HilAIt: Automatic Video Highlighting System Leveraging Audio, Text, Facial, and Semantic AI

Danica Xiong
Stanford University
daxiong@stanford.edu

Tony Xia
Stanford university
tonyx717@stanford.edu

Abstract

We have developed an Automatic Highlighting System that leverages multiple AI technologies to process stream videos and generate highlights. The system takes in various types of input, including video, audio, facial data, chat logs, and game APIs, each processed through separate, parallelized pipelines. By utilizing state-of-the-art Language Models (LLMs), Optical Character Recognition (OCR), and facial classification algorithms, the system evaluates and classifies moments to determine their highlight-worthiness. We evaluate the necessity of multiple data inputs and the feasibility of generating quality clips with multiple data inputs.

1. Introduction

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, the ability to effectively curate and highlight compelling moments from extensive streams has become increasingly valuable for tournament holders, content creators, and fans. However, manual curation of highlights is time-consuming and often subjective, relying heavily on human judgment digging through hundreds of hours of content. The sheer volume of content generated during live streams poses a daunting task for streamers seeking to distill key moments for their audience. Without automated tools to assist in this process, streamers are burdened with the arduous task of sifting through hours of footage, diminishing their productivity and detracting from the overall quality of their content. Furthermore, video content contains a lot of nuances in which traditional AI pipelines miss out on. Excluding video, there is also facial emotion, textual. and audio clues, that we would like to leverage in our pipeline.

Our application automates the generation of start and end timestamps for video highlights, significantly reducing the time content creators need to find quality clips. The primary objective was to build a comprehensive, end-to-end pipeline that produces high-quality video highlights with minimal manual intervention. We aimed to answer key questions regarding the feasibility of generating quality clips using this diverse array of data inputs and whether all these data types are necessary for optimal performance. Additionally, a crucial aspect of our project was ensuring that the fully integrated system operates efficiently within a reasonable timeframe, leading to our parallelized combination framework.

Our key contributions are the following:

1. We devised a parallelized system that seamlessly integrates multiple data modalities, including video, emotion, APIs, audio, and chat sentiment. This holistic approach allows for a comprehensive analysis of stream content, facilitating the identification of highlight-worthy segments.

2. Introducing a trainable scoring system, we empower users to customize and refine highlight selection criteria according to their preferences and objectives, thereby enhancing the adaptability and utility of our system.

3. Through soliciting feedback from streamers, we conducted a thorough evaluation of our system's performance in terms of highlight quality and time-saving efficiency. This feedback-driven iterative process ensured the practical relevance and effectiveness of our solution.

4. Our analysis delves into the necessity and impact of each data type on the overall performance of the system. By elucidating the role and significance of these data modalities, we provide valuable insights into the underlying mechanisms driving highlight generation.

5. We examine 4 architectures trained on the FER2013 dataset and analyse the strengths and weaknesses of each model.

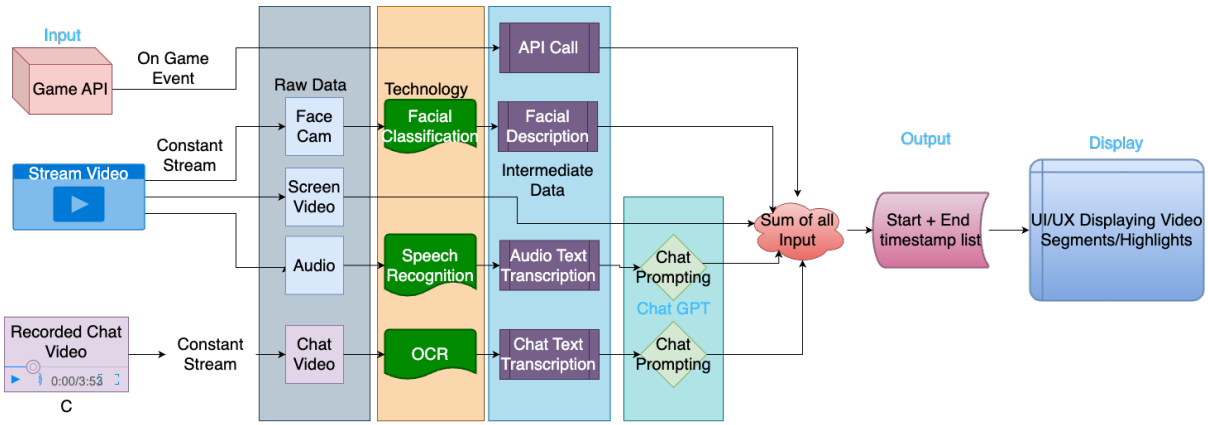

Figure 1. The following is our data processing pipeline. From the video Id, we get 3 inputs: Stream Video, Chat Video, and Game API. Eachis split into their raw data, ie. Face Cam, Screen Video, Audio, Chat video, be processed by their corresponding models, and then used as input to timestamp the video.

