A Scalable Agentic AI Framework for Video Classification and Summarization via LoRA Fine-Tuning and LangGraph

Gangadhar S Shiva
Team 3rd
gangadhar@example.edu

Nagarajan Mahalingam
Team 3rd
nagarajan@example.edu

Akshobhya Rao BV
Team 3rd
akshobhya@example.edu

***Abstract*—This technical report details the design and implementation of a scalable Agentic AI framework intended for comprehensive video analysis, encompassing classification, summarization, and natural language processing (NLP) integration. The primary goal is to create a robust pipeline capable of handling large video datasets efficiently by leveraging parameter-efficient fine-tuning (PEFT) using LoRA for model adaptation and LangGraph for constructing autonomous, multi-step decision-making agents. The core focus of the current implementation is the enhanced video summarization module, which employs YOLOv8 for object detection, coupled with the Structural Similarity Index (SSIM) for concurrent detection of movement and abrupt scene changes. Initial results confirm the successful integration of YOLOv8 and highlight the critical sensitivity of SSIM thresholds on the final output, providing a clear path for future parameter optimization and full agent integration.**

***Index Terms*—Agentic AI, Video Classification, Video Summarization, YOLOv8, LoRA, PEFT, LangGraph, SSIM, Keyframe Extraction**

I. Introduction

The exponential growth of video data mandates the development of sophisticated and scalable analytical tools. Traditional methods relying on exhaustive frame processing or full model re-training are often computationally prohibitive. This project addresses these challenges by proposing an Agentic AI architecture designed for efficient video classification and summarization. The framework is inherently modular, enabling rapid deployment and continuous improvement across various video-centric tasks, from action recognition to content indexing.

The overall objective is to migrate from monolithic AI solutions to a distributed, agent-based system where specialized models and logic modules cooperate to achieve complex analytical goals. The initial phase documented herein focuses on creating a robust, multi-criteria keyframe extraction mechanism.

II. Literature Review

A. Object Detection and Video Analysis

You Only Look Once (YOLO) has become the state-of-the-art for real-time object detection due to its superior speed and accuracy. The transition from YOLOv5, which was initially attempted in the environment, to the optimized YOLOv8 model provides enhanced performance and a streamlined API for integration into the video processing pipeline. The task of video summarization, unlike static image detection, requires not only object localization but also an understanding of temporal significance. Previous works often rely solely on visual saliency or shot boundary detection.

B. Temporal Significance through SSIM

To address the temporal aspect, this project employs the Structural Similarity Index (SSIM), an established metric for perceptual image quality. SSIM is utilized in two distinct contexts within the pipeline:

- **Movement Detection:** Comparing contiguous frames (F_t and F_{t-1}) to capture subtle motion or continuous action. A low SSIM score in this context implies significant, potentially interesting, movement.
- **Scene Change Detection:** Comparing the current frame (F_t) with the last selected keyframe ($K_{t'}$), where $t' < t$, to identify abrupt visual discontinuities or cuts, which signify a new scene.

The combination of SSIM and YOLO ensures that keyframes are selected based on both semantic content (objects) and visual dynamics (movement/scene cuts).

C. Scalable Fine-Tuning with LoRA

To ensure scalability and efficient adaptation of large base models for specific video classification tasks, the framework is designed to utilize Low-Rank Adaptation (LoRA), a Parameter-Efficient Fine-Tuning (PEFT) technique. LoRA injects trainable rank decomposition matrices into the layers of pre-trained models, dramatically reducing the number of trainable parameters for downstream tasks. This approach minimizes storage and computational requirements during the fine-tuning stage, a crucial feature for a model intended for deployment in diverse environments.

D. Agentic Workflow with LangGraph

The overarching architecture leverages LangGraph, a library built on top of LangChain, to define the structure of stateful, multi-actor applications. LangGraph enables the creation of cyclical graphs where different AI agents (e.g., a Summarization Agent, a Classification Agent, an NLP Agent) can interact,

make decisions, and pass information in a defined sequence. This architecture ensures the system can dynamically adapt its processing path based on the input video characteristics or user query.

III. Methodology

A. Multi-Criteria Keyframe Extraction

The keyframe extraction process follows a tiered filtration system:

1) **Object-Centric Pre-filtering:** A frame must first satisfy the semantic criterion by containing a minimum count (≥ 1) of desired objects (e.g., 'person', 'car', 'bicycle') detected by YOLOv8.
2) **Dynamic Filtering:** The pre-filtered frame must then satisfy at least one of the dynamic criteria: significant movement (low consecutive SSIM) or scene change (low keyframe-to-current SSIM).

This methodology ensures that the extracted keyframes are not merely visually distinct but also semantically relevant to the task (e.g., an action involving a person or a vehicle).

