

Development of a Multimodal LLM for Video Understanding Applications

Nazarbayev University
CSCI 409 Senior Project II
Final Project Report
Group 36
April 20, 2026

Group Members: Sayat Serik, Yerasylzhan Burkitbay, Aisaule Zaneshova, Amina Yerdas
Advisor: Nguyen Anh Tu

1. Executive Summary

This project develops a video understanding system based on multimodal Large Language Models (LLMs). We frame the video understanding ability to be able to perform specific tasks such as captioning and visual question answering (VQA) depending on the video provided. Therefore, our main goal is to develop a system using LLMs with aforementioned capabilities. In addition, as a basis for data we used CapERA dataset, a dataset with captioned Aerial Videos and set an objective to extend this dataset to extend the understanding not only to Aerial Video but also action recognition. These were our two main objectives.

To achieve these objectives, we experimented and designed 2 setups of training LLMs. First is based on using the non-visual part, decoder part, of the models. This setup inputs not videos but sampled frames processed into feature vectors via CLIP and custom Multi-Layer Perceptron. The second setup uses the built-in visual pipeline of existing models such as GIT and Gemma-3-4b. For VQA, we decided to use the simple first setup but adapted the input to include the question tokens too. We have developed and trained 4 decoder-only and 2 built-in setups for captioning with decent metric scores. To be exact, the evaluations have shown CIDEr scores in ranges from 0.65 up to 0.86.

Before training the models, however, to facilitate both captioning and VQA tasks, we created a single multimodal dataset containing all 1,500 manually generated captions from the UAVHuman dataset combined with the CapERA dataset. This resulted in a combined caption dataset of 4,364 total videos. Upon completion of the merged dataset, a VQA dataset was developed over the top of it. Ultimately, this produced 10,711 QA (question/answer) pairs with an average of 2.45 QA-pairs per video. Therefore, the project delivers not only a video captioning pipeline, but also a broader framework for multimodal video understanding.

The project fits into the whole computing-solution lifecycle: it creates two architectural approaches (decoder-only with a custom temporal encoder and end-to-end fine-tuning with a built-in visual pipeline), implements them across six model configurations with a shared training and inference codebase, and tests them using standard captioning metrics on held-out test data, along with VQA metrics and user feedback.

To wrap it all under a testable and observable result, we deployed the inference into a serverless backend and wrapped it in an interface.

2. Introduction

Videos are one datatype that can be very dense in data due to the sheer size of files even though it takes only a few seconds. The size only gets bigger the more precision in visual quality we want. In addition, videos are prone to human error. Most people are most likely to miss some detail or give wrong answers when asked simply because of association and visual perception biases they form depending on the content consumed. Thus video understanding applications can be crucial assistants in data correction, by providing an assistant to double check or process large amounts of data faster than it is manually done. This project is significant for such use cases because it develops a general training, testing and evaluation setup to develop video systems

capable of classifying threats on videos, simply put do surveillance in higher throughputs. It can be reframed to any task provided that sufficient data is given.

As described above, our solution has 2 setups, decoder-only and built-in, that can be adapted to 2 tasks (captioning and VQA) and trained and evaluated on 2 datasets (original CapERA and CapERA extended with UAVHuman). The extended dataset is generated using the same technique described in Bashmal et al. and transformed into a QA dataset by generating the question and answers from the captions. This paper will: identify the constraints we faced, provide brief literature review on the existing approaches, describe the approach and execution of the project, describe and show evaluations, and conclude with future prospects.

3. Identification of Constraints

The project was affected by a number of different types of constraints that all had an effect on design, implementation, deployment, and operation.

Economic. The main economic problem was the cost of GPU computing. It takes a lot of GPU memory to train multimodal video models, and cloud GPU time is charged by the hour. This directly influenced the choice of models (smaller ones like T5-small and Gemma-3-270M when possible), the training strategy (LoRA instead of full fine-tuning on Gemma-3-4B and GIT), and the number of frames sampled per video (4–8 instead of dense sampling). It also made it so the team could only do a certain number of ablation runs.

Environmental. Training and fine-tuning big models uses a lot of electricity, which means that carbon emissions happen at the datacenter level. To lessen their impact on the environment, the team chose parameter-efficient fine-tuning, cached CLIP features once and used them again and again, and didn't do unnecessary retraining. These choices lower costs and energy use, but they also limit the search for other architectures.

Social. The system works with videos of people and is positioned as a tool for tasks like finding threats through surveillance. There is a social risk of misuse with this, such as monitoring certain groups too closely or automating decisions that should be made by people. The team dealt with this at the design level by making the system an assistant that checks human judgment instead of replacing it and only using publicly available datasets that have been used in academic research.

Legal and political. Different places have different rules for surveillance-adjacent AI, and some pretrained models can't be exported or used. The project relies on publicly accessible datasets (CapERA, UAV-Human) and openly licensed models (Gemma, Qwen, GIT, T5, CLIP). The licenses were followed, and the work was limited to academic research. A legal review that is specific to the jurisdiction would be needed for a production deployment, which is not in the scope of this.

Ethical. When you work with videos of people doing things that can be identified, it raises privacy issues. Also, captions and answers that are automatically generated can be wrong or biased in scenes that aren't clear. The team utilized datasets in which subjects consented to research utilization, produced alternative captions via a filtering pipeline that verifies semantic

fidelity, and identified the hallucination risk in the Future Work section. The system has bias because it uses pretrained models like CLIP, SigLIP, and LLMs, and it can't be completely removed at this scale.

Safety and health. Real threat detection systems like this one would have safety risks because they could miss threats on one side and wrongly accuse people on the other. The current system is a research prototype and should not be used in situations where safety is important.

Sustainability. Beyond environmental cost, sustainability also covers maintainability. The decoder-only pipeline was designed to be modular: the temporal encoder, projection layer, and backbone LLM are each swappable, so the system can be updated as stronger models are released without rewriting the training loop. This extends the useful lifetime of the codebase.

Collectively, these constraints produced a central trade-off: the decoder-only approach is cheap, fast to train, and more accessible to resource-constrained teams, but it trades some caption quality compared to the built-in visual pipeline. The project mitigates this by providing both approaches so downstream users can choose based on their own constraints.

