for system administrators, data scientists, and end users to interact with the system. It includes dashboards, reporting tools, and alert management interfaces

3. Detailed System Design (using Object Orientation or alternative)

The design leverages a machine learning-based approach, specifically focusing on responsibility-driven design principles to ensure each component of the system is modular, scalable, and maintainable.

Data Ingestion Component:

- Subcomponents: Data collectors, data preprocessors, data validators
- Description: Manages the acquisition and preprocessing of data from multiple sources

Feature Extraction Component:

- Subcomponents: Feature engineers, contextual analysers
- Description: Extracts relevant features from the raw data for model training and inference

Model Training Component:

- Subcomponents: Data splitter, model trainer, hyperparameter tuner, model evaluator
- Description: Handles the training and evaluation of machine learning models

Inference Component:

- Subcomponents: Real-time inference engine, alert generator
- Description: Applies trained models to detect anomalies in real-time data streams

Database Component:

- Subcomponents: Data repositories, feature stores, model repositories, result stores
- Description: Stores and manages data, features, models, and detection results

User Interface Component:

- Subcomponents: Dashboards, reporting tools, alert management interface
- Description: Provides user-facing interfaces for interaction with the system

Integration Component:

- Subcomponents: API gateways, notification services
- Description: Manages external integrations and notifications

Security Component:

- Subcomponents: Authentication modules, authorization modules, encryption services, auditing modules
- Description: Ensures system security and data protection.

3.1.The Detailed Design and Justification

The machine learning-based design approach is justified due to:

- **Adaptability**: Allows the system to learn and adapt to new patterns of normal and abnormal behaviour over time.
- **Accuracy**: Enhances detection accuracy through the use of advanced algorithms and continuous model improvement.
- **Scalability**: Supports scaling by processing large volumes of data and handling increasing numbers of data sources.
- **Maintainability**: Simplifies updates and improvements by encapsulating functionalities within distinct components.

3.2.Design Verification

- **Scenario-Based verification**: Demonstrating that the system can support key use cases such as detecting suspicious activities in surveillance footage or identifying anomalies in user behaviour logs.
- **Unit testing**: Writing unit tests for individual components to validate their functionality and adherence to design specifications.
- **Integration testing:** Testing interactions between different components to ensure seamless data flow and system functionality.
- **Model evaluation**: Using performance metrics such as precision, recall, F1-score, and ROC-AUC to evaluate the effectiveness of trained models.
- **User acceptance testing (UAT)**: Engaging end-users to test the system in real-world scenarios and provide feedback on usability and performance.

4. Implementation

This section will examine our system's status of the architectural design, the detailed design, and the Software Requirements Specification (SRS). It's like taking a pulse check on our invention to evaluate how closely it fits our original plan and whether any unanticipated detours have happened.

4.1. System State Assessment

- **Functional Requirements:**

- A collection of videos is entered by the user into the model to serve as training data. As a result, the system notifies the user of the operation's status and obtains the training video dataset.
- Modify training data to improve the effectiveness and efficiency of the learning process. Changes are made to the original training dataset on the system that supports the learning process.
- The model can predict abnormal occurrences or objects by analysing data from the data processing operation. Any variables generated by the models (such as weight and bias) are stored in the system during this learning process so that the algorithms can improved the output later.
- In order to determine which variables are ideal for anomaly detection, the model is constantly modifying the variables that are employed within its algorithms.

- **Interactions:**

- Using HTTP, a popular RESTful API protocol, for Internet of Things systems that include both physical devices like alarm systems and security cameras and the primary software
- Real-time Streaming Protocol (RTSP), which allows security cameras to record and analyse live footage.
- HTTPS provides secure access to the administrator user interface (UI) for more extensive control settings, such as managing accounts, downloading and releasing data storage, logs, and so on.

