AI Powered Video Digest

Smart Online Video Summarization
using Deep Learning and Generative AI

Sparsh Jaggi, Intern, Tata Steel Limited

Anisha Dutta, Project Mentor, Analyst, TSM
Moromee Das, Project Supervisor, Head, TSM

Internship Registration No. : VT20241264

Abstract

This project presents an approach for generating context-rich summaries of video content using a combination of generative AI and computer vision techniques. Our method utilizes a computer vision-based model to extract key visual information from the video frames, providing a detailed understanding of the scene and its objects. This information is then integrated into a generative AI model, enabling the creation of summaries that capture the essence of the video while incorporating the rich context derived from the visual elements. By seamlessly combining computer vision and large language models, our system generates summaries that are not only informative but also accurately reflect the dynamic aspects of the video content. The proposed method offers a promising solution for automated video summarization, enhancing information retrieval and comprehension within the rapidly growing landscape of video data.

Keywords. *Smart Video Summarization, Video Extraction, Deep Learning, Computer Vision, YOLO Object Detection, Generative AI, Gemini LLM, Google Colab.*

Contents.

1. Introduction

The ability to automatically generate concise and informative summaries of video content is increasingly crucial in our data-driven world. With the explosion of video data online, manual summarization is becoming an unsustainable approach. This project explores the potential of leveraging generative AI models to create context-rich summaries from video content, integrating insights from a computer vision tool for enhanced accuracy.

Our approach utilizes a two-pronged strategy: 1) extracting key visual information from the video using YOLO, a robust object detection algorithm, and 2) feeding this information into a generative AI model to produce a comprehensive summary that captures the essence of the video. This method offers several advantages over traditional video summarization techniques:

- Automatic extraction of key visual elements: YOLO's efficient object detection capabilities eliminate the need for manual annotation, allowing for automated and scalable processing of video data.
- Contextual understanding through generative AI: By incorporating the visual information from YOLO into the generative model, we aim to produce summaries that are richer in context and more informative than summaries solely based on textual analysis.
- Enhanced accuracy and relevance: Combining computer vision and generative AI allows for a more comprehensive understanding of the video content, leading to summaries that are more accurate and relevant to the user.

This project contributes to the ongoing research in video summarization by exploring the potential of integrating computer vision and generative AI for more robust and insightful video content analysis. The results of this project have the potential to revolutionize how we consume and understand video content, enabling faster access to information and deeper comprehension of complex narratives.

2. Project Objective

To integrate the visual information extracted through computer vision with the textual data derived from the generative AI model to create summaries that encapsulate both visual and textual aspects of a web-based video, downloaded through application-based URL extraction.

3. Key Learnings during the Project

This project explored the potential of automated video summarization by integrating a suite of advanced technologies. We began by extracting video content from YouTube using Pytube, providing a diverse dataset for analysis. Leveraging deep learning and computer vision, we employed the YOLO algorithm to meticulously identify and segment key objects and scenes within these videos. To generate contextually rich summaries, we harnessed the power of generative AI, specifically the Gemini LLM, which enabled the model to comprehend and synthesize the video content. The entire process was seamlessly managed on Google Colab, providing a robust platform for development and testing. This project demonstrated the efficacy of combining deep learning, computer vision, and generative AI to produce accurate and insightful automated video summaries.

1. Computer Vision – OpenCV

OpenCV (Open-Source Computer Vision Library) stands as a cornerstone in the realm of computer vision and machine learning. This robust, open-source library provides a comprehensive toolkit enabling developers to tackle a wide array of tasks, from image and video analysis to object detection, facial recognition, and real-time processing. Its versatility is further amplified by its support for multiple programming languages, including Python, C++, and Java, making it accessible to a broad range of developers and researchers.

