ISSN: 0976-9102 (ONLINE)
DOI: 10.21917/ijivp.2025.0503

TAILORED VIDEO SUMMARIZATION: CATERING TO PATIENT AND IMPATIENT USERS

K.R. Sarath Chandran, Adithi Shankar, Geethapriya Thandavamurthi
Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, India

Abstract

People across various fields rely on automated video summarization tools to manage extensive video content efficiently. This research focuses on developing a dynamic, user-centered approach to video summarization, accommodating both patient and impatient user needs. The system aims to handle lengthy videos by identifying and cataloging all objects within them. It follows a three-step process: Object-of-Interest selection, object detection/localization, and video summarization. For patient viewers, it offers comprehensive scene identification and storage. For impatient users, it provides concise summaries quickly. By adapting itself to individual preferences, this research will make videos more accessible and useful by providing personalized video summaries which will help avoid information overload in various spheres such as security, entertainment, or personal documentation. This research used deep learning models like YOLOv8, ResNeXt as well as LSTM to implement this user- centric approach to video summarization.

Keywords:

OOI, Patient User, Impatient User, YOLOv8, ResNeXt, LSTM, RNN, CNN

1. INTRODUCTION

Video summarization is one of the most important technologies that help reduce time in managing an ever- growing volume of video content in varied fields. In essence, video summarization condenses lengthy videos into concise summaries offering valuable advantages. The application of video summarization in security applications, like in the utilization of dash cameras by law enforcement agencies, helps in identifying and cataloging relevant scenes to aid the process of investigation. It also helps similarly in the entertainment sector by speeding up the retrieval of specific scenes, thereby providing a better understanding of the narrative dynamics to the viewers. In personal documentation, video summarization helps in pinpointing the relevant moments within an extensive video archive.

Our objective is to simplify the process of reviewing videos as lengthy videos can be challenging for individuals to analyze on their own thoroughly. We strive to develop a system that enables users to obtain summarized versions of videos. This system will function by identifying and preserving crucial scenes that align with the user's interests. It will achieve this through the implementation of advanced techniques for object detection, frame extraction, and summarization. Users will have the opportunity to specify the Objects of Interest they desire. Subsequently, the system will retain only those scenes featuring the designated objects as the output for the patient users. The output video will then be further summarized while ensuring the focus still lies solely on the OoI and this is returned as an output for the impatient users.

At the core of our strategy lies the recognition that users have diverse needs and preferences. While certain individuals seek thorough scene identification and comprehensive video summaries, others prioritize swift and concise highlights of relevant material. This distinction emphasizes the significance of adapting video analysis to accommodate both time- constrained and patient users, each with distinct goals and temporal limitations.

The research that we have embarked on has three major stages that entail OoI selection, object detection, and localization, as well as video summarization. Our main aim of doing this study is to come up with effective ways of solving the problem of video summarization in today's data-rich society because of the vast quantity of footage that needs to be watched.

2. EXISTING WORKS

The following are some existing systems that have been proposed by various research efforts.

Ul Haq et al. [10] have presented a journal paper that proposes a user-customized video summarization system based on deep learning. This system allows the users to select a set of objects that they would like their final video summary to be based on. These objects like cars, human beings, buses, etc are called Objects of Interest (OoI). The results produced by their system demonstrate high levels of accuracy ranging from 99.2% to 99.9%. This paper also describes the development of a desktop application to allow user-driven video summarization based on the selected OoIs. A significant research gap in the paper is the fact that it focuses only on catering to impatient users who want quick access to essential actions and events within videos, therefore requiring concise and action-centric video summaries. The paper does not address the needs of patient users who seek detailed video summaries containing all frames related to specific OoIs detected by the YOLOv3 framework. This one-sided approach does not provide a balanced solution that caters to both impatient and patient users.

Panagiotakis et al. [7] proposed a new approach to personalized video summarization by using a recommender system. The system is modeled to create video summaries based on individual user preferences by combining user-provided video segmentation data and the duration of features in the video segments. The primary objective is to produce video summaries that closely match the subjective criteria as well as the preferences of the user. However, a major limitation of their study is the lack of comparative analysis with established video summarization techniques or algorithms. This absence makes it challenging to evaluate the efficiency and performance of the proposed system against existing methods in the field.

Negi et al. [5] proposed a deep learning-based framework for efficient video summarization, using object detection and

unsupervised learning methodologies. Their system involves several steps. Initially, the frames are extracted from the video input. This is followed by object detection utilizing YOLOv5, used to select frames containing the target object. Next, features are extracted using VGG-16 and ResNet-50, along with Principal Component Analysis (PCA) employed for feature compression. Later, K-means clustering is applied to extract the best candidate frames. Post-processing involves utilizing the Pearson Correlation Coefficient (PCC) to eliminate redundant frames and extract final keyframes. The results of this system demonstrate better performance when compared to existing models. Specifically, an enhanced recall score was achieved.

Negi et al. [6] wrote another paper that focused on the performance evaluation of the previously proposed approach. They used a wide range of metrics such as summary length, precision, recall, PR curve, and mean average precision (mAP). These metrics are quantitative measures that are capable of assessing the quality of the summarization process. The proposed approach is able to identify the most effective video summarization framework with the best summary length under varying conditions. The paper also offers insights into system resource utilization during model training. This gives us an idea of the computational requirements needed to implement the proposed approach. This aspect of the research contributes valuable information regarding the practical feasibility and scalability of the proposed method in real-world applications.