- **Dependencies:**

- Software Requirements:

  +) Programming language: primarily Python, with support for JSON format

  +) Operating system: compatible with Windows, Linux, and macOS

  +) Frameworks: TensorFlow or PyTorch

- Hardware Requirements:

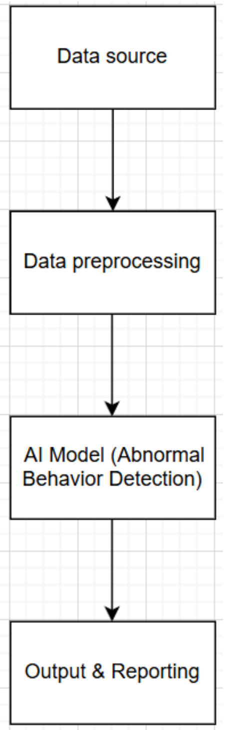
  +) CPU: Core i5 Intel (minimum)

  +) Memory: 8 GB RAM

  +) Storage: At least 50GB (the project dataset may need up to 40GB of storage)

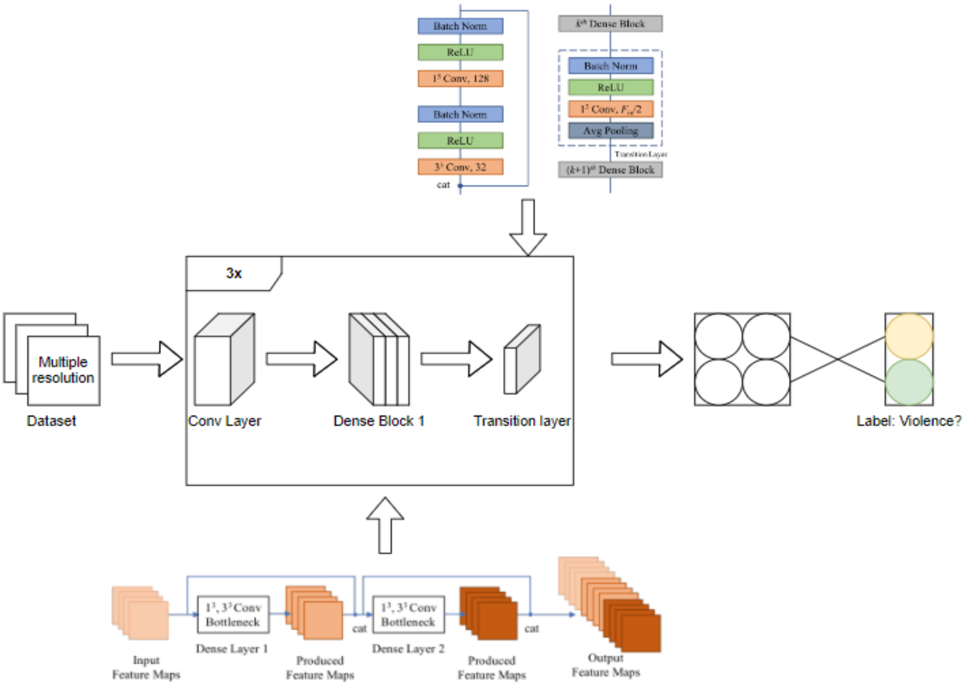
- **Data Flow:**

The project entails creating a novel artificial intelligence model that can identify anomalous activity in datasets. The AI model functions as a prototype and may eventually be integrated into more extensive systems for anomaly detection and behaviour analysis.



4.2. Key Implementation Decisions

- **Choice of Model:**



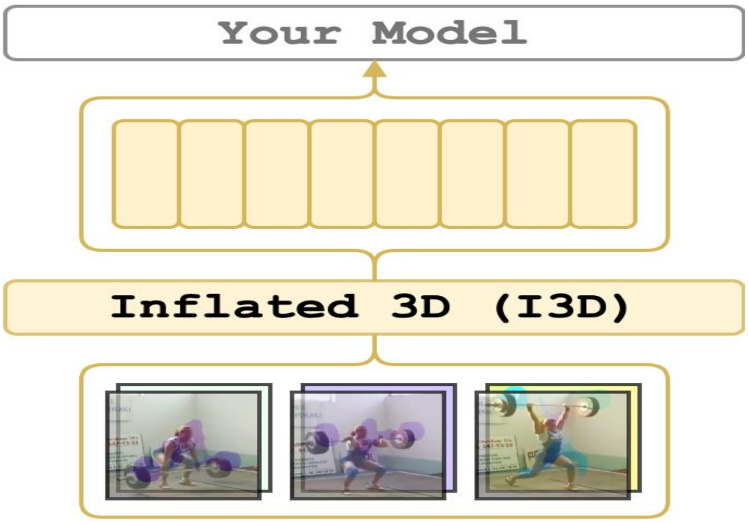
3D-Convolutional Neural Network - (3DCNN) Model