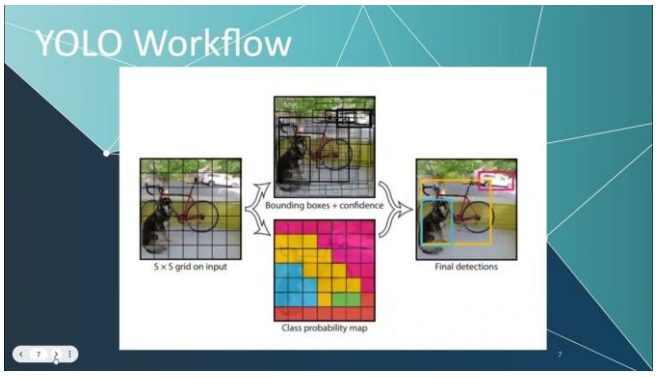
My journey exploring OpenCV has encompassed a diverse set of fundamental functionalities and techniques, including:

1. Image and Video Manipulation:
    - Reading and Writing Images and Videos: Extracting data from image and video files, and saving processed results.
    - Resizing and Rescaling: Adjusting image and video dimensions for efficient processing and display.
    - Essential Functions:
        - Grayscale Conversion: Transforming images from color to monochrome for simpler analysis.
        - Blurring: Smoothing images to reduce noise and enhance certain features.
        - Edge Detection: Identifying boundaries and sharp transitions within images.
        - Dilation and Erosion: Modifying image shapes for more precise feature extraction.
2. Image Transformations:
    - Translation: Shifting image content horizontally or vertically.
    - Rotation: Rotating images by specific angles.
    - Flipping: Mirroring images horizontally or vertically.
    - Cropping: Extracting specific regions of interest from images.
3. Color Spaces:
    - Conversion between BGR, Grayscale, and RGB: Manipulating color representations for different applications.
4. Image Operations:
    - BITWISE Operations: Applying logical operations to images, such as AND, OR, XOR, to modify pixels based on their values.
    - Histogram Computation: Analyzing image pixel distribution for feature extraction and color analysis.
5. Advanced Applications:
    - Face Detection: Utilizing Haar Cascades or built-in recognizers for detecting faces in images and videos.

OpenCV's vast capabilities, coupled with its ease of use and comprehensive documentation, make it an invaluable resource for developers and researchers venturing into the exciting world of computer vision.

2. Object Detection – Yolo

Object detection, the task of identifying and classifying objects within an image or video, has been revolutionized by the emergence of the YOLO (You Only Look Once) algorithm. Unlike traditional methods that often involve multiple processing steps, YOLO tackles this challenge with a unique single-pass approach. It leverages a neural network to simultaneously predict bounding boxes and probabilities for objects within a gridded representation of the input image. This streamlined architecture allows YOLO to achieve remarkable speed and accuracy, making it a prime choice for real-time applications.



YOLO's efficiency stems from its innovative design. By dividing the input image into a grid and processing it in a single neural network pass, it avoids the computationally expensive iterative steps of traditional methods.

This enables YOLO to process data significantly faster, achieving real-time performance vital for applications such as:

1. Surveillance: Real-time monitoring and object identification for security purposes.
2. Autonomous Driving: Accurate object detection for safe navigation and decision-making in self-driving vehicles.
3. Computer Vision Tasks: A wide range of applications requiring rapid object recognition and analysis, from robotics to medical imaging.

The combination of high accuracy and real-time capabilities positions YOLO as a transformative force in the field of object detection, enabling the development of more sophisticated and efficient solutions for a multitude of applications.

3. Web Video Download – PyTube

In the world of online video content, YouTube reigns supreme, offering a vast library of videos spanning countless topics. However, accessing this content offline or for further processing often requires downloading, a process that can be cumbersome and time-consuming. Enter PyTube, a Python library designed to simplify this task.

PyTube provides a user-friendly interface for effortlessly downloading videos from YouTube. It empowers developers to retrieve video streams, select different video resolutions, and even extract audio tracks – all with a few lines of code. This ease of use extends to automation, enabling users to download individual videos or entire playlists with a single script. PyTube's value extends beyond mere downloading. Its seamless integration into Python scripts allows developers to incorporate YouTube video extraction into larger data pipelines and projects. This opens up a world of possibilities for applications that require online video content for further processing, including:

1. Video Summarization: Generating concise summaries of videos for efficient information retrieval.
2. Video Analysis: Extracting data and insights from video content, such as sentiment analysis or object tracking.
3. Offline Viewing: Creating personal libraries of videos for convenient access without an internet connection.

