Multi-Modal Classroom Video Summary Generator

Hari Raagav T R[1], Lakshmi Narayan P[1], Aryan Kumar[1], Geethika Kommineni[1] and Dr. Sandesh B J[1]

[1]Department of Computer Science, PES University, Bangalore, Karnataka, India.

Contributing authors: hariraagavtr@gmail.com; lakshminarayanpaidisetty@gmail.com; aryan.kumar.052001@gmail.com; kommineni.geethika@gmail.com; sandesh_bj@pes.edu;

Abstract

Video recordings of classroom lectures are widely used in higher education as an essential learning tool. However, students frequently have difficulty of having to skim through the entire video in order to discover the pertinent content when they need to rapidly recall a certain idea or lecture. In this article, we suggest a novel approach to this problem by automatically creating a video summary of the recorded lecture in the classroom. Our approach involves using the audio from the lecture video to identify time intervals corresponding to specific topics, as well as generating time intervals from the video footage of the classroom. These time intervals are then combined to create a summary video that covers all the main subtopics covered in the classroom lecture. We evaluate our approach on a dataset of recorded classroom lectures and show that it is able to accurately generate informative and concise video summaries. By giving students the ability to quickly access the information they need without having to view the entire video, our suggested method has the potential to significantly increase the efficiency and efficacy of learning from recorded classroom lectures.

Keywords: Automatic Speech Recognition, Speech Summarization, Computer Vision, Image Quantization, Object Detection, Mediapipe, Timestamps

1 Introduction

Classroom lecture videos are gaining immense popularity as both - a primary resource for distant learning and as a supportive resource for in-person lectures. The increased usage of classroom lecture videos demands for a tool which can aid students understand lectures more quickly and effectively. Most lectures are usually an hour long which makes it difficult for a student to re-watch it. Also, staying focused throughout is often considered as a challenge. This leads to a situation where one could miss out on the important details. When preparing for an exam, it is a monumental task for a student to cover all the topics and comprehend the plethora of information in the multiple hour long lecture videos. Research [7] has found that shorter videos are engaging for students. Further, studies [8] found students expressed that the summary videos made helped them review the subject before examination. Summary videos also are a great approach to help put the information studied into context.

It is a difficult task to fully automate the video's information extraction and summarization process to produce a shorter format video content that is easy to digest. The challenge of information extraction from lecture videos [1] is a difficult task as a single fixed camera is positioned over the whiteboard and is frequently used in lectures that are self-recorded. It does not have extensive production and transcript annotation. As a result, a pipeline is required to summarise lectures using visual text rather than a text transcript.

To solve this, the solution suggested in this paper would assist the students by providing them with a summary of the complete classroom lecture video. Generating a video summary using the solution proposed enhances both traditional lecture-based education and online learning as teachers or professor's can refer to the previous lecture video summaries on similar topics and prepare to take a class. The multi-modal classroom lecture video summary generator produces a summary using audio as one of the inputs and the video as the other. These inputs are then processed to generate the key timestamps from the audio pipeline and the video pipeline. Initially a classroom lecture video is passed to the model which is processed by the audio pipeline and the video pipeline.The audio pipeline performs pre-processing on the video by first converting it to mp3 format and then obtaining the transcript using AssemblyAI speech to text model which cleans the audio and provides the transcript of the lecture video.

To generate highlights the AssemblyAI audio summarization tool is used, which returns a JSON format file with the subtopics and the time intervals of the particular subtopics. After obtaining the time intervals of the particular subtopics, these subtopics are stored and returned to a merger module. To generate the video timestamps of the key moments, we first define what these key moments would be. For example, as explained in [9], in football, a key moment might be when a player gets red card or when a goal is scored, similarly the key moment we take in the video pipeline is when the professor is writing on the board, and to filter out all the moments the professor is writing on the board, we also detect the percentage of changes in the board and set a

threshold to count that frame in the video as a key moment. After we obtain the time intervals from both the audio and the video pipeline we combine the time intervals obtained and return the final classroom video summary. As the subtopic labeling is done in the audio pipeline we also return the subtopics with the timestamps from the audio pipeline.

Results are calculated quantitatively and qualitatively. For quantity measure, the duration of the original classroom lecture and the generated summary video are considered. The results here indicate the ability of the proposed methodology to be able to produce a short summary. For quality measure, metrics such as precision, recall, and F-Score have been calculated. The original classroom lecture video was manually annotated to indicate the presence of important frames and these frames were compared to the frames in the generated summary video to depict the representation of the original lecture video into the summary. A larger F-score indicates that the summary generated has a better representation of the original classroom video lecture.