2. Related Works

2.1. Facial Emotion Recognition: State of the Art Performance on FER2013

Facial Emotion Recognition (FER) [3] has witnessed significant advancements in recent years, with state-of-the-art performance demonstrated on benchmark datasets such as FER2013. FER2013, a widely used dataset in the field, comprises over 35,000 labeled facial images categorized into seven emotion classes. Researchers have achieved remarkable accuracy in emotion classification tasks on this dataset, leveraging deep learning architectures such as CNNs and RNNs. Integrating FER capabilities into our facial recognition pipeline enables us to capture nuanced emotional responses displayed by streamers during live gameplay sessions.

2.2. Robust Speech Recognition via Large-Scale Weak Supervision

Large-scale weak supervision [10] capitalizes on the abundance of unlabeled speech data available on the internet and other sources, leveraging weak supervision signals such as automatic speech recognition (ASR) outputs, text transcripts, and keyword spotting to train speech recognition models. By harnessing the power of weakly annotated data at scale, researchers aim to alleviate the burden of manual annotation and enhance the robustness and adaptability of speech recognition systems. We use this to transcribe the streamer's speech to text.

2.3. Unsupervised Extraction of Video Highlights Via Robust Recurrent Auto-encoders

In Unsupervised Extraction of Video Highlights Via Robust Recurrent Auto-encoders [13], the authors tackle the challenge of unsupervised highlight extraction by leveraging recurrent auto-encoders, a type of neural network capable of capturing temporal dependencies in sequential data. The authors conduct experiments on diverse video datasets spanning various domains, including sports, entertainment, and gaming. The primary limitation of their paper is the absence of a comprehensive evaluation concerning the quality of their extracted highlights.

2.4. Video Highlight Prediction Using Audience Chat Reactions

This paper [2] takes the opposite approach as the prior one. The authors address the challenge of highlight prediction by harnessing the interaction data present in chat logs during live video streams. They construct a dataset and train V-CNNs on it; however, their study lacks a comprehensive evaluation of their approach similar to the paper above.

2.5. Video Highlight Prediction Using Audience Chat Reactions

The key contribution of Ping et al.'s [9] approach lies in the integration of lag-calibrated time-sync comments with concept-emotion mapping techniques. By aligning comment timestamps with corresponding video timestamps and mapping concept and emotion attributes to each comment, the authors construct a rich dataset for highlight detection and summarization. The authors integrate both video and chat data, employing sentiment analysis—a unique approach not previously observed in the above literature. However, again, the authors do not provide much on evaluation.

3. Methods

3.1. Video and Chat Collection

We aimed to simplify the input process for streamers, requiring only the video ID of a Twitch VOD as input. However, due to Twitch regulations, direct downloading of video or chat data is restricted. To circumvent this limitation, we utilized the TwitchDownloader tool developed by Lewis Pardo (available at https://github.com/lay295/TwitchDownloader). This prebuilt CLI tool facilitates the downloading of VODs in 1-hour segments, which are subsequently concatenated to form the full-length VOD. Additionally, the tool performs OCR on the chat, transcribing messages into a JSON file with corresponding timestamps aligned to the beginning of the video.

3.2. Audio Transcription Pipeline

Automatic Speech Recognition (ASR) transcribes the streamer's speech for subsequent sentiment analysis using GPT. After evaluating various options, we chose to utilize OpenAI's open-source ASR model, Whisper [11].

We first convert the MP4 stream video into an MP3 audio file. A challenge with Whisper is that it does not record timestamps with the transcriptions, making it difficult to align audio transcriptions with other components in downstream tasks. To address this, we segment the audio input into 30-second clips and run the pipeline on each clip independently. This segmentation offers several advantage of parallel processing. The transcription for a 14 hour vod takes 3 hours without parallelization. The transcribed messages in 30 second intervals is fed into the sentiment prompting pipeline below.

3.3. Sentiment Prompting

The transcribed chat and audio data undergo segmentation into variable-length intervals, with our chosen interval duration set to 1 minute. Utilizing OpenAI's GPT API, we prompt the model to provide a rating on a scale from 1 to 10, indicating the level of excitement exhibited by the chat, along with a single emotion chosen from a predefined set of options: ["angry", "disgust", "fear", "happy", "sad", "surprise", "neutral"].

During our experimentation phase, we initially tested with a scale of 1 to 5 and various prompts. However, we observed that this limited scale resulted in an overabundance of maximum ratings (5s) being assigned by the model. To address this issue, we settled on expanding the scale to 1 to 10, providing the model with a broader range for assigning ratings. Additionally, we enhanced the model's contextual understanding by specifying that the chat messages pertained to gameplay within the popular video game 'League of Legends'. This context proved essential for interpreting gamer-specific phrases such as 'poggers', which may convey excitement or enthusiasm synonymous with 'wow'.

