Hilait: Automatic Highlighting System Leveraging Facial, Audio, Text Sentiment AI

Tony Xia
tonyx717@stanford.edu

Danica Xiong
daxiong@stanford.edu

1. Introduction (Background and Setup)

1.1. Problem Statement

In recent years, the popularity of live streaming platforms for gaming, such as Twitch, YouTube Gaming, and Facebook Gaming, has surged exponentially, establishing a burgeoning industry around live video game content. Within this landscape, digital content creation, streamers and content creators face the significant challenge of efficiently identifying and highlighting key moments from hours of video footage. Manually sifting through lengthy videos to find exciting clips is time-consuming and labor-intensive. This problem is exacerbated for small streamers who may lack the resources to hire editors, thus limiting their ability to produce engaging short-form content for platforms like TikTok and YouTube.

We have developed Hilait, an Automatic Highlighting System that integrates multiple AI technologies to process diverse data inputs — video, audio, facial expressions, chat data, and game API events — in parallel. By doing so, we aim to create a comprehensive solution that not only reduces the workload for streamers and editors but also ensures the quality and relevance of the generated highlights.

We seek to answer whether it is possible to generate high-quality clips using a multimodal data approach and to determine the relative importance of each data type in producing these highlights. Our core constraints are:

- **Ease of Use**: The system must be user-friendly, allowing streamers to easily integrate it into their workflows without extensive technical knowledge or setup.
- **High Clip Quality**: The tool must consistently produce high-quality clips that capture the most engaging and relevant moments. Similar to what an editor would choose for their video.
- **Time Efficiency**: The system should significantly reduce the time streamers spend searching for and editing highlight clips. This includes being able to process video in a reasonable timeframe.

1.2. Inputs

To maintain the system's ease of use, streamers need only to input the ID of the Twitch video they want to highlight. This ID is taken directly from the Twitch video link. Our pipeline automatically handles the following processes:

- **Download Full VOD:** The system downloads the complete Twitch VOD as an MP4 file.
- **Chat Data:** The associated chat data is also downloaded and stored in a JSON file.

To incorporate game API data, users need to provide additional information due to limitations of Riot's Remote API. Currently, we handle this portion manually ourselves, but this process can be easily automated by building our app in C++ and using Riot's Local API, explained in the API section:

- **Gamer ID:** The unique Username and Hidden Tag of the streamer's gaming account.
- **Champions Information:** The names of the champions involved in their game.
- **Timestamp:** A single timestamp of in game time to ensure proper alignment with the video data.

1.3. Outputs

The primary output of our system is a series of highlighted video clips. These clips are generated based on specified intervals along with a corresponding score of how interesting the minute is. The default settings are is a clip length of 1 minute, however, this can be adjusted to produce shorter or longer clips as needed.

1.4. Problem Crux

Our project faced two primary challenges that were essential to its success:

1. **Combining Multiple Modalities of Data:** The first major challenge was integrating diverse data types