2 Related Work

2.1 Single-Modal Video Summarization

The papers using one mode for summarizing the videos, for example, [1], [2], [3], and [4] took the video as the input and applied various supervised and unsupervised techniques to obtain the summary by providing key frames of the video and a summary of the video representative of the original video fed as the input.

The method proposed in [1], created a summary by finding feature representations in the lecture videos. This was done by detecting handwritten content regions and determining unique content written on the whiteboard. The paper proposed deep metric learning for feature representation that is used through multi-scale histograms of gradients and embeddings. Four key methods used to model the product was the detection of content, tracklet formation, conflict detection, and video segmentation forming keyframe summaries. The paper presented a novel solution, but errors while detecting smaller content regions lead to recall errors that resulted in precision errors

[2] uses a sequential decision-making process for video summarization using a deep summarization network. The principal idea is to predict a probability for each frame in the video showing how likely a frame is to be selected. The paper proposes a novel solution by using a reward function that is used for the reinforcement learning process. The architecture follows an encoder-decoder framework involving Markov models as well. The encoder extracts all important features of the frame using a CNN, while the decoder uses these features to produce the next hidden states using a bidirectional RNN. The reward function, in simple words, indicates how good the summary is. The video must not exclude any important information from the original video but must also be diverse at the same time. Information that might not be very important could also be a prerequisite for the actual information to be summarized. The

research does not consider audio while summarizing the video, and as it is a classroom lecture video the summary could be improved by considering the audio.

In [3], the key frames are extracted and using these key frames a summarized video is produced. This is done by uniform sampling, image histograms, and SIFT. For uniform sampling, an important concept is to extract or, select every k^{th} frame from thle video, where k is determined based on the length of the input video. The usual choice of summarized video length is about 5-15 percent of the original video. As a result, no semantic relevance is preserved. The histogram of the image provides information about its overall distribution, specifying the number of pixels for a particular brightness value from 0 to 256. In summary, the histogram can be extracted for every frame. The difference between the histograms of any two frames helps to determine the significant difference between them. If there are significant differences, it indicates a sudden change in the video scene that may contain interesting themes. SIFT has been used as it is invariant to scaling, rotation, translation, and small deformations and partially invariant to lighting, making it a robust descriptor for local functionality. Using K-means clustering the interesting frames are included in the summary and the key frames that are rich in local features but lack information and interesting content are filtered out. Gauss clustering is used to allocate query data points and maximize posterior probabilities. Hard clustering assigns data points to exactly one cluster. Soft clustering assigns a score to each cluster's data points. The score value indicates the strength of the association between the data point and the cluster. In [4], the lecture video is summarized by dividing it into segments or intervals based on the similarity between consecutive video frames. It uses a graph-based algorithm that evaluates the similarities between images extracted from the video segment. Initially, the images are obtained from the frames in the lecture video with the timestamps, then a rank is assigned to each image for inclusion in the video summary. Then a distance containing the ranks of the images is initialized and computed based on the similarity of the images. The images that represent the segment based on importance, rank, and similarity are selected.

Typically, all single-modal video summarization techniques consider video as input, and supervised techniques can learn relevant cues that are challenging to extract from hand-crafted heuristics in ground truth summaries. As a result, supervised methods frequently perform better than unsupervised ones. Generally, in most video summary generators the modes of input that are considered are visual content and textual content but do not consider audio as an input for summarization, and in a classroom setting audio plays a vital role in providing the context about what is being explained.

2.2 Multi-Modal Video Summarization

The papers [5], [6] used audio and video as a mode and produced a video summary representing the original video. [5] generates a multi-modal video summary using an explicit text-based query. The paper uses a specialized

attention network and contextualized word representations. The text-based query is used to control the video summary and the video is processed via a visual attention mechanism. Both were then combined through an interactive attention network to create a video summary generator. One major advantage was seen when Contextualized word representations over BoW (Bag of Words) were used to understand the semantic meaning of the text-based query.