For time, it takes 0.5 hours to run sentiment analysis on a 14 hour VOD

3.4. Game API Pipeline

Navigating the intricacies of the game API pipeline posed several challenges, primarily due to the disparity in data alignment between video and real-time API data streams. Due Twitch's lack of transparency for exact video start times, aligning game matches with video timestamps proved particularly complex.

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.

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:

$$AlignedEvent = VideoTimeStamp$$

$$-GameTimeStamp + APIEventTime$$

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. Face Data Pipeline

The face data in our system is detected using DeepFace [1]. This tool provides the center x,y point of the face along with the dimensions of the bounding box h,w. We use this to crop the image accordingly and convert it to grayscale. The cropped grayscale image is then resized to 48x48 pixels to match the FER2013 dataset specifications.

One challenge we encountered was the detection of multiple faces when the streamer watched videos or browsed websites containing faces. This caused our face detection algorithm to recognize and process an excessive number of faces simultaneously. Fortunately, DeepFace includes a feature that allows for the detection of a specific face. To leverage this, at the beginning of each stream, we provide DeepFace with a reference screenshot of the streamer's face, enabling it to focus solely on detecting this particular face throughout the stream.

The video is split into frames and every 20 frames is fed into our model. This step is fully parallelizable and is the bottleneck of our system. It takes 3 hours to finish performing Audio transcription and sentiment analysis on a 14 hour VOD. However, if frame processing were parallelized, this time would be reduced significantly.

3.6. Facial Emotion Models

We built a custom CNN, resnet, and then fine tuned on a resnet made by imagenet trained a resnet v50 but it overfit way too fast. We opted for a ResNet-18 model because it performed slightly better due to it having fewer layers. By incorporating a 0.4-0.6 dropout and increased the weight decay to from 0.001 to 0.1, and it improved the validation accuracy from 0.53 to 0.56. However, the model still overfit too fast.

For the fine-tuned ResNet-50 pre-trained on Image Net, we achieved a validationaccuracy of 0.58 but it also suffered from rapid overfitting.

Our CNN architecture actually performed the best because it wasn't able to overfit so quickly, achieving a validation accuracy of 0.6

3.7. Combination Step

The combination step is designed to compute a new score for each specified interval, which in our case is 1 minute.

For each minute, we analyze the video frames and their corresponding FER emotions. The scoring mechanism for emotions is as follows:

- If the emotion is not neutral, the total score is increased by the confidence of the emotion.
- If the emotion is happy, it is increased by 0.5*confidence resulting in 1.5*confidence contribution from the emotions.
- If it is surprised, it is increased by the confidence, resulting in 2*confidence. The confidence is between 0 and 1 so. All frames within the minute are summed.

API events are recorded per second, and all events within the minute are aggregated to contribute to the score:

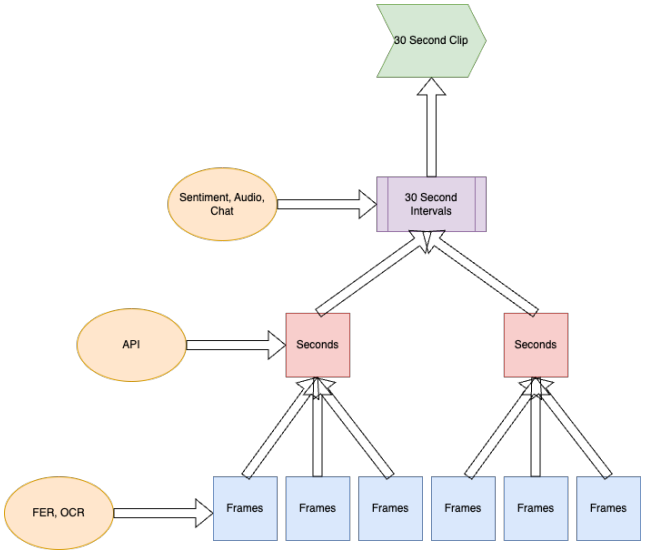

Figure 2. The following is our combination step where the results of all of our components are put into the final score. All components are parallelized and combined at the end

- A death event contributes 0 points.
- A kill event contributes 2 points each.
- All other events contribute 1 point each.

Chat sentiment and audio transcription are already split into intervals previously, and the score in that interval is given by GPT. If the sentiment interval is smaller than the highlight interval, all chat and audio scores within the interval is added to the current score.

This resulting in the following formula:

$$\begin{aligned} Score = 0.3 * chat + 0.3 * audio + 0.3 * 2* \\ \mathbb{1}[API == KILL] + 0.3 * \mathbb{1}[API! = KILL] + \\ 0.5 * 0.1 * \mathbb{1}[FER == Happy] + 0.1* \\ \mathbb{1}[API == Surprised] + 0.1 * \mathbb{1}[API! = Neutral] \end{aligned} \tag{1}$$

We tuned this based off of generating highlights, evaluating them ourselves, and then tuning the weights again. All streamers used the above scoring mechanism. Obviously we want to learn the score later on, however due to data constraints, we settled on this.