There have been patents filed for systems generating video summarizations like Karakotsios et al. [3] who developed a method for generating summarizations based on user input and feedback. The first video summarization produces frames of video data that are likely to be of interest to the user. User feedback on the first video summarization is collected and used for the second summarization. The second video summarization is provided to the user as output. This allows a high level of personalization by the user. The two-step process improves the relevance and effectiveness of the generated summaries.

Saini et al. [8] performed a comparative analysis of various deep learning approaches used in video summarisation. This paper also provides a list of potential recommended applications based on the literature. This paper reviews deep learning-based video summarization methods, including the Multi-edge optimized LSTM RNN for VS, which achieved an impressive F-score of 92.4%, outperforming other recent techniques on the VSUMM dataset. This focus on LSTM's exceptional performance is a key takeaway from this research.

Wang et al. [11] proposed a video summarization network that utilizes an encoder-decoder framework, integrating a convolutional neural network (CNN) for feature extraction and a bidirectional long short-term memory (LSTM) network for decoding. It also makes use of a self-attention mechanism to emphasize key features during summarization. Experimental results on two datasets show its effectiveness compared to seven other methods, confirming that it is indeed a valuable method to perform video summarisation.

Lin et al. [4] have proposed a novel hierarchical LSTM network with attention for video summarization. Instead of using a standard 2D ResNeXt, they have employed a 3D ResNeXt to extract a more delicate video representation. After this, the system employs a hierarchical LSTM with a bottom layer as well as a top layer. The bottom layer of the LSTM is capable of generating a fine-grained analysis while the top layer can identify more abstract moments. Then, attention mechanisms are used to fine-tune the summary by selecting only the important information in the summary while ignoring other information. This system aims at capturing temporal dependencies present in the video.

Deshpande et al. [1] provided experimental results on different methods of object identification namely Region-based Convolutional Neural Networks (RCNN), Faster-RCNN, and You Only Look Once (YOLO). It was concluded that YOLO was the best in terms of speed but also that YOLO was not the most accurate. The paper concluded that YOLO was still capable of providing efficient object detection without compromising on performance. This paper provided necessary research insights into which object detection framework was best suited for the research at hand.

A similar comparison was made by Tan et al. [9] who published a survey that explores the three object detection methods RetinaNet, SSD, and YOLO v3 in image recognition, focusing on pill identification. It concluded that YOLO v3's faster convergence makes it suitable for quickly adapting to changes in pharmacy settings. A limitation of this paper is the fact that it measures performance based only on pill identification. It does not measure the performance of object detection for general scenarios with multiple objects. Nevertheless, YOLO's performance remains unmatched.

Jain et al. [2] discuss the usage of the COCO dataset in object detection. The COCO dataset, a large-scale object detection dataset, is widely used in AI and computer vision projects. Object detection involves both grouping similar objects and accurately localizing them using bounding boxes. The COCO dataset also provides 80 classes of objects that it can detect. This makes COCO suitable for usage in generic object detection projects. Moreover, the presence of these 80 classes also allows an easy selection of Objects of Interest concerning this current research by the user.

3. PROPOSED METHODOLOGY

3.1 PATIENT USERS

This phase involves three main modules: Object Counting, Extracting Frames Containing Objects of Interest (OoI), and Detailed Summary Creation. In the Object Counting module, Ultralytics YOLO performs object detection used to count different objects in the input video. This gives the user an idea of the most common objects present. This is used to create the drop-down for selecting the Object of Interest. The Extracting Frames Containing OoI module uses OpenCV and YOLO to extract frames containing the OoIs mentioned as input by the user. These frames are then saved for further processing. These saved frames serve as input for the Impatient User and Summary Video Creation modules. Lastly, the Detailed Summary Creation module converts the annotated frames into a video file, adjusts its frame rate, and converts it to the mp4 format for easier download.

3.2 IMPATIENT USERS

This phase contains four modules: Data Handling, Summarization Model, Training, and Short Summary Creation.

The Data Handling Module manages data loading and preprocessing. The SumMeDataset class extracts image frames and their associated importance scores from a CSV file. The Summarization Model Module defines the architecture of the summarization model, which contains the newresNext encoder for feature extraction and the Model class for LSTM-based summarization. The Training

Module handles the training process. It uses MSE loss and Adam optimizer to improve model parameters iteratively. Lastly, the Short Summary Creation Module displays the summarization output visually after selecting the most relevant frames based on their predicted scores. Together, these modules can generate a short video summary of the initial input video.

The overall pipeline diagram is illustrated in Fig.1.

4. IMPLEMENTATION DETAILS

4.1 PATIENT USER

4.1.1 Object Counting Module:

This module involves passing the input video through an algorithm that implements object detection and counts the different objects present in the video. The custom-made algorithm provides information regarding the most commonly present objects in the input video.

The algorithm begins by importing necessary libraries including Ultralytics YOLO for object detection, OpenCV for video processing, and the operator module for sorting dictionaries. Following this, an instance of the YOLO model is created using a pre-trained model file named yolov8n.pt. The algorithm proceeds to open the video file for processing using OpenCV's VideoCapture function, obtaining essential parameters such as width, height, and frame rate to define the object counting region. A rectangle is then defined across the video frame using the width and the height and is used as the region for counting objects. Then, an instance of the ObjectCounter class from the Ultralytics solutions module is instantiated, configuring it with visualization options, counting regions, and preferences.

Next, the algorithm's parameters are modified to detect all 80 classes defined by the COCO dataset. Additionally, a dictionary is initialized to store counts for each class. The algorithm proceeds to iterate through each frame of the video, performing object detection using the YOLO model. Object counts are updated for each detected object class based on the tracking results.

After all the frames are iterated through, the dictionary containing object counts is sorted in descending order based on the count values. Classes with counts less than the threshold value are filtered out to focus on significant object classes. Finally, the sorted list of object classes along with their corresponding counts is printed. This gives the user a good idea as to which objects are present in the video and how frequently.