In [6], the paper aims to automatically segment and index lecture videos in a much more efficient and faster method compared to the older methods of video segmentation and indexing. They improved the performance by using seed words to train the model. Seed words are the words recognized from the slide text. The model was tested on 500 lecture videos and achieved an average percentage improvement of 44.9 percent F score compared to indexing using unimodal approaches. The lecture videos served as the input and using OCR and ASR they obtained the transcripts of the lecture video and for indexing, they used LDA to obtain the titles and subtitles. The multi-modal video indexing has 2 main pipelines one which transcribes video text into textual descriptions and the other one which transcribes audio segments into text descriptions. The video pipeline segments videos into key frames based on whenever there is a slide transition. OCR is then used to recognize textual data. In the audio pipeline whenever there is a slide transition in the video, audio is segmented according to these slide transitions. CMU Sphinx which is an ASR Model is used to obtain transcripts of the audio segments. The transcriptions obtained using the OCR and the ASR engines give a collection of text documents containing the course content explained in the class by the lectures through the presentation in the class. The LDA model is used to divide these document collections into groups by using the co-occurrence relationship between the words. The transcription documents are taken as a mixture of latent topics and topics are taken as a mixture of latent words. These models are useful in extracting topics and sub-topics, indexing, and labeling the videos. The limitation of LDA models is that they scarify the performance on rare topics and in the case of lecture videos where if the slide text is limited or not enough to appropriately obtain audio segments this could negatively impact performance. They overcome this by making the LDA a semi-supervised LDA algorithm that takes a known parameter called seed word which is derived from the presentation slides. As a teacher or lecturer would use the board for explaining the topics the research didn't include the retrieval and processing of the blackboard or the whiteboard which is one of the major aspects of a lecture video.

3 Design And Architecture

3.1 Design and Architecture Overview

The overview of the architecture is shown in Fig. 1. Initially, a unprocessed classroom lecture video is fed into the model, which is channeled to the audio and video pipelines. The important time intervals are returned in both

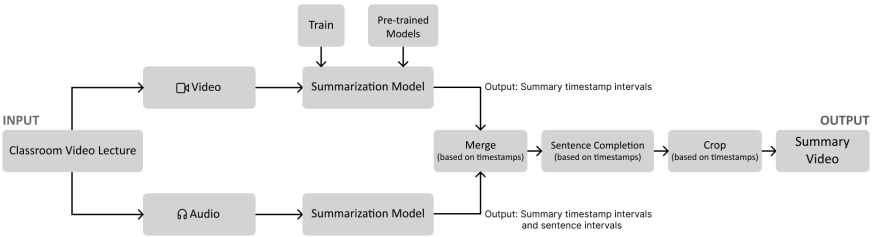

Fig. 1 Design Overview.

pipelines which are then merged and refined to produce the final summary intervals. These are then finally used to crop the original lecture video to produce the summary video.

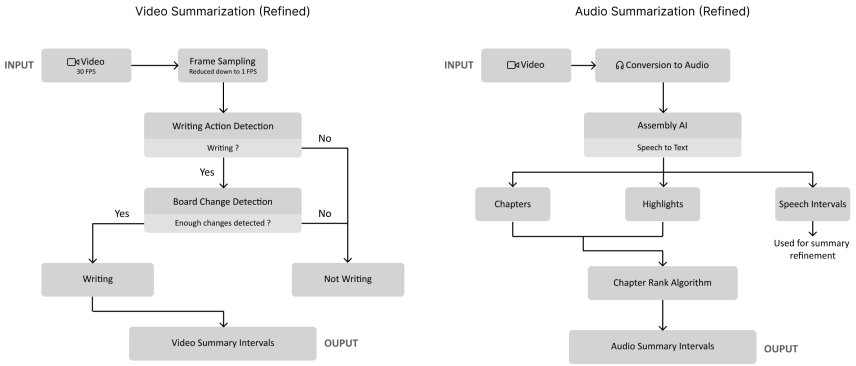

Fig. 2 Refined Audio and Video Summary Design.

3.2 Video Pipeline Design

Fig. 2 sheds light on the overall high level workflow of the video pipeline. Classroom lecture videos are of long duration which implies there is occurrence of an enormous number of frames that must be considered for any video processing application. The goal of this pipeline is to extract the key moments that will constitute the formation of the summary video. Key moments can be described as instances where the teacher is writing on the blackboard or whiteboard. Considering all the frames of a video to extract key moments is time and resource consuming. Hence, the frames are sampled to reduce total processing done on the video. Once sampled, writing action detection algorithm is performed on each frame to decide whether the teacher was writing in the frame or not.

Unfortunately, this algorithm has a tendency to yield false positive results. For example, if the teacher was pointing to something in a series of frames and the writing detection algorithm yielded positive outputs (indicating the teacher was writing), the corresponding time intervals should not be included in the summary intervals. Thus, each frame is also fed into a board change detection algorithm in order to verify the output of the writing detection algorithm.