With its simple interface, powerful features, and seamless integration with Python, PyTube has become an invaluable tool for developers and researchers working with online video content. It streamlines the process of downloading YouTube videos, enabling a wide range of applications and unlocking new opportunities for video analysis and manipulation.

To download a video from Youtube :

Importing library : `from pytube import Youtube`

4. Video Data Handling – OpenCV

OpenCV (Open Source Computer Vision Library) is a comprehensive toolkit for handling and processing video data, widely utilized in computer vision applications. It provides a variety of functionalities essential for video data manipulation, including reading, writing, and processing video files.

1. **Extracting Frames**: OpenCV allows for frame-by-frame extraction from video streams using the cv2.VideoCapture function. This capability is crucial for tasks that require analysis of individual frames, such as object detection, motion tracking, and video summarization.
2. **Resizing Frames**: The library offers the cv2.resize function, which is used to scale frames to different dimensions. Resizing is important for normalizing input data for machine learning models and for optimizing processing speed and memory usage.
3. **Video Writing**: OpenCV can write processed frames back to a video file using the cv2.VideoWriter function. This is useful for creating new video content after applying various transformations or analyses to the frames.
4. **Real-time Video Processing**: OpenCV supports real-time video capture from cameras and other devices. This enables live video processing applications, such as surveillance, augmented reality, and interactive systems.
5. **Image Transformations and Filtering**: OpenCV includes a wide range of image processing functions, such as blurring, sharpening, edge detection, and color space transformations. These tools are essential for pre-processing video data before analysis.
6. **Integration with Machine Learning**: OpenCV can work alongside machine learning frameworks like TensorFlow and PyTorch, facilitating the implementation of deep learning models for tasks like object detection, face recognition, and scene segmentation.

OpenCV's versatility and efficiency make it an indispensable library for video data handling, supporting a broad spectrum of applications from simple frame extraction to complex real-time video analysis