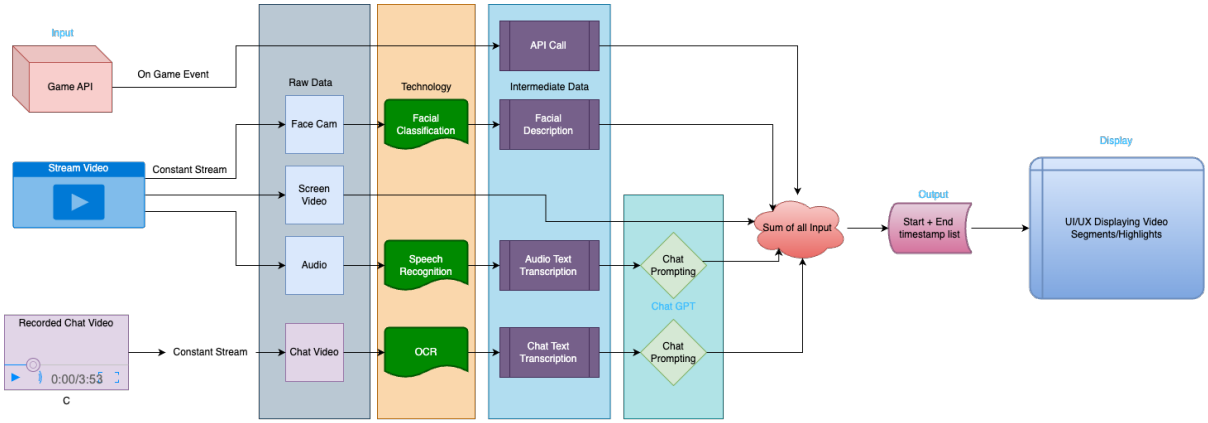

Figure 1. The following is our data processing pipeline. From the video Id, we get 3 inputs: Stream Video, Chat Video, and Game API. Eachis split into their raw data, ie. Face Cam, Screen Video, Audio, Chat video, be processed by their corresponding models, and then used as input to timestamp the video.

into a cohesive scoring system. Each data modality—video frames, audio streams, chat logs, and game API events—has its own temporal characteristics and formats include frames, seconds, event times (relative to game start time), chat (relative to video start time).

2. **Generating Meaningful Highlights:** The second challenge was to generate highlights that were not only accurate but also meaningful and engaging.

Many previous works only focus on a single input mode to highlight, ie.Video or Chat: *Video Highlights Detection and Summarization with Lag-Calibration based on Concept-Emotion Mapping of Crowd-sourced Time-Sync Comments* [5] and *Video Highlight Prediction Using Audience Chat Reactions* [2], *Unsupervised Extraction of Video Highlights Via Robust Recurrent Auto-encoders* [8] Furthermore, these papers do not have meaningful evaluation on how good their highlights were.

2. Approach

3. Video Pipeline

Due to Twitch restrictions on VOD downloading, we utilize the CLI tool, Twitch Downloader, to download full VODs. Given the video ID, we download the VOD in 1-hour segments and stitch them together using Ffmpeg. This combined video is then passed to the Audio and Facial pipelines for further processing. For testing purposes, the shortest VOD we processed was 2.5 hours, and the longest was 14.5 hours.

3.1. Facial Expression Recognition Pipeline

The Facial Expression Recognition (FER) pipeline is designed to analyze the facial expressions of the streamer throughout the VOD. The goal is to classify the streamer's facial expressions at fixed time intervals to determine if their emotion is worth highlighting.

The input is video frames from the MP4 in 20 frame intervals. The pipeline consists of two main components: a face detector and a facial expression classifier. The face detector identifies and clips the streamer's face from the entire video frame, while the facial expression classifier analyzes the clipped face to determine the streamer's emotion at that specific frame.

3.1.1 Facial Detection

To accurately recognize the streamer's face, we first employ YOLOv8 [4] to detect candidate faces within the entire stream window. The model outputs a list of detected faces. Next, using a pre-trained VGG network provided by DeepFace [7], we compute the similarity between each detected face and an identity image provided by the user. The face with the highest similarity score is selected and passed into the facial expression classifier. This ensures that the correct face is consistently identified throughout the video. (We determined this was necessary because one of the streamers searched up "hairlines" on Google and our system was overwhelmed by images of male faces and their hairlines.)

3.1.2 Facial Emotion Classification

The facial expression classifier processes the selected face and outputs a score for each of the seven emotion labels defined in the FER 2013 dataset [3]. This score is used as an indicator of the streamer's emotional state, helping to identify moments in the video that are worth capturing. The emotions are classified as ["angry", "disgust", "fear", "happy", "sad", "surprise", "neutral"] We trained several

architectures to classify these faces, but will not be going in to depth due to it being less relevant for this class.

3.2. Audio Pipeline

The Automatic Speech Recognition (ASR) component is a crucial part of our system, aimed at transcribing the streamer's voice lines for downstream sentiment analysis using GPT. We chose OpenAI's open source ASR model, Whisper [6], for this task due to its accuracy and robustness.

To process the audio, we first convert the MP4 stream video into an MP3 audio file. Whisper's transcription does not include timestamps, making it challenging to align the audio transcription with other components in downstream tasks. To address this, we segment the audio input into into 30-second clips and run the pipeline on each clip independently. This approach allows us to parallelize audio transcription.

After processing, we obtain a CSV file containing audio transcriptions segmented into 30-second intervals. This structured output by seconds is passed into downstream for Sentiment Analysis.