The board change detection algorithm determines if or not the output of the writing detection algorithm was correct based on the how much change was detected on the board. If the teacher was found to be writing, the respective time intervals are included in the summary. Otherwise, the time intervals are disregarded.

3.3 Audio Pipeline Design

With reference to the pipeline shown in Fig. 2, the classroom video is initially converted to an audio file. This is sent to the AssemblyAI server which returns three documents, namely the chapters file which is the abstractive summary of the different topics discussed during the lectures along with their timestamps, the highlights file which is used to rank each word depending on the word frequency in the lecture obtained, and finally the speech intervals file that will later be used for sentence completion.

We then rank each of the summaries generated in the chapters file using the chapter rank algorithm. The time intervals of the higher-ranking summaries are selected for the final audio summary intervals. These intervals will be used to merge with the video summary intervals to create the final summary.

3.4 Finalizing Summary Timestamps

Once we obtain the merged summary intervals from the video and audio pipelines, we refine it using the speech intervals generated by AssemblyAI. The summary intervals are adjusted to make sure the clips don't begin at the middle of a sentence or end any sentences abruptly. This way, silences are also removed from the final summary video.

4 Implementation

4.1 Frame Sampling

Processing every frame is not very computationally effective, hence we reduce the number of frames considered. Generally, videos are recorded at 30 FPS. Instead of taking all 30 frames of a second into consideration, we take 1 frame as a representation of each second.

4.2 Writing Detection Algorithm

Each frame from the sample is processed to detect a teacher and their posture. Out of the thirty-three landmarks of the detected posture that are tracked

using the Mediapipe library (shown in Fig. 3), the right hip, right shoulder, right elbow and the right wrist are taken into consideration (assuming the teacher is right-handed).

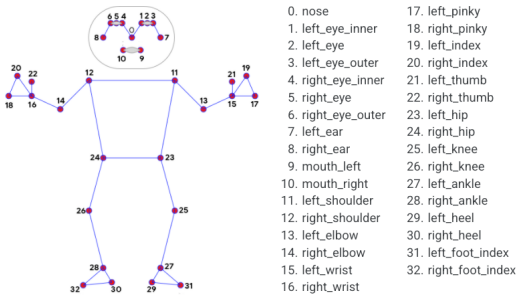

Fig. 3 Posture Landmarks in Mediapipe.

The angle formed between the hip, shoulder and elbow is termed as 'angle at shoulder' and the angle formed between the shoulder, elbow and wrist is termed as 'angle at elbow', as shown in Fig. 4. These two angles are calculated at every frame to decide if the teacher is writing or not. The detection of the action is based on the following condition:

$$60^\circ \leq angle\ at\ elbow \leq\ 180^\circ \tag{1}$$

$$60^\circ \leq angle\ at\ shoulder \leq\ 180^\circ \tag{2}$$

Once detected, the image frame and the status (writing or not writing) is returned as a tuple for further processing.

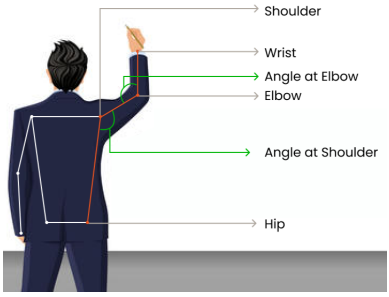

Fig. 4 Landmarks Considered for Writing Action.

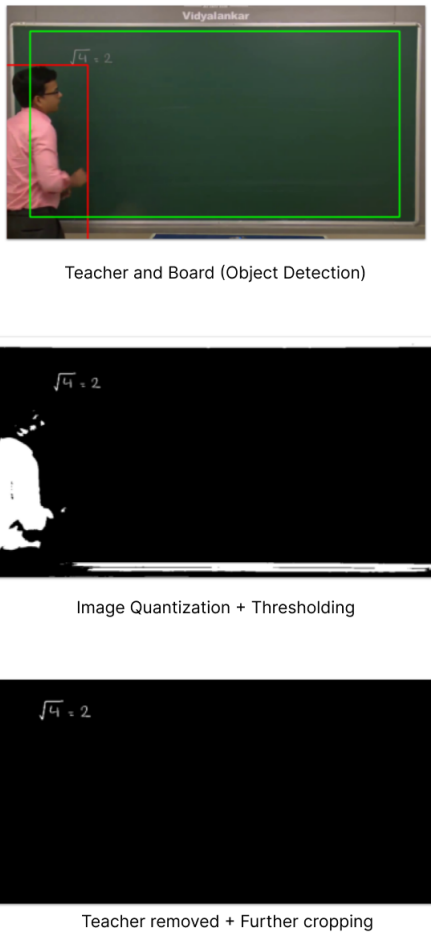

Fig. 5 Frame Preprocessing

4.3 Frame Preprocessing

Each frame undergoes a series of preprocessing steps before it is used by the board change detection algorithm.

