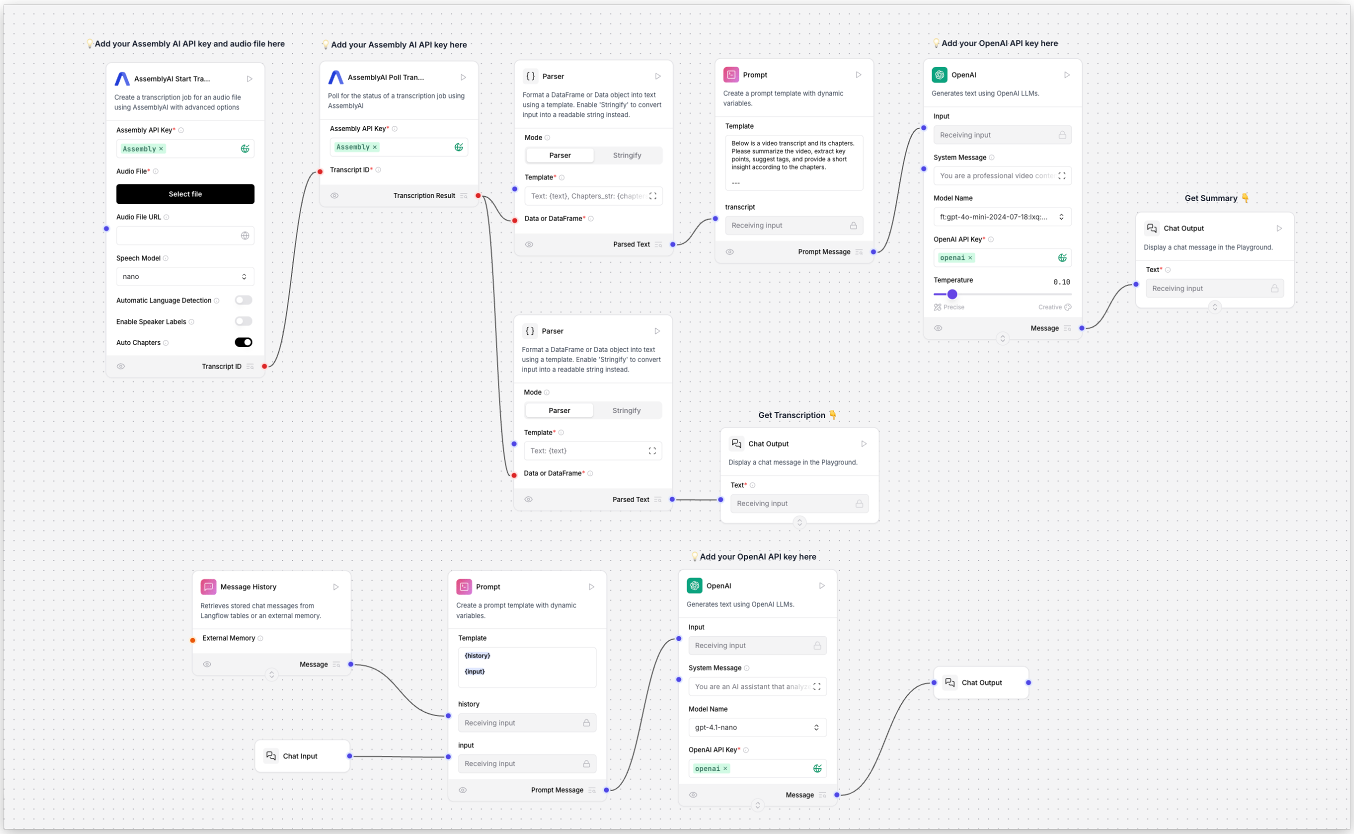
VidSummarize Technical Report

1. Background & Motivation

With the surge in online video content, users increasingly need efficient ways to comprehend and manage long videos. Mainstream video summarization tools like NoteGPT offer strong features but are **expensive and subscription-based**, making them unfriendly for individuals or occasional users with uncertain needs. My motivation for developing VidSummarize stemmed from a desire for **on-demand, low-cost, and flexible usage**. VidSummarize is designed to be open-source and self-deployable, aiming to match leading products in output quality while offering greater flexibility and portability, requiring only minimal API usage.