4. Dataset and Features

4.1. FER Dataset

We utilized the FER 2013 [3] dataset for training and evaluation. This dataset comprises 28,709 training examples and 3,589 test examples, each represented as a 48x48 grayscale image.

One significant issue with the FER 2013 dataset is its lack of diversity. The dataset predominantly features adult male and female faces of Caucasian descent, which results in suboptimal performance when recognizing emotions in children, elderly individuals, and non-Caucasian faces. This limitation highlights the need for more diverse datasets to improve the generalizability of facial emotion recognition systems.

We considered other datasets, but they had their own limitations. For instance, the JAFFE [4] dataset contains only 213 images of Japanese women and is not publicly accessible for use. The CKPlus [7] dataset only includes 981 images. On the other hand, the CelebA [6] dataset offers a substantial 202,599 images and Affectnet [8] had 0.4 million images, but were not publicly available, and PyTorch didn't have a pre-trained model for them.

Ultimately, we chose the FER 2013 dataset despite its limitations. To enhance the robustness of our model, we preprocessed the images by introducing occlusions and bars on the sides, simulating the cropping that occurs in video streams and increasing our training data size.

4.2. Video and Audio Dataset

With the consent of five streamers, we were able to use their stream VODs as our video data. Each streamer provided one VOD, with durations ranging from a minimum of 2.5 hours to a maximum of 14.2 hours. In addition to the video data, we saved the corresponding chat data for all five videos in JSON format. The audio was extracted from the video files and saved as MP3 files for separate processing. For the video data, the MP4 files were read in as frames at specified intervals, typically every 10 or 20 frames. We didn't process every single frame as it would take too much compute power and time.

5. Results

We evaluated our results using both quantitative and qualitative metrics. To align with our goal of reducing the workload for streamers and their editors, we measured the amount of time saved in the editing process. Additionally, we asked streamers and their editors to rate the generated clips on a scale from 1 to 10, and compared these ratings with our system-generated scores to assess likeness.

To ensure the robustness of our system, we performed a component-wise analysis, verifying the functionality and performance of each individual component.

Lastly, we gathered qualitative feedback from the streamers and their editors to gain insights into the usability and practical effectiveness of our system.

The results of our evaluation are as follows:

Method	Time (s)	Correctness
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 face detection methods

5.1. Component Analysis

For our facial expression classification tasks, we rely on DeepFace's prebuilt face detector to accurately identify the streamer's face. [12]. These methods include traditional facial detectors, RetinaFace, YOLOv8, and MTCNN.

We evaluated these face detection methods on four random screenshots of Twitch streams. Our observations indicate that YOLOv8 significantly outperforms the other methods in terms of speed while maintaining high accuracy. Consequently, YOLOv8 emerges as our preferred choice for face detection in live stream scenarios due to its efficiency and reliability.

5.1.1 Facial Emotion Classification

The FER2013 dataset comprises of seven emotions: ["angry", "disgust", "fear", "happy", "sad", "surprise", "neutral"]. Our initial approach to classifying images into these categories involved using a Convolutional Neural Network (CNN). Given that FER2013's data is formatted as 48x48 grayscale images, the input to our network had a single channel of size 48x48. We preprocess our frames to match this format.

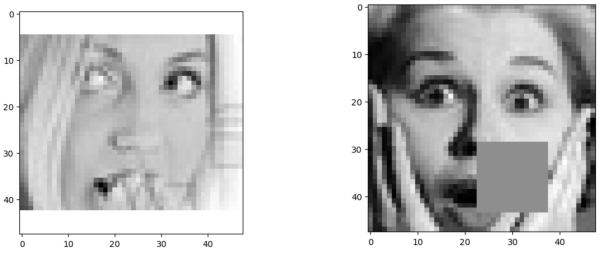

Figure 3. Image preprocessing performed

5.1.2 Face Detector

The architecture of our CNN included the following layers: Conv2D, Batch Normalization, ReLU, MaxPool, Conv2D, Batch Norm, ReLU, MaxPool. The fully connected layer was Dropout(0.5), Linear, Batch Norm, Dropout, ReLU, Linear, Batch Norm, Dropout (0.5), ReLU, Linear. This fully connected layer was used in our resnet and pretrained

resnet as well. We realized we needed a lot of regularization, hence the dropout. Our ResNet18 architecture consists of 18 convolutional layers. Each convolutional layer is followed by a batch normalization layer and a downsample. Finally, our ResNet18 on pretrained ImageNet just overrode the fully connected layer in the pretrained model with the linear layer specified above.

Upon testing, we observed that ResNet50 and its pre-trained variant overfit the data rapidly due to their higher capacity. Consequently, ResNet18 and the pre-trained ResNet18 emerged as our preferred models for this task, striking a better balance between performance and generalization.