4. Background and Related Work

The survey on video understanding describes multiple approaches to designing video understanding LLMs such as LLM as decoder/summarizer/regressor and others, even though early focus was typically on assigning one class label to a short video clip (Tang et al., 2024). We use the LLM as a decoder setup, since the more tightly coupled integration of visual understanding with language generation is more advanced and prone to error. Even the built-in visual pipelines use this setup for its simplicity even though the handling is different from the usual LLM as decoder setup. However, current reviews on video based large language models indicate that there is an increasing trend towards tightly integrating visual understanding with language generation capabilities so that models can describe actions, track events as they occur over time and respond to open ended questions about video content (Tang et al., 2024).

The need for strong visual-language alignment before inputting features into a language model is supported by the example of Video-LLaVA (Lin et al., 2024) Lin et al. demonstrated that misaligned visual features can lead to very inconsistent and semantically weak outputs. This motivated the design of a custom MLP projection module for each model to map the CLIP-extracted video feature into the embedding space of the corresponding LLM. The MLP is the explicit alignment layer for the decoder-only setups.

Wang et al. signified the importance of data quality and scale by finding that simply increasing the sheer volume of data does not resolve the issue of low correspondence between the video content and the text (2024). This observation directly impacted the captioning methodology and dataset construction pipeline in this project, forcing us to be more careful.

Domain-specific challenges are addressed by CapERA (Bashmal et al. 2023), a dataset containing captioned aerial UAV videos. The videos have distinct difficulties such as the small scale of humans relative to their surroundings, dynamic camera position, and environmental

conditions. But they also introduce a methodology for generating captions, using paraphrasing and back-translation. We too adopt the same methodology when extending the dataset with UAV-Human videos, ensuring consistency in caption quality and diversity across both sources.

To sum up, these works point to 2 core requirements: capable vision encoder, and a well-designed alignment layer. The decoder-only approach was selected due to being fairly simple to implement and adapt and integrate with the core requirements, using CLIP as a vision encoder and a custom MLP as projection layer. The build-in pipeline approach, using GIT (Wang et al., 2022) and Gemma-3-4B serves as a higher quality comparison point as these models handle alignment internally through architectures specifically designed for multimodal generation.

5. Project Approach

System Overview

The system is a multimodal video understanding pipeline that takes a video as input and produces either a natural language caption or an answer to a question about the video. At a high level, the system has four main components: a feature extraction stage, a projection layer, a language model, and an inference backend with a frontend interface. These components are connected in sequence. Video frames are first encoded into numerical feature vectors, those vectors are projected into a format the language model understands, and the language model then generates text output. The backend exposes this pipeline through an API, and the frontend allows a user to upload a video and receive a response.

Hardware and Compute Constraints

Training was conducted on NVIDIA A100 and L4 GPUs accessed through cloud computing. The A100 (40GB VRAM) was used for training larger models such as Gemma-3-4B, while smaller models like T5-small and Qwen2.5 were trained on fewer resources. Inference is served on a single L4 GPU via Modal, a serverless cloud platform. These hardware constraints directly influenced several design decisions: the number of frames sampled per video was kept low (4–8 frames), large models were fine-tuned using LoRA rather than full fine-tuning, and smaller model variants were preferred where possible.

Third-Party Components and Their Integration

The project relies on several pre-built models and libraries that the team did not implement from scratch:

CLIP (ViT-L/14) - developed by OpenAI, CLIP is a vision encoder trained on image-text pairs. It was used to extract visual features from sampled video frames. The team integrated CLIP using the OpenAI CLIP library, passing frames through the encoder and averaging the resulting feature vectors across frames (mean pooling) to produce a single fixed-size representation per video.

T5-small - a small encoder-decoder language model developed by Google. It was integrated in decoder-only mode for captioning and adapted for VQA by feeding video features through a gate mechanism alongside encoded question tokens.

Microsoft GIT (Generative Image-to-Text Transformer) - a vision-language model developed by Microsoft that includes a built-in CLIP-ViT visual encoder. It was used in two ways: with its visual pipeline bypassed (decoder-only setup) and end-to-end with its native visual pipeline (built-in setup).

Qwen2.5 - a decoder-only language model developed by Alibaba. It was used in the decoder-only captioning setup, with a custom MLP added to map CLIP features into its embedding space.

Gemma-3 - a family of language models developed by Google DeepMind. The 270M variant was used in the decoder-only setup. The 4B variant was used in both setups and, due to its size, was fine-tuned using LoRA. Gemma-3-4B uses a SigLIP vision encoder in its built-in pipeline and distinguishes visual from text tokens using token type IDs.

LoRA (Low-Rank Adaptation) - a parameter-efficient fine-tuning technique implemented via the Hugging Face PEFT library. Instead of updating all model weights, LoRA inserts small trainable matrices into the attention layers, drastically reducing memory usage and training time. It was applied to Gemma-3-4B and GIT in the built-in pipeline setup.

Hugging Face Transformers - the primary library used to load, configure, and fine-tune all language models. It provided tokenizers, model classes, and training utilities.

Modal - a serverless cloud platform used to deploy the inference backend. It handles GPU provisioning, container startup, and request routing without requiring a permanently running server.

FastAPI - a Python web framework used to build the three inference endpoints (/health, /caption, /vqa).

Approach 1: Decoder only setup

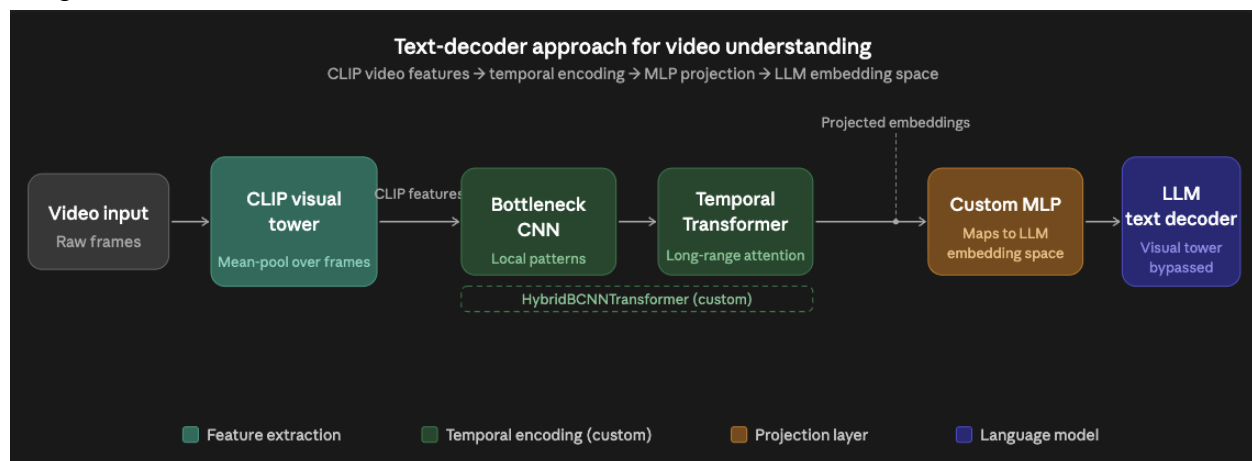
In this setup, if there is an internal visual pipeline of the model, it is bypassed entirely. We use CLIP ViT-L/14 to extract features from 8 sample frames per video, these features are mean-pooled to produce 1 vector per sample frame. Before projecting features into the LM, they