The Fig.2 illustrates the architecture of the Object Counting module.

4.1.2 Extracting Frames Containing OoI Module

This module uses OpenCV for frame extraction. The extracted frames are then passed onto the object detection algorithm, YOLO. Frames that contain the OoIs specified by the user are saved in a separate folder. These frames are then passed as input to the Impatient User part of the research. The frames are also passed onto the Summary Video Creation Module, which creates the output video for the Patient User. The flow of the code is discussed next.

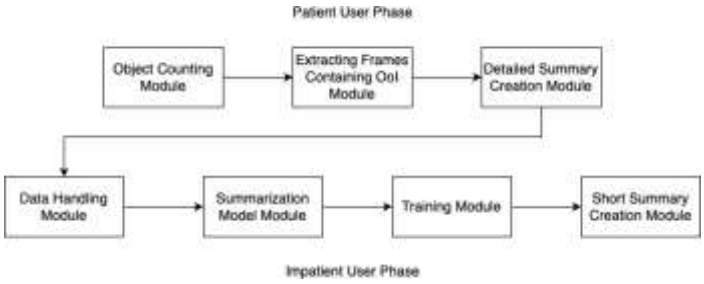

Fig.1. Overall Pipeline

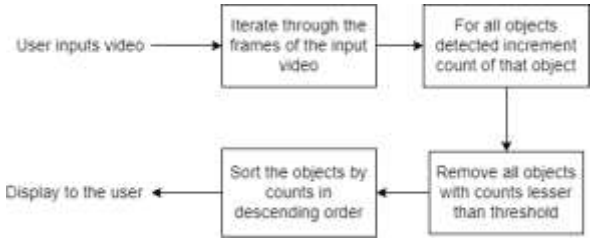

Fig.2. Object Counting

The process begins by asking the user to input a list of objects of interest (OoI). Next, an instance of the YOLO model is instantiated, using a pre-trained model file named yolov8m.pt. After this, the designated video file, specified by the variable vid, is opened for processing. To organize the output, an output folder named frames is created to store the frames containing the OoI. The algorithm then iterates through each frame of the video. Within each iteration, the YOLO model is used to predict objects present in the frame. The predictions are then filtered to include only those pertaining to the specified OoI. Then the frame is annotated using a bounding box and corresponding class label. Each annotated frame is saved as an image within the output folder named frames. The process terminates upon reaching the end of the video, stopping the frame processing loop.

The Fig.3 illustrates the architecture of the Object Detection and Frame Extraction module.

4.1.3 Detailed Summary Creation Module:

This module completes the task of converting the sequence of images with the OoI present into a video file, adjusting its frame rate, and converting it into a different video format for easier download. The flow of the code is discussed next.

Initially, the images in the frames folder are sorted using the natsort library to ensure proper sequencing. The images are iterated over, and each frame is added to the video file using the write method of the VideoWriter object. After writing all frames, the video file is closed and saved.

Next, the code reads the generated video file and adjusts its frame rate. The frame rate is increased by a factor of 32 to accelerate the video. A new VideoWriter object is created with the updated frame rate and each frame of the original video is written to the new video file.

Next, the moviepy library is utilized to convert the newly created video file to the mp4 format, which is a commonly used

format for web-based applications and platforms. Finally, the resulting mp4 file is downloaded. The Fig.4 illustrates the architecture of the Patient User Summary.

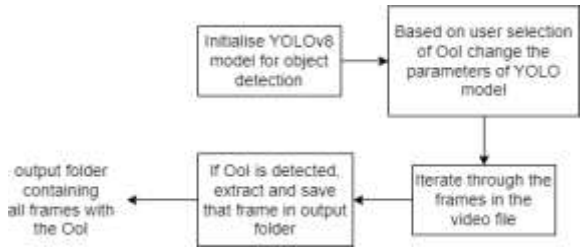

Fig.3. Object Detection and Frame Extraction Creation module

4.2 IMPATIENT USER

4.2.1 Data Handling Module:

The Data Handling Module handles data loading, preprocessing, and batching. Our implementation consists of the SumMeDataset class that inherits from PyTorch's Dataset class. This class facilitates loading and preparing the image and importance score data for training and evaluating the model.

It takes several arguments during initialization:

- annotations_filename: The path to a CSV file containing video names and their corresponding importance scores or ground truth scores for each frame.
- img_dir: The path to the directory containing the image frames for the videos.
- transform (optional): A function used to preprocess images (e.g., converting them to tensors).
- target_transform (optional): A function used to preprocess the importance scores.

The class loads annotations from a CSV file using pandas. It extracts frame scores for the specified video and stores references to transform and target transform functions. The len_() returns the total number of frames (and scores) in the dataset. The core lies in the _getitem_() method. This method receives an index input and then retrieves the corresponding image and score. It loads the image from the directory, extracts the score from the DataFrame, and applies any transformations to both the image and score, subsequently returning them as a tuple. Hence, data formatting for training is ensured by this module.