Initially, the frame is fed into two object detection models to detect the coordinates of the board and the teacher. The board is then cropped out of the frame using the detected coordinates. Image quantization is performed using k-means clustering algorithm to convert all pixels into the two most dominant colors of the image. Subsequently, the converted image is then converted into a binary image with only black and white pixels. This makes sure that the board and the content written on the board are easily differentiable. In case the bounding box of the teacher overlaps with the coordinates of the board, the coordinates of the teacher inside the cropped image are recalculated once

the board is cropped out. This region is replaced with the contents of the previously preprocessed image, thus removing the teacher from the current frame. This also makes sure that even if the teacher is covering content on the board, it will be retained. It is further cropped to make sure that the borders of the board are removed and only the contents of the board are present in the final preprocessed image. The steps can be visualized using Fig. 5.

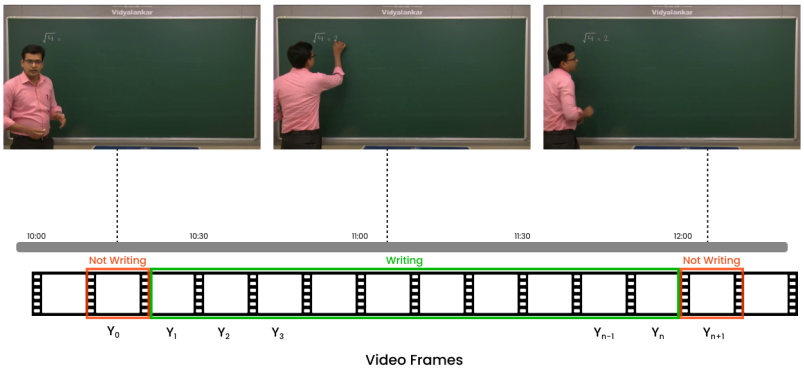

Fig. 6 Example Timeline with Annotated Writing Action

4.4 Board Change Detection Algorithm

The video frames and their respective status (writing or not writing) are obtained from the writing detection algorithm. But as discussed earlier, it could also yield false positive results, that is, the model could return the status as "writing" when the teacher was not. In order to verify its output, we use this board detection algorithm.

The key idea of this algorithm is to check the board contents in order to make sure the teacher was writing.

But when a teacher is writing, often the contents that are written on the board are covered by his/her hands. It would be difficult to check for changes on the board in that case. We have thus, compare the board contents when the teacher is not writing rather than when the teacher is writing. Though this sounds counter-intuitive, it can be easily visualized using Fig. 6. Instead of detecting changes every time the teacher is possibly writing (that is, frames y_1 to y_n), we simply check changes between frames y_9 and y_{n+1}. This way, we would always only need to check the changes between two images - before and after the teacher writes on the board.

These frames are preprocessed to retrieve the contents of the board, and a simple linear check is performed to check how many pixels are different between the two frames. If the change percentage exceeds a certain threshold,

we conclude the teacher was writing between the time of occurrences of the two frames that we worked with. In layman terms, if enough change was found between frames y_0 and y_{n+1}, we verify that that the teacher was actually writing from y_1 to y_n.

4.5 Time Interval Generation for Video

Using the modules mentioned above, we have formulated a linear algorithm to generate summary intervals for the video pipeline. The pseudo code of the algorithm is presented in Algorithm 1.

All the modules explained till this point are used in the above algorithm. All sampled frames are fed into the writing detection algorithm which returns a boolean value denoting whether the teacher was writing in the frame or not. The board change algorithm is performed in order to verify this output. If enough change is made on the board, we append the intervals during which the teacher was writing into the summary intervals.