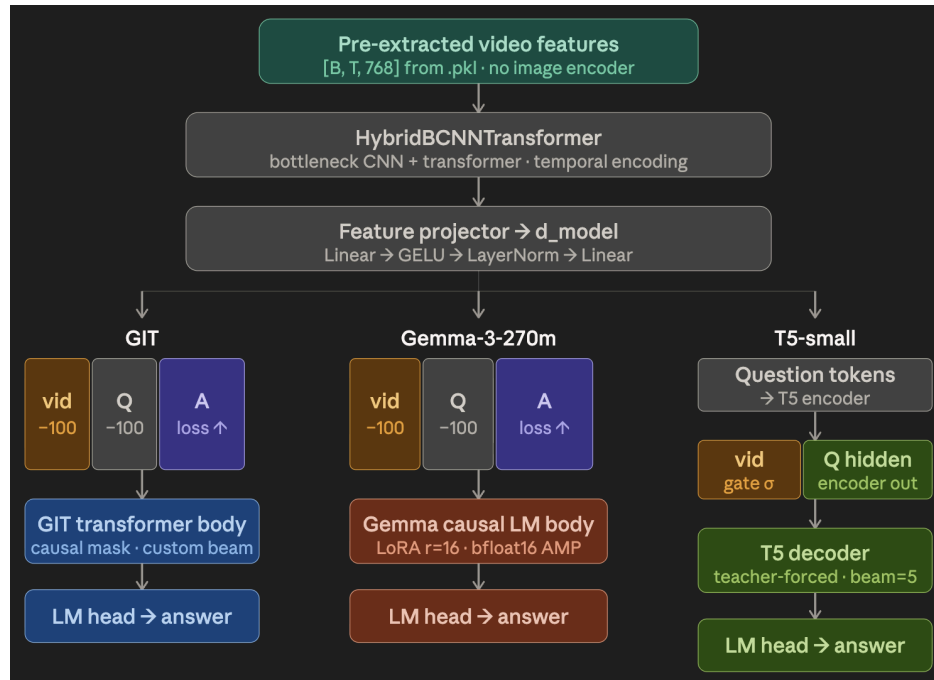
are processed by a temporal encoding module, we used Bottleneck CNN and Transformer. Bottleneck CNN compresses the feature dimension to smaller bottleneck size, and applies convolution across the time axis for local patterns. Then Transformer applies self-attention across frames to capture long-range dependencies in the relationship between frames. Then the output is then mapped to LLM embedding space via Multi-Layer Perceptron. The projected features are prepended to the text tokens and become an input to the language model's decoder.

Despite the general approach being similar, each model has special features which require different handling of inputs. GIT uses causal attention masks so that text tokens can attend to the visual tokens, while Gemma-3-4B uses token_type_ids to distinguish between visual and text tokens; And T5-small is a text-only model.

Overall this approach is very cheap, but it trades off quality for simplicity. The reason is the language models were not initially trained to perceive features in this format. For instance, Gemma-3-270m and t5-small are text only LLMs, while GIT and Gemma-3-4b have their own visual pipeline.

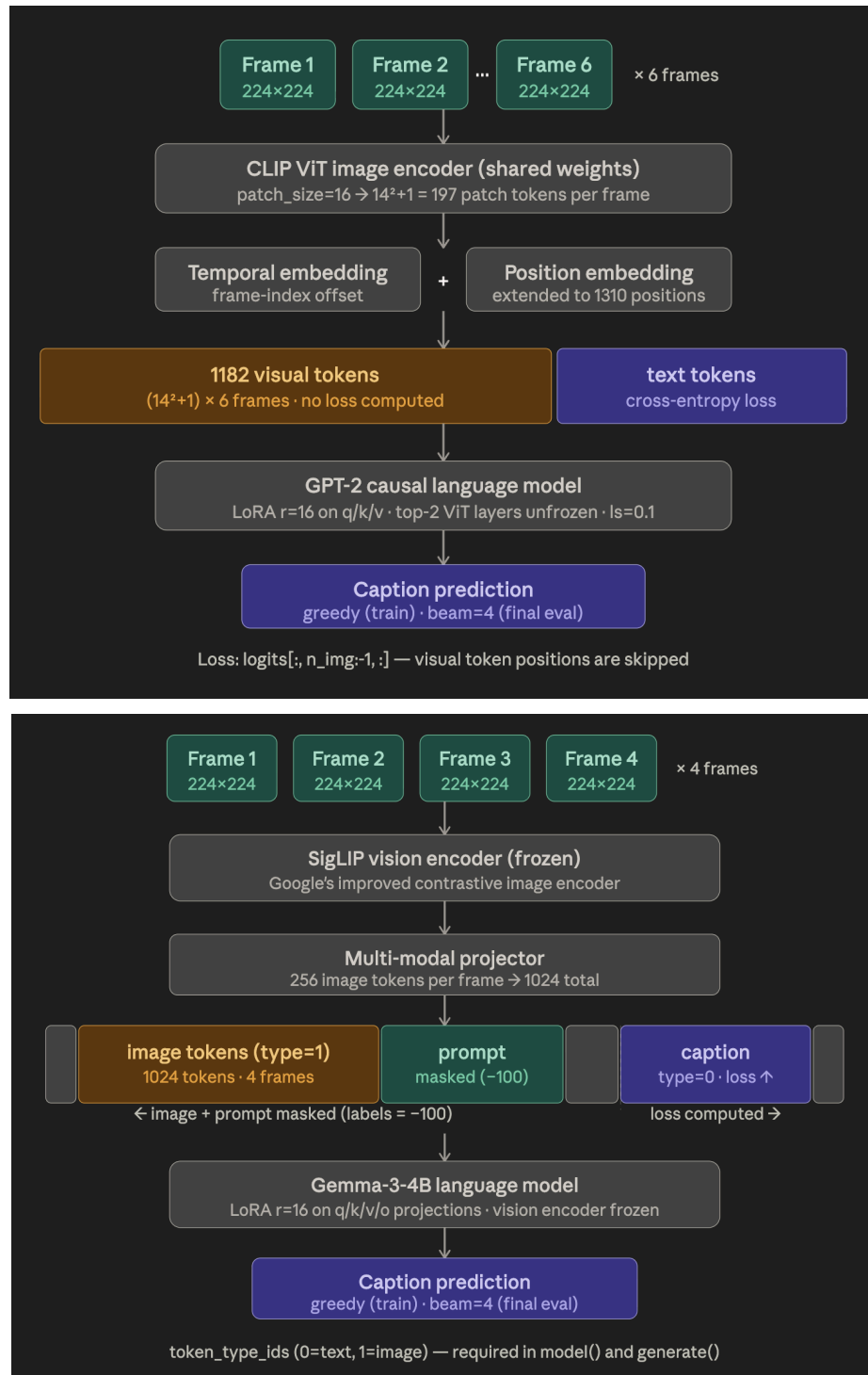
For VQA, this setup was reused with a slight modification of input. The tokens for question's text were appended after the video features tokens, and both were label masked during loss computation.