- **Encoder (newresNext):** The new_resNext class is an encoder. It extracts high-level features from input images using a ResNeXt convolutional neural network. During initialization, the network can be customized. You can specify parameters like fc_size, large, and pre-trained which is by default True. fc_size sets the output size of an added fully connected layer. large indicates how big the ResNeXt model is. As pre-trained is True, it uses learned weights from a pre-trained ResNeXt model. Based on these parameters, the class selects an appropriate ResNeXt architecture from the torchvision.models module. This could either be **resnext5032x4d** or **resnext10132x8d**. The forward method of the new_resNext class defines the forward pass computation inside the network. It takes an input tensor representing image data and passes it through the selected ResNeXt model. The resulting image features then get returned by the forward method.
- **RNN (Model):** This class also inherits from torch.nn.Module and defines the core architecture of the video summarization model. It takes several arguments during initialization, including the input_size from the ResNeXt model, output_size which gives the number of predicted importance scores, hidden_size of the LSTM layer, and n_layers which gives the number of LSTM layers to be stacked. The class creates an LSTM layer with these specifications and a final linear layer to map the LSTM output to the predicted importance scores. The forward method receives a batch of encoded image features extracted by the encoder (newresNext) and passes it through the LSTM layer which processes the sequential input data and captures temporal dependencies and patterns across frames in the video. The output of the LSTM layer is then passed through a fully connected linear layer, which maps the LSTM output to the desired output size. This method returns a tuple containing both the predicted scores and the hidden state of the LSTM, which might be useful for further processing. Additionally, the Model class defines an initialization method, initHidden, to initialize the hidden state of the LSTM layer. This method ensures that the hidden state is appropriately initialized before the forward pass computation begins.

The Fig.6 illustrates the architecture of the Data Handling Module.

4.2.2 Summarization Model Module:

This module defines the architecture of our neural network model. It includes two primary classes: newresNext and Model.

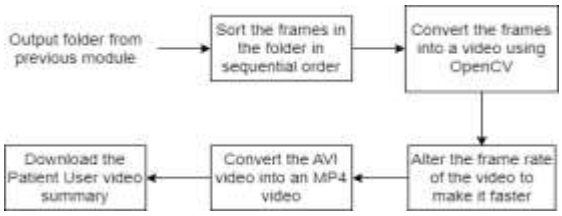

Fig.4. Patient User Summary Creation

	A	B	C	D
1	video_name	start_frame	end_frame	gt_score
2	Bike_Polo	0	1	0.2
3	Bike_Polo	1	2	0.2
4	Bike_Polo	2	3	0.2
5	Bike_Polo	3	4	0.2
6	Bike_Polo	4	5	0.06666666667
7	Bike_Polo	5	6	0.06666666667
8	Bike_Polo	6	7	0.06666666667
9	Bike_Polo	7	8	0.06666666667

Fig.5. Annotations.csv

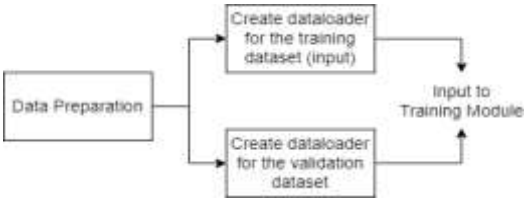

Fig.6. Data Handling

The Fig.7 illustrates the architecture of the Summarization Model Module.

4.3 TRAINING MODULE

The Training Module handles the training process of our model. It includes functions for initializing the model, defining loss functions, optimizing parameters, and evaluating the model's performance.

First, the training module defines the loss function and optimizer. In this case, Mean Squared Error (MSE) loss is chosen as the loss function. MSE measures the average squared difference between the predicted scores and the ground truth scores, providing a quantitative measure of the model's performance. The equation for MSE is defined at Eq.(1):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - Y_i^{'}\right)^2 \quad (1)$$

where Y_i is the predicted scores and $Y^{'}$ is the ground truth scores.

The Adam optimizer is then used to optimize the model parameters by updating them based on the computed gradients of the loss function. This optimization process aims to minimize the difference between predicted and actual scores, hence improving the model's ability to generate accurate summaries.

Moving on to the training loop, this iterative process begins by iterating over batches of data obtained from the data loader. Each batch consists of a subset of samples from the dataset, allowing for efficient processing and parameter updates. Within the loop, the data and the model are moved to the appropriate computational device, whether it be a CPU or GPU, to perform computations. For each batch, the model conducts a forward pass to compute predictions for the input data. These predictions are then compared to the ground truth scores, and the loss is calculated using the previously defined.

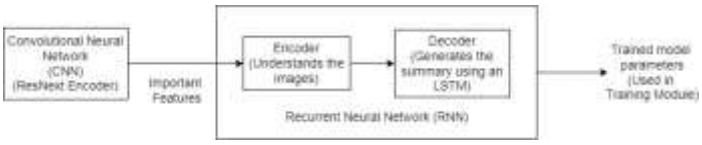

Fig.7. Summarization Models

Subsequently, the optimizer steps in to update the model parameters based on the gradients of the loss function obtained during the backward pass. This step, known as backpropagation, is essential for adjusting the model's parameters in the direction that minimizes the loss, thereby improving the model's performance over time. By iteratively repeating this process for multiple epochs, the training module ensures that the neural network model learns to generate accurate summaries by fine-tuning its parameters based on the provided training data.

Periodically, the model performance is evaluated on a separate validation dataset to ensure it generalizes well to unseen data. This helps prevent overfitting, where the model learns to memorize the training data rather than generalize to new examples. In the provided code, a validation loop is implemented to assess the model's performance on the validation dataset, calculating the validation loss.

Overall, the training module manages the entire training process, starting from model setup to optimizing the parameter, This allows the model to effectively learn from the data and improve its summarization capabilities. The Fig.8 illustrates the architecture of the Training Module.

4.4 SHORT SUMMARY CREATION MODULE

The Short Summary Creation module in the provided code segment serves an important role in extracting the output frames generated by the summarization model and representing it in a visually interpretable format. Its process involves several key steps.

