International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

AI-Driven Video Prompt Analysis

Allen Saji*, Ashik David Roy, Nithin V. James, Reenphy George, Prof. Smitha Jacob
*Department of Computer Science, St. Joseph's College of Engineering and Technology, Palai, Kerala, India

ABSTRACT
The vast amount of video data, particularly in surveillance domains, demands an advanced method for efficient content extraction. Traditional approaches face challenges in terms of time, labor, and accessibility. This paper introduces an AI-driven video prompt analysis system, a solution designed for interaction with video data through natural language prompts. The core engine, powered by Python and Yolo V8, enables video analytics, integrating FastAPI, PostgreSQL, and a Python-based video fetcher for video extraction. The user-friendly interface, made with NextJS, TypeScript, and Tauri, ensures an intuitive user experience. The system ensures continuous monitoring, scalability, real-time notifications, and security.
Keywords: Surveillance, AI-driven, Security, Prompt analysis, Video extraction.

I. INTRODUCTION

The ever-expanding increase of video statistics in surveillance domains has propelled the need for superior methodologies in content extraction. Conventional strategies regularly fall short in addressing the demanding situations posed using the sheer extent of records, the labor intensive nature of the analysis, and the shortage of user friendly interfaces. In response to these demanding situations, our project introduces a groundbreaking AI-driven video analysis system. This system seeks to redefine the interplay paradigm with video records by leveraging natural language for content extraction.

The core engine of our system harnesses the power of Python and Yolo V8 for video analytics. To facilitate smooth communication, we integrate fastAPI, PostgreSQL, and a Python-based video fetcher. This combination not only enhances the efficiency of the core engine but also lays the foundation for a responsive and dynamic system. With a user friendly approach, we craft an interface using NextJS, TypeScript, and Tauri, ensuring accessibility for both technical and non-technical users. The subsequent sections will delve into the methodology, and more, providing a detailed idea of the technical factors, key findings, and the broader implications of our work.

II. METHODOLOGY

Our methodology follows a systematic and comprehensive approach to ensure the successful implementation of the AI-driven video prompt analysis system. The key steps in our methodology includes dataset collection, AI model implementation,, real time object detection, performance optimization, user training, ethical

considerations, continuous monitoring and feedback, scalability for large datasets, real time notifications, and user authentication and authorization.

A. Core Engine

The core engine serves as the backbone of our system, leveraging YoloV8 for video analysis and object detection. It analyzes incoming video feeds in real-time or uploaded videos, detecting actions and objects of interest. Once identified, these actions and objects are recorded in the knowledge base for future reference and inference, enabling efficient data retrieval and analysis

B. APIs Integration

Developed using Python with FastAPI, the API integration module handles user requests and data retrieval from the knowledge base. Acting as the intermediary between the frontend, Large Language Models (LLM), and the knowledge base, the backend APIs facilitate seamless communication and data transfer. This component is responsible for managing user interactions, including chat functionalities, and ensuring efficient data exchange between different system modules.

C. Video Fetcher

The Video Fetcher is built using Python and MoviePy library and it plays an important role in processing user prompts and retrieving specific video frames. Upon receiving prompts from users, it communicates with the Large Language Models (LLM) to TRIM relevant video footage based on the specified criteria. By efficiently accessing video data in real time, the Video Fetcher ensures smooth integration with the system, thereby enhancing the overall user experience.

D. Front End Interface

The Frontend Interface serves as the primary interface for users to interact with the system, it offers a platform for accessing and analyzing video data. Developed using NextJS, TypeScript, and Tauri, this enables users to input prompts, upload videos, and access real time CCTV footage with ease. It interacts up close with the system to transmit user prompts and receive processed video frames. Additionally, it interacts with the API Integration to deliver real time updates and notifications to users, ensuring timely and relevant information.

E. Large Language Models

The Language Learning Model (LLM) is an AI system designed to understand and process human language. It's trained on vast amounts of text data to learn patterns and semantics, enabling it to interpret and respond to user queries. In our project, the LLM is fine-tuned to process prompts related to video analysis tasks, such as extracting timestamps or identifying specific actions in video footage.

III. IMPLEMENTATION

A. System Flow

The system flow initiates with users interacting with the Frontend Interface, inputting natural language prompts, or uploading videos. These prompts are processed by the Core Engine, leveraging the YoloV8 model

for video analysis. The API Integration Module facilitates the smooth exchange of data between the Core Engine and external sources. Simultaneously, the Video Fetcher Module responds to user prompts by retrieving and trimming relevant video frames. The coordinated interaction of these modules ensures an integrated and efficient system workflow.