Algorithm 1 Video Summary Intervals Generation

```
summary_intervals ← empty_list()
writing ← false
prev_frame ← preprocess(first_frame)
for each frame returned by writing action detector do
    current_frame ← preprocess(current_frame)
    if teacher is writing in current_frame then
        writing ← true
    else
        if writing = true then
            changes ← diff(prev_frame, current_frame)
            if changes > THRESHOLD then
                append new interval to summary_intervals
            end if
        else
            prev_frame ← current_frame
        end if
        writing ← false
    end if
end for
return summary_intervals
```

4.6 Audio Timestamp Generation Algorithm

As explained in the audio pipeline design, initially, the video is converted into an audio file which would further be utilized in this section of audio summary

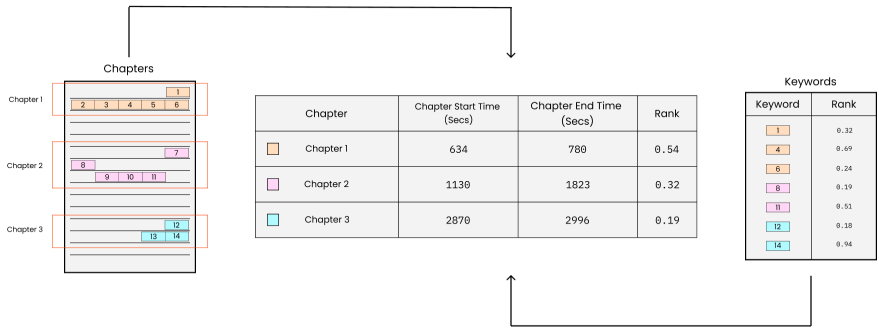

Fig. 7 Chapter Ranking Algorithm

generation. Firstly, an API call to AssemblyAI is fired to generate a highlights file, a chapters file and a speech intervals file.

The chapters are abstractive summaries formulated based on subtopic labeling. It is generated for that particular subtopic along with the time intervals for those subtopics when they are covered in the original audio file. The keywords file contains all the words and their respective ranks generated using the transcript. Stop words like "a, an, the, is" are removed, therefore would hold a rank of zero while ranking the chapters in the later part of the algorithm.

As shown in Fig. 7, using the above generated data, we add the ranks of each word that are present in that particular chapter summary to generate the rank for the chapter. In other words, the rank of each chapter is calculated by taking the summation of the product of rank of the word obtained in keywords file and frequency of the word in the given chapter.

$$Chapter\ Rank\ =\ \sum_{i=0}^{n} r(w_i) * f(w_i) \tag{3}$$

Where $f(w_\mathrm{i})$ is the frequency of the word w_i, $r(w_\mathrm{i})$ is its rank and n is the number of words in the chapter.

The chapters and their ranks are then sorted in descending order, following which we take the chapters with highest ranks along with their timestamps to be a part of the summary. We have statically chosen the threshold for the maximum duration of audio summary to be twenty-five percent of the total duration of the lecture video. Therefore, we consider chapter interval is considered if and only if the there is enough duration is left to be filled in the audio summary. In the rare case where the duration of a particular chapter is beyond a required duration but the chapter rank is high enough to not be discarded, we also include that chapter.

4.7 Merging of Time Intervals

After all the steps performed above, we are able to generate two time intervals i.e., one for the audio processed segment of the original video and the other of the processed video. Since each of the time intervals denote a segment of importance obtained by different means, hence we consider the union of all the time intervals. The final time interval generated here after the union, is the time interval that would be used to crop and make the summary.

4.8 Refinement of Time Intervals

Since these time intervals generated are based on actions performed and keywords detected, when merging them together, the summary is often turned out to be incomprehensible. Certain sentences or words are not completed, making the summary hard to understand. Sometimes individual clips started in the middle of the sentence and sometimes the clips were cut before he/she could finish the sentence. This leads to lots of inconsistencies while generating summarized video.

To solve this problem, we refine the summary intervals using the speech intervals generated by AssemblyAI which contains all sentences spoken by the teacher along with their respective start and end times. So, if the teacher doesn't speak, i.e., is silent for a period of time, it will not be recorded in this file, making it easy to remove silent time periods in the final video generated. Using the speech intervals, the previously generated time intervals are modified in order to make sure the individual clips include complete sentences spoken by the teacher. If an interval begins in the middle of a sentence, the start time is taken back to the beginning of the sentence. If an interval ends before the teacher completes her sentence, the end time is pushed to the end of the sentence. And finally, the moments when the teacher is not speaking, is cut out of the final summary intervals.

4.9 Cropping and Merging Video Segments

As a part of the penultimate step, we use the time intervals from the above step and crop the original video, hence obtaining clips of the most important segments of the lecture.