3.3. Chat Pipeline

The chat pipeline leverages a prebuilt tool, Twitch Downloader, which performs Optical Character Recognition on the chat and outputs a JSON file. The JSON file contains two key pieces of information: the timestamp of each message, relative to the start of the video, and the content of the message. This data is then passed to a sentiment analysis model for further processing.

3.4. API

There were several challenges we encountered when aligning the API to the video. API events were relative to the start of the game, and the start of the game is given in PST. All other data format times are relative to the start of the video. Due Twitch's lack of transparency for exact video start times, we could not align videos using the World Clock.

To align the API to the video, we took several steps: For each match, we record a single timestamp "GameTimeStamp" displayed in the video's top right corner, along with the corresponding video time "VideoTimeStamp". Additionally, we manually save the Streamer's username and the characters in the current game. We had to do this because usernames contained hidden tags which couldn't be transcribed with OCR. Furthermore, live game APIs could not be accessed unless our app was running locally on the player's computer. We also could not perform OCR on the character names because many of the streamers had "overlays" which is art that covers portions of their stream, including the character names. Thus, manual transcription was the most reliable option.

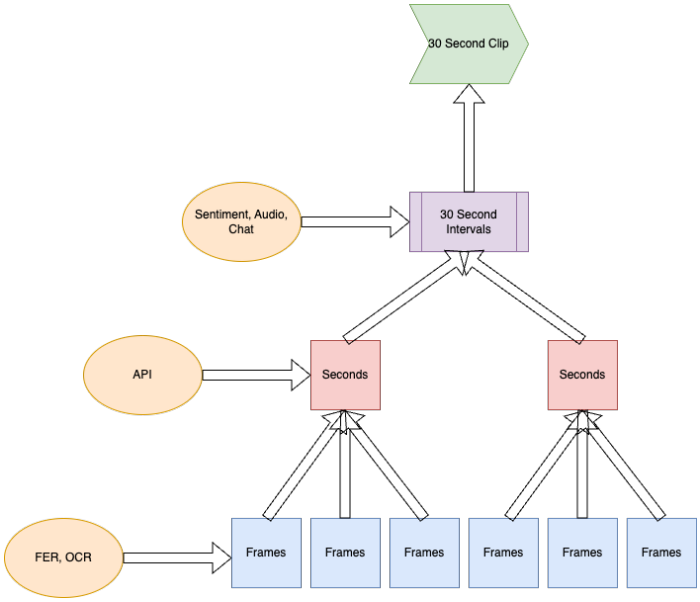

Figure 2. The following is our combination step where the results of all of our components are put into the final score. All components are parallelized and combined at the end

We then queried and cached data from the Riot API for the 200 most recent games played by the Streamer. We perform a search for matching character sets in every game. Upon identifying matches, we queried the Riot API for match timestep data, which returns player positions, states, and events occurring throughout the game. Events include monster kills, character kills, deaths, and assists.

We aligned the real-time API Event with video timestamps with the following equation:

$$\begin{aligned}\text{AlignedEvent} = \text{VideoTimeStamp} - \text{GameTimeStamp} \\ + \text{APIEventTime}\end{aligned}$$

This formula facilitated the synchronization of event data from the API with the corresponding moments in the video. The event and relative timestamp to the video are used for further processing. Due to API query limitations, it takes 2-4 minutes to generate all API calls for a 14 hour VOD.

3.5. Sentiment Analysis Pipeline

We utilize GPT to provide a score and determine the general emotion of the content. The process varies slightly between chat messages and audio transcriptions but follows a similar approach. For chat sentiment analysis, we aggregate chat messages into 1-minute intervals. Each interval's messages are sent to GPT with a prompt informing it that the content is from a Twitch chat. We ask GPT to rate the overall sentiment from 1 to 10 and identify the general emotion