2. System Architecture



💡Add your Assembly AI API key and audio file here

AssemblyAI Start Tra...
Create a transcription job for an audio file using AssemblyAI with advanced options
Assembly API Key*: Assembly ×
Audio File*: Select file
Audio File URL
Speech Model: nano
Automatic Language Detection: off
Enable Speaker Labels: off
Auto Chapters: on
Transcript ID

💡Add your Assembly AI API key here

AssemblyAI Poll Tran...
Poll for the status of a transcription job using AssemblyAI
Assembly API Key*: Assembly ×
Transcript ID*
Transcription Result

Parser
Format a DataFrame or Data object into text using a template. Enable 'Stringify' to convert input into a readable string instead.
Mode: Parser | Stringify
Template*: Text: {text}, Chapters_str: {chapters
Data or DataFrame*
Parsed Text

Prompt
Create a prompt template with dynamic variables.
Template:
Below is a video transcript and its chapters. Please summarize the video, extract key points, suggest tags, and provide a short insight according to the chapters.

---
transcript: Receiving input
Prompt Message

💡Add your OpenAI API key here

OpenAI
Generates text using OpenAI LLMs.
Input: Receiving input
System Message: You are a professional video conten
Model Name: ft:gpt-4o-mini-2024-07-18:lxq:...
OpenAI API Key*: openai ×
Temperature: 0.10
Precise | Creative
Message

Get Summary 👇

Chat Output
Display a chat message in the Playground.
Text*: Receiving input

Parser
Format a DataFrame or Data object into text using a template. Enable 'Stringify' to convert input into a readable string instead.
Mode: Parser | Stringify
Template*: Text: {text}
Data or DataFrame*
Parsed Text

Get Transcription 👇

Chat Output
Display a chat message in the Playground.
Text*: Receiving input

Message History
Retrieves stored chat messages from Langflow tables or an external memory.
External Memory
Message

Chat Input

Prompt
Create a prompt template with dynamic variables.
Template:
{history}
{input}
history: Receiving input
input: Receiving input
Prompt Message

💡Add your OpenAI API key here

OpenAI
Generates text using OpenAI LLMs.
Input: Receiving input
System Message: You are an AI assistant that analyze
Model Name: gpt-4.1-nano
OpenAI API Key*: openai ×
Message

Chat Output

Main process nodes

1. **AssemblyAI Start Transcription**
   - Users upload audio/video files (or provide audio URLs), enter their AssemblyAI API Key, and select parameters (e.g., model, auto-chaptering).
   - A Transcript ID is generated.
2. **AssemblyAI Poll Transcription**
   - The system polls the transcription status and retrieves the complete transcript.
3. **Parser**
   - Parses the transcript (supports chapter segmentation and formatted output).

4. **Prompt (Summarization)**
    - Combines transcript text and chapters to generate a structured prompt, fed into a fine-tuned OpenAI GPT-4o mini model.
5. **OpenAI (Summarization)**
    - Calls the fine-tuned model to generate a structured video summary (chapter summaries, key points, tags, etc.).
6. **ChatOutput (Summarization)**
    - Displays the summary results, viewable directly in the Playground chat window.
7. **Chat Workflow**
    - Manages message history, prompts, OpenAI calls (supports fine-tuned models), and ChatOutput, enabling multi-turn Q&A based on transcript/summary.

Design Highlights

- **Extensibility**: All nodes are modular and replaceable, supporting custom fine-tuned models and parameters.
- **High Portability**: Deployable locally or on the cloud (e.g., Hugging Face Spaces), with API Keys managed securely via environment variables.
- **User-Friendly Interaction**: Users can complete upload, transcription, summarization, and Q&A—all in a single interface.