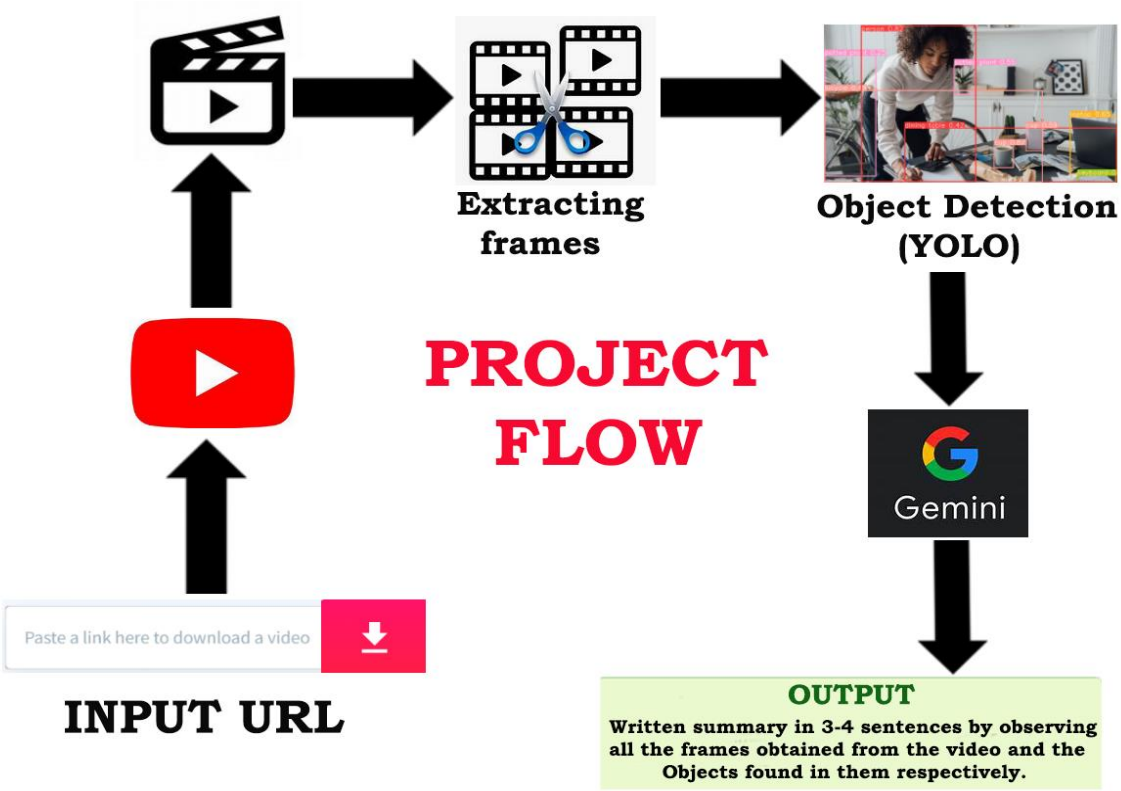
5. Open Source LLM – Gemini

Gemini is an open-source Large Language Model (LLM) known for its advanced capabilities in natural language understanding and generation. As part of the growing ecosystem of open-source AI tools, Gemini stands out for its ability to handle a wide range of language processing tasks, including text summarization, translation, question answering, and more.

Key features of Gemini include:

1. **Advanced NLP Capabilities**: Gemini excels in natural language processing, allowing it to understand context, generate coherent and contextually relevant text, and perform tasks that require a deep understanding of language.
2. **Scalability**: Designed to handle large datasets, Gemini can process substantial amounts of text data, making it suitable for applications that require high throughput and robust performance.
3. **Customization and Fine-Tuning**: Users can fine-tune Gemini on specific datasets to tailor its performance to particular domains or tasks. This adaptability ensures that the model can be optimized for a wide range of applications.
4. **Integration with Other Tools**: Gemini is designed to integrate seamlessly with various software environments and tools, such as Python and machine learning frameworks like TensorFlow and PyTorch. This makes it easy to incorporate into existing workflows and pipelines.
5. **Open Source**: As an open-source project, Gemini benefits from community contributions and transparency. Users can inspect the model's architecture, contribute to its development, and modify it to suit their needs.
6. **Accessibility**: Being open-source, Gemini is accessible to researchers, developers, and organizations without the need for costly licenses, promoting innovation and experimentation across various fields.

4. Project Flow



1. Extracting Video from Web URL
   The first stage of the project starts with entering URL of any Youtube Video , by which the Video gets downloaded in the Local Storage with the help of Pytube.

2. Extracting Frames from Video
   After the Video is downloaded in the Local Storage , all the frames of the Video are extracted by using VideoCapture and stored in a folder .

3. Framewise Object Detection
   When all the frames of the Video are extracted , each of them are passed through YOLO for detecting objects in the image/frame.

4. Querying LLM to form textual response
   After detecting the objects from all the frames , the object string is used as INPUT and passed through Gemini LLM which gives out a summary of the Video taking Objects detected as a reference.

5. Implementation

INSTALL Required Libraries :

```
!pip install pytube
!pip install ultralytics
!pip install google-generativeai
```

Enter URL to Download Youtube Video :

```
import cv2
from pytube import YouTube
url = input("Enter your URL here :")
link = YouTube(url)
video=link.streams.get_highest_resolution()
title=video.title
video.download()
print (title)
```