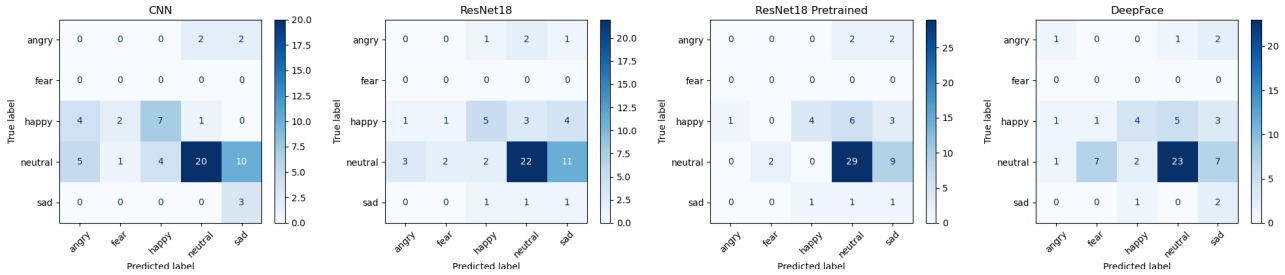

Figure 3. Confusion Matrices When Evaluated on MiniFace

from the following categories: "angry," "disgust," "fear," "happy," "sad," "surprise," and "neutral." The same happens to 1 the audio analysis, except we state in the prompt that a streamer playing X game is speaking.

3.6. Combination and Scoring

We combine all scores at the end of the pipeline. The Sentiment pipeline reveals scores for Chat and Audio transcription which is used directly. We count the API events and give each event type a weight. We also value non-Neutral emotions and weigh Happy and Surprised higher. We tuned the by generating highlights, evaluating them ourselves, and then tuning the weights again. All streamers used the same scoring mechanism. We plan to train a network for this regression task, however, due to data constraints and tight timeline, we decide to use a manually tuned scoring system for now. This resulting in the following formula:

$$\begin{aligned}\text{Score} = {} & 0.3 \cdot \text{Chat Score} + 0.3 \cdot \text{Audio Score} \\ & + 0.6 \cdot count(\text{API} = \text{kill}) + 0.3 \cdot count(\text{API} = \text{other}) \\ & + 0.05 \cdot \mathbb{1}[\text{FER} = \text{happy}] + 0.1 \cdot \mathbb{1}[\text{FER} = \text{surprised}] \\ & + 0.1 \cdot \mathbb{1}[\text{FER} \neq \text{neutral}]\end{aligned}$$

where Chat Score, Audio Score $\in [0, 10]$ are the scores we retrieve from GPT, $count$ counts the the number of specified API events within the time window, and FER is the most likely emotion returned by the FER pipeline.

3.7. Clip Display and Rating Collection

To streamline the process of viewing and rating clips, we developed a website where all generated clips are displayed for evaluation. This website serves as a user-friendly interface for streamers and editors to assess the quality of the highlights and provide feedback.

Users can watch each clip and rate its quality on a scale from 1 to 10. These ratings are saved in our SupaBase database which we'd like to use to train a scoring model.

4. Results and Evaluation

To evaluate our system, we perform quantitative and qualitative evaluations on clip quality, utility provided to streamer, and component wise analysis to ensure each component is contributing valid scores to our final score.

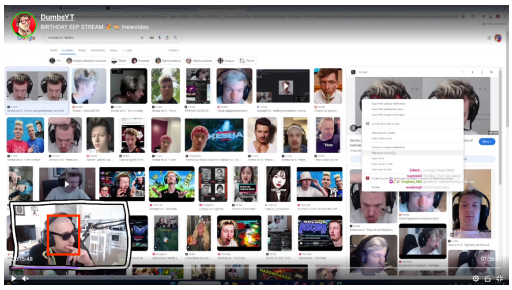

Figure 4. A challenging example of face recognition

4.1. Evaluation of the FER Pipeline

As our system's output directly depends on the quality of each of the components of our system, we need to ensure that each of our pipelines is outputting results fast and accurately.

We evaluated the components of our facial expression recognition pipeline on their speed, robustness, and accuracy.

4.1.1 Speed

We evaluated the speed and accuracy of different face detectors provided by DeepFace. We randomly picked 4 screenshots from different streams and passed them into the face detectors and checked the correctness of the output. We discovered that YOLOV8 was able to detect the faces correctly much faster than the other models (Table 3).