Approach 2: Built-In Visual Pipeline

In this setup, raw video frames are input into the models as intended. GIT processes 6 frames through a built-in CLIP encoder which produces visual patch tokens. The visual patch tokens are then passed through its GPT-2 based transformer decoder. Gemma-3-4B, on the other hand, processes 4 frames through SigLIP, producing 256 image tokens for each frame. Then, the image tokens are concatenated with text tokens and passed into the language model. Both models were fine-tuned using LoRA. This approach is much more expensive and compute hungry to train but produces higher quality captions because the visual encoding and language generation are both fine-tuned on the target domain.

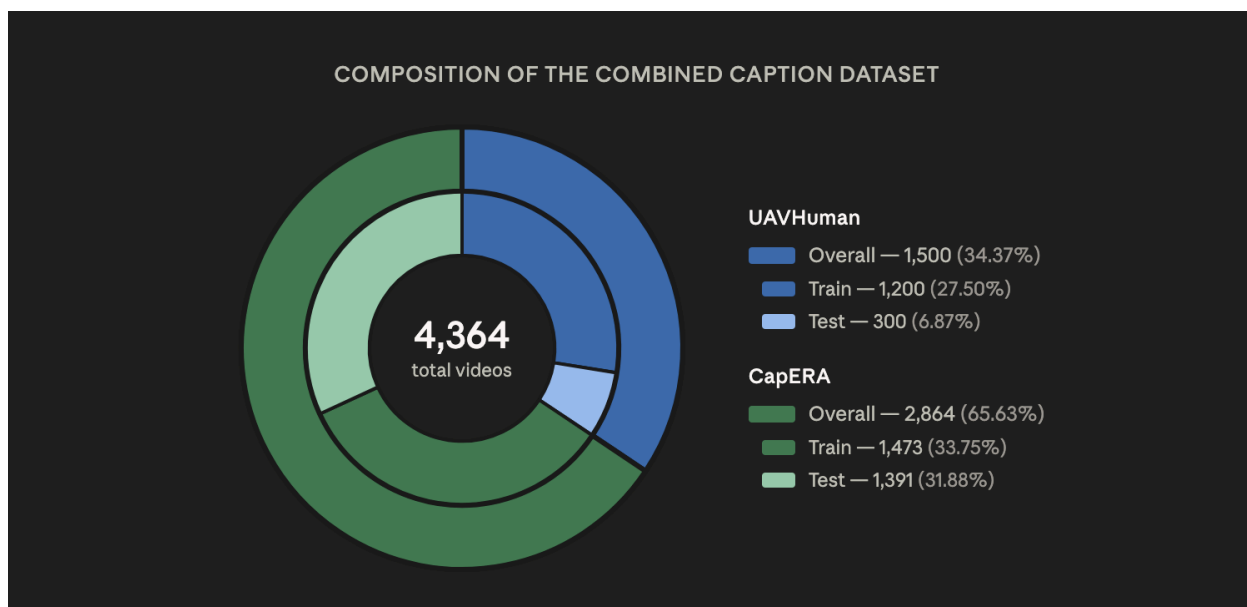


Dataset Construction and Preparation

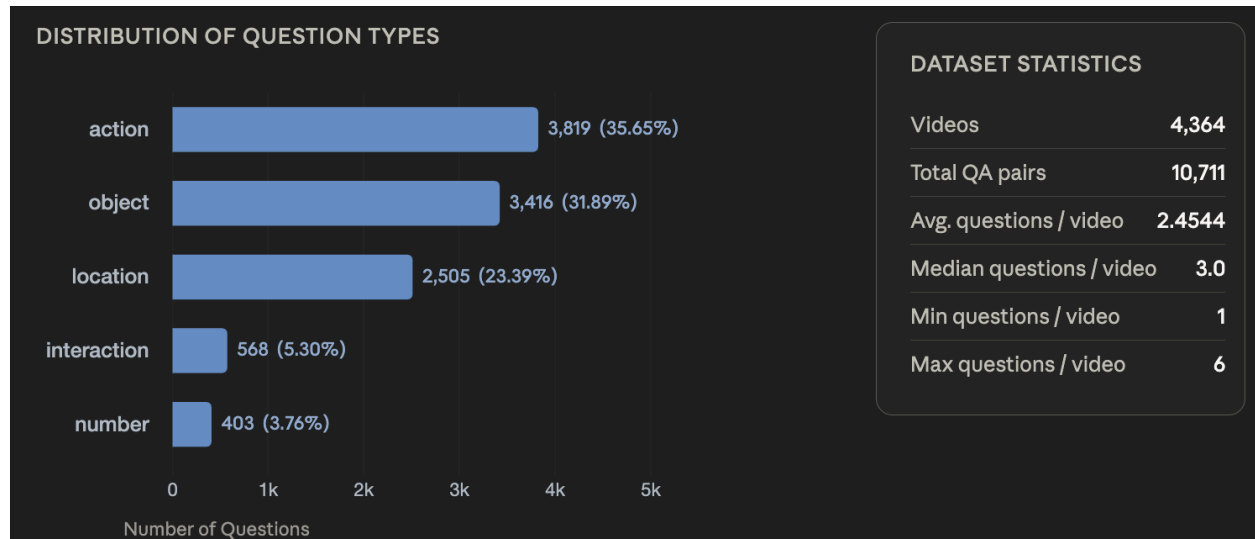
In order to help us work on the Video Captioning model, we developed a unified multimodal dataset by integrating our own annotated version of UAVHuman into the CapERA dataset. In developing this dataset pipeline, there were four stages involved.