Extracting Frames and detecting Objects in each Frame :

```
import cv2

import os
import pandas as pd
import numpy
from pytube import YouTube
from ultralytics import YOLO
model = YOLO("yolov8n.pt", "v8")
df = pd.DataFrame(columns = ['Frame no.','column1','coloumn2',……,'column80'])
varr=['a','b',……..(80 variables)]
for ini in range(len(varr)):
  varr[ini]=0
arr=[*array of objects detected by YOLO*]
vid=cv2.VideoCapture(*Path of the Video*)    # Insert Your path to Local Video here
currentframe=0
if not os.path.exists('data'):
    os.makedirs('data')

while(True):
  success, frame= vid.read()
  if not success:
    break
  cv2.imwrite('/content/data/frame'+str(currentframe)+'.jpg',frame)
  currentframe+=1
  if cv2.waitKey(1) & 0xFF==ord('q'):
    break
vid.release()
cv2.destroyAllWindows()
```

```
loop=currentframe-1
frame=0
while(loop>=0):
  results = model("/content/data/frame"+str(loop)+".jpg")
  for box in results[0].boxes:
    #x1, y1, x2, y2 = [round(x) for x in box.xyxy[0].tolist()]
    class_id = box.cls[0].item()
    print(class_id)
    for cl in range(len(arr)):
      if class_id==cl:
        varr[cl]+=1
  df.loc[len(df)] =
[frame,varr[0],varr[1],varr[2],varr[3],varr[4],varr[5],varr[6],varr[7],varr[8],varr[9],varr[10],varr[11],varr[12],……
…(Values of the column)…….varr[79]]
  frame+=1
  for ini1 in range(len(varr)):
    varr[ini1]=0
  loop-=30
```

Getting a String of Detected Objects in each frame :

```
objectString = ""
cols = df.columns[1:]
for i in range(int(len(df))):
  objectString = objectString+'{frame number = '+str(i)
  for colname in cols:
    objectString = objectString+" "+str(colname)+"= "+ str(df.T[i][colname])+", "
  objectString = objectString+'},'
print(objectString)
```

Gemini Response :

```
import google.generativeai as genai
def getGeminiResposne(objectsFound):
    inputPrompt = "*INPUT PROMPT* \n\n Objects Detected : {}".format(objectsFound)
    genai.configure(api_key='')
    model = genai.GenerativeModel('gemini-1.5-flash')
    rawResponse = model.generate_content(inputPrompt)
    response = rawResponse.text.replace('\n', '').replace('*', '')
    return response
```

Final OUTPUT :

```
getGeminiResposne(objectString)
```

SAMPLE OUTPUT :

```
'The video shows a scene with people, a television, and laptops. The number of people and laptops varies acı
imultaneously visible. This suggests a scene involving multiple individuals potentially working or interact:
```

6. Challenges and Solutions

Challenge	Solution
1. Video Downloading Issues	Utilized Pytube for reliable and efficient downloading of videos from YouTube.
2. Frame Extraction Performance	Optimized OpenCV code to efficiently extract frames without losing synchronization with audio.
3. Object Detection Accuracy	Implemented YOLO (You Only Look Once) for high-accuracy real-time object detection.
4. High Computational Requirements	Used Google Colab for access to powerful GPUs, ensuring smooth processing and training.
5. Generating Coherent Textual Summaries	Integrated Gemini LLM to generate accurate and contextually relevant summaries from visual data.
6. Diverse Facial Analysis	Deployed DeepFace for comprehensive facial analysis including age, gender, emotion, and race detection.
7. Integration of Multiple Technologies	Carefully coordinated the workflow to ensure seamless integration of Pytube, OpenCV, YOLO, Gemini, and DeepFace.
8. Real-time Processing	Optimized pipeline components for speed and efficiency to handle real-time video processing.
9. Handling Variability in Video Quality and Content	Applied robust pre-processing steps and used adaptable models like YOLO and DeepFace.
10. Scalability of the System	Designed the system to be scalable and easily deployable in cloud environments like Google Colab.

7. Additional Learning

1. Image Augmentation – Keras

KerasCV is an extension of Keras specifically designed to handle computer vision tasks, providing advanced tools and functionalities for image augmentation. Image augmentation in KerasCV involves generating modified versions of images to expand and diversify the training dataset, thereby enhancing the model's ability to generalize and perform better on unseen data.