Firstly, the module starts by determining the number of output frames to include in the summary based on a predetermined desired output ratio. This ratio dictates the proportion of the video content that should be present within the summary. By employing the calculate_output_frames function, the code dynamically computes the optimal number of frames to represent the summarized content. Understanding the ground truth scores of every frame is vital for this computation. By doing so, the chosen frames can accurately encompass the most significant parts of the video.

$$\text{frame\_ratio} = \text{gt\_score}(i)/\text{total\_score} \quad (2)$$

Once the number of output frames is established, the module proceeds to select the top-ranked frames based on their predicted scores. Leveraging the predicted scores, preds, the code identifies the frames with the highest ranking and extracts them for inclusion in the summary. These selected frames are then saved as individual image files in a designated output directory and converted into a video to provide as the output.

With the output frames secured, the module moves to the visualization phase. Here, the code retrieves the saved output frames from the output directory and prepares them for display. Utilizing the make_grid function from the PyTorch library, the output frames are organized into a well-organized grid layout. This layout enhances the presentation of the summary by arranging the frames in a structured and easily digestible manner.

Finally, the visualized output frames are showcased to the user through the show function. This function uses Matplotlib to generate a visualization of the output frames and display them within the output of the code execution. By presenting the output frames in a graphical format, users can assess the quality and relevance of the summarized content. This visual representation improves the interpretability of the summarization results, facilitating informed decision-making and evaluation of the summarization model's performance.

In summary, the visualization module plays a pivotal role in presenting the output frames of the summarization model in a visually appealing and informative manner. The Fig.9 illustrates the architecture of the Visualization Module.

5. RESULT

5.1 PATIENT USER

In Fig.10, we notice a video frame where the Object of Interest is a bicycle. Our system uses the YoLoV8 object detection model to detect and label bicycles in each frame. In the screenshot, we see boxes around the bicycle with labels showing the model's capability to recognize the specified OoI.

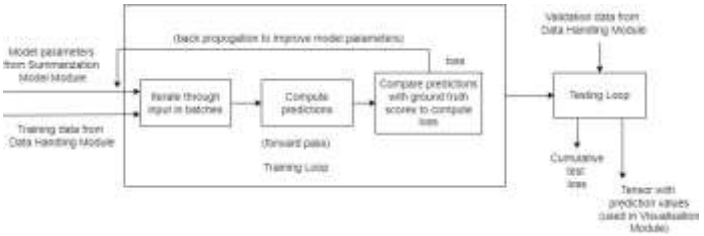

Fig.8. Training Module

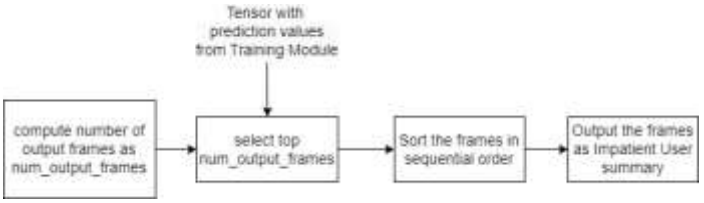

Fig.9. Short Summary Video Module

Fig.10. Frame with OoI as Bicycle

Fig.11. Frame with OoI as Bus

Fig.12. Frame with OoI as Bicycle and Bus

Similarly, Fig.11 shows a frame from the video with a bus as the focus. The YOLO v8 model successfully identifies and labels the presence of multiple buses. This shows how flexible it is in detecting objects.

The Fig.12 shows a frame where both the bicycle and the bus are entered as the OoI. Both objects in each frame are detected and labeled. This shows how the system can handle multiple objects at once with ease, showcasing its versatility.

The Fig.13 showcases frames extracted from the video summary, highlighting scenes where a bus is identified as the OoI. These frames are part of the summarized video created using OpenCV's 'VideoWriter' tool. In this case, the summary only includes frames where a bus is identified as the OoI.

Fig.13. Patient User Video Summary with Bus as OoI

5.2 IMPATIENT USER

In this example, footage showing kids playing with leaves is caught on video. The car is taken as the Object of Interest. First, the Patient User model processes it, looking for cars. It removes frames without cars, giving a subset. Next, that car-only subset feeds into the Impatient User model. With 1291 total input frames, this model uses ResNeXt and LSTM to analyze the content, extracting key frames. Using a pre-determined formula, it calculates 30 frames as the optimal number for summarizing. The Fig.14 represents these 30 selected frames arranged in a visually informative grid format.

5.3 PERFORMANCE ANALYSIS

5.3.1 Patient Users Detailed Summary Generation Times:

The Table.1. contains the data regarding the amount of time taken to generate a summary with single OoI as well as multiple OoIs.

Table.1. Summary Generation Times

Sl. No.	OoI	Frames with OoI	Time Taken (seconds)
1	1 OoI - bicycle	46	77.48
2	1 OoI - bus	192	103.11
3	2 OoIs - bicycle and bus	231	111.44

The table reveals a noticeable trend: as the number of OoIs increases, there is a corresponding increase in the time required to generate a summary. This correlation is because of the rise in the number of frames containing the specified OoIs, particularly when the object is present throughout the video. The increase in OoIs directly impacts the computational effort needed for the summarization process.

5.3.2 Comparison of Object Detection Algorithms:

Existing research work has shown that YOLO is the best suited Object Detection Algorithm for this research. YOLO's high speed capabilities overrule its decreased accuracy. The high speed of YOLO is made clear by the study conducted by Deshpande et al. [1] represented in Fig.15 and Fig.16.