The first stage was manual captioning. A total of 1500 videos from UAVHuman were manually captioned in order to develop grounded primary captions. Next, to improve the diversity of captions, we duplicated the method used in CapERA. For each primary caption, we generated 10 possible alternative captions: five paraphrased captions using an LLM (gpt-5.4 in our case) and five back-translated captions from multiple languages (french, german, spanish, italian and russian). Next, we chose the five most suitable of the ten options using a filtering pipeline that utilized a combination of semantic similarity and diversity metrics. This allowed each video to be paired with multiple valid descriptions while keeping the captions faithful to the original visual content. The filtering code uses sentence embeddings and a Maximal Marginal Relevance-style selection strategy to keep the alternatives both relevant to the primary caption and different from one another.

Next was standardizing and merging UAVHuman dataset and feature extraction. Visual features were extracted using 8 frames per video to match the data structure used in the CapERA dataset. This resulted in both datasets being able to share a common training and testing pipeline. The resulting merged dataset had 1500 captioned UAVHuman videos and 2864 captioned CapERA videos for a total of 4364 videos, with each video having one primary and 5 alternative captions.



Finally, as the project evolved and as we added VQA capabilities to our model, it also needed a dataset. The VQA dataset was constructed from the merged dataset constructed earlier. All question/answer pairs were generated using gpt-5.4, directly from the primary captions. Using information from the primary caption 5 types of questions were generated, which are: action, object, location, interaction and number. Unsubstantiated/hallucinatory questions were removed, duplicate pairs were eliminated, and remaining pairs were reviewed for consistency. The final VQA set is composed of 4364 videos and 10711 question/answer pairs with an average of 2.45 questions per video.



Overall Workflow

A video enters the system and is decoded into individual frames. A fixed number of frames are sampled uniformly. In the decoder-only setup, these frames are passed through CLIP and the resulting features are averaged and projected via MLP. In the built-in setup, raw frames are passed directly to the model's visual encoder. The resulting visual representations are combined with text inputs (a prompt for captioning, or a question for VQA) and fed into the language model decoder, which generates output token by token. During inference, this entire pipeline runs inside a Modal container and is triggered via a POST request to the FastAPI backend.

6. Project Execution

Semester 1

The project began with researching the GIT and Gemma 3 and datasets in parallel, and experimenting with straightforward integration approaches. We started manually captioning UAV-Human videos, initially completing 1000 captions by the end of first semester. Early on, the first challenge was that the videos varied in content so the captions the team members were writing was varying in level of detail and format. We established how it would be written and formatted before continuing. This was time consuming but necessary, as the correspondence of data is very important for captions.

When it comes to experimenting with GIT and Gemma-3 using their built-in visual pipeline and using them as-is, we encountered a problem. Even though our setup ran without errors, captioning quality was very poor. The correct way to structure inputs, handle the maskings, loss computations for video captioning were not straightforward, especially with GIT since its implementation in the transformers library was incompatible with new versions and later on we figured a way to fix it but not in this semester. The progress stalled at this point but we made an

effort to learn and try out new things like LoRA. Although it did not produce practical results, it gave a working foundation to build the correct training pipeline.

Semester 2

Following discussion with the advisor, we reframed the approach. Instead of using the internal visual pipeline, we decided to bypass them entirely and feed pre-extracted CLIP features into the model decoder. It significantly simplified the setup and made the approach a little bit more model-agnostic. It allowed us to focus on getting one clean pipeline to work right.

At the start of the semester, the responsibilities were divided in two. Dataset group and model group. The dataset was expanded to 1500 videos. After the original 1500 primary captions were generated using the same methodology as previously outlined (CapERA-style), five paraphrases and five back-translations were added to each. From these ten alternatives, the five best ones were selected based on semantic similarity & diversity through a filtering pipeline. The UAV-Human subset was then standardized, merged with CapERA, and used as the basis for the final VQA dataset. One of the major problems encountered at this point was creating a consistent captioning format. Since there were multiple contributors to the manual captioning of the original video frames, the original captions included varying amounts of detail, were of variable length and often included spelling errors along with minor variations in formatting conventions. Thus, processing and cleaning all 1,500 captions became a very significant portion of our overall task since the quality of the additional captions depended upon the quality of the original captions. Another decision we had made was in the methodology of constructing the VQA dataset. Since we were nearing the end of the semester when adding VQA to our model, we went with a much quicker data build methodology, by only using the main caption for creating all the questions. It allowed us to complete this part of the project within the project time frame, however, as a result the VQA dataset produced was less complex than most other QA type datasets created from more detailed annotations or through multiple steps of editing.

The model group implemented the decoder-only pipeline across four models. Each with different handling of masking and token structures. The team also designed and implemented the custom HybridBCNNTransformer temporal encoding module, which processes CLIP features through a bottleneck CNN for local patterns and a Transformer encoder for long-range dependencies before projection into the LLM. This module was built to be swappable, allowing the team to experiment with different temporal configurations during training. The same decoder-only architecture was then adapted for VQA by appending question tokens after the video embeddings. The built-in pipeline experiments with GIT and Gemma-3-4B using LoRA were run as a second approach, producing the best overall results with CIDEr scores reaching 0.86.

Both groups collaborated on the inference backend and frontend. The Modal serverless backend and FastAPI endpoints were built and tested together, and the local JavaScript frontend, covering chat, metrics, and project information, was developed jointly. Evaluation and metric analysis were also shared across the team.

What Went Well

The decision to pivot to the decoder-only approach early in semester two gave the team a working, extensible pipeline quickly. The modular design of the temporal encoder meant configurations could be swapped without rewriting training code. The dataset pipeline, once the captioning quality issues were resolved, produced a clean and well-structured combined dataset that supported both tasks.