Key Augmentation Techniques in KerasCV:

1. **Rotation**: Randomly rotating images by a specified degree range to help the model recognize objects from different angles.

2. **Translation**: Shifting images horizontally or vertically to make the model robust against positional changes.
3. **Shearing**: Applying affine transformations that slant the shape of images, helping the model learn invariant features.
4. **Zooming**: Randomly zooming into images to train the model on various scales and perspectives.
5. **Flipping**: Horizontally or vertically flipping images to augment the dataset with mirrored versions.
6. **Brightness Adjustment**: Randomly changing the brightness levels to make the model robust against lighting variations.
7. **Contrast Adjustment**: Modifying the contrast of images to handle different lighting conditions.
8. **Saturation and Hue Adjustments**: Altering color saturation and hue to improve the model's color invariance.
9. **Gaussian Noise**: Adding random noise to images to simulate real-world conditions and reduce overfitting.

Implementation:

KerasCV integrates seamlessly with the Keras API, using the keras.preprocessing.image.ImageDataGenerator class or custom layers for more advanced operations.

2. Face Detection – OpenCV

OpenCV (Open Source Computer Vision Library) is widely used for face detection tasks due to its robustness and ease of use. It provides pre-trained models and a variety of tools for detecting faces in images and videos.

Key Steps in Face Detection with OpenCV:

1. **Load OpenCV Library**: Import the necessary OpenCV library functions.
2. **Load Pre-trained Face Detection Model**: OpenCV provides pre-trained models based on Haar Cascades or deep learning-based models such as Single Shot Multibox Detector (SSD).
3. **Read Image or Video**: Capture or read the image or video frame where face detection needs to be performed.
4. **Convert Image to Grayscale**: Convert the image to grayscale to simplify processing, as face detection algorithms typically perform better on single-channel images.
5. **Detect Faces**: Apply the face detection algorithm to the grayscale image to identify the locations of faces.
6. **Draw Bounding Boxes**: Draw rectangles or bounding boxes around the detected faces for visualization.

Example Using Haar Cascades:

Haar Cascades is a popular method for object detection, including face detection, provided by OpenCV.

```
import cv2 as cv
imgg = cv.imread("person1.jpg", cv.IMREAD_COLOR)
img=cv.resize(imgg,(640,640))
gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
cv.imshow('Gray',gray)
haar_cascade= cv.CascadeClassifier('haar_face.xml')
```

```
faces_rect=haar_cascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=15)
print(f'Number of faces found = {len(faces_rect)}')
for(x,y,w,h) in faces_rect :
   cv.rectangle(img,(x,y),(x+w,y+h),(0,255,0),thickness=2)
cv.imshow('Detected Faces',img)
```

3. Facial Analysis Detection -Deepface

DeepFace is an advanced facial recognition and analysis library for Python that leverages state-of-the-art deep learning models. It provides a range of functionalities for facial analysis, including age and gender prediction, emotion detection, and race recognition.

Key Steps in Facial Analysis with DeepFace:

1. **Install DeepFace**: Ensure you have the DeepFace library installed in your environment.

   ```
   pip install deepface
   ```

2. **Import the Library**: Import DeepFace along with other necessary libraries such as OpenCV for image handling.
3. **Read and Preprocess the Image**: Load the image containing the face to be analyzed.
4. **Perform Facial Analysis**: Use DeepFace to analyze the face, obtaining various attributes like emotion, age, gender, and race.
5. **Display Results**: Visualize the analysis results, such as drawing bounding boxes around detected faces and displaying predicted attributes.