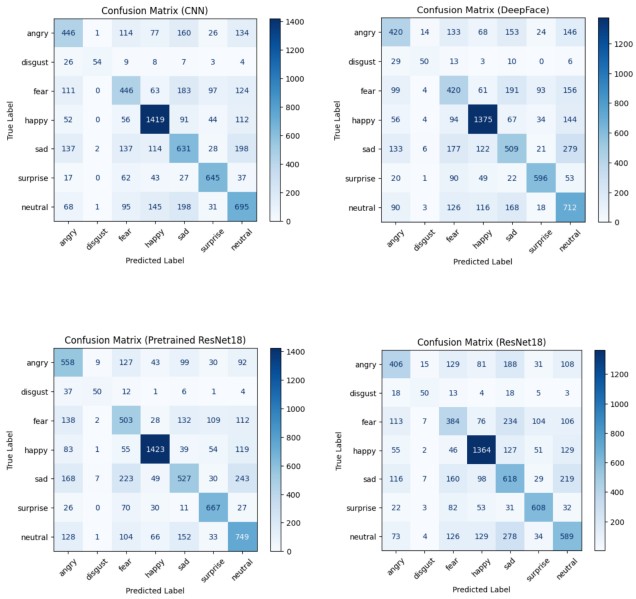

Figure 4. Confusion matrices for all architectures. Many of them misclassified fear and sadness, and happiness and neutral. This makes sense because these emotions are similar.

The final accuracy's compared to SOTA [5] architectures are as follows: Keep in mind, many of these models used much larger datasets to train on, not just FER2013's training set. It makes sense that ResNet performs better with larger datasets compared to CNNs.

Model	Accuracy(percent)
Our CNN	61
Our ResNet18	58
Our Pretrained ResNet18	63.3
Human	65
CNN	62.2
GoogleNet	65.2
VGG+SVM	66.40
Attention CNN	70.02
ARM(ResNet)	71.38
VGG	73.28

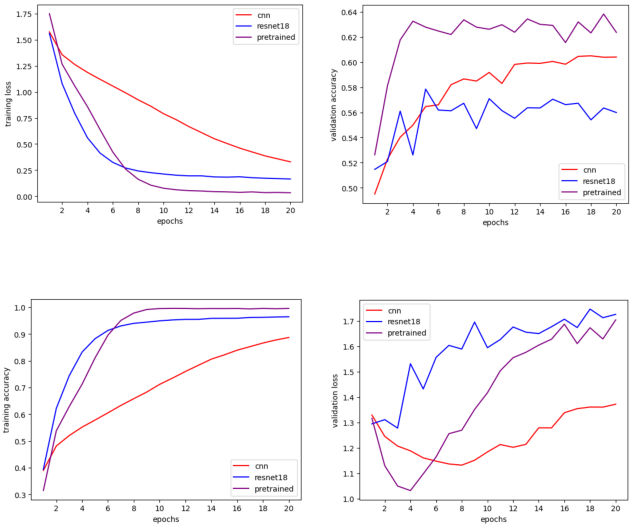

Figure 5. Loss graphs confirm that our ResNets are overfitting.

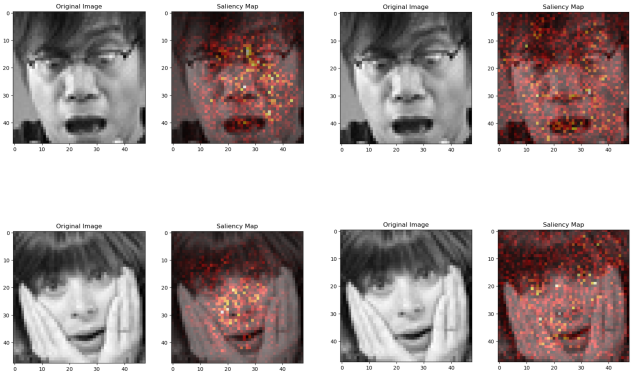

Figure 6. Saliency Maps for pretrained ResNet(left), CNN (right). ResNet shows more interest around the Eyes and Mouth

5.2. Clip Quality Feedback

The feedback on the quality of our generated clips was varied. The most notable success was from the 14-hour VOD, 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.

High Follower Streamers: 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: One streamer gave our overall highlight quality a 9/10. The other filled out our form with the following ratings:

Our Score	Streamer Rating
6	6
6	5
5	8
5	5
5	7.5
4	5
4	5.5
4	7
4	5
4	4

This streamer 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: Conversely, the two streamers with the highest rankings (both top 200 players in North America, with 100k and 200k followers each) were less satisfied. One 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 it may place an overemphasis on multiple kills and chat.

The other streamer 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.

One streamer gave our clips a 3/10 on overall clip quality, whereas the other one gave a 5/10 with the following ratings:

Our Score	Streamer 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.

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 should consider 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.