What Was Difficult

Getting the training pipelines right for each model took significantly longer than expected, particularly around attention masking and loss computation. The captioning inconsistencies in semester one added unplanned cleaning work. The VQA dataset, while functional, was constrained by the time available and is simpler than comparable benchmarks. Hardware limitations throughout the project restricted the number of training runs and configurations that could be tested.

7. Evaluation

To evaluate whether the system successfully solves the problem, we adopted a standard suite of natural language generation metrics used in video captioning literature. The suite includes BLEU-1-4, CIDEr, METEOR and ROUGE-L. These metrics were computed with references to all five human-annotated reference captions provided in the CapERA test set. A system that scores well across these metrics, particularly on CIDEr which is specifically designed for captioning tasks, can be considered to have learned describing aerial events.

Our first approach produced strong results across all four models. On the original test set, Microsoft/GIT achieved the highest BLEU scores (BLEU-4: 0.3945, CIDEr: 0.7640), while Gemma-3-270M led on METEOR (0.4230) and produced competitive CIDEr (0.7418), indicating richer semantic overlap with human references. T5-small proved surprisingly competitive given its size (CIDEr: 0.6705, BLEU-4: 0.3374), and Qwen2.5 performed consistently if slightly below the others. Training on the extended dataset (which added data augmentation and more caption variety) yielded broadly similar results, with Gemma-3-4B emerging as the strongest on METEOR (0.4065) and CIDEr (0.7040), confirming that larger language models better capture the descriptive style of human annotators even with fewer training epochs.

Our second approach, fine-tuning GIT and Gemma-3-4B end-to-end on raw video frames through their built-in visual pipelines with LoRA, produced the best absolute numbers overall: Gemma-3-4B reached CIDEr 0.8640, BLEU-4 0.2110, METEOR 0.2189, and ROUGE-L 0.4560, while GIT reached CIDEr 0.8319 and ROUGE-L 0.4573. The CIDEr improvement from ~ 0.76 to ~ 0.86 for GIT and ~ 0.70 to ~ 0.86 for Gemma-3-4B between the two approaches demonstrates that end-to-end visual fine-tuning, when stabilized with LoRA and proper gradient clipping, extracts richer aerial-domain representations than decoupled feature extraction.

CAPTIONING RESULTS ON THE ORIGINAL CAPERA TEST SET.

Model	B-1	B-2	B-3	B-4	CIDEr	METEOR	ROUGE-L
T5-small	0.5855	0.4738	0.3955	0.3374	0.6705	0.1920	0.4092
Microsoft/GIT	0.6550	0.5401	0.4575	0.3945	0.7640	0.2319	0.4355
Gemma-3-270M	0.3184	0.1972	0.1328	0.0870	0.7418	0.4230	0.4238
Qwen2.5	0.2688	0.1600	0.1061	0.0684	0.6581	0.3793	0.4007

CAPTIONING RESULTS ON THE EXTENDED CAPERA TEST SET.

Model	B-1	B-2	B-3	B-4	CIDEr	METEOR	ROUGE-L
T5-small	0.5857	0.4750	0.3983	0.3419	0.6494	0.1964	0.4112
Microsoft/GIT	0.6743	0.5516	0.4657	0.4011	0.7270	0.2154	0.4192
Gemma-3-270M	0.3040	0.1929	0.1351	0.0923	0.6535	0.3843	0.3875
Gemma-3-4B	0.3166	0.2005	0.1424	0.1002	0.7040	0.4065	0.4093

CAPTIONING RESULTS USING BUILT-IN VISUAL PIPELINES (RAW VIDEO FRAMES).

Model	B-1	B-2	B-3	B-4	CIDEr	METEOR	ROUGE-L
Microsoft/GIT	0.489	0.3701	0.2722	0.2268	0.8319	0.2180	0.4573
Gemma-3-4B	0.527	0.387	0.291	0.211	0.8640	0.2189	0.4560

VQA evaluation using manual testing. Instead of using reference-based automatic metrics, the team conducted systematic manual testing to evaluate the VQA pipeline qualitatively. This decision was made on purpose: the VQA dataset was created towards the end of the second semester, and since its responses come from primary captions rather than direct video inspection, standard exact-match and F1 scores against these references would primarily measure agreement with caption-derived answers rather than true video comprehension. Future Work prioritizes metrics like ANLS and BERTScore that are more appropriate for free-form VQA.

Each of the four team members used the deployed frontend to upload a variety of videos for manual testing, which included both unseen clips and videos from the CapERA and UAV-Human test splits. They then asked questions that covered all five of the question types in our dataset (action, object, location, interaction, and number). We evaluated each response based on three factors: whether it was based on the video content rather than being a hallucination; whether it was relevant to the question; and whether it used natural language fluently. For action and object questions, where the answer space is strictly limited by the training captions, responses were consistently grounded and on-topic across most models deployed on the backend (T5-small, GIT, Gemma). Questions about locations also did well. The weakest category was interaction and number questions; responses were still coherent and believable, but they were frequently vague or insufficiently detailed. This is consistent with the restriction that the training signal for these question types comes from brief primary captions rather than in-depth annotations. The

Appendix contains representative samples of each circumstance. The deployed frontend was used during development by the team, not by outside evaluators. A formal user study is listed as a priority extension in Future Work.

Revised Section 8: Conclusion and Possible Future Work

Summary of findings and Contributions

We successfully developed a video understanding system capable of two tasks: video captioning and visual question answering. The work produced two concrete contributions. Firstly, a dataset with 4,364 videos with captions and 10,711 question-answer pairs. Secondly, a modular training and inference pipeline that was tested across six configurations and two distinct strategies.

The most significant technical finding was that end-to-end fine-tuning with the built-in visual pipeline consistently outperforms the decoder-only approach. Gemma-3-4B with its native SigLIP encoder reached a CIDEr score of 0.86, compared to 0.70 when trained in the decoder-only setup. GIT showed a similar improvement, rising from 0.76 to 0.83. These results can be explained by the fact that jointly fine-tuning the vision encoder and language model on the target domain extracts richer representations. However, the decoder-only approach is still fairly competitive and significantly cheaper to train, making it a practical baseline for resource-constrained settings.