In our pursuit of detecting abnormal behaviours in video data, we decided to go beyond traditional 2D CNNs. Instead, we opted for a 3D convolutional neural network (3D CNN).

- The videos in the data collection can be in any format. Prior to beginning the subsequent processes, the data will be resized at the pre-processing stage.
- The first Convolution Layer uses the frames supplied by the dataset to create intermediate feature maps.
- The output feature maps are created by the 3x3x3 Convolutional layer known as the Dense block.
- By down sampling the features and matching the quantity of input and output feature maps between neighbouring blocks, the transition layers among the three Dense blocks help to streamline the feature extraction process.
- The video's final categorization is determined by using the field of features provided by the Global Average Pooling layer.

So why do we opt for this model? Because it inherently captures both spatial and temporal features from video frames. By considering not only individual frames but also their sequential context, the 3D CNN can learn motion patterns and subtle dynamics that are crucial for abnormality detection. We chose the I3D (Inflated 3D ConvNet) architecture, pre-trained on large-scale video datasets, as our base model.

- **Training Strategy:**



The Inflated 3D Model (I3D)

Our training strategy involved fine-tuning the pre-trained I3D model on our abnormal behaviour dataset. We froze the lower layers to retain the learned features and only updated the top layers specific to our task. Additionally, we experimented with different learning rates and found that gradual unfreezing (i.e., gradually allowing lower layers to update) improved convergence. Our training loss included both binary cross-entropy (for abnormal/normal classification) and temporal consistency loss (to encourage smooth predictions across consecutive frames).

- **Handling Violent Scenes:**

Given the sensitive nature of our task, we carefully curated our training data. We included violent scenes or any content that could be harmful. Additionally, during preprocessing, we applied motion-based filtering to collect frames with sudden, aggressive movements. Our goal was to ensure that the model learned to detect abnormal behaviour, especially being biased toward violence.

- **Performance Metrics:**

To assess our system's performance, we chose two complementary metrics: mean average precision (mAP) and F1-score. Here's why:

- mAP: Since abnormal events are relatively rare in video streams, mAP accounts for precision across different confidence thresholds. It considers both detection accuracy and localization precision. We computed mAP by evaluating our model's predictions against ground truth bounding boxes.
- F1-Score: Abnormal behaviour detection is a trade-off between minimizing false negatives (missing actual abnormal events) and controlling false positives (flagging normal events as abnormal). F1-score balances precision and recall, making it suitable for our task.

- **Deployment Considerations:**

For deployment, we explored two options: edge deployment on an embedded device (such as a Raspberry Pi) and cloud deployment. Edge deployment would allow real-time inference in surveillance scenarios, while cloud deployment would provide scalability and centralized management. We leaned toward edge deployment but kept the cloud option open for future scalability needs.

5. References

- 3D Convolutional Neural Network, 2024, "3D Convolutional Neural Network - A guide for Engineers", viewed 9 July 2024, https://www.neuralconcept.com/post/3d-convolutional-neural-network-a-guide-for-engineers

- TK Kaushik Jegannathan,2022,Building a 3D-CNN in TensorFlow, Analytics Vidhya, viewed 9 July 2024, https://www.analyticsvidhya.com/blog/2022/05/building-a-3d-cnn-in-tensorflow/

- v-iashin, 2022, Video Features Documentation - *The Inflated 3D Model*, viewed 9 July 2024, https://v-iashin.github.io/video_features/