1. **Interaction Between Modules:**
   - 1.1 Frontend Interface Module interacts with the Core Engine Module: Users input prompts and the Core Engine processes them, providing relevant responses through the Front end Interface.
   - 1.2 Integration Module interacts with the Core Engine Module: Facilitates secure communication, enabling data transfer, authentication, and logs between the Core Engine and external entities.
   - 1.3 Video Fetcher Module interacts with the Core Engine and Frontend Interface Modules: Retrieves specific video segments based on user prompts and smoothly delivers them to the Front End Interface.

2. **Process flow:**

The system flow is designed to efficiently process user prompts, analyze real time video footage, and provide relevant outputs. Here's a step-by-step explanation of the flow:

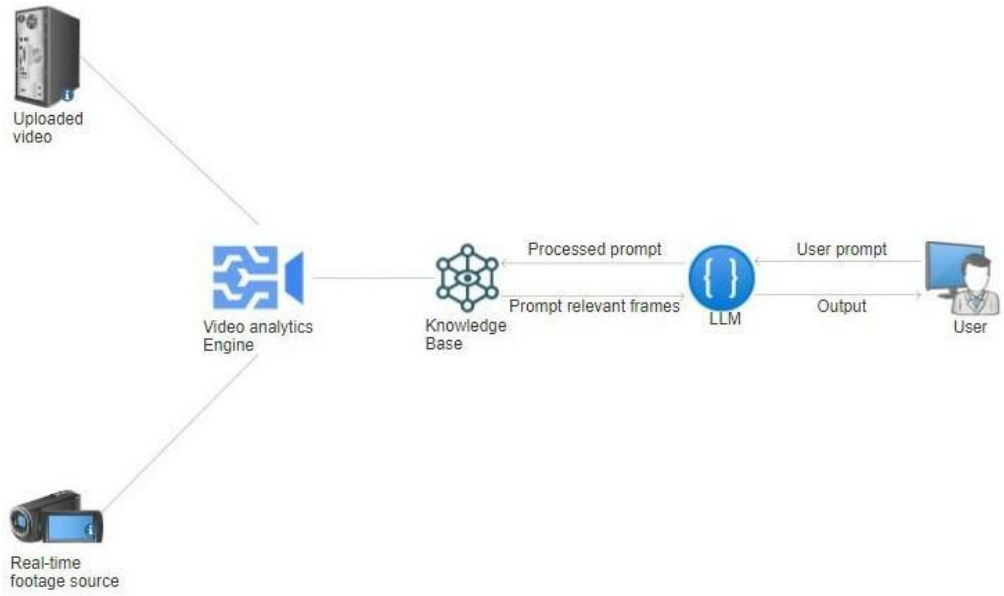

Fig. 1: AI-Driven Video Prompt Analysis System Design

- 2.1. User Input Prompt: The process initiates when a user inputs a prompt, specifying the desired information or action.
- 2.2. Large Language Model (LLM): The user prompt is then processed by the Large Language Model (LLM), which interprets and extracts key information from the natural language input.
- 2.3. Processed Prompt: The processed prompt is generated by the LLM, capturing the details and intent of the user's input.
- 2.4. Knowledge Base: The processed prompt interacts with the Knowledge Base, a storage for information and data relevant to the system.
- 2.5. Video Analytics Engine: Simultaneously, real time footage sources, such as surveillance cameras, are continuously analyzed by the Video Analytics Engine.

2.6. Prompt Relevant Frame Identification: The Video Analytics Engine identifies frames in the real time footage that are relevant to the user's prompt, using video analytics techniques.

2.7. LLM Interaction with Output: The frames identified by the Video Analytics Engine are communicated back to the LLM, improving the understanding of the user prompt.

2.8. User Output: The LLM generates an output based on the processed prompt and relevant frames, presenting meaningful information to the user.

2.9. Loop Completion: The entire process forms a dynamic loop, allowing users to receive outputs based on their prompts and ensuring continuous interaction with real time video analytics.

In summary, these interactions enable a smooth flow of data and commands between different parts of the system, ensuring that users can efficiently input prompts, receive relevant responses, and access the necessary video segments for analysis.

B. Module Implementation

This system's backbone is created by the implementation with a core engine module, which supports video analytics processes. It is a module that has been built using advanced models for object detection and analysis in video frames. Once it receives video data from either CCTV feeds or uploaded files, the core engine initializes object detection models. It identifies objects through careful examination and then sorts them out to be kept alongside timestamps as results. The core engine collaborates with other modules to exchange video data and process results toward perfect communication and improved performance for the entire system.