Future Work

Priority evaluation extensions. Two evaluation gaps in the current work should be addressed first in any continuation. First, a quantitative VQA evaluation using semantic-aware metrics such as ANLS and BERTScore, alongside exact match and token F1, would make the VQA pipeline's quality directly comparable to published benchmarks. To avoid the circularity noted in Section 7, this evaluation should be run against a held-out QA set generated from detailed annotations rather than from primary captions. Second, a structured user study in which external participants upload videos through the deployed frontend, receive captions and VQA responses, and rate relevance, fluency, and usefulness on a short questionnaire would provide evidence of practical utility that automatic metrics cannot capture.

Dataset improvements. The most immediate improvement to the underlying data is richer VQA annotations. Because questions in the current dataset were generated only from primary captions rather than from direct video observation, the resulting QA set is simpler than established VQA benchmarks. Future work should generate questions from more detailed annotations so that interaction and number questions - the weakest categories in our manual testing - are supported by a stronger training signal.

Model improvements. Stronger visual encoders specific to the aerial-video domain could improve performance, since most current models face the UAV challenges described in Section 3. With more compute available, a systematic comparison of temporal encoding configurations beyond

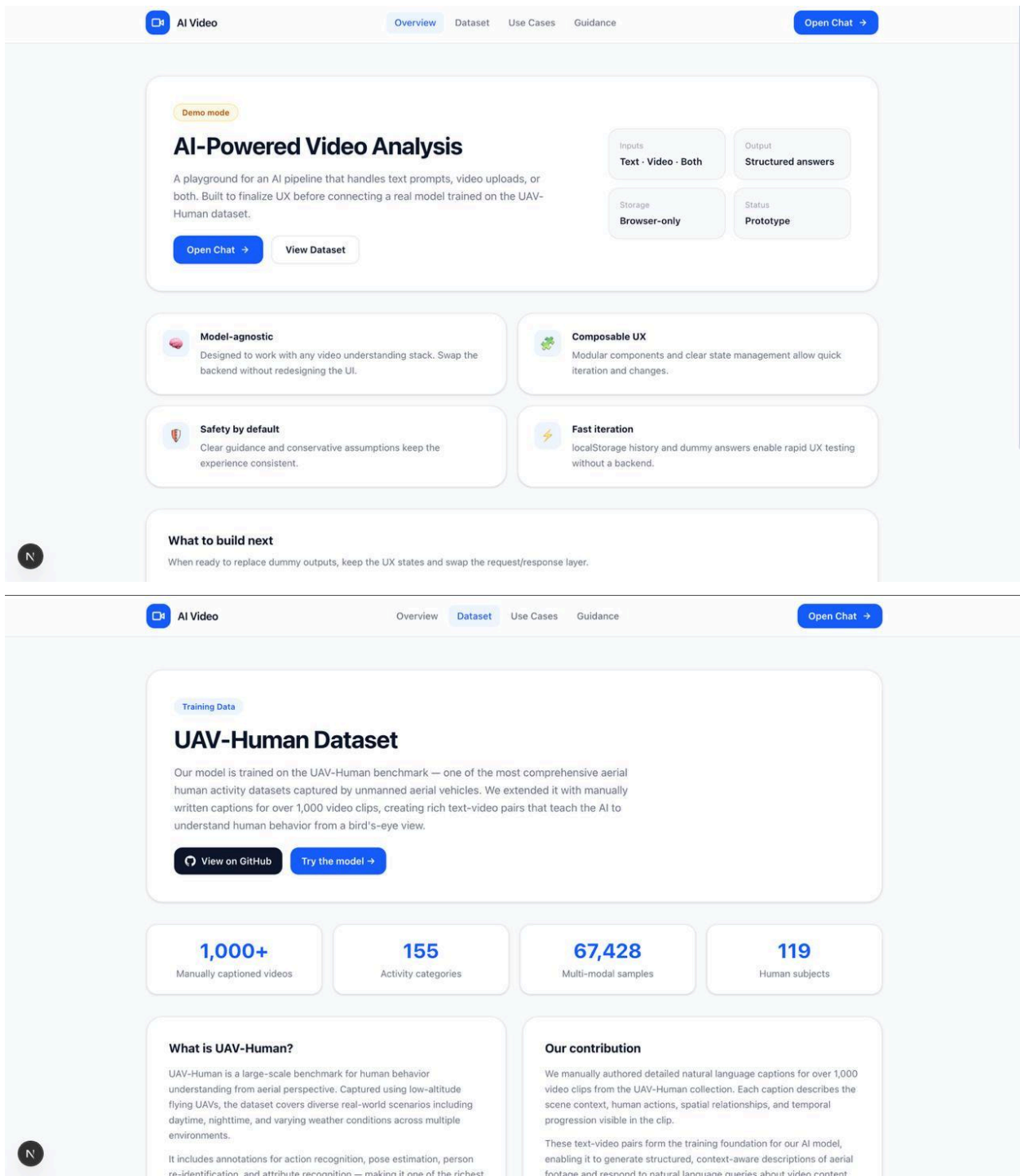
the BCNN + Transformer combination we used would be worthwhile, as would experiments with more advanced alignment layers such as cross-attention in place of a single MLP projection.

Additional tasks. Beyond captioning and VQA, the pipeline could be extended to other video understanding tasks. Event detection, where the system identifies salient moments or transitions within a video, and temporal retrieval, where the system locates the time at which a queried event occurs, are natural next steps given the same backbone architecture.

9. References

- Bashmal, L., Bazi, Y., Al Rahhal, M. M., Zuair, M., & Melgani, F. (2023). CapERA: Captioning events in aerial videos. *Remote Sensing*, 15(8), 2139. <https://doi.org/10.3390/rs15082139>
- Lin, B., Ye, Y., Zhu, B., Cui, J., Ning, M., Jin, P., & Yuan, L. (2024). Video-LLaVA: Learning united visual representation by alignment before projection. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.emnlp-main.342>
- Li, T., Liu, J., Zhang, W., Ni, Y., Wang, W., & Li, Z. (2021). UAV-Human: A Large Benchmark for Human Behavior Understanding with Unmanned Aerial Vehicles. *arXiv*. <https://doi.org/10.48550/arXiv.2104.00946>
- Mou, L., Hua, Y., Jin, P., & Zhu, X. (2020). ERA: A Dataset and Deep Learning Benchmark for Event Recognition in Aerial Videos. *arXiv*. <https://doi.org/10.48550/arXiv.2001.11394>
- Tang, Y., Bi, J., Xu, S., Song, L., Liang, S., Wang, T., Zhang, D., An, J., Lin, J., Zhu, R., Vosoughi, A., Huang, C., Zhang, Z., Liu, P., Feng, M., Zheng, F., Zhang, J., Luo, P., Luo, J., & Xu, C. (2024). Video understanding with large language models: A survey. *arXiv*. <https://arxiv.org/abs/2312.17432>
- Wang, Y., He, Y., Li, Y., Li, K., Yu, J., Ma, X., Li, X., Chen, G., Chen, X., Wang, Y., He, C., Luo, P., Liu, Z., Wang, Y., & Qiao, Y. (2024). InternVid: A large-scale video-text dataset for multimodal understanding and generation. *International Conference on Learning Representations (ICLR) 2024*. <https://doi.org/10.48550/arXiv.2307.06942>
- Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C., & Wang, L. (2022). GIT: A Generative Image-to-text Transformer for Vision and Language. <https://doi.org/10.48550/arXiv.2205.14100>