Example Code:

```
import cv2
from deepface import DeepFace

# Load an image using OpenCV
```

```
image_path = 'path_to_image.jpg'
image = cv2.imread(image_path)

# Perform facial analysis using DeepFace
analysis = DeepFace.analyze(image, actions=['age', 'gender', 'emotion', 'race'])

# Display the results
for face in analysis['instances']:
    x, y, w, h = face['region']['x'], face['region']['y'], face['region']['w'], face['region']['h']
    age = face['age']
    gender = face['gender']
    emotion = face['dominant_emotion']
    race = face['dominant_race']

    # Draw a bounding box around the face
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Annotate the image with the analysis results
    cv2.putText(image, f'Age: {age}', (x, y - 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
    cv2.putText(image, f'Gender: {gender}', (x, y - 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
    cv2.putText(image, f'Emotion: {emotion}', (x, y + h + 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0),
2)
    cv2.putText(image, f'Race: {race}', (x, y + h + 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

# Display the output image
cv2.imshow('Facial Analysis', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Explanation:

1. **Install and Import DeepFace**: The DeepFace library is installed and imported along with OpenCV.
2. **Load an Image**: An image is loaded using OpenCV.
3. **Perform Analysis**: The DeepFace.analyze function is used to analyze the image, specifying actions like age, gender, emotion, and race detection.
4. **Display Results**: The results are displayed by drawing bounding boxes around faces and annotating the image with the predicted attributes.

Features of DeepFace:

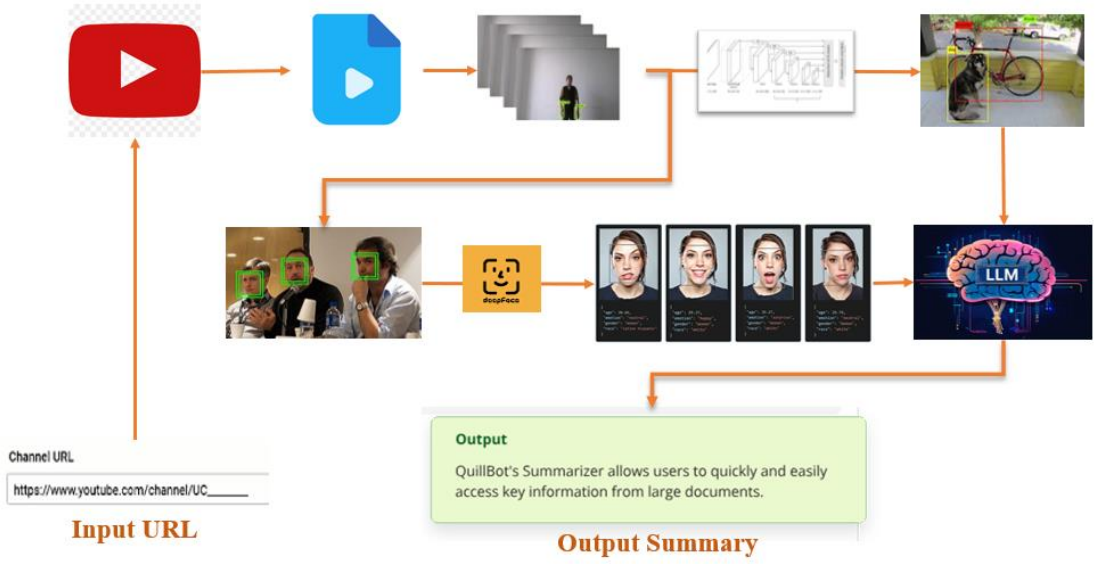
1. **Age Prediction**: Predicts the age of the detected face.
2. **Gender Detection**: Determines the gender of the detected face.
3. **Emotion Detection**: Identifies the dominant emotion (e.g., happy, sad, angry).
4. **Race Prediction**: Predicts the race of the detected face.

DeepFace provides a robust and comprehensive toolset for facial analysis, making it a valuable asset for projects involving facial recognition and analysis.

6. Further Directions

- Facial Expression Identification :
    - Go Beyond Faces: Combine facial expressions with other clues like audio, text, and speech analysis for a more accurate understanding of emotions.
    - Use Smarter Models: Explore specialized deep learning models built for recognizing facial expressions to improve accuracy and detail.
    - See in Real-Time: Implement real-time facial expression recognition to dynamically update the video digest as it plays, like in live streaming.
    - Understand the Context: Analyse expressions in relation to the video's content, like dialogue, scene, and speaker, to avoid misinterpretations.
    - Capture Intensity: Add intensity levels to each expression, showing how strong the emotion is, to provide a more nuanced understanding.
- Sentiment Analysis of video :
    - Listen and Watch: Combine audio and video analysis for a deeper understanding of sentiment, considering tone of voice, speech patterns, and facial expressions.
    - Track Emotion Over Time: Analyse how sentiment evolves throughout the video to reveal shifts in emotions or attitudes.
    - Understand the Context: Analyse sentiment in relation to the video's context, like humour or sarcasm, to avoid misinterpretations.
- Customised Object Detection Model preparation :
    - Keras augmentation techniques can be used to create huge sized dataset from smaller set of images, which will further be used to train a customized model.