To be able to make a complete summary video, we must merge all the above generated video clips which is done in the ultimate step of the entire methodology. At the end of this step, we get a single summary video which is much more comprehensible and easier to understand.

5 Results and Discussion

In order to test our solution, a customized dataset was created. It contains 10 lecture videos ranging from 25 minutes to 58 minutes in length. The subjects of the videos were chosen such that the dataset included both theoretical and numerical content. The considered subjects include Computer Vision,

Table 1 Quantitative and Qualitative Measures

Video Subject	Video Length[1]	Summary Video Length	Precision	Recall	F1 Score
Computer Vision	0:25:32	0:17:02	0.4812	0.8259	0.6082
Civil Engineering	0:46:01	0:23:40	0.5509	0.6129	0.5803
Environmental Engineering	0:42:00	0:18:24	0.6411	0.5499	0.592
AFLL	0:58:09	0:22:16	0.7496	0.4941	0.5956
Secure Programming using C	0:45:22	0:16:01	0.494	0.384	0.4321
Python	0:52:17	0:25:59	0.995708	0.610986	0.757286
Linear Algebra	0:46:24	0:28:18	0.770203	0.7266	0.747798
Virtual Reality	0:47:18	0:22:31	0.819	0.664023	0.733417
Design and Analysis of Algorithms	0:56:54	0:18:44	0.514694	0.397794	0.448755
Mechanical Engineering	0:33:11	0:13:31	0.9383	0.445058	0.603745

Table 2 Confusion Matrix for Video Summarization

Manually Annotated Summary	Generated Summary	Parameter
Frame exists	Frame exists	True Positive
Frame does not exist	Frame does not exist	True Negative
Frame exists	Frame does not exist	False Negative
Frame does not exist	Frame exists	False Positive

Linear Algebra, Civil, Environmental Engineering, Python, Secure Programming using C, Virtual Reality, Design and Analysis of Algorithms, Mechanical Engineering and Automata Theory (AFLL). To evaluate the model, we have considered two broad metrics that are based on the quantity and quality of the summarized video.

To evaluate quantity, the length of summarized video and the actual video is considered. The 10 classroom lecture videos have an average time of 45.19 minutes. The average summary length for these 10 videos is 20.39 minutes. It is noticed that the summaries generated for numerical-based subjects usually a have longer summary length as it involves a lot of writing to cover the problems discussed in the lecture.

$$Precision = \frac{No.\ of\ desired\ frames\ in\ summary}{Length\ of\ generated\ summary} \tag{4}$$

$$Recall = \frac{No.\ of\ desired\ frames\ in\ summary}{Length\ of\ ideal\ summary} \tag{5}$$

To evaluate quality, F-score is considered. Our dataset is manually annotated to mark important frames in the original lecture video which is then tested against the summary generated. F-score ranges betweeen 0.43 to 0.75 on our custom created dataset. The macro average F1 score calculated is 0.6099. Table 1 shows our model performance in terms of quality and the quantity of the summary videos generated. The scores obtained in Table 1 are based on the manual annotations performed which is highly subjective. Key frame comparison was done using the confusion matrix as represented in Table 2.

The total time interval obtained from the video pipeline and audio pipeline was

[1] given timestamps are referred in the HH:MM:SS format

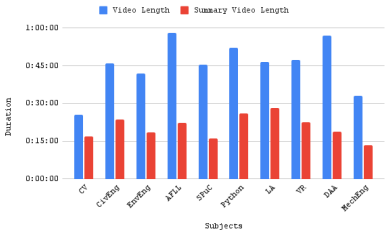

Fig. 8 Original and Summary Length

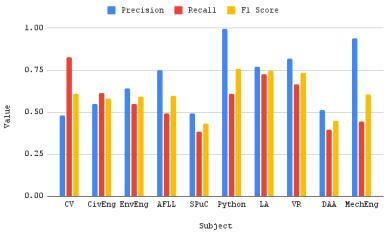

Fig. 9 Precision, Recall and F-score

dependent on the nature of the subject. In subjects such as Computer Vision and AFLL which are writing intensive, the summary produced relied more on the video pipeline. Whereas, for concept heavy subjects such as Computer Architecture and Python, the summary relied greatly on the audio pipeline. It was observed that the summaries dependent on the video pipeline produced a longer summary in duration when compared to the summaries dependent on the audio pipeline.