Methods	Fast R-CNN	Faster R-CNN	YOLO
Time (Sec)	83.81045	73.88907	19.389161

Methods	**Fast R-CNN**	**Faster R-CNN**	**YoLo**
Time (s)	83.81045	73.88907	17.389161

Fig.14. Frames from Summarized Video

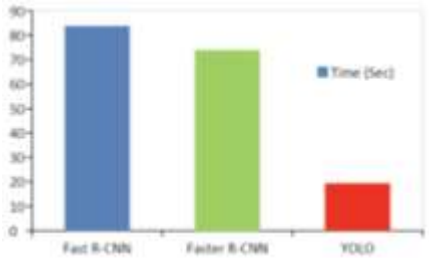

Fig.15. Comparative Analysis Based on Time [1]

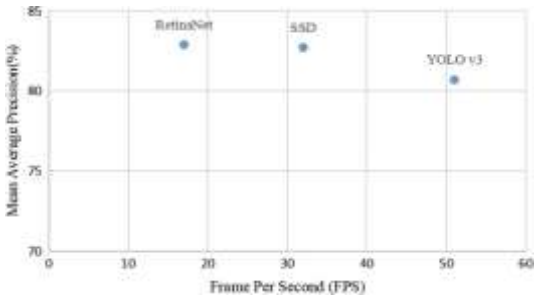

Fig.16. Time Taken to Detect [1]

Tan et al. [9] highlights the minor drawback that YOLO models have, that is, the lower accuracy. The trade-off between speed and accuracy concerning this research made it clear that speed was more important than accuracy. This is highlighted in Fig.17.

Fig.17. Performance of Deep Learning Models

5.4 PERFORMANCE METRICS ON YOLOV8 CONFUSION MATRIX

A confusion matrix is a tool used in machine learning and statistics to evaluate the performance of a classification model. It provides a summary of the predictions made by the model compared to the actual ground truth values across different classes.

Here's how a confusion matrix works:

- True Positives (TP): These are the cases where the model correctly predicts the positive class.
- True Negatives (TN): These are the cases where the model correctly predicts the negative class.
- False Positives (FP): Also known as Type I errors, these are the cases where the model incorrectly predicts the positive class when it's actually negative.
- False Negatives (FN): Also known as Type II errors, these are the cases where the model incorrectly predicts the negative class when it's actually positive.

The Fig.18 and Fig.19 show the confusion matrix of the 80 different classes defined by COCO128 dataset used to train the YOLOv8 with different confidence thresholds. The main diagonal shows the number of correct predictions made from each class. Since most of the values in the given images lie on the diagonal, one can conclude that the model works fairly well. It is clear that the model does not work well only for background detection.

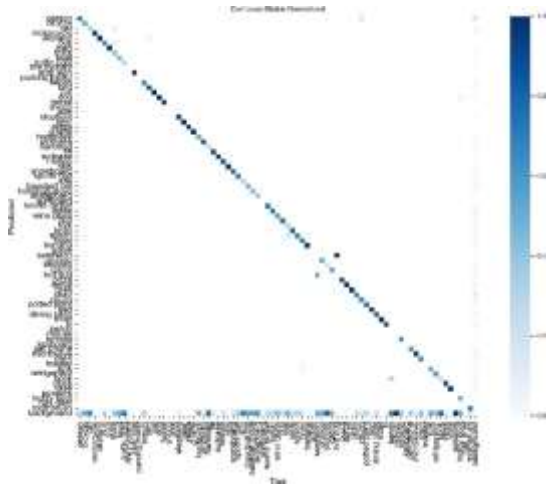

Fig.18. Confusion Matrix with Confidence Threshold = 0.001

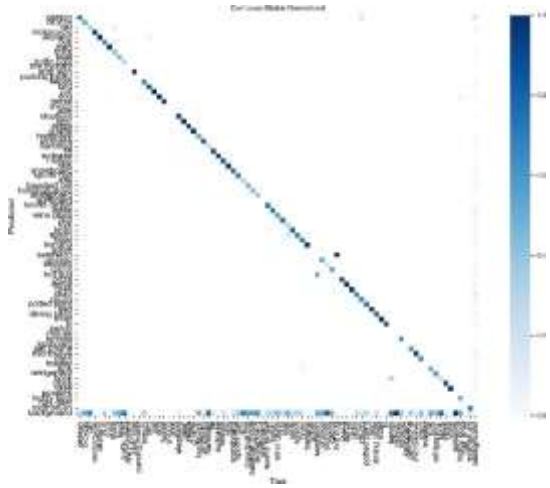

Fig.19. Confusion Matrix with Confidence Threshold = 0.250

5.4.1 F1 Confidence Curve:

The F1 score serves as a comprehensive metric for assessing a model's accuracy, taking into account both precision and recall. Specifically, it represents the harmonic mean of precision and recall, thereby providing a balanced evaluation of the model's performance across different classes.

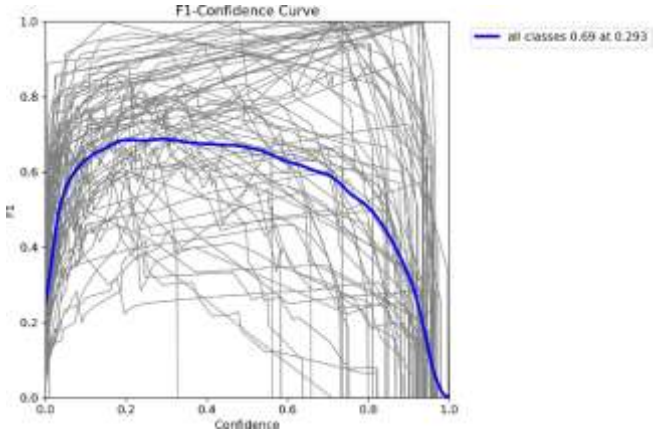

Fig.20. F1 Curve with Confidence Threshold = 0.001

The F1 Confidence Curve in Fig.20 and in Fig.21 offers a graphical representation of the F1 score across various confidence thresholds. As the F1 score reflects the model's overall effectiveness, a higher F1 score denotes superior performance. The given image shows that the F1 score settles close to 0.7 making the model fairly accurate while not being overfitted.