B. Agentic AI Orchestration

The entire video processing task is orchestrated as a state machine using LangGraph. The graph nodes include:

- **Data Ingestion Node:** Handles video decoding and frame dispatch.
- **Summarization Node:** Executes the multi-criteria keyframe extraction.
- **Classification Node (Future):** Fine-tunes a LoRA-enabled vision model and classifies the video based on the keyframes.
- **NLP/Transcription Node (Future):** Transcribes audio and generates a summary using a Large Language Model (LLM).

The core advantage of this methodology is the decoupling of specialized tasks into independent, reusable agents.

1) LangGraph Agent Development: The next phase of agentic orchestration includes:

- **State Machine Definition:** Define the full state dictionary and nodes for NLP, Classification, and Orchestration.
- **Agent Implementation:** Build the three primary agents: Summarization Agent (current functionality), NLP Agent (audio transcription and text summary), and Classifier Agent (LoRA-based model inference).
- **Workflow Cycles:** Implement decision-making cycles for intelligent video analysis, such as the Classifier Agent deciding if a video requires a full NLP summary based on its initial content analysis.

2) LoRA Fine-Tuning Integration: For efficient model adaptation:

- **Base Model Selection:** Select a suitable Vision Transformer (ViT) or backbone from a common video model as the base for classification.
- **PEFT Integration:** Configure LoRA adapters for the selected base model using the PEFT library.
- **Training Pipeline:** Develop the training loop to fine-tune the LoRA adapters on the video dataset for target classification tasks (e.g., action recognition).

The successful implementation of these steps will complete the vision for a scalable, efficient, and intelligent Agentic AI video analysis platform.

IV. System Design

A. Data Layer

This layer manages the ingestion of raw video files (.avi) and the persistent storage of processed outputs (keyframes, summary videos, LoRA weights). It handles video decoding using OpenCV and file management using standard Python libraries.

B. Processing Layer

This layer houses the core computational modules:

- **Vision Module:** Contains the YOLOv8 model for real-time object detection and the SSIM calculation utility.
- **Fine-Tuning Module (Planned):** Implements the LoRA logic using the PEFT library for adapting vision transformers to target classification tasks.
- **Media Synthesis Module:** Assembles extracted keyframes into the final summary video.

C. Agentic Layer

The highest layer is the LangGraph State Machine. The state manages the flow of information between agents, including the current video path, extracted keyframe list, and system control flags (e.g., model status, error reports). The graph logic determines the next step based on the outcome of the current node's execution.

V. Workflow and Algorithm

The operational workflow for the video summarization module is sequential and iterative:

1) **Initialization:** The YOLOv8n model is loaded once. The video path is established, and processing parameters are set.
2) **Frame Iteration:** The system enters a loop, reading frames (F_t) sequentially from the video stream.
3) **Detection Check:** YOLOv8 inference is performed on F_t to count relevant object detections.
4) **Movement Check:** SSIM is calculated between F_t and F_{t-1}.
5) **Scene Check:** SSIM is calculated between F_t and the last recorded keyframe ($K_{t'}$).
6) **Keyframe Decision:** If the frame meets the object count threshold AND (movement threshold OR scene change threshold), F_t is added to the keyframe list.
7) **Output Generation:** The list of keyframes is converted into a summary video.

Function summarize_video_yolo($video\_path$, $model$, $classes$, c_{thresh}, m_{thresh}, s_{thresh})
Initialize $keyframes \leftarrow []$, $frame\_count \leftarrow 0$
Initialize $prev\_gray \leftarrow NULL$, $prev\_key\_gray \leftarrow NULL$
Open Video Capture (cap)
while cap is open and reading frames **do**
  $F_t \leftarrow$ Read current frame
  $F_t^{gray} \leftarrow$ Convert F_t to grayscale
  $frame\_count \leftarrow frame\_count + 1$
  // Semantic Check (YOLO)
  $results \leftarrow model(F_t)$
  $detections \leftarrow$ Count objects in $classes$
  $consider \leftarrow (detections \geq c_{thresh})$
  // Movement Check (SSIM)
  $movement \leftarrow FALSE$
  if $prev\_gray \neq NULL$ **then**
    $ssim_m \leftarrow SSIM(prev\_gray, F_t^{gray})$
    if $ssim_m < m_{thresh}$ **then**
      $movement \leftarrow TRUE$
    end if
  end if
  $prev\_gray \leftarrow F_t^{gray}$
  // Scene Change Check
  $scene\_change \leftarrow FALSE$
  if $keyframes$ is EMPTY **then**
    $scene\_change \leftarrow TRUE$
  else
    $ssim_s \leftarrow SSIM(prev\_key\_gray, F_t^{gray})$
    if $ssim_s < s_{thresh}$ **then**
      $scene\_change \leftarrow TRUE$
    end if
  end if
  // Final Selection
  if $consider$ AND ($movement$ OR $scene\_change$) **then**
    Add F_t to $keyframes$
    $prev\_key\_gray \leftarrow F_t^{gray}$
  end if
end while
return $keyframes$

Fig. 1. Keyframe Extraction Algorithm

VI. Implementation

A. Dependency Management

The environment required the installation of specialized libraries, including `ultralytics` for YOLOv8, `peft` and `torch` for LoRA (future integration), `langgraph` for the agent framework (future integration), and `moviepy` and `opencv` for media processing.