Method	Time (s)	Accuracy
opencv	1.499	1/4
mtcnn	3.435	4/4
fastmtcnn	2.911	4/4
retinaface	3.268	4/4
yolov8	0.964	4/4

Table 1. Performance comparison of different methods

Transcription	Without Context	W/ Context	Explanation
Like way too much shuntu now. Oh she flashes on me too.Wow everyone's flashing on me. I could have lived there.	6: Excited	8: Excited	Enemy team use important spell "Flash" to ambush streamer
key to We win 4v5. Okay, whatever. We won by the way. Like 4v5, we win. Like, I don't know why we surrendered.	8: Angry	6: Confused	Team surrenders when streamer believes they could win

Table 2. Prompting w/ and w/o context

Transcription	1-5	1-10	Explanation
Bye bye. Woo! Woo! Ah! Jin is broken! So now I can just carry my teammates. No matter how bad they play. I'm gonna honor Lee.	5: Excited	10: Excited	Streamer wins a fight against entire enemy team
Dude, Jinn is so fucking broken. Holy fuck, I almost killed Grace, man. I didn't have a head ghost and just like...spent my golds. That's an easy triple kill.	5: Excited	8: Excited	Streamer almost wins a fight against 3 players

Table 3. Prompting scales 1-5 and 1-10

Even though the sample size was small, we believed that YoloV8 demonstrated enough advantages over the other models, and thus we decided to use YoloV8 as our face detector.

4.1.2 Robustness

The robustness of the face detector is also important to the success of the pipeline. As streamers can display anything on stream, we need to be able to differentiate the streamer's face from others' face.

Figure 4 shows an example of running our model on frames with multiple faces. As indicated by the red box near the bottom left corner of the image, our pipeline was able to identify the correct face.

4.1.3 Accuracy

We ran evaluations of our self-trained CNN, ResNet18, pretrained and finetuned ResNet18, and DeepFace provided facial expression classifier on the FER2013 dataset. While the hyperparameters are still not perfectly tuned, we can see that after fine tuning, the pre-trained ResNet18 can reach a validation accuracy of 63%.

In addition, we constructed a mini dataset, **MiniFace**, consisting of 61 stream screenshots with streamer expressions labeled manually by us. We then evaluated our FER pipeline on this mini dataset. ResNet18 pretrained on ImageNet [1] and finetuned on FER2013 performed the best, getting a 55% accuracy on the dataset. CNN and DeepFace achieved a 49% accuracy, while ResNet18 trained from scratch got only 45% of the labels correct.

Figure 3 shows the confusion matrices of each of these models when evaluated on MiniFace. From the matrices, we see that all models tend to mix up neutral and sad faces. While features like smiles and frowns are big enough to be retained, the differences between a sad face and a neutral face can be more nuanced and small in scale, causing the information to be lost when we downscaled the images before passing into the neural networks. Currently, this issue does not affect the model output as much, as sadness is one of the less common emotions expressed by streamers. We therefore decided to recategorize sad faces to neutral faces as a workaround. In the future, to mitigate this issue, we could train a larger network that expects higher definition inputs on larger facial expression datasets. With less information lost in the process, we expect the classifiers to be better at differentiating sad and neutral faces.

4.2. Sentiment Analysis Results

We prompted GPT in different ways, with and without game and platform context. The prompt without context was: *This is a streamer's audio transcription. rate it from 1-10. 1 is bored and 10 is very engaged.*

The prompt with context adds the sentence: *This is a **Twitch** streamer's audio transcription. He is playing **League of Legends***.For the first transcription, the prompt with context yields a better result. Flash is the most important spell in the game with a 2 minute cool down. Using multiple flashes from many enemy teammates is an important clip.

Team surrendering occurs very often (After 9 minutes, 70 percent of players vote to surrender). Thus, rating the 2nd transcription an 8 may overboard. Though you may