5.3. Time Reduction Feedback

To calculate the amount of time saved in finding clips, we initially asked streamers and editors to estimate their usual time spent on this task and the time saved using our system. While we hoped for specific numbers, the consistent feedback we received from all five streamers was that the system "saved a lot of time." but did not follow up on exactly how much time they spent.

To obtain better quantitative metrics, we used google doc's history to check how long it took for streamers to rate every generated clip. Out of the two streamers who rated all 10 clips:

One streamer took 53 minutes to rate clips from a 2.5-hour VOD. Another streamer took 30 minutes to rate clips from a set of 8-hour VODs. The primary complaint was that it took too long to copy and paste the timestamps from the CSV we generated into the Twitch player. In response, we developed a website to automatically play these clips, further streamlining the process and addressing the feedback effectively.

The overall time reduction was from variable hour vods (2.5-14 hours) to 0.5-1 hour, with most of the time used on pasting timestamps into the video player. Overall, the feedback highlights the significant time savings our system provides, while also pointing out areas for further improvement to enhance the user experience.

5.4. Emergent Results

One of the most surprising and exciting was that our system was its ability to empower small streamers to enter the Short Form Content domain. Many small streamers, who lack the resources to hire editors and the time to sift through hours of their own VOD data, found our clip highlighting system to be a game-changer. By using our system, these streamers were able to automatically generate engaging highlights from their streams and upload them to

platforms like TikTok and YouTube Shorts.

The feedback from one small streamer (100k followers) has been overwhelmingly positive. He reported that our system not only saved them a significant amount of time but he was excited to use it to help him grow his audience by consistently producing high-quality short-form content. For many, this meant the difference between sporadic content updates and a steady stream of engaging clips, which in turn boosted their visibility and engagement on social media.

6. 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?

Our analysis suggests that integrating multiple data sources enhances the richness and accuracy of the generated highlights. For instance, although facial recognition can be pre-trained and parallelized per video frame with minimal additional cost, it also brings minimal benefit to sentiment analysis, especially when APIs and Chat overshadow it. For large streamers with bustling chat activity and comprehensive API data, this is the case. The reactions and events are easily captured in our system and often leads to the exciting clips that people often expect for video highlighting.

However, for smaller streamers with minimal chat interaction and games with no API support (such as those playing indie games), 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 also stand by the importance of audio transcription and sentiment analysis. While the literature on audio transcription is less extensive compared to video, it remains a critical component of our system. Many streamers expressed a preference for meaningful and educational content conveyed through their own speech rather than merely capturing exciting moments and reactions. This underscores the need for accurate audio transcription and sentiment analysis to capture and highlight these significant verbal interactions.

7. Future Works

7.1. Educational Content Detection

We plan to modify the scoring algorithm and prompting techniques to better capture educational content desired by certain streamers. By increasing the weight of audio transcription and enabling GPT to classify whether the content is educational, we will try to cater to streamers who focus on in-depth explanations and strategies.

7.2. Learning the Scoring Function

Instead of relying on manual tuning, we eventually want to train a model to learn the optimal scoring function. This would give us a learned contribution of each datapoint (video, audio, chat, etc.) to the overall score which would give us insights into what data is actually important. We are currently collecting ratings through our website and plan to outsource clip rating to friends and other gamers online to gather more data for this purpose. We are very excited about this if we are able to collect enough data.

7.3. Generating Variable Length Clips

We intend to support the generation of variable length clips, maintaining continuity for clips longer than one minute.

7.4. Improving Game API Integration

Currently, we manually transcribe the characters in each game and the streamer's username. This cant be done with OCR because usernames are hidden and character names are blocked by streamer overlays. This process can be streamlined by developing a C++ application that accesses game memory in real-time. Since these APIs are not available remotely, local access on the game client is necessary.

8. Conclusion

In this project, we developed an Automatic Highlighting System that leverages multiple AI technologies to process various inputs, including video, audio, facial expressions, chat data, and game API events. Each input follows a distinct, parallelized pipeline, using AI tools like LLMs, OCR, and custom facial classification to identify highlight-worthy moments in stream videos. Our system effectively generates start and end timestamps for these highlights, aiming to reduce the workload for content creators by automating the clip selection process.

Our project highlights the potential of AI-driven tools in transforming content creation workflows, making it accessible for streamers of all sizes to produce high-quality, engaging content with minimal manual effort. We hope future works with this project will help us understand what people find "interesting" and "entertaining" across different types of data. We are excited about the future possibilities and the continuous improvement of our system to better serve the streaming community.

9. Contributions & Acknowledgements

We'd like to thank all authors mentioned in this paper for their contributions to the field of AI, vision, and systems.

We'd like to thank Chengsu Deng, our mentor in CS231N for his ideas on video alignment and advice on Visual NN architectures and tuning. We'd like to thank Kayvon Fatahalian, our CS348K professor for his emphasis and wisdom on strong evaluation metrics, goal setting, and constraint identification for systems.
Note: Code contains API keys, file uploaded instead of GitHub.