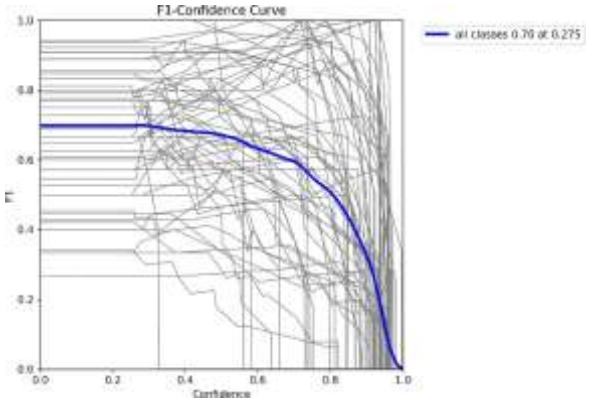

Fig.21. F1 Curve with Confidence Threshold = 0.250

5.4.2 Precision-Recall Curve:

The Precision-Recall Curve depicted in Fig.22 and in Fig.23 illustrates the balance between precision and recall across various threshold values in a classification model.

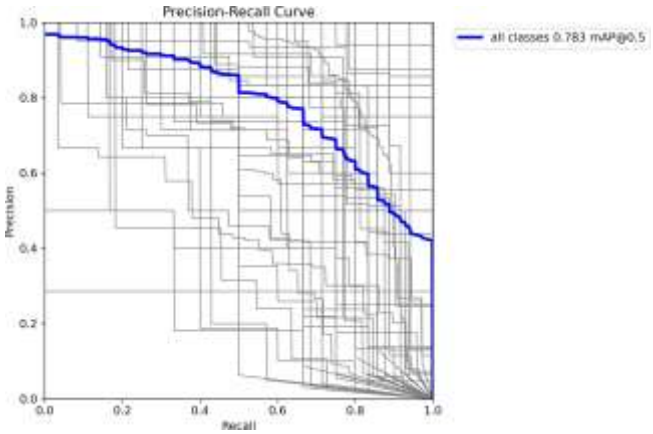

Fig.22. PR Curve with Confidence Threshold = 0.001

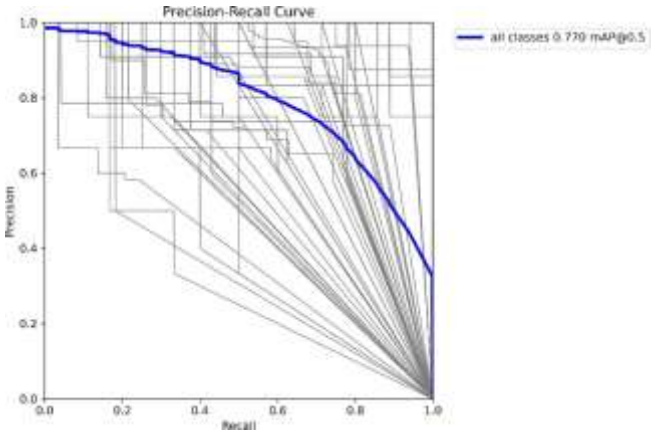

Fig.23. PR Curve with Confidence Threshold = 0.250

Precision, the ratio of true positive predictions to the total positive predictions, measures the model's accuracy in identifying positive instances. Recall (or sensitivity), the ratio of true positive predictions to the total actual positive instances, indicates the model's ability to capture all positive instances. This curve depicts how adjusting the classification threshold impacts the trade-off between precision and recall, aiding in the assessment of model performance.

5.5 COMPARISON OF IMPATIENT USERS SHORT SUMMARY GENERATION TIMES

The comparison between the short summary generation times of two different approaches, one utilizing only ResNeXt and LSTM, and the other incorporating YOLO, ResNeXt, and LSTM, reveals notable differences as shown in Fig.24

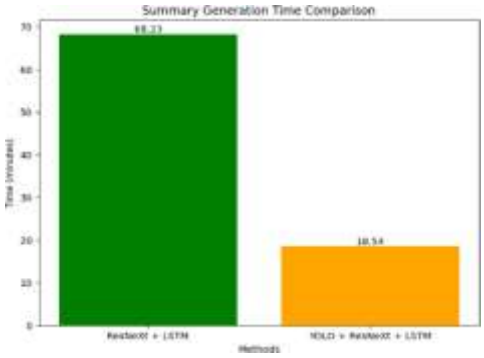

Fig.24. Comparison of Short Summary Generation Times

In the approach using just ResNeXt and LSTM, the time needed for video summary creation is considerably longer. This is because a much larger number of frames, totaling 3187, must be processed. However, when YOLO is incorporated alongside ResNeXt and LSTM, fewer frames need analysis only 1291. Consequently, the combined YOLO, ResNeXt and LSTM method takes less time to generate summaries. YOLO's object detection capabilities allow selection of fewer relevant frames for summarization. So utilizing YOLO with ResNeXt and LSTM results in faster video summarization than employing ResNeXt and LSTM alone.

5.6 COMPARISON OF MSE, MAE, RMSE VALUES

The performance of different models inside the proposed system is compared in the following graphs using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Squared Error (MSE). MSE measures the average squared difference between the actual and predicted values in a regression problem. It provides a measure of the overall model accuracy, with lower values indicating better performance.