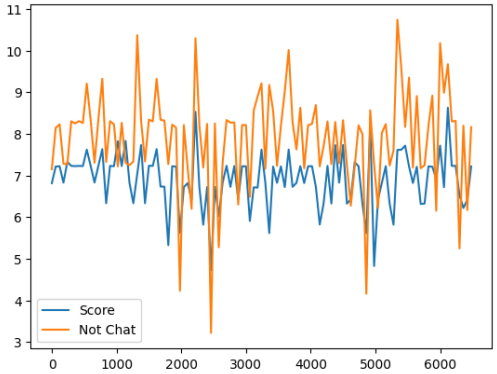

Figure 5. Contribution of chat data to final score

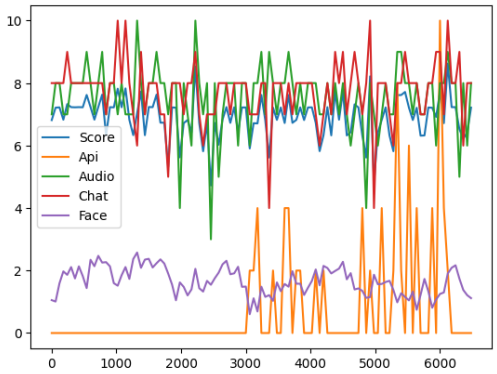

Figure 6. Overall contribution of each type of data to the final Score

argue it should be an 8 since the streamer disagrees with the surrender. We realized without the context of "Twitch" and "League of Legends", GPT would miss out on important contexts such as what "Flash" referred to, and certain internet slang used in chat rooms.

We also prompted GPT on different scales: from 1-5 and 1-10. When given a larger range, GPT is able to give more nuanced scores. Both of the above transcriptions were rated 5 on the 1-5 index. However, when rated from 1-10, GPT was able to differentiate between a big win and a slightly smaller win.

Thus for best performance, we prompted GPT to rate scores from 1-10 and provided it the game the streamer is streaming, and that they are using Twitch.

4.3. Scoring Results

We highlight the top 10 clips from every game. Most highlighted clips are above a 7 in score. The reason why the graphs have so many fluctions is because one minute could change very quickly in terms of highlight ability. This is part of why we are builting the system, and as stated by streamer T, this is what makes or break a good editor.

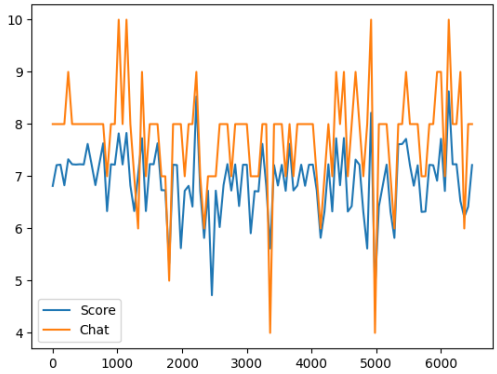

Figure 7. Contribution of non-chat data to final score

Due to the API, the clip can have a weight higher than 10 since we accumulate the number of events in a time interval, so the score ranges from 0 to ∞ with the maximum score encountered as a 13. In figure 5 and 7, you can see the contribution of non chat data vs chat data. They have about equal contribution which makes sense since audio is weighted 0.3 and APIs have sporadic contribution. From figure 6, you can see that the green audio and red chat graph share similar peaks sometimes, but also have various times where they have their own peaks signifying higher contribution from just audio or just chat. The API also offers random contributions, however there are certain moments such as at time 14000 where only chat and audio are responsible for the peak and not APIs. From the data, we do think that Facial data contribution is consistently very low and relatively flat compared to APIs. Instead of accumulating all emotions to adjust the score, an alternative approach may be to try to use emotions to categorize highlights by emotion, ie. Angry vs Surprised vs Neutral highlights.

4.4. Quantitative Clip Quality Results

The feedback on the quality of our generated clips was varied. For the sake of privacy, the 5 streamers who gave feedback will be referred to as N, M, T, S, E. 3 out of 5 streamers did not to provide feedback by filling out the rating forms (M,T,S), indicating a potential lack of interest or time constraints, it is notable that none of the respondents offered concrete details regarding the time taken to view the clips or identify superior alternative clips.

High Follower Streamers (T and M): The two streamers with the highest follower counts (2 million and 400k followers each) were very satisfied with our clips. They appreciated how our system effectively highlighted the exciting moments of the game, aligning well with the type of content they aimed to share with their large audiences.

We analyzed the difference between the streamers' ratings and our system's ratings for the clips. Here are some key insights: Streamer M gave our overall highlight quality

a 9/10. Streamer T filled out our form with the following ratings:

Our Score	Streamer T's Rating
6	6
6	5
5	8
5	5
5	7.5
4	5
4	5.5
4	7
4	5
4	4

Streamer T gave an overall rating of highlight quality a 9/10 as well, and the difference in their rating vs our rating as 1.3 per clip, with a minimum difference of 0 and a maximum difference of 3.