9.1. Danica's contributions

- Built pipeline for API. Used Riot's API documentation to perform data collection, data aggregation and alignment with video.
- Built pipeline for Chat collection and sentiment analysis. Used prebuilt OCR tool (Twitch Downloader https://github.com/lay295/TwitchDownloader) to read Twitch chat and transcribe text into CSV files. Processed data for sentiment analysis.
- Built sentiment analysis portion of project. Sent chat data and transcription data to GPT and prompted it for score.
- Built and tuned custom ResNet50 ResNet18 (not torch's ResNet50) architecture for facial recognition.
- Built and tuned pretrained ResNet50 ResNet18 on ImageNet for facial recognition.
- Preprocessed image data to have occlusions and different crops.
- Built website frontend & backend to crowdsource ranking data. Backend data stored in SupaBase.
- Built combination pipeline, taking in facial data, chat sentiment, audio transcription sentiment, face analysis, and generating highlighted clip timestamps.
- Communicated with streamers and editors to gather their evaluations and feedback.
- Generated loss/accuracy graphs
- Generated Saliency Maps and Network Visualizations
- Debugged each others code!

9.2. Tony's contributions

- Built Audio transcription pipeline utilizing Whisper, tuned Whisper to better process our video data
- Built Facial Detection pipeline utilizing Deep Face. Finetuned facial data to recognize streamer's face.
- Built and tuned CNN architecture for facial recognition
- Built data loader for Facial Recognition
- Generated loss/accuracy graphs
- Generated confusion matrix
- Labeled 200 datapoints on streamer data
- Debugged each others code!

9.3. Specifications

The data pipeline for facial classification and facial classification was written for CS231N. This includes the CNN, ResNet, pretrained ResNet on ImageNet, Saliency Maps, Network Visualizations, Data Loading, Video to frame pipeline, Data Preprocessing, Loss Curves, Deep Face used for face location, and Deep Face vs ResNet vs CNN analysis of the FER2013 dataset.

References

[1] H. Cate, F. Dalvi, and Z. Hussain. Deepface: Face generation using deep learning. *CoRR*, abs/1701.01876, 2017.

[2] C.-Y. Fu, J. Lee, M. Bansal, and A. C. Berg. Video highlight prediction using audience chat reactions, 2017.

[3] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, Z. Chuang, and Y. Bengio. Challenges in representation learning: A report on three machine learning contests, 2013.

[4] M. Kamachi, M. Lyons, and J. Gyoba. The japanese female facial expression (jaffe) database. *Availble: http://www. kasrl. org/jaffe. html*, 01 1997.

[5] Y. Khaireddin and Z. Chen. Facial emotion recognition: State of the art performance on fer2013, 2021.

[6] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[7] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 94–101, 2010.

[8] A. Mollahosseini, B. Hasani, and M. H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1):18–31, Jan. 2019.

[9] Q. Ping and C. Chen. Video highlights detection and summarization with lag-calibration based on concept-emotion mapping of crowdsourced time-sync comments. In L. Wang, J. C. K. Cheung, G. Carenini, and F. Liu, editors, *Proceedings of the Workshop on New Frontiers in Summarization*, pages 1–11, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[10] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[11] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision, 2022.

[12] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[13] H. Yang, B. Wang, S. Lin, D. Wipf, M. Guo, and B. Guo. Unsupervised extraction of video highlights via robust recurrent auto-encoders, 2015.

10. Supplemental Pictures

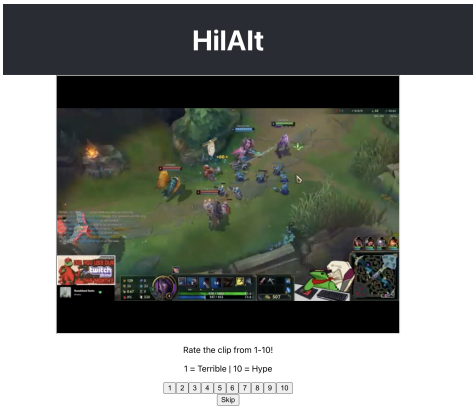

Figure 9. Hilait.com! Coming soon, our automatic clip rating (data collecting) website!