B. YOLOv8 Integration and Refinement

The initial implementation used `torch.hub.load` for YOLOv5, which failed due to environment constraints, necessitating a dummy model fallback. The final successful implementation involved:

- Explicitly installing `ultralytics`
- Loading the model: `yolo_model = YOLO('yolov8n.pt')`
- Adapting the keyframe function to accept the pre-loaded model and correctly parsing the inference results

This refinement was critical to ensure the object detection criteria were accurately applied.

C. SSIM Calculation

The SSIM function was used for all perceptual comparisons. To maintain computational efficiency, SSIM was only calculated on grayscale versions of frames. The thresholds were initialized as:

- **Movement Threshold:** 0.95 (sensitive to minor changes)
- **Scene Change Threshold:** 0.85 (sensitive to moderate changes)

A lower SSIM score indicates a greater difference between compared images.

VII. Results and Analysis

A. Quantitative Results

The revised implementation with the fully functional YOLOv8n model was tested on a sample video from the dataset. The test scenario involved analyzing a video of 300 frames, focusing on the desired classes ['person', 'car', 'bicycle'] with a minimum detection count threshold of 1.

- **Total Frames Analyzed:** 300
- **Extracted Keyframes:** 254

The resulting summary video was successfully generated and visually confirmed the keyframe selection.

B. Threshold Sensitivity Analysis

The most significant finding is the high sensitivity of the SSIM thresholds, particularly the movement threshold of 0.95. With 254 keyframes extracted from 300 total frames (84.7%), the resulting summary video is nearly as long as the original. This implies that the movement threshold is too high, registering even subtle, perceptually irrelevant changes as significant motion.

The SSIM metric ranges from 0 (no similarity) to 1 (perfect similarity). A threshold of 0.95 for movement meant that any frame pair with a dissimilarity greater than 5% triggered a keyframe selection, leading to an overly verbose summary. This demonstrates the critical importance of parameter tuning in dynamic video analysis.

Recommendation for Movement: To achieve a more concise summary, the movement threshold must be aggressively lowered to a value between 0.60 and 0.80, capturing only substantial motion.

Recommendation for Scene Change: The scene change threshold of 0.85 appears more balanced for identifying scene cuts, though this, too, may benefit from tuning based on the desired level of granularity.

VIII. Recommendations

A. Parameter Optimization Strategy

A dedicated hyperparameter search must be performed to optimize the SSIM thresholds. This should involve testing a range of values and correlating the resulting summary length and quality with human-annotated ground truth. A possible optimization function $\mathcal{L}$ could minimize the difference between the summary length L_s and a target summary length L_t, subject to maintaining a minimum precision score P_{min} on capturing critical events:

$$\min_{\theta_{move}, \theta_{scene}} (L_s - L_t)^2 \quad \text{s.t.} \quad P(Events) \geq P_{min} \tag{1}$$

where θ represents the threshold parameters.

B. Scalability Improvement: Frame Sampling

The current frame-by-frame processing is computationally expensive. For videos with high frame rates (e.g., 60 FPS), the processing time is linear with video duration. We recommend implementing a frame sampling strategy:

- Process only one frame every N frames (e.g., $N = 5$) by default.
- Use a fast pre-processing agent (e.g., a simple color histogram change detection) to flag high-probability event areas, temporarily reducing N to 1 for that segment.

This hybrid approach balances computational load with event detection accuracy.

IX. Conclusion

The initial phase of the Scalable Agentic AI framework has successfully established a multi-criteria video summarization module. The implementation verified the correct integration and functionality of the YOLOv8 object detection model, which serves as the semantic backbone for keyframe selection. The use of SSIM for simultaneous movement and scene change detection was confirmed, although the results demonstrated a high sensitivity in the initial threshold parameters, leading to an over-extraction of keyframes. This finding provides clear direction for targeted optimization. The established architecture is modular and robust, laying a solid technical foundation for the subsequent integration of LoRA-based classification and the LangGraph-driven agentic system.

References

[1] J. Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE CVPR*, 2016.

[2] Z. Wang et al., "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600-612, 2004.

[3] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," in *Proc. ICLR*, 2022.