High-Ranking Streamers (S and N): Conversely, the two streamers, with the highest rankings (both top 200 players in North America, with 100k and 200k followers each) were less satisfied. Streamer S mentioned that while the clips were exciting, they did not match the educational content he typically produced, which focused on in-depth gameplay strategies. He insisted that there was a very important clip that we missed out on, where he played a strategy for a full minute, and it resulting in a single kill. This could be an issue with our interval system not taking into account clip contexts longer than a minute. Also our scoring system may place an overemphasis on multiple kills and chat.

Streamer N felt that our system overemphasized moments where he "got loud," suggesting that the combination of API and chat excitement indicators was too sensitive. Interestingly, we did not factor in how loud the audio was, only the transcribed audio and it's sentiment.

Streamer S gave our clips a 3/10 on overall clip quality, whereas Streamer N gave a 5/10 with the following ratings:

Our Score	Streamer N's Rating
10	10
8	0
8	2
7	2
7	5
6.5	3
6	0
6	0
6	7
6	5

This table shows an average difference of 3.25 with a minimum difference of 0 and a maximum difference of 8.

This feedback indicates that our system might have overvalued certain moments due to the convergence of multiple excitement signals, especially playing too much value on APIs and Chat reactions. In fact, it's likely that the 0.3 weight on chat and 0.3 weight on the APIs were too high, especially for nuanced educational content. A potential fix for this would be adding a parameter in the sentiment pipeline called "educational", where GPT can rate the educational content. We could make a separate score for this for streamers that want to focus on different types of content.

These mixed results reveal important areas for improvement. Different streamers have different content preferences, which suggests a need for more customizable and nuanced models. To better cater to diverse needs, we are considering developing separate models tailored to different types of content. This customization would enhance the relevance and quality of the highlights generated for each individual streamer.

4.5. Qualitative Clip Feedback

The most notable success was from the 14-hour VOD (streamer M), where 9 out of 10 clips were used in a highlight video by a streamer with 2 million followers. The only misstep occurred when GPT mistakenly identified the chat spamming "hello" at the start of the stream as a highly exciting moment. This feedback underscores the potential of our system but also highlights areas for improvement.

The most popular complaint from streamers was the difficulty in viewing the clips. Initially, we provided them with a CSV file containing timestamps of their entire stream, ranked by score. However, they found it cumbersome to copy and paste the timestamps into their browser to view the clips.

In response to this feedback, we developed hilait.com, a platform that automatically displays clips along with their timestamps. This website not only simplifies the viewing process but also asks streamers to rate each clip. These ratings are recorded for future score training purposes.

The most significant quality complaint came from Streamers N and S, who noted that our system failed to capture nuanced conversations and educational content. While our system excels at identifying "hype" clips, it struggles with subtler, more meaningful moments.

To address this, we propose increasing the weight of audio transcription and expanding the range of parameters beyond basic emotions like "happy" or "sad" to include categories like "educational" and "sentimental." Additionally, enhancing the role of facial expression analysis could be crucial, as nuanced conversations are often reflected in facial cues.

Our goal is to make these adjustments tunable, allowing streamers to customize the style of highlights they prefer. This flexibility would enable our system to better meet the

diverse needs of different streamers, from those who want to capture thrilling gameplay moments to those who focus on in-depth educational content.

4.6. Time Saved Metric

Streamers did not give us a concrete time as to how much time they saved. However, they did state that their expectation from their editors that they watch their VODs from start to finish to find the best clips. The consensus from all Streamers was that it made the process much faster. In order to gauge how much time they actually saved, we checked the google timeline feature and checked how long it took for them to fill out our ratings form. Since we only got ratings from streamers T and N, here are the results:

Streamer	VOD Length	Time spent with tool
T	8 hours	28 minutes
N	2.5 hours	53 minutes

Keep in mind, Streamer N did not know that you could copy and paste timestamps into a link to jump to that time, and complained about copy pasting taking too long.