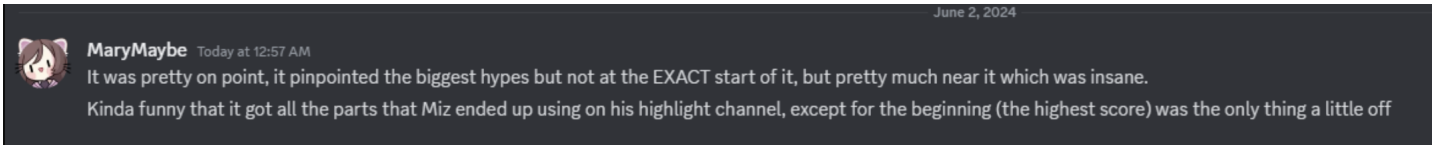

Figure 7. Exciting Streamer feedback. Highlights used for Mizkif, streamer with 2 million followers

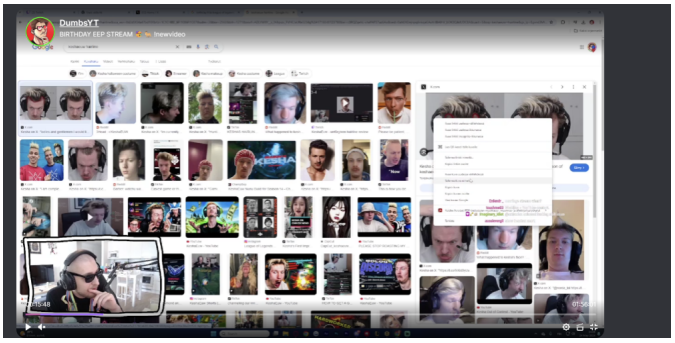

Figure 8. Streamer opens ”hairline” google search and bugs out our facial detection (fixed by specifying face)

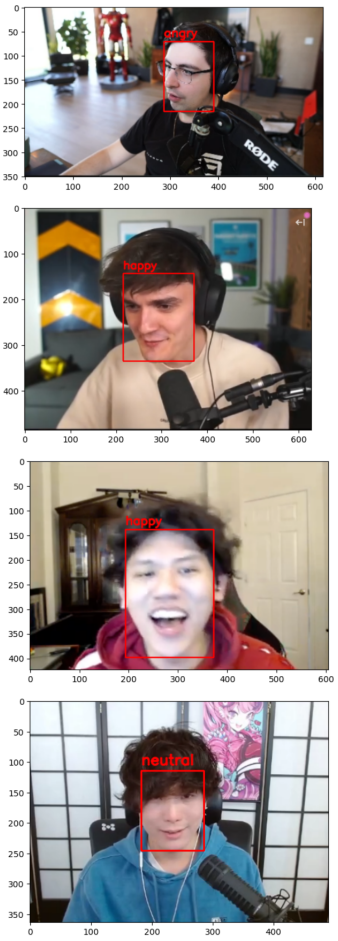

Figure 10. FER results

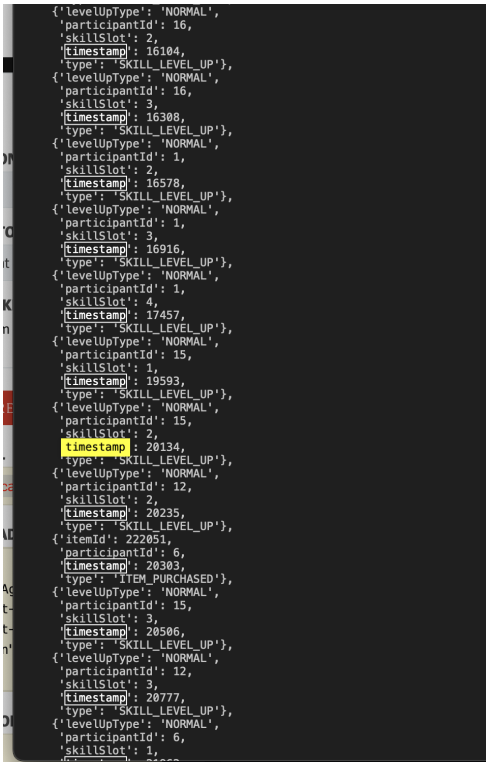

Figure 11. API Data with relative timestamp per match

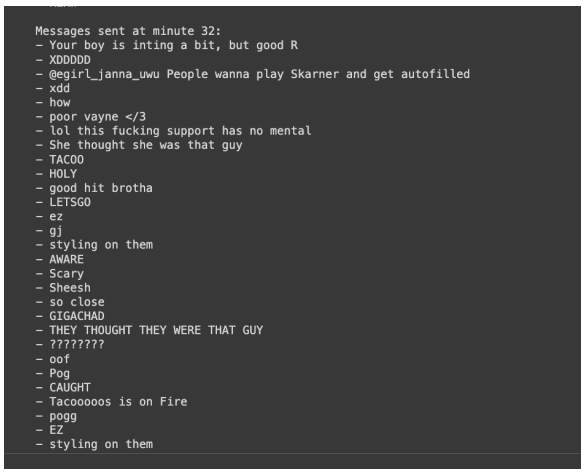

Figure 12. Transcribed Chat at minute 32

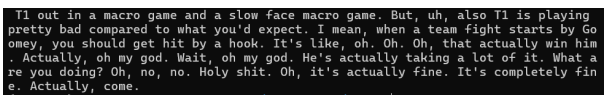

Figure 13. Transcribed audio data in 10 seconds