6 Conclusion

This paper presents a novel solution for classroom video summarization by taking video and audio into account as 2 individual modes. The final lecture video summary is generated by identifying the important timestamps in the original classroom lecture video, which are obtained through the video and audio pipelines. The timestamps in the audio pipeline are obtained using chapters, highlights, and transcripts that allow us to identify the key moments in the classroom lecture video. The teachers' posture acts as a basis for identifying the key moments which are then used to extract key frames in the video pipeline. Once extracted, the key frames are filtered out using a board change detection algorithm. Finally, the timestamps from the audio and the video pipeline are merged together and parts of the original classroom video are cropped out and put together using the timestamps and the final classroom video summary is obtained.

The accuracy of the lecture video can be further improved by taking more factors like classroom slides, multiple boards, and moving boards into account. The results produced qualitatively and quantitatively show that the summary can be used in real-world scenarios by teachers and students.

7 Statements and Declarations

7.1 Availability of Data and Material

The dataset used here is a custom made dataset by gathering videos from different sources. The dataset can be made available upon request from the corresponding authors.

7.2 Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

7.3 Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

7.4 Author Contributions

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by all authors. All authors read and approved the final manuscript.

References

[1] Kota, B.U., Stone, A., Davila, K., Setlur, S. and Govindaraju, V., 2021, January. Automated whiteboard lecture video summarization by content region detection and representation. In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 10704-10711). IEEE.

[2] Zhou, K., Qiao, Y. and Xiang, T., 2018, April. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).

[3] Jadon, S. and Jasim, M., 2020, October. Unsupervised video summarization framework using keyframe extraction and video skimming. In 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA) (pp. 140-145). IEEE.

[4] Rahman, M.R., Shah, S. and Subhlok, J., 2020, December. Visual summarization of lecture video segments for enhanced navigation. In 2020 IEEE International Symposium on Multimedia (ISM) (pp. 154-157). IEEE.

[5] Huang, J.H., Murn, L., Mrak, M. and Worring, M., 2021, August. Gpt2mvs: Generative pre-trained transformer-2 for multi-modal video summarization. In Proceedings of the 2021 International Conference on Multimedia Retrieval (pp. 580-589).

[6] Husain, M. and Meena, S.M., 2019, February. Multimodal fusion of speech and text using semi-supervised LDA for indexing lecture videos. In 2019 National Conference on Communications (NCC) (pp. 1-6). IEEE.

[7] P. J. Guo, J. Kim, and R. Rubin, "How video production affects student engagement: An empirical study of mooc videos," in Proceedings of the first ACM conference on Learning@ scale conference, 2014, pp. 41–50.

[8] Whatley, J. and Ahmad, A., 2007. Using video to record summary lectures to aid students' revision. Interdisciplinary Journal of E-Learning and Learning Objects, 3(1), pp.185-196.

[9] Tovinkere, Vasanth, and Richard J. Qian. "Detecting semantic events in soccer games: Towards a complete solution." IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.. IEEE Computer Society, 2001.

[10] Reconstructive Sequence-Graph Network for Video Summarization, Bin Zhao, Haopeng Li, Xiaoqiang Lu, Senior Member, IEEE, Xuelong Li*, Fellow, IEEE

[11] Combining Global and Local Attention with Positional Encoding for Video Summarization, Evlampios Apostolidis; Georgios Balaouras; Vasileios Mezaris; Ioannis Patras, IEEE

[12] Video summarization with supervised learning, Jayanta Basak; Varun Luthra; Santanu Chaudhury, IEEE

[13] Video Summarization Using Deep Neural Networks: A Survey Evlampios Apostolidis, Eleni Adamantidou, Alexandros I. Metsai, Vasileios Mezaris, Senior Member, IEEE, and Ioannis Patras, Senior Member, IEEE

[14] Video Summarization with Long Short-Term Memory, Ke Zhang, Wei-Lun Chao1, Fei Sha, and Kristen Grauman

[15] K. Davila and R. Zanibbi, "Whiteboard video summarization via spatio-temporal conflict minimization," in International Conference on Document Analysis and Recognition (ICDAR), 2017.

[16] Ejaz, N.; Mehmood, I.; and Baik, S. W. 2013. Efficient visual attention based framework for extracting key frames from videos. Signal Processing: Image Communication 28(1):34–44.

[17] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in ECCV, 2014.

[18] Chekuri Choudary and Tiecheng Liu. 2007. Summarization of Visual Content in Instructional Videos. Multimedia, IEEE Transactions on 9 (12 2007), 1443 – 1455.