Furthermore, we received enthusiastic feedback from Streamer E, a friend of Streamer T, who reported that our tool enabled him to create short-form content for the first time. Previously, he had been unable to do so due to the lack of an editor and not having enough time. This highlights the potential of our system to empower smaller streamers by significantly reducing the barriers to content creation. Lastly, Streamer M, N, T, and E were all interested in using our tool again, under the assumption that they would have the website to navigate the suggested clips. This suggests that despite output, the streamers appreciated the concept the tool.

5. Discussion

A philosophical question arising from our system is: How much data are we losing when we solely process video data? Do non-video forms of data provide essential meaning, or do they merely add irrelevant information at a high cost?

A glaring result is that facial recognition does not contribute very highly to the weight of our final score. Despite it being pre-trained and parallelized per video frame, resulting in minimal additional cost, it adds no benefit when it is overshadowed by game APIs and Chat reactions. This is abundantly clear for popular streamers with many chat viewers and playing a game with supported APIs.

However, for smaller streamers with minimal chat interaction, playing indie games with no API support, facial data becomes an invaluable asset. In these contexts, the streamer's facial expressions can convey emotions and reactions that are otherwise would've been expressed with chat and API calls, providing crucial context that would be lost. We'd like to explore this avenue further, especially in horror games where facial recognition may a more crucial role.

We also display importance of audio transcription and sentiment analysis. Despite the comparatively sparse literature on audio highlighting, it remains a pivotal element of our system. Our biggest feedback was that streamers prioritize substantive and educational content conveyed through their verbal discourse, emphasizing the need for precise audio transcription and sentiment analysis to capture and elevate these significant verbal interactions.

6. Conclusion

In conclusion, our Automatic Highlighting System represents a significant advancement in the realm of content creation for streamers. By building an end to end system that harness various forms of AI data, we have developed a tool that streamlines the process of identifying and highlighting compelling moments from video streams.

Through our evaluation and feedback from streamers and editors, we have demonstrated the feasibility of building an end to end system with multiple data inputs. We have also streamlined workflows and significant time savings were consistently reported across a diverse range of streamers, from those with large followings to smaller creators. Our tool has democratized the content creation process, enabling small streamers to compete in the short-form content domain alongside their larger counterparts.

Although our clip quality feedback was a mixed bag, we are hopeful for improvement. Looking ahead, fine-tuning the scoring algorithm and exploring additional data sources could enhance the accuracy and relevance of generated highlights. Additionally, continued collaboration with streamers and content creators will ensure that our system evolves to meet the evolving needs and preferences of its users.

We hope our tool unlocks new possibilities for storytelling and engagement, ultimately enriching the streaming experience for both creators and viewers.

7. Team Responsibilities

7.1. Tony's

- Audio Transcription pipeline
- Facial Detection pipeline
- Deepface/Facial Emotion Recognition pipeline
- Facial dataset collection and benchmarking
- Network training and evaluation
- CNN for facial recognition (not really relevant for this class?)

- Confusion matrices

7.2. Danica's

- Game API pipeline
- Chat Transcription Pipeline
- Sentiment Analysis Pipeline
- Final Combination and Scoring
- Video Pipeline w/ CLI
- ResNet and Pretrained models for facial recognition, image preprocessing (not really relevant for this class?)
- Saliency Maps, Class Visualization
- Website and backend to collect rating data
- Working with streamers to get feedback

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 5

[2] Cheng-Yang Fu, Joon Lee, Mohit Bansal, and Alexander C. Berg. Video highlight prediction using audience chat reactions, 2017. 2

[3] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests, 2013. 2

[4] Ultralytics LLC. Yolov8: Real-time object detection and segmentation, 2024. Accessed: 2024-05-17. 2

[5] Qing Ping and Chaomei Chen. Video highlights detection and summarization with lag-calibration based on concept-emotion mapping of crowd-sourced time-sync comments, 2017. 2

[6] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision, 2022. 3

[7] Sefik Serengil and Alper Ozpinar. A benchmark of facial recognition pipelines and co-usability performances of modules. *Bilisim Teknolojileri Dergisi*, 17(2):95–107, 2024. 2

[8] Huan Yang, Baoyuan Wang, Stephen Lin, David Wipf, Minyi Guo, and Baining Guo. Unsupervised extraction of video highlights via robust recurrent auto-encoders, 2015. 2