This way it allows easy transfer of information, user authentication processes, and log maintenance. Secure entry into the core engine is ensured by this module while also confirming who users are by recording how they interact with the system fully. Together with other modules like the Core Engine, the API integration module is vital because it facilitates the exchange of information as well as ensures strong systems at work.

The module that fetches videos is an important part since it bridges the gap between user prompts and the actual video content. This module interprets and processes queries from the front end interface to obtain information like frames and timestamps. The use of concurrency in the module ensures that many instances of video data are operated on at once thereby increasing its speed and ability to respond quickly. After processing videos, this module sends them to a core engine for storage or further analysis which makes the system effective and friendly to users.

IV. RESULTS AND DISCUSSION

Language Understanding Module (LLM) has shown its ability to extract important information from user input prompts. The case of the given prompt "Find the time between 12 AM and 3 PM when a blue SUV with the plate number MH09X4587 entered the parking area." illustrates how LLM can accurately identify and classify elements like color, vehicle type, and license plate number. This module uses natural language processing techniques to efficiently understand users questions and change them into structured JSON, which is further analyzed for appropriate responses.

Fig.2: LLM Output from the User Input Prompt

In Fig. 2 above, this diagram shows a visual representation of the extracted information in JSON output format. In such as an organization, features like color, cars's makes, as well as Registration numbers are arranged systematically, allowing easy access and understanding by other system parts. For example, It is used in structuring data for efficient handling by users. queries, resulting in smooth communication between users and machines. The JSON output demonstrates how unstructured Inputs are converted into actionable data by the system, thus enabling effective video content extraction and analysis.

Fig.3: API Routes

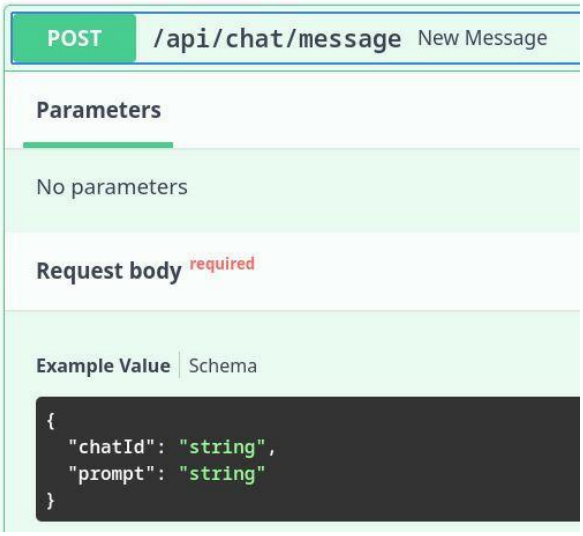

Fig.4: Request body of "/api/chat/message" that creates new message

Also, the API module has successfully completed important tasks. vital to the system, such as generating chat IDs and uploading videos. Fig. 3 shows that a new chat instance is started. by the POST request to the ”/api/chat/” endpoint through our system’s API implementation. Here, people can talk within the system. Afterward, users are required to upload video footage. pertaining to the chat. The process of uploading involves making a POST request to the ”/api/footage/upload” endpoint represented in Fig. 4. It receives files and then employs our model to find what objects there are in the footage uploaded. here. Thereafter, these objects and their timestamps get stored in a knowledgebase for further analysis.

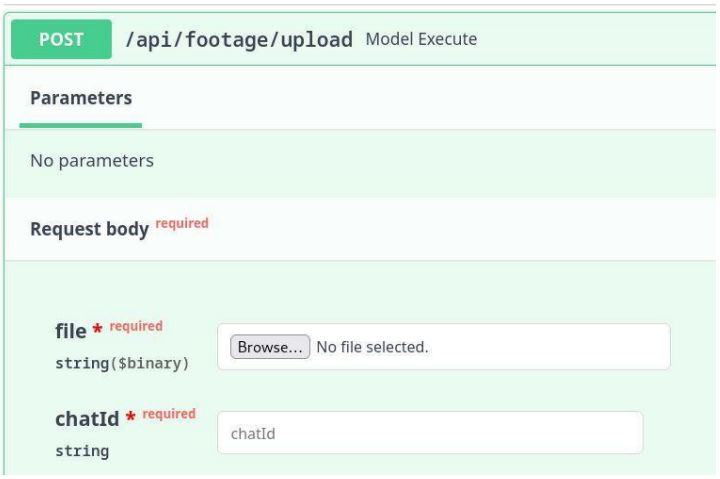

Fig.5: Request body of "/api/footage/upload" that uploads video and runs the model

A user may create new messages within the chat once. footage has been uploaded and processed through a POST request towards the ”/api/chat/message” endpoint. Fig. 5 explains how the request body looks when creating such a message. Moreover, any user could be able to fetch all chats just by doing GET requests on the”/api/chat” endpoints. Also, data about any given chat can be accessed through a GET request on ”/api/chat/?id” endpoint with chat ID as a parameter value entered into it as well. In addition, users can delete particular chats by sending DELETE requests via URLs ending with “id,” where “id” represents this specific chat’s ID again.