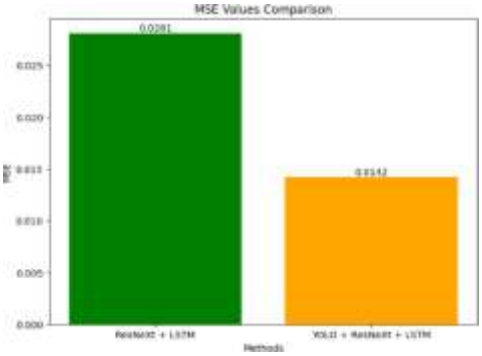

Fig.25. Comparison of MSE Values

MAE calculates the average absolute difference between the actual and predicted values. It offers a straightforward interpretation of prediction errors, where smaller values signify higher accuracy.

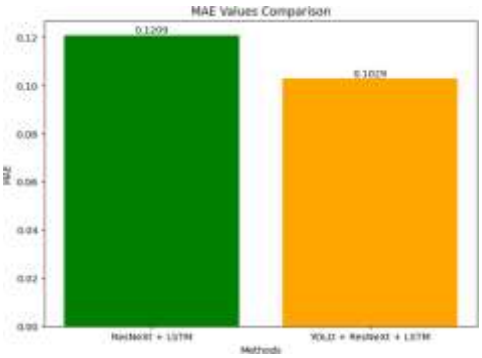

Fig.26. Comparison of MAE Values

RMSE is the square root of the MSE and provides a measure of the standard deviation of the prediction errors. It is commonly used to evaluate the spread of errors around the regression line, with lower values indicating better model performance.

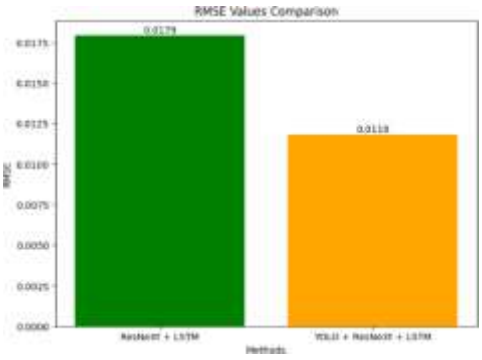

Fig.27. Comparison of RMSE Values

The graphs clearly show that our proposed model incorporating YOLO with LSTM and ResNeXt performs better than the other model using only LSTM and ResNeXt, as seen by its lower values for each of these criteria.

6. CONCLUSION

This paper provides an effective video summarization technique that caters to user's preferences. This addresses the need for managing large amounts of video content. Advanced techniques are used for Object of Interest detection, localization, and summarization. Results indicate the capability of the model to identify and catalog the OoIs. It creates a summary assembling all frames that contain the OoI for the patient user and a further summarized concise video for the impatient user. Additionally, it is observed that our proposed framework integrating YOLO with ResNext and LSTM is much faster in terms of summary generation time. This integration mitigates information overload and enhances accessibility to critical information in security, entertainment, and personal documentation domains. While this research deals with video summarization with respect to the content present in terms of scenes in images, it does not consider the audio. Potential future work could include the summary of the audio present in the video. Including audio in the video summary would be useful for an even wider range of applications.

REFERENCES

[1] H. Deshpande, A. Singh and H. Herunde, "Comparative Analysis on YOLO Object Detection with OpenCV", *International Journal of Research in Industrial Engineering*, Vol. 9, No. 1, pp. 46-64, 2020.

[2] S. Jain, S. Dash and R. Deorari, "Object Detection using Coco Dataset", *Proceedings of International Conference on Cyber Resilience*, pp. 1-4, 2022.

[3] Karakotsios, Kenneth Mark, Deon Poncini and Matthew Alan Townsend, "User Input-based Video Summarization", U.S. Patent, No. 10, 2020.

[4] J. Lin, S.H. Zhong and A. Fares, "Deep Hierarchical LSTM Networks with Attention for Video Summarization", *Computers and Electrical Engineering*, Vol. 97, pp. 1-9, 2022.

[5] A. Negi, K. Kumar and P. Saini, "Object of Interest and Unsupervised Learning-based Framework for an Effective Video Summarization using Deep Learning", *IETE Journal of Research*, pp. 1-12, 2023.

[6] A. Negi, K. Kumar, P. Saini and S. Kashid, "Object Detection based Approach for an Efficient Video Summarization with System Statistics Over Cloud", *Proceedings of International Conference on Electrical, Electronics and Computer Engineering*, pp. 1-6, 2022.

[7] Panagiotakis Costas, Harris Papadakis and Paraskevi Fragopoulou, "Personalized Video Summarization based Exclusively on user Preferences", *Proceedings of International Conference on Information Retrieval*, pp. 305-311, 2020.

[8] P. Saini, K. Kumar, S. Kashid, A. Saini and A. Negi, "Video Summarization using Deep Learning Techniques: A Detailed Analysis and Investigation", *Artificial Intelligence Review*, Vol. 56, pp. 12347-12385, 2023.

[9] L. Tan, T. Huangfu, L. Wu and W. Chen, "Comparison of RetinaNet, SSD and YOLO V3 for Real-Time Pill Identification", *BMC Medical Informatics and Decision Making*, pp. 1-11, 2021.

[10] Ul Haq and Hafiz Burhan, "An Effective Video Summarization Frame-Work based on the Object of Interest using Deep Learning", *Mathematical Problems in Engineering*, Vol. 2022, pp. 1-25, 2022.

[11] Wang Ziyan, "Video Summarization Generation with Self-Attention and Random Forest Regression", *Proceedings of International Symposium on Computer Applications and Information Systems*, Vol. 12721, pp. 349-356, 2023.