    - Customized model would help to extract precise classes of objects the user wants instead of the usual 80 classes, which can significantly increase the accuracy.

Appending these blocks to the earlier flow, the updated flow would look like the following. Here, in addition to explaining the objects found in each frame of the video, we will also be able to extract the facial expressions of the persons detected in the frames of the video, this will provide addition information to the final output.



9 .Conclusion

This project successfully demonstrated the power of integrating multiple advanced technologies to create a comprehensive pipeline for video summarization and facial analysis. The workflow, designed for efficiency and accuracy, encompasses the following key steps:

1. *Video Acquisition*: Utilizing Pytube, we seamlessly extracted videos from YouTube, providing the foundation for further processing.
2. *Frame Extraction*: OpenCV, a versatile computer vision library, enabled us to extract individual frames from the videos, preparing the data for subsequent analysis.
3. *Object Detection*: The state-of-the-art YOLO algorithm was employed to accurately identify and segment key objects and scenes within each frame, offering a detailed visual understanding of the content.
4. *Generative AI Summarization*: The Gemini Large Language Model (LLM) was leveraged to generate coherent and contextually rich summaries based on the detected objects and scenes, demonstrating the power of generative AI in comprehending and summarizing visual content.
5. *Facial Analysis*: DeepFace, a powerful facial analysis tool, was utilized to extract detailed insights from detected faces, including age, gender, emotion, and race, providing valuable information for various applications.

Key Learnings and Outcomes:

- *Synergy of Technologies*: This project highlighted the effectiveness of integrating various AI and machine learning tools, showcasing their ability to work collaboratively to solve complex tasks.
- *Advanced Video Summarization*: By seamlessly combining object detection with generative AI, we developed a robust system capable of efficiently summarizing video content, paving the way for applications like video indexing, content creation, and security.
- *In-depth Facial Analysis*: The utilization of DeepFace enabled us to extract meaningful information from faces, opening doors for applications ranging from demographic studies to user profiling and surveillance.
- *Real-Time Processing*: The use of tools like OpenCV and YOLO ensured that the system could handle real-time video data, making it suitable for live applications.
- *Scalability and Flexibility*: Implementing the project on Google Colab provided a scalable environment for development and testing, demonstrating its potential for deployment in production environments.

Overall Impact: This project underscores the transformative potential of combining deep learning, computer vision, and natural language processing to automate and enhance the analysis and summarization of video content. The insights gained and the methodologies developed can be further refined and expanded for broader applications, contributing significantly to the advancement of multimedia processing and artificial intelligence.

10. Bibliography

YOLOv5: Object Detection in PyTorch

- URL: https://github.com/ultralytics/yolov5

YOLO: Real-Time Object Detection

- URL: https://pjreddie.com/darknet/yolo/

OpenCV Documentation

- URL: https://docs.opencv.org/4.x/

Keras Documentation

- URL: https://keras.io/

Gemini Documentation

- URL: Gemini for Google Cloud for Google Cloud documentation

PyTube Documentation

- URL: https://pytube.io/

Implementation on Google Colab

- URL: https://colab.research.google.com/