3. Fine-Tuning & Model Optimization

Dataset Preparation

- **Source**: 20 Ali Abdaal YouTube videos covering diverse topics.
- **Process**: AssemblyAI transcription → automatic chapter extraction → manual annotation of structured summaries (overview, chapter summaries, key points, tags, insights).
- **Data Structure**: Includes full transcript, chapter segmentation, and target structured summary.

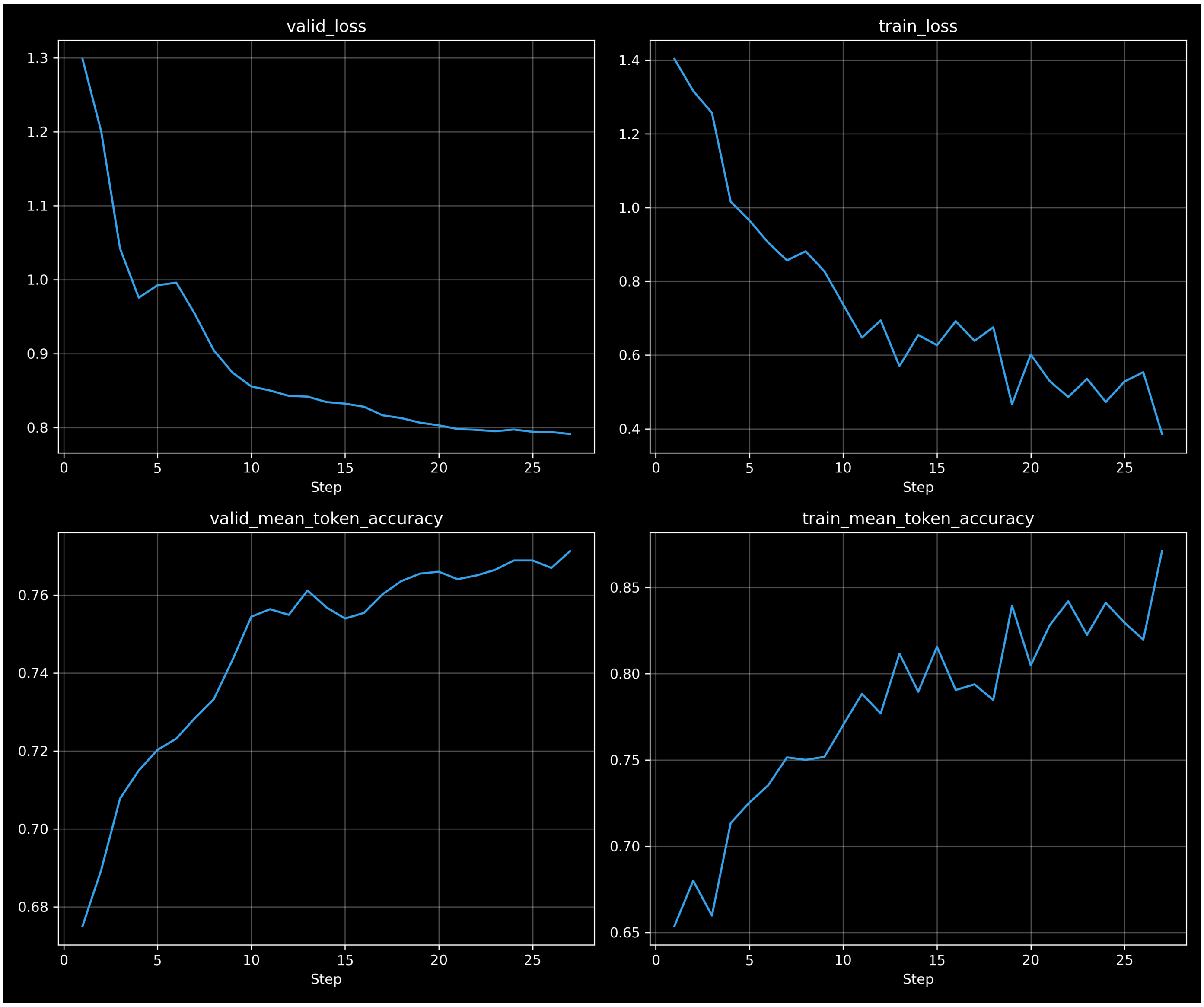
Fine-Tuning Process

- **Base Model**: OpenAI gpt-4o-mini (64k context, low cost, high performance).
- **Platform**: OpenAI official fine-tuning platform.
- **Method**: Supervised fine-tuning, focusing on structured output capabilities.

First Run

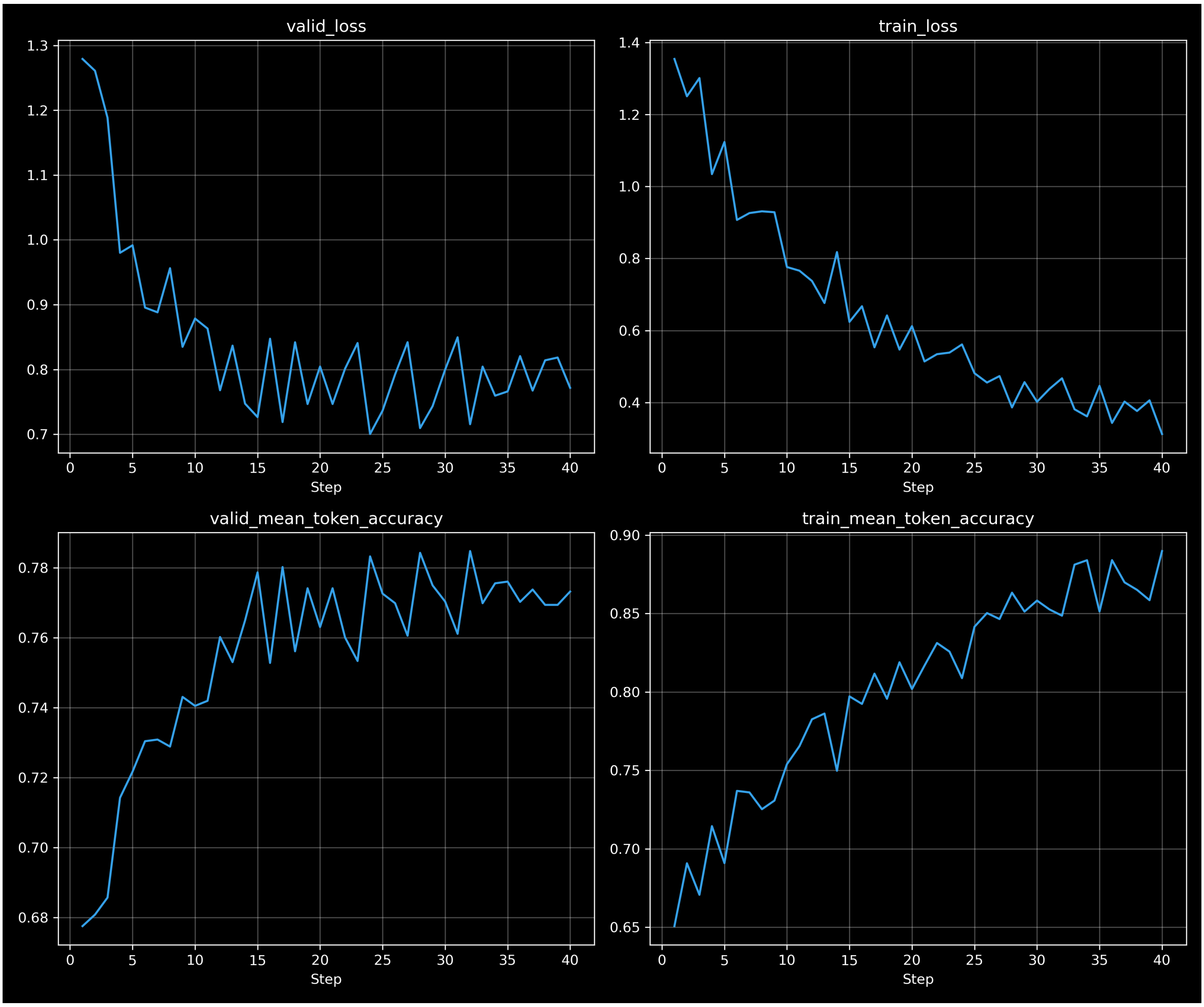
- **Settings**:
    - 18:2 split
    - batch size: 2
    - epochs: 3
    - LR multiplier: 1.8

- **Observations**:
  - Training and validation loss dropped quickly.
  - Token accuracy improved, but validation loss plateaued early.
  - Some instability in validation accuracy.



Second Run

- **Settings**:
  - 16:4 split
  - batch size: 2
  - epochs: 5
  - LR multiplier: 1.0
- **Observations**:
  - Smoother, more stable decrease in both training and validation loss.
  - Validation mean token accuracy improved and stabilized.
  - Reduced gap between training and validation, indicating less overfitting and better generalization.



Analysis:

The first run, with a higher learning rate and fewer epochs, led to rapid loss reduction but less stable validation accuracy. The second run, with a lower learning rate and more epochs, produced smoother curves, higher and more stable validation accuracy, and better generalization. This highlights the importance of careful hyperparameter tuning, especially with small datasets.

4. Challenges & Solutions

4.1 Automation & Cloud Environment Limitations

Automated Integration

A major goal was to automate extraction and transcription of YouTube video content using tools like `yt-dlp` and `youtube-transcript-api`. However, on public cloud platforms (e.g., Hugging Face Spaces, DataStax), several issues arose:

- **YouTube IP Blocking**:
  YouTube actively blocks or requires verification for automated requests from known cloud IPs. Both `yt-dlp` and `youtube-transcript-api` failed with errors like “Sign in to confirm you’re not a bot” or `IndexError: list index out of range`. This is a well-documented issue (see [GitHub Issue #5486]).
- **Dependency Management**:
  While external libraries (e.g., `yt-dlp`) could be installed via Dockerfile customization, runtime errors persisted due to YouTube’s anti-bot measures.
- **Authentication Workarounds**:
  Exporting browser cookies to a `cookies.txt` file and mounting it in Docker could bypass some restrictions but poses significant security risks, especially for public or open-source deployments. Only recommended for private use.
- **YouTube Data API Limitations**:
  The official YouTube Data API can fetch metadata but **cannot** provide direct access to media files or transcripts, so it cannot replace `yt-dlp`.

Final Solution

Given these barriers, the most practical approach for public cloud deployments is to **require users to manually upload audio/video files** for transcription and summarization. Automated YouTube extraction is feasible only on local or private servers where IP blocking and authentication can be managed securely.

4.2 Deep Component Customization

AssemblyAI Component Customization

The native Langflow AssemblyAI component **did not support the `auto_chapter` parameter**, meaning that even if AssemblyAI backend supported auto-chaptering, it could not be configured or invoked in the visual interface. I modified the component source code to **add an `auto_chapter` option**, enabling users to activate automatic chapter segmentation with a single click during upload. This enhances automation and lays a foundation for structured summaries and chapter-based outputs.

- **Technical Details**: Added `auto_chapter` to parameter definitions and ensured correct API call propagation.
- **Effect**: Users can check “Auto Chapters” during upload to receive chapter-structured transcripts, greatly improving summary accuracy and readability.

OpenAI Component Customization

Langflow’s default OpenAI component **did not support specifying custom fine-tuned model names**, only allowing selection from official base models. This was inconvenient for scenarios requiring custom models. I deeply modified the frontend and backend code, **changing the model name parameter to a freely editable and persistent dropdown**, supporting default values.

- **Technical Details**: Changed the model name parameter from a static dropdown (base models only) to `DropdownInput`, allowing custom input and persistence.
- **Effect**: Users can directly enter their fine-tuned model name, eliminating the need to switch manually each time, greatly improving efficiency and experience.

API Key Security Management

All API keys (AssemblyAI and OpenAI) are **managed via environment variables**, preventing exposure in the frontend or logs. Users configure a `.env` file or environment variables at deployment; the frontend only shows placeholder prompts, never the actual keys.

- **Technical Details**: Components read from environment variables; frontend displays “API Key configured” without revealing the value.
- **Effect**: Ensures key security, suitable for local, private cloud, and public cloud deployments.

5. Performance Evaluation & Product Benchmarking

- **Output Quality**: The fine-tuned GPT-4o-mini model generates **structured, chaptered, and focused high-quality summaries**. In chapter segmentation, key point extraction, and tag recommendation, the results **match those of mainstream products like NoteGPT**. In tests, the fine-tuned model produced more logical, focused summaries, accurately extracting key video information and outperforming the base model.
- **Cost Advantage**: VidSummarize **requires no subscription**, consuming only minimal API quota per use, making **per-use cost extremely low**—ideal for individuals and small teams, avoiding high monthly or annual fees.
- **Flexibility & Portability**
  - **Multi-Environment Deployment**: Supports local, private cloud, Hugging Face Spaces, etc., with flexible and replaceable components.
  - **Strong Customization**: Supports custom fine-tuned models, parameters, and workflow nodes, facilitating secondary development and personalized adjustments.
  - **Easy Maintenance & Upgrades**: All custom code and dependencies are well-documented for future maintenance and feature expansion.
- **Interactive Experience**: The system supports **multi-turn dialogue**, allowing users to deeply explore and analyze content based on transcripts and summaries, meeting needs from “quick overview” to “in-depth analysis” and enhancing user experience.

6. Limitations & Future Directions

Limitations

- **Limited Data Volume**: Currently, only 20 video samples were used for fine-tuning, limiting model generalization. Expanding the dataset across domains and styles is needed for robustness.
- **Domain Adaptability**: The model is mainly tuned for Ali Abdaal’s style; performance on other domains or video structures requires further validation and optimization.
- **Automation Restrictions**: On cloud platforms like Hugging Face Spaces, IP blocking by YouTube limits automated audio/video extraction and transcription; manual upload is required.
- **Weak Batch & Concurrency Capabilities**: Currently single-task flow; lacks batch processing or high concurrency support, making it less suitable for large-scale or team collaboration.

- **Limited Evaluation Dimensions**: Evaluation mainly relies on automated metrics (e.g., validation loss, token accuracy), lacking human evaluation and finer-grained metrics (e.g., ROUGE, BLEU), making comprehensive assessment difficult.

Future Directions

- **Batch & Concurrency Support**: Plan to introduce batch upload and multi-task concurrent processing to improve efficiency and support more simultaneous users.
- **Private Deployment & Automation**: Enable deployment on private servers with dedicated IPs for full-process automation from link to audio/video, overcoming cloud IP blocking.
- **Multimodal Expansion**: Integrate image and audio summarization for richer content processing.
- **RAG (Retrieval-Augmented Generation)**: Combine knowledge base and RAG techniques to enhance Q&A accuracy and depth.
- **Larger-Scale Fine-Tuning & Evaluation**: Expand the dataset, try larger models and advanced fine-tuning methods (e.g., DPO), and introduce human evaluation and multidimensional automated metrics for improved performance and reliability.

7. Conclusion

VidSummarize achieves low-cost, flexible, and portable video content summarization and interactive Q&A, matching mainstream products in output quality. It is suitable for individuals and small teams on-demand. Through deep customization and fine-tuning, the system delivers high-quality output and a strong user experience, making it a powerful complement and alternative to existing subscription-based products.

References

- Technical Report: Fine-Tuning GPT-4o-mini for Video Content Summarization
- Langflow
- AssemblyAI
- OpenAI
- GitHub Issue: Langflow YouTube Integration