10. Appendix



AI Video

OverviewDatasetUse CasesGuidance

Open Chat →

Use Cases

Examples designed for the current UI — text prompts, video uploads, and hybrid inputs.



Video summarization
Turn raw aerial footage into clean, structured summaries with timeline and key moments.

VideoHybrid



Event detection
Find specific moments matching a condition — timestamps with evidence for each occurrence.

PromptHybrid



Quality checks
Validate whether a video meets specific requirements with pass/fail and improvement tips.

Prompt



Explain what happened
Plain-English explanations for non-technical users with simple terms.

Hybrid



Instruction extraction
Convert tutorial videos into numbered steps, prerequisites, and warnings.

VideoHybrid



Q&A on video content
Ask targeted questions and get direct answers with references to timestamps.

PromptHybrid

AI Video

OverviewDatasetUse CasesGuidance

Open Chat →

Guidance

Best practices for writing prompts, uploading videos, and interpreting outputs.

Clear prompts

Safe uploads

Structured outputs

Best input

Prompt + Video combined

Avoid

Personal data and private info

Goal

Clear, structured answers

Write better prompts
Strong prompts lead to better outputs.

- Specify the output format (bullets, timeline, JSON structure).
- State the audience (beginner, expert, internal reviewer).
- Define constraints (length, number of points, strictness).
- If you need timestamps, ask for them explicitly.

Upload responsibly
Treat video as sensitive data by default.

- Avoid faces, names, addresses, and ID documents.
- Keep clips short and focused for better results.
- Add a clear warning in the prompt for sensitive content.
- Production: add retention policy and deletion tools.

Interpret answers correctly
Teach users what to trust and what to verify.

- Ask for evidence: timestamps, quotes, or visible cues.
- Request multiple hypotheses for ambiguous content.

Recommended output structure
A consistent format helps both UI and users.

- Short summary (2–3 sentences)
- Key moments with timestamps

AI Video ChatCAPTIONGemma

Clear

Swimming_047 .mp4CAPTION · gemma02:35 PM

people swimming in a pool in a treelined garden27.5s02:35 PM

What is going on here?Swimming_047 .mp4VQA · gpt02:36 PM

swimming pools and a small lake with a big fish next to a small building26.8s02:36 PM

Swimming_047 .mp4CAPTION · gemma02:37 PM

group of people swimming in swimming pool in yard surrounded by trees27.8s02:37 PM

N

Type a question for VQA, or leave empty to caption...

History9

02:37 PMgemmaCaptionSwimming_047 .mp4group of people swimming in swimming pool in yard surrounded by trees

02:36 PMgptVQAWhat is going on here?swimming pools and a small lake with a big fish next to a small building

02:35 PMqwenCaptionSwimming_047 .mp4people swimming in a pool in a treelined garden

02:34 PMtsVQAWhat is happening in the video?swimming pool

11:39 AMqwenCaptionSwimming_047 .mp4people swimming in a pool in a treelined garden

11:39 AMtsVQAWhat is happening in the video?swimming pool

11:38 AMgptVQAWhat is happening in the video?

Frontend pages

Parameters

Value

Reset

Name	Description
question required string (query)	Question to ask about the video What is happening in the video?
model string (query)	Model to use: t5 t5

Request body required

multipart/form-data

video required
string

Choose File

Soccer_001 .mp4

Execute

Clear

Responses

Curl

```
curl -X 'POST' \  
  'https://ssayat747--video-caption-fastapi-app.modal.run/vqa?question=What%20is%20happening%20in%20the%20video%3F&model=t5' \  
  -H 'accept: application/json' \  
  -H 'Content-Type: multipart/form-data' \  
  -F 'video=@Soccer_001 .mp4;type=video/mp4'
```

Request URL

```
https://ssayat747--video-caption-fastapi-app.modal.run/vqa?question=What%20is%20happening%20in%20the%20video%3F&model=t5
```

Server response

Code	Details
200	Response body <pre>{ "model": "t5", "question": "What is happening in the video?", "answer": "people are playing football in a stadium", "runtime_ms": 36616 }</pre> Download

Calling Inference for t5 VQA from backend API

Name	Description
question * required string (query)	Question to ask about the video What is happening in this video?
model string (query)	Model to use: t5 gemma git git

Request body * required

multipart/form-data

video * required
string

Responses

Curl

```
curl -X 'POST' \
  'https://ssayat747--video-caption-fastapi-app.modal.run/vqa?question=What%20is%20happening%20in%20this%20video%3F&model=git' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'video=@Soccer_001 .mp4;type=video/mp4'
```

Request URL

```
https://ssayat747--video-caption-fastapi-app.modal.run/vqa?question=What%20is%20happening%20in%20this%20video%3F&model=git
```

Server response

Code	Details
200	Response body <pre>{ "model": "git", "question": "What is happening in this video?", "answer": "people are playing football in a field surrounded by trees and adjacent to several other fields", "runtime_ms": 40733 }</pre> Response headers <pre>alt-svc: h3=""; ma=2592000 content-encoding: zstd content-type: application/json date: Wed, 01 Apr 2026 11:31:45 GMT modal-function-call-id: fc-01KN4C26JH3577X7NVJRFRNYWH vary: accept-encoding</pre>

Calling Inference for t5 VQA from backend API

Link to Repository:

https://github.com/saikoda11/sp2025_vid_llm/blob/main/README.md?plain=1

Extended Dataset:

<https://drive.google.com/drive/folders/1nS5RERyo3z9BFvgut7RkbwOfdOIWNVhd?usp=sharing>