V. CONCLUSION

In conclusion, the AI-powered video analysis system signifies a major advancement in surveillance technology. Integrating Python, YoloV8, FastAPI, PostgreSQL, NextJS, TypeScript, and Tauri, the system provides a platform for video content extraction. The core engine, fueled by YoloV8 and Python, facilitates real time object detection, while the API module ensures secure communication. The Python based video fetcher enhances user interaction, and the user friendly interface, developed with NextJS, TypeScript, and Tauri, enables excellent use. The system promises a continuous monitoring, scalability, real time notifications, user training, authentication, and ethical guidelines. This positions the system as a responsible and user friendly solution, contributing to security, efficiency, and the application of AI in surveillance, among other areas.

VI. ACKNOWLEDGMENT

The progression of our work owes much to the invaluable support and guidance from various individuals. Our sincere thanks to the management of St. Joseph's College of Engineering and Technology for this opportunity. Special gratitude to Dr. V. P. Devassia, our Principal, for his unwavering support. We appreciate the valuable contributions of Dr. Joby P.P. and Prof. Kishore Sebastian. A special acknowledgment to our project guide, Prof. Smitha Jacob, and the entire staff of the Department of Computer Science and Engineering for their constant encouragement and support throughout this phase.

VII. REFERENCES

[1] You Only Look Once: Unified, Real-Time Object Detection by Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

[2] A Review of Yolo Algorithm Developments by Peiyuan Jiang, Daji Ergu*, Fangyao Liu, Ying Cai, Bo Ma, The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 \& 2021)

[3] Research on Traffic Sign Detection Based on Improved YOLOv8 by Zhongjie Huang1, Lintao Li1, Gerd Christian Krizek2, Linhao Sun, Journal of Computer and Communications, 2023, 11, 226-232

[4] YOLO-PAI: Real-time handheld call behavior detection algorithm and embedded application by Zuopeng Zhao, Tianci Zheng, Kai Hao, Junjie Xu, Shuya Cui, Xiaofeng Liu, Guangming Zhao, Jie Zhou, Chen He

[5] YOLO-based Human Action Recognition and Localization by Shubham Shindea, Ashwin Kotharia, Vikram Gupta

[6] YOLO*C — Adding context improves YOLO performance by Goran Oreski

[7] Object detection in crowded scenes via joint prediction by Hong-hui Xu, Xin-qing Wang, Dong Wang, Bao-guo Duan, Ting Rui

[8] Faster-YOLO: An accurate and faster object detection method Yunhua by Yin, Huifang Li, Wei Fu

[9] A real-time object detection algorithm for video by Shengyu Lua, Beizhan Wanga, Hongji Wanga, Lihao Chenb, Ma Linjiana, Xiaoyan Zhangc

[10] What are large language models supposed to model? by Idan A. Blank Cognitive Sciences; Vol 27

[11] AccDecoder: Accelerated Decoding for Neural-enhanced Video Analytics by Tingting Yuan, Liang Mi, Weijun Wang†‡, Haipeng Dai, Xiaoming Fu; University of Gottingen, Germany; Nanjing University, China

[12] Video analytics using deep learning for crowd analysis by Md Roman Bhuiyan1, Junaidi Abdullah1, Noramiza Hashim1 & Fahmid Al Farid1; Multimedia Tools and Applications (2022) 81:27895–27922, https://doi.org/10.1007/s11042-022-12833-z

[13] Language as Queries for Referring Video Object Segmentation by Jiannan Wu, Yi Jiang2 Peize Sun, Zehuan Yuan, Ping Luo1, The University of Hong Kong, ByteDance, HKU-TCL Joint Research Centre for Artificial Intelligence

[14] Forensic Digital Analysis For CCTV Video Recording by Pria Sukamto, Ispandi, Arman Syah Putra, Nurul Aisyah, Rohmat Toufiq, International Journal Of Science, Technology & Management, ISSN: 2722-4015

[15] Multi-frame-based adversarial learning approach for video surveillance by Prashant W. Patil, Akshay Dudhane, Sachin Chaudhary, Subrahmanyam Murala; Pattern Recognition 